

UNIVERSIDAD NACIONAL DE LA PATAGONIA SAN JUAN BOSCO

**Facultad de Ingeniería
Sede Trelew**

Tesina de grado presentada para la obtención del título
Licenciado en Sistemas
(Or. Planificación, Gestión y Control de Proyectos Informáticos):

Ganadería 4.0: Acercando nuevas tecnologías a pequeños y medianos productores de la región



Alumnos

APU Allende, Lucila Abigail

APU Garibaldi, Jorge Anele Raul

Tutor

Dr. Firmenich Diego

Co-Tutor

Lic. Leonardo Morales

Agradecimientos

Agradezco a mi familia por el infinito acompañamiento en todos estos años de estudio y aprendizaje. A mis amigos de toda la vida y a los que me ha dado el transcurso de esta hermosa carrera, los cuales me han empujado a seguir adelante en todo momento. A mi gran compañera Lucila por ser un pilar más que fundamental para llevar a cabo, no solo esta tesina, sino también los últimos años de carrera compartidos y a nuestros tutores con los cuales hemos formado un gran equipo de trabajo.

Anele

Me gustaría expresar mi más sincero agradecimiento a todas las personas que contribuyeron de alguna manera en la realización de esta tesis. En primer lugar, agradezco a mi compañero Anele por estar siempre disponible para avanzar juntos en este proyecto, incluso durante las misas de los sábados por la mañana. También quiero agradecer a mi familia, amigos y compañeros por su apoyo constante y motivación, lo cual fue una fuerza impulsora para mí y por entender el tiempo que esto requería. No puedo dejar de mencionar a nuestros tutores, Leo y Diego, cuya orientación fue fundamental para guiarnos cuando nos encontramos perdidos. Su apoyo y ánimo fueron una fuente constante de motivación para culminar nuestra tesina. A todos ellos, les agradezco profundamente.

Lucila

Resumen

Las comunidades rurales de la provincia del Chubut actualmente enfrentan grandes desafíos productivos ante las condiciones climáticas adversas y las sequías cada vez más recurrentes en los últimos años. Estos problemas afectan negativamente a la producción y la rentabilidad de los campos, en particular a la producción de carne y lana, llevando en algunos casos, al cierre de los establecimientos productivos o incluso al abandono de los mismos. Para abordar este desafío, es necesario apoyar el desarrollo rural a través de la entrega de información de calidad y conocimientos en prácticas agrícolas sustentables, aprendizaje en la gestión productiva y adopción de tecnología, eliminando así, las barreras culturales y de falta de capacitación.

En esta tesina, se investigará el marco de referencia actual en la industria ganadera provincial, incluyendo conceptos de relevancia como: Industria 4.0, Ganadería de Precisión, Inteligencia Artificial enfocada en los ámbitos productivos y la disponibilidad de herramientas de gestión existentes en el mercado, haciendo principal hincapié en la oportunidad de incorporación de la tecnología como herramienta central en la mejora productiva y el desarrollo ganadero sustentable.

También se detallará la investigación llevada a cabo para desarrollar un sistema de soporte a la toma de decisiones (DSS) específico para la gestión productiva ovina en los campos de la zona patagónica desde las investigaciones iniciales hasta el desarrollo final del proyecto. Este sistema busca mejorar la toma de decisiones de los productores combinando información de producción y datos meteorológicos obtenidos a través de sistemas externos de datos y otras fuentes de información. Además, se ahondará en el detalle de los algoritmos de inteligencia artificial implementados, que amplían el conjunto de funcionalidades ofrecidas dentro del DSS.

Finalmente, se presentarán las conclusiones y recomendaciones derivadas de este trabajo, con el fin de ofrecer una guía útil para aquellos interesados en profundizar estos temas o en aplicarlos en trabajos futuros.

Organización de los Contenidos

Capítulo 1: Introducción

En este capítulo se introduce el tema principal, la motivación y análisis contextual, propuesta de desarrollo, y objetivos en el estudio. Aquí se presenta una visión general del trabajo y se establecen las bases para su desarrollo.

Capítulo 2: Marco Teórico

En este capítulo se presenta una visión exhaustiva de los fundamentos teóricos en los cuales se basa el desarrollo propuesto, como Industria 4.0, Inteligencia Artificial, Ganadería en Chubut, Procesamiento de Lenguaje Natural, entre otros.

Capítulo 3: Estado del Arte

En este capítulo se presentan las investigaciones realizadas a nivel global, nacional y provincial sobre el concepto de Ganadería 4.0, así como las aplicaciones específicas que se han desarrollado para su implementación en el sector ganadero. También se discuten los desafíos y obstáculos que enfrentan los ganaderos en la actualidad en relación a la adopción de esta nueva tecnología.

Capítulo 4: Datos y modelos

En este capítulo se detalla el proceso de obtención y limpieza de los datos provenientes de fuentes públicas, así como también se describen los análisis estadísticos y visuales llevados a cabo. Además, se lleva a cabo el entrenamiento y la comparación de modelos para la predicción de kilos de lana, corderos, finura y rinde en una aplicación web.

Capítulo 5: Desarrollo Realizado

En este capítulo se realiza una introducción a las herramientas y tecnologías utilizadas en la realización del proyecto. Además, se presentan las principales características de la aplicación desarrollada, incluyendo detalles exhaustivos sobre su funcionamiento y diseño.

Capítulo 6: Conclusiones y Trabajos Futuros

En este capítulo final, se presenta un resumen del trabajo realizado contemplando los objetivos planteados al inicio del proyecto. Además, se exponen algunas posibles líneas de investigación futuras, con el objetivo de continuar profundizando en el tema propuesto.

Glosario

AFIP (Administración Federal de Ingresos Públicos)

AIS (Sistemas de identificación automática del inglés Automatic Identification System)

AJAX (Asynchronous JavaScript and XML)

AMR (Valores de resistencia a los antimicrobianos del inglés Antimicrobial Resistance)

APSIM (Simulador de sistemas de producción agrícola del inglés Agricultural Production Systems Simulator)

BA (Bienestar Animal)

BCS (Medición de la condición corporal del inglés Body Condition Score)

BGP (Buenas Prácticas Ganaderas)

BLUP (Valores de predicción lineal sin sesgos óptimos del inglés Best Linear Unbiased Prediction)

BW (Peso Corporal del inglés Body Weight)

CART (Árboles de Clasificación y Regresión del inglés Classification and Regression Trees)

CIT (Complejos, Individuales y variantes en el Tiempo del inglés Complexes, Individuals and Time)

CNN (Redes Neuronales Convolucionales del inglés Convolutional Neural Network)

CSS (Cascading Style Sheets)

CSV (valores separados por comas del inglés Comma Separated Values)

DBN (Deep Belief Networks)

DEP (Diferencias Esperadas en la Progenie del inglés Expected Progeny Differences)

DL (Aprendizaje Profundo del inglés Deep Learning)

DMB (Deep Belief Networks)

DOM (Document Object Model)

DRY (Don't Repeat Yourself)

DSS (sistemas de apoyo a la toma de decisiones del inglés Decision Support Systems)

EBV (Valores estimados de cría del inglés Estimated Breeding Values)

EIS (sistemas de información ejecutiva del inglés Executive Information Systems)

EV (vehículos eléctricos del inglés Electric Vehicles)

FAO (Organización de las Naciones Unidas para la Agricultura y la Alimentación del inglés Food and Agriculture Organization)

FRC (calculador de ración de alimento del inglés Feed Ration Calculator)

GBM (Mejoramiento Genético Basado en Marcadores del inglés Genomic Selection or Marker Assisted Selection)

GEI (Gases de Efecto Invernadero)

GIS (Sistema de información geográfica del inglés Geographic Information System)

GNSS (Sistema global de navegación por satélite del inglés Global Navigation Satellite System)

GPS (Sistema de Posicionamiento Global del inglés Global Position System)

HTML (HyperText Markup Language)

I4.0 (Industria 4.0)

IA (Inteligencia Artificial del inglés Artificial Intelligence)

IIoT (Internet Industrial de las Cosas del inglés Industrial Internet of Things)

INTA (Instituto Nacional de Tecnología Agropecuaria)
 IoT (Internet de las cosas del inglés Internet of Things)
 IRT (termografía infrarroja del inglés Infrared Thermography)
 KNN (k vecinos cercanos del inglés K-Nearest Neighbors)
 LDA (Análisis Discriminante Lineal del inglés Linear Discriminant Analysis)
 LMR (residuos de medicamentos veterinarios del inglés Livestock Medicinal Residues)
 LPWAN (Red de Área Amplia de Baja Potencia del inglés Low-Power Wide-Area Network)
 ML (aprendizaje automático del inglés Machine Learning)
 MVC (Modelo-Vista-Controlador del inglés Model-View-Controller)
 NDVI (índice de vegetación de diferencia normalizada del inglés Normalized Difference Vegetation Index)
 NLP (Procesamiento de Lenguaje Natural del inglés Natural Language Processing)
 OLAP (Sistema de Análisis Online del inglés Online Analytical System)
 PA (agricultura de precisión del inglés Precision Agriculture)
 PF (alimentación de precisión del inglés Precision Feeding)
 PLF (ganadería de precisión del inglés Precision Livestock Farming)
 PROLANA (Programa para el Mejoramiento de la Calidad de la Lana)
 QDA (Análisis Discriminante Cuadrático del inglés Quadratic Discriminant Analysis)
 RAM (Random Access Memory)
 RF (Random Forest)
 RFI (medición del consumo residual del inglés Residual Feed Intake)
 RFID (identificación por radiofrecuencia del inglés Radio Frequency Identification)
 RNA (Redes Neuronales Artificiales del inglés Artificial Neural Networks)
 SENASA (Servicio Nacional de Sanidad y Calidad Agroalimentaria)
 SF (agricultura inteligente del inglés Smart Farming)
 SMOTE (técnica de sobremuestreo de minorías sintéticas del inglés Synthetic Minority Over-sampling Technique)
 SVM (máquina de vectores soporte del inglés Support Vector Machine)
 TDVI (índice de temperatura y diferencia de vegetación del inglés Temperature-Difference Vegetation Index)
 TI (Tecnologías de la Información del inglés Information Technology)
 TIC (Tecnologías de la Información y la Comunicación del inglés Information and Communication Technologies)
 TSCH (saltos de canal en franjas horarias del inglés Time-Slotted Channel Hopping)
 UAV (vehículo aéreo no tripulado del inglés Unmanned Aerial Vehicle)
 UE (Unión Europea del inglés European Union)
 UG (Unidades Ganaderas)
 UWB (banda ultraancha del inglés Ultra-Wideband)
 WEF (Foro Económico Mundial del inglés World Economic Forum)
 WWW (World Wide Web)

Índice general

Agradecimientos	2
Resumen	3
Organización de los Contenidos	4
Glosario	5
Índice general	7
Índice de figuras	12
Índice de tablas	14
Capítulo 1: Introducción	15
1.1 Motivación	15
1.2 Objetivos	17
1.2.3 Objetivo general	17
1.2.3 Objetivos específicos	17
1.3 Alcance y límites	17
1.4 Resultados esperados	17
Capítulo 2: Marco teórico	18
2.1 Industria 4.0	18
2.1.1 ¿Qué es la industria 4.0?	19
2.1.2 ¿Por qué es importante la Industria 4.0?	19
2.1.3 Industria 4.0 en el Mundo	19
2.3.4 Industria 4.0 en Argentina	20
2.1.5 Industria 4.0 en Chubut	21
2.2 Ganadería 4.0	22
2.2.1 Industria 4.0	22
2.2.2 Internet de las cosas	23
2.2.3 Big Data	23
2.2.4 5G	24
2.2.5 Computación en la nube	24
2.2.6 Ganadería de Precisión	25
2.2.7 Smart Farming	27
2.3 Ganadería en Chubut	29
2.3.1 Chubut	29
2.3.2 Ganadería Ovina	30
2.3.3 Producción de Lana	30
2.3.4 Producción de Carne	33
2.4 Inteligencia Artificial	33
2.4.1 ¿Por qué inteligencia artificial?	34
2.4.2 Definición	35
2.4.3 Ramas de la Inteligencia Artificial	36

2.4.4 Áreas de la Inteligencia Artificial	37
2.5 Machine Learning	40
2.5.1 ¿Qué es el aprendizaje automático?	40
2.5.2 ¿Por qué aprendizaje automático?	41
2.5.3 Conceptos sobre datos	42
2.5.4 Conceptos sobre aprendizaje	43
2.5.5 Tipos de datos	44
2.5.6 Tipos de problemas	44
Clasificación	44
Regresión	45
Agrupamiento	45
Extracción de reglas	45
2.5.7 Tipos de soluciones	45
Modelos lineales	46
Modelos de árbol	49
Redes neuronales	51
2.5.8 Tipos de aprendizajes	51
Aprendizaje supervisado	51
Aprendizaje no supervisado	53
Aprendizaje por refuerzo	53
2.5.9 Etapas de implementación del aprendizaje máquina	54
Recopilación de datos	55
Exploración y preparación de datos	55
Entrenamiento	55
Evaluación del modelo	56
Mejora del modelo	58
2.6 Procesamiento de Lenguaje Natural	60
2.6.1 ¿Qué es el Procesamiento del Lenguaje Natural?	61
2.6.2 Definición de lenguaje	61
2.6.3 Definición de lenguaje natural	62
2.6.4 Definición de lenguaje formal	63
2.6.5 Procesamiento Computacional del Lenguaje Natural	63
2.6.6 Aplicaciones del NLP	64
2.6.7 Arquitectura de un sistema de NLP	64
2.6.8 Corpus lingüístico	66
2.7 Sistemas de apoyo a la toma de decisiones	66
2.7.1 Explotación de las bases de datos	67
2.7.2 Definición	67
2.7.3 Componentes de un sistema de soporte a la decisión	68
Capítulo 3: Estado del arte	69
3.1 Análisis de las Publicaciones Mundiales, Argentinas y de Chubut sobre PLF	69

3.1.1 ¿Cuáles son los países líderes en investigación?	69
3.1.2 ¿Cómo se ha investigado PLF en Argentina?	71
3.1.3 PLF en Chubut: ¿Qué se ha publicado hasta ahora?	75
3.2 Transformando la Producción Ganadera con la PLF	76
3.2.1 ¿Cómo está revolucionando PLF a la industria ganadera global?	76
Bienestar animal	76
Sanidad animal	78
Peso corporal del ganado	79
Seguimiento de animales	80
Personal técnico	84
3.2.2 Ganadería de precisión en Argentina: ¿Realidad o Aspiración?	85
Industria 4.0	85
Ganadería 4.0	86
3.2.3 La experiencia de la Patagonia en la implementación de PLF en la ganadería.	89
3.3 Avances Recientes de PLF en la Ganadería Ovina	91
3.3.1 ¿Cómo está transformando la PLF la producción ovina en todo el mundo?	91
3.3.2 PLF en la producción ovina en Argentina. ¿Un camino hacia la sostenibilidad productiva?	93
3.4 Sistemas de Soporte a la Toma de Decisiones: ¿Cómo se está utilizando en la PLF?	97
3.4.1 ¿Cómo están ayudando los DSS a nivel mundial?	97
DSS para empresas agrícola y ganaderas	98
Aplicaciones relacionadas con Ganadería Ovina	102
3.4.2 DSS en la ganadería argentina: ¿Un desafío pendiente?	104
3.5 ¿Cómo Afectarán estos Avances Tecnológicos a la Producción Ganadera?	106
3.5.1 PLF e IoT: ¿Cuáles son los desafíos a nivel global?	107
Desafíos económicos	107
Aplicabilidad	108
Aspectos Éticos	108
3.5.2 ¿Argentina 4.0?	109
3.5.3 Los retos de la implementación de PLF en la ganadería ovina de Chubut.	110
El Problema del Clima en Chubut	111
El Problema de la Adopción de tecnologías informáticas en Chubut	113
Capítulo 4: Datos y modelos	114
4.1 Interacción con fuentes abiertas de datos	114
4.1.1 Datos abiertos y públicos	115
4.1.2 Portales de Datos en Argentina	116
Datos Argentina	117
Sistema de Información Patagonia Sur	120
Fundación Barrera Zoofitosanitaria Patagónica	122
Federación Lanera Argentina	122
4.2 Herramientas esenciales	123

¿Por qué Python?	123
Librerías de Python y otras herramientas	124
4.3 Crear dataset de entrenamiento	126
4.3.1 ¿Por qué es importante la preparación de datos para el ML?	127
4.3.2 Recolección de datos	127
Datos meteorológicos	128
Datos sobre existencias ovinas	129
Datos sobre lana ovina	131
Datos sobre áreas geográficas	132
4.3.3 Limpieza y depuración de datos	133
Datos meteorológicos	134
Datos sobre existencias ovinas	136
Datos sobre lana ovina	137
4.3.4 Comparar datasets obtenidos	141
Datos meteorológicos	141
Datos sobre existencias ovinas	141
Datos sobre lana ovina	142
4.3.5 Combinar datasets seleccionados	142
4.4 Análisis del dataset	144
4.4.1 Análisis estadístico	145
Dataset 1: Datos a nivel departamental	145
Dataset 2: Datos a nivel departamental más El Mirabueno	146
Dataset 3: Datos a nivel departamental actualizado	147
4.4.2. Visualización de datos	148
Gráficos	148
Dataset 1: Datos a nivel departamental	155
Dataset 2: Datos a nivel departamental más El Mirabueno	156
Dataset 3: Datos a nivel departamental actualizado	158
4.5 Entrenamiento de modelos	162
Modelos regresión lineal	162
Modelos de bosques aleatorios	166
Otros modelos probados	166
Capítulo 5: Desarrollo realizado	167
5.1 Aplicación Web	168
5.1.1 Modelo arquitectónico	168
5.2 Tecnologías utilizadas	169
5.3 Funcionamiento de la Aplicación	172
5.3.1 Registro e Inicio de sesión	172
5.3.2 Pantalla Principal	173
5.3.3 Pantalla Mi Campo	175
5.3.4 Pantalla Datos Cargados	178

5.3.5 Pantalla Perfil	179
5.3.6 Estimaciones Machine Learning	181
5.3.7 Bot Telegram	183
Capítulo 6: Comentarios finales	186
Observación de resultados y conclusiones	187
Líneas futuras de investigación	188
Bibliografía	190
Anexo I	202

Índice de figuras

- 2.2.1. *Smart Farming.* (Página 28)
- 2.4.1. *Diagrama de Venn ramas generales de IA.* (Página 36)
- 2.4.2. *Diagrama de Venn ramas específicas de IA.* (Página 38)
- 2.5.1. *Gráfico del conjunto de datos de onda.* (Página 47)
- 2.5.2. *Predicciones de un modelo lineal en el conjunto de datos de ondas.* (Página 48)
- 2.5.3. *Un árbol de decisiones para distinguir entre varios animales.* (Página 50)
- 2.5.4. *Etapas de implementación de un algoritmo de aprendizaje máquina.* (Página 55)
- 2.5.5. *División de datos en validación cruzada quíntuple.* (Página 59)
- 2.6.1. *Definición de lenguaje natural.* (Página 62)
- 2.6.2. *Definición de lenguaje formal.* (Página 63)
- 2.6.3. *Arquitectura de un sistema de NLP.* (Página 65)
- 3.1.1. *Evolución de investigaciones científicas por año.* (Página 69)
- 3.1.2. *Cantidad de publicaciones en principales revistas científicas.* (Página 70)
- 3.1.3. *Países líderes en investigación.* (Página 70)
- 3.1.4. *Número de artículos relacionados con PLF publicados a nivel mundial en relación con la especie.* (Página 71)
- 3.1.5. *Instituciones de investigaciones Argentinas.* (Página 75)
- 3.2.1. *Porcentaje de sensores más utilizados por cada área de interés en PLF.* (Página 81)
- 3.5.1. *Chubut Köppen.* (Página 111)
- 4.4.1. *Distribución de frecuencia producción de telares.* (Página 149)
- 4.4.2. *Diagrama de caja para altura de árboles.* (Página 150)
- 4.4.3. *Diagrama de dispersión temperatura vs. venta de helados.* (Página 151)
- 4.4.4. *Diagrama de dispersión iris.* (Página 152)
- 4.4.5. *Diagrama de barras ventas en diferentes modelos.* (Página 153)
- 4.4.6. *Diagrama de barras ventas en diferentes modelos.* (Página 154)
- 4.4.7. *Diagrama de barras ventas en diferentes modelos.* (Página 155)
- 4.4.8. *Gráfico multilíneas para finura de departamentos de la meseta.* (Página 157)
- 4.4.9. *Gráfico multilíneas para rinde de departamentos.* (Página 158)
- 4.4.10. *Gráfico multilíneas para existencia de departamentos.* (Página 160)
- 4.4.11. *Gráfico multilíneas para ovejas de departamentos.* (Página 161)
- 4.4.12. *Gráfico multilíneas para corderos de departamentos.* (Página 161)
- 4.4.13. *Gráfico multilíneas para kilos de lana de departamentos.* (Página 162)
- 4.5.1. *Gráfico 3D, predicción de lana para dataset 3.* (Página 163)
- 4.5.2. *Gráfico 3D, predicción de corderos para dataset 3.* (Página 164)
- 4.5.3. *Mejor modelo de predicción de lana.* (Página 165)
- 4.5.4. *Mejor modelo de predicción de corderos.* (Página 165)
- 5.1.1. *Modelo arquitectónico de la aplicación.* (Página 168)
- 5.3.1. *Registro de usuario de la aplicación.* (Página 173)
- 5.3.2. *Inicio de sesión de la aplicación.* (Página 173)
- 5.3.3. *Pantalla principal.* (Página 174)
- 5.3.4. *Filtros de consulta disponibles.* (Página 175)
- 5.3.5. *Pantalla mi campo con gráficos de producción.* (Página 176)

- 5.3.6. Pantalla mi campo con gráficos climáticos. (Página 177)*
- 5.3.7. Pantalla datos cargados de la aplicación. (Página 178)*
- 5.3.8. Pantalla mi perfil de la aplicación. (Página 180)*
- 5.3.9. Solicitud de datos para login Telegram. (Página 180)*
- 5.3.10. Sección Telegram de la aplicación. (Página 180)*
- 5.3.11. Modelo de estimación de kg. de lana. (Página 181)*
- 5.3.12. Modelo de estimación de corderos. (Página 182)*
- 5.3.13. Modelo de estimación de finura de lana. (Página 182)*
- 5.3.14. Modelo de estimación de rinde de lana. (Página 183)*
- 5.3.15. Chat de Telegram. (Página 184)*
- 5.3.16. Comando info Telegram. (Página 185)*
- 5.3.17. Comando por voz Telegram. (Página 185)*

Índice de tablas

2.3.1. Clasificación de los productos finales según Diámetro de Fibras. (Página 31)

2.4.1. Tópicos de la Inteligencia Artificial. (Página 38)

3.1.1. Publicaciones PLF en Argentina. (Página 71)

3.2.1. Tecnologías utilizadas para monitorear animales. (Página 82)

4.3.1. Comparación datasets existencias ovinas. (Página 141)

4.3.2. Comparación datasets producción de lana. (Página 142)

Capítulo 1: Introducción

Chubut es una de las provincias con mayor cantidad de ovinos y de productores ganaderos dedicados a la cría ovina del país y aunque este producto ha perdido importancia relativa desde la aparición de las fibras sintéticas en sustitución de la lana, es todavía una actividad importante. En términos demográficos, Chubut es una de las provincias argentinas con menor densidad poblacional, esto sumado a condiciones ambientales adversas (bajas temperaturas, vientos, escasas lluvias, desertificación de campos, clima árido a semiárido) hacen que el desarrollo de actividades ganaderas en comunidades de pequeños y medianos productores se vea seriamente limitado en su crecimiento. Las actividades ganaderas que se llevan a cabo, se realizan de manera tradicional y aún hoy cuentan con una escasa o nula incorporación de tecnologías de uso corriente en las ciudades y otras áreas rurales del país como por ejemplo telefonía rural, energías alternativas, tanto eléctrica como eólica, que afectan problemas de calefacción, comunicación, seguridad, acceso a mercados, gestión de empresas, indumentaria rural, y acceso a la salud. Asimismo, la ejecución de las políticas del sector agropecuario en Chubut se apoya principalmente en soporte de papel, sin registro electrónico, y a pesar de que, durante el último tiempo hubieron intentos de incorporación de tecnología que mostraron la factibilidad de diversas tareas, no se materializaron en decisiones políticas, por lo tanto han permanecido con idéntica gestión durante los últimos años [17][19][21].

En base a lo anterior, este proyecto de tesina presenta una propuesta de incorporación de tecnología informática para productores ganaderos de Chubut, que busca ayudarlos a enfrentar ciertos problemas y de esta manera mejorar la toma de decisiones en ámbitos productivos.

1.1 Motivación

En el Siglo XXI hemos ingresado a la *Industria 4.0 (I4.0)*. Los materiales y dispositivos inteligentes se comunican entre sí mientras interactúan con el entorno, en este contexto, las computadoras interconectadas con los mismos, conducen a una constante producción de información. La recolección de datos y su utilización constituyen componentes indispensables de los sistemas de producción para producir con eficiencia. La *inteligencia artificial (IA)*, elemento central de esta transformación, la acumulación creciente de grandes cantidades de datos, el uso de algoritmos para procesarlos y la interconexión masiva de sistemas y dispositivos digitales, corresponde a una nueva manera de organizar los medios de producción. La incorporación de tecnología y eficiencia productiva es una necesidad para asegurar el desarrollo futuro de las familias de ganaderos e industriales que se dedican a esta actividad. La Industria 4.0 aplicada a la producción ganadera se conoce como ganadería de precisión, del inglés *Precision Livestock Farming (PLF)* y es un concepto que tanto su significado como su definición están, aún hoy, en un proceso de construcción no solo debido a que se nutre de miradas, enfoques y orígenes de implementación diversos; sino también porque los desarrollos tecnológicos evolucionan a ritmo acelerado y exigen replantear los conceptos aprendidos. A nivel mundial, es un proceso que lleva largos años en desarrollo y que ha sido asociado con denominaciones como *Ganadería inteligente* o *Ganadería 4.0*. La ganadería de precisión, en un sentido amplio, se entiende como el proceso de recolección de

información, automatización de procedimientos, robotización y miniaturización de tecnología para la obtención de datos, con el fin de generar mayor eficiencia, ahorro de tiempo en los procesos, manejo sostenible de recursos, mejoras en la competitividad, reducción de costos y potenciación del agregado de valor y calidad de los productos. Es decir, la Ganadería 4.0 significa utilizar las herramientas disponibles para comprender el sistema productivo, anticipar decisiones, tomar datos certeros con procesos sencillos y lograr medir la actividad de los animales, su producción y el impacto ambiental real de un productor en su establecimiento. En la ganadería tradicional, las decisiones a menudo se basan, únicamente, en la experiencia del productor. En el enfoque PLF, estas decisiones se basan en datos cuantitativos, como la cantidad de lana obtenida en la esquila así como su finura y rinde, la cantidad de pariciones de corderos en el año, etc. Para realizar la obtención, análisis y estudio de dichos datos, los sistemas PLF utilizan análisis de datos, *Aprendizaje Automático (ML)*, sistemas de control y *Tecnologías de la Información y la Comunicación (TICs)*. Algunos de los algoritmos de machine learning utilizados pueden ser: *K-Nearest Neighbors (KNN)*, *Random Forest (RF)*, *Support Vector Machine (SVM)* y, un enfoque más nuevo, *Deep Belief Networks (DBN)*, *Árboles de Clasificación y Regresión (CART)*, *Análisis Discriminante Lineal (LDA)* y *Análisis Discriminante Cuadrático (QDA)* [11].

En Argentina la producción ovina representa un rubro importante dentro del sector ganadero. Tradicionalmente, el país orientó la explotación del ganado ovino hacia la obtención de lana y carne, fundamentalmente en la Patagonia, donde por el ambiente es difícil desarrollar otra actividad agropecuaria; y hoy tiene tres grandes desafíos a corto plazo: el posicionamiento en el mercado global en sintonía con el mercado interno, reducir el impacto ambiental y aumentar la eficiencia en los procesos. La integración de la Industria 4.0 sumada a las buenas prácticas ganaderas, representan la forma más segura e inteligente de hacer ganadería en el siglo XXI porque los mercados internacionales, cada día exigen mayores precisiones sobre diversos aspectos, como por ejemplo el bienestar animal, la resistencia a los antibióticos, los gases de efecto invernadero, y los residuos de medicamentos veterinarios. Estos aspectos a los cuales los países compradores dan cada vez más importancia, se están instalando, también, en nuestra sociedad [55].

En el panorama argentino de producción ovina, Chubut se presenta como una de las provincias de mayor capacidad productiva y, sin embargo, un gran porcentaje de sus actividades ganaderas son llevadas a cabo siguiendo un enfoque «tradicional» apoyados en sistemas de producción anticuados que aún hoy se encuentran vigentes, lo cual lleva a que muchos productores tengan ciclos productivos difíciles acompañados de una baja rentabilidad. En adición a esto, gran parte de las herramientas o aplicaciones informáticas existentes en el mercado pertenecientes al ecosistema de aplicaciones ganaderas, tienen un enfoque mayormente orientado a las características económicas o financieras de la administración de establecimientos ganaderos sin ser capaces de brindar ayuda a la toma de decisiones basándose en datos históricos de los campos [21].

Teniendo en cuenta los aspectos mencionados anteriormente, es que surge la motivación de ofrecer, como futuros profesionales informáticos, una herramienta adaptada a nuevas tecnologías que sirva como ayuda a la toma de decisiones en los ámbitos productivos de las zonas ganaderas de la región.

1.2 Objetivos

La elaboración de esta tesina representa una excelente oportunidad para adquirir nuevos conocimientos y habilidades, centrando el enfoque en el aprendizaje y la mejora constante. En este sentido, a continuación, se expone el objetivo general de la tesina acompañado de los objetivos específicos que se utilizarán para alcanzarlo.

1.2.3 Objetivo general

El objetivo general de la presente tesina, es elaborar un sistema informático a partir del estudio y análisis de los principales factores que afectan la producción de un campo ganadero en la zona de Chubut. Se propone desarrollar una aplicación web que permita, en base al procesamiento de datos históricos climáticos y de producción, asistir la toma de decisiones para mitigar y contrarrestar problemas de relevancia a los productores ganaderos en su ámbito productivo.

1.2.3 Objetivos específicos

- ❖ Investigar, estudiar y documentar el marco teórico de referencia.
- ❖ Interactuar con fuentes abiertas de información para la obtención de datasets existentes relacionados a variables ganaderas de interés.
- ❖ Desarrollar una aplicación web que, utilizando tecnologías modernas y técnicas de inteligencia artificial, asista al productor ganadero en la gestión de su campo y proporcione funcionalidades que mejoren la toma de decisiones sobre el futuro cercano de su inversión.

1.3 Alcance y límites

La presente tesina, plantea el desarrollo de una aplicación web que permita organizar datos meteorológicos, estructurar datos de producción, presentar opciones para visualizar los datos productivos según distintas variables de interés, contar con opciones de visualización de datos históricos y, mediante diversos algoritmos, realizar predicciones y/o estimaciones que brinden previsibilidad para ayudar a tomar mejores decisiones productivas. Además, la aplicación web permitirá la recolección diaria de datos mediante la interfaz misma de la aplicación o a través de un teléfono celular. La experimentación de los modelos estará limitada a datos de los departamentos de la provincia de Chubut, adquiridos de fuentes de datos públicos.

1.4 Resultados esperados

Al concluir esta tesina, el resultado esperado es una aplicación informática experimental que permita mejorar la toma de decisiones en la producción ganadera de la región. La aplicación deberá ofrecer una interfaz intuitiva y de rápida interpretación, con el fin de aumentar su usabilidad y su eficiencia en el proceso de toma de decisiones. Además, se espera haber ampliado y profundizado los conocimientos adquiridos durante el cursado de la carrera,

especialmente en las áreas de inteligencia artificial, visualización de la información y el desarrollo de sistemas de soporte para la toma de decisiones.

Capítulo 2: Marco teórico

En el presente capítulo, se expondrán las bases teóricas que sustentan la presente tesina. Se iniciará con una revisión de la Industria 4.0, seguida de una descripción de la Ganadería 4.0 y también cómo se lleva adelante la ganadería ovina en Chubut. En las últimas secciones, se abordarán, de forma más técnica, conceptos como la Inteligencia Artificial, el Aprendizaje Automático, el *Procesamiento del Lenguaje Natural (NLP)* y los *Sistemas de Soporte para la Toma de Decisiones (DSS)*.

2.1 Industria 4.0

A grandes rasgos, hasta el momento, se han producido cuatro revoluciones tecnológicas, pasando de la Industria 1.0 a la 4.0. La Industria 1.0 progresó las actividades humanas desde la producción agrícola lenta y de pequeña escala hasta una sociedad industrializada; la gama de productos estaba restringida y la mayoría de los productos básicos eran de origen agrícola. Durante la Industria 2.0, los bienes industriales aumentaron significativamente tanto en volumen como en variedad; la electricidad, los dispositivos electrónicos y mecánicos y los vehículos motorizados representaron importantes avances técnicos y tecnológicos. La Industria 3.0 se caracterizó por desarrollos tecnológicos que tuvieron efectos significativos en la industria electrónica en particular, como la transición de lo analógico a lo digital y la modificación de los productos electrónicos, pasando de un diseño integral a uno modular acortando su ciclo de vida promedio. Finalmente, luego de los avances tecnológicos continuos de las últimas décadas, emergió la denominada Industria 4.0, la cual ha ido asumiendo un papel central en el discurso entre empresarios, expertos, académicos y legisladores acerca del futuro de los sistemas de producción industrial. En la actualidad, la Industria 4.0 es ampliamente reconocida como una transformación radical en los sistemas económicos y de producción a nivel global [11].

El término Industria 4.0 hace referencia a la denominada Cuarta Revolución Industrial, que nace de la evolución tecnológica impulsada por el desarrollo de los sistemas embebidos, su conectividad y la oportuna convergencia del mundo virtual y físico. Todo esto brinda enormes capacidades de integración de objetos, información y personas que tienen el potencial de impulsar un salto cualitativo en la producción y utilización de bienes y servicios [5]. La Industria 4.0 promete una revolución al fusionar técnicas avanzadas de operaciones y producción con tecnologías inteligentes que se integrarán en las organizaciones, las personas y los activos. Esta fusión incluye el uso de tecnologías inteligentes como la robótica, la inteligencia artificial, el análisis de datos masivos, la nanotecnología y el IoT. Las organizaciones deben identificar las tecnologías adecuadas y hacer inversiones estratégicas para, de esa manera, poder aprovechar las oportunidades de negocio en un mercado en constante evolución [2].

2.1.1 ¿Qué es la industria 4.0?

La Industria 4.0, consiste en la digitalización de la industria y todos los servicios relacionados con la empresa, es decir, se produce una unión entre el mundo virtual y el real, ya que se utilizan las nuevas tecnologías en todas las partes de la empresa, incluyendo los procesos productivos. De esta forma, las instituciones son capaces de autogestionarse de forma más autónoma, adaptándose a los requisitos dinámicos del mercado. La Industria 4.0 permite interconectar todas las unidades de producción de una empresa para que sea capaz de adaptarse a los medios de producción y a los recursos de los que dispone para intentar aprovecharlos al máximo [1].

2.1.2 ¿Por qué es importante la Industria 4.0?

Lograr entender el potencial de esta cuarta revolución industrial es de vital importancia para las organizaciones en general debido a que no solo afectará a los procesos de fabricación tradicionales como se conocen actualmente, sino que propone un alcance mucho más amplio, afectando a todos los sectores de las industrias, e incluso a la sociedad. La Industria 4.0 puede mejorar las operaciones de negocio y el crecimiento de los ingresos, transformando los productos, la cadena de suministro y las expectativas de los clientes. Es factible que dicha revolución modifique la forma en la que realizamos nuestras actividades, incluso puede influir en la manera en que los clientes interactúan con las empresas y en las experiencias que esperan tener al hacerlo [2].

Además, las tecnologías relacionadas con la Industria 4.0 también pueden conducir a productos y servicios completamente nuevos. El uso de sensores y dispositivos portátiles, el análisis y la robótica, entre otros, permitirán mejoras en los productos de diversas maneras, desde la creación de prototipos y pruebas hasta la incorporación de conectividad a productos previamente desconectados. Estos cambios en los productos se traducen, a su vez, en cambios en la cadena de suministro y, consecuentemente, en los clientes [2].

2.1.3 Industria 4.0 en el Mundo

La cuarta revolución industrial ha hecho que la manufactura y los sistemas de producción tengan cambios de una manera rápida e inesperada, por lo que los países han tenido que irse adaptando a estos cambios. Un reporte realizado por el *World Economic Forum (WEF)* en el 2018 reveló que sólo 25 países son líderes, es decir, se encuentran en la mejor posición para lograr una ventaja competitiva en tecnologías de la Industria 4.0 [4]. Para este ranking el WEF analiza los «motores» de los sistemas de producción de cien países, como son la tecnología y la innovación, el capital humano, el comercio global y la inversión, el marco institucional, recursos sostenibles y el entorno de la demanda. También estudia la estructura de la producción, que mide el tamaño y la complejidad de la producción de un país. Según este reporte, a la cabeza están las grandes economías mundiales, Estados Unidos y Alemania, y países en donde la I+D tiene dimensiones industriales, Japón, Corea del Sur, Dinamarca, Suecia o Finlandia [6].

Alemania se ha establecido como el punto de referencia mundial de la Industria 4.0, caracterizados por tener una industria manufacturera especializada y exitosa por su

innovación y soluciones de software, este país se sitúa en uno de los mejores lugares para ser líder en el mercado y el proveedor principal de soluciones para la I4.0. En los EE. UU. la Industria 4.0 está teniendo, de facto, un impacto importante en las iniciativas de fabricación inteligente; otros países donde la Industria 4.0 se está adoptando incluyen el Reino Unido (Industria 4.0 y el trabajo en torno a 4IR), Japón (donde hay, una colaboración con la Iniciativa Robot Revolution de Japón), China (donde el esquema de la Industria 4.0 está en la base de 'Made in China 2025') y las numerosas iniciativas de la *Unión Europea (UE)*. El 23 de marzo de 2017, solo la UE analizó planes para alinear las 12 iniciativas nacionales de transformación de la industria ya existentes y las 9 futuras. Básicamente, todos ellos están aprovechando el concepto de Industria 4.0, independientemente de sus muchos nombres diferentes [3][4].

Existen dos variables claves para poder unirse a la cuarta revolución industrial, una de ellas es la adopción y difusión de nuevas tecnologías, es decir que las empresas absorban masivamente las nuevas tecnologías digitales, de manera que los procesos productivos en los distintos sectores y tipos de empresa se implementen a través de conexiones inalámbricas que asocian a máquinas, software y datos lo cual es conocido bajo la denominación de *Internet Industrial de las Cosas (IIoT)*. De esta manera la naturaleza de las empresas muta hacia esquemas más flexibles y descentralizados, con un mayor rol de las plataformas digitales como sistemas de intercambio. La segunda variable es la adquisición de las habilidades y conocimientos que serán demandados en la cuarta revolución industrial por parte de los trabajadores, es decir, que el avance de la educación y la formación de habilidades equipare al de la tecnología. Así, la transformación estructural implica el pasaje de trabajadores desde puestos de baja productividad hasta otros de productividad media o alta [8].

En resumen, algunos países líderes en la Industria 4.0 se encuentran en esta posición avanzada, como resultado de haber realizado grandes inversiones en tecnologías y capacitación de su fuerza laboral. En general, la preparación de los países para la Industria 4.0 es un proceso en constante evolución y los gobiernos y empresas deben trabajar juntos para impulsar la adopción de estas tecnologías y aprovechar los beneficios que ofrecen.

2.3.4 Industria 4.0 en Argentina

En el año 2018, el World Economic Forum realizó un estudio sobre la preparación para el futuro de la producción, cuyos resultados indican que Argentina no se encuentra preparada para enfrentar los desafíos venideros en la adaptación a la Industria 4.0. Este estudio dividió 100 países dentro de cuatro categorías:

- ❖ Países líderes, 25 en total.
- ❖ Países heredados, 10 en total.
- ❖ Países con economías de alto potencial, 7 en total.
- ❖ Países nacientes, 58 en total.

Argentina se encuentra en la última categoría, y allí ocupa el puesto 50 en estructura de producción con 4.91 puntos. Los «motores» o impulsores de los sistemas de producción son los que posicionan un país en una vía más o menos ventajosa para beneficiarse de la cuarta revolución industrial. De diez puntos posibles, Argentina obtiene un total de 4.4 de media: 3.8 en tecnología e innovación (posición 68), 4.9 en capital humano (posición 56), 3.1 en

comercio global e inversión (posición 92), 4.2 en el marco institucional (posición 83), 5.8 en recursos sostenibles (posición 63) y 5.0 en el entorno de la demanda (posición 42) [7].

Estos valores no son una sorpresa teniendo en cuenta que durante las sucesivas revoluciones tecnológicas Argentina tendió a ubicarse en el grupo de los «rezagados» debido, principalmente, a la incapacidad de buena parte de las empresas y los trabajadores para absorber completamente las nuevas tecnologías y traducirlas en ganancias de productividad. Sólo un puñado de «ganadores locales» logró seguir los desarrollos globales.

Los países en desarrollo pueden enfrentar mayores desafíos para adoptar la Industria 4.0 debido a limitaciones en su infraestructura tecnológica, educativa y financiera, así como a la falta de habilidades y capacitación en su fuerza laboral. Argentina podría mejorar su adopción y aprovechamiento de la Industria 4.0 mediante la inversión en infraestructura tecnológica y educativa, la colaboración público-privada y la implementación de políticas públicas que fomenten la adopción de estas tecnologías y la transformación digital de las empresas.

2.1.5 Industria 4.0 en Chubut

Chubut es una región que enfrenta múltiples desafíos en áreas claves para su desarrollo. Uno de los mayores problemas es la situación económica, caracterizada por la escasez de empleos y el aumento de los precios de los productos básicos. Además, existe una falta de acceso a la formación y capacitación necesarias para preparar a la población para los trabajos del futuro, afectando la empleabilidad y las oportunidades de progreso de la región. Los fenómenos climáticos extremos también juegan un papel importante en la realidad de la provincia. La sequía, el frío y el calor extremo son factores que dificultan la producción agropecuaria y la industria local, dificultando así, el desarrollo económico de la región. A su vez, esto ralentiza la adopción de nuevas tecnologías y el avance hacia una posible Industria 4.0.

Considerando que los dos pilares más fuertes para entrar en la cuarta revolución industrial como sociedad en conjunto, son la difusión por parte del gobierno (quizás a través de nuevas políticas) y un sistema educativo que permita conseguir personas con capacidades para adquirir las habilidades tecnológicas necesarias, Chubut no pareciera contar con las herramientas educativas y económicas para incorporarse a la Industria 4.0. Si es cierto, que a pesar de no estar bajo un gobierno que brinde el apoyo necesario, las empresas pueden potenciarse al «unir fuerzas» o «tomar valor» y empezar su transformación. Esto las ayudará a optimizar los niveles de calidad, porque al disponer de una industria automatizada se puede ser más preciso con pesos, medidas etc. Lógicamente, si hay más eficiencia, habrá menos costes. Los procesos automatizados requieren menos personal, por lo que puede haber menos errores. También se reducen los tiempos de producción y hay mayor seguridad del personal al no exponerse a tantos procesos productivos peligrosos. Se logra aumentar en gran medida la competitividad empresarial y se ofrece una mejor respuesta a las necesidades de cada mercado. Cuando aumenta la eficiencia del uso de recursos, se consigue un mejor cuidado del medioambiente [1][9][10].

2.2 Ganadería 4.0

Dado el aumento de la escala de las explotaciones y del número de animales criados, la capacidad de observación y la experiencia práctica en la que confiaban los agricultores en el pasado ya no es suficiente para garantizar una gestión de rebaños diaria de manera eficiente. Según un estudio realizado en 2006 por la *Organización de las Naciones Unidas para la Agricultura y la Alimentación (FAO)*, para el año 2050 la agricultura mundial deberá aumentar hasta en un 70% su capacidad de producir alimentos para poder abastecer la demanda alimenticia. Por otro lado, la presión de los precios en el mercado mundial obliga a los ganaderos a centrarse en la eficiencia y la producción a gran escala. El aumento constante del tamaño de las explotaciones ganaderas y de los rendimientos productivos de los animales, representa una exigencia cada vez mayor tanto para el propio animal como para el ganadero. Además de la creciente demanda de productos derivados de animales, la ganadería moderna se enfrenta a varios retos: aumentar la producción al tiempo que aumenta la salud y el bienestar animal y reducir el uso de energía y las emisiones de efecto invernadero derivando en una mejora en la eficiencia del uso de recursos naturales; o dicho de otra manera, aumentar la capacidad y la competitividad del sector agroalimentario. Estos temas son delicados y están fuertemente regulados por leyes nacionales e internacionales; dada la importancia, los métodos para monitorear y controlar los requisitos regulatorios obligatorios y voluntarios deben ser rápidos, efectivos y eficientes. El futuro de la economía en general pasa por la digitalización del medio rural. Esta adaptación digital es necesaria para que continúe siendo un medio vivo y poblado y un sector competitivo y sostenible económica, social y medioambientalmente. Términos como ganadería de precisión, *Agricultura de Precisión (PA)*, *Smart Farming (SF)* e Industria 4.0 están cada vez más extendidos en los ámbitos productivos y de desarrollo. Hasta la fecha, varios países y las principales organizaciones mundiales utilizan estos términos como pilares para el desarrollo de una gestión agrícola sostenible y fácil de usar [11][12].

A continuación, en esta sección, describiremos los **pilares** principales que sostienen a la ganadería 4.0 y finalmente se abordan los conceptos de PLF y SF.

2.2.1 Industria 4.0

En la sección anterior (ver sección 2.1) mencionamos la industria 4.0, un nuevo paradigma que se enfoca en la automatización de fábricas, la integración de internet en los procesos industriales y la utilización de TICs para desarrollar dispositivos, máquinas y sistemas inteligentes. Este proceso busca lograr una producción industrial completamente automatizada e interconectada, y es impulsado por innovaciones tecnológicas como el *Internet de las Cosas (IoT)*, *Big Data*, vehículos eléctricos, impresión 3D, *Computación en la Nube*, inteligencia artificial y *Sistemas Ciberfísicos* [11]. Estas tecnologías están diseñadas para mejorar la eficiencia, reducir costos y aumentar la flexibilidad y la adaptabilidad en los procesos industriales, y están transformando la forma en que las empresas producen y entregan bienes y servicios a sus clientes. La Industria 4.0 es un paso importante en la

evolución de la producción industrial y se espera que siga transformando la manera en que vivimos y trabajamos en el futuro.

2.2.2 Internet de las cosas

El Internet de las cosas es una red de dispositivos interconectados que se comunican, detectan e interactúan con entornos internos y externos a través de tecnología integrada. Como tecnología clave en la Industria 4.0, IoT está cambiando varios aspectos de nuestras actividades diarias al generar un entorno conectado inteligente. Hogares inteligentes, Internet industrial y dispositivos portátiles son algunos de los ejemplos de uso diario. Este enfoque no solo evalúa objetos, sino también procesos, datos, personas, incluso animales y fenómenos atmosféricos, todos los cuales pueden considerarse como una variable [11]. Paralelamente al desarrollo, la evolución y la difusión del IoT, la tecnología de sensores se hizo disponible en muchas formas y tamaños. Aunque el objetivo de los sensores es claramente recopilar datos, el propósito principal de la IoT es facilitar la transferencia de datos a través de una red para almacenarlos en la nube, acceder a ellos a distancia y procesarlos mediante la computación en nube en potentes servidores remotos. La ingente cantidad de datos recopilados (y almacenados) por la combinación de IoT y sensores dio lugar al concepto de big data y, con él, los científicos buscaron la modernización de la agricultura [13].

En la actualidad se pueden recopilar datos sobre diversos escenarios de producción ganadera. El monitoreo incluye tres aplicaciones clave: monitoreo de ganado, monitoreo de campo y monitoreo de gases de efecto invernadero. Los avances en la tecnología de sensores permiten la captura de medidas fisiológicas, de comportamiento y de productividad de animales individuales para ayudar al paradigma de producción inteligente/precisa y sostenible. El IoT permite a los granjeros supervisar el ganado mediante el uso de múltiples sensores que monitorean múltiples variables animales como la temperatura, la frecuencia cardíaca y la digestión. Las aplicaciones de monitoreo de campo están diseñadas para informar sobre diferentes condiciones, como la fertilidad del suelo, la temperatura, la humedad, el contenido de gas, la presión del aire y del agua, y la presencia de enfermedades en los cultivos. El uso de sensores y dispositivos IoT también ha logrado eliminar en cierta medida la necesidad de intervención manual [11][13].

2.2.3 Big Data

Big Data es un término que hace referencia a conjuntos de datos de gran tamaño, complejidad y variedad, los cuales no pueden ser procesados y analizados de forma efectiva utilizando tecnologías y herramientas convencionales. Estos datos son generados por diversas fuentes, tales como redes sociales, dispositivos móviles, sensores, registros de transacciones y otros. En el caso específico de PLF, estos datos son generados principalmente por redes de sensores inalámbricos, estaciones meteorológicas, Vehículos Aéreos no Tripulados (UAV) y servicios externos.

La complejidad en los datos se debe en gran parte a la naturaleza no estructurada y diversa de los mismos. El análisis de Big Data se centra en encontrar patrones, tendencias y relaciones dentro de los datos, con el fin de mejorar la toma de decisiones empresariales y generar valor para las organizaciones. Además, para procesar y analizar estos datos, se requiere el uso de

tecnologías y herramientas específicas, como Hadoop, Spark, NoSQL, y otras plataformas de procesamiento distribuido. La capacidad de manejar Big Data se ha convertido en un requisito fundamental para muchas empresas y organizaciones en diversos sectores, incluyendo finanzas, salud y comercio electrónico, entre otros [11][152].

2.2.4 5G

La tecnología 5G es la última generación de redes móviles que se caracteriza por su alta velocidad, baja latencia y capacidad para conectar una gran cantidad de dispositivos de manera simultánea. Esto permite la creación de redes IoT más robustas y eficientes, en las que sensores y dispositivos inteligentes recopilan y transmiten datos en tiempo real.

Dentro de la industria agrícola, esta tecnología cuenta con enorme potencial de desarrollo debido a que puede mejorar la eficiencia en el uso de los recursos y aumentar la productividad de los campos. Con sensores conectados a través de la red 5G, es posible detectar las necesidades de agua y nutrientes de los cultivos y ajustar automáticamente el suministro para obtener el mejor rendimiento y calidad de la cosecha. Además de que puede mejorar la eficiencia en el uso de maquinaria agrícola y el monitoreo de condiciones ambientales, como la temperatura y la humedad, lo que puede prevenir enfermedades y reducir los costos de mantenimiento.

Sin embargo, las zonas rurales enfrentan un desafío para acceder a esta nueva tecnología debido a la menor inversión en infraestructura y menor densidad poblacional, es por ello que es importante fomentar la inversión en infraestructura de redes móviles y la capacitación de los agricultores para que puedan aprovechar las oportunidades que ofrece la tecnología 5G.

Las aplicaciones 5G en agricultura y ganadería incluyen vehículos aéreos no tripulados, monitoreo en tiempo real, asesoramiento en línea y mantenimiento predictivo, realidad aumentada y virtual, robots impulsados por IA, análisis de datos y repositorios en la nube [11].

2.2.5 Computación en la nube

La computación en la nube es una forma de utilización de recursos computacionales que tiene como característica principal el uso de internet como canal de operación. Los recursos de cómputo y los datos se distribuyen en computadoras y otros dispositivos compartidos y se entregan a los usuarios finales a pedido. La computación en la nube favorece a las organizaciones en el abaratamiento de costos de infraestructura. No se requiere que los usuarios finales tengan control directo sobre los fundamentos de la «nube» o que posean conocimiento profesional de la misma; todo lo que necesitan saber es qué tipo de recursos requieren y cómo Internet puede proporcionar esos servicios adecuados. Las cinco características de la computación en la nube son: servicio bajo demanda, acceso a Internet, sondeo de recursos, elasticidad rápida y capacidad de cálculo. En general, IoT vincula Internet a ciertos sensores, ejecuta los programas y proporciona control remoto. La computación en la nube y los macrodatos ofrecen numerosas herramientas para abordar el problema del procesamiento y el almacenamiento de todos los requisitos y limitaciones asociados con la agricultura inteligente [11][85].

2.2.6 Ganadería de Precisión

La idea de la agricultura de precisión se presentó por primera vez a principios de la década de 1990 en los EE. UU. En 1997, la Cámara de Representantes describió a PA como un «sistema agrícola integrado basado en información y producción, diseñado para aumentar la eficiencia, la productividad y la rentabilidad de la producción agrícola completa, específica del sitio y a largo plazo, al mismo tiempo que se minimizan los impactos en la vida silvestre y el medio ambiente». Si bien durante mucho tiempo se ha asumido que los procesos que involucran organismos vivos (procesos biológicos) son demasiado complicados para ser monitoreados y regulados automáticamente, las nuevas tecnologías emergentes están brindando oportunidades para desarrollar herramientas de control y monitoreo en línea completamente automatizadas para muchos de estos procesos. Los animales, como todos los organismos vivos, son complejos, individualmente distintos y variantes en el tiempo, y son el núcleo principal de los denominados sistemas CIT [11].

La ganadería tradicional se enfoca en la cría de animales para obtener productos que son consumidos por la población. En particular, el ganado es una fuente importante de proteína en la dieta humana, aportando alrededor del 33% de la proteína total consumida. Recientemente, ha surgido el concepto de Ganadería de Precisión (PLF): un enfoque holístico que agrega tecnologías de información y comunicación (TICs) para mejorar el proceso de cultivo. PLF juega un papel importante en la cuarta revolución industrial, ya que utiliza las TICs para reducir los costos de inversión y aumentar tanto la producción como la salud animal [14].

En la ganadería tradicional, muchas veces las decisiones se basan únicamente en la experiencia del productor. En PLF, tales decisiones se basan en datos cuantitativos. Además, se pueden obtener datos cuantitativos en tiempo real, y para obtenerlos y estudiarlos, los sistemas PLF utilizan análisis de datos, ML, sistemas de control y TICs [14].

En la comunidad de la ciencia animal, se han asignado diferentes terminologías, por ejemplo, ganadería de precisión, ganadería inteligente y agricultura animal inteligente, por nombrar algunas, al mismo paradigma: cómo aumentar de forma sostenible la producción de alimentos manteniendo el bienestar animal y reduciendo la carga medioambiental mediante la fusión de la adquisición de datos (sensores), el almacenamiento (IoT) y la transformación con el análisis de predicción utilizando herramientas de inteligencia artificial (IA). Estas terminologías pueden tener diferencias conceptuales notables o sutiles, pero, en definitiva, buscan el mismo resultado de gestionar de forma inteligente o precisa las explotaciones ganaderas [13]. Algunas de sus definiciones son:

- ❖ PLF se puede describir como el sistema tecnológico para el seguimiento en tiempo real de animales de hacienda, destinado a gestionar la variabilidad temporal de la unidad de producción más pequeña manejable, conocido como el «enfoque por animal» [11].
- ❖ PLF utiliza los principios y la tecnología de la ingeniería de procesos para gestionar la producción ganadera mediante sensores «inteligentes» para controlar el crecimiento animal, la obtención de productos animales como leche, carne y huevos, las enfermedades endémicas, el comportamiento animal y las condiciones internas dentro de la unidad de producción, como la temperatura y las emisiones de gases [13].

- ❖ PLF es un enfoque novedoso en los sistemas de producción ganadera que tiene como finalidad integrar en la gestión de la actividad el monitoreo y control automatizado de la producción, reproducción, salud y bienestar animal, así como la huella ambiental de la ganadería en tiempo real. Esto se logra mediante el uso intensivo de tecnología y la ingeniería de procesos para mejorar la sostenibilidad y productividad ganadera [11] [15].

Los principales objetivos de PLF son: (i) identificar la alimentación del ganado más adecuada, (ii) reducir el impacto ambiental a través de una gestión eficiente de los recursos, (iii) gestionar los procesos de cultivo para hacer una sinergia perfecta con la alimentación del ganado, (iv) garantizar la seguridad alimentaria a través de la trazabilidad (registro documental desde la producción hasta el consumo) de los productos, y (v) mejorar la salud animal y la eficiencia de los cultivos [14].

Para obtener un sistema de control y seguimiento funcional de este tipo, se deben cumplir tres condiciones. La primera condición es monitorear constantemente las variables animales y que esta información sea analizada consistentemente. El segundo requisito previo es la disponibilidad de un pronóstico realista (expectativas) sobre cómo diferirán las variables animales o cómo responderá el animal a los cambios ambientales. La tercera condición es que estas predicciones, junto con las mediciones digitales, se incorporen a un algoritmo analítico para el seguimiento o manejo automatizado de animales y para el monitoreo en línea de la salud y el bienestar animal. Aunque el principio general de utilizar tecnología y automatización para aumentar la precisión en la fabricación industrial es fácilmente transferible a PLF, el cambio de productos inertes a animales vivos presenta problemas únicos que deben tenerse en cuenta al crear procedimientos de PLF [11].

En la actualidad se dispone de instrumentos precisos, potentes y de bajo costo debido al avance tecnológico de la última década. Estos incluyen cámaras, micrófonos, sensores, sistemas de redes inalámbricas, conexiones a Internet y almacenamiento en la nube. El propósito de estos instrumentos tecnológicos no es sustituir, sino ayudar a un agricultor que sigue siendo el aspecto más importante del buen manejo animal [11].

Además de los beneficios mencionados anteriormente, el gran potencial de PLF se centra en la posibilidad de obtener alertas tempranas o notificaciones, que ofrecen al agricultor el poder de actuar tan pronto como surjan los primeros signos de deterioro del bienestar o la salud animal. De hecho, se han desarrollado modelos de predicción precisos en el contexto de PLF que envían mensajes de advertencia a los agricultores en función de la información de los insumos ambientales y animales y pueden ayudar a detectar cualquier desviación del patrón habitual. Gracias a la información detallada reportada con respecto al estado de su ganado, los productores pueden tomar fácilmente medidas correctivas. Los continuos desarrollos tecnológicos y la validación de aspectos teóricos y prototipos contribuyen a la creación de herramientas de diagnóstico que pueden detectar problemas de granja sin manipular a los animales (recolección de datos no invasiva y sin contacto) o inducir estrés, brindando la capacidad de reconocer un brote de enfermedad días antes de lo que lo haría cualquier agricultor. En este contexto, los beneficios para los agricultores incluyen una mejor toma de decisiones, mayor atractivo para los jóvenes agricultores y un efecto beneficioso en la resolución de las deficiencias analíticas del usuario final a través de la conversión de datos

sin procesar en información útil que actualmente solo se puede obtener a través del análisis y la interpretación de expertos [11].

PLF se centra en el vínculo entre diversas disciplinas científicas y las partes interesadas de la industria ganadera. El enfoque PLF implica un alto grado de cooperación entre muchos campos de estudio, incluidos los científicos de animales, los veterinarios clínicos, los técnicos, los científicos de datos, los ingenieros y otros. La tarea más importante que enfrentan las aplicaciones de campo con orientación hacia el PLF es la de convertir diferentes tipos de datos provenientes de sensores y diversas fuentes en información aplicable, y esto es precisamente lo que la ciencia de datos se encarga de hacer. Hay dos impedimentos en este sentido: (1) el problema de la ciencia de datos asociado con el desarrollo de técnicas basadas en datos, capaces de generar predicciones dependiendo de una variedad de mediciones de múltiples fuentes; y (2) la generación de conocimiento de utilidad a partir de predicciones basadas en datos, requiere resolver el problema de traducción asociado con la interpretación de dichas predicciones. [11].

La escala en la que se gestiona el proceso está implícita en la definición de PLF. En términos generales, cuanto menor sea el tamaño de la operación en la que se utiliza PLF, mayor será el costo y se requerirá un mayor rendimiento de la inversión de capital para justificar una gestión más detallada y precisa de la empresa. Finalmente, la digitalización brinda la capacidad de hacer que la agricultura sea más competitiva. La llegada de las TIC en la industria ganadera y el uso cada vez mayor de IoT han abierto una nueva era de conectividad en la que las cosas, las personas y los animales forman parte de un intercambio de redes de datos, lo que lleva a una nueva filosofía de agricultura [11].

En resumen, los procesos de PLF e Industria 4.0 no son cronológicamente distintos ni totalmente independientes entre sí. La innovación industrial ha permitido a los investigadores utilizar los sistemas más innovadores para cambiar y mejorar el enfoque de gestión de la ganadería. Al mismo tiempo, el uso de tecnologías de monitoreo en tiempo real basadas en Internet, fomenta la innovación industrial. [11].

2.2.7 Smart Farming

Podemos definir smart farming como el conjunto de tecnologías pertenecientes a la llamada Cuarta Generación Industrial, que incluye entre sus herramientas: Tecnologías de la Información y Comunicación, técnicas analíticas de *Big Data* y la minería de datos, servicios en la nube, Internet de las Cosas e Inteligencia Artificial, entre otras estrategias aplicadas en el contexto de la agricultura y la ganadería, con el fin de optimizar la calidad de los productos del campo, preservar los recursos naturales y proteger de manera más eficaz el medio ambiente [56].

Se puede observar en [54] las diferentes ramas que componen el ámbito del smart farming y las tecnologías asociadas a cada una de ellas. Basándose en esta información se construyó la figura 2.2.1.

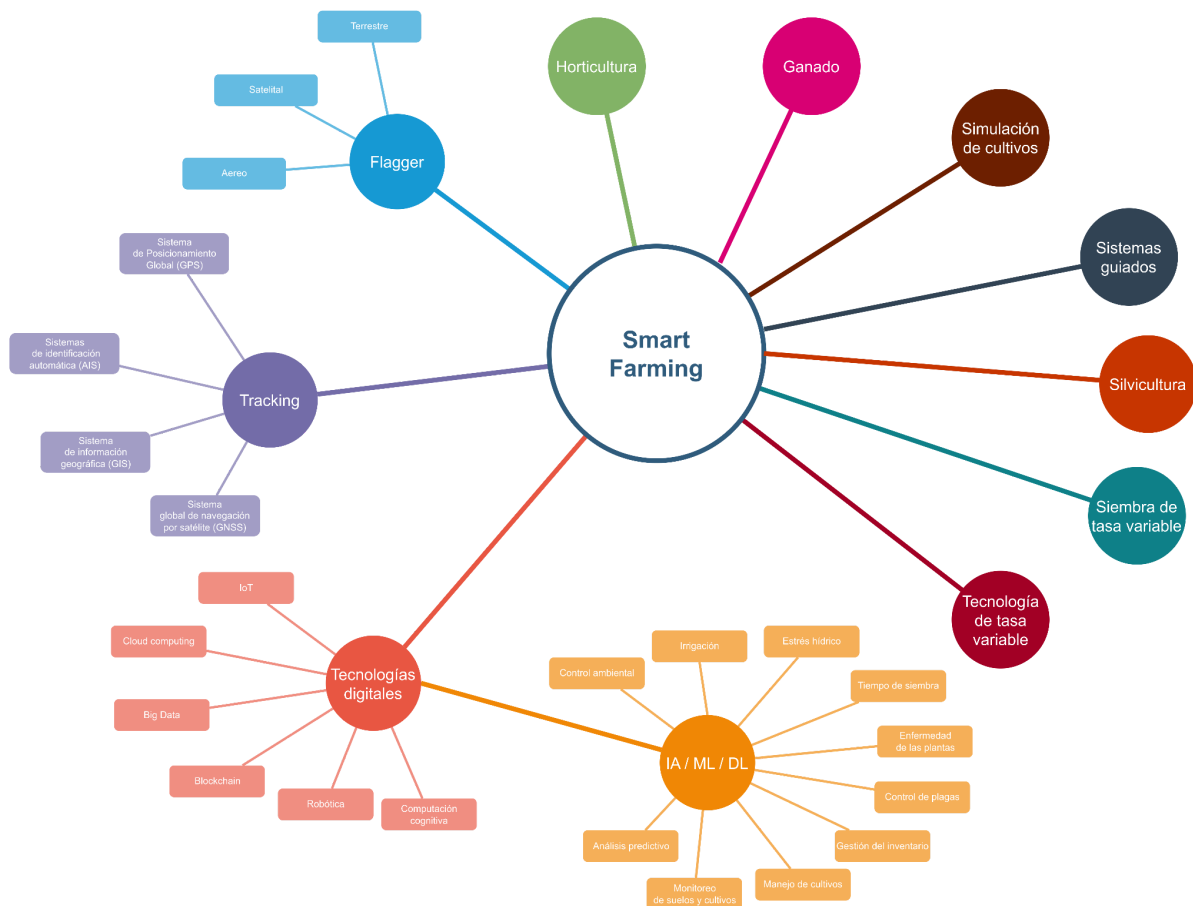


Figura 2.2.1. Smart Farming.

El Smart Farming se divide en varias ramas que buscan aplicar tecnologías innovadoras en la producción agropecuaria. Una de ellas es Tracking, que se enfoca en el seguimiento y localización de objetos agrícolas en tiempo real mediante el uso de tecnologías como:

1. *Sistema de Posicionamiento Global (GPS)*: Sistema que utiliza señales de satélite para determinar la ubicación exacta de los objetos agrícolas.
2. *Sistemas de identificación automática (AIS)*: Tecnología utilizada para identificar y rastrear objetos agrícolas en tiempo real.
3. *Sistema de información geográfica (GIS)*: Tecnología que permite la visualización y análisis de datos geoespaciales para una mejor toma de decisiones.
4. *Sistema global de navegación por satélite (GNSS)*: Tecnología que utiliza señales de satélite para guiar la navegación y el movimiento de objetos agrícolas.

Otra rama es Horticultura, que busca mejorar la producción de hortalizas y frutas mediante la implementación de tecnologías. Ganado se enfoca en el manejo del ganado y la producción de carne y leche. Flagger utiliza vehículos terrestres, satelitales y aéreos (drones) para el seguimiento y monitoreo de objetos agrícolas, mientras que Sistemas Guiados utiliza tecnologías para guiar maquinarias en la siembra y cosecha de cultivos. Las Tecnologías Digitales, como Big Data, IoT, Cloud Computing, Blockchain y Robótica, también se utilizan en la producción agropecuaria. Además, la Tecnología de Tasa Variable se enfoca en la aplicación de insumos agrícolas en cantidades variables según las necesidades de cada zona

del cultivo, mientras que la Siembra de Tasa Variable se enfoca en la siembra de semillas en cantidades variables según las necesidades de cada zona del cultivo [57][58][59].

2.3 Ganadería en Chubut

Los sistemas de producción ovina se enfrentan a numerosos desafíos, que hacen de la toma de decisiones un proceso lleno de riesgos e incertidumbres. En este sentido, cualquier herramienta de PLF es útil, ya que permite a los tomadores de decisiones evaluar el comportamiento de las variables y sus interrelaciones, además de utilizar información previa o relacionada para predecir resultados y simular diferentes escenarios. El advenimiento de los modelos de predicción ha permitido monitorear el peso de un animal y determinar el mejor momento para su venta. Además, permite a los productores estimar el peso de faena y los principales cortes comercializables del animal, antes del sacrificio. Toda esta información está directamente asociada a la rentabilidad y el éxito de la actividad productiva [16].

2.3.1 Chubut

La Provincia del Chubut, tiene una superficie de 224.686 km² extendiéndose entre los paralelos 42° y 46° de latitud sur, y en base a los datos del INDEC del último censo disponible del 2022, la población total de la provincia es de 603.120 habitantes, de la cual aproximadamente el 8,8% es población rural [17][18][19]. Está ubicada al centrosur de la región patagónica, que ocupa la mitad sur del país, limitando al norte con Río Negro, al este con el mar Argentino (océano Atlántico), al sur con Santa Cruz y al oeste con Chile, cuyo límite está determinado por la cordillera de los Andes [20].

La actividad económica más importante de la provincia es la explotación de hidrocarburos fósiles no renovables (petróleo, gas butano) siendo ésta la que produce el 13% del petróleo argentino, el casi 2 % del gas total del país, el 21% de la pesca nacional y existen minas e importantes yacimientos internacionales de uranio, plomo, oro y plata. También es la provincia con mayor cantidad de ovinos y de productores ganaderos dedicados a la cría ovina del país y aunque este producto ha perdido importancia relativa desde la aparición de las fibras sintéticas en sustitución de la lana, es aún hoy una actividad considerable [17][21].

Chubut es una de las provincias con menor densidad de poblacional de Argentina, esto sumado a las condiciones ambientales adversas (bajas temperaturas, vientos, escasas lluvias, desertificación de campos, clima árido a semiárido) hacen que el desarrollo de actividades ganaderas en comunidades de pequeños y medianos productores se vea seriamente limitado en su crecimiento. Además, el desarrollo de tales actividades se lleva a cabo de manera tradicional y aún hoy cuenta con una escasa o nula incorporación de tecnologías de uso corriente en las ciudades u otras áreas rurales del país, como por ejemplo telefonía rural, energías alternativas, tanto eléctrica como eólica, que afectan problemas de calefacción, comunicación, seguridad, acceso a mercados, gestión de empresas, indumentaria rural, y acceso a la salud. Asimismo, la ejecución de las políticas del sector agropecuario en Chubut se apoya en sistemas antiguos, con soporte en papel, sin registro electrónico, implementadas hace cincuenta años, y a pesar, que durante el último tiempo hubo intentos de incorporación de tecnología que mostraron la factibilidad de diversas tareas, no se materializaron en

decisiones políticas hacia el sector, por lo tanto, han permanecido con idéntica gestión durante todos estos años [18][21][22].

2.3.2 Ganadería Ovina

Chubut es la provincia líder en la cría de ovinos en Argentina, con la mayor cantidad de cabezas de ganado ovino en el país. Además, cuenta con una cultura de producción ovina de la raza merino arraigada por más de 100 años, lo que le otorga una ventaja comparativa en la producción de lanas finas. Su geografía se caracteriza por un ecosistema árido que ocupa el 85% de su territorio, donde la lluvia es escasa y variable a lo largo de los años. Solo en una delgada franja costera y en la precordillera se registran promedios ligeramente superiores, mientras que en el centro de la provincia predominan los peladales y estepas arbustivas, limitando las posibilidades económicas a la cría extensiva de ovinos [33].

En Chubut, las empresas ganaderas se especializan en la cría de la raza Merino Australiano, cuya lana posee una finura media provincial de 20.5 micras. Esta actividad se realiza principalmente en sistemas lanero-extensivos, aunque también hay una pequeña porción de sistemas mixtos ovino-vacuno en áreas como los valles, mallines y campos de precordillera en el Oeste y Sudoeste. A pesar de la existencia y la disponibilidad de nueva tecnología que ha surgido en la última década, estos sistemas tienen una baja adopción debido a factores como la falta de recursos financieros para invertir en tecnologías avanzadas y la escasez de mano de obra especializada en el sector. También, es relevante mencionar que la mayoría de los propietarios no residen en las propiedades, lo que puede generar una falta de involucramiento y control en la gestión del negocio, ya que el campo está a cargo de personal asalariado generalmente de baja calificación, en una densidad que varía entre los 1500 a 2500 ovejas por puesto de empleo permanente. Además, los productores pueden resistirse a adoptar nuevas tecnologías por falta de información y capacitación sobre su uso y beneficios, lo que puede generar una brecha entre el potencial y la realidad en términos de productividad y rentabilidad. Sin embargo, es importante destacar el impacto positivo del Programa PROLANA en la producción de lana en la provincia. Este programa, impulsado desde la Estación Experimental Agropecuaria Chubut del *Instituto Nacional de Tecnología Agropecuaria (INTA)*, ha tenido su origen en la región y ha logrado un gran desarrollo en la misma. Actualmente, PROLANA abarca casi las dos terceras partes de la producción provincial esquilada por máquinas habilitadas, lo que ha contribuido a mejorar la calidad y presentación de las lanas. Otro avance significativo ha sido la amplia adopción en la última década de la esquila preparto, que ha permitido mejorar aún más la calidad de la lana producida en la provincia [33].

Es importante mencionar que existe una ley ovina, la cual es una política de estado que proporciona recursos enfocados específicamente en alcanzar los objetivos de la cadena agroindustrial ovina nacional, así como también es una herramienta organizativa y financiera sectorial que tiene en cuenta las tendencias globales y los escenarios futuros [64].

2.3.3 Producción de Lana

La lana es una fibra natural de origen animal producida por diversas razas de ovinos en diferentes partes del mundo. Esta diversidad se manifiesta en un amplio rango de diámetros

de fibra y también en otras características de la misma: largo, color, etc. El diámetro de fibra se mide en micrómetros (micras, micrones), que equivale a 0.001 milímetros, y es el principal factor que determina qué productos textiles se pueden fabricar con cada tipo de lana, y en consecuencia su valor [64]. La lana debe ser conceptuada como una materia prima textil, regida por estrictas especificaciones, de ahí que el criador que produce lana «a medida» con buen manejo, eleva la calidad y, por ende, el valor de su zafra. Se calcula que cerca del 60% del stock ovino nacional es de raza Merino, comúnmente dan vellones más uniformes y con finuras promedio de 20.5 micrones, el cual es criado casi exclusivamente en la meseta central patagónica. La demanda mundial de lanas de 19.5 micras y más finas es por lejos la que más rápido ha crecido entre todas las fibras del mundo. También «sabiendo seleccionar» correctamente las razas doble propósito, se puede lograr vellones pesados de fibras uniformes en diámetro y largo. Todo indica que el éxito comercial de un lote de lana dependerá cada vez más de un alto nivel zootécnico en el manejo [34][46].

En relación al uso de la lana producida a nivel mundial, se han establecido estimaciones generales que se enfocan en dos categorías principales de utilización: la fabricación de prendas de vestir y el uso de lana en interiores. La lana producida a nivel mundial para uso en textiles de interiores supera en volumen a la lana utilizada en vestimenta. Los avances de la tecnología han ampliado el rango de productos que se pueden fabricar con lanas de diversos tipos, contribuyendo a borrar una buena parte de la identificación histórica de esta fibra con productos invernales, ásperos y gruesos. Frente al crecimiento en el consumo total de todas las fibras textiles, la producción mundial de lana ha disminuido, por lo cual el porcentaje de participación de la lana dentro del total de las fibras textiles es ahora menor que hace 10 años. Con un 1.1% de participación en el mercado total de fibras, es muy difícil conseguir niveles de inversión significativos en desarrollo de nuevos productos, en innovación y en nuevas tecnologías. Estos factores hacen que los textiles fabricados con lana tengan mayor dificultad para adaptarse a las exigencias cambiantes de los consumidores [64]. Se utilizó la información proporcionada por [64] para construir la Tabla 2.3.1, la cual clasifica los productos finales según el diámetro de la fibra presente en ellos. Sin embargo, es importante tener en cuenta que estos grupos no son compartimentos estancos y que existe cierta flexibilidad y migración entre ellos. Esto se debe a razones de precios y/o demandas específicas de los mercados. Por lo tanto, aunque la tabla proporciona una guía útil, es posible que los productos finales se superpongan entre los diferentes grupos debido a la naturaleza dinámica del mercado [64].

Tabla 2.3.1. Clasificación de los productos finales según Diámetro de Fibras.

Rango de Finura (micras)	Tipos de Productos Finales
< 18	Tejidos planos peinados ultralivianos.
	Tropicales; vestimenta formal de primavera/verano.
	Tejido de punto (sweaters) tipo Cashmere o Alpaca
	Vestimenta asociada a marcas de diseño de alta gama.
18.5 a 22	Tejidos peinados ligeros; gabardinas, tropicales, casimires.

	Mezclas lana/sintéticos especiales. Tejidos elastizados para ropa deportiva, o de uso interior liviana.
22.5 a 23.5	Tejidos peinados en pesos superiores a 280gr/m. Sargas, franelas finas. Faldas y chaquetas de pesos medios.
23.5 a 26.5	Tejidos peinados pesados; mezclas peinado/cardado.
	Tejido de punto en galgas medias
	Hilados para tejer a mano.
	Mezclas lana/sintético para múltiples usos.
	Vestimentas especiales para usos en actividades de alto impacto.
26.5 a 28.5	Abrigos tipo Loden o Melton (tejidos cardados de más de 400 gr/m.) para climas fríos.
	Tejidos cardados tipo Tweed.
	Tejido de punto en galgas gruesas (Shetland; double ribs)
	Tejidos cardados finos con superficie de pelo (Plush o velour), simulando falsa piel.
	Mantas y otros abrigos de cama (hilados cardados).
> 28.5	Tejidos para tapicerías, decoración de interiores.
	Recubrimiento de suelos (alfombras y moquetas)
	Filtros industriales y otros usos de textiles técnicos.
	Fieltros.

Las variaciones en la producción de lana responden al número de animales esquilados y la producción individual de estos y ambos valores a la variabilidad climática fundamentalmente. Como se trata de valores de lana sucia, el clima puede modificar también el rendimiento al lavado, reduciendo en alguna medida las diferencias entre años. El número de animales esquilados también puede ser afectado por condiciones económicas críticas como las que atravesó el sector durante la segunda parte de la década pasada [33].

La Argentina, a pesar de las crisis por las que ha atravesado la actividad, acredita una larga tradición ovejera. La meseta central patagónica, seriamente afectada por diferentes grados de desertificación y donde la receptividad ha caído de manera alarmante, sigue siendo la región más importante de producción de lanas finas. El mercado internacional de lanas es muy competitivo y exigente. Nuestro país, es tomador y no formador de precios, de modo que para tener protagonismo y lograr precios satisfactorios, es imperioso trabajar en forma integrada en todas las etapas de la producción y el procesamiento de la fibra [46].

2.3.4 Producción de Carne

La producción de carne ha sido un objetivo ocasional en los sistemas extensivos de producción de lana sujeta de manera directa a la variabilidad climática y su incidencia en la disponibilidad forrajera. Esto se traduce en una muy baja tasa de extracción de carne de la majada, determinada por bajas señaladas recurrentes y altas mortandades. [33].

En términos generales, los rebaños ovinos de la zona tienen un «doble propósito» es decir que pueden producir tanto lana como carne, en mayor o menor grado, variando algo en su aptitud cárnica las razas más tradicionales y sus cruza. El productor puede obtener un buen cordero preferido por la demanda, a los 4 meses de edad con 22 a 25 kilos de peso vivo. Es importante saber que el cordero nacido sobre el rebrote de primavera, tiene virtualmente el doble de ganancia de peso diaria comparado con un cordero nacido «a contramano», y supera el 50% de rinde en el gancho. [34].

El período alrededor del parto es, el más crítico en el ciclo anual de producción ovina, ya que en esta época se produce entre el 70% y 80% de las muertes en las ovejas y corderos, sumado a las pérdidas por abortos que pueden superar el 5%. El bajo porcentaje de señalada se debe a las siguientes causas:

- ❖ Ovejas y carneros subfértiles e infértiles
- ❖ Mortalidad embrionaria
- ❖ Alta mortalidad neonatal

Algunos estudios realizados por el INTA, tanto en la Patagonia como en la Provincia de Buenos Aires, demuestran que las dos causas primarias más frecuentes de muertes perinatales son el hambre y el frío [47]. Una herramienta gratuita y esencial a disposición del productor es la correcta elección de la época para el servicio donde, el otoño es la estación más propicia para el apareamiento y esto es válido para todas las razas ovinas en todo el mundo. Siendo más específicos, desde el punto de vista fisiológico, abril es el mes más positivo: mayor cantidad de ovejas en celo con un 10% a 20% de ovulaciones dobles (melliceras) y en orden descendente, abril, marzo y febrero son los meses más apropiados para el servicio en nuestra región geográfica. La rentabilidad en la producción animal está estrechamente ligada a la fertilidad, que debe ser alta y sostenida en el tiempo. Esto tiene dos beneficios importantes: primero, permite un progreso genético rápido debido a una mayor presión de selección; y segundo, genera más animales disponibles para la venta [48].

El empresario rural dirige una «fábrica sin techo», por así decirlo, hecho que le obliga a interpretar las vicisitudes ambientales con la mayor precisión posible. Aquí es donde entran en juegos las herramientas que proporcionen al empresario rural previsión y estimación de variables de interés. En resumen, el éxito comercial en la producción animal depende de la óptima combinación de factores ambientales, biológicos y climáticos que favorezcan el proceso reproductivo [48].

2.4 Inteligencia Artificial

La Inteligencia Artificial (IA) es una herramienta tecnológica innovadora que comienza a abandonar el laboratorio y los libros de ciencia ficción para convertirse en una realidad que

puede transformar de manera radical la forma en que conocemos un sinfín de aspectos cotidianos en nuestra sociedad [39]. La IA es una de las ramas de las ciencias de la computación que más interés ha despertado en la actualidad, debido a su enorme campo de aplicación. La búsqueda de mecanismos que nos ayuden a comprender la inteligencia, el pensamiento humano y el complejo hecho de realizar modelos o simulaciones de estos, es algo que ha motivado a muchos científicos a elegir esta área de investigación [35].

Desde su origen, la IA tuvo que lidiar con el conflicto de que no existía una definición clara y única de inteligencia, y es por esta razón que aún en la actualidad, no exista una definición única. Originalmente, se construyó e inspiró en base a conocimientos y teorías existentes en otras áreas del conocimiento; algunas de las principales fuentes son las ciencias de la computación, la filosofía, la lingüística, las matemáticas y la psicología. Cada una de estas ciencias cooperó no solamente con los conocimientos desarrollados en ellas, sino también con sus herramientas y experiencias contribuyendo así a la gestación y desarrollo de esta nueva área del conocimiento [35].

Para muchos campos de la ciencia, la IA es un concepto relativamente novedoso y para la ciencia animal puede ser fundamental para acelerar la transición de la mentalidad convencional, de enfocarse en la maximización de la producción, a una perspectiva más consciente que busque mejorar la eficiencia de la producción mientras se reducen los recursos asignados para tal fin. Combinar el modelado basado en conceptos con el modelado basado en datos, mediante la fusión paralela de modelos mecanicistas y de inteligencia artificial, dará lugar a una nueva forma de mecánica híbrida inteligente, que junto con la recopilación de datos mediante PLF, es primordial para superar la situación actual de la producción ganadera para lograr la sostenibilidad [13].

2.4.1 ¿Por qué inteligencia artificial?

El hombre se ha aplicado a sí mismo el nombre científico de hombre sabio (*homo sapiens*) como una valoración de la trascendencia de sus (nuestras) habilidades mentales, tanto para la vida cotidiana como para el propio sentido de identidad. Los esfuerzos de la Inteligencia Artificial, por su parte, se enfocan en lograr la comprensión de entidades inteligentes. Por ello, una de las razones de su estudio y análisis es aprender acerca de los propios seres humanos, pero a diferencia de la Filosofía y de la Psicología (que también se ocupan de la inteligencia), los esfuerzos de la IA están encaminados tanto a la construcción de entidades inteligentes, como a su comprensión [49].

Otra razón más por la que se estudia la IA es porque las entidades inteligentes así construidas son interesantes y útiles. No obstante, a pesar de las fases tempranas en las que todavía se encuentra, mediante la IA ha sido posible crear diversos productos sorprendentes y de trascendencia. Si bien es imposible pronosticar con precisión lo que se puede esperar de esta disciplina en el futuro, es evidente que las computadoras que posean una inteligencia a nivel humano o superior, tendrán repercusiones importantes en la vida diaria, así como en el devenir de la civilización [49].

Inteligencia Artificial es una acepción acuñada a mediados del siglo XX, cuyo desarrollo se ha caracterizado por una sucesión de períodos alternativos de éxito y abandono de la misma. Al principio, la idea intuitiva de la IA creó expectativas que no siempre han sido cubiertas, y

desde luego no en el grado que se había esperado, de manera un tanto ilusoria. Pero actualmente se puede considerar que el enfoque computacional inteligente no depende de inmediatos y probados resultados, sino que está avalado por sus logros y su desarrollo a lo largo de varias décadas, por lo cual se ha consolidado en el ámbito de la computación como una aceptación totalmente asumida, aunque sometida todavía a controversia en algunos sectores científicos [49].

De hecho, el problema que aborda la IA es uno de los más complejos: ¿cómo es posible que un diminuto cerebro (sea biológico o electrónico), tenga capacidad para percibir, comprender, predecir y manipular un mundo que en tamaño y complejidad lo excede? [49] La IA siempre ha tenido como modelo natural las funcionalidades inteligentes del hombre, enfocándose en distintos aspectos. Su primera motivación fue intentar construir máquinas que pudieran pensar como el ser humano, o al menos emularle en alguna capacidad de tal modo que denotará cierta inteligencia [49].

La IA es una de las disciplinas más nuevas y ha resultado ser mucho más compleja de lo que inicialmente se pensó ya que las ideas modernas que se tienen de ella se caracterizan por su gran riqueza, sutilidad y por lo interesantes que son [49].

A continuación definiremos lo que es la IA.

2.4.2 Definición

Si deconstruimos el concepto de Inteligencia Artificial, nos encontramos con dos palabras. Por un lado la **inteligencia**, que está vinculada a saber elegir las mejores opciones para resolver algún tipo de problema; existen diversos tipos de inteligencia según sus atributos y procesos, como la inteligencia operativa, la inteligencia biológica o la inteligencia psicológica. Por otra parte la palabra **artificial**, es un adjetivo que señala aquello hecho por mano, arte o ingenio del hombre [36].

La noción de inteligencia artificial fue desarrollada en referencia a sistemas o máquinas creados por los seres humanos que imitan la inteligencia humana para realizar tareas, y pueden mejorar iterativamente a partir de la información que recopilan; constituyendo así agentes racionales no vivos [36][38]. La racionalidad, en este caso, es interpretada como la capacidad para maximizar un resultado esperado y se manifiesta de varias formas, como [36]:

- ❖ Los chatbots, que utilizan la IA para comprender más rápido los problemas de los clientes y proporcionar respuestas más eficientes.
- ❖ Los asistentes inteligentes para programadores, que utilizan la IA para analizar información crítica proveniente de grandes conjuntos de datos de texto libre para mejorar la escritura del código fuente.
- ❖ Los motores de recomendación, que utilizan la IA para proporcionar recomendaciones automatizadas, por ejemplo, para programas de TV según los hábitos de visualización de los usuarios.

Los dispositivos que cuentan con inteligencia artificial poseen la capacidad de ejecutar distintos procesos análogos al comportamiento humano, como la devolución de una respuesta por cada entrada (similar a los actos reflejos de los seres vivos), la búsqueda de un estado entre todos los posibles según una acción específica o la resolución de un problema mediante el seguimiento de una lógica formal [36].

La IA trata mucho más sobre el proceso, la capacidad de pensamiento y el análisis de datos, que sobre cualquier formato o función en particular siendo su objetivo mejorar significativamente las capacidades y contribuciones humanas, lo que la convierte en un activo empresarial muy valioso [38].

Existen dos aproximaciones fundamentales dentro del amplio espectro de la Inteligencia Artificial. Una es la Inteligencia Artificial simbólica, la cual se caracteriza por un alto nivel de abstracción y vista microscópica. A esta categoría pertenecen la psicología clásica, los sistemas basados en el conocimiento, el aprendizaje simbólico de máquina, técnicas de búsqueda, y el procesamiento de lenguaje natural. La segunda aproximación se caracteriza por un bajo nivel de abstracción y modelos biológicos microscópicos. Las redes neuronales y los algoritmos genéticos se encuentran en esta categoría. Además de estas dos aproximaciones de la Inteligencia Artificial, se han desarrollado nuevas herramientas denominadas «inteligencia computacional», las cuales tienen aplicación en la gestión financiera empresarial, especialmente en el tratamiento de la información imprecisa. Dentro de estas técnicas de reciente implementación se pueden mencionar: los sistemas borrosos (fuzzy systems), la teoría de los conjuntos aproximados (rough sets theory), y sistemas caóticos (systems chaotics). Los sistemas borrosos y la teoría de los conjuntos aproximados se pueden emplear con las técnicas de la Inteligencia Artificial simbólica [37].

2.4.3 Ramas de la Inteligencia Artificial

El ecosistema de la Inteligencia Artificial actualmente se encuentra en un constante flujo de aprendizaje, cambios y crecimiento aunque, en términos generales, se puede separar en 3 ramas o categorías principales: IA, ML y DL como se muestra en la figura 2.4.1.

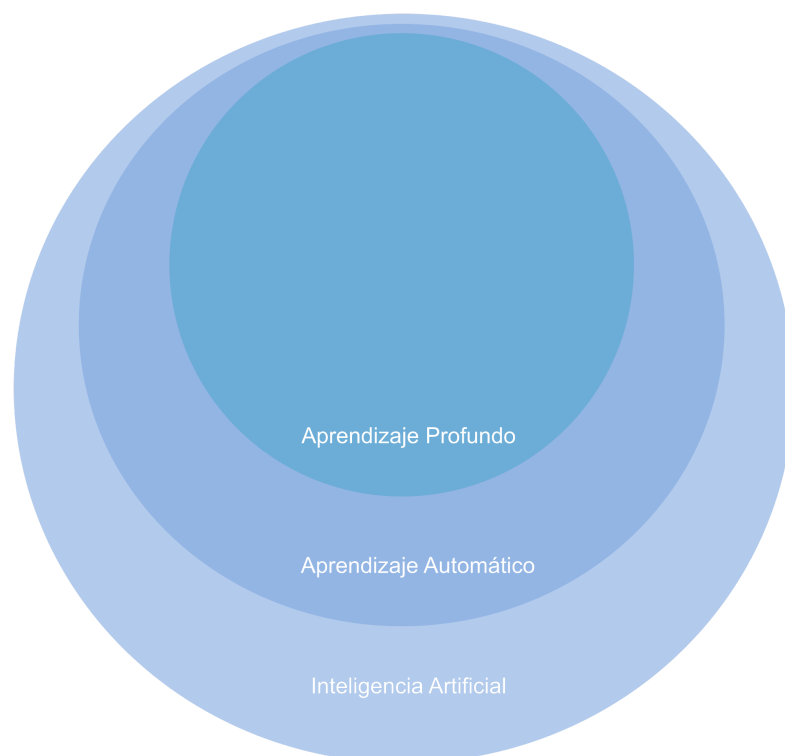


Figura 2.4.1. Diagrama de Venn ramas generales de IA.

Existen otras maneras más específicas de clasificar a las diferentes ramas de la inteligencia artificial [23]:

- ❖ Por tipo de supervisión: se puede clasificar en aprendizaje supervisado, semi supervisado o automático.
- ❖ Por función de inteligencia: se puede separar en inteligencia general o, si es de propósito específico, se llama inteligencia estrecha o específica.
- ❖ Por función humana: Visión máquina (o visión por computadora), procesamiento de lenguaje natural, aprendizaje máquina y reconocimiento de patrones.
- ❖ Por técnicas de búsqueda: Estas técnicas son muy utilizadas en IA. Estos programas examinan todas las posibles soluciones y eligen la mejor. Este tipo de técnicas se han utilizado mucho en juegos de estrategia, como ajedrez, Go, así como también como solución a problemas de logística, redes computacionales, entre otros.
- ❖ Basados en lógica: Estos algoritmos de IA básicamente son un conjunto de declaraciones que expresan hechos y reglas en relación con el dominio de un problema.
- ❖ Planeación: Esta rama tiene que ver con la planeación óptima de un objetivo. Cuando se conoce el objetivo y el entorno, se puede llegar a la optimización de los recursos de tal forma que se maximice la ganancia con el costo mínimo.
- ❖ Heurística: Son técnicas utilizadas para resolver un problema dado, lo cual es práctico y útil para resolver en el menor tiempo. Esto es debido a que se pueden buscar estrategias para resolver un problema sin que se asegure un óptimo, como es el caso de la planeación.
- ❖ Programación genética: La programación genética es una manera de hacer que los programas puedan resolver una tarea imitando la genética y evolución de las especies. Los programas se codifican a manera de cromosomas (los cuales se componen de genes) y usan un algoritmo que pueda realizar una tarea específica, de tal forma que cada generación de individuos codificados en cromosomas ofrezcan mejores individuos, es decir, mejores soluciones.
- ❖ Inteligencia basada en enjambre: Esta rama de la IA toma de ejemplo cómo se comportan algunas especies de manera colectiva para resolver un problema en específico. Por ejemplo, la ya conocida aplicación del problema del viajero frecuente (TSP) puede ser resuelto mediante el comportamiento de las hormigas (conocido como ant colony o ACO) para encontrar la ruta más corta de su nido a una fuente de comida.

2.4.4 Áreas de la Inteligencia Artificial

En la actualidad existe una gran cantidad de áreas específicas en las que se trabaja bajo el concepto de inteligencia artificial, cada una de estas cuenta con herramientas y/o técnicas propias para lograr sus objetivos, algunas de estas se enlistan en la Tabla 2.4.1 [35].

Tabla 2.4.1. Tópicos de la Inteligencia Artificial.

Generación del Lenguaje Natural	Razonamiento Basado en Casos	Procesamiento del Lenguaje Natural
Reconocimiento de Imágenes	Algoritmos Genéticos	Computación Evolutiva
Reconocimiento de Patrones	Programación Lógica	Redes Neuronales
Sistemas Tutores Inteligentes	Interfaces Inteligentes	Aprendizaje Máquina
Ontologías	Optimización Multiobjetivo	Sistemas Híbridos Inteligentes
Lógica Difusa	Programación Evolutiva	Realidad Aumentada
Vida Artificial	Algoritmos Bio-inspirados	Teoría de Automatas
Asistentes digitales virtuales	Chatbot	Minería de Datos
IA Distribuida y Sistemas Multiagentes	Sistemas Expertos y Sistemas Basados en Conocimiento	Representación y Administración del Conocimiento

Dentro de la IA un área que ha ganado relevancia en las últimas dos décadas es el desarrollo de agentes. Esto ha surgido por un lado, por la necesidad de abordar problemas complejos que son muchas veces difíciles de tratar por un sistema monolítico o que son de naturaleza distribuida, ya sea porque los datos o los procesos están distribuidos. Por otro lado, hay muchos autores donde consideran que ciertos comportamientos inteligentes más que ser modelados como conductas complejas individuales, pueden surgir de la interacción de conductas simples [35].

La figura 2.4.2 muestra cómo podemos clasificar algunas de las anteriores áreas mencionadas dentro de las diferentes ramas de la Inteligencia Artificial.

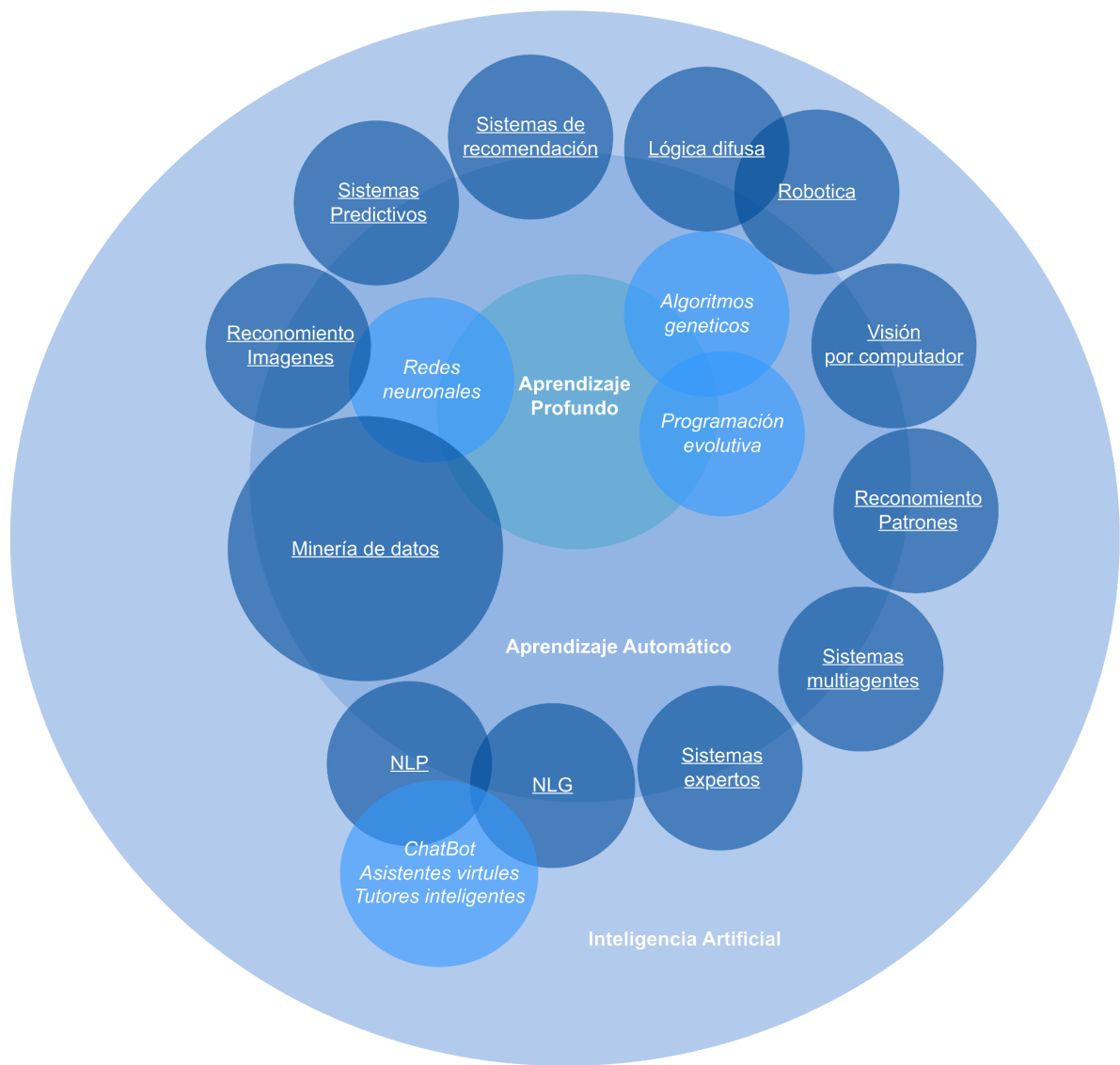


Figura 2.4.2. Diagrama de Venn ramas específicas de IA.

Podemos ver que la minería de datos es un área de la inteligencia artificial que se extiende por el aprendizaje automático, el aprendizaje profundo y se emplea en las redes neuronales. También podemos notar que los Chatbots, Asistentes Virtuales y Tutores Inteligentes son tipos de inteligencia artificial que se basan en el procesamiento y generación de lenguaje natural, los cuales se encuentran en el aprendizaje automático. Los sistemas expertos, multiagentes, predictivos y de recomendación se encuentran en el área intermedia entre la IA y el ML, ya que dependen de este último para poder aprender, al igual que la visión por computadora y el reconocimiento de patrones. El reconocimiento de imágenes requiere del uso de redes neuronales, las cuales se encuentran en la rama del aprendizaje profundo, al igual que la programación evolutiva, la computación evolutiva y los algoritmos genéticos. Por último, podemos observar que la robótica y la lógica difusa son tipos de inteligencia artificial que están interconectados y se encuentran en el ámbito del aprendizaje automático.

Mencionadas las distintas áreas y ramas de la inteligencia artificial comenzaremos la siguiente sección basando el enfoque en una de las ramas con mayor auge actual de la IA: Machine Learning; brindando detalles sobre su origen, características y conceptos claves para su desarrollo e implementación.

2.5 Machine Learning

Durante muchos años, los humanos hemos buscado detalles estructurales en la información, tales como reglas o patrones con los que describir, con precisión, la manera en cómo funcionan los sistemas importantes que nos rodean, sean estos sistemas agrícolas, biológicos, físicos, financieros u otros. Estos detalles estructurales de la información permiten comprender mejor un sistema, predecir con precisión su comportamiento futuro y, en última instancia, manipularlo. No obstante, a lo largo de la historia, ha resultado desafiante encontrar la estructura idónea que describa un sistema específico. Incluso en tiempos más recientes, los datos disponibles que registren las características emitidas por dicho sistema han sido sumamente limitados. Además, la capacidad para calcular, para probar varios métodos y verificar cuál presenta un mejor funcionamiento se ha limitado a lo que se podía lograr con calculadoras o equipos con escasos recursos informáticos. Ambos factores limitaron la gama de problemas que, en el pasado, se pudieron investigar y los forzaron, inevitablemente, a utilizar enfoques filosóficos y/o visuales para la extracción de conocimiento a partir de la información. Sin embargo, hoy en día gracias a los avances de la informática tenemos un mundo inundado de datos (derivado de la digitalización de gran parte de las cosas que hacemos y nos rodean), y contamos con un poder informático de enormes dimensiones al alcance de la mano para almacenar y procesar grandes cantidades de datos (básicamente, porque los ordenadores y sus procesadores cuestan mucho menos que antes, lo que produce un abaratamiento de almacenamiento). Bajo estas condiciones se puede abordar actualmente una gran gama de problemas mucho más compleja, y adoptar un enfoque mucho más empírico para la extracción de información que en el pasado, y es en este contexto que aparece el concepto de machine learning (ML), denominado también aprendizaje de máquina o automático, que corresponde a programas computacionales que buscan optimizar los parámetros de un modelo usando datos [35][40][42].

2.5.1 ¿Qué es el aprendizaje automático?

El aprendizaje automático es un subcampo de la inteligencia artificial. Este ayuda a las máquinas a aprender e interaccionar como seres humanos con la ayuda de algoritmos y datos intentado extraer todo el conocimiento posible. Dado un conjunto de datos, un algoritmo de machine learning aprende diferentes propiedades de ellos e infiere las propiedades de los mismos que se pueden presentar en el futuro. Esta ciencia del aprendizaje es un campo de investigación en la intersección de la estadística, la inteligencia artificial y la informática, y también se conoce como análisis predictivo o aprendizaje estadístico. A partir de esta definición, se puede concluir que el objetivo del aprendizaje máquina es desarrollar sistemas que permitan a los ordenadores aprender y generalizar comportamientos [40][41].

A lo largo de los últimos años, la aplicación de métodos de aprendizaje automático se ha hecho omnipresente en la vida cotidiana. Desde recomendaciones automáticas de películas, comidas y productos hasta radio en línea personalizada y reconocimiento de amigos en fotos, muchos sitios web y dispositivos modernos utilizan algoritmos de aprendizaje automático en su núcleo. Cuando se observa un sitio web complejo como Facebook, Amazon o Netflix, es muy probable que cada parte del sitio incluya múltiples modelos de aprendizaje automático [41].

Fuera de las aplicaciones comerciales, el aprendizaje automático ha tenido una gran influencia en la forma en que se realiza la investigación basada en datos debido a que las herramientas utilizadas en machine learning se han aplicado a diversos problemas científicos:

- ❖ en Astronomía, para la comprensión de estrellas o la búsqueda de planetas distantes,
- ❖ en Física, para el descubrimiento de nuevas partículas,
- ❖ en Medicina, como auxiliar en el diagnóstico de diversas patologías, para el análisis de secuencias de ADN y la provisión de tratamientos personalizados contra el cáncer;
- ❖ en sistemas financieros, analizando el mercado de valores y detectando fraudes en el uso de tarjetas de crédito; o
- ❖ en informática, aplicado en sistemas de reconocimiento de habla, lenguaje escrito, entre otras.

Sin embargo, una aplicación no necesita ser tan grande o cambiar el mundo para beneficiarse del aprendizaje automático [40][41].

2.5.2 ¿Por qué aprendizaje automático?

En los primeros tiempos de las aplicaciones «inteligentes», muchos sistemas utilizaban reglas codificadas a mano de decisiones «sí» y «no» para procesar datos o ajustarse a las entradas del usuario. La creación manual de reglas de decisión es factible para algunas aplicaciones, sobre todo para aquellas en las que los humanos tienen un buen conocimiento del proceso a modelar. Sin embargo, utilizar reglas codificadas a mano para tomar decisiones tiene dos grandes desventajas [40]:

- ❖ La lógica necesaria para tomar una decisión es específica de un dominio y una tarea concreta. Cambiar la tarea, aunque sea ligeramente, puede requerir reescribir todo el sistema.
- ❖ El diseño de reglas requiere una comprensión profunda de cómo debe tomar una decisión un experto humano.

Un ejemplo de dónde falla este enfoque manual es en la detección de caras en imágenes. Hoy en día, cualquier smartphone puede detectar una cara en una imagen. Sin embargo, la detección de rostros era un problema sin resolver hasta 2001. El principal problema es que la forma en que los píxeles son «percibidos» por el ordenador es muy diferente de cómo los humanos percibimos una cara. Esta diferencia en la representación hace que sea básicamente imposible para un ser humano dar con un buen conjunto de reglas para describir lo que constituye una cara en una imagen digital. Con el aprendizaje automático, sin embargo, basta con presentar a un programa una gran colección de imágenes de caras para que un algoritmo determine qué características son necesarias para identificar un rostro [40].

Existen conceptos clave en el aprendizaje máquina que sientan las bases para comprender este campo. Estos pueden dividirse en dos áreas sustanciales: los conceptos sobre datos y los conceptos sobre aprendizaje. Los conceptos sobre datos otorgan la nomenclatura apropiada para describir los datos y sus conjuntos. Los conceptos sobre aprendizaje describen el aprendizaje obtenido a partir de los datos [40].

2.5.3 Conceptos sobre datos

Como se mencionó anteriormente, los métodos de aprendizaje máquina buscan aprender a partir de los ejemplos y es importante tener una representación de los datos de entrada que un ordenador pueda entender. Los datos pueden estructurarse en filas y columnas, como una tabla de base de datos o como una hoja de cálculo; estos son conocidos como «estructura tradicional de datos», y son comunes en el campo del aprendizaje máquina. Cada dato sobre el que se quiere razonar (por ejemplo, cada cliente, cada transacción) es una fila, y cada propiedad que describe ese dato (por ejemplo, la edad de un cliente o el importe o la ubicación de una transacción) es una columna [40][41].

Existe una variada terminología utilizada al describir los datos de entrada; los conceptos básicos para datos del aprendizaje máquina se definen a continuación [40]:

- ❖ Observación: Es la entidad más pequeña, con propiedades de interés para un estudio que puede ser registrado. Por ejemplo, un cliente de una tienda online.
- ❖ Características: Son las propiedades o atributos de las observaciones que pueden ser útiles para el aprendizaje. Por ejemplo, podemos describir a los usuarios por su edad, su sexo, cuándo crearon la cuenta y con qué frecuencia compran en una tienda online.
- ❖ Tipo de datos: Las características tienen un tipo de datos. Estos pueden ser de valor real o entero, o pueden tener un valor categórico u ordinal. Por ejemplo, la edad (25 años) sería un entero, el sexo (F o M) sería categorico.
- ❖ Conjuntos de datos: Una colección de observaciones es un conjunto de datos y, cuando se trabaja con métodos de aprendizaje máquina, generalmente se requieren algunos conjuntos de datos para diferentes propósitos. Por ejemplo, todos los clientes de una tienda online.
- ❖ Datos de entrenamiento: Conformen un conjunto de datos que se incorpora al algoritmo de aprendizaje máquina para entrenar al modelo.
- ❖ Datos de prueba: Constituyen un conjunto de datos utilizado para validar la precisión del modelo, pero que no se emplea para entrenar al modelo.

Además de los datos mencionados con anterioridad, existen otros, como imágenes, vídeos y texto. Estos son llamados «datos no estructurados», y para poder ser aplicados a métodos de aprendizaje máquina, deben ser transformados a una forma estructurada de datos. Se podría describir la imagen de un tumor por los valores en escala de grises de cada píxel, o tal vez utilizando el tamaño, la forma y el color del tumor [40][41].

2.5.4 Conceptos sobre aprendizaje

El aprendizaje máquina se basa en el aprendizaje con algoritmos; algunos conceptos básicos y de alto nivel sobre aprendizaje se definen a continuación [40]:

- ❖ **Inducción:** La inducción es un proceso de razonamiento utilizado por los algoritmos de aprendizaje máquina para aprender de los datos de entrenamiento. Durante este proceso, se construye un modelo a partir de los datos, lo que permite al algoritmo hacer predicciones y tomar decisiones informadas.
- ❖ **Generalización:** La generalización es un proceso clave en el aprendizaje automático, ya que se busca encontrar un patrón o modelo que sea significativo y representativo de los datos de entrenamiento. Una vez que se ha identificado este modelo, se pueden hacer predicciones o tomar decisiones basadas en él. El objetivo final es que este modelo pueda aplicarse a nuevos conjuntos de datos no vistos durante el entrenamiento y aún así obtener resultados precisos y fiables.
- ❖ **Sobreentrenamiento:** El sobreentrenamiento es un fenómeno que se produce cuando un modelo de aprendizaje automático aprende los datos de entrenamiento de forma demasiado precisa y pierde la capacidad de generalización, es decir, no puede aplicar el conocimiento adquirido a nuevos datos. Este problema se conoce también como «overfitting» u «overfit», ya que el modelo «se ajusta demasiado» a los datos de entrenamiento, perdiendo la capacidad de hacer predicciones precisas en datos nuevos o desconocidos.
- ❖ **Subentrenamiento:** El subentrenamiento se produce cuando un modelo no ha aprendido lo suficiente sobre la estructura de los datos y, como resultado, tiene un rendimiento deficiente en la mayoría de los datos, incluido el conjunto de entrenamiento. Aunque la generalización es buena, el modelo subentrenado no puede ajustarse a los datos de entrenamiento y, por lo tanto, no puede capturar toda la variabilidad en los datos. Este fenómeno también se conoce como «underfit» o «underfitting».
- ❖ **Aprendizaje en línea:** El aprendizaje en línea es una técnica de aprendizaje automático que se basa en la alimentación constante de los algoritmos de aprendizaje con nuevos datos a medida que están disponibles. Para ser efectivo, el aprendizaje en línea debe ser robusto para tratar con datos ruidosos y puede producir modelos más actualizados y precisos al estar alineados con el estado actual del conjunto de datos del tema en cuestión.
- ❖ **Aprendizaje fuera de línea:** También conocido como «aprendizaje batch», se refiere a un enfoque de entrenamiento de modelos de Machine Learning donde el modelo se entrena por lotes con un conjunto fijo de datos de entrenamiento. Es decir, el modelo se entrena con todos los datos de entrenamiento disponibles de una sola vez y luego se utiliza para hacer predicciones. A diferencia del aprendizaje en línea, no hay actualizaciones continuas del modelo a medida que se reciben nuevos datos.

Después de establecer los fundamentos de los datos y el aprendizaje, es importante conocer los distintos tipos de problemas y soluciones que existen en el campo del aprendizaje

máquina, así como los tipos de datos y aprendizaje que se utilizan en el desarrollo de los algoritmos de ML.

2.5.5 Tipos de datos

Para realizar un análisis de datos y modelado predictivo eficiente, es importante conocer el tipo de datos involucrados, para así determinar la mejor visualización, el análisis de datos o el modelo estadístico adecuado. Los datos se pueden clasificar en cuatro tipos principales: continuos, discretos, categóricos, binarios y ordinales.

- ❖ Los datos continuos son aquellos que pueden tomar cualquier valor dentro de un intervalo; por ejemplo, el volumen de un líquido, la temperatura, el peso de una persona o la cantidad de tiempo que un usuario pasa en una página web.
- ❖ Los datos discretos, por su parte, se limitan a valores enteros; como el número de personas en una habitación, el número de hijos que tiene una persona, la cantidad de unidades vendidas en un mes o la cantidad de veces que un anuncio se ha mostrado en una página web.
- ❖ Los datos categóricos se refieren a valores que solo se pueden tomar de un conjunto específico de categorías; por ejemplo, el color de un objeto, el tipo de comida que se sirve en un restaurante (italiana, mexicana, vegetariana), el color de los ojos de una persona o el modelo de un coche.
- ❖ Los datos binarios son un caso especial de los datos categóricos, con solo dos categorías: verdadero o falso. Por ejemplo, sí/no, verdadero/falso, comprado/no comprado, etc.
- ❖ Por último, los datos ordinales son datos categóricos con un orden explícito; como la calificación numérica de un examen (de 1 a 5), la escala de calificación de un hotel (1 estrella, 2 estrellas, 3 estrellas), la clasificación de un producto por calidad (malo, regular, bueno, excelente) o el nivel de educación (primaria, secundaria, universitaria).

2.5.6 Tipos de problemas

Existen varias clases comunes de problemas en el aprendizaje máquina. Las clases de problemas que se mencionan a continuación son arquetipos para la mayoría de los problemas, a los que nos referimos cuando implementamos el aprendizaje máquina [40]:

Clasificación

En la clasificación, el objetivo es predecir una etiqueta de clase, que es una elección de una lista predefinida de posibilidades; es decir, los datos tienen una etiqueta, lo que significa que se les asignó una clase. La decisión que se modela consiste en asignar etiquetas a nuevos datos no etiquetados. Esto se puede entender como la tarea de generar un discriminante que modele las similitudes o diferencias entre grupos. La clasificación a veces se divide en clasificación binaria, que es el caso especial de distinguir entre exactamente dos clases, y clasificación multiclase, que es la clasificación entre más de dos clases. Se puede pensar la clasificación binaria como si se tratara de responder una pregunta de sí/no. La clasificación de correos electrónicos como spam o no spam es un ejemplo de un problema de clasificación

binaria. Un ejemplo de un problema de clasificación multiclase sería predecir en qué idioma está un sitio web a partir del texto contenido dentro del sitio web donde las clases serían una lista predefinida de posibles idiomas [40][41].

Regresión

Para las tareas de regresión, el objetivo es predecir un número continuo. Los datos se encuentran etiquetados con un valor real en lugar de una etiqueta. La decisión que se modela es la relación entre entradas y salidas. Predecir el ingreso anual de una persona a partir de su educación, su edad y el lugar donde vive es un ejemplo de una tarea de regresión. Al predecir los ingresos, el valor previsto es una cantidad y puede ser cualquier número en un rango determinado. Otro ejemplo de una tarea de regresión podría ser predecir el rendimiento de una granja de maíz dados atributos como rendimientos anteriores, clima y cantidad de empleados que trabajan en la granja. El rendimiento nuevamente puede ser un número arbitrario [40][41].

Agrupamiento

En el agrupamiento, el objetivo es dividir los datos en grupos según la similitud y otras medidas de estructura natural en los datos, pero sin etiquetas predefinidas. La decisión que se modela consiste en encontrar patrones y estructuras dentro de los datos, y asignar cada punto de datos a un grupo determinado. El agrupamiento a veces se divide en agrupamiento jerárquico y agrupamiento no jerárquico. En el agrupamiento jerárquico, los datos se agrupan en un árbol jerárquico, mientras que en el agrupamiento no jerárquico, los grupos se forman de manera plana. Un ejemplo de agrupamiento podría ser la segmentación de clientes en grupos con datos demográficos similares para identificar patrones de comportamiento y adaptar campañas de marketing a cada grupo. Otra aplicación podría ser la identificación de grupos de pacientes con características similares en los registros médicos para mejorar la atención médica y la toma de decisiones clínicas [40][51].

Extracción de reglas

En la extracción de reglas, se busca descubrir patrones y relaciones ocultas en los datos mediante la generación de reglas proposicionales que describen las relaciones entre los diferentes atributos del conjunto de datos. Estas reglas pueden ser enunciadas en forma de antecedente/consecuente o si/entonces y su objetivo es proporcionar una explicación de las correlaciones encontradas en los datos. Un ejemplo es el descubrimiento de patrones de compra: en un conjunto de datos de transacciones de clientes en una tienda, se pueden extraer reglas como «si un cliente compra un paquete de pañales y una botella de leche, entonces es probable que también compre una caja de cereal» o «si un cliente compra pañales, entonces es probable que también compre cerveza» [40][51].

2.5.7 Tipos de soluciones

Las categorías anteriores nos permiten caracterizar el tipo de problema a afrontar. Y en cuanto a soluciones, los algoritmos de machine learning, se pueden agrupar en tres grupos [42]: Modelos lineales, modelos de árbol y redes neuronales. Los modelos pueden ser

inductivos, cuando permiten hacer predicciones sobre el futuro o bien descriptivos cuando permiten generar conocimiento a partir de los datos [35].

Modelos lineales

Los modelos lineales son una clase de modelos que se utilizan ampliamente en la práctica y se han estudiado mucho en las últimas décadas, con raíces que se remontan a más de cien años. Trata de encontrar una línea que se «ajuste bien» a la nube de puntos que se disponen. Aquí destacan modelos muy conocidos y usados como la regresión lineal (también conocida como la regresión de mínimos cuadrados), y la regresión logística (adaptación de la regresión lineal a problemas de clasificación cuando son variables discretas o categóricas). Estos dos modelos tienen el problema del «overfit» y al ser modelos relativamente simples, no ofrecen resultados muy buenos para comportamientos muy complicados, pero de todas formas pueden llegar a ser muy potentes. Los modelos lineales hacen una predicción utilizando una función lineal de las características de los datos de entrada [41][42].

Modelos lineales para regresión

Para la regresión, la fórmula de predicción general para un modelo lineal es la siguiente:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \dots + w[p] * x[p] + b$$

Aquí, $x[0]$ a $x[p]$ denota las características de un solo punto de datos (en este ejemplo, el número de características presentes en los datos es p), w y b son parámetros del modelo que se aprenden, e $\hat{y}$ es la predicción que el modelo hace. Para un conjunto de datos con una sola característica, esto es:

$$\hat{y} = w[0] * x[0] + b$$

denominada también como la ecuación de una línea.

Aquí, $w[0]$ es la pendiente y b es el desplazamiento del eje y . Para más características, w contiene las pendientes a lo largo de cada eje de características. Alternativamente, se puede pensar en la respuesta pronosticada como una suma ponderada de las características de entrada, con pesos (que pueden ser negativos) dados por las entradas de w .

Para ilustrar el algoritmo de regresión, utilizaremos un conjunto de datos de ondas sintéticas. Este conjunto de datos tiene una característica de entrada única y una variable objetivo continua (o respuesta) que queremos modelar. La figura 2.5.1 [41] presenta una gráfica que muestra una característica única en el eje x y el objetivo de regresión en el eje y , lo que permite visualizar de manera clara la relación entre estas dos variables.

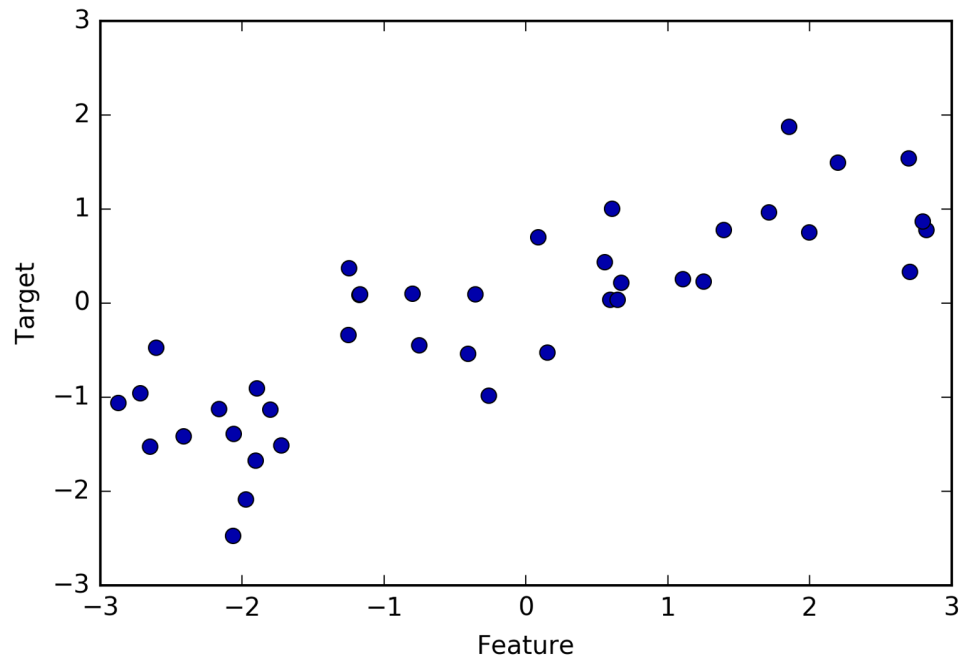


Figura 2.5.1. Gráfico del conjunto de datos de ondas, con el eje X que muestra la característica y el eje Y que muestra el objetivo de regresión.

Tratar de aprender los parámetros $w[0]$ y b en un conjunto de datos unidimensionales como el de ondas podría conducir a una línea como la de la figura 2.5.2 [41]:

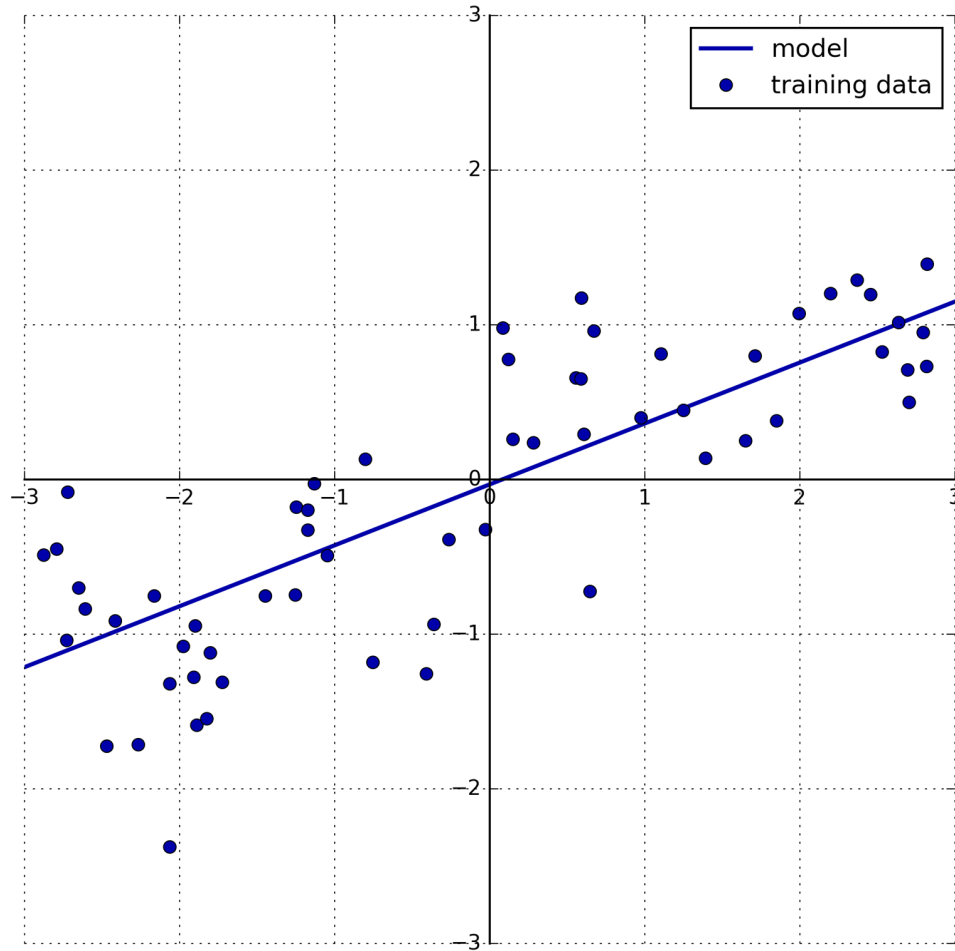


Figura 2.5.2. Predicciones de un modelo lineal en el conjunto de datos de ondas.

Mirando $w[0]$ vemos que la pendiente debería estar alrededor de 0.4, lo cual podemos confirmar visualmente en la gráfica. La intersección es donde la línea de predicción debe cruzar el eje y; esto es ligeramente por debajo de cero, lo que también se puede confirmar en la imagen. Los modelos lineales para regresión se pueden caracterizar como modelos para los cuales la predicción es una línea para una sola característica, un plano cuando se usan dos características o un hiperplano en dimensiones más altas (es decir, cuando se usan más características) [41].

La idea de utilizar una línea recta para hacer predicciones puede parecer demasiado limitante, ya que parece que todos los detalles finos de los datos se perderán. En cierta forma, esto es verdad; es una suposición fuerte (y algo poco realista) que nuestro objetivo $\hat{y}$ sea una combinación lineal de las características. No obstante, observar los datos de forma unidimensional puede proporcionar una perspectiva parcial. Para conjuntos de datos con muchas características, los modelos lineales pueden ser extremadamente poderosos; especialmente si hay más características que puntos de datos de entrenamiento, en cuyo caso cualquier objetivo $\hat{y}$ se puede modelar de manera perfecta (en el conjunto de entrenamiento) como una función lineal [41].

Sobreajuste en modelos lineales

Existen muchos modelos lineales diferentes para la regresión. La diferencia entre estos modelos radica en cómo se aprenden los parámetros w y b del modelo a partir de los datos de entrenamiento y cómo se puede controlar la complejidad del modelo. La regresión lineal es una técnica utilizada para encontrar los valores de los parámetros que minimizan el error cuadrático medio entre las predicciones y los valores reales en un conjunto de datos de entrenamiento. A diferencia de otros métodos, la regresión lineal no tiene parámetros para controlar la complejidad del modelo, volviéndose más poderoso a medida en que el conjunto de datos se vuelve más grande. Sin embargo, esta mayor potencia puede llevar a un problema de sobreajuste, donde el modelo se «aprende» demasiado los datos de entrenamiento y no generaliza bien a nuevos datos. Por lo tanto, es importante considerar la existencia de sobreajuste al utilizar la regresión lineal en conjuntos de datos de alta dimensionalidad [41].

Para evitar esta situación existen numerosas técnicas de regulación sobre los modelos lineales, de las cuales en el desarrollo de este trabajo emplearemos: la regresión de Ridge y la regresión de Lasso. La regresión de cresta o Ridge Regression es un método de regularización que utiliza una función de costo que agrega una penalización por el tamaño de los coeficientes de regresión. Esto significa que los coeficientes de regresión se reducen para evitar el sobreajuste, lo que puede llevar a una mejor generalización del modelo a nuevos datos. La idea detrás de la regresión de cresta es agregar un término de regularización a la función de costo de la regresión lineal ordinaria, que penaliza los coeficientes de regresión más grandes. En este modelo, la cantidad de penalización se controla mediante un parámetro de ajuste llamado λ , que puede variar entre cero (lo que significa que no hay penalización) y un valor muy grande (lo que significa que todos los coeficientes de regresión se reducirán a cero). Este parámetro de ajuste λ se selecciona mediante una técnica llamada validación cruzada. La regresión de Lasso, por otro lado, es una técnica de regularización similar a la regresión de cresta, pero utiliza una penalización diferente, en lugar de agregar una penalización cuadrática, utiliza una penalización de valor absoluto. Esto tiene el efecto de reducir algunos coeficientes de regresión a cero, lo que significa que algunas de las variables predictoras se eliminan del modelo por completo. La regresión Lasso es útil para seleccionar características importantes en un conjunto de datos [41].

En resumen, tanto la regresión de cresta como la regresión Lasso son técnicas de regularización que se utilizan para evitar el sobreajuste en modelos de regresión lineal. Ambas técnicas son útiles cuando hay una gran cantidad de variables predictoras o cuando las variables predictoras están altamente correlacionadas.

Modelos de árbol

Son modelos precisos, estables y más sencillos de interpretar básicamente porque se construyen reglas de decisión que se pueden representar como un árbol generando una mejor visualización y de esta manera, la comprensión del modelo simplificando sus características funcionales. A diferencia de los modelos lineales, pueden representar relaciones no lineales para resolver problemas y al ser más precisos y elaborados, se gana en capacidad predictiva, pero se pierde en rendimiento. En estos tipos de modelos, destacan los árboles de decisión y los random forest (una media de árboles de decisión) [42].

Árboles de decisión

Los árboles de decisión son modelos ampliamente utilizados para tareas de clasificación y regresión en un conjunto de datos. Esencialmente, aprenden una jerarquía de preguntas que tienen respuestas del tipo if/else, y así logran conducir a una decisión teniendo como objetivo llegar a la respuesta correcta haciendo la menor cantidad posible de preguntas. Esta serie de preguntas se puede expresar como un árbol de decisiones, como se muestra en la figura 2.5.3 [41].

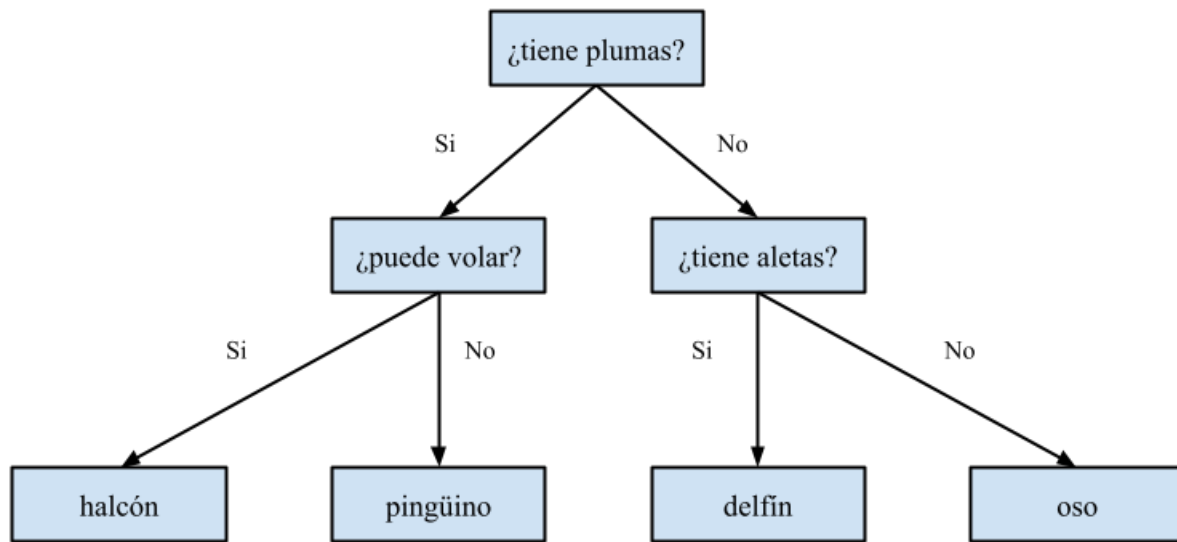


Figura 2.5.3. Un árbol de decisiones para distinguir entre varios animales.

En esta ilustración, cada nodo del árbol representa una pregunta o un nodo terminal (también llamado hoja) que contiene la respuesta. Las flechas conectan las respuestas a una pregunta, con la siguiente pregunta qué se haría. En el ejemplo, se construye un modelo para distinguir entre cuatro clases de animales (halcones, pingüinos, delfines y osos) utilizando tres características «tiene plumas», «puede volar» y «tiene aletas». En lugar de construir estos modelos a mano, podemos aprenderlos de los datos mediante el aprendizaje supervisado.

Dentro del modelo, las «preguntas» se conocen como pruebas y se utilizan para analizar los datos que representan las características continuas presentes. En el aprendizaje automático, estos datos no se presentan como características binarias del tipo sí/no, sino como características continuas. Al utilizar un árbol de decisión, el algoritmo examina todas las posibles pruebas disponibles para encontrar la que sea más informativa en cuanto a la variable objetivo. Luego, al hacer una predicción, se recorre el árbol utilizando las pruebas en cada nodo hasta encontrar la hoja correspondiente al nuevo punto de datos, y se toma como salida el valor medio de los puntos de entrenamiento en esa hoja [41].

En resumen, para un árbol de decisión, en el ámbito del aprendizaje automático, aprender implica analizar datos continuos utilizando pruebas cuidadosamente seleccionadas para obtener la información más relevante. Los árboles se pueden utilizar tanto para tareas de clasificación como de regresión, y la predicción se realiza mediante la exploración del árbol y la selección de la hoja correspondiente al nuevo punto de datos.

Bosques aleatorios (Random forest)

Uno de los principales inconvenientes de los árboles de decisión mencionados anteriormente, es que tienden a sobreajustar los datos de entrenamiento. Los bosques aleatorios son una forma de abordar este problema. Un bosque aleatorio es esencialmente una colección de árboles de decisión, en la que cada árbol es ligeramente diferente de los demás. La idea que subyace a los bosques aleatorios es que cada árbol puede hacer un trabajo de predicción relativamente bueno, pero es probable que sobreajuste parte de los datos. Si construimos muchos árboles individuales que funcionen bien y se sobreajusten de diferentes maneras, podemos reducir la cantidad de sobreajuste promediando sus resultados. Esta reducción de la sobreadaptación, al tiempo que se mantiene el poder predictivo de los árboles, puede demostrarse utilizando matemáticas rigurosas [41].

Para aplicar esta estrategia, necesitamos construir muchos árboles de decisión. Cada árbol debe realizar un trabajo aceptable de predicción del objetivo, y también debe ser diferente de los demás árboles. Los bosques aleatorios reciben su nombre de la inyección de aleatoriedad en la construcción de árboles para garantizar que cada árbol sea diferente. Hay dos formas de aleatorizar los árboles en un bosque aleatorio: seleccionando los puntos de datos utilizados para construir un árbol y seleccionando las características en cada prueba de división [41].

Redes neuronales

Las redes artificiales de neuronas tratan, en cierto modo, de replicar el comportamiento del cerebro, donde existen millones de neuronas que se interconectan en red para enviarse mensajes unas a otras. Esta réplica del funcionamiento del cerebro humano es uno de los modelos de IA con mayor auge actual, debido a las habilidades cognitivas de razonamiento que adquieren. Estas redes tienen la particularidad de tener un funcionamiento similar al cerebro humano, lo que las hace prácticamente insuperables en términos de precisión. El problema, como el cerebro humano, es que son lentos de entrenar y necesitan una gran cantidad de cómputo para su funcionamiento aunque afortunadamente, la «revolución de los datos» ha permitido mejorar sustancialmente la capacidad de estas redes; tanto en términos de información como de procesamiento, lo que ha contribuido a mejorar su desempeño. [42].

2.5.8 Tipos de aprendizajes

En el aprendizaje máquina se conocen tres tipos de aprendizaje: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. A continuación daremos una breve descripción de cada uno de ellos.

Aprendizaje supervisado

En el aprendizaje supervisado, el usuario proporciona al algoritmo pares de entradas y salidas deseadas (también denominadas etiquetas), y el algoritmo encuentra la forma de producir la salida deseada a partir de una entrada. Es un proceso donde se generalizan las relaciones entre las observaciones de sus características de entrada y salida. En concreto, el algoritmo es capaz de crear una salida para una entrada que nunca ha visto antes sin ayuda humana, es

decir este modelo de «relación» especula una salida o etiqueta para producir nuevas observaciones. Este tipo de aprendizaje se utiliza cuando se necesita predicción, es decir una situación en la que se tiene una variable de salida, ya sea cuantitativa o cualitativa, que se desea predecir basándose en un conjunto de características [35][40][41].

Los algoritmos de aprendizaje automático que aprenden a partir de pares de entrada/salida se denominan algoritmos de aprendizaje supervisado porque un «supervisor» proporciona monitoreo a los algoritmos en forma de las salidas deseadas para cada ejemplo del que aprenden [41]. Algunos ejemplos de tareas de aprendizaje automático supervisado son:

- ❖ Clasificación de sentimientos en comentarios de redes sociales: La entrada es un comentario de redes sociales, y la salida deseada es clasificar si el sentimiento expresado es positivo, negativo o neutro. Para crear un conjunto de datos para entrenar un modelo de aprendizaje automático, se necesitaría recopilar comentarios de redes sociales y etiquetarlos con las categorías de sentimiento correspondientes.
- ❖ Predicción del precio de una vivienda basada en características del inmueble: La entrada son las características de una vivienda, como el número de habitaciones, la ubicación, el tamaño, etc., y la salida deseada es el precio estimado de la vivienda. Para construir un conjunto de datos para entrenar un modelo de aprendizaje automático, se requeriría recopilar datos de ventas de viviendas pasadas, incluyendo sus características y precios de venta.
- ❖ Clasificación de correos electrónicos como spam o legítimos: La entrada es el contenido de un correo electrónico, y la salida deseada es clasificar si el correo es spam o legítimo. Para crear un conjunto de datos para entrenar un modelo de aprendizaje automático, se necesitaría recopilar una gran cantidad de correos electrónicos etiquetados como spam o legítimos, y utilizarlos para entrenar el modelo de clasificación.
- ❖ Pronóstico del clima a partir de datos meteorológicos: En este caso, la entrada son datos meteorológicos históricos, como la temperatura, humedad, velocidad del viento, etc. y la salida deseada es el pronóstico del clima, como soleado, lluvioso o nublado. El proceso de obtención de datos puede ser relativamente sencillo, ya que se pueden recopilar datos meteorológicos históricos de fuentes públicas, como estaciones meteorológicas o bases de datos meteorológicas, y utilizarlos para entrenar el modelo.
- ❖ Diagnóstico de enfermedades a partir de datos clínicos: En este caso, la entrada son datos clínicos de pacientes, como resultados de pruebas de laboratorio, síntomas, historiales médicos, etc. y la salida deseada es el diagnóstico de la enfermedad, como diabetes, enfermedades cardíacas, etc. El proceso de obtención de datos puede ser más complejo, ya que se necesita una base de datos clínica con información detallada de pacientes, así como la participación de expertos médicos para etiquetar los datos y determinar los diagnósticos correctos.

Un aspecto interesante de estos ejemplos es que, aunque las entradas y salidas parecen bastante sencillas, el proceso de recolección de datos para algunas tareas puede ser muy diferente. Desde la recopilación de datos históricos disponibles públicamente que, en principio pareciera requerir un esfuerzo menor comparado con otras tareas, hasta la necesidad

de datos clínicos detallados y la participación activa de expertos para etiquetar y validar esos datos [41].

Aprendizaje no supervisado

En el aprendizaje no supervisado, sólo se conocen los datos de entrada y no se proporcionan datos de salida conocidos al algoritmo; es decir existe un conjunto de datos que contienen diversas características de determinados individuos, sin que ninguna de ellas se considere una variable de salida que se desee predecir. Este es un proceso de aprendizaje para generalizar la estructura en los datos donde no se requiere predicción; las estructuras naturales son identificadas y explotadas para relacionar ciertas observaciones respecto a otras. En este caso la tarea de aprendizaje es describir cómo están organizados los datos, posibles asociaciones entre ellos o agrupamientos [35][40]. Algunos ejemplos de aprendizaje no supervisado son:

- ❖ Identificación de tendencias en datos de redes sociales: En un conjunto de datos de redes sociales, se puede realizar un análisis no supervisado para identificar las tendencias o temas más relevantes que están siendo discutidos por los usuarios. No se conoce de antemano cuáles son esos temas ni cuántos puede haber, por lo que se trata de un problema de aprendizaje no supervisado.
- ❖ Clasificación de noticias en categorías temáticas: Dado un conjunto de noticias sin etiquetas de categoría, se puede utilizar el aprendizaje no supervisado para agrupar las noticias en diferentes categorías temáticas, como deportes, política, tecnología, entre otros. No se tienen resultados conocidos previamente y se busca identificar posibles asociaciones entre las noticias.
- ❖ Análisis de comportamientos de navegación en un sitio web: Para entender cómo los usuarios interactúan con un sitio web, se puede realizar un análisis no supervisado de los datos de navegación para identificar patrones de comportamiento, como páginas visitadas con mayor frecuencia, secuencias de navegación comunes o patrones anormales de navegación. No se sabe de antemano qué comportamientos son considerados normales o anormales, por lo que se trata de un problema de aprendizaje no supervisado.

Para las tareas de aprendizaje supervisado y no supervisado, es importante tener una representación de los datos de entrada que un ordenador pueda entender. Sin embargo, hay que tener en cuenta que ningún algoritmo de aprendizaje automático será capaz de hacer una predicción sobre datos de los que no tiene información. Por ejemplo, si la única característica que tiene de un paciente es su apellido, ningún algoritmo podrá predecir su sexo. Sencillamente, los datos no contienen esta información. Si añade otra característica que contenga el nombre del paciente, tendrá mucha más suerte, ya que a menudo es posible saber el sexo por el nombre de una persona [41].

Aprendizaje por refuerzo

En el aprendizaje por refuerzo, un «agente» debe observar el entorno del cual el algoritmo tiene que “aprender”, como también seleccionar y realizar acciones con la finalidad de obtener recompensas a cambio o sanciones en forma de recompensas negativas. Luego, debe

aprender por sí mismo cuál es la mejor estrategia, llamada también política, para obtener la mayor recompensa con el paso del tiempo. Con una política, se define qué acción debe hacer el agente cuando se encuentra en una situación dada [40]. Algunos ejemplos de aprendizaje por refuerzo son:

- ❖ Entrenamiento de un dron para realizar entregas autónomas: En este caso, el dron aprende a través de la interacción con el entorno, como los mapas de navegación y los obstáculos en su ruta. Recibe recompensas por entregar los paquetes de manera exitosa y a tiempo, y es penalizado por errores como caídas o entregas tardías. El objetivo es maximizar la eficiencia y precisión en las entregas a lo largo del tiempo.
- ❖ Optimización de la gestión de tráfico en una intersección: En este ejemplo, el agente de aprendizaje interactúa con una intersección de tráfico, ajustando los semáforos y el flujo de vehículos para minimizar la congestión y maximizar el flujo vehicular. Recibe recompensas por mantener un flujo suave y eficiente del tráfico, y es penalizado por congestiones o accidentes. El objetivo es maximizar la eficiencia del tráfico en la intersección.
- ❖ Control de una red de sensores para la monitorización del medio ambiente: En este caso, el agente de aprendizaje controla una red de sensores distribuidos en un área de estudio, ajustando los parámetros de muestreo y transmisión de datos para obtener una cobertura óptima y eficiente del medio ambiente. Recibe recompensas por obtener mediciones precisas y completas, y es penalizado por errores en la cobertura o consumo excesivo de energía. El objetivo es maximizar la calidad y eficiencia de las mediciones ambientales.

Una vez definidos y comprendidos los tipos de problemas, los tipos de datos y los tipos de aprendizajes, se puede pasar a conocer cuáles son las etapas requeridas para el desarrollo de sistemas basados en el aprendizaje máquina [40].

2.5.9 Etapas de implementación del aprendizaje máquina

El aprendizaje automático usa la teoría estadística para construir modelos matemáticos, pues de esta manera es posible hacer inferencias a partir de una muestra. La implementación de algoritmos de aprendizaje máquina implica una serie de pasos que son muy similares al proceso de modelado estadístico. Estos se dividen en cinco etapas: recopilación de datos, exploración y preparación de datos, entrenamiento, evaluación del modelo y, por último, mejora del modelo. En la figura 2.5.4 [40] se pueden observar las etapas de implementación del aprendizaje máquina de una forma esquemática. Cada etapa se define a continuación [35][40].

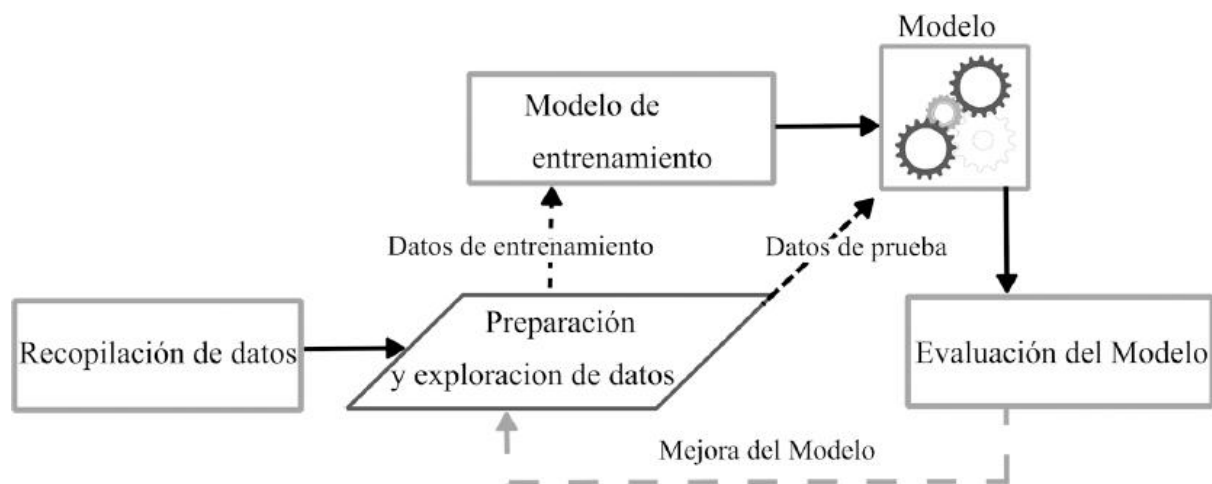


Figura 2.5.4. Etapas de implementación de un algoritmo de aprendizaje máquina.

Recopilación de datos

Este paso implica reunir el material de aprendizaje que implementará un algoritmo para generar un conocimiento procesable. En la mayoría de los casos, los datos deberán combinarse en una sola fuente, como un archivo de texto, hoja de cálculo o base de datos [40].

Exploración y preparación de datos

La calidad de cualquier proyecto de aprendizaje automático se basa, en gran medida, en la calidad de sus datos de entrada. Para mejorar la calidad de los datos, estos requieren de una preparación especial para el proceso de aprendizaje. Esta preparación implica arreglar o limpiar los llamados datos «desordenados», eliminando datos innecesarios, y recodificando los datos para ajustarse a las entradas esperadas de la técnica de aprendizaje máquina [40].

Entrenamiento

Una vez que los datos han sido preparados, estos alimentan a la técnica o algoritmo específico, el cual construirá un modelo con base en los datos de entrenamiento [40]. La ciencia de la computación es requerida en esta fase para la implementación de algoritmos de optimización eficientes [35].

Generalización, sobreajuste y subajuste

En el aprendizaje automático, se busca construir un modelo sobre los datos de entrenamiento y luego poder hacer predicciones precisas sobre datos nuevos e invisibles que tengan las mismas características que el conjunto de entrenamiento que fue utilizado. Si un modelo es capaz de hacer predicciones precisas sobre datos no conocidos previamente, es posible afirmar que el modelo puede generalizar desde el conjunto de entrenamiento al conjunto de prueba, el objetivo perseguido es construir un modelo que sea capaz de generalizar con la mayor precisión posible. Frecuentemente los modelos se construyen con el objetivo de hacer predicciones precisas en el conjunto de datos de entrenamiento. No obstante, si el conjunto de entrenamiento y prueba tienen muchas similitudes, se espera que el modelo también sea preciso en el conjunto de prueba. Sin embargo, esto puede salir mal si se emplea una

estrategia de sobreajuste al entrenar un modelo muy complejo, lo cual puede impedir que el modelo generalice bien con nuevos datos o en el conjunto de prueba. La única forma de anticipar si un modelo entrenado funcionará correctamente con nuevos datos desconocidos es evaluarlo en el conjunto de prueba. Se espera que los modelos simples generalicen mejor los nuevos datos, por lo que se busca encontrar el modelo más simple [41].

El sobreajuste ocurre cuando se ajusta un modelo demasiado cerca de las particularidades del conjunto de entrenamiento, obteniendo un modelo que funciona bien en ese conjunto, pero que no puede generalizarse a nuevos datos. Por otro lado, si el modelo es demasiado simple, no podrá capturar toda la variabilidad en los datos, obteniendo un bajo porcentaje de confiabilidad incluso en los datos de entrenamiento, lo cual se conoce como desajuste. En este sentido, cuanto más complejo sea nuestro modelo, mejor podremos predecir sobre los datos de entrenamiento. No obstante, si el modelo es demasiado complejo, nos centraremos demasiado en cada punto de datos individual del conjunto de entrenamiento, lo que impedirá que el modelo se generalice bien a nuevos datos. Por lo tanto, se busca encontrar el punto óptimo en el medio que produzca el mejor rendimiento de generalización [41].

Relación de la complejidad del modelo con el tamaño del conjunto de datos

Es importante tener en cuenta, que la complejidad del modelo de aprendizaje automático está fuertemente ligada a la variabilidad de las entradas contenidas en su conjunto de datos de entrenamiento. Cuanta más variedad tenga su conjunto de datos, más complejo puede ser el modelo que se puede utilizar sin caer en el sobreajuste. Por lo general, la recopilación de más puntos de datos genera más variabilidad, lo que permite construir modelos más complejos. Contar con mayor cantidad de datos y construir adecuadamente modelos más complejos puede hacer maravillas en las tareas de aprendizaje supervisado. Nunca subestime el poder de los datos. Además, tener un conjunto de datos más grande también puede ayudar a reducir el impacto de valores atípicos o errores en los datos, lo que puede conducir a un modelo más preciso. Sin embargo, es importante tener en cuenta que simplemente agregar más datos no siempre es la solución. La calidad de los datos es fundamental, y un conjunto de datos grande pero de baja calidad puede ser peor que un conjunto de datos pequeño pero de alta calidad. En última instancia, encontrar un equilibrio entre la cantidad y la calidad de los datos es clave para construir modelos de aprendizaje automático precisos y efectivos [41].

Evaluación del modelo

Debido a que cada modelo puede tener un sesgo en su construcción que afecte la solución del problema, es crucial evaluar la eficiencia del algoritmo durante su entrenamiento. La evaluación de la precisión del modelo depende del tipo de modelo utilizado y en ocasiones requiere el desarrollo de medidas de rendimiento específicas para la aplicación prevista [40]. En nuestro caso, al utilizar modelos de regresión, se seleccionaron métricas específicas para evaluar su desempeño, tales como:

Varianza explicada

La varianza explicada es una medida que indica qué porcentaje de la varianza de la variable de salida puede ser explicado por el modelo. Esta métrica se utiliza para evaluar la dispersión

o variación de los puntos de datos, y su mejor puntuación posible es 1.0, lo que significa que el modelo explica toda la variabilidad de los datos de forma perfecta. No obstante, una puntuación muy alta en esta métrica puede indicar que el modelo está sobreajustado a los datos de entrenamiento, lo que significa que no se generalizará bien a nuevos datos. Por lo tanto, es importante encontrar un equilibrio en el valor de esta métrica para evitar el riesgo de overfitting.

Error medio absoluto

El error medio absoluto (MAE) es una medida estadística que representa la media de las diferencias absolutas entre el valor objetivo y el valor predicho. A diferencia del error cuadrático medio (MSE), el MAE no eleva los errores al cuadrado, lo que significa que no penaliza los errores grandes. Esta característica hace que el MAE no sea muy sensible a valores atípicos, por lo que no se recomienda utilizar esta métrica en modelos en los que se deba prestar atención a estos valores. En general, cuanto más cercano a cero sea el valor del MAE, mejor será el rendimiento del modelo.

Error cuadrático medio

El error cuadrático medio (MSE) es una métrica que mide la media de las diferencias al cuadrado entre el valor objetivo y el valor predicho. Al elevar al cuadrado los errores, esta métrica magnifica los errores grandes, lo que puede ser problemático si hay valores atípicos en nuestro conjunto de datos. Por lo tanto, es importante utilizar esta métrica con precaución. Un valor de MSE cercano a cero indica una buena precisión del modelo en la predicción de los valores objetivos.

Raíz cuadrada de la media del error al cuadrado

El RMSE es una métrica que se calcula como la raíz cuadrada del MSE, el cual es la media de las diferencias entre el valor objetivo y el valor predicho al cuadrado. Al elevar al cuadrado los errores, se magnifican los errores grandes, por lo que es necesario utilizarla con precaución cuando existen valores atípicos en el conjunto de datos. La ventaja de utilizar el RMSE es que presenta el error en las mismas unidades que la variable objetivo, lo que facilita su interpretación. Se considera que un valor de RMSE más cercano a cero indica una mayor precisión del modelo en la predicción de la variable objetivo.

R cuadrado

El coeficiente de determinación, también conocido como R^2 , es una métrica que se utiliza para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo. El resultado del coeficiente de determinación oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. Por el contrario, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto,

menos fiable será. R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

R cuadrado ajustado

El coeficiente de determinación ajustado (R^2 ajustado) es una métrica que soluciona el problema del R^2 al penalizar la inclusión de variables explicativas no significativas en el modelo. El R^2 ajustado se calcula teniendo en cuenta tanto la cantidad de variables explicativas como el R^2 del modelo. Cuantas más variables se agreguen al modelo, menor será el R^2 ajustado, a menos que estas variables agreguen valor predictivo significativo. A diferencia del R^2 , el R^2 ajustado proporciona una medida más precisa de la calidad del ajuste del modelo a los datos de entrenamiento, lo que lo hace más recomendable para el análisis.

Mejora del modelo

Para lograr un mejor rendimiento en los modelos de aprendizaje automático, es importante considerar estrategias avanzadas que puedan aumentar su precisión. En algunos casos, puede ser necesario cambiar a un tipo diferente de algoritmo que pueda realizar la tarea de manera más precisa. Además, puede ser útil complementar los datos de entrenamiento con información adicional que sea más representativa del problema que se está abordando. Esto puede implicar trabajo preparatorio adicional, como se indica en el paso 2 del proceso (véase figura 2.5.4) [40].

Es importante destacar que existen métodos más sólidos para evaluar el rendimiento de la generalización, como la validación cruzada, que permite una evaluación más precisa del rendimiento del modelo como también la búsqueda en cuadrícula o Grid Search, que es un método eficaz para ajustar los parámetros en modelos supervisados y obtener un mejor rendimiento de generalización. Además, existen métodos para evaluar el rendimiento de la clasificación y la regresión que van más allá de las medidas predeterminadas de precisión y R^2 proporcionadas por el método de puntuación. Estos métodos pueden proporcionar una comprensión más detallada y permitir la mejora del rendimiento del modelo en tareas de aprendizaje supervisado. Algunos de ellos son [41]:

Validación cruzada

La validación cruzada es un método estadístico de evaluación del rendimiento de generalización más estable y exhaustivo que el uso de una división en un conjunto de entrenamiento y otro de prueba. En la validación cruzada, los datos se dividen repetidamente y se entrenan varios modelos. La versión más utilizada de la validación cruzada es la validación cruzada k-fold, en la que k es un número especificado por el usuario, normalmente 5 o 10 que indica la cantidad de grupos en los que se dividirán los datos. Al realizar esta validación los datos se dividen repetidamente y se entrenan varios modelos. Cuando se realiza una validación cruzada quintuple, los datos se dividen primero en cinco partes de (aproximadamente) el mismo tamaño, llamadas pliegues. A continuación, se entrena una secuencia de modelos. El primer modelo se entrena utilizando el primer pliegue como conjunto de prueba, y los pliegues restantes (2-5) se utilizan como conjunto de entrenamiento. El modelo se construye utilizando los datos de los pliegues 2-5, y luego se

evalúa la precisión en el pliegue 1. A continuación, se construye otro modelo, esta vez utilizando el pliegue 2 como conjunto de prueba y los datos de los pliegues 1, 3, 4 y 5 como conjunto de entrenamiento. Este proceso se repite utilizando los pliegues 3, 4 y 5 como conjuntos de prueba. Para cada una de estas cinco divisiones de los datos en conjuntos de entrenamiento y de prueba, calculamos la precisión. Al final, hemos recopilado cinco valores de precisión. El proceso se ilustra en la figura 2.5.4 [41]:

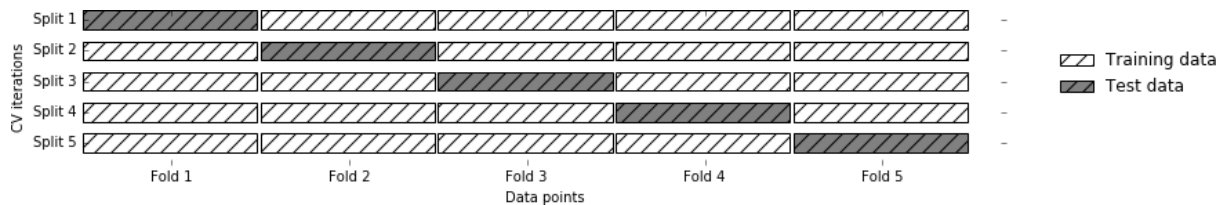


Figura 2.5.5. División de datos en validación cruzada quintuple.

La validación cruzada es una técnica que ofrece múltiples ventajas en comparación con una única división de los datos en un conjunto de entrenamiento y otro de prueba. En una división aleatoria de datos, es posible llevar a una distribución desigual de los ejemplos difíciles y fáciles de clasificar entre los conjuntos de entrenamiento y prueba, lo que resulta en una precisión irrealmente alta o baja. Por el contrario, la validación cruzada garantiza que cada ejemplo esté en el conjunto de entrenamiento y prueba exactamente una vez, lo que asegura que el modelo generalice bien para todos los datos y que las puntuaciones de validación cruzada sean representativas de la verdadera precisión del modelo. Por otro lado, esta técnica proporciona información sobre la sensibilidad del modelo a la selección del conjunto de datos de entrenamiento, ya que las precisiones obtenidas pueden variar entre diferentes pliegues de los datos. Esto nos permite utilizar nuestros datos de manera más eficiente, ajustando el modelo con un mayor porcentaje de datos en cada iteración de validación cruzada en comparación con una única división de los datos. Aunque la validación cruzada tiene la desventaja de aumentar significativamente el costo computacional al requerir el entrenamiento de múltiples modelos, sigue siendo una técnica muy útil para evaluar la capacidad de generalización de un modelo y obtener una estimación más precisa de su rendimiento en datos no vistos [41].

Validación cruzada estratificada k-fold y otras estrategias

Dividir el conjunto de datos en k pliegues empezando por la primera- k -ésima parte de los datos, como se describe en la sección anterior, puede no ser siempre una buena idea. El problema con el método de validación cruzada convencional es que, al dividir los datos en k partes iguales, puede haber una distribución desigual de las clases en cada parte, y provocar una mala evaluación del modelo. Por ejemplo, si existe una clase que es menos frecuente en los datos, es posible que no se incluya en una de las partes, lo que resultaría en una evaluación sesgada del modelo. La solución a este problema es el método de validación cruzada estratificada o k -fold estratificado. Este método, en lugar de dividir los datos de forma aleatoria en k partes iguales, se asegura que cada parte tenga la misma proporción de clases que el conjunto de datos completo. De esta manera, cada grupo de datos representará

de manera más precisa la distribución de las clases en los datos originales, lo que conlleva a una evaluación más justa y precisa del modelo [41].

Grid Search (Búsqueda en cuadrícula)

Una vez analizadas las técnicas para evaluar cuán «bueno» es un modelo generalizando datos, es posible dar el siguiente paso y mejorar el rendimiento de generalización del modelo ajustando sus parámetros. Encontrar los valores de los parámetros importantes de un modelo (los que proporcionan el mejor rendimiento de generalización) es una tarea compleja, pero necesaria para casi todos los modelos y conjuntos de datos, para ello es posible aplicar la técnica de Grid Search. *Grid Search* es un método de optimización de parámetros que se utiliza en el aprendizaje automático. Estos parámetros se definen antes del entrenamiento del modelo, como la tasa de aprendizaje, la profundidad del árbol de decisión, el valor de C en la regresión logística, etc. Estos parámetros pueden influir significativamente en el rendimiento del modelo y, por lo tanto, es necesario encontrar los valores óptimos para ellos [41].

Este método funciona como una técnica sistemática que permite explorar el espacio de parámetros de un modelo. Para ello se crea una cuadrícula o matriz que contiene todas las posibles combinaciones de valores de los parámetros del modelo que se desean evaluar. Luego, se entrena y se evalúa el modelo para cada combinación de parámetros, utilizando validación cruzada para obtener una estimación del rendimiento del modelo en datos no vistos. Grid Search es una técnica de optimización de parámetros en modelos de aprendizaje automático, cuyo objetivo es encontrar la mejor combinación de valores de parámetros que maximice el rendimiento del modelo en los datos de validación. Esta técnica es especialmente útil para modelos con múltiples parámetros que deben ajustarse y no se conoce la mejor combinación de valores para maximizar el rendimiento. Aunque puede ser computacionalmente costosa, esta técnica permite explorar un espacio de parámetros más amplio que una búsqueda manual y mejorar el rendimiento del modelo [41].

2.6 Procesamiento de Lenguaje Natural

La mayor parte del conocimiento científico es el resultado de muchos años de investigación, con frecuencia sobre temas aparentemente no relacionados. Y lo es mucho más en las ciencias de la computación, en donde el recurso más importante que poseemos es información y conocimiento. En la época actual el uso de los recursos naturales, industriales y humanos depende del manejo eficiente de la información y conocimiento. Desde épocas remotas hasta la actualidad, el cúmulo de conocimientos ha experimentado un incremento exponencial y ha sido registrado en diversas formas, como documentos, libros y artículos, siendo almacenados en diferentes medios como formato impreso y, con la llegada de las computadoras y el procesamiento digital, en formato electrónico. Este avance tecnológico ha propiciado un mayor crecimiento en el registro y acceso al conocimiento. Sin embargo, lo que es conocimiento para nosotros (los seres humanos) no lo es para las computadoras. La computadora almacena datos e información en archivos, puede copiar tal archivo, respaldarlo, transmitirlo, borrarlo, pero no puede buscar las respuestas a preguntas formuladas, hacer inferencias lógicas sobre su contenido, generalizar y resumirlo, es decir, hacer todo lo que las personas normalmente hacemos con el texto. Porque no lo puede entender [43].

2.6.1 ¿Qué es el Procesamiento del Lenguaje Natural?

El Procesamiento del Lenguaje Natural (NLP) es una disciplina que se encuentra en la intersección de distintas ciencias, tales como las Ciencias de la Computación, la Inteligencia Artificial y la Psicología Cognitiva. Su idea central es la de darle a las máquinas la capacidad de leer y comprender los idiomas que hablamos los humanos. La investigación del Procesamiento del Lenguaje Natural tiene como objetivo responder a la pregunta de cómo las personas son capaces de comprender el significado de una oración oral/escrita y cómo las personas entienden lo que sucedió, cuándo y dónde sucedió; y las diferencias entre una suposición, una creencia o un hecho [39]. Ahora desde la perspectiva de la IA, el estudio del lenguaje natural tiene dos objetivos [43]:

- ❖ Objetivo 1: Facilitar la comunicación con la computadora para que accedan a ella usuarios no especializados. Esto se soluciona consiguiendo un intérprete para una clase de aplicaciones en un dominio restringido, que haga de traductor entre el computador y el usuario.
- ❖ Objetivo 2: Modelar los procesos cognitivos que entran en juego en la comprensión del lenguaje para diseñar sistemas que realicen tareas lingüísticas complejas (traducción, resúmenes de textos, recuperación de información, etc). Estos procesos son complejos ya que intervienen grandes cantidades de conocimiento de naturaleza diferente (morfología, sintaxis, semántica, pragmática) y mecanismos de tratamiento variados (de comparación, búsqueda, inferencia aproximada, deducción, etc).

2.6.2 Definición de lenguaje

El lenguaje es una de las herramientas centrales en nuestra vida social y profesional. Entre otras cosas, actúa como un medio para transmitir ideas, información, opiniones y sentimientos; así como para persuadir, pedir información, o dar órdenes. Asimismo, el lenguaje humano es algo que está en constante cambio y evolución; y que puede llegar a ser muy ambiguo y variable [39].

Un lenguaje se puede definir de diferentes formas: desde el punto de vista funcional lingüístico se define como una función que expresa pensamientos y comunicaciones entre la gente. Esta función puede realizarse mediante signos escritos (escritura) o mediante señales y vocales (voz). Desde un punto de vista formal se define como un conjunto de frases, que generalmente es infinito y se forma con combinaciones de elementos tomados de un conjunto (usualmente finito) llamado alfabeto, respetando un conjunto de reglas de formación (sintácticas o gramaticales) y de sentido (semánticas). Además de las características fundamentales del lenguaje debe considerarse que sea funcional, es decir, el lenguaje debe permitirnos expresar nuestras ideas. El lenguaje será bueno en la medida en que sea fácil de leer, fácil de entender y fácil de modificar. Lo mismo ocurre en los lenguajes formales [43].

Podemos distinguir entre dos clases de lenguajes: los lenguajes naturales (inglés, alemán, español, etc) y lenguajes formales (matemático, lógico, programable, etc) [43].

2.6.3 Definición de lenguaje natural

Un *lenguaje natural* (LN) es aquel que se ha desarrollado a lo largo del tiempo para la comunicación humana, como el español, el alemán o cualquier otro idioma que evoluciona de manera orgánica. A diferencia de los lenguajes formales, que están definidos por reglas preestablecidas, los lenguajes naturales no siguen una gramática estricta y las reglas gramaticales suelen surgir después de que se hayan utilizado en la comunicación. El lenguaje natural es el medio que utilizamos en nuestra vida cotidiana para comunicarnos con otras personas y que ha evolucionado a partir de la experiencia humana y perfeccionado hasta el punto de poder ser utilizado para analizar situaciones complejas y razonar de manera sutil.

Por otro lado, aunque los lenguajes naturales no siguen reglas estrictas de sintaxis, su estructura puede ser modelada fácilmente por lenguajes formales, similares a los utilizados en las matemáticas y la lógica. Este hecho permite que los lenguajes naturales también puedan ser analizados y estudiados desde un enfoque más formal.

Algunas de las propiedades que caracterizan a estos tipos de lenguajes incluyen su evolución constante, su capacidad para expresar ideas complejas y sutiles, y su capacidad de ser modelados por lenguajes formales. Sin embargo, a pesar de su riqueza y flexibilidad, los lenguajes naturales también presentan desafíos en términos de ambigüedad, variabilidad dialectal y cambios en su estructura a lo largo del tiempo aunque a pesar de estas características siguen siendo una herramienta fundamental para la comunicación humana y el razonamiento en diversas áreas del conocimiento. Los lenguajes naturales se pueden caracterizar por las siguientes propiedades [43]:

1. Un lenguaje natural se define a partir de una gramática G , sin embargo, este se enriquece progresivamente modificando así también la gramática que la define. Esto dificulta la formalización de la definición de G .
2. Un lenguaje natural tiene un gran poder expresivo debido a la riqueza del componente semántico (polisemántica). Esto dificulta aún más la formalización completa de su gramática.

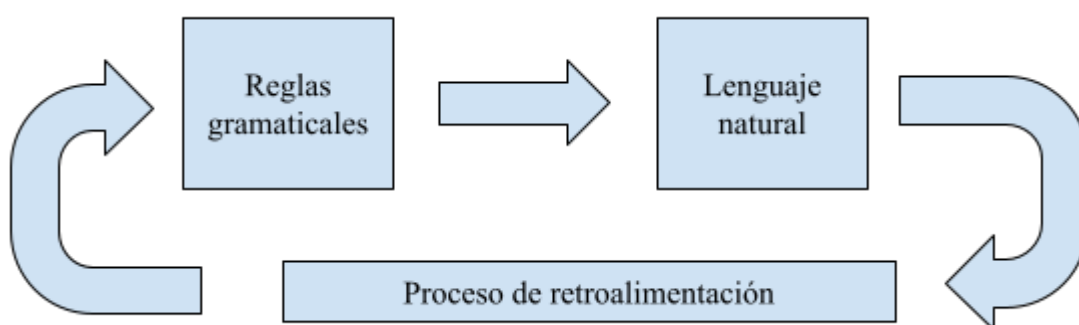


Figura 2.6.1. Definición de lenguaje natural.

La figura 2.6.1 [43] representa el ciclo continuo que se da en la definición del lenguaje natural. Este proceso nunca finaliza, ya que a medida que se adquiere más conocimiento y se incorporan nuevas palabras y expresiones, se sigue perfeccionando la capacidad para entender y expresarse.

2.6.4 Definición de lenguaje formal

El lenguaje formal es aquel que el hombre ha desarrollado para expresar las situaciones que se dan en específico en cada área del conocimiento científico. Los lenguajes formales pueden ser utilizados para modelar una teoría de la mecánica, física, matemática, ingeniería eléctrica, o de otra naturaleza, con la ventaja de que en estos toda ambigüedad es eliminada. Los que más se destacan son los lenguajes de programación de computadoras, que se definen considerando un conjunto de componentes léxicos, reglas gramaticales y una delimitación semántica [43].

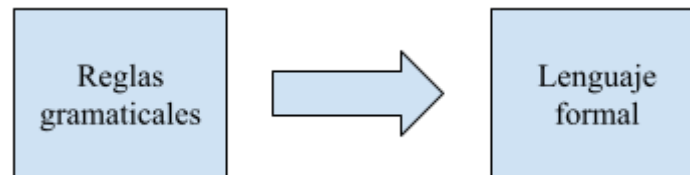


Figura 2.6.2. Definición de lenguaje formal.

La figura 2.6.2 [43] ilustra el ciclo finito utilizado en la definición de un lenguaje formal. A diferencia del lenguaje natural, que puede ser ambiguo y subjetivo, los lenguajes formales tienen una sintaxis y semántica precisas y bien definidas. En resumen, los lenguajes formales se desarrollan a partir de una gramática G preestablecida y tienen un componente semántico mínimo, aunque existe la posibilidad de incrementarlo según la teoría a formalizar. Su sintaxis produce oraciones no ambiguas y tienen una completa formalización, lo que les otorga un gran potencial en la construcción computacional. El proceso inicia con la definición de un conjunto de símbolos y reglas de formación de cadenas de símbolos, también conocido como gramática. A partir de esta gramática, se pueden generar un conjunto finito de cadenas de símbolos que conforman el lenguaje formal. Estas cadenas pueden ser interpretadas de diversas maneras, como expresiones matemáticas, programas de computadora o cualquier otro tipo de información estructurada. [43].

2.6.5 Procesamiento Computacional del Lenguaje Natural

En el campo de la Inteligencia Artificial, una tarea esencial es la manipulación de lenguajes naturales utilizando herramientas de computación. En este contexto, los lenguajes de programación desempeñan un papel crucial, ya que actúan como el enlace necesario entre los lenguajes naturales y su procesamiento por una máquina. El NLP consiste en la utilización de un lenguaje natural para comunicarnos con la computadora, debiendo ésta entender las oraciones que le sean proporcionadas. El uso de estos lenguajes naturales, facilita el desarrollo de programas que realicen tareas relacionadas con el lenguaje o bien, desarrollar modelos que ayuden a comprender los mecanismos humanos relacionados con el lenguaje [43].

2.6.6 Aplicaciones del NLP

Los algoritmos de Procesamiento del Lenguaje Natural suelen basarse en algoritmos de aprendizaje automático. En lugar de codificar manualmente grandes conjuntos de reglas, el NLP puede confiar en el aprendizaje automático para aprender estas reglas automáticamente analizando un conjunto de ejemplos y haciendo una inferencia estadística. En general, cuanto más datos analizados, más preciso será el modelo. Las aplicaciones del NLP son muy variadas, ya que su alcance es muy grande, algunas de las aplicaciones son:

- ❖ **Análisis de opinión:** Utilizando NLP, se pueden analizar las opiniones de los clientes en redes sociales o comentarios de productos para identificar si son positivos, negativos o neutrales.
- ❖ **Generación de resúmenes automáticos:** Utilizando técnicas de NLP, se pueden crear resúmenes automáticos de documentos largos o artículos de noticias, proporcionando una visión general rápida del contenido.
- ❖ **Análisis de tono de voz:** El NLP se puede utilizar para analizar el tono de voz en grabaciones de llamadas de servicio al cliente o en entrevistas para identificar si el hablante está emocionalmente positivo, negativo o neutral.
- ❖ **Detección de sarcasmo:** Los modelos de NLP pueden ser entrenados para identificar el sarcasmo en el lenguaje natural, lo que puede ser útil en la detección de actitudes o intenciones ocultas en el texto.
- ❖ **Generación de contenido creativo:** Utilizando técnicas de NLP, se pueden generar automáticamente historias, poemas o piezas de contenido creativo basadas en patrones y estructuras del lenguaje.
- ❖ **Análisis de similitud de texto:** Con NLP, se pueden comparar y analizar textos para determinar su similitud o diferencia en términos de contenido o estilo de escritura.
- ❖ **Extracción de información de facturas o documentos legales:** Utilizando NLP, se puede extraer automáticamente información relevante, como números de factura, fechas o nombres, de documentos empresariales o legales.
- ❖ **Predicción de intenciones de usuario en asistentes virtuales:** Los modelos de NLP pueden ser entrenados para predecir las intenciones de los usuarios en interacciones con asistentes virtuales, lo que permite una mejor comprensión y respuesta a las necesidades del usuario.

2.6.7 Arquitectura de un sistema de NLP

La arquitectura de un sistema de NLP se sustenta en una definición del lenguaje natural por niveles; se utilizan seis niveles de comprensión con el objetivo de descubrir el significado del discurso. Estos niveles son: fonológico, morfológico, sintáctico, semántico, discursivo y pragmático [39][43].

- ❖ **Nivel Fonológico:** Este nivel se enfoca en cómo las palabras se relacionan con los sonidos que representan. Aquí se presta atención a la fonética, la forma en que las palabras son pronunciadas. Este nivel es importante cuando procesamos la palabra hablada, no así cuando trabajamos con texto escrito [39][43].
- ❖ **Nivel Morfológico:** Aquí se observa cómo las palabras se construyen a partir de unas unidades de significado más pequeñas llamadas morfemas. Aquí nos interesa realizar

un análisis morfológico del discurso; estudiar la estructura de las palabras para delimitarlas y clasificarlas [39][43].

- ❖ Nivel Sintáctico: Trata de cómo las palabras pueden unirse para formar oraciones, fijando el papel estructural que cada palabra juega en la oración y que sintagmas son parte de otros sintagmas; o dicho de otra manera aquí se realiza un análisis de sintaxis, el cual incluye la acción de dividir una oración en cada uno de sus componentes [39][43].
- ❖ Nivel Semántico: Se observa significado de las palabras y cómo los significados se unen para dar significado a una oración, también se refiere al significado independiente del contexto, es decir de la oración aislada. Este nivel es un complemento del anterior, en el análisis semántico se busca entender el significado de la oración. Las palabras pueden tener múltiples significados, la idea es identificar el significado apropiado por medio del contexto de la oración [39][43].
- ❖ Nivel Pragmático: Aquí se analizan las oraciones en distintas situaciones y cómo su uso afecta el significado de las mismas. Este nivel se ocupa del análisis de oraciones y también de cómo su significado cambia dependiendo de la situación [39][43].
 - Se reconoce un subnivel recursivo: El discursivo, que trata de cómo el significado de una oración se ve afectado por las oraciones inmediatamente anteriores. El nivel discursivo examina el significado de la oración en relación a otra oración en el texto o párrafo del mismo documento [39][43].

Todos los niveles descritos aquí son inseparables y se complementan entre sí. El objetivo de los sistemas de NLP es incluir estas definiciones en una computadora y luego usarlas para crear una oración estructurada y sin ambigüedades con un significado bien definido [39].

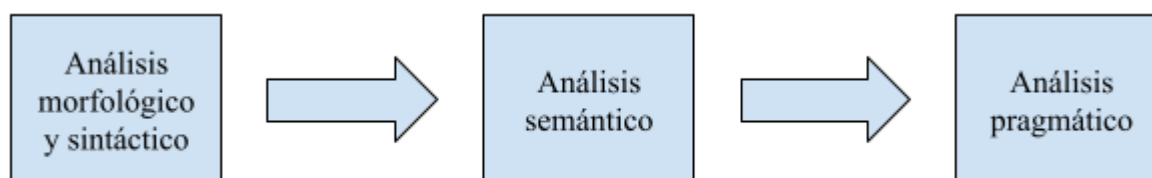


Figura 2.6.3. Arquitectura de un sistema de NLP.

La figura 2.6.3 [43] ilustra la arquitectura del sistema de procesamiento del lenguaje natural, que permite a la computadora interpretar y analizar las oraciones proporcionadas. Este sistema consta de varios pasos, como se explica en detalle [43]:

1. El usuario le indica a la computadora lo que desea hacer.
2. La computadora analiza las oraciones proporcionadas, evaluando su morfología y sintaxis para determinar si cumplen con las reglas gramaticales. En esta etapa, el analizador léxico (scanner) identifica los componentes léxicos previamente definidos, mientras que el analizador sintáctico (parser) verifica si se cumple el orden gramatical correcto entre los elementos identificados por el scanner.
3. A continuación se realiza un análisis semántico de las oraciones para comprender el significado de cada una y asignarles expresiones lógicas.

4. Después, se realiza un análisis pragmático de las oraciones, teniendo en cuenta su contexto y las oraciones anteriores, para determinar la expresión final que la computadora debe realizar.
5. Finalmente, se ejecuta la expresión final para obtener el resultado y proporcionar al usuario.

En resumen, el sistema de procesamiento del lenguaje natural sigue un proceso riguroso para interpretar y analizar las oraciones proporcionadas por el usuario, lo que permite una mayor interacción entre la persona y la computadora en distintas aplicaciones, como los chatbots y los asistentes virtuales.

Los elementos comunes de cualquier arquitectura estándar de un sistema para el Procesamiento del Lenguaje Natural son [39]:

- ❖ Reconocimiento de voz: Convertir una palabra hablada en un conjunto de palabras. Las palabras habladas se componen de una serie de parámetros relacionados con el sentido de la audición.
- ❖ Comprensión del lenguaje: El objetivo de este elemento es generar un significado para las palabras habladas que posteriormente será utilizado por la gestión del diálogo.
- ❖ Gestión del diálogo: La tarea principal de este elemento es coordinar y mantener unidas todas las partes del sistema y conectarse con otros sistemas.
- ❖ Comunicación con sistemas externos: Como sistemas expertos, sistemas de bases de datos u otras aplicaciones informáticas.
- ❖ Generación de respuesta: Establecer el mensaje que el sistema debe entregar.
- ❖ Salida de voz: Uso de diferentes técnicas para producir el mensaje desde el sistema.

2.6.8 Corpus lingüístico

Hoy en día, es indispensable el uso de buenos recursos lingüísticos para el desarrollo de los sistemas de NLP. Estos recursos son esenciales para la creación de gramáticas, en el marco de aproximaciones simbólicas; o para llevar a cabo la formación de módulos basados en el aprendizaje automático [39].

Un corpus lingüístico es un conjunto amplio y estructurado de ejemplos reales de uso de la lengua. Estos ejemplos pueden ser textos (los más comunes) o muestras orales (generalmente transcritas), conformando un conjunto de textos relativamente grande, creado independientemente de sus posibles formas o usos. Es decir, en cuanto a su estructura, variedad y complejidad, un corpus debe reflejar una lengua, o su modalidad, de la forma más exacta posible y en cuanto a su uso, preocuparse de que su representación sea real. La idea es que representen al lenguaje de la mejor forma posible para que los modelos de NLP puedan aprender los patrones necesarios para entender el lenguaje. Encontrar un buen corpus sobre el cual trabajar no suele ser una tarea sencilla [39].

2.7 Sistemas de apoyo a la toma de decisiones

Los sistemas de apoyo a la toma de decisiones (DSS) constituyen una clase de alto nivel de sistemas de información computarizada. Los DSS coinciden con los sistemas de información gerencial en que ambos dependen de una base de datos para abastecerse de información. Sin

embargo, difieren en que el DSS pone énfasis en el apoyo a la toma de decisiones en todas sus fases, aunque la decisión definitiva es responsabilidad exclusiva del encargado de tomarla. Los sistemas de apoyo a la toma de decisiones se ajustan más al gusto de la persona o grupo que los utiliza que a los sistemas de información gerencial tradicionales [44].

2.7.1 Explotación de las bases de datos

Las bases de datos se construyen con el propósito de almacenar una gran cantidad de datos, ya sea en organizaciones pequeñas o en grandes empresas. En la actualidad, la generación de datos es un proceso que brinda enormes cantidades de información diaria en cualquier tipo de organización. Estos datos se han convertido en la fuente de nuevas herramientas para el aprovechamiento de la información contenida en las bases de datos que además, están evolucionando rápidamente, a la par de las tecnologías que las respaldan. Con el aumento del ancho de banda en las conexiones, sistemas de almacenamiento masivo, procesadores de alta velocidad y profesionales más capacitados en el manejo de datos, las tecnologías emergentes en la explotación de bases de datos se están desarrollando a una velocidad vertiginosa.[45].

Los almacenes de datos o Data Warehouse constituyen la arquitectura base para el sistema de soporte a la toma de decisiones y la minería de datos, debido a que son el acúmulo de datos que la organización requiere para sus gestiones de cada día, como pueden ser ventas, compras, pedidos, trazabilidad, producción, logística, marketing, gestión de personal, etc [45]. Los sistemas de soporte a la toma de decisiones van más allá de ser simples sistemas de información para el procesamiento y almacenamiento de datos. Son sistemas que utilizan técnicas emergentes de análisis de datos almacenados con el fin de tomar decisiones más informadas. Estos sistemas permiten, por ejemplo, determinar qué productos son más rentables, cómo mejorar la rentabilidad de productos con menor desempeño, identificar a quién y cómo se adquieren estos productos, analizar la demanda en distintos países, desarrollar estrategias para incrementar las ventas mejorando la eficiencia y la velocidad, así como reducir la necesidad de intermediarios. Las tomas de decisiones no tienen por qué ser siempre comerciales, pueden ser políticas, sociales, de salud, de prevención de riesgos, de control de amenazas naturales o humanas, y un largo etcétera [45].

Los Data Warehouse aportan una metodología para el análisis de datos masivos. A veces, es necesario aglutinar esa ingesta de datos en divisiones más pequeñas con el fin de centrarlos por temas o áreas de negocio, y así poder tomar decisiones más localizadas. Este subconjunto de datos de todo el Data Warehouse se denomina Data Mart. Se consultan mediante herramientas OLAP (On Line Analytical System) para la construcción de sistemas de información para directivos (EIS) y para los sistemas de ayuda a la toma de decisión [45].

En la actualidad, los algoritmos que analizan esos datos usan la inteligencia artificial y los esquemas de los sistemas expertos. Los sistemas expertos son un área de la inteligencia artificial que simulan el proceso de aprendizaje. Estos almacenan datos y conocimiento con el fin de obtener conclusiones y ayudar a tomar decisiones [45].

2.7.2 Definición

Los sistemas de soporte a la decisión son una herramienta fundamental dentro de la inteligencia de negocio, ya que su objetivo principal es el análisis de datos. Estos sistemas se

utilizan para explotar bases de datos de gran tamaño, como Data Warehouse corporativos o Data Marts, que pueden ser fuentes de información muy valiosa para la toma de decisiones empresariales. Los programas informáticos que conforman estos sistemas utilizan diversos métodos analíticos, tales como algoritmos de optimización, análisis de decisiones y programas de planificación de rutinas, entre otros. Estos métodos permiten resolver los desafíos inherentes a los programas de gestión de generaciones anteriores y brindan una mayor capacidad para tomar decisiones basadas en datos.

Además, los sistemas de soporte a la decisión trabajan en conjunto con otras herramientas de inteligencia de negocio, como los cuadros de mando y los sistemas de información ejecutiva. En conjunto, estas herramientas forman un grupo integral para el análisis y la toma de decisiones estratégicas en el ámbito empresarial. Algunos de los tipos de sistemas de soporte a decisiones más utilizados en la actualidad pueden ser [45]:

- ❖ Sistemas de apoyo a decisiones de grupo.
- ❖ Sistemas de información ejecutiva.
- ❖ Sistemas de información gerencial.
- ❖ Sistemas expertos basados en inteligencia artificial.

2.7.3 Componentes de un sistema de soporte a la decisión

Un DSS generalmente consta de varios componentes principales que trabajan juntos para ayudar en el proceso de toma de decisiones. Estos componentes pueden variar dependiendo del diseño y la implementación del DSS, pero los principales son [149]:

- ❖ Base de Datos: Es el repositorio de datos que el DSS utiliza para acceder a la información necesaria para el análisis y la toma de decisiones. Puede ser una base de datos interna de la empresa o datos externos adquiridos de fuentes confiables.
- ❖ Modelos Analíticos: Son las herramientas y técnicas analíticas que se utilizan para analizar los datos y generar información relevante para la toma de decisiones. Esto puede incluir modelos matemáticos, algoritmos de minería de datos, técnicas de análisis estadístico, simulaciones, entre otros.
- ❖ Interfaz de Usuario: Es la interfaz a través de la cual los usuarios interactúan con el DSS. Puede ser una interfaz de usuario o una interfaz de línea de comandos, y permite a los usuarios ingresar datos, realizar análisis, visualizar resultados y realizar acciones de toma de decisiones.
- ❖ Herramientas de Visualización: Son las herramientas que permiten la visualización de datos y resultados de análisis de una manera comprensible. Pueden incluir gráficos, tablas, mapas u otras representaciones visuales de datos.
- ❖ Motor de Toma de Decisiones: Es el componente que utiliza los datos, modelos analíticos y las preferencias del usuario para generar recomendaciones o apoyar en la toma de decisiones. Puede incluir reglas de decisión, algoritmos de optimización, técnicas de análisis multicriterio, entre otros.

Capítulo 3: Estado del arte

3.1 Análisis de las Publicaciones Mundiales, Argentinas y de Chubut sobre PLF

En el campo de la Ganadería de Precisión, la utilización de nuevas tecnologías como el Internet de las Cosas está transformando la forma en que se gestiona y monitorea el ganado en todo el mundo. A continuación se analizará la producción científica existente sobre el tema, tanto a nivel global como local.

3.1.1 ¿Cuáles son los países líderes en investigación?

Como hemos visto a lo largo de esta tesina, la ganadería de precisión es un conjunto de técnicas y tecnologías que permiten la obtención de información de los animales y su entorno para mejorar la producción y el bienestar animal. En los últimos años, ha habido un aumento significativo en la producción científica relacionada con tecnologías de agricultura y ganadería de precisión. Una exploración realizada en 2019 reveló la existencia de más de 2600 publicaciones entre 2014 y 2019, tal como se muestra en la figura 3.1.1 [52]:

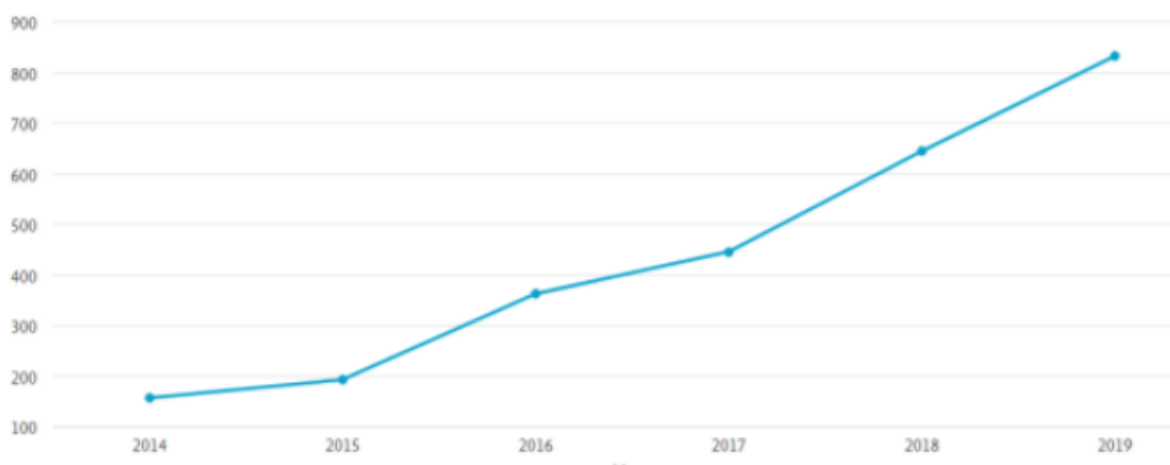


Figura 3.1.1. Evolución de investigaciones científicas por año.

Los principales indicadores de esta exploración muestran que la investigación en este tema ha tenido una tendencia de crecimiento positivo en los últimos años, siendo 2019 el año con la mayor cantidad de publicaciones. Las revistas científicas más relevantes en este campo son: Computers and Electronics in Agriculture (189 publicaciones), Nongye Gongcheng Xuebao Transactions of the Chinese Society of Agricultural Engineering (76 publicaciones), Sensors Switzerland (74 publicaciones), International Journal of Innovative Technology and Exploring Engineering (59 publicaciones) y International Journal of Recent Technology and Engineering (49 publicaciones), tal como se puede observar en la figura 3.1.2 [52].

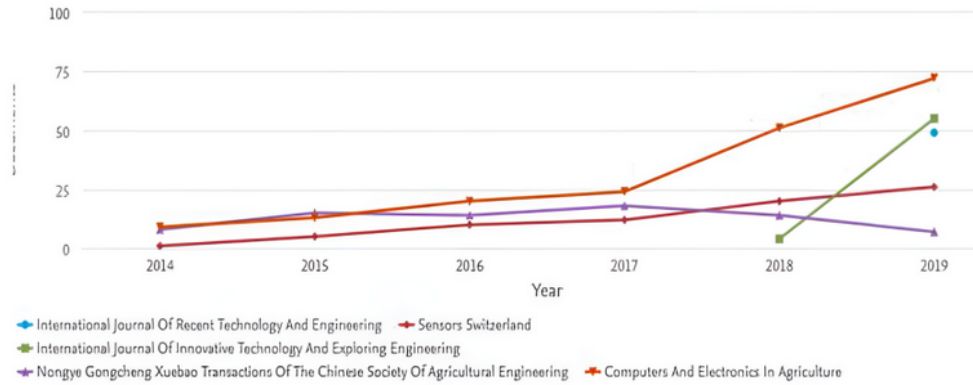


Figura 3.1.2. Cantidad de publicaciones en principales revistas científicas.

Además, se identificaron los países líderes en investigación en este tema, encabezados por China, Estados Unidos, India, Australia y Reino Unido, según el ranking de publicaciones científicas así como se muestra en la figura 3.1.3 [52].

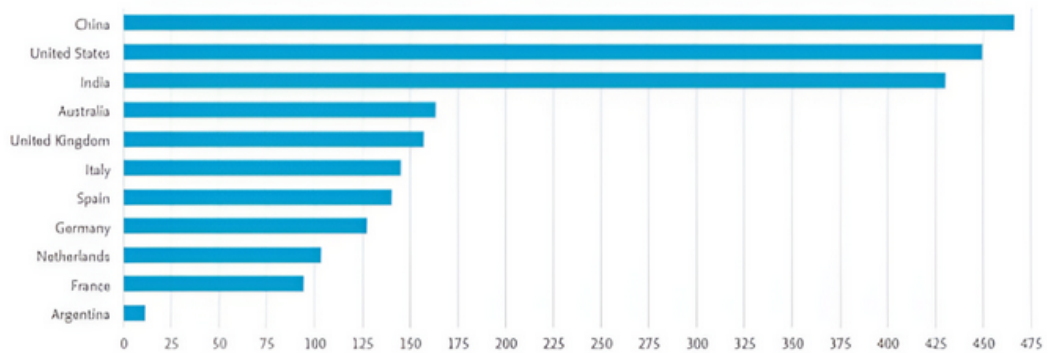


Figura 3.1.3. Países líderes en investigación.

Aunque Argentina figura más abajo en el ranking, la producción científica en este país también está presente en este campo con 11 publicaciones [52].

En una investigación mas reciente, se analizaron los artículos relacionados con PLF publicados en todo el mundo durante 2022, los cuales se ramificaron en cuatro especies: bovino (azul), porcino (naranja), ovino y caprino (verde) y aves (amarillo). Los resultados indican que los países que encabezaron el ranking general de publicaciones científicas en 2019 siguen liderando actualmente, siendo China, Estados Unidos, India y Australia los primeros, seguidos por el Reino Unido [11].

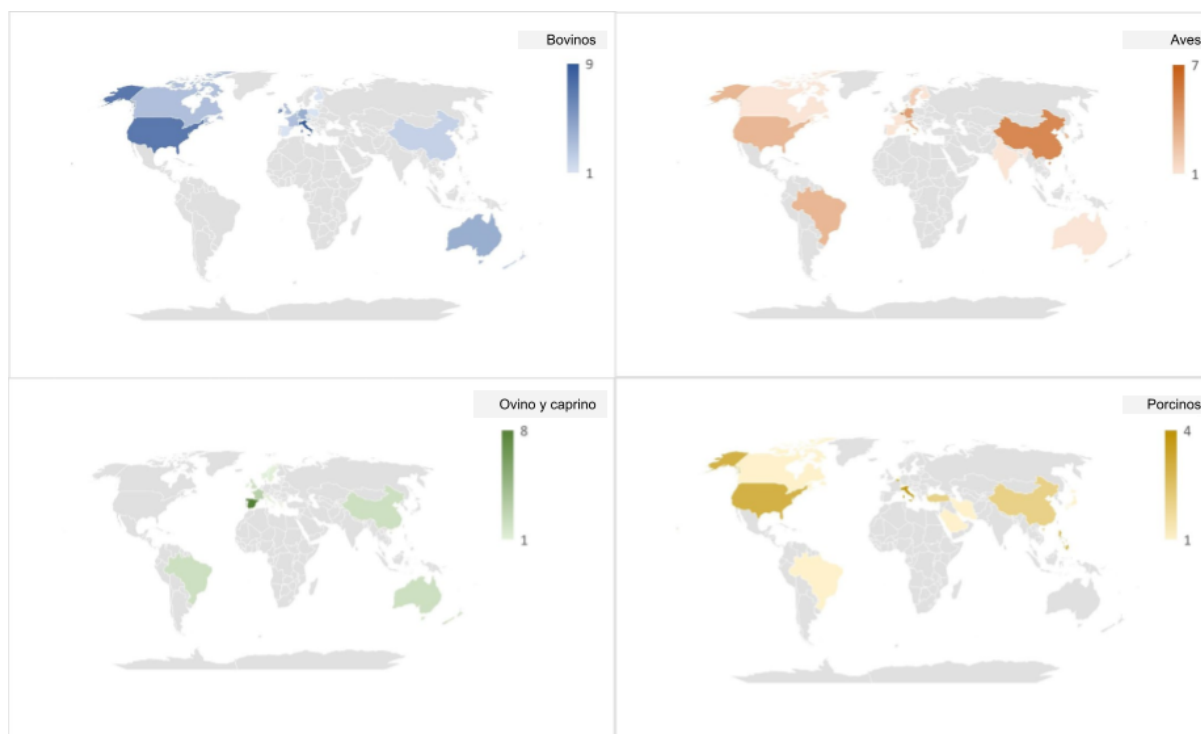


Figura 3.1.4. Número de artículos relacionados con PLF publicados a nivel mundial en relación con la especie.

En la figura 3.1.4 [11] Argentina no figura en el mapa de publicaciones, lo que indica una presencia relativamente baja en el campo de la PLF en comparación con otros países.

3.1.2 ¿Cómo se ha investigado PLF en Argentina?

En Argentina, se ha llevado a cabo investigación en el ámbito de la ganadería de precisión principalmente en el marco de instituciones como el INTA y universidades. Con el propósito de visibilizar el aporte de la investigación Argentina, se confeccionó la tabla 3.1.1 con datos de [52] donde se identificaron las siguientes 11 publicaciones:

Tabla 3.1.1. Publicaciones PLF en Argentina.

TÍTULO DEL DOCUMENTO	AUTORES	AÑO	ACCESO
Innovation obstacles in an emerging high tech sector: The case of precision agriculture in Argentina	Lachman J., López A.	2019	Privado
Estimating body condition score in dairy cows from depth images using convolutional neural networks, transfer learning and model ensembling techniques	Alvarez J.R., Arroqui M., Mangudo P., Toloza J., Jatip D., Rodriguez J.M., Teyseyre A., Sanz C., Zunino A., Machado C., Mateos C.	2019	Público

TÍTULO DEL DOCUMENTO	AUTORES	AÑO	ACCESO
Prediction of Frost Events Using Machine Learning and IoT Sensing Devices	Diedrichs A.L., Bromberg F., Dujovne D., Brun-Laguna K., Watteyne T.	2018	Público
Body condition estimation on cows from depth images using Convolutional Neural Networks	Rodríguez Alvarez J., Arroqui M., Mangudo P., Toloza J., Jatip D., Rodríguez J.M., Teyseyre A., Sanz C., Zunino A., Machado C., Mateos C.	2018	Público
Drought and temperature limit tropical and temperate maize hybrids differently in a subtropical region	Casali L., Rubio G., Herrera J.M.	2018	Público
A flexible and practical approach for real-time weed emergence prediction based on Artificial Neural Networks	Chantre G.R., Vigna M.R., Renzi J.P., Blanco A.M.	2018	Privado
Using SmartMesh IP in Smart Agriculture and Smart Building applications	Brun-Laguna K., Diedrichs A.L., Dujovne D., Taffernaberry C., Léone R., Vilajosana X., Watteyne T.	2018	Público
A pattern recognition approach for detecting and classifying jaw movements in grazing cattle	Chelotti J.O., Vanrell S.R., Galli J.R., Giovanini L.L., Rufiner H.L.	2018	Público
Machine learning applied to the prediction of citrus production	Díaz I., Mazza S.M., Combarro E.F., Giménez L.I., Gaiad J.E.	2017	Público
Assessing groundwater pollution hazard changes under different socio-economic and environmental scenarios in an agricultural watershed	Lima M.L., Romanelli A., Massone H.E	2015	Público
Collar con tecnología GPS para monitoreo animal	E. Gorandi, N. Clemares, A. Moltoni	2015	Público

A continuación, brindaremos un breve resumen que permitirá obtener una idea del tema de estas investigaciones nacionales.

Una de las investigaciones, llevada a cabo por Lachman y López [96], examina los obstáculos que afectan el crecimiento y la innovación en empresas de agricultura de precisión en Argentina. El trabajo concluyó en que los obstáculos de mercado y de costos tienen un impacto negativo, mientras que las barreras de conocimiento tienen un impacto positivo en la relevancia de las actividades de I+D, es decir, cuanto mayores son los obstáculos relacionados con el conocimiento, mayor es la importancia que se otorga a la inversión en I+D para superarlos y así mejorar el desempeño de las empresas en este sector; y además se identificaron diferentes estrategias que las empresas han puesto en marcha para superar las barreras que enfrentan.

En otra investigación se ha propuesto un sistema automático basado en técnicas de aprendizaje automático para mejorar la precisión de la estimación de la BCS. El sistema mejorado ha logrado buenos resultados en comparación con el sistema base, aumentando la precisión global de las estimaciones de BCS. La BCS es un método para evaluar las reservas de grasa corporal y el balance energético acumulado de las vacas lecheras, pero su calificación es subjetiva y requiere mucho tiempo [97].

En otro artículo se describe un componente de un sistema de predicción de heladas basado en IoT. Este sistema utiliza algoritmos de aprendizaje automático entrenados por lecturas pasadas de sensores de temperatura y humedad para predecir temperaturas futuras, pero también incorpora información termodinámica del entorno para mejorar la precisión de las estimaciones. Se desarrolló un modelo para cada ubicación, incluyendo en su entrenamiento información de lecturas de sensores de todas las demás ubicaciones, seleccionando de forma autónoma las más relevantes. Los modelos se entrenaron utilizando varios algoritmos de aprendizaje automático sobre datos de cinco estaciones meteorológicas en la provincia de Mendoza, Argentina, utilizando la *Técnica de Sobremuestreo de Minorías Sintéticas (SMOTE)* para aumentar los datos escasos de eventos de heladas. Los resultados experimentales mostraron que la selección de los vecinos más relevantes y el entrenamiento con SMOTE reducen los errores de predicción y mejoran el rendimiento de los predictores de regresión y clasificación. Los autores concluyen que este enfoque es un recurso válido y útil para los responsables de la toma de decisiones y los productores [98].

También existe una segunda investigación que habla sobre cómo el método BCS se utiliza para medir la condición corporal de las vacas y su importancia para su producción, reproducción y salud. Varios estudios han intentado automatizar el proceso utilizando técnicas de aprendizaje automático, y este trabajo propone un sistema basado en Redes Neuronales Convolucionales (CNNs) que ha obtenido buenos resultados. El sistema alcanzó una precisión del 78% en la estimación de BCS dentro de 0,25 unidades de diferencia respecto a los valores reales [99].

Además, se realizó un estudio que indica que aunque la región semiárida y subhúmeda del Chaco en Argentina se considera inadecuada para el cultivo de maíz, está siendo cada vez más adoptada. Los modelos mixtos multivariantes identificaron que la cantidad de precipitaciones durante febrero es el determinante principal del rendimiento del maíz, y los híbridos templados de maíz tienen el potencial de aumentar el rendimiento de grano por cada milímetro de lluvia durante febrero. Esto proporcionará información valiosa para los

fitomejoradores para obtener genotipos de maíz mejor adaptados al Chaco semiárido y subhúmedo [100].

También se presentó un enfoque basado en *Redes Neuronales Artificiales (RNA)* para predecir el surgimiento de malas hierbas, utilizando datos meteorológicos fácilmente disponibles. Los modelos populares de predicción de la emergencia requieren parámetros poblacionales específicos y condiciones microclimáticas del suelo, que son difíciles de estimar. En cambio, el enfoque basado en RNA proporciona una opción flexible para el desarrollo de modelos de predicción que pueden adaptarse a especies que muestran un patrón de aparición muy distribuido a lo largo del año [101].

En otro estudio se desplegaron dos redes inalámbricas de baja potencia en un huerto de duraznos y un edificio inteligente, ambas utilizando la tecnología SmartMesh IP, con nodos que presentan una fiabilidad de extremo a extremo del 100% y una vida útil prevista de entre 4 y 8 años. Se demostró que los enlaces inalámbricos son simétricos gracias al uso de *Salto de Canal en Franjas Horarias (TSCH)*, y se concluyó que TSCH es una solución IoT adecuada para aplicaciones de agricultura y edificios inteligentes [102].

Sabemos que uno de los objetivos de la ganadería de precisión es el seguimiento continuo en tiempo real de la salud y bienestar de los animales. La monitorización acústica es una forma práctica de evaluar el comportamiento alimentario, pero los algoritmos actuales tienen limitaciones. En una investigación se creó un nuevo algoritmo que muestra resultados prometedores al utilizar técnicas de reconocimiento de patrones y aprendizaje automático para mejorar la capacidad de discriminación en entornos ruidosos y con un margen mínimo de costo computacional [103].

Otro estudio propone un sistema inteligente para predecir el rendimiento de los cítricos antes del periodo de cosecha utilizando datos históricos de campo y espectrales de imágenes de satélite. Para ello se construyó un conjunto de datos de 250 filas y se probaron varios algoritmos de aprendizaje automático, obteniendo buenas puntuaciones de predicción; el enfoque del estudio puede ser utilizado en otros cultivos arbóreos [104].

Hay otro artículo que propone un modelo para evaluar los cambios en el riesgo de contaminación de las aguas subterráneas en dos escenarios socioeconómicos y medioambientales diferentes. El Escenario de Sostenibilidad Relegada mostró un cambio negativo en el sistema acuífero, mientras que el Escenario de Reformas de la Sostenibilidad mostró cambios más equilibrados en el uso del suelo con una tendencia hacia la sostenibilidad. Los resultados en el escenario observado en 2011, mostraron que el 54% de la cuenca del arroyo Dulce sigue mostrando una contribución de moderada a muy baja al peligro de contaminación de las aguas subterráneas [105].

En la figura 3.1.5 [52] se observa que el Consejo Nacional de Investigaciones Científicas y Técnicas lidera la cantidad de investigaciones argentinas con un total de 10, seguido por Doctores en Universidades para Transferencia Tecnológica (D-TEC), Universidad de Buenos Aires, Universidad Nacional del Centro de la Provincia de Buenos Aires, Institut National de Recherche en Informatique et en Automatique (INRIA), Universidad Diego Portales, Universidad Tecnológica Nacional y Comisión de Investigaciones Científicas - La Plata, todos ellos con 2 investigaciones cada uno. Además, FICH-UNL/CONICET y Universitat Oberta de Catalunya también aparecen en la lista con una investigación cada uno.

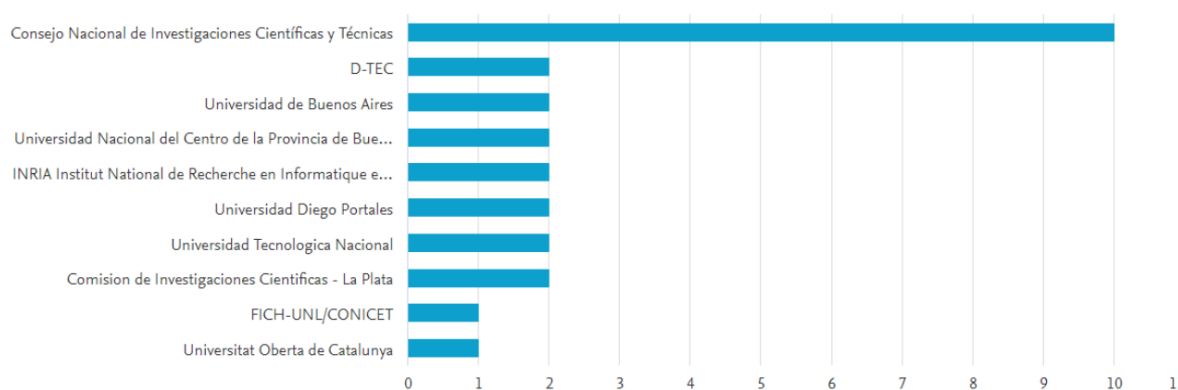


Figura 3.1.5. Instituciones de investigaciones Argentinas.

En general, la investigación en el campo de la PLF en Argentina se ha enfocado en la aplicación de tecnologías como el monitoreo remoto, la teledetección, los sistemas de información geográfica y el análisis de datos para mejorar la gestión y eficiencia de la producción ganadera.

3.1.3 PLF en Chubut: ¿Qué se ha publicado hasta ahora?

En 2016 se llevó a cabo un estudio sobre cómo la *Identificación por Radiofrecuencia (RFID)* puede tener un impacto positivo frente a los problemas que los productores presentan al momento de cargar los datos de la esquila. El proyecto consiste en un sistema de trazabilidad de la producción, el cual se inicia con el registro de la información proveniente de la esquila y culmina con la posibilidad de utilización comercial de la información y de gestión de productores individuales y cooperativas de productores, entre otros [22].

Este proyecto busca brindar a los productores ovinos de la Región Patagónica un mecanismo semiautomático de registro electrónico del proceso de esquila. Dicho registro se llevará a cabo mediante un equipo acoplable a una báscula electrónica, para el pesaje de los fardos de lana producidos. Además, el equipo deberá ser compatible con un mecanismo de impresión y poseer autonomía energética. Este equipo permitirá cumplir con todos los requisitos de registro que tiene PROLANA, así como minimizar los posibles errores en la carga y manipulación de la información de los romaneos [22].

También en 2016 se llevó a cabo una tesis enfocada principalmente en las redes de diálogo en los sistemas de producción de conocimiento de las poblaciones rurales dispersas de Gualjaina (Chubut). Para ello se realizó una investigación en la comunidad de Sierras de Gualjaina como unidad social inserta en el sistema rural territorial, para reconocer a los actores que intervienen en la producción de conocimiento para la actividad ganadera ovina y caprina, y las redes de diálogo que establecen estos actores. Realizando entrevistas semiestructuradas a productores, técnicos y agentes privados presentes en la comunidad, la observación participante y la revisión de fuentes primarias y secundarias, se obtienen los datos a partir de los cuales se hace una aproximación al territorio, se identifican las características de los principales capitales con que cuentan los integrantes de la comunidad y las relaciones que entablan [90].

3.2 Transformando la Producción Ganadera con la PLF

La ganadería de precisión es un campo de investigación y desarrollo que ha experimentado un rápido crecimiento a nivel mundial en los últimos años, como se puede observar en las publicaciones científicas revisadas anteriormente. Actualmente, existen diversas tecnologías que se están utilizando para mejorar la eficiencia en la producción de carne y leche, reducir los costos de producción, minimizar el impacto ambiental de la ganadería, reducir la producción de desechos, entre otros. Entre estas tecnologías se incluyen sensores, dispositivos de monitoreo, sistemas de alimentación automatizados, sistemas de identificación de animales, y software de análisis de datos. Además, la implementación de técnicas de análisis predictivo, inteligencia artificial y aprendizaje automático está permitiendo a los productores tomar decisiones más informadas y mejorar la gestión de sus sistemas de producción. A continuación se analizarán las herramientas más innovadoras y relevantes que se han desarrollado para la ganadería de precisión.

3.2.1 ¿Cómo está revolucionando PLF a la industria ganadera global?

La tecnología ha evolucionado con el tiempo y se ha convertido en un aliado fundamental para la producción ganadera. Las TICs han hecho posible recopilar y procesar datos en tiempo real sobre el comportamiento y la salud de los animales, permitiendo predecir y describir el comportamiento asociado a una producción animal más eficiente mediante el uso de Machine Learning. Actualmente, la ganadería de precisión se ha consolidado como una solución tecnológica capaz de satisfacer la creciente demanda de alimentos a nivel global, especialmente en el sector ganadero, permitiendo el desarrollo de modelos de producción más sostenibles y eficientes que garantizan la sanidad y el bienestar de los animales. La aplicación de esta tecnología también permite mantener un número óptimo de animales por explotación, encontrar soluciones rápidas a sus enfermedades y definir un modelo de producción más eficiente [14]. Además, el uso de sensores para el seguimiento animal se ha vuelto altamente efectivo para recoger información sobre el comportamiento de los animales, lo que permite determinar sus necesidades individuales y brindar atención personalizada y óptima para mejorar la producción. Los sensores son cruciales para la ganadería, ya que permiten monitorear y optimizar el rendimiento del ganado de manera efectiva en aspectos como el **bienestar** y la **sanidad animal**, así como también para medir el **peso de los animales** y ayudar a los operarios y **personal técnico** en sus tareas y toma de decisiones. Actualmente, existe una amplia variedad de sensores disponibles para la ganadería, lo que brinda a los productores la capacidad de aplicar técnicas de ganadería de precisión y lograr una producción más eficiente y sostenible.

Bienestar animal

El bienestar animal es un tema complejo y multifacético en el que intervienen dimensiones científicas, éticas, económicas, culturales, sociales, religiosas y políticas. Hace algunas décadas atrás, el objetivo principal de una política de bienestar animal era mejorar las condiciones de cría, transporte y sacrificio, entre otras, para respetar las necesidades psicológicas y el comportamiento de los animales, mejorando así su productividad. Este

objetivo sigue siendo adecuado en muchos países. Sin embargo, hoy en día los paradigmas han cambiado y las nuevas consideraciones conllevan a adoptar otra visión sobre las problemáticas en materia de bienestar animal, en particular respecto de la producción animal, donde la conciencia de una coherencia necesaria entre bienestar animal y desarrollo sostenible de la ganadería se está gestando y ciertamente contribuirá al desarrollo de nuevas prácticas mucho más respetuosas de los animales. Cabe destacar que en los países más favorecidos el consumidor no sólo se preocupa por el precio de los productos o la salubridad de su alimentación, sino que también se interesa por la manera de producción de los alimentos y en particular los métodos de cría, transporte y sacrificio de los animales. Esto muestra cómo el bienestar animal se relaciona con el bienestar del hombre, la diversidad y el medio ambiente en diferentes niveles de la sociedad. Además, existe una estrecha relación entre el bienestar animal y la productividad en la industria ganadera. Los animales que reciben un cuidado adecuado tienden a ser más rentables, ya que su salud y bienestar influyen directamente en su productividad. Por ejemplo, los animales que se mantienen sanos y bien cuidados pueden reducir los costos de producción al minimizar la necesidad de tratamientos médicos y medicamentos. Asimismo, un manejo adecuado de los animales puede resultar en un aumento en la producción, ya que los animales que se sienten seguros y cómodos tienen un mejor desempeño en términos de crecimiento y rendimiento. Por lo tanto, garantizar un buen nivel de bienestar animal es esencial no solo desde una perspectiva ética, sino también desde un punto de vista económico en la producción ganadera [65][66].

La evaluación del bienestar animal es así uno de los pilares de sistemas de producción, eficientes y sostenibles y debe realizarse de manera válida y fiable en todo el mundo. En 2015, se finalizó el proyecto AWIN con la publicación de cinco protocolos de evaluación del bienestar en las explotaciones centrados en indicadores basados en el animal para cabras, ovejas, burros, caballos y pavos. Estos protocolos, que han sido comprobados por su validez y confiabilidad, se desarrollaron con el objetivo de que fueran viables en condiciones de granja, en términos de duración, costo y facilidad de evaluación. Se estimó necesario un sistema digitalizado con el fin de mejorar la eficiencia de la colecta de datos, reduciendo el tiempo requerido y los errores de transcripción y permitiendo a la vez ingresar los datos de manera automática. Por consiguiente, se desarrollaron cuatro aplicaciones móviles disponibles para dispositivos Android. I-WatchTurkey (para la cría intensiva de pavos), AWINHorse (para caballos adultos estabulados en caballerizas individuales), AWINGoat (para cabras lecheras en sistemas intensivos), y AWINSheep (para ovejas adultas en sistemas de producción de leche y/o carne) se diseñaron específicamente para guiar al usuario paso por paso durante la visita a la explotación, sobre la colecta de datos en la granja y ofrecer inmediatamente resultados visuales de la evaluación. Los veterinarios y técnicos pueden servirse de las aplicaciones AWIN en su práctica diaria, el resultado en tiempo real puede facilitar el diálogo con los productores, sugiriendo acciones de mejora del bienestar animal. Los productores pueden utilizarlas para comparar la situación de bienestar de sus animales con las de otras explotaciones con sistemas similares de cría o gestión e identificar más fácilmente las causas de un menor rendimiento. Pese a que las aplicaciones AWIN son fáciles de emplear, se requiere una formación específica sobre la manera de evaluar y puntuar los indicadores de bienestar con el fin de armonizar la evaluación y obtener datos fiables. Las

aplicaciones AWIN se desarrollaron para que fueran asequibles y sin conexión, características que contribuyen a un uso extensivo de los protocolos de bienestar animal, sin límites geográficos. Asimismo, incrementan la eficacia y estandarización de la colecta de datos en la granja y preparan el camino para el desarrollo de datos mundiales de bienestar animal accesibles [67].

Sanidad animal

La sanidad animal es un elemento fundamental para el bienestar de los animales. Actualmente, el uso de tecnologías portátiles o sensores en el manejo de la salud animal está ganando cada vez más popularidad, siendo una herramienta importante para mejorar la gestión sanitaria de los animales de manera progresiva. Estos sensores, si están bien diseñados e implementados, pueden ayudar a detectar alteraciones fisiológicas indicando enfermedades y colaborando en la disminución de pérdidas económicas de los productores. A diferencia de las mediciones manuales que requieren mucha mano de obra y pueden variar en precisión debido a la capacitación del evaluador o la falta de ella, una tecnología de sensor ideal debe proporcionar datos confiables, continuos y en tiempo real que reflejen la condición fisiológica de un animal individual. La tecnología de sensores ha demostrado tener éxito en ayudar a mejorar el bienestar de las aves de corral, cerdos, vacas y ovejas [74].

Como parte de PLF, los principios de *Alimentación de Precisión (PF)* se basan en dispositivos de medición en línea, métodos computacionales y dispositivos de alimentación que hacen posible alimentar a los animales individualmente, con la cantidad adecuada de nutrientes en el momento adecuado. La variabilidad de los requerimientos de nutrientes según la etapa fisiológica se ha integrado con éxito en modelos tradicionales de nutrición en cerdos. Sin embargo, no se dispone de modelos predictivos que funcionen en tiempo real, aunque esto es precisamente lo que requieren los sistemas de PF para las cerdas lactantes. Por lo tanto, existe la necesidad de procesos que brinden predicciones fiables del rendimiento a partir de mediciones sencillas [72].

También se está utilizando tecnologías para la detección temprana de cojeras, combinando algoritmos de aprendizaje automático con datos que se recogen de sensores colocados en las patas de los animales, como conteo de pasos, el tiempo de reposo y los cambios por hora. Estos datos se analizan en la nube y las anomalías se envían al dispositivo móvil de los productores para tratar a los animales afectados de inmediato y evitar problemas mayores. A diferencia de un enfoque general, este caso personaliza los modelos de datos para ajustarse a medida que cambian las condiciones climáticas y agrícolas. Al detectar la cojera temprana antes de que pueda ser capturada visualmente, los costos del tratamiento disminuyen mientras se mejora el bienestar animal [73].

Por último, se han utilizado algoritmos de aprendizaje automático, como el reconocimiento de la cara de los cerdos mediante redes neuronales convolucionales, y se han identificado diversos sensores relevantes para medir la salud animal, tales como cámaras 2D y 3D,

micrófonos, termistores y acelerómetros. Estas tecnologías tienen el potencial de mejorar significativamente la salud y el bienestar de los animales de campo [14].

Peso corporal del ganado

En la producción animal, el *Peso Corporal (BW)* del ganado es una característica muy importante y ampliamente utilizada que tiene un impacto significativo en el consumo de alimento, el potencial reproductivo, el comportamiento social, el balance energético y la gestión general de los rebaños. Puede utilizarse indirectamente para evaluar el estado de salud y bienestar de los animales y para determinar el momento de su comercialización. Los cambios grandes o bruscos en el peso corporal pueden indicar la presencia de una enfermedad, condiciones de alojamiento inadecuadas, problemas de bienestar, errores en la alimentación o una selección genética ineficiente. En la actualidad existen dos métodos principales para medir el peso corporal del ganado [60]:

- 1) métodos de pesaje directo que utilizan básculas industriales de peso parcial o peso completo, capaces de soportar ganado pequeño, mediano o grande, y
- 2) métodos indirectos basados en las relaciones entre las medidas de las partes del cuerpo y el peso corporal.

Aunque las básculas son muy precisas, su adquisición, finalidad y tamaño de funcionamiento, calibración repetida y costes de mantenimiento asociados a su colocación en entornos de alta variabilidad térmica y corrosivos son significativos y superan los límites de asequibilidad y sostenibilidad de las explotaciones pequeñas y medianas e incluso de los operadores comerciales. Para sobrellevar esto, los investigadores han desarrollado métodos de pesaje indirecto representados por modelos de regresión que relacionan las mediciones morfométricas y las características de las imágenes tomadas con el BW del ganado [60].

Aunque la predicción del peso corporal de los animales de campo, a partir de mediciones biométricas y morfométricas observadas en diferentes periodos de crecimiento en bovinos, porcinos, ovinos y caprinos ha sido el objetivo de muchos estudios de investigación anteriores, que aplicaron técnicas de regresión estadística tradicionales como la regresión lineal, múltiple y de ridge, su éxito se vio limitado por la dependencia múltiple y las complejas relaciones entre las mediciones (variables) [60].

Si se tiene en cuenta el objetivo de la ganadería mundial de aumentar la densidad, la productividad y el bienestar de los animales y reducir los costes de alimentación y las emisiones de gases de efecto invernadero, la perspectiva de sustituir las mediciones biométricas y morfométricas manuales por mediciones automáticas sin contacto e integrar los resultados en sistemas inteligentes de ML y DL es de gran interés y tiene por objeto optimizar la gestión del ganado y permitir el seguimiento individual de la salud, el bienestar y el crecimiento de los animales en tiempo real [60]. Aunque la investigación actual muestra resultados prometedores predominantemente en ganado vacuno y porcino, todavía hay muchas vías por explorar para una mejor automatización de todo el proceso de estimación de BW. Los resultados actuales también se ven limitados por el reducido tamaño de los conjuntos de datos, su baja calidad, su escasa variabilidad y la falta de información anotada (verdad sobre el terreno) y, lo que es más importante, por la limitada disponibilidad de información que pueda compartirse. Por lo tanto, la creación de repositorios y bases de datos

de acceso público que almacenen imágenes 2D y 3D y las correspondientes mediciones biométricas/morfométricas (y otras complementarias) de las razas ganaderas son esenciales para el desarrollo y la mejora de los enfoques actuales de visión por computadora, Machine Learning y Deep Learning, que tienen un claro potencial para mejorar las estimaciones del peso corporal sin contacto [60].

En la industria ganadera, existen diversas soluciones tecnológicas para el pesaje de animales. Algunas empresas líderes en este sector son GrowSafe de Canadá, Bosch Precision Livestock Platform de Alemania, Rice Lake Weighing Systems de EE. UU. y Diverseco Industrial Scales and Weighing Systems de Australia. Estas empresas ofrecen soluciones de pesaje pasivo mediante sistemas de báscula que cuentan con sensores avanzados, lo que les permite medir el peso de los animales, registrar los resultados y transmitir la información a través de redes cableadas o inalámbricas.

Por otro lado, también existen otras empresas que ofrecen sistemas de pesaje dinámico, como Arvet CIMA Control Pig y CIMA Control Cow Scaling Systems de España, GEA iNTELIWEIGH Walk Over Weigh System de Alemania, así como grupos de investigación que han desarrollado básculas personalizadas. Estos sistemas permiten pesar a los animales en movimiento mediante básculas de tránsito o de paso [60].

Seguimiento de animales

En cuanto al seguimiento activo de animales se han desarrollado nuevas tecnologías para su vigilancia principalmente basadas en algoritmos de aprendizaje automático y visión artificial. La revisión de la literatura [14] destaca cuatro investigaciones que desarrollan éstos tópicos y se enfocan en la monitorización de animales de campo.

En la primera, se analizan los enfoques tecnológicos más importantes para la monitorización de animales, y se muestra cómo el análisis de imágenes y sonidos se puede utilizar para crear representaciones digitales de los mismos. En la segunda, se propone un debate sobre cómo el aprendizaje automático podría utilizarse para monitorizar una variedad de factores de salud y bienestar en vacas lecheras, incluyendo la temperatura interna y superficial, frecuencia respiratoria, tasa de sudoración, patrón de marcha, comportamiento, dimensiones físicas, peso y puntuación de condición corporal. En la tercera revisión, se analizan las implicaciones de las nuevas tecnologías de sensores inteligentes en la producción avícola, y se describe la relación entre la tecnología de sensores, el análisis de grandes volúmenes de datos y el Internet de las cosas, y cómo estas tecnologías pueden mejorar la productividad de la industria avícola. Por último, se presentan métodos para clasificar o priorizar las alertas que se producen en los rebaños, y se sugiere que las investigaciones futuras deberían centrarse en enfoques alternativos de detección y modelos, utilizando la probabilidad previa o el riesgo de que se produzca una condición [14].

A continuación, se analizará cómo se utilizan los diferentes sensores en distintas áreas de interés para cada especie animal. La figura 3.1.1 fue construida en base a datos de [11] y muestra el porcentaje de uso de cada sensor para cada área de interés. Las áreas de interés a considerar son: comportamiento animal, detección de celo, detección de enfermedades, ordeño, desempeño animal, monitoreo de alimentación, identificación y seguimiento. Los

sensores a considerar son: acelerómetros, extracción de cuadros de video, sensor de latidos del corazón, sensor de temperatura de la piel, sensores de temperatura y humedad ambiental, sistemas acústicos, bebederos y comederos electrónicos, identificación por radiofrecuencia (RFID), modelo 3D, sensores de presión, sistemas de localización de banda ultraancho (UWB), termografía infrarroja (IRT), plataforma de pesaje automática, robot AMS, red de área amplia de baja potencia (LPWAN), vehículo aéreo no tripulado (UAV) y puertas electrónicas.

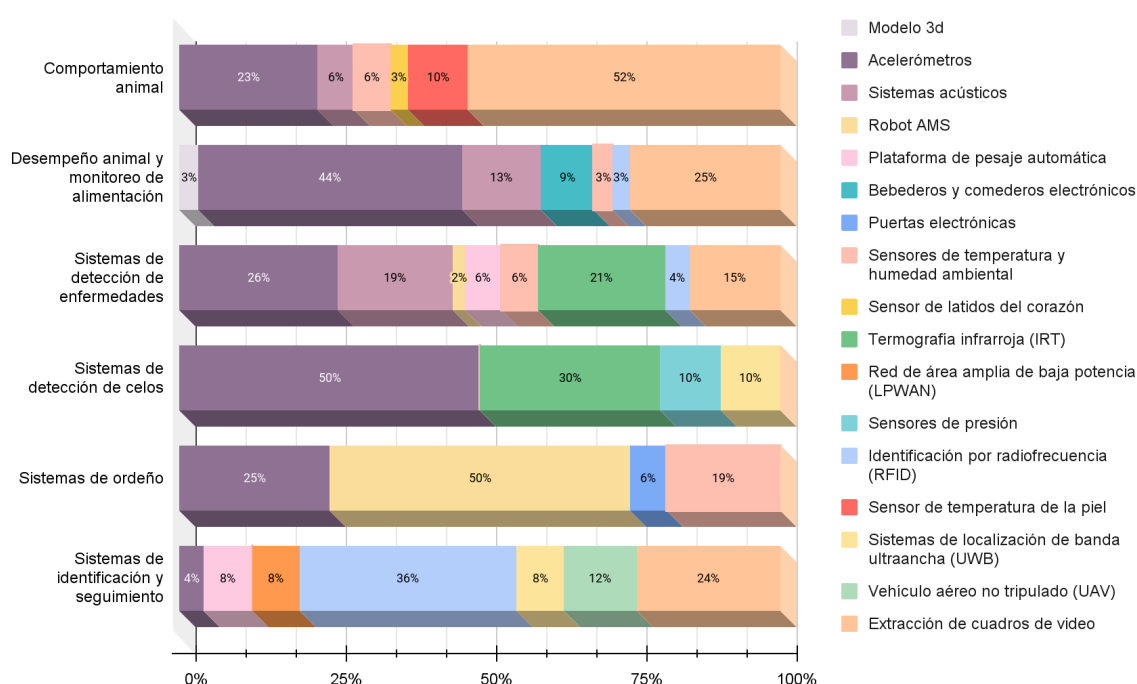


Figura 3.2.1. Porcentaje de sensores más utilizados por cada área de interés en PLF.

Se puede observar en la figura anterior que las tecnologías más utilizadas incluyen acelerómetros, sensores de temperatura y humedad ambiental, termografía infrarroja (IRT), identificación por radiofrecuencia (RFID) y extracción de cuadros de video. Por otro lado, algunas tecnologías se utilizan en un número limitado de áreas, como el robot AMS y la plataforma de pesaje automática. Analicemos los porcentajes para cada tecnología:

- ❖ **Acelerómetros:** Son muy utilizados en comportamiento animal y monitoreo de alimentación, con un 44% de uso en esta área, también se utilizan en sistemas de detección de enfermedades (26%) y en detección de celos (50%). Además, los sistemas de ordeño (25%) también utilizan acelerómetros, lo que sugiere que son una tecnología versátil y útil en diferentes aplicaciones en la ganadería.
- ❖ **Plataforma de pesaje automática:** Es una tecnología con un uso del 8% en sistemas de identificación y seguimiento, lo que sugiere que se utiliza principalmente para el seguimiento del peso de los animales.
- ❖ **Sensores de temperatura y humedad ambiental:** Se utilizan en comportamiento animal (6%), monitoreo de alimentación (3%), detección de enfermedades (6%) y sistemas de ordeño (19%), lo que sugiere que se utilizan en diferentes aplicaciones en la

ganadería y son una tecnología útil para monitorear las condiciones ambientales en las que se encuentran los animales.

- ❖ Termografía infrarroja (IRT): Utilizada principalmente en sistemas de detección de celos (30%) y en detección de enfermedades (21%), siendo una tecnología importante para la detección temprana de estas condiciones en los animales.
- ❖ Identificación por radiofrecuencia (RFID): Se utiliza en sistemas de detección de enfermedades (4%) y en sistemas de ordeño (36%), siendo útil para la identificación y el seguimiento de los animales.
- ❖ Extracción de cuadros de video: Se utiliza principalmente en comportamiento animal (52%) y en sistemas de identificación y seguimiento (24%), lo que sugiere que es una tecnología importante para el monitoreo del comportamiento animal y el seguimiento de los animales.

La figura 3.2.1 muestra que hay una variedad de tecnologías que se utilizan en diferentes aplicaciones en la ganadería, como el monitoreo del comportamiento y la alimentación, la detección temprana de enfermedades y la identificación y seguimiento de los animales. La siguiente tabla 3.2.1 fue confeccionada en base a datos de [11] y resume las tecnologías utilizadas para monitorear los parámetros en las diferentes especies animales. Las especies animales a considerar son: bovinos, porcinos, aves de corral, ovinos y caprinos.

Tabla 3.2.1. Tecnologías utilizadas para monitorear animales.

Áreas de interés	Tecnologías	Bovino	Ovino y Caprino	Porcino	Aves de corral
Comportamiento animal	Acelerómetros	✓	✓		
	Extracción de cuadros de video	✓		✓	✓
	Sensor de latidos del corazón		✓		
	Sensor de temperatura de la piel		✓	✓	
	Sensores de temperatura y humedad ambiental	✓			✓
	Sistemas acústicos		✓	✓	
Desempeño animal y monitoreo de alimentación	Acelerómetros	✓	✓		
	Bebedores y comederos electrónicos			✓	✓
	Extracción de cuadros de video	✓		✓	✓
	Identificación por	✓			

	radiofrecuencia (RFID)				
	Modelo 3d	✓			
	Sensores de temperatura y humedad ambiental			✓	
	Sistemas acústicos	✓			✓
Sistemas de detección de celos	Acelerómetros	✓	✓		
	Sensores de presión		✓		
	Sistemas de localización de banda ultraancha (UWB)	✓			
	Termografía infrarroja (IRT)	✓	✓		
Sistemas de detección de enfermedades	Acelerómetros	✓			✓
	Extracción de cuadros de video	✓	✓		✓
	Identificación por radiofrecuencia (RFID)	✓	✓		
	Plataforma de pesaje automática	✓			
	Robot AMS	✓			
	Sensores de temperatura y humedad ambiental				✓
	Sistemas acústicos	✓		✓	✓
	Termografía infrarroja (IRT)	✓	✓		
Sistemas de identificación y seguimiento	Acelerómetros	✓			
	Extracción de cuadros de video	✓	✓	✓	
	Identificación por radiofrecuencia (RFID)	✓	✓	✓	
	Red de área amplia de baja potencia (LPWAN)	✓			
	Sistemas de localización de banda ultraancha (UWB)	✓	✓		
	Vehículo aéreo no tripulado (UAV)	✓	✓		
Sistemas de ordeño	Acelerómetros	✓	✓		
	Puertas electrónicas		✓		

	Robot AMS	✓	✓		
	Sensores de temperatura y humedad ambiental	✓			

La tabla 3.2.1 proporciona una visión general de las diferentes tecnologías utilizadas en la producción animal, y se observa que hay varias tecnologías que se utilizan en múltiples áreas. Por ejemplo, los acelerómetros son una tecnología ampliamente utilizada en todas las áreas de interés, y la extracción de cuadros de videos es una herramienta valiosa para el seguimiento de ovinos, bovinos y aves en términos de controlar su salud, comportamiento y desempeño animal. Sin embargo, también existen tecnologías específicas para una sola área, como la termografía infrarroja, que se utiliza exclusivamente en los sistemas de detección de celos y enfermedades en bovinos. Además, los robots AMS son más utilizados en bovinos, caprinos y ovinos, lo que indica que esta tecnología es específica para ciertas especies animales. En términos generales, se puede apreciar una amplia variedad de tecnologías utilizadas en la producción animal, aunque algunas resultan más aplicables que otras en función de su uso en distintas áreas y especies animales.

Personal técnico

Con el avance de la ganadería de precisión, el personal técnico y los operadores de campo se especializan en tareas específicas gracias a la información detallada y precisa que tienen a su disposición. Esta información tiene un impacto directo en las decisiones que deben tomar, lo que les permite visualizar qué áreas requieren mejoras. Esta etapa es crucial, ya que se inicia la interacción entre el sistema y el operador para definir los controles y verificaciones necesarias. El sistema no solo genera alertas sobre posibles problemas, sino que también propone soluciones para implementar acciones operativas. La implementación de estas tecnologías en la ganadería de precisión tiene múltiples ventajas a corto y largo plazo, como la optimización de los procesos productivos, la mejora de las condiciones de trabajo y vida de los operarios, la mejora del bienestar animal, el manejo individual de los animales y la satisfacción de las demandas de los consumidores en cuanto a la trazabilidad de los procesos, origen y seguimiento de los productos alimenticios [117].

La ganadería de precisión ofrece una mejora en la eficiencia y sostenibilidad económica, productiva, ambiental y social de la producción. La toma de decisiones se realiza a partir de datos sistematizados y precisos, lo que permite abarcar todo el proceso, desde la detección de preñez hasta la llegada del producto al consumidor final. Además, las tecnologías de ganadería de precisión mejoran las condiciones de trabajo y vida de los operarios, reducen los tiempos y facilitan las tareas de alto esfuerzo. La mejora del bienestar animal es otra ventaja importante, ya que proporciona monitoreos más precisos y un mayor control sobre el comportamiento y estado de salud de los animales, lo que permite anticiparse a episodios de enfermedad. El manejo individual de los animales aumenta la eficiencia productiva y permite una adecuación óptima de los recursos de alimentación, lo que contribuye a la sustentabilidad ambiental y a la reducción de costos económicos. Finalmente, las tecnologías de ganadería de precisión satisfacen las demandas de los consumidores en cuanto a la trazabilidad de los procesos, origen y seguimiento de los productos alimenticios [117].

3.2.2 Ganadería de precisión en Argentina: ¿Realidad o Aspiración?

Uno de los pilares de la Ganadería de Precisión es la industria 4.0, la cual se refiere a la integración de tecnologías digitales, físicas y biológicas para crear sistemas de producción inteligentes y altamente eficientes. En Argentina, la implementación de la Industria 4.0 se ha convertido en un objetivo clave dentro del Plan de Desarrollo Productivo Argentina 4.0, el cual busca transformar la economía del país mediante la modernización y digitalización de la industria [61].

Industria 4.0

En el Siglo XXI, hemos ingresado a la era de la Industria 4.0, donde la información, la toma de datos y su utilización son componentes esenciales para producir con eficiencia. La inteligencia artificial, elemento central de esta transformación, permite la acumulación creciente de grandes cantidades de datos, el uso de algoritmos para procesarlos y la interconexión masiva de sistemas y dispositivos digitales, lo que corresponde a una nueva manera de organizar los medios de producción. El objetivo es la puesta en marcha de un gran número de «fábricas inteligentes» o «smart factories», capaces de una mayor adaptabilidad a las necesidades y a los procesos de producción, así como a una asignación más eficiente de los recursos, abriendo así la vía a una nueva revolución industrial [54].

El Plan de Desarrollo Productivo en Argentina establece que la implementación de la Industria 4.0 es fundamental para mejorar la competitividad del país en el mercado internacional y aumentar la productividad y eficiencia de las empresas. Este enfoque se ha concentrado principalmente en el sector manufacturero y en las pymes [61].

La adopción de la Industria 4.0 es un reflejo del actual panorama empresarial que se encuentra en un constante cambio e innovación. Es evidente que factores como el crecimiento de la economía digital, la madurez tecnológica y la demanda creciente de productos personalizados han impulsado esta transformación digital. Sin embargo, a pesar de los avances tecnológicos, el uso de tecnologías habilitadoras como la inteligencia artificial, la robótica y la automatización sigue siendo limitado. Los principales desafíos a enfrentar incluyen la falta de capacitación y conocimiento en tecnologías 4.0, escasez de recursos para inversión en nuevas tecnologías y retos como la falta de inversión en infraestructura de comunicaciones y la necesidad de una mayor inversión en investigación y desarrollo. También es necesario mejorar la capacitación y la formación en habilidades digitales, así como fomentar una cultura de innovación y experimentación. Es por ello que para impulsar la adopción de tecnologías 4.0 en las empresas, se han implementado medidas como incentivos fiscales y programas de capacitación y financiamiento para la innovación tecnológica. Además, se ha creado un ecosistema de innovación y emprendimiento, con la creación de polos tecnológicos y la colaboración público-privada en la investigación y desarrollo de tecnologías avanzadas. De esta manera, el gobierno y las empresas trabajan juntos en abordar estos desafíos y aprovechar las oportunidades tecnológicas para mejorar la competitividad y el crecimiento económico del país. El Plan de Desarrollo Productivo Argentina 4.0 contempla diversas políticas públicas para impulsar la adopción de la Industria 4.0, como el Programa de Transformación Productiva, que busca brindar financiamiento y asistencia técnica a empresas para la adopción de nuevas tecnologías y procesos productivos. En este sentido, se busca

fomentar la innovación y el desarrollo de nuevas tecnologías, así como también la capacitación de recursos humanos en las áreas de ciencia, tecnología, ingeniería y matemáticas [61].

Ganadería 4.0

En Argentina, la gestión de las explotaciones ganaderas comienza a aprovechar las tecnologías y está experimentando un cambio hacia la incorporación de las mismas que le permitan adaptar y mejorar sus procesos productivos y aumentar la eficiencia. Aunque el sector ha sido tradicionalmente reacio a la digitalización, se reconoce cada vez más la necesidad de aprovechar los recursos tecnológicos y herramientas de última generación para el manejo ganadero. En este proceso de mejora tecnológica, la identificación individual es la base fundamental para implementar la ganadería de precisión. Al proporcionar una identidad permanente a lo largo de la vida del animal, se obtiene un control específico de cada individuo en lugar de trabajar con promedios grupales, lo que contribuye directamente a la toma de decisiones en cuanto al manejo sanitario y ambiental de las majadas [107].

Actualmente están surgiendo diversas iniciativas de actores del sistema de ciencia y tecnología y del sector privado para generar soluciones innovadoras en el sector ganadero, aprovechando el potencial tecnológico y la capacidad productiva del país. INTA está desarrollando herramientas destinadas a dar respuestas de precisión en diferentes áreas como, por ejemplo: alimentación, producción láctea, comportamiento animal, manejo, nutrición, consumo residual, entre otras temáticas de investigación. El desafío que se presenta es el de generar información para la toma de decisiones, con tecnologías altamente integradas que intervengan desde el monitoreo de la actividad reproductiva de los animales hasta la góndola [107].

En cuanto a la producción ganadera, los sistemas extensivos de rumiantes, bovinos, ovinos, bubalinos y caprinos, presentan como dificultad la posibilidad de controlar en su totalidad el ambiente, además de la libre movilidad de los animales en un predio con extensiones delimitadas. Sin embargo, la incorporación de tecnología y eficiencia productiva es necesaria para asegurar el desarrollo futuro de las familias de ganaderos e industriales que se dedican a esta actividad. En Argentina y Mercosur, existen tecnologías y dispositivos disponibles que permiten la implementación de la Ganadería de Precisión, [54] tales como:

- ❖ Plataforma de autopeso: Pesa los animales sin necesidad de encontrarse en movimiento [54].
- ❖ Comederos inteligentes: Desarrollados por el INTA, miden el consumo individual de cada animal e intentan evaluar la deficiencia del consumo. Es un claro ejemplo de innovación abierta generada a través de una articulación público-privada que incluso llevó a estos comederos a una etapa de comercialización [53].
- ❖ Collares de seguimiento: Nos indican el lugar dónde se encuentra el animal en tiempo real y qué actividad está desarrollando (rumia, traslado, toma de agua, descanso, celo o enfermedades). También estos collares permiten evaluar el desempeño reproductivo de un animal [53][54].

- ❖ **Balanzas de pesada al paso:** Desarrollados por la Universidad de Australia con la que el INTA tiene un convenio, es una plataforma por donde los animales pasan caminando para tomar agua en un solo sentido de circulación y permite detectar, por ejemplo, la aparición de un nuevo animal o vaca preñada. El desafío fue aplicar esa tecnología y adaptarla a las necesidades del productor local. A partir de su implementación, se realizó el desarrollo de software para satisfacer los requerimientos de información del productor [53].
- ❖ **Dispensadores para tambo:** El objetivo era el desarrollo de soluciones a mediana escala. Aquí los dispensadores identifican el orden en el que entra el animal y permiten suministrar la ración de alimento recomendada por el nutricionista [53].
- ❖ **Cámaras infrarrojas:** Que identifican al animal y permiten observar si existe comportamiento fuera de lo normal (rengueras, celos, aumento de la actividad) [54].
- ❖ **Monitores de aguadas:** Cada uno de los puntos puede ser monitoreado continuamente y ahorra mucho tiempo de recorrida a caballo o en camioneta para verificar si los tanques están vacíos [53].

Estos sistemas no necesitan fibra óptica ni conectividad vía señal de celular, 3G o 4G. Además, en cuanto al ambiente, la utilización de índice verde (NDVI) y el seguimiento de humedad del suelo (TDVI), imágenes de cámaras multiespectrales en drones e imágenes satelitales brindan información de la oferta forrajera en cantidad, y a futuro también brindarán información de calidad del forraje ofrecido [54].

El avance de las técnicas de explotación de la información y de bases de datos a gran escala mediante inteligencia artificial no tiene precedentes. En nuestro país, el ya mencionado INTA a la par de otras empresas privadas, han tomado nota de estos avances y se han orientado al desarrollo de dispositivos como los mencionados, pero adaptados a la realidad de nuestros sistemas productivos [64].

Particularmente en Navarro, Provincia de Buenos Aires, se ha iniciado un proceso en un tambo, en colaboración con el INTA, con el objetivo de desarrollar un software para el manejo automatizado de la suplementación del ganado. Este trabajo multidisciplinario ha involucrado a profesionales de diversas especialidades, como veterinarios, nutricionistas, ingenieros electromecánicos y personal del tambo. Para lograr este avance tecnológico, se han utilizado caravanas de identificación por radiofrecuencia (RFID), sensores y mini PCs para procesar localmente la información y aplicar un algoritmo que determina la cantidad de comida que necesita cada animal. Después de un año y medio de pruebas, el sistema está conectado y en funcionamiento en el tambo. Actualmente, se están realizando pequeñas mejoras para aumentar la confiabilidad y seguridad, y se está trabajando en el desarrollo de una aplicación móvil que se pueda utilizar con o sin internet [95].

La Estación Experimental Agropecuaria «El Colorado» del INTA situada en Formosa ha llevado a cabo capacitaciones sobre el uso de drones en el sector agropecuario. Los drones, originalmente desarrollados para actividades no agrícolas, ahora son ampliamente utilizados por agricultores y técnicos. Con cámaras, GPS y sensores, los drones permiten obtener imágenes de alta definición que facilitan el monitoreo de grandes extensiones de tierra en

zonas rurales. En solo una hora, se puede relevar más de mil hectáreas, lo que permite una localización temprana de plagas e invasiones de malezas y un control de ganado más eficiente. La información recopilada también permite planificar con anticipación recorridos y aplicar agroquímicos en focos de plagas, además de conocer el potencial de rendimiento de cada sector del campo para planificar la cosecha. De esta manera, los drones se están posicionando como un complemento eficaz a las imágenes satelitales en el monitoreo y control agrícola [108].

En la provincia de Corrientes se realizó un estudio sobre la aplicación de la ganadería de precisión en rodeos de cría, enfocándose en la categoría vientres. Se utilizaron tecnologías como la identificación electrónica, el monitoreo reproductivo y la evaluación genética para mejorar la eficiencia y productividad de los rodeos de cría, lo que resultó en beneficios económicos para los productores. El estudio resalta que la ganadería de precisión se enfoca en la utilización de tecnologías para recopilar información y tomar decisiones más eficientes y precisas en la gestión de la producción ganadera. Además, se destaca la importancia de la capacitación y el acceso a la tecnología, así como también la necesidad de trabajar en conjunto con otros actores del sector para avanzar en su implementación [62].

También INTA ha desarrollado una tecnología inalámbrica llamada LoRa (Long Range) que permite conectar múltiples dispositivos a Internet sin necesidad de cables físicos, creando una red de malla inalámbrica en la que cada nodo se comunica directamente con otros nodos en su área cercana. LoRa es una tecnología de modulación en radiofrecuencia que destaca por su alta tolerancia a las interferencias, alta sensibilidad para recibir datos, bajo consumo, largo alcance de 10 a 20 km, baja transferencia de datos de hasta 255 bytes y conexión punto a punto. En el ámbito ganadero, la tecnología LoRa se utiliza para monitorear a los animales mediante sensores colocados en collares o en su cuerpo que envían información sobre su ubicación, temperatura corporal y actividad. Esta información se puede utilizar para detectar problemas de salud y signos de celo, así como mejorar la gestión del rodeo al conocer la ubicación y movimiento de los animales. Además, la tecnología LoRa también se utiliza para monitorear los niveles de agua en los bebederos y la temperatura en los corrales, lo que permite una gestión más eficiente y sostenible de los recursos y una mejor planificación de la alimentación y el agua para los animales. La tecnología LoRa ha sido instalada en diversas localidades como Chamical (La Rioja), Balcarce (Buenos Aires) y Anguil (La Pampa) [63].

En la industria lechera, se ha incorporado la robótica para automatizar los procesos de producción en tambos. En esta línea de trabajo, el INTA se asoció con tres empresas para evaluar el rendimiento de los dispositivos. En los módulos de ordeño automático, un brazo robótico identifica la posición de los pezones de la vaca mediante una cámara infrarroja y realiza la rutina de limpieza y desinfección de los mismos. Además, el sistema cuenta con un sistema de lavado automático tanto para la unidad de ordeño como para el equipo de frío, lo que garantiza la uniformidad en las tareas y la obtención de rutinas consistentes, lo que a su vez repercute positivamente en la calidad final de la leche. Asimismo, las puertas separadoras se presentan como dispositivos estratégicos, ya que permiten ahorrar tiempo y liberar al

operario para que pueda realizar otras actividades, como la gestión de los datos que se generan [95].

Otra línea de investigación y validación de tecnología son los alambrados virtuales. La propuesta consiste en cambiar los tradicionales alambrados eléctricos por otros de tipo virtual. Se busca reemplazar el estímulo visual del hilo y las varillas por sonidos, esperando que el animal aprenda que tiene que modificar su dirección de desplazamiento al escucharlos para evitar la aparición del estímulo eléctrico de baja intensidad, menor a la de los eléctricos comunes. Los aspectos relacionados al estudio de la respuesta animal a los estímulos en ambientes controlados están avanzando y se probó un prototipo de collar en condiciones de pastoreo real a fines del año 2020 con buenos resultados [95].

Dos aspectos centrales en la producción ganadera extensiva son la disponibilidad de agua y forraje. Los beneficios que pueden aportar la incorporación de tecnologías para la gestión de estos recursos son de alto valor agregado; ya que evitan la necesidad de desplazarse grandes distancias sólo para observar el correcto funcionamiento de molinos o destinar tiempo a recorrer los potreros para evaluar la condición del pastizal. Un ejemplo de estas implementaciones se dio en General Conesa, Provincia de Río Negro, donde fue probado, con éxito, un prototipo con sensores de profundidad de agua encargado de medir los niveles en las aguadas y en los tanques de reserva para los animales, además la prueba del dispositivo funciona correctamente en una zona donde la cobertura de internet no es buena; ya que los datos podían ser enviados resistiendo grandes distancia sin pérdidas mayores [95].

Además, en todo el país, grupos de INTA se encuentran trabajando en el desarrollo de una metodología para determinar la disponibilidad de forraje utilizando imágenes obtenidas por drones y satélites que permitan planificar la utilización de los pastizales y monitorear su condición en forma remota. Existen distintos tipos de dispositivos disponibles en el mercado, algunos de ellos cuentan con buena autonomía, pueden abarcar grandes áreas en poco tiempo, permiten anexar distintos tipos de cámaras y pueden ser utilizados para el conteo de animales aunque también presentan algunas desventajas como la fragilidad que presentan frente a algunas condiciones ambientales (viento y lluvias). Su funcionalidad se está validando en distintos ambientes productivos del país [95].

3.2.3 La experiencia de la Patagonia en la implementación de PLF en la ganadería.

En la región sur de Argentina, la actividad ganadera ovina es la más común, aunque también se encuentra una pequeña proporción de ganado bovino. Además de los programas nacionales que promueven la adopción de tecnologías ganaderas avanzadas, existen en la Patagonia ejemplos concretos de tecnologías y prácticas que están siendo implementadas para mejorar la eficiencia y productividad de la producción ovina [55].

Entre estas prácticas se encuentra la evaluación genética implementada por el Provino (Servicio Nacional de Evaluación Genética). Esta evaluación se lleva a cabo utilizando registros de producción, análisis de muestras de lana (diámetro de fibra y rinde) y datos obtenidos mediante ultrasonografía; el Provino ofrece dos versiones de evaluación genética:

Provino Básico y Provino Avanzado. Como resultado de esta evaluación, se generan informes que contienen las Diferencias Esperadas en la Progenie (DEP's). Estas medidas cuantifican el mérito genético de los animales evaluados y permiten avanzar en la selección hacia los objetivos deseados [55].

Otra práctica destacada son las *Buenas Prácticas Ganaderas (BPG)*, que se han convertido en la forma más segura e inteligente de hacer ganadería en el siglo XXI. La implementación de BPG es necesaria para satisfacer las demandas de los mercados internacionales, que cada día exigen mayores precisiones sobre diversos aspectos, como el bienestar animal (BA), la resistencia a los antibióticos (AMR), los gases de efecto invernadero (GEI), y los residuos de medicamentos veterinarios (LMR's). Estos temas, que los países compradores dan cada vez más importancia, se están instalando, también, en nuestra sociedad [55].

Con el objetivo de mejorar el seguimiento de los recursos naturales, se han realizado avances significativos en la calibración de evaluaciones mediante el uso de sensores remotos MODIS. En particular, se ha logrado un gran progreso en el cálculo de la Productividad Primaria Neta Aérea a partir de imágenes satelitales MOD17/A3. Esta técnica se ha vuelto fundamental debido a que los relevamientos de campo de pastizales resultan costosos y difíciles de llevar a cabo en grandes extensiones de tierra. Esto abre la posibilidad de calcular el forraje consumible de los campos, que es una porción de esta producción total, y de generar mapas de receptividad regionales para ovinos, bovinos y guanacos además de que posibilitan el análisis de los niveles de pastoreo en relación a la receptividad animal [55].

También se han utilizado sensores remotos para el seguimiento de sequías mediante el índice verde (NDVI) de imágenes diarias MODIS sintetizadas en imágenes quincenales de 250x250 píxeles que provee el INTA Castelar. La comparación de cada píxel con valores históricos promedios permite identificar áreas que están por arriba o por debajo del promedio histórico [55].

El INTA reconoce la importancia de la información meteorológica para la toma de decisiones en la producción agrícola. Es por ello que desarrolló una red de 38 estaciones meteorológicas automáticas en toda la región, cuya información está disponible y actualizada en <http://sipas.inta.gob.ar>. Además, se están utilizando drones industriales para la evaluación de pastizales y el seguimiento del estrés hídrico en los cultivos, así como para verificar la distribución uniforme del riego y el arreo de animales en los campos [55].

Para fomentar la adopción de tecnologías avanzadas en la gestión de las empresas agrícolas, se llevan a cabo capacitaciones y charlas en toda la Patagonia, muchas de ellas lideradas por el INTA y el Ministerio de Producción. También se están impulsando iniciativas para vincular tecnologías disponibles en diferentes partes del país, fomentar la colaboración público-privada y facilitar la transferencia de conocimientos. La prueba de tecnologías en entornos reales y la interacción con los usuarios acortan los tiempos de desarrollo y mejoran las oportunidades de los productos y servicios que se ofrecen en toda la cadena de valor [94].

En resumen, la ganadería de precisión se presenta como una gran oportunidad tanto a nivel nacional como regional para desarrollar tecnologías que permitan la automatización y gestión remota de tareas en el ganado, logrando mayor eficiencia en el uso de los recursos y

fomentando la innovación en las formas de producir. Esta tecnología se ha convertido en una herramienta clave para la industria ganadera, y su adopción puede impulsar la interacción entre diferentes campos del conocimiento, como la ingeniería, la informática y la biología, lo que puede llevar a nuevas soluciones para los desafíos del sector [107].

3.3 Avances Recientes de PLF en la Ganadería Ovina

A continuación, se detallarán las tecnologías y sensores que están involucrados en la producción ovina a nivel mundial, incluyendo aquellas que son principalmente utilizadas como parte de los avances en ganadería 4.0 adaptados para su uso en ovinos. Posteriormente, se analizará el impacto de la Ley Ovina en Argentina, la cual promueve el uso de tecnologías por parte de los productores, y por último se examinarán las tecnologías que actualmente se están utilizando y comercializando en el sector ovino en el país.

3.3.1 ¿Cómo está transformando la PLF la producción ovina en todo el mundo?

Según un estudio realizado en 2014, la población bovina mundial se estimó en 1430 millones de animales, mientras que la población ovina y caprina se fijó en 1870 millones. A pesar de este alto número de rumiantes, la investigación sobre enfoques de PLF en ovejas y cabras es significativamente menor en comparación con el ganado bovino. En la siguiente sección se destacan los estudios más recientes sobre el uso de la tecnología e IoT en este sector minoritario [11].

Identificación de animales y seguimiento automático

Desde 2010, la Unión Europea ha establecido una legislación que requiere la identificación y registro obligatorios de caprinos y ovinos nacidos en ese año o en años posteriores. Los sensores de identificación electrónica para pequeños rumiantes se pueden encontrar en diferentes formas, como crotales, bolos ruminales o como chips inyectados debajo de la piel, dependiendo de la aplicación específica. Además de la identificación, existe un gran interés en monitorear continuamente los rebaños, especialmente en las granjas de pastoreo, es por ello que varios estudios recientes se han centrado en el uso de sensores GPS y drones para monitorear animales en tiempo real ofreciendo resultados con una precisión de entre 96-97%. De esta manera, esta tecnología ofrece nuevas posibilidades para monitorear el comportamiento del rebaño, identificar animales enfermos o heridos y mejorar el bienestar animal en general [11].

Sistemas de ordeño automático

En las explotaciones ovinas y caprinas, la automatización no es un concepto demasiado extendido, sin embargo, en los últimos años, la demanda de sistemas de ordeño automático ha aumentado significativamente. Debido a las características morfológicas y anatómicas únicas de cada especie, se requieren máquinas con atributos específicos en cuanto a cantidad de tiempo y presión negativa que faciliten la tarea de ordeñado. Se han realizado varios estudios para perfeccionar estas máquinas de ordeño automático para ovejas y cabras, con excelentes resultados [11].

Detección de celo

En la cría de pequeños rumiantes, la reproducción y la detección temprana de enfermedades son aspectos fundamentales para lograr el éxito en la actividad. Por esta razón, estas áreas son de gran interés para el monitoreo automatizado.

Uno de los avances más significativos en este campo es el desarrollo de un detector automático capaz de reconocer de manera unívoca a las ovejas y determinar la aceptación de los machos, lo que ha sido aplicado con éxito para diagnosticar el periodo de celo. Este dispositivo logra un 100% de diagnóstico de celo y también proporciona información cuantitativa sobre el comportamiento sexual tanto de la oveja como del carnero.

Además de este dispositivo, se han utilizado otros sensores como acelerómetros y cámaras termográficas para rastrear las características reproductivas de carneros y ovejas. Estos sensores permiten el monitoreo continuo del comportamiento animal, lo que permite una detección temprana de problemas de salud y cambios en el comportamiento, lo que a su vez permite una intervención más rápida y efectiva [11].

Detección de Enfermedades

Las enfermedades infecciosas y la cojera son extremadamente comunes en la cría de pequeños rumiantes. El uso de tecnología para la detección temprana de tales condiciones ha sido investigado en el contexto de minimizar las pérdidas económicas en los campos. En este sentido, un excelente predictor de la salud del animal es su temperatura corporal. Con ello los productores y veterinarios pueden emplear sensores de temperatura implantados directamente en los animales o sensores externos no invasivos para examinar la manada y detectar cualquier anomalía. Otro ejemplo es el uso de la termografía infrarroja para diagnosticar cojeras en ovejas, investigaciones recientes demostraron que esta técnica tiene una sensibilidad diagnóstica del 77% y una especificidad del 78% [11].

Desempeño animal, monitoreo de alimentación y comportamiento animal

El pastoreo y la rumia son las actividades más esenciales para los rumiantes, ya que la mayor parte del día se dedican a ellas. El monitoreo continuo del comportamiento alimentario es fundamental para la salud, la productividad y el bienestar de los rumiantes. El uso de sensores para la recopilación automatizada de datos y software para clasificar y reconocer actividades permite un progreso sustancial en la superación de estas tareas que de otro modo serían desafiantes. Anteriormente se han empleado diferentes tipos de acelerómetros, colocados en las orejas, el cuello o las patas de los animales, para monitorear el comportamiento, cada uno con su respectivo porcentaje de confiabilidad (99, 93 y 100%, respectivamente) [11].

Todos los sensores descritos en esta y las secciones anteriores se pueden utilizar para controlar el bienestar de las ovejas y las cabras. Dado que el bienestar se ve afectado por una multitud de circunstancias, uno de los objetivos de la técnica PLF es analizar y resumir grandes cantidades de datos para proporcionar experiencia técnica con una solución específica. Es por ello que se ha desarrollado una tecnología específica que combina sensores de temperatura mínimamente o no invasivos dentro y fuera del animal, así como acelerómetros y detectores de sonido, para crear algoritmos y aprendizaje automático capaces de detectar el bienestar animal en las granjas y durante el transporte [11].

Cámaras automáticas

El uso de cámaras automáticas para identificar ovinos ha sido objeto de estudio principalmente en Australia con resultados sumamente alentadores. Para entrenar el algoritmo de identificación que utilizan, es necesario adquirir una gran cantidad de imágenes de cada animal, así como de diferentes regiones del cuerpo. La tecnología de imágenes corporales (laterales y superiores) también ha sido utilizada para determinar el peso de los animales, esta técnica se basa en la correlación entre el peso registrado mediante la balanza y el estimado por las imágenes obtenidas. Según estudios realizados, se ha encontrado una correlación cercana al 88,6% entre ambos métodos, lo que sugiere que esta tecnología puede ser una alternativa viable para medir el peso de los animales de manera más eficiente y precisa. Otra de las posibilidades que ofrecen las cámaras automáticas es la estimación de la condición corporal e indicadores de cobertura de cara y presencia de arrugas. El análisis de imágenes también se utiliza para clasificar las carcasas de acuerdo a su conformación, engrasamiento e incluso la composición tisular (relación músculo-grasa-hueso). Esta información se puede utilizar para estimar la influencia de factores ambientales sobre la calidad final de la carne y también para estimar el valor genético de los animales. Estos dispositivos también se usan para detectar problemas de laminitis y otras enfermedades a partir del análisis automático de vídeos de los animales desplazándose [64].

Sistemas de ahijamiento

Estos dispositivos permiten identificar automáticamente la relación madre-cría en condiciones de campo, utilizando la identificación por Radiofrecuencia (RFID) mediante caravanas electrónicas colocadas en los ovinos de interés. El equipo ahijador registra los animales identificados cuando circulan en un lugar de paso obligado, como un pasillo ubicado entre el lugar donde pastorean y el lugar donde toman agua. La información se transmite a una base de datos para su posterior procesamiento, asociando el dato de la caravana y el tiempo en el que se registró. Posteriormente, los datos recopilados son procesados mediante un algoritmo particular a través de una aplicación que da como resultado una asociación madre-hijo. Cabe destacar que el equipo es totalmente autónomo, ya que se alimenta mediante energía solar y bancos de baterías.

Esta información es fundamental para los programas de evaluación genética, ya que los datos genealógicos nos permiten separar correctamente los efectos ambientales de los genéticos. Sin embargo, en general, la información disponible está limitada a los planteles puros de pedigrí y/o algunos pocos planteles PPC. Con este tipo de sistemas, se podrían masificar los registros genealógicos y mejorar la precisión de las evaluaciones genéticas [64].

3.3.2 PLF en la producción ovina en Argentina. ¿Un camino hacia la sostenibilidad productiva?

Desde hace algunos años atrás se viene gestionando un proyecto para modificar la ley ovina, la misma en su versión original del 2001, ha demostrado ser una excelente norma, diseñada mirando al futuro, aquel que hoy ya es presente. Se sostuvo aún con serias limitaciones en su financiamiento e incluso resistió malos manejos; trascendió más de ocho administraciones y

fue modelo para otros proyectos de ley agropecuarios similares, además de haber sido un muy buen negocio para las arcas del Estado y toda la sociedad. La actividad ovina nacional abarca aproximadamente 100.000 productores en sus distintas escalas y modalidades; desde los muy pequeños, los mixtos, hasta aquellos que en esas diferentes escalas tienen a esta producción exclusivamente como un «monocultivo ovino». Incluye 14.8 millones de cabezas ovinas; un valor en la tranquera de 200 millones de dólares y otro tanto en la cadena de valor agregado. El cordero argentino compite de igual a igual en el ámbito internacional, logrando un lugar destacado en los mercados más exigentes. Sus lanas son conocidas en el mundo por su calidad, a partir del trabajo realizado en conjunto entre la industria y el PROLANA, programa de mejoramiento de la calidad de lanas que financia la ley ovina. Pero la oveja no sólo es producción sino principalmente ocupación del territorio, arraigo y cultura, desde el norte hasta el sur del territorio argentino [64].

La «Ley Ovina» ha superado varios desafíos y se prepara para enfrentar nuevos, como la regeneración de tierras, el bienestar animal, las buenas prácticas laborales y el equilibrio de carbono. La nueva normativa contempla la elaboración de planes estratégicos para las cadenas agroindustriales de lana, carne ovina y producciones socio-culturales relacionadas con la ovina [64]. Para enfrentar estos desafíos, se están investigando y adaptando herramientas internacionales para su uso en la producción de rebaños ovejeros.

Identificación Individual

Sin duda el mayor aporte de estas tecnologías se basa en la posibilidad de cambiar las escalas de manejo pasando de la majada al individuo. El uso de las caravanas con chips de baja radiofrecuencia (RFID) es sin duda la tecnología más utilizada para esto y la que mejor interactúa entre los animales y los dispositivos inteligentes. Ya son de uso común en lechería y en ganadería bovina mientras que en el sector ovino recién su adopción aún es incipiente. Existen también caravanas de alta frecuencia que permiten la identificación de grupos de animales en simultáneo, por ejemplo en una jaula de transporte, pero requieren la utilización de otros tipos de lectores que no están disponibles en el mercado en forma masiva. También existen sistemas de identificación basados en análisis de imagen, en particular ese tipo de tecnologías se han investigado y llegan a obtener resultados muy positivos [64].

Análisis de los hábitos de pastoreo

En cuanto a los hábitos alimenticios de los animales, el Instituto de Ingeniería Rural del INTA desarrolló un collar de posicionamiento global para estudiar los hábitos y recorridos de los ovinos en sistemas extensivos patagónicos. Con la innovación de esta tecnología nacional, se reducen los costos y se facilita la toma de decisiones en cuanto al manejo de los pastizales y el ganado. Esta herramienta de geoposicionamiento permite investigar los patrones de distribución y circulación de los animales en sistemas extensivos y ante distintas condiciones climáticas y ambientales. Esta tecnología ya está siendo utilizada a nivel nacional y adaptada a los requerimientos de cada territorio en particular en La Rioja, San Luis, Tucumán, Salta, Río Negro y Chaco [56].

Comederos inteligentes

Actualmente existen dos desarrollos disponibles. El primer tipo de comederos se orientó inicialmente a la medición del consumo residual (RFI). Estos comederos fueron desarrollados por el INTA Anguil, probados inicialmente en bovinos y actualmente se está trabajando en adaptar la versión disponible para ovinos, lo cual implica algunas modificaciones en la estructura de acceso y el tamaño de las bateas. Los comederos están equipados con una antena de lectura de caravanas RFID que detecta el momento en que cada animal se acerca al comedero y registra el peso de la batea al inicio y al final de cada comida. Luego calcula el consumo individual y mediante un software específico se puede relacionar ese consumo con la ganancia de peso de los animales. El Consumo Residual, es la diferencia entre el consumo esperado para la ganancia de peso del animal y el consumo que efectivamente realizan los animales. Aquellos que presentan valores negativos de RFI consumen menos alimento que el esperado y por lo tanto son más eficientes, mientras que los que tienen un valor positivo consumen más alimento del esperado [64].

La identificación y selección de los animales más eficientes no sólo contribuiría una mayor eficiencia productiva sino también a reducir la contribución de la ganadería ovina a la emisión de gases de efecto invernadero. Este aspecto es particularmente relevante en el mercado de lanas ya que hay una creciente demanda de los consumidores relacionada con la incorporación de indicadores de impacto ambiental y la implementación de esquemas de manejo sustentables. Los comederos de consumo residual podrían integrarse en programas de evaluación genética a través de centrales de prueba conectadas entre sí o incluso campos que los utilicen bajo un mismo protocolo de registro de información. El segundo tipo de comederos en desarrollo está pensado para realizar suplementaciones o dosificaciones variables. También están equipados con antenas capaces de leer las caravanas RFID identificando al animal y permite dosificar una cantidad de alimento variable para cada individuo mediante un mecanismo de rodillo sinfín. Los usuarios pueden programar a través de una plataforma móvil, la cantidad específica de alimento para cada animal e incluso en qué lugar pueden comer cada individuo. Ambos sistemas proveen información acerca del comportamiento de los animales para lo cual no fueron pensados inicialmente pero que permiten estudiar las diferencias en el comportamiento ingestivo y cómo se relacionan con otras características productivas [64].

Balanzas automáticas y sistemas integrados

Las balanzas automáticas están mayormente relacionadas y con mayor presencia en el sector lechero y el porcino aunque recientemente surgieron algunas soluciones para la ganadería bovina y ovina de cría. El INTA Anguil, junto a la Universidad de Queensland, trabajaron conjuntamente en el desarrollo y la adaptación de un modelo de balanza de pesado al paso para bovinos cuyo principio puede aplicarse perfectamente en ovinos. Estas balanzas al igual que los sistemas de ahijamiento se deben ubicar en un lugar de paso forzado. Cuentan con un pasillo previo compuesto por paneles que conducen a los animales hacia la balanza donde se registra el peso y la identificación animal. Una vez más es necesario que los individuos estén identificados con caravanas RFID ya que las balanzas poseen una antena de lectura integrada [64].

Luego de mencionados los equipos tecnológicos podemos hablar de su disponibilidad comercial [64]:

- ❖ Los comederos de consumo residual y los de dosificación variable ya se encuentran en el mercado y existen empresas locales encargadas de su fabricación, comercialización y distribución.
- ❖ En el caso de los sistemas de ahijamiento los equipos existen pero con presencia en otros países y el costo de importación aún es sumamente elevado para las condiciones locales; los equipos nacionales están en la etapa de validación tanto en establecimientos comerciales como en estaciones experimentales.
- ❖ Lo mismo sucede con las balanzas automáticas, aunque en este caso ya existen varias pruebas que avalan los resultados y la robustez de los equipos para las condiciones de nuestros sistemas.

La incorporación de estas tecnologías en la cadena agroindustrial ovina ofrece un excelente valor agregado y es de gran potencial, lo que ha llevado a que los principales países competidores de Argentina en materia ovina, las estén adoptando. La industria ovina nacional debería tener una estrategia para hacer lo mismo en base a las posibilidades locales. Es improbable que en el corto plazo un gran número de productores adquieran equipos como los mencionados, los usen para tomar datos y administrar sus campos, pero es perfectamente posible incorporar estas tecnologías en los establecimientos que más impacto tienen sobre el resto, como por ejemplo, las cabañas que participan de los programas de mejora genética animal. Algunos equipos y elementos como las caravanas RFID, los ahijadores y balanzas automáticas tienen un costo que puede ser afrontado por campos medianos y grandes. Por otro lado, los equipos más complejos como los comederos de consumo residual podrían incorporarse en estaciones experimentales integradas a las redes de evaluación genética como también, los sistemas de análisis de imágenes se podrían incorporar en la industria frigorífica recuperando información referida a la calidad de carcasa restante y rendimiento comercial. La integración de esta información con datos del genotipo de los animales se podría utilizar para implementar evaluaciones genómicas en caracteres complejos o los que se expresan tarde en la vida de los animales [64].

En definitiva se trata de identificar cual es la información relevante a capturar en cada eslabón de la cadena, quienes son los que aportan información de más valor y cómo se integra luego en un sistema que permita obtener el mayor beneficio para todos. La competitividad de una industria no sólo se basa en las condiciones macroeconómicas como el tipo de cambio, que sin lugar a dudas es muy relevante, sino también en la capacidad para anticipar nuevos escenarios. Es decir que no solamente hay que producir y vender pensando en lo que se demanda hoy, sino también en lo que se demandará en el futuro cercano y además es importante notar que se trata de un ejercicio colectivo que involucra no solo a los productores sino también a la industria y el estado. Los sectores que no son capaces de hacerlo corren serios riesgos de volverse obsoletos o en todo caso llegar tarde al negocio [64].

3.4 Sistemas de Soporte a la Toma de Decisiones: ¿Cómo se está utilizando en la PLF?

A medida que se avanza en la ganadería de precisión, surgen nuevas oportunidades para desarrollar tecnologías y estrategias que mejoren la eficiencia y sostenibilidad de la producción ganadera. El objetivo central es alcanzar una toma de decisiones basada en información diaria, confiable y completa de cada animal y su entorno en tiempo real. Esto implica la recolección de datos precisos sobre el peso, el consumo de alimento, los suplementos necesarios y el estado general de salud, así como también sobre el ambiente y la infraestructura, como la temperatura y humedad ambiental, y el estado de los motores de las bombas o los bebederos. Estos datos se procesan mediante técnicas de Machine Learning e inteligencia artificial, y se almacenan localmente o se aseguran mediante tecnologías de Blockchain. Esta fase del proceso permite generar información de alta calidad, que se devuelve al productor para ayudarlo en la toma de decisiones. Al ampliar continuamente las posibilidades de obtener información relevante y de calidad sin la necesidad de intervención humana para su recolección y carga, se pueden aprovechar al máximo las capacidades de la inteligencia artificial y del aprendizaje automático para tratar grandes volúmenes de datos. El análisis exhaustivo de los datos puede aumentar la rentabilidad y potenciar las posibilidades de trazabilidad y seguimiento animal. La integración de tecnologías, como la inteligencia artificial, el Blockchain y el IoT, ha permitido una transformación innovadora en el proceso de manejo, ofreciendo complementariedad en los procesos y una mayor disponibilidad de datos específicos para cada necesidad [107]. En medio de la evolución de los sensores y el IoT, los investigadores empezaron a fomentar el concepto de ciencia de precisión basándose en la premisa de que aumentaría de forma sostenible la producción de alimentos y el bienestar animal. La comunidad agrícola adoptó el concepto, y se acuñó el término agricultura de precisión como medio de optimizar la productividad preservando los recursos mediante la idea de gestión integral de la explotación. La adopción de la agricultura de precisión no se produjo con la rapidez prevista debido a la falta de herramientas de apoyo a la toma de decisiones adecuadas para aplicarla. La evolución de la tecnología animal de precisión siguió de cerca a la agricultura de precisión, aunque estudios independientes ya habían propuesto la gestión individual de los animales mediante DSS para aumentar la rentabilidad y la productividad. Las herramientas de tipo DSS son el componente informático de la ciencia de precisión que integra el análisis de datos con el análisis predictivo, es decir, la creación de modelos [13].

3.4.1 ¿Cómo están ayudando los DSS a nivel mundial?

La aplicación de tecnologías de teledetección, generación de imágenes y otras mediciones a distancia permitirán seguir desarrollando las herramientas DSS para mejorar la gestión productiva. Estas herramientas requieren interfaces que los productores puedan utilizar fácilmente e idealmente puedan vincularse a otras empresas a lo largo de la cadena de suministro. Algunos ejemplos de herramientas DSS son los programas de mejora genética, y las herramientas de gestión de precisión para mejorar la gestión de los pastos, el pastoreo y la nutrición, con el fin de mejorar la fiabilidad a la hora de cumplir las especificaciones del

mercado objetivo. La capacidad de vincular herramientas de medición objetiva, integrar datos en toda la cadena de suministro y utilizar los denominados cuadros de mando que permiten acceder fácilmente a una serie de DSS también respaldará las mejoras en las industrias ganaderas extensivas [13].

DSS para empresas agrícola y ganaderas

A continuación, se ofrecerá una revisión global de las aplicaciones tipo DSS disponibles en el mercado mundial para empresas agrícolas y ganaderas. Estas aplicaciones se enfocan en diferentes áreas, tales como el manejo financiero de la producción, el impacto ambiental de la ganadería y la mejora genética. Todas ellas buscan mejorar la toma de decisiones en regiones productivas.

GPFARM

El Great Plains Framework for Agricultural Resource Management (GPFARM), es utilizado en Norteamérica y se puede aplicar a todo el dominio de la explotación; sirve para realizar análisis económico de explotaciones agrícolas y ganaderas y funciona como sistema de información y para el apoyo a la toma de decisiones multicriterio [14][75]. La agricultura sostenible en terrenos extensivos es un problema complejo que exige la consideración de muchos factores, procesos e instituciones interrelacionados. Por lo tanto los productores deben ser capaces de modificar sabiamente las prácticas de gestión de las explotaciones agrícolas y ganaderas para:

- 1) aprovechar las ventajas de la economía global;
- 2) mantenerse al día de los nuevos sistemas de cultivo, gestión de plagas y labranza; y
- 3) cumplir la nueva legislación para proteger los recursos del suelo, el aire y el agua.

El GPFARM se probó en explotaciones integradas de cultivo y producción ganadera. Entre sus características más destacadas figuran las siguientes [75]:

- ❖ Combinación de bases de datos específicas de cada lugar y modelos medioambientales con análisis económicos para proporcionar una planificación estratégica de toda la explotación agrícola y ganadera.
- ❖ Análisis y comparación de múltiples escenarios de gestión; estos análisis pueden realizarse en unidades individuales de gestión de la tierra (por ejemplo, áreas de suelos similares o gestión de cultivos) o combinaciones de unidades de gestión de la tierra hasta el campo.
- ❖ Bases de datos específicas para cada lugar (actualmente pobladas para las condiciones climáticas y los sistemas de cultivo del este de Colorado).

BREEDPLAN

Breedplan es un sistema de evaluación de rendimiento genético de animales de raza pura. Fue desarrollado por la Universidad de Nueva Inglaterra en Australia y utiliza información recopilada sobre características de interés productivo como el peso, la calidad de la carne y la tasa de crecimiento para producir informes detallados sobre el rendimiento de un animal. Breedplan utiliza una técnica llamada Mejoramiento Genético Basado en Marcadores (GBM)

para proporcionar información precisa y confiable sobre la genética de un animal; también utiliza un sistema de evaluación genética basado en la tecnología BLUP que incorpora procedimientos de análisis de características múltiples para producir valores reproductivos estimados (EBV) para el ganado registrado en una variedad de características de producción importantes. En el cálculo de los EBV para cada animal se incluye la información de rendimiento y pedigrí del propio animal, el rendimiento de todos los parientes conocidos, el rendimiento de cualquier descendencia que el animal pueda tener, la relación conocida entre rasgos y cualquier información genómica que se encuentre disponible [82].

Los informes de Breedplan se utilizan en todo el mundo para ayudar a los criadores de animales de raza pura a tomar decisiones informadas sobre la selección y cría de animales. Se han desarrollado más de 60 evaluaciones Breedplan diferentes, que representan más de 14 millones de animales, casi 40 razas de ganado vacuno y al menos 100 asociaciones de razas distribuidas en 14 países. Breedplan también ofrece una variedad de herramientas y recursos para ayudar a los criadores a utilizar el sistema de evaluación de rendimiento genético de manera efectiva. Los usuarios pueden acceder a información sobre la selección de animales, la interpretación de informes de Breedplan y la incorporación de datos en el sistema [82].

GRAZPLAN

El software GRAZPLAN de CSIRO es aplicado en empresas de pastoreo en Australia [14]. El proyecto GRAZPLAN tiene como objetivo principal la transferencia efectiva de nueva información y tecnología a la práctica agrícola; para ello ha desarrollado tres herramientas de apoyo a la toma de decisiones disponibles comercialmente para ayudar a los agricultores a mejorar la gestión de sus explotaciones. Estas herramientas se derivan de una serie de modelos informáticos que se basan en los resultados de investigaciones científicas y se dirigen no solo a agricultores y asesores agrícolas, sino también a organizaciones industriales, agronegocios, bancos y compañías de seguros. Estos sistemas están diseñados para utilizarse junto con el clima local y los datos del campo para probar la relevancia de diferentes procedimientos de manejo para campos individuales. El DSS principal, GrazPlan, se utiliza para evaluar y optimizar las decisiones de gestión a largo plazo en términos de rentabilidad y sostenibilidad; es un sistema bastante general en su aplicación y de estructura modular. La base de datos de registros meteorológicos diarios de toda Australia, es la base para los otros dos DSS más pequeños: MetAccess y LambAlive. MetAccess está diseñado para mostrar y analizar registros meteorológicos diarios, proporcionando a los usuarios estimaciones de la probabilidad de patrones climáticos específicos dentro del rango de datos de una localidad determinada. LambAlive, por su parte, está diseñado para predecir el riesgo de muerte de corderos por mal tiempo en localidades y rebaños específicos, permitiendo al usuario probar diferentes procedimientos que le permitan reducir estas pérdidas [76][77].

EcoMod

EcoMod es un modelo biofísico de simulación para los sistemas de pastoreo de Australia y Nueva Zelanda [14]. Este modelo incluye módulos para el crecimiento y utilización de pasturas, dinámica de agua y nutrientes, fisiología y producción animal y una gran variedad de opciones para el manejo de forrajes, riego y aplicación de fertilizantes. Además dentro del modelo es posible definir hasta 100 terrenos de pastoreo independientes para representar la

variación espacial dentro de un campo teórico, los cuales pueden tener diferentes tipos de suelo, estado de nutrientes, especies de pastos, manejo de fertilizantes y riego, pero están sujetos al mismo clima. También es posible simular estrategias de pastoreo rotativo de uso común y pastoreo continuo con números de animales fijos o variables [78].

APSIM

El simulador de sistemas de producción agrícola (APSIM por sus siglas en inglés) es utilizado en Australia para la modelización de cultivos [14]. APSIM es un modelo integral desarrollado para simular procesos biofísicos en sistemas agrícolas, particularmente en lo que se refiere a los resultados económicos y ecológicos de las prácticas de gestión frente al riesgo climático. También es utilizado para explorar opciones y soluciones para los dominios de problemas de seguridad alimentaria, adaptación y mitigación del cambio climático y emisiones de carbono. APSIM contiene muchos de los modelos clave necesarios para explorar los cambios en los paisajes agrícolas con capacidades que van desde la simulación de la expresión génica (proceso por el cual la información genética se convierte en un producto génico, como una proteína), y la simulación de granjas con campos múltiples (por ejemplo, una simulación de la interacción entre diferentes cultivos en una misma granja). APSIM se estructura en torno a módulos de planta, suelo y gestión. Estos módulos incluyen una amplia gama de cultivos, pasturas y árboles, procesos del suelo, incluido el balance hídrico, pH del suelo, erosión y una gama completa de controles de gestión. APSIM surgió de la necesidad de herramientas que proporcionaran predicciones precisas de la producción de cultivos en relación con el clima, el genotipo, el suelo y el factor de gestión al tiempo que abordaban los problemas de gestión de recursos a largo plazo [79].

Norfor

El sistema Nordic Feed Evaluation (Norfor) es utilizado en Escandinavia y está centrado en factores de nutrición animal y alimentación [14]. Es un modelo semi-mecanicista, estático y de base científica que predice el suministro y las necesidades de nutrientes para el mantenimiento, la producción de leche, el crecimiento y la preñez del ganado vacuno. El modelo puede dividirse en cinco partes: (1) una sección de entrada que describe las características del animal y los alimentos disponibles; (2) un módulo que simula los procesos en el tracto digestivo y el metabolismo intermediario, denominado calculador de ración de alimento (FRC); (3) un módulo que predice la ingesta de alimento; (4) un módulo que predice la estructura física de la dieta; y (5) una sección de salida que describe el suministro de nutrientes, sus balances y las respuestas de producción. Las variables de entrada del modelo son las características del animal y del alimento. En el caso de las vacas lecheras, las principales variables de entrada son el peso corporal, la fase de lactancia, el día de preñez y la producción diaria de leche prevista o potencial. Para los animales en crecimiento (toros, bueyes y novillos), las variables de entrada son el peso corporal y la ganancia media diaria de peso. También se requieren para el modelo los índices de degradación de las fracciones solubles y potencialmente degradables del alimento [80].

Norfor utiliza ecuaciones matemáticas para estimar la cantidad de energía, proteína y otros nutrientes que se encuentran en los alimentos para el ganado, como el heno, la hierba natural y los concentrados. Estas ecuaciones se basan en datos de análisis químicos de los alimentos

y en estudios de digestibilidad realizados en el ganado. Además esta aplicación permite a los ganaderos y nutricionistas de animales optimizar la dieta de su ganado para maximizar la producción de leche y carne, mientras se minimiza el costo y se reduce la emisión de gases de efecto invernadero. Ahora es posible calcular el impacto climático del cultivo y el secuestro de carbono de algunos forrajes. Los nuevos insumos son las tres variables que más afectan el impacto climático y el secuestro de carbono. Estos son el rendimiento de la cosecha, la aplicación de nitrógeno de estiércol y nitrógeno fertilizante por hectárea y año. Por lo tanto, es posible calcular el impacto climático específico del campo y la cantidad de hectáreas requeridas [81].

Finnegans GO CERES

Este aplicativo se utiliza para la gestión de una empresa agropecuaria de forma online, gracias a que cuenta con una infraestructura completamente en la nube. Con esta herramienta, es posible controlar tanto los costos de los insumos aplicados en la producción como también los costos de las labores llevadas a cabo, tanto si fueron contratadas como si se realizaron con maquinaria propia. Además, permite llevar un seguimiento completo de las actividades ganaderas, como la cría, recría, invernada y feedlot, a la vez que permite integrar la gestión agrícola y ganadera con la gestión de compras de servicios, lo que favorece un control más preciso de los precios y facturación de los servicios contratados.

Finnegans también facilita un mejor control del negocio, ya que se puede conocer en detalle las existencias, entregas, certificaciones y liquidaciones de todas las campañas agrícolas permitiendo resolver toda la gestión administrativa, incluyendo la generación de cuentas corrientes de clientes y proveedores, presentaciones contables, cálculos e interacciones con el fisco para asegurar el cumplimiento. Además, se dispone de herramientas de seguimiento para gestionar la trazabilidad de cada tropa de ganado y la identificación de cada animal, lo que garantiza la sanidad del rodeo. También se puede controlar el cumplimiento de planes sanitarios y la disponibilidad de vacunas u otros insumos.

Por último, es posible llevar un control del inventario de alimentos y tener información sobre qué se le dio de comer a cada animal y cuánto alimento consumen, lo que permite optimizar el uso de los recursos y mejorar la eficiencia de la empresa [68].

VISUAL Gestión Agro

Es un software desarrollado para el sistema operativo windows y está orientado a productores agropecuarios, ingenieros agrónomos, licenciados en administración rural, licenciados en administración de empresas agropecuarias que asesoran a productores, estudios contables que deseen llevar la gestión de empresas agropecuarias y contratistas rurales que deseen llevar el seguimiento y la gestión de las labores realizadas a terceros. Esta herramienta combina diversas tecnologías de información que permiten al productor y a la empresa agropecuaria sistematizar el flujo de datos vinculado con su actividad integrando la información administrativa, contable, impositiva y financiera con las netamente económicas y técnicas de campo, de manera tal que le permitan a los usuarios planear y gestionar con eficiencia. Algunas de las tareas específicas que se pueden realizar son: compras y ventas; facturación electrónica; libros de IVA; cuentas corrientes de clientes y proveedores; pagos y cobranzas;

retenciones de IVA, cálculo de ganancias; libros de caja y bancos; conciliación bancaria; exportación a aplicativos de AFIP y rentas, entre otros [69].

La adopción y uso de los modelos, aplicaciones y software mencionados en este capítulo puede presentar, en principio, dificultades para los agricultores y empresas agrícolas, ya que su adopción dependerá en gran medida de su facilidad de uso y de la identificación de las mejores prácticas para su aplicación en las actividades productivas. A pesar de los posibles desafíos que puedan presentarse en la implementación de estas herramientas software, su uso puede brindar múltiples beneficios como por ejemplo, la aplicación de estos modelos y software para mejorar la eficiencia en el uso de recursos, como el agua y los fertilizantes, reducir los costos de producción, optimizar la gestión de los cultivos y aumentar el rendimiento y la calidad de los productos agrícolas [14].

Aplicaciones relacionadas con Ganadería Ovina

A continuación se presentan los sistemas informáticos disponibles en el mercado para la gestión de la ganadería ovina. Es importante destacar que la mayoría de estos sistemas abarcan una amplia gama de tareas de gestión comercial y financiera, incluyendo ventas, compras, clientes y contabilidad fiscal, además de permitir un seguimiento detallado de los animales en términos de engorde, sanidad y mortalidad. Sin embargo, pocos de estos sistemas tienen en cuenta la gestión de la pastura para el mantenimiento adecuado del suelo, y no se encontró ninguna aplicación que utilice todos estos datos para realizar estimaciones productivas, por ejemplo, en combinación con la información meteorológica. A continuación se describen las aplicaciones informáticas encontradas:

Ovi Control

Es un software para ovinos desarrollado en colaboración con ganaderos y veterinarios del sector que cuenta con una amplia variedad de características, entre las que se destacan: control de hembras reproductoras, engorde, libros de registro, administración y control económico.

En el control de reproductoras, el software permite conocer la ficha histórica de las madres, la selección automática de ovejas por multicriterios, resultados individuales y generales, control sanitario y trazabilidad. En el engorde, se puede calcular los costes de producción, controlar los índices de conversión y ganancias, y realizar un seguimiento de la trazabilidad sanitaria y alimentaria. En cuanto a la administración, el software cuenta con funciones de contabilidad fiscal y analítica, seguimiento de la tesorería, cálculo de impuestos y edición de los modelos oficiales, y previsiones presupuestarias. Además, en el control económico, se puede llevar un registro de los proveedores y clientes, introducir y editar pedidos, facturas, y hacer un seguimiento de la trazabilidad comercial, así como realizar estadísticas de ventas y compras.

El software también cuenta con alertas específicas para la tasa de mortalidad en partos, litros de leche producidos por hembra y día, el coste del litro de leche, y las ovejas poco productivas, entre otras alertas. Por último, mediante informes, se pueden obtener datos específicos y de interés relacionados con destetes, pariciones, vacunas, envío al matadero, entre otros. El software permite hacer comparativas entre exportaciones, campos,

reproductoras, animales sementales, entre otras opciones, lo que resulta útil para veterinarios, asociaciones, cooperativas y fábricas de alimentos balanceados [70].

Ovitec

Este programa de gestión de ovinos está diseñado para mejorar la toma de decisiones de los agricultores y está disponible en plataformas web, escritorio y móvil. La herramienta cuenta con una amplia gama de características, como un panel de control que ofrece una vista general del campo desde la pantalla principal del sistema, registro de historial de trabajo, rutinas de manejo y control de hembras reproductoras, información sobre machos y reproducciones, registros de individuos, gestión de engorde, integración con dispositivos móviles compatibles con RFID y lectores de códigos de barras, sistema de alimentación automática, gestión de alimentos, gestión económica, informes de planificación de tareas y análisis, registros médicos, transferencia de animales entre campos, exportación de historiales de animales a un archivo y la posibilidad de importarlos a otra base de datos. Además, permite crear informes a través de múltiples campos, comparándolos por tipo de campo, región o cualquier otro criterio. También tiene una integración contable que permite la importación desde fuentes de datos externas, como programas de contabilidad externa o Microsoft Excel [71].

OvinApp

Es un software gratuito de gestión de ganado ovino y caprino basado en la nube que permite a los usuarios gestionar, controlar, analizar y tomar decisiones basadas en información proveniente de registros de administración de recursos y de manejo del ganado. Entre las características importantes de OvinApp se encuentran: gestión de animales que permite llevar un registro individualizado de cada animal, incluyendo información como raza, peso, edad, historial médico, estado reproductivo, entre otros; control de inventario que permite llevar un control detallado de los suministros y materiales utilizados en la explotación, así como los productos generados por el ganado (como la lana o la carne); planificación y seguimiento de actividades que permite planificar y realizar un seguimiento de las actividades diarias relacionadas con el manejo del ganado, como la alimentación, la vacunación, el esquilado, entre otros; registro de eventos para registrar eventos importantes, como la muerte de un animal, la venta de un lote, o la detección de alguna enfermedad en el ganado; análisis y reportes que permite generar informes personalizados sobre el rendimiento del ganado, la rentabilidad de la explotación y otros indicadores clave; además al ser una aplicación en la nube, permite acceder a la información desde cualquier lugar con conexión a internet y desde cualquier dispositivo [84].

Ovinca

Es un software de gestión integral para la cría de ovinos y cabras, desarrollado por USATI LTDA que es una empresa pionera y líder en Latinoamérica en el desarrollo de sistemas de información en la industria ganadera. Entre sus características destacadas, podemos mencionar: registro detallado del ganado que permite llevar un control exhaustivo de la información de cada animal, incluyendo su estado sanitario, movimientos y reproducción; manejo de la alimentación con herramientas para la planificación y gestión de la alimentación

del ganado, teniendo en cuenta su edad, peso y estado fisiológico; análisis de la rentabilidad a través de la gestión de los costos de producción y los ingresos por venta de animales que permiten realizar un seguimiento de la rentabilidad de la explotación ganadera; control de la reproducción que incluye herramientas para el control del ciclo reproductivo de los animales y el seguimiento de los resultados de la inseminación artificial; registro de eventos sanitarios que permiten llevar un control de las vacunas, tratamientos y otros eventos sanitarios que afectan al ganado [86].

Ranch Manager Open

Es un software de gestión de ganado utilizado para llevar un registro detallado de la producción ganadera específicamente diseñado para la gestión del ganado ovino.

Entre las principales características de este software, se encuentra el registro de animales, que permite llevar un seguimiento exhaustivo de cada animal, incluyendo información como su edad, peso, raza, estado de salud, medicación y otros datos importantes. Todo esto, a partir de múltiples etiquetas de identificación como metal, plástico, registro gubernamental, tatuaje y marca. Otra función importante es la gestión de forrajes, que permite llevar un registro de las pasturas del campo y su uso. Esto resulta fundamental para garantizar un manejo adecuado del ganado y la eficiencia en la producción. También se encuentra la función de registro de ventas y compras, que permite llevar un control detallado de las transacciones y precios asociados. La herramienta también cuenta con una sección de informes y estadísticas, que resulta muy útil para analizar el rendimiento de la producción. Por otro lado, la sincronización con dispositivos móviles facilita el acceso a la información y la gestión de los datos desde cualquier lugar. Además, la interfaz de usuario personalizable permite adaptar el software a las necesidades específicas de cada usuario. Es importante destacar que el software cuenta con una versión gratuita, Ranch Manager Open Lite, que ofrece funcionalidades básicas para la gestión de ganado y pastos. De esta manera, se facilita el acceso a esta herramienta a un mayor número de usuarios y se fomenta la gestión responsable y eficiente del ganado [83].

Los sistemas mencionados anteriormente son solo algunos ejemplos de las opciones que podemos encontrar en el mercado para gestionar un emprendimiento ganadero orientado a ovinos. El objetivo principal de estos sistemas es proporcionar información clara y precisa sin abrumar al usuario. Es común que los sistemas sobrecargados de tareas sean evitados por los dueños de los campos debido a su aparente complejidad inicial, por lo que se busca que estos sistemas posean interfaces intuitivas de uso para facilitar su adopción.

3.4.2 DSS en la ganadería argentina: ¿Un desafío pendiente?

En Argentina, existen diversas opciones de sistemas DSS orientados a la gestión agropecuaria, tanto de empresas privadas como por parte del gobierno. Estas herramientas ofrecen la posibilidad de hacer seguimiento de animales, gestionar las finanzas y contabilidad, todo ello teniendo en cuenta las regulaciones y trámites correspondientes en AFIP y SENASA, incluyendo la cumplimentación de todos los formularios necesarios. Además, la mayoría de estos sistemas permiten generar informes y estadísticas que resultan

útiles para la toma de decisiones. y su utilización puede ser de gran ayuda para mejorar la eficiencia y productividad en la gestión agropecuaria en Argentina.

CampoGI

Es un software de gestión ofrecido por AgroGI para empresas agropecuarias que ofrece una amplia gama de funciones que facilitan la gestión y el control de las operaciones diarias en el campo. Esta herramienta cuenta con un módulo de gestión de ganado, que permite llevar un registro completo y actualizado de los animales y sus actividades, como alimentación, vacunación, pesaje y reproducción. Además, incluye una sección de control de costos y producción, con la cual se puede llevar un registro detallado de los costos y los ingresos de la explotación ganadera. También permite realizar facturación electrónica y emisión de documentos a través de la AFIP. CampoGI también ofrece la posibilidad de generar informes personalizados y estadísticas así como de integrarse con otros programas de contabilidad y gestión empresarial mientras que permite analizar también la información económica-financiera y los costos productivos en distintas monedas [89].

Huella 4.0

Este software fue desarrollado por SoftHuella y es una solución de gestión de ganado ovino y bovino que permite a los usuarios llevar un control detallado de las actividades y recursos de su explotación. Entre sus características se incluyen el registro y seguimiento de animales, la gestión de inventarios, la planificación de actividades y eventos, el control de gastos e ingresos, y la generación de informes y estadísticas. El programa cuenta con una interfaz que puede ser personalizada para adaptarse a las necesidades específicas de cada usuario; y además, es compatible con dispositivos móviles, permitiendo el acceso remoto a la información en tiempo real. El software también incluye funciones de seguimiento sanitario y reproductivo, así como la posibilidad de integrar datos de identificación electrónica de animales. También dentro de las características se incluye la posibilidad de completar formularios de SENASA y formularios oficiales de otros países. Cuenta con una versión para productores y otra para veterinarios [87].

Gestión Ganadería

El software es ofrecido por Physis y está diseñado específicamente para la gestión de ganado. Esta aplicación tiene una amplia variedad de funciones que permiten al usuario llevar un registro detallado de su ganado, así como de las actividades integrales relacionadas con la producción. La aplicación permite la gestión de la información de los animales, como peso, raza, edad, sexo y fecha de nacimiento, y también cuenta con la posibilidad de cargar información sobre las enfermedades y tratamientos de cada animal. Además, tiene herramientas para controlar el estado sanitario del rebaño y para llevar un registro detallado de los gastos y los ingresos asociados a la producción; en caso de ser necesario, se pueden emitir facturas electrónicas, y otros documentos vinculados a la AFIP. También incluye un módulo para la generación de informes y estadísticas que permiten analizar el rendimiento del rebaño. La aplicación es compatible con dispositivos móviles y tiene la opción de sincronizar la información en tiempo real, lo que facilita la gestión del ganado desde cualquier lugar que cuente con conexión a internet [88].

SIF Móvil

Aplicación desarrollada por el Ministerio Nacional de Agricultura, Ganadería y Pesca que permite a los productores de bovinos y porcinos consultar desde su dispositivo celular, el resultado de la faena de sus tropas. La aplicación es gratuita y permite acceder en forma directa a información de rendimiento y calidad de la tropa declarada por el establecimiento faenador al Ministerio de Agricultura, Ganadería y Pesca. Está disponible en la tienda de aplicaciones Play Store y es compatible únicamente con sistema android [91].

CaprinApp

Es una herramienta de uso cotidiano para el productor caprino, con la cual es posible llevar un registro controlado de sus animales además de generar y gestionar información de utilidad para su sistema productivo, pudiendo así tomar decisiones con mayor grado de certeza. Dentro de los datos que se pueden gestionar se encuentran: la identificación individual de cada animal (adulto, cría o recría) con su raza, pesajes, procedencia y edad, entre otros ítems generales; asignación de servicios y ocurrencia de partos, lo que permite una mejor administración dentro del plano reproductivo; eventos sanitarios o de manejo en los corrales para programar calendarios recomendados por los médicos veterinarios, entre otros. La aplicación a su vez, brinda la opción de compartir los datos de la majada y sus indicadores productivos vía redes sociales para facilitar la comunicación con otros técnicos o asesores y así tener a disposición mayores herramientas para recomendar puntos de mejora o acciones particulares de trabajo. La app fue desarrollada por INTA y el Ministerio de Economía Argentino y está disponible en la tienda de aplicaciones Play Store [92].

En conclusión, resulta crucial contar con herramientas de apoyo a la toma de decisiones que estén focalizadas en la explotación y en las necesidades específicas del agricultor, y que además consideren las normas interoperables en los sistemas de ganadería extensiva. Es evidente que se requiere un consenso en cuanto a las normas de intercambio de datos, con el fin de demostrar el valor de los mismos, compartidos tanto a nivel de explotación como a nivel regional, lo que se traduce en beneficios comerciales y públicos. Por tanto, es fundamental seguir trabajando en la investigación y el desarrollo de sistemas de apoyo a la toma de decisiones en la ganadería, para lograr una gestión más eficiente y sostenible de los recursos naturales, y así mejorar la productividad y competitividad del sector ganadero [14].

3.5 ¿Cómo Afectarán estos Avances Tecnológicos a la Producción Ganadera?

Los avances tecnológicos en la producción ganadera pueden tener un impacto significativo en la eficiencia, la sostenibilidad y la rentabilidad del sector. En primer lugar, la implementación de tecnologías como la genética de precisión, la reproducción asistida, la alimentación y suplementación personalizada, y la monitorización en tiempo real del estado de los animales, pueden permitir una producción más eficiente y una mayor calidad de la carne, leche y otros productos derivados. Además, el uso de tecnologías para el monitoreo de la salud de los

animales puede permitir una detección temprana de enfermedades, reduciendo la propagación de enfermedades y minimizando el uso de antibióticos y otros tratamientos. También puede permitir una mejor gestión de los recursos naturales, como el agua y la energía, a través del uso de sensores y tecnologías de monitoreo para optimizar el uso de estos recursos. Sin embargo, algunas tecnologías pueden ser muy costosas de implementar, transformándose en una barrera para los productores de menor escala o con menos recursos financieros. La PLF requiere un conocimiento de las tecnologías involucradas y de los procesos de producción ganadera, por lo tanto, es necesario capacitar a los productores ganaderos para que comprendan cómo usar y aprovechar al máximo la tecnología, además de que puede requerir cambios culturales y de mentalidad productiva que puede ser difícil de implementar.

3.5.1 PLF e IoT: ¿Cuáles son los desafíos a nivel global?

PLF es ampliamente conocido como un enfoque para emplear insumos de manera eficiente y lograr los mejores resultados posibles, además de ser un sistema sostenible desde una amplia variedad de perspectivas. Según la literatura, IoT experimentará mejoras tecnológicas significativas como tecnología orientada a servicios. Sin embargo, vale la pena señalar que la responsabilidad de aprovechar las soluciones inteligentes que IoT proporcionará en la gestión del ganado seguirá estando en manos humanas. Este nuevo enfoque basado en el monitoreo y la tecnología podría plantear algunos desafíos en los sistemas de producción ganadera. Los principales retos por enfrentar de este sistema son resumidos a continuación [11].

Desafíos económicos

La industria alimenticia es una de las más grandes del mundo, pero también una de las más inestables debido a la dependencia directa o indirecta de las influencias climáticas y estacionales. Esta inestabilidad se refleja en los márgenes de beneficio mínimos que caracterizan a la industria. La adopción de tecnologías IoT para la producción de alimentos, como el uso de sensores confiables y eficientes, es vital para mejorar la estabilidad y rentabilidad de la producción. Sin embargo, el costo de estas tecnologías a menudo no es cuantificable y su dinámica de estimación es complicada, ya que la arquitectura IoT está en constante cambio y desarrollo. Además, en áreas remotas donde no se puede acceder a un servicio de Internet estable, los precios pueden aumentar, lo que hace que la inversión en tecnología sea más difícil de justificar [11].

Para abordar estos desafíos, es necesario contar con profesionales altamente capacitados en una variedad de campos, incluyendo ingeniería eléctrica, informática, ciencia de datos, ingeniería de sistemas biológicos y producción agrícola. Es de suma importancia invertir en la educación y capacitación de estos nuevos profesionales de la ganadería para garantizar que se desarrollen soluciones innovadoras y sostenibles en el tiempo para la producción de alimentos. En última instancia, el objetivo es mejorar la rentabilidad y estabilidad de la industria alimentaria, reducir su impacto ambiental y garantizar que se puedan satisfacer las necesidades nutricionales de una población mundial en constante crecimiento [11].

Aplicabilidad

Los agricultores rara vez son expertos en ciencias y tecnología. Por esta razón, se requiere un sector de servicios especializado para mantener los componentes tecnológicos, interpretar los datos recopilados por los sensores, formular y brindar consejos fáciles y apropiados a los agricultores de manera regular e involucrar a los usuarios en el desarrollo tecnológico. Además, sería deseable construir un marco completamente integrado a través del cual todos los componentes del sistema pudieran suministrarse a los usuarios finales y que estos pudiesen realizar el despliegue funcional de los sistemas PLF en las granjas de manera autónoma [11].

La falta de colaboración entre investigadores, ingenieros y proveedores de tecnología ha generado una baja tasa de adopción de tecnologías para la producción de alimentos en los campos. La producción e implementación de estas tecnologías a menudo falla cuando los componentes del dispositivo se desarrollan de forma independiente y compiten con productos ya existentes. Superar este obstáculo y lograr una mejor coordinación entre desarrolladores y proveedores de herramientas de producción de alimentos representa el mayor de los desafíos, que de lograrse, puede resultar en la producción de sistemas integrados de mayor calidad [11].

Los sistemas basados en IoT que se utilizan en PLF suelen implicar el uso de sensores de todo tipo y es importante que dichos sensores coincidan con la realidad de los campos, sin causar una pérdida excesiva de tiempo para los productores, sino más bien ahorrando un tiempo valioso. De lo contrario, se perderían las ventajas de estos nuevos sistemas. Por ejemplo, los sensores deben tener baterías de larga duración para permitir que el monitoreo continuo en tiempo real sea efectivo. Por último, los sistemas PLF se caracterizan a menudo por una cierta heterogeneidad. A pesar de los considerables esfuerzos para abordar este problema, se necesita más trabajo para establecer un equilibrio entre el uso de sistemas fiables, energética y económicamente sostenibles, intuitivos y bien cubiertos [11].

Aspectos Éticos

La introducción de los sistemas de producción de alimentos basados en la tecnología IoT plantea preocupaciones éticas en relación con la pérdida de empleo en el sector agrícola. La automatización del control y cuidado del ganado, por ejemplo, reduciría la necesidad de trabajadores agrícolas y haría obsoletas muchas de las habilidades actuales en los campos productivos. Sin embargo, la relación entre el empleo agrícola y la producción de alimentos es crucial y perder estos trabajos podría tener un impacto significativo en el individuo y la sociedad en general. Las prácticas agrícolas convencionales se centran en tratar a los animales de forma individual, lo que favorece la relación entre humanos y animales. Esta forma de cuidado podría verse amenazada con la implementación de sistemas PLF basados plenamente en tecnología IoT. Además, el cambio podría resultar en una pérdida de interés por parte de los ganaderos en su profesión y en la relación con los animales, lo que tendría importantes consecuencias a largo plazo en la economía y la opinión pública sobre la PLF. Otra preocupación ética específicamente relacionada con la tecnología PLF es la capacidad de los ganaderos para cumplir con sus responsabilidades en el cuidado de los animales a través del monitoreo de un circuito cerrado. Si bien la tecnología de monitoreo puede ser

eficaz para el control y la supervisión de los animales, es importante que los ganaderos mantengan una conexión emocional con su trabajo y su ganado para garantizar que los animales reciban el mejor cuidado posible. Las preocupaciones sobre la propiedad de los datos, cómo se usan y almacenan y quién tiene acceso a ellos también pueden contribuir a la falta de fe en la tecnología entre los agricultores y los asesores y generar sentimientos de vulnerabilidad, especialmente en situaciones en las que las cámaras graban constantemente. Debido a que grandes cantidades de datos se almacenan rutinariamente en «nubes», también existen preocupaciones sobre la seguridad y la integridad de los mismos [11].

Finalmente es crucial tener en cuenta la opinión de los consumidores, ya que su decisión de compra es la que finalmente determina el éxito o el fracaso de las cadenas de valor que utilizan PLF. Además, la inclusión de las perspectivas de los clientes desde el principio del proceso de desarrollo de la innovación puede mejorar la aceptación social de los avances tecnológicos. Para entender mejor la opinión de los consumidores sobre la producción ganadera de precisión, se llevó a cabo una encuesta. A pesar de los numerosos beneficios descubiertos durante la investigación, la encuesta reveló tres preocupaciones comunes de los consumidores. En primer lugar, existe la preocupación de que la integración de tecnologías PLF aumentará la industrialización en la producción ganadera, lo que podría tener consecuencias negativas para el bienestar animal y la sostenibilidad a largo plazo. En segundo lugar, los consumidores están preocupados por la vulnerabilidad de las tecnologías y los datos de PLF relacionados al uso indebido y al delito cibernético. A medida que la tecnología avanza, es importante garantizar que las tecnologías utilizadas en la producción ganadera sean seguras y protejan los datos de los consumidores. En tercer lugar, los consumidores expresaron que la información sobre PLF no se comunica lo suficiente como para permitir decisiones de compra informadas. Esto sugiere que los consumidores necesitan más información sobre las tecnologías PLF y cómo se utilizan en la producción ganadera para tomar decisiones informadas sobre lo que compran [11].

3.5.2 ¿Argentina 4.0?

La incorporación de tecnologías avanzadas en la ganadería ha dado lugar a un cambio radical en la forma de gestión de la producción. Esto supone un gran desafío para los productores argentinos, quienes deben competir en un mercado global cada vez más exigente en cuanto a la calidad de los productos y la eficiencia en los procesos de producción. La adopción de estas tecnologías conlleva una inversión significativa, un costo adicional de mantenimiento y personal capacitado, pero los resultados económicos a largo plazo justifican su incorporación. Además, el sector ganadero se enfrenta a desafíos importantes como la falta de interés y mano de obra, los márgenes de beneficios reducidos y la migración campo-ciudad que afectan a la producción animal. Por estos motivos es crucial impulsar la adopción de tecnología para revertir estos efectos, reducir la brecha tecnológica y fomentar la integración de diferentes sistemas de procesos. Las políticas públicas y programas de investigación en tecnología ganadera deben ser parte de las plataformas de impulso para responder a las nuevas demandas [107].

Es importante destacar que la adopción de tecnología también puede ser un catalizador para mejorar la calidad de vida de los productores y reducir el impacto ambiental de la producción ganadera, lo que a su vez contribuirá a la sostenibilidad del sector y al arraigo rural. Las nuevas tecnologías permiten el monitoreo y la gestión remota del rodeo, lo que aumenta la eficiencia y reduce costos, mientras que ofrece una oportunidad para la conservación de especies silvestres, adaptando el uso de tecnologías en la gestión de su monitoreo y conservación. Por otro lado, el costo de inversión y mantenimiento de estas tecnologías puede ser una limitante para su adopción en la actividad ganadera. Por lo tanto, es necesario impulsar programas de políticas públicas y de investigación en tecnología ganadera, así como fomentar la articulación público-privado para el diseño de soluciones adaptadas a cada contexto. En este sentido, se debe reflexionar sobre la opinión de los consumidores y las percepciones de los mismos deben incorporarse en la investigación de PLF, ya que el éxito o el fracaso de sus cadenas de valor está determinado en última instancia por las decisiones de compra de los consumidores. La inclusión de las perspectivas de los clientes al principio del proceso de desarrollo mejora la aceptación social de los avances tecnológicos [107].

3.5.3 Los retos de la implementación de PLF en la ganadería ovina de Chubut.

En la Patagonia, la disminución de hectáreas productivas para la ganadería ovina ha llevado a una baja carga promedio de ovinos por hectárea. Sin embargo, gracias a las nuevas tecnologías satelitales del INTA, se ha determinado que la carga ovina está correcta en función a la disponibilidad de forraje y que incluso hay una leve tendencia positiva en el estado general de los suelos en los últimos 20 años, a pesar del crecimiento descontrolado de las poblaciones de guanacos que habitan las regiones cuya dieta es compartida en un 80% con los lanares. El abandono de áreas y la conservación de tierras, ya sea de manera formal o por elección, están teniendo consecuencias no deseadas, como el aumento de la población de depredadores, en particular el zorro y el puma. Los patrones climáticos también han cambiado significativamente en los últimos 50 años, y ahora los productores ovinos se enfrentan a sequías recurrentes en lugar de inviernos largos y crudos. Estas sequías afectan la producción de pasto, disminuyen los niveles de agua subterránea y aumentan la inversión en infraestructura de agua como aguadas, molinos y bombas solares [64].

Para mantener un negocio rentable, los productores necesitan buscar más tierras para mantener un equilibrio mínimo en el nivel de ganado. Las pérdidas en la producción agropecuaria son un factor preocupante y muchas veces subestimado en cada establecimiento, lo que puede llevar al cierre de emprendimientos. Para evitar esto, es necesario implementar buenas prácticas y certificaciones en las producciones y buscar alternativas genéticas y económicas para mejorar la rentabilidad. Los productores más avanzados ya están implementando estas medidas, pero otros necesitan empezar a recorrer un camino similar. Sin embargo, la capacidad financiera de cada uno será determinante en la rapidez con la que puedan hacerlo. En los próximos años, será necesario priorizar varios temas estratégicos en el sector ovino. Ante la previsión de veranos cada vez más secos y calurosos, y con inviernos rigurosos intermitentes, es esencial implementar la suplementación a gran escala. Un caso específico donde se demostró su efectividad fue durante el duro invierno del año 2020, ya que invertir el 10-15% del valor de una oveja permitió salvar tanto a la oveja como a su cordero.

Además, es fundamental llevar a cabo una suplementación estratégica a lo largo del año para medir y corregir la condición corporal de los animales en diferentes etapas del ciclo reproductivo. Si bien estas prácticas han sido probadas con éxito en la región, su adopción requiere de una inversión significativa [64].

El Problema del Clima en Chubut

La Provincia del Chubut se encuentra dentro de las latitudes medias del hemisferio Sur, siendo éste uno de los condicionantes más importantes de su clima. Según la clasificación climática de Köppen - Geiger , basada en la temperatura, la precipitación y la vegetación, podemos ver en la figura 3.5.1 [30], que dentro de la provincia se encuentran diferentes tipos de climas [29].

Tipos de clima del Chubut



Tipos de clima según Köppen

■ BWk – Árido frío	■ Cfc – Oceánico templado subpolar
■ BSk – Semiárido frío	■ Dsb – Continental mediterráneo
■ Csb – Mediterráneo oceánico	■ Dsc – Continental mediterráneo subpolar
■ Csc – Mediterráneo subalpino	■ Dfc – Subpolar húmedo
■ Cfb – Oceánico templado	■ ET – Tundra

Fuente: Present and future Köppen-Geiger climate classification maps at 1-km resolution. Nature Scientific Data. Beck, H.E., Zimmermann, N. E., McVicar, T. R., Vergopolan, N., Berg, A., & Wood, E. F.

Figura 3.5.1. Chubut Köppen.

Por lo tanto el clima es frío y húmedo en la zona occidental o andina, descendiendo las temperaturas hacia el sur. El centro de la provincia es predominantemente árido y de grandes

amplitudes térmicas con una zona de oasis dada por los lagos Colhué Huapi y Musters, mientras que más al este, en el valle inferior del río Chubut y en la costa el clima se mantiene aún bastante seco si bien beneficiado por la humedad proveniente del Mar Argentino y la acción moderadora térmica del influjo oceánico. El clima de la provincia no es uniforme, sino que se distinguen tres zonas geográficas con tipos de clima bien diferenciados [31]:

- ❖ El sector de los Andes posee un clima frío por la latitud y muy húmedo a causa de los vientos que llegan del Pacífico. Las precipitaciones van de los 700 a los 2.500 mm, y se producen en los meses invernales. La temperatura media es de 8° C [31].
- ❖ La zona de las mesetas y terrazas escalonadas es de clima árido con veranos calurosos, de días soleados y prolongados e inviernos fríos; el otoño y la primavera son muy breves. Las lluvias disminuyen hacia el este y los vientos del oeste son fuertes y constantes, fenómeno que erosiona los suelos e impide el crecimiento de la vegetación. En la zona del litoral atlántico, el mar no llega a moderar el clima regional, ya que las costas son elevadas [31].
- ❖ El área de Península de Valdés tiene un clima con veranos cálidos e inviernos fríos. Es una zona de transición entre los climas templados del centro del país, con lluvias estivales, y los climas fríos con lluvias invernales. El viento, con dominio del cuadrante oeste, reina en la meseta alternando en el litoral con frecuentes sudestadas. Las temperaturas máximas son de 40°C y más y las mínimas de -10°C. El aire se caracteriza por ser seco o muy seco, con unos 170 a 250 mm de lluvia anuales. Además, la presencia de heladas cubren 248 días como promedio. Esto se traduce en la aridez general del área [31][32].

Estas condiciones ambientales caracterizan al área geográfica como marginal para la producción ovina, actividad de elevado riesgo si se consideran además los eventos climáticos extremos y cíclicos de sequía, temporales y nevadas [24]. En cuanto a las consecuencias, suele considerarse en estudios científicos o técnicos que las sequías generan tanto la reducción del forraje (con impactos variables según la intensidad, frecuencia y/o duración del fenómeno); como de las provisiones de agua para la población y los animales. En relación a la ganadería, ambas cuestiones perjudican la productividad, disminuyendo la cantidad de kilos de lana y carne obtenidas en un ciclo productivo. Cuando la intensidad del fenómeno es profunda o se extiende en el tiempo, afecta la condición corporal de los animales llegando a provocar su muerte o la de sus corderos, la disminución de los niveles de preñez y por ende de pariciones. Estas pérdidas generan dificultades para la reposición del ganado perdido, lo que gradualmente lleva al envejecimiento de la hacienda. Como consecuencia, se puede producir la descapitalización de los productores y el deterioro de sus ingresos, siendo difícil la posterior recuperación durante los siguientes ciclos productivos [25]. Por lo tanto las condiciones climáticas afectan directamente a la productividad del pastizal y animal de los sistemas ganaderos extensivos, generando una cierta inestabilidad productiva y económica a las empresas ovinas [24]. La degradación generalizada que se observa en la zona tanto de los pastizales como del suelo, en gran parte originada por un manejo inadecuado, constituye el principal problema ecológico productivo de la región [26]. La sequía tuvo una importante relevancia debido a que generó alteraciones en los medios de vida, en función de sus consecuencias en el ámbito de la producción y del trabajo. Expertos sostuvieron la

importancia de modificar las formas de producción de los ganaderos, para que se puedan «adaptar» a las sequías, que son cada vez más recurrentes en los últimos años y que, con esas medidas, sus efectos sean menos notorios. De esta manera, propusieron tecnologías y prácticas de manejo que permitieran generar menores consecuencias negativas en lo productivo, como la realización de diagnósticos en los animales, ajuste de carga animal por hectárea, pariciones controladas y traslados hacia zonas con mayores pasturas [25]. A pesar de que las prácticas propuestas permiten un uso sustentable de los recursos naturales y un aumento en la eficiencia de la producción, la adopción de éstas aún no se ha generalizado. La esquila preparto, práctica que permite un incremento en el número de corderos señalados y mejoras en la calidad de la lana, ha logrado una mediana adopción (menor a 50%) al igual que PROLANA (esquila desmaneada, clasificación y acondicionamiento de lana y mejor precio de la misma). La suplementación estratégica y de emergencia asociada al seguimiento de la condición corporal a lo largo del año no es una práctica habitual en los productores, aunque se observa una tendencia en su adopción otorgando una mayor previsibilidad a los sistemas productivos extensivos. Por último, la realización de pariciones controladas (cobertizos) constituye una práctica que tiene relativa adopción sólo en pequeños productores.

Sin detallar cada una de las prácticas disponibles y su nivel de adopción predial, se puede decir, que en términos generales, el uso de las prácticas probadas que permiten incrementar la productividad de la ganadería, mejorar la calidad de los productos primarios resultantes de la actividad, y generar impacto y estabilidad a los sistemas, no está generalizado, ya sea por cuestiones culturales, económicas, políticas o de estrategia de intervención [26].

El Problema de la Adopción de tecnologías informáticas en Chubut

Cada vez más las organizaciones tienen la necesidad de ser más competitivas y tener información real disponible para la toma de decisiones. Hoy se requiere de herramientas orientadas a disminuir los tiempos a la hora de tomar una decisión, generar información confiable, disminuir costos, aumentar productividad y conducir a múltiples análisis de «¿qué pasaría si?».

Los sistemas que hoy apoyan el proceso de toma de decisiones están orientados a los altos ejecutivos y personas que tienen esta responsabilidad dentro de una organización, estos son conocidos como DSS, EIS y numerosos sistemas expertos que proveen información en tiempo real, de manera gráfica y resumida a sus usuarios finales [27]. En la provincia del Chubut las personas que se dedican a la producción agropecuaria, basan generalmente su actividad en la ganadería ovina extensiva. De un total de 4.089 Unidades Ganaderas (UG), un 37% de los productores se encuentra en los departamentos de Cushamen y Gastre y el 80% de los productores chubutenses tienen menos de 1.000 ovinos; este dato refleja un sector productor de muy baja escala. Por otro lado, la mayor cantidad de ovinos se ubica en los departamentos de Tehuelches y Río Senguer, donde se concentra el 31% de la existencia de ovinos del Chubut. Desde el punto de vista de la capacidad de gestión para la toma de decisiones, y tomando como unidad mínima un establecimiento con 3.000 animales, se observó en la encuesta del 2014 que el 94% de los productores en Chubut tienen una baja a nula capacidad de tomar decisiones [21].

Se sabe que la apropiación de sistemas como los DSS en la agroindustria es lenta y baja, debido a la complejidad que perciben sus usuarios, las pocas estrategias de adopción y a la necesidad de modificar patrones de conducta. Hay quienes agregan que, como consecuencia de no usar Tecnologías de la Información (TI) en los sistemas productivos actuales se generan errores o retrasos en las decisiones, desconocimiento de las pérdidas o ausencia de utilidades, desestimación o abandono y por ende migración de productores hacia las ciudades. Las TIC no solo permiten el control de los procesos organizacionales para mejorar la eficiencia del proceso en sí, sino que el sector ganadero también agrega valor al brindar la posibilidad de trazabilidad desde la gestación del animal hasta la venta o consumo de los productos [28].

Desde el INTA los gobiernos provinciales y el gobierno nacional, se están llevando a cabo proyectos para fomentar la adopción de nuevas tecnologías entre los ganaderos. Esto se está logrando a través de la realización de capacitaciones y talleres en los que se enseña el manejo y uso de estas herramientas. Además, se están ofreciendo incentivos económicos y beneficios fiscales para aquellos que decidan implementar estas tecnologías en sus sistemas productivos. Todo esto con el objetivo de aumentar la eficiencia y sostenibilidad de la producción agropecuaria, reducir el impacto ambiental y mejorar la calidad de vida de los productores y sus familias. A pesar de estos esfuerzos, todavía queda mucho por hacer para lograr una adopción generalizada de estas nuevas tecnologías en el sector ganadero.

Capítulo 4: Datos y modelos

El objetivo de este capítulo es presentar el proceso completo de obtención de los modelos de inteligencia artificial utilizados en la aplicación desarrollada, los cuales se basan en datos climáticos y productivos de distintas regiones de la provincia. Se describirá en detalle el proceso de adquisición de los datos, considerando la interacción con fuentes de información abiertas para su obtención y el procesamiento llevado a cabo para su posterior análisis y utilización. Además, se ahondará en cómo fueron medidos durante el proceso de entrenamiento, detallando el proceso de selección de los modelos finales. Todo esto permitirá comprender la metodología utilizada para generar los modelos de inteligencia artificial que son la base de la aplicación.

4.1 Interacción con fuentes abiertas de datos

En los últimos años, se ha producido una evolución en el acceso a información de carácter público, denominado «Datos Abiertos» u «Open Data». Esto implica que los datos de interés común sean puestos a disposición de la sociedad para que puedan ser utilizados en la generación de nuevas ideas y aplicaciones, con el objetivo de fomentar la innovación, el conocimiento y las oportunidades de negocio. Los datos abiertos son aquellos que están disponibles en formatos reutilizables, siguen estándares, buenas prácticas y usan explícitamente una licencia libre que permite su reutilización por terceros. Este movimiento no sólo busca aumentar la transparencia en la gestión pública, sino que también respalda el concepto de un «Gobierno Abierto», mejorando la interoperabilidad entre entidades públicas

y permitiendo que la sociedad cree sus propias aplicaciones en lugar de que el Estado las genere [123].

4.1.1 Datos abiertos y públicos

La apertura de los datos permite que los datos ocultos o restringidos sean accesibles públicamente para que la ciudadanía y otros agentes puedan aprovecharlos. La filosofía de esto denominado datos abiertos, persigue que ciertos tipos de datos estén disponibles libremente sin restricciones de derechos de autor, patentes u otros mecanismos de control, para que cualquier persona los pueda acceder, usar, reutilizar y compartir libremente, según el decálogo del reutilizador de datos del sector público. Existe un concepto de datos públicos, que no debe confundirse con datos abiertos, ya que los primeros son generados o recolectados por el gobierno, y están disponibles para el acceso y uso del público en general, siempre y cuando no estén restringidos por la ley, la privacidad o la seguridad. Pero para que los datos sean considerados como datos abiertos, es necesario que estén disponibles en línea para que cualquier persona pueda acceder, utilizar y compartirlos. Además, es esencial que estos datos estén estructurados de una manera que permita su procesamiento automático y que se publiquen en formatos reutilizables como CSV o JSON en lugar de formatos propietarios que requieren licencia paga para su uso. Es importante también que se adhieran a una licencia explícita que permita la reutilización de los datos con cualquier fin, incluyendo el comercial. Sin una licencia, los datos no se consideran realmente datos abiertos y es posible que las personas usuarias carezcan de permiso para acceder, utilizar y compartirlos debido a los derechos de autor o la legislación sobre conjuntos de datos [124][125].

De acuerdo con la Open Knowledge Foundation, hay diez principios que definen los datos abiertos del gobierno. Primero, los datos públicos que se consideran abiertos deben ser completos, a menos que estén sujetos a limitaciones como la privacidad o la seguridad y deben estar electrónicamente almacenados. Segundo, deben ser primarios y publicados tal como fueron recogidos sin procesamientos y formas agregadas, con el mayor nivel de detalle posible. Tercero, los datos también deben ser oportunos y, cuarto, accesibles a la gama más amplia de usuarios posibles, por lo tanto se deben actualizar de forma ágil. Quinto, deben ser procesables por máquinas y, sexto, el acceso no debe ser discriminatorio, sin necesidad de registro. Séptimo, los datos no deben usar formatos propietarios, sino que deben estar disponibles en un formato sobre el que ninguna entidad tenga el control exclusivo. Octavo, los datos deben ser publicados con licencias libres y, noveno, deben permanecer accesibles a lo largo del tiempo. Como último principio, el acceso a los datos debe ser sin costo. Las únicas excepciones permitidas son aquellas relacionadas con cuestiones de privacidad, seguridad y restricciones de privilegios sostenidas por otras leyes [126]. Otra forma de definir los datos públicos abiertos, es a través de seis principios establecidos en la Carta Internacional de Datos Abiertos. El primer principio es «abierto por defecto», lo que significa que los datos deben estar disponibles al público a menos que exista una justificación clara para mantenerlos cerrados. El segundo principio es la oportunidad y la exhaustividad en la publicación de datos, lo que significa que la información debe ser relevante y actual. El tercer principio es la accesibilidad y la usabilidad, lo que implica que los datos deben ser fáciles de encontrar y legibles por máquinas. El cuarto principio es la comparabilidad e

interoperabilidad, lo que significa que los datos deben ser fáciles de comparar y combinar con otros conjuntos de datos. El quinto principio es mejorar la gobernanza y la participación ciudadana, lo que significa que la transparencia de los datos puede mejorar los servicios públicos y ayudar a que los gobiernos rindan cuentas. El sexto y último principio es el desarrollo inclusivo y la innovación, lo que significa que los datos abiertos pueden impulsar el desarrollo económico inclusivo y abordar desafíos globales como el cambio climático [127].

Es esencial que los ciudadanos tengan acceso a los datos generados por la administración, ya que estos se financian con los impuestos de toda la población. La utilización de estos datos por parte de la sociedad puede generar nuevos servicios y soluciones creativas que la administración no está obligada ni capacitada para ofrecer. La disponibilidad de datos abiertos se ha convertido en una práctica cada vez más común para promover la transparencia en la gestión pública y fomentar la innovación y el conocimiento a través del acceso libre a la información. En este sentido, es fundamental asegurar la confianza de los ciudadanos en que sus datos personales serán protegidos adecuadamente y en que los datos abiertos seguirán siendo relevantes, completos y accesibles [123].

4.1.2 Portales de Datos en Argentina

Los portales de datos abiertos son una herramienta clave en el proceso de apertura de datos, ya que permiten almacenar, compartir, conectar y visualizar los conjuntos de datos. Además, se convierten en un punto de encuentro entre el gobierno, organizaciones, empresas, ciudadanos, investigadores y periodistas interesados en la información pública. Existen diferentes modelos de portales de datos abiertos, siendo los más completos aquellos que cuentan con secciones específicas para facilitar el acceso a los distintos conjuntos de datos, incluyendo opciones de búsqueda, metadatos, información sobre licencias y acceso a los datos. Asimismo, en estos portales se pueden encontrar herramientas para trabajar con los datos publicados, aplicaciones que se han desarrollado con ellos, e historias construidas a partir de los datos [125].

Durante el desarrollo de esta tesina, nuestro objetivo fue recopilar información productiva sobre la ganadería ovina en la provincia de Chubut, incluyendo datos sobre la producción de lana y carne, y también información climática de la región, abarcando un período de al menos 20 años. Para ello, recurrimos a diversos portales de datos, siendo el Portal de Datos Abiertos de Argentina uno de los más destacados, gracias a su sección especializada en agroganadería y la información que brinda sobre este sector. Además, consultamos el tablero de ovinos del Ministerio de Agricultura, Ganadería y Pesca (MAGyP) de Argentina, que proporciona estadísticas, leyes, informes y otros recursos relacionados con la producción ovina. También exploramos plataformas que, si bien no están catalogadas como portales de datos, proveían información útil para nuestro estudio, como la sección de servicios e información de la Fundación Barrera Zoofitosanitaria Patagónica (Funbapa), que ofrece datos sobre los ingresos y egresos agroeconómicos de la Patagonia, la sección de estadísticas de la Federación Lanera Argentina y el SIPAS del INTA, que brinda información sobre los proyectos, actividades y

servicios del INTA en la Patagonia. Todos estos recursos han sido esenciales para llevar adelante nuestra investigación y el desarrollo de esta tesina.

Datos Argentina

El sitio web «Datos Argentina» [128] es el portal de datos abiertos del gobierno Argentino, y tiene como finalidad poner a disposición de la ciudadanía una amplia variedad de datos y estadísticas de diversas áreas temáticas, como economía, salud, educación, transporte, medio ambiente, justicia, agroganadería, entre otras. En la página principal del sitio, los conjuntos de datos se encuentran clasificados por categorías y existe una sección de exploración que permite buscarlos y filtrarlos. Además, el sitio cuenta con secciones especiales para desarrolladores y para el sector empresarial, donde se pueden encontrar herramientas y recursos para el uso y análisis de los datos, y también contiene una sección de datos seriales.

Uno de los primeros conjuntos de datos explorados fue el de movimientos ovinos del período 2013 al 2018 [129]. Este conjunto de datos incluye información sobre los movimientos de ovinos por categoría y mes durante ese periodo, y fue recopilado por SENASA. Según los metadatos del dataset, los datos se actualizan mensualmente siendo la última actualización realizada el 19 de noviembre de 2019 y se encuentra en formato CSV. El conjunto de datos consta de 16 columnas, que incluyen la fecha en formato ISO, la ubicación de salida y entrada, la provincia y departamento de origen, la provincia y departamento de destino, así como el número de animales en diferentes categorías, como carneros, ovejas, borregos, capones y corderos. Aunque fue examinado cuidadosamente, se determinó que este conjunto de datos no era adecuado para nuestros modelos y no se pudo encontrar una manera efectiva de utilizarlo. Sin embargo, sigue siendo un conjunto de datos potencialmente interesante para futuras investigaciones.

Otro conjunto de datos evaluado fue el de existencias ovinas de 1875 a 1964 [130]. Este dataset recopila datos anuales provinciales sobre las existencias ovinas en Argentina durante un período de casi un siglo, aunque existen años sin registros asentados. La Dirección de Ovinos, Caprinos y Camélidos fue la encargada de reunir esta información, y según los datos adicionales sobre el dataset, el formato es CSV, la frecuencia de actualización es anual, la fecha de creación fue el 26 de octubre de 2018 y el último cambio se realizó el 17 de enero de 2019. El conjunto de datos cuenta con seis columnas que incluyen el ID del país, el nombre del país, el ID de la provincia, el nombre de la provincia, el año y la cantidad de cabezas de ovinos (un número decimal). A pesar de que no encontramos información útil para nuestros modelos, ya que necesitábamos datos a partir de 1985 y este conjunto de datos termina antes de ese año, consideramos que puede ser valioso para futuras investigaciones en las que se desee analizar el crecimiento o la disminución de la cantidad de ovinos en las provincias argentinas a lo largo de los años.

Revisamos otro conjunto de datos llamado Existencia Ovinas 1965-2017 [131], que también presenta información anual sobre las existencias ovinas en Argentina, pero esta vez desde 1965 hasta 2017. Sin embargo, algunos años no cuentan con registros. Este conjunto de datos

fue recopilado por la Dirección de Ovinos, Caprinos y Camélidos y según los metadatos se encuentra en formato CSV, la frecuencia de actualización es anual, la fecha de creación es el 26 de octubre de 2018 y la última actualización se realizó el 30 de abril de 2019. Este conjunto de datos consta de nueve columnas que incluyen la fuente (duplicada en dos atributos), el ID del país, el nombre del país, el ID de la provincia, el nombre de la provincia, el año de inicio, el año de finalización y la cantidad de cabezas de ovinos. A pesar de que algunos años no se registraron, esta información podría ser valiosa para futuros estudios que analicen el crecimiento o la disminución de la población ovina en diferentes provincias. Nosotros nos enfocamos en los datos de la provincia de Chubut, específicamente desde 1985 hasta 2017, los cuales utilizaremos para combinarlos con otros conjuntos de datos.

Además, revisamos un conjunto de datos de serie de tiempo de existencias ovinas entre 1875 y 2017 [132]. Este conjunto de datos, creado el 16 de mayo de 2019 y con el último cambio el 10 de octubre de 2019, cuenta con una frecuencia de actualización anual y formato en CSV. La información es recopilada por la Dirección de Ovinos, Caprinos y Camélidos y consta de 26 columnas, incluyendo un índice de tiempo en formato ISO y las provincias de Argentina con la cantidad de cabezas de ganado ovino en cada una de ellas. Aunque la información es interesante, no resultó útil para los modelos que teníamos pensado entrenar.

El siguiente conjunto de datos analizado se denomina «Ovinos por provincia y EAPs - CNA-02» [133]. La sigla CNA hace referencia a la Clasificación Nacional de Actividades Económicas de Argentina y, en este caso, el «-02» hace referencia a la clasificación específica utilizada para las actividades agropecuarias y forestales. Este dataset presenta información sobre la cantidad de ovinos por provincia y tipo de explotación agropecuaria, en términos de cabezas y EAPs, lo que significa que se está registrando la cantidad de ovinos por cada unidad productiva dedicada a la actividad agropecuaria en la provincia correspondiente. El responsable de recopilar estos datos fue el Instituto Nacional de Estadística y Censos (INDEC), y según la información adicional del dataset el formato utilizado es CSV, la frecuencia de actualización se indica como «eventual», la fecha de creación fue el 14 de febrero de 2020, y el último cambio se realizó el 19 de febrero de 2020. El conjunto de datos consta de siete columnas: el año, en formato ISO de tiempo; el nombre de la provincia; el código de la provincia; la cantidad de cabezas ovinas en explotaciones con límites definidos; la cantidad de explotaciones con límites definidos que tienen ovinos; la cantidad de cabezas ovinas en explotaciones sin límites definidos; y la cantidad de explotaciones sin límites definidos que tienen ovinos. Aunque no utilizamos esta información en nuestros modelos, el conocimiento sobre las EAPs podría abrir nuevos campos de investigación.

El siguiente dataset explorado fue el de indicadores de ovinos, lana y carne [134]. El dataset analizado contiene información sobre los indicadores de ovinos para valores anuales, trimestrales y mensuales. Según los metadatos, este conjunto de datos es recopilado por el Ministerio de Economía a través de la Subsecretaría de Programación Regional y Sectorial, en formato CSV, la frecuencia de actualización es trimestral y su fecha de creación fue el 1 de octubre de 2019, siendo su última actualización el 2 de marzo de 2023. A pesar de que ni en

el portal, ni en la página de origen se encuentra información detallada acerca de las columnas, al descargar el dataset se pudo observar que contenía 14 columnas. La primera corresponde al ID del sector, seguida del nombre del sector y un ID de variable, que parecía indicar un identificador para las variables observadas con un valor entre 1 y 18. La columna «actividad» describe el producto o actividad, como por ejemplo la carne ovina, lana fina, mediana o gruesa, entre otras. Luego se presenta un indicador que puede corresponder al consumo aparente, consumo per cápita, producción o precio interno de referencia. La columna «unidad de medida» indicaba la unidad de medición de cada indicador, por ejemplo toneladas de carne con hueso o kilogramos per cápita. La fuente de información también se encuentra como columna, y puede ser MAGyP, FLA o SENASA, mientras que la frecuencia de actualización se encuentra en la columna siguiente, pudiendo ser anual, mensual, trimestral, y otras. El dataset también cuenta con información acerca de la cobertura, que siempre es nacional, y del alcance, que puede ser a nivel país, provincia o departamento. El ID y nombre del alcance también se encuentran especificados en columnas separadas. También existe una columna para el índice de tiempo, es decir, la fecha, y finalmente la última columna es el valor correspondiente al producto o indicador específico de esa fila. A pesar de la gran cantidad de información que contenía el dataset, se optó por utilizar únicamente los datos referentes a las existencias o stock ovino en la provincia de Chubut, dividido por departamentos, y a la lana en dicha provincia. De esta manera, se redujo la información a un subconjunto que resultó ser relevante para el objetivo de la investigación.

Además, también evaluamos otro conjunto de datos de existencias ovinas [135] que muestra las existencias desglosadas por provincia, partido y departamento, clasificadas por categoría, con totales anuales desde marzo de 2008 hasta la última actualización. Según información adicional, el responsable de recopilar estos datos fue la Dirección Nacional de Sanidad Animal, la frecuencia de actualización es anual y la fecha de creación fue el 12 de septiembre de 2019, con la última actualización el 28 de noviembre de 2019. Por lo tanto, las existencias en este conjunto de datos se extienden hasta ese año. El conjunto de datos consta de 10 columnas que representan el año correspondiente a la recolección, en formato ISO, la provincia, su nombre y código, el nombre del partido o departamento y su código, la cantidad de carneros (machos adultos en edad reproductiva), ovejas (hembras adultas en edad reproductiva), borregos (machos jóvenes de reposición), capones (machos castrados), corderos y corderas (crías de ambos sexos). De este conjunto de datos, nos enfocamos en los datos de Chubut entre 2008 y 2019, y para complementar la información faltante del año 2020, utilizamos otra fuente del portal de Senasa [136], donde encontramos información actualizada. Este portal también muestra las existencias para el año 2022, pero la información del año 2021 no está disponible.

El último conjunto de datos examinado se denomina «Movimientos de subproductos animales 2013 a 2018» [137]. Este conjunto de datos recopila información sobre los movimientos de los productos animales durante el período de 2013 a 2018, en formato CSV. El Servicio Nacional de Sanidad y Calidad Agroalimentaria fue el encargado de recolectar esta información, que se actualiza mensualmente, la fecha de creación fue el 27 de agosto de 2019 y el último cambio se realizó el 5 de septiembre de 2019. El conjunto de datos cuenta

con 22 columnas, entre las que se incluyen la fecha en formato ISO, el lugar de origen y destino de los productos, el nombre y código de la provincia de origen y destino, así como la cantidad de cueros y pieles, la cantidad de lana sucia, la cantidad de lana limpia, la cantidad de vellón de lana sucia, la cantidad de lana sucia que no corresponde a vellón, la cantidad de pelos y cerdas, la cantidad de estiércol, la cantidad de guano aviar, la cantidad de guano, la cantidad de cama de pollo, la cantidad de aves muertas, la cantidad de pasto, la cantidad de huesos y despojos de faena, la cantidad de trofeos de caza y la cantidad de astas por volteo de ciervos, todos ellos expresados en números enteros. De este conjunto de datos, nos centramos en la cantidad de lana que se envió desde Chubut hacia otras provincias, ya que asumimos que esta fue la producción de lana de Chubut durante ese período.

El portal «Datos Argentina», también fue una fuente valiosa de información para obtener datos sobre la producción ovina de Chubut. Sin embargo, uno de sus principales problemas es que no se actualiza con la frecuencia necesaria para el desarrollo de este trabajo. En efecto, como fue señalado anteriormente, la mayoría de los datasets disponibles en el portal no han sido actualizados desde el año 2019, lo que implica que los datos no son oportunos y accesibles. Además, es necesario destacar que la página web presenta una importante lentitud al cargar sus diferentes funcionalidades, lo que puede generar una experiencia poco satisfactoria para el usuario. Por otra parte, sería recomendable que los nombres de los dataset incluyeran alguna nomenclatura que permitiera identificarlos de manera más clara y precisa, y que, además, cada uno contará con una descripción que brinde información detallada sobre su contenido. Asimismo, se observa que algunos datasets carecen de información acerca de las columnas y los tipos de datos que contienen, lo que puede dificultar su uso y análisis. Sin embargo, tenemos que destacar que el portal cumple con el principio de que su formato es accesible para todos, ya que la mayoría de los datasets están en formato CSV y se pueden descargar fácilmente. Otro aspecto positivo es que el portal permite consultar la fuente de origen de los datasets; donde se observó que la mayoría de los datasets disponibles provienen de SENASA o del MAGyP. En conclusión, el portal Datos Argentina es una buena fuente de información para obtener datos sobre la producción ovina de Chubut, pero es necesario tener en cuenta sus limitaciones en cuanto a la frecuencia de actualización y accesibilidad de los datos, así como la falta de información detallada en algunos datasets.

Sistema de Información Patagonia Sur

El SIPAS [138] es un sitio web desarrollado por el Instituto Nacional de Tecnología Agropecuaria que ofrece una gran cantidad de información valiosa para los productores agropecuarios. El sitio proporciona acceso a estadísticas y censos, leyes y reglamentaciones, documentos sobre políticas públicas e institucionales, tesis de posgrados, eventos naturales y enlaces de interés. Además, el sitio cuenta con una amplia variedad de informes técnicos e información sobre diferentes temas agropecuarios, incluyendo ovinos, bovinos, porcinos, recursos naturales, agricultura, fruticultura, horticultura, forestales, apicultura y desarrollo rural. La sección de meteorología del SIPAS permite obtener datos actuales, pronósticos e informes especiales sobre el tiempo y el clima, incluyendo alertas de incendios, plagas y heladas para diferentes áreas de la Patagonia. También ofrece información sobre economía y

mercado, como el margen bruto de los sistemas productivos y la evolución de los precios para la carne ovina, bovina, porcina y la lana. Asimismo, ofrece acceso a la cartografía digital de la Patagonia, un sistema integrado de información agropecuaria, el seguimiento satelital de la producción agropecuaria y el SIG 250 del Instituto Geográfico Nacional. El SIPAS también incluye una sección de monitoreo productivo, que describe la metodología y evolución del IVNr para la Patagonia Sur, así como informes técnicos sobre el EEA Chubut, EEA Esquel y EEA Santa Cruz. Además, cuenta con una red de información agropecuaria, una sección de maras para el monitoreo de zonas áridas y semiáridas, novedades y una videoteca de videos institucionales y técnicos sobre producción animal, recursos naturales, agricultura y desarrollo rural.

Nos enfocamos en la sección de agrometeorología del sitio SIPAS y encontramos que la lista de estaciones meteorológicas en la Patagonia era información útil para nuestra investigación. La gran mayoría de estas estaciones se encuentran en la provincia de Chubut y nos permitieron obtener un registro histórico detallado del clima. Cada estación proporciona información sobre su ubicación, la propiedad, así como estadísticas sobre las temperaturas mínimas, medias y máximas mensuales, la humedad, la frecuencia de heladas, el promedio de lluvia y la evapotranspiración. También se puede encontrar información sobre la evolución del punto de rocío y condiciones apropiadas para la aplicación de agroquímicos, entre otros datos.

Inicialmente, fue posible la descarga y obtención de los registros históricos de cada estación meteorológica en formato Excel, pero esta funcionalidad dejó de estar disponible en abril de 2021. A pesar de comunicarnos de manera directa con gente responsable en INTA, no obtuvimos respuestas que especificaran si los datos volverían a estar disponibles en algún momento. Luego de esto los datos no volvieron a estar disponibles, por lo que tuvimos que buscar otra forma de recuperar los datos históricos. Utilizamos web scrapers para acceder a las estadísticas de cada sonda, que tienen boletines meteorológicos divididos en años y meses con las medias. Sin embargo, para la aplicación, necesitábamos una API que nos proporcionará datos climáticos. Originalmente, planeábamos utilizar los datos históricos y actuales del SIPAS para crear una API personalizada para nuestra aplicación, pero debido a que el recurso dejó de ser accesible a través de la página web, tuvimos que buscar otra alternativa.

La página web del SIPAS no es un portal de datos abiertos y tiene varias limitaciones que la hacen poco amigable para el usuario. La carga de información es lenta y el acceso a los datos meteorológicos se encuentra en un formato propietario de Microsoft, que es Excel, por lo que no cumple con el principio de datos abiertos. Además, en la actualidad, la página web está descontinuada y no ofrece soporte para situaciones de este tipo, por lo que no se puede acceder a los datos que contiene. A pesar de estas limitaciones, la página resultó muy útil para recuperar los datos históricos del clima de Chubut, los cuales no fue posible encontrar en otros portales, incluso en aquellos que ofrecen servicios de pago. En resumen, aunque en última instancia debimos recurrir a la utilización de web scrapers para obtener la información, la página del SIPAS fue una herramienta valiosa para nuestro trabajo.

Fundación Barrera Zoofitosanitaria Patagónica

La Fundación Barrera Zoofitosanitaria Patagónica (Funbapa) se creó en 1992 para atender las necesidades del sector agropecuario de la región en materia de salud y calidad, a través de la implementación de programas técnicos, políticos y financieramente sostenibles. Su página web [139] ofrece diversas secciones, en su sección de inicio se pueden encontrar noticias, tweets y videos recientes, y enlaces a datos diarios de la red agrometeorológica regional y al mercado regional de hacienda bovina para precios de remates en la Patagonia Norte. También posee una sección institucional para conocer la organización, las autoridades y el equipo técnico, y una sección de programas y actividades que incluye programas de control y erradicación de plagas, certificación de origen de cebollas frescas, banco de vacunas contra la fiebre aftosa, entre otros. La sección de servicios e información incluye estadísticas de ingresos y egresos de productos, precios de mercado de hacienda regional, costos de producción de hortalizas, publicaciones institucionales, y una biblioteca que proporciona información sobre turismo y actividades productivas en la zona. Es importante señalar que varias secciones del contenido se encuentran desactualizadas y no reciben actualizaciones periódicas. De todas las secciones y recursos que ofrece la página web, solo nos enfocamos en las estadísticas de ingreso y egreso de productos buscando información sobre carne ovina. Sin embargo, al revisar los documentos en formato PDF encontramos tablas que detallan los egresos de hacienda y sus productos, divididos por mes, provincia y destino, pero no especifican con claridad cuáles son las carnes de origen ovino. A pesar de que estos datos podrían trabajarse, no encontramos la información que necesitábamos. Además, solo se dispone de información desde el año 2001 hasta el 2014, lo cual es poca información para nuestra investigación. Aunque la página no fue útil para nuestro objetivo, cabe destacar que conocer la cantidad de ingresos y egresos de carne en la región patagónica es valioso. No obstante, para investigaciones específicas, resulta necesario contar con información clara y actualizada, que permita obtener datos precisos.

Federación Lanera Argentina

En Buenos Aires, el 10 de diciembre de 1929, un grupo de comerciantes y representantes de firmas comerciales laneras decidió fundar la Federación de la Lana Argentina. El Poder Ejecutivo Nacional le otorgó su personería jurídica el 18 de septiembre de 1930. En su sitio web [140] se pueden encontrar las últimas noticias, y los estándares nacionales de bienestar animal ovino, entre otras cosas, y se observan actualizaciones relativamente periódicas con fechas que van desde 2022 hasta 2023. Contiene una sección de procesos, donde se encuentran blogs que explican los procesos industriales, la clasificación y el lavado de la lana, y la producción lanera, incluyendo la esquila de la lana. Además, cuentan con una sección de publicaciones que incluye planillas, circulares, estadísticas anuales, estadísticas mensuales, información cambiaria y bursátil, boletines, mercado australiano y mercado mohair. De todas estas secciones, sólo se puede acceder a las estadísticas anuales y mensuales sin un usuario y contraseña. Las estadísticas anuales, que van desde 1998 hasta 2021, se presentan en formato PDF con secciones en inglés y español. Estos informes contienen tablas, resúmenes y gráficos que explican las estadísticas anuales sobre la lana en Argentina. Encontramos que la tabla que muestra la población y producción de ovinos por provincia era

de gran utilidad para nosotros, ya que incluía información detallada sobre la producción anterior, la actual y una predicción para la próxima zafra. En esta tabla, se divide por provincias la cantidad de ovinos y la producción en toneladas de lana fina, mediana y gruesa, así como los totales del país. En cada PDF nos enfocamos en la información de la producción actual de cada tabla para obtener datos precisos sobre la industria lanera en Argentina.

Esta página web no es un portal de datos abiertos, y por ende no cumple con el principio de formato abierto, ya que los datos se presentan en tablas dentro de archivos PDF. En nuestro caso, necesitábamos obtener información sobre la población y producción ovina de la provincia de Chubut para la zafra actual, y como no fue posible hacer scrapers debido al formato de los datos, debimos realizar una búsqueda y obtención de información manual dentro de los 24 archivos disponibles, que cubrían el período desde 1998 hasta 2021, y encontrar las tabla de población y producción ovina por provincias. Aunque esta tabla mejoró con el tiempo, los primeros reportes eran muy confusos debido a la información desorganizada, afortunadamente, la información ahora está más ordenada, aunque aún está en formato PDF. Entonces, para poder crear un dataset que contenga toda la producción de lana de Chubut desde 1998 hasta 2021, debimos seleccionar la información correspondiente a la zafra actual y almacenarla en un archivo en formato CSV. Aunque esta página no es lenta en comparación con otras mencionadas anteriormente, sería beneficioso si pudieran proporcionar su información en formatos abiertos y mejorar así, el aprovechamiento de esa información.

4.2 Herramientas esenciales

Para crear el dataset y los modelos de Machine Learning, utilizamos *Python* y varias de sus bibliotecas, como *Scikit-learn*. Esta última se construye sobre las bibliotecas científicas *NumPy* y *SciPy* de Python. Asimismo, empleamos *Pandas*, *Matplotlib* y *Seaborn*, herramientas que nos permitieron manipular los datos y visualizarlos de manera más clara. Además, utilizamos *Jupyter Notebook*, un entorno de programación interactivo que funciona a través del navegador web. Para llevar a cabo los scrapers, usamos la biblioteca de *Selenium* en combinación con *Pandas*.

¿Por qué Python?

Python es ampliamente utilizado en la ciencia de datos debido a su versatilidad y su amplia gama de bibliotecas y herramientas para la carga de datos, visualización, estadística, procesamiento del lenguaje natural, procesamiento de imágenes y mucho más. Esto proporciona a los científicos de datos una gran cantidad de funciones generales y especializadas para trabajar. Además, una de las principales ventajas de utilizar Python es la capacidad de interactuar directamente con el código, utilizando un terminal u otras herramientas como Jupyter Notebook, lo que permite una iteración rápida y una interacción sencilla con los datos. En el ámbito del aprendizaje automático y el análisis de datos, la capacidad de iterar rápidamente es fundamental, ya que los datos son los que dirigen el análisis. Python también es capaz de crear interfaces gráficas de usuario complejas y

servicios web, así como de integrarse con sistemas ya existentes. Esto hace que sea una herramienta valiosa para aquellos que buscan desarrollar soluciones de análisis de datos y aprendizaje automático en un entorno ágil y eficiente [41].

Librerías de Python y otras herramientas

Scikit-learn

Scikit-learn es un proyecto de aprendizaje automático de código abierto que ofrece una gran variedad de algoritmos de última generación y cuenta con una comunidad de usuarios muy activa. Es la biblioteca de Python más destacada para el aprendizaje automático y funciona muy bien con otras herramientas científicas del lenguaje. Al ser un proyecto de código abierto, es libre de usar y distribuir, lo que significa que cualquiera puede obtener fácilmente el código fuente para ver lo que está pasando detrás de escenas, y además, cuenta con una documentación exhaustiva sobre cada algoritmo. Para utilizar scikit-learn, es necesario instalar primero NumPy y SciPy, y para el trazado y el desarrollo interactivo, también se debe instalar matplotlib, *IPython* y Jupyter Notebook. Por lo tanto, lo mejor es utilizar distribuciones de Python preempaquetadas, que proporcionan todos los paquetes necesarios [41].

Anaconda

Anaconda fue la solución preempaquetada de Python elegida para esta tesina, ya que es una distribución de Python creada para el procesamiento de datos a gran escala, el análisis predictivo y la computación científica. Anaconda viene con NumPy, SciPy, Matplotlib, Pandas, *IPython*, Jupyter Notebook y Scikit-learn. La disponibilidad de estas bibliotecas en Anaconda ayuda a acelerar el proceso de análisis y a simplificar el desarrollo de aplicaciones y algoritmos de aprendizaje automático. Es una solución muy conveniente y útil para eliminar los problemas de dependencia de paquetes y control de versiones, sin una instalación existente de los paquetes científicos de Python [41].

Jupyter Notebook

Jupyter Notebook es una herramienta de software libre que permite la creación de documentos interactivos que combinan texto, código ejecutable, gráficos y otros elementos multimedia; es compatible con varios lenguajes de programación, lo que lo hace una herramienta muy versátil y útil para cualquier proyecto que implique programación. Estos documentos se denominan «cuadernos» y se pueden compartir fácilmente con otros usuarios, lo que los convierte en una herramienta muy útil para la colaboración en proyectos de ciencia de datos. En un cuaderno Jupyter, cada celda puede contener ya sea texto en formato Markdown o código ejecutable, esto permite que los usuarios puedan explicar y documentar su trabajo a medida que avanzan en el proceso de análisis de datos. Además, el Jupyter Notebook proporciona un ambiente interactivo para la ejecución del código, lo que permite a los usuarios ver los resultados de su código en tiempo real. También se integra fácilmente con otras herramientas populares de ciencia de datos como NumPy, Pandas, Matplotlib y Scikit-learn [41].

NumPy

NumPy es un paquete crucial para la computación científica en Python, ya que ofrece una gran cantidad de funciones para trabajar con matrices multidimensionales, así como para realizar operaciones matemáticas avanzadas, como álgebra lineal; y también proporciona herramientas para la generación de números pseudoaleatorios. Para poder trabajar con scikit-learn, es necesario conectar cualquier tipo de datos que se quiera utilizar a una matriz NumPy, ya que ésta es la estructura de datos principal y los datos deben estar en esa forma. Por lo tanto, la clase `ndarray` es la pieza central de la funcionalidad de NumPy, la cual representa una matriz multidimensional, que es similar a la lista normal de Python, pero puede almacenar y operar con datos de manera mucho más eficiente y es importante tener en cuenta que todos los elementos de la matriz deben ser del mismo tipo de datos para su correcta manipulación [41].

SciPy

SciPy es una biblioteca esencial para la computación científica en Python, ya que proporciona una amplia variedad de funciones para tareas avanzadas. Con ella, se pueden realizar operaciones de álgebra lineal, optimización de funciones, procesamiento de señales, funciones matemáticas especiales, distribuciones estadísticas, entre otras cosas. En scikit-learn, SciPy es una biblioteca integral y versátil que es esencial para la ciencia de datos y el aprendizaje automático en Python [41].

Matplotlib

La visualización de datos es una parte esencial del análisis de datos, ya que permite comprender la información de manera más intuitiva. En este sentido, Matplotlib es una biblioteca imprescindible para la visualización de datos en Python, ya que proporciona una amplia gama de opciones para crear gráficos de alta calidad, incluyendo gráficos de líneas, histogramas, diagramas de dispersión, entre otros. Matplotlib es muy útil para crear visualizaciones personalizadas y adaptadas a las necesidades específicas de cada proyecto, en particular, al trabajar en el entorno interactivo de Jupyter Notebook, podemos mostrar figuras directamente en el navegador usando los comandos `%matplotlib notebook` y `%matplotlib inline`. El primero proporciona una experiencia interactiva que permite explorar los datos con herramientas de navegación como zoom, panorámica y selección de áreas, mientras que el segundo se utiliza para mostrar gráficos estáticos en el notebook [41].

Seaborn

Seaborn es una biblioteca de visualización de datos para Python que se basa en matplotlib y proporciona una interfaz de alto nivel para crear gráficos informativos y atractivos, incluyendo funcionalidades como gráficos de distribución, gráficos de regresión y gráficos de matriz, así como temas predefinidos para mejorar la apariencia de los gráficos y facilitar su personalización. En general, Seaborn se utiliza para crear visualizaciones de datos más complejas y avanzadas que las que se pueden crear con Matplotlib [141].

Pandas

Pandas es una biblioteca de Python especializada en el manejo y análisis de datos. Su estructura de datos principal, el DataFrame, es similar a una tabla de una hoja de cálculo de Excel, lo que permite trabajar con datos de manera intuitiva y efectiva. Pandas ofrece una gran variedad de métodos para modificar y operar sobre los datos en un DataFrame, incluyendo la capacidad para realizar consultas y uniones de tablas similares a SQL. A diferencia de NumPy, que requiere que todos los elementos de una matriz sean del mismo tipo, Pandas permite que cada columna de un DataFrame tenga un tipo de dato distinto, como enteros, fechas, números en coma flotante y cadenas. Además, Pandas es capaz de trabajar con datos provenientes de una gran variedad de formatos de archivo y bases de datos, incluyendo SQL, archivos de Excel y archivos de valores separados por comas (CSV). Esto lo convierte en una herramienta valiosa para la limpieza y preparación de datos [41].

Selenium

Selenium es una herramienta de automatización de pruebas para aplicaciones web. Permite simular interacciones de un usuario con una página web, como hacer clic en botones, rellenar formularios y navegar por diferentes páginas. Además, también puede ser utilizado para extraer información de una página web, y es compatible con múltiples navegadores web, como Chrome, Firefox y Edge. Es una herramienta muy útil para la automatización de tareas repetitivas en la web, como la extracción de datos o la realización de pruebas de regresión y es compatible con múltiples lenguajes de programación, incluido Python [142].

4.3 Crear dataset de entrenamiento

Una de las primeras acciones realizadas al comenzar este proyecto, fue la obtención del acceso a un pequeño conjunto de datos sobre la producción de un campo dedicado a la ganadería ovina llamado «El Mirabueno», ubicado en el departamento de Mártires. Este conjunto de datos contenía información sobre la cantidad de animales esquilados, cantidad de ovejas, cantidad de corderos, cantidad de kilos de lana esquilados, el rinde y la finura de la lana, la cantidad de kilos de lana esquilados por animal y el porcentaje de pariciones desde el año 1985 hasta 2020. Con esta información, nos propusimos expandir este conjunto de datos con información de otros campos de la región, pero a pesar de indagar en la fuente de los mismos no fue posible obtener más datos de este tipo. Por consiguiente, decidimos cambiar el enfoque y comenzar a recopilar datos públicos de internet sobre ganadería ovina en Chubut, para, de esta manera, confeccionar un conjunto de datos más grande que se asemejara a los datos de El Mirabueno, pero esta vez a nivel departamental.

En esta sección, se describirá el proceso llevado a cabo para preparar los datos necesarios para entrenar los modelos de inteligencia artificial de la aplicación desarrollada, considerando a la preparación de los datos como una gran tarea que implica seguir una serie de pasos que comienzan con la recopilación, seguido de procesos de limpieza y depuración, y por último validación y visualización.

4.3.1 ¿Por qué es importante la preparación de datos para el ML?

La preparación de datos es una etapa crucial y consume aproximadamente el 80% del tiempo en un proyecto de ML. Esta tarea es especialmente importante en la actualidad, donde los datos fluyen en todas partes y se utilizan en una amplia variedad de aplicaciones. La preparación de datos consiste en recopilar, limpiar y etiquetar datos en bruto para prepararlos para el procesamiento y análisis posteriores. Los datos pueden provenir de diversas fuentes y formatos, incluidos datos estructurados y no estructurados, como imágenes y documentos. Es esencial prestar atención a la calidad de los datos y asegurarse de que se preparen adecuadamente, ya que el ML se basa en datos precisos y completos para obtener resultados confiables, dado que los datos incorrectos, incompletos o sesgados pueden afectar negativamente a la precisión de los modelos y sus predicciones. Aunque la preparación de datos puede ser un proceso tedioso y que requiere mucho tiempo, existen herramientas especializadas que ayudan a automatizar y acelerar el proceso; al invertir tiempo y recursos en la preparación de datos, se pueden obtener resultados más precisos y confiables en los proyectos de ML, lo que puede tener un impacto significativo en su éxito [143].

Durante la preparación de los datos para el aprendizaje automático, resulta fundamental comprender tanto la tarea que se quiere resolver como los datos con los que se trabaja. De hecho, esta comprensión constituye probablemente la parte más crítica del proceso de aprendizaje automático. No basta con seleccionar al azar un algoritmo y aplicarlo a los datos, sino que es necesario entender en profundidad lo que ocurre en el conjunto de datos antes de construir cualquier modelo. Cada algoritmo es diferente en cuanto al tipo de datos y al entorno del problema para el que funciona mejor. Para lograr una solución de aprendizaje automático efectiva, es preciso responder, o al menos tener en mente, ciertas preguntas clave. Por ejemplo: ¿Qué pregunta o preguntas se intenta responder? ¿Son los datos recopilados suficientes para responder a esa pregunta? ¿Cómo se puede plantear la(s) pregunta(s) como un problema de aprendizaje automático? ¿Qué características de los datos permitirán hacer las predicciones correctas? ¿Cómo se medirá el éxito de la solución? ¿Cómo interactuará la solución de aprendizaje automático con otras partes de la investigación?. No obstante, en el contexto más amplio, los algoritmos y métodos de aprendizaje automático son solo una parte de un proceso mayor para resolver un problema concreto. Es importante tener siempre presente el panorama general, para evitar el riesgo de perder de vista los objetivos finales al profundizar en los aspectos técnicos del aprendizaje automático. También es esencial tener en cuenta las suposiciones que se pueden estar haciendo, tanto explícitas como implícitas, al construir modelos de aprendizaje automático [41].

4.3.2 Recolección de datos

El proceso de recolección de datos es fundamental en el aprendizaje automático, ya que implica obtener información de diversas fuentes, como portales, bases de datos, la nube, aplicaciones y dispositivos. A pesar de su importancia, este proceso puede ser desafiante y tedioso debido a que los datos pueden estar en diferentes formatos y tipos. Además, el aumento constante del volumen de datos también presenta un desafío significativo para su

integración. Sin embargo, es crucial para el éxito del aprendizaje automático y es necesario abordar de manera efectiva para obtener información valiosa [143].

Datos meteorológicos

Para recolectar datos meteorológicos, utilizamos la información disponible en la página del SIPAS, anteriormente, fue posible acceder a estos datos descargando archivos en formato Excel, pero dicha funcionalidad actualmente no cuenta con mantenimiento. Actualmente, existen dos opciones para acceder a los datos, ambas ubicadas en la sección de boletines meteorológicos: la primera permite acceder a los datos diarios de las sondas climáticas en distintos meses y años, mientras que la segunda opción muestra un resumen anual para cada año disponible, incluyendo la media mensual para cada mes. Es sobre este último conjunto de información sobre el que ejecutamos scrapers, ya que la información es idéntica a la que se encuentra en la sección por mes, pero totalizada y promediada. Ambas opciones proporcionan información detallada sobre la temperatura máxima, mínima y media, la cantidad de lluvia, la velocidad del viento, la humedad, la dirección y velocidad máxima del viento, la radiación solar, la evapotranspiración y la presión atmosférica en unidades de hectopascales (hPa).

Para combinar los datos de las 60 sondas pertenecientes a las estaciones meteorológicas del INTA, se encontró un primer obstáculo: cada sonda tenía un ID que no necesariamente se correspondía con el número de sondas disponibles, lo que resultó en algunas sondas con números de ID mayores al total de 60 sondas. Para abordar este problema, se determinó cuál era el número de ID más alto y luego se recorrió un rango de números que avanzaba del 1 al 72 para recopilar los datos de cada una de las sondas disponibles. Se encontró que la página web que contenía los datos meteorológicos tenía una URL diferente para cada tipo de dato que se necesitaba obtener, como la temperatura mínima, media, máxima, humedad y lluvias. Para recopilar los datos se configuró el driver de Selenium como un driver de Chrome que utilizaba el servicio ChromeDriverManager y se lo estableció en el modo «headless», es decir, sin interfaz gráfica, para mejorar el rendimiento y reducir el consumo de recursos, dado que la ejecución en modo gráfico consumía muchos más recursos. También se estableció el tamaño de la ventana del navegador, que se abriría cuando se usará el controlador de Chrome, esto nos fue útil para simular una ventana del navegador con un tamaño específico al ejecutar Selenium en segundo plano (headless), lo que nos permitió obtener resultados más precisos. Por último, se añadió un manejador para realizar un registro de las acciones relevantes del proceso del scraper.

Posteriormente, se implementó una función encargada de recorrer las 60 sondas meteorológicas mediante un rango de números que iba desde el 1 al 72, ya que como mencionamos antes, algunas sondas tenían un ID superior al número de sondas existentes. Con el driver configurado y la función «obtener_datos» implementada, se accedía a cada una de las URLs correspondientes a cada tipo de datos, cambiando el ID de la estación meteorológica. Dado que la página web era lenta, se incorporó una pausa de 120 segundos para esperar a que se cargara completamente, antes de acceder a los nodos de HTML y extraer los elementos de interés, como los valores de temperatura, lluvia, humedad y año. También se registró la información adicional de cada estación meteorológica, como el

nombre, la ubicación y el propietario, y se guardó toda esta información en un DataFrame. Por último, se exportó el DataFrame a un archivo CSV para ser utilizado posteriormente en el análisis de datos.

Después de completar el proceso de extracción de datos, se generaron cinco conjuntos de información correspondientes a la temperatura máxima, temperatura media, temperatura mínima, humedad y lluvia, los cuales se guardaron en archivos CSV. Cada uno de estos archivos incluía información sobre la estación meteorológica, el año, los promedios mensuales y el promedio anual, con un total de 613 filas. Para obtener más detalles sobre el proceso de recolección de datos, consulte el Anexo I (Sección 1.1). Es importante destacar que durante la revisión de los registros se identificaron valores faltantes y espacios en blanco, los cuales serían corregidos en una etapa posterior.

Datos sobre existencias ovinas

Para recolectar los datos sobre las existencias ovinas de la provincia de Chubut, se emplearon algunos de los conjuntos de datos abiertos encontrados en el portal «Datos Argentina» y otras fuentes. En cada uno de los siguientes conjuntos de datos, se realizó una investigación preliminar para determinar su contenido y su utilidad para nuestro análisis. Si se consideró que los datos eran relevantes, se procedió a filtrarlos para incluir solo la información correspondiente a la región de interés.

Indicadores de ovinos

El primer dataset que nos fue útil, fue el de indicadores de ovinos [134]. Este dataset es muy completo y contiene información sobre las existencias ovinas, la producción de lana y la producción de carne, recopilados de distintas fuentes como el MAGyP, la FAO, FLA y SENASA. Los indicadores se presentan en formato anual, trimestral y mensual, y el punto negativo es que no había una descripción de qué valor representaba cada columna. Después de cargar los datos en un DataFrame para examinar su contenido, descubrimos que se trataba de un conjunto de datos un tanto complejo que requería un análisis minucioso para encontrar los datos que realmente nos interesaban. El dataset contenía 8.055 filas y 14 columnas. En un primer análisis, la columna que parecía contener la información de interés era la de «actividad_producto_nombre», por lo que revisamos los strings únicos en esa columna para identificar los productos que aparecían en el dataset, los cuales resultaron ser nueve. Como primera medida, buscamos información sobre la producción de carne en la provincia, pero no fue posible encontrar datos desglosados por provincia o departamento; por lo tanto, no pudimos utilizar esta información para nuestro análisis. Luego, examinamos la columna «ovinos» del dataset, que asumimos que son las existencias ovinas y descubrimos que había datos a nivel departamental, lo que nos permitió identificar los 16 departamentos de Chubut donde habían datos sobre existencias ovinas. Esta información iba a ser útil en nuestro análisis y por ende la guardamos en un archivo CSV. En síntesis, aunque no logramos obtener información específica acerca de la producción de carne en la provincia de Chubut, sí pudimos recopilar datos sobre las existencias ovinas a través de fuentes de datos abiertos

disponibles. Se incluye el detalle completo de este análisis en el Anexo I (Sección 1.2) para su revisión.

Existencias ovinas 1965-2017

También analizamos el dataset de existencias ovinas 1965 a 2017 [131], obtenido del portal de datos públicos «Datos Argentina». Este conjunto de datos contiene información anual sobre la cantidad de ovejas en el país, con detalles por provincia. Aunque cuenta con una pequeña documentación, fue necesario explorarlo para comprender su contenido. Una vez convertido el archivo CSV en un DataFrame, verificamos que tenía 1.137 filas y ocho columnas: país, provincia, desde qué año, hasta qué año, la cantidad y la fuente. Al buscar los datos de la provincia de Chubut, encontramos 49 filas desde 1965 hasta 2017 con las existencias ovinas en la región, las cuales fueron guardadas en un archivo CSV. Además, la columna «fuente» nos dio una pista sobre otros lugares donde podríamos buscar información adicional sobre ovinos en el país. En particular, nos encontramos con «Estimación Dto. Ovinos y Lanas - SAGPyA matemática con datos existentes», «Censos INDEC», «Encuesta Nacional Agropecuaria 93- 94- 95 y 01- SAPGyA e INDEC», «Estimación Dto. Ovinos y Lanas - SAGPyA en base a datos objetivos» y «SIGSA - Dirección de Control de Gestión y Programas Especiales - Dirección Nacional de Sanidad Animal - SENASA». Al buscar en Google «Estimación Dto. Ovinos y Lanas - SAGPyA matemática con datos existentes» o «Estimación Dto. Ovinos y Lanas - SAGPyA en base a datos objetivos», se llegó a la sección de ovinos de la página del MAGyP y a la sección de Datos MAGyP; y ambos son la fuente de origen para los datos que se encuentran en Datos Argentina, por lo tanto, ya los habíamos consultado. En cambio, al buscar «Censos INDEC» o «Encuesta Nacional Agropecuaria 93-94-95 y 01- SAPGyA e INDEC», se accedió a diferentes secciones de la página del INDEC; sin embargo, sólo se encontró información general sobre la cantidad de ovejas que se exportan y consumen en el país, sin detalles por provincia. Por último, al investigar «SIGSA - Dirección de Control de Gestión y Programas Especiales» y «Dirección Nacional de Sanidad Animal - SENASA», no se encontró información relevante para nuestro análisis. Para obtener más detalles sobre la recolección de datos de este dataset, consulte el Anexo I (Sección 1.3).

Existencias ovinas 2008-2019

Este conjunto de datos [135] fue obtenido del portal de datos públicos «Datos Argentina» y contiene información sobre las existencias ovinas del país desde 2008 hasta 2019, categorizado por carneros, ovejas, borregos, capones, corderos y corderas. Aunque se proporcionó una breve descripción de las columnas en la sección anterior, fue necesario verificar si los datos eran relevantes para nuestro análisis. Lo siguiente fue cargar el archivo CSV en un DataFrame para obtener más información y descubrimos que el conjunto de datos tenía 5.795 filas y 13 columnas, que incluían información sobre el número de ovinos en cada provincia y departamento. Para determinar si el conjunto de datos contenía información sobre Chubut, buscamos qué provincias estaban presentes en el conjunto de datos y encontramos que Chubut estaba incluido. Luego, filtramos todas las filas correspondientes a Chubut para obtener un conjunto de datos de 180 filas y 8 columnas, ya que eliminamos las columnas de identificación de provincia y departamento. Finalmente, ordenamos este conjunto de datos

por año y departamento y lo almacenamos en un archivo CSV. El Anexo I (Sección 1.4) proporciona detalles completos sobre todo lo mencionado anteriormente.

Existencias ovinas 2008-2020

Este conjunto de datos [136] fue obtenido de la página de SENASA y contiene información sobre las existencias ovinas en el país. Los datos son totales anuales desde marzo de 2008 hasta la última actualización en marzo de 2020. Aunque no se dispone de documentación detallada sobre el conjunto de datos, se encontraron cuatro archivos en formato xlsx: uno de ellos las existencias ovinas por categoría y departamento del 2008 al 2017, y los otros tres contienen las existencias ovinas por departamento a marzo de 2018, 2019 y 2020, respectivamente. Para facilitar su análisis, los archivos xlsx fueron convertidos a formato CSV; además, se desglosó el archivo que contiene datos del 2008 al 2017 en archivos CSV anuales, obteniendo así un total de doce archivos. A pesar de que este dataset tiene las mismas columnas que el anterior, es decir, carneros, ovejas, borregos, capones, corderos y corderas, hay una diferencia clave: en este caso, la información de la provincia y el departamento está contenida en la misma columna. La razón de esto es que la información se encuentra totalizada, es decir, se presentan los datos de cada departamento dentro de una misma provincia para luego hacer un total de todos los departamentos en esa misma provincia. Este dataset tuvo que pasar por un proceso de limpieza, que se detallará más adelante.

Datos sobre lana ovina

Para recolectar los datos sobre la producción de lana ovina de la provincia de Chubut, se emplearon algunos de los conjuntos de datos abiertos encontrados en el portal «Datos Argentina» y también se utilizaron otras fuentes. En cada uno de los siguientes conjuntos de datos, se realizó una investigación preliminar para determinar su contenido y su utilidad para nuestro análisis. Si se consideró que los datos eran relevantes, se procedió a filtrarlos para incluir solo la información correspondiente a la región de interés.

Indicadores de ovinos

Este dataset no solo proporciona información sobre las existencias ovinas en Argentina, sino que también incluye datos sobre la producción de lana en formato anual, trimestral y mensual. Entre los productos que se incluyen en el dataset están: lana fina de 20 micrones con un rendimiento de peine del 55%, lana mediana de 27 micrones con un rendimiento de peine del 55%, lana sucia, lana sucia de cruce fino, lana sucia fina, lana sucia gruesa y lana sucia mediana. A continuación, nos propusimos encontrar información específica sobre la producción de estos siete productos en la provincia de Chubut. Al filtrar los datos por producto y provincia, descubrimos que no había información disponible sobre la producción de lana fina de 20 micrones, lana mediana de 27 micrones y lana sucia de cruce fino en la provincia de Chubut. Sin embargo, sí encontramos datos sobre la producción de lana sucia, lana sucia fina, lana sucia mediana y lana sucia gruesa en Chubut. Por lo tanto, guardamos esta información en cuatro archivos CSV. Se incluye el detalle completo de este análisis en el Anexo I (Sección 1.2) para su revisión.

Movimiento de ovinos

El siguiente dataset analizado fue el de movimiento de ovinos [137] obtenido del portal de datos abiertos «Datos Argentina». A diferencia del dataset anterior, este contaba con una descripción que nos proporcionó información general. No obstante, fue necesario examinar su contenido para determinar su utilidad. El dataset constaba de 8.257 filas y 23 columnas, algunas de las cuales eran la cantidad de cuero, la cantidad de lana sucia, la cantidad de lana limpia, la cantidad de guano aviar, la cantidad de huesos y despojos de faena, entre otros. De todas estas columnas, las que nos resultaron de mayor utilidad, fueron aquellas relacionadas con la provincia y la lana. Con estos datos, lo primero que hicimos fue filtrar los valores por Chubut en las columnas de provincia de origen y de destino, comprobando así que había movimientos de subproductos que involucraban a la provincia. Luego filtramos aquellas filas que se correspondían a la lana, e hicimos la suposición de que las filas con origen en Chubut correspondían a la producción de lana, y nos centramos en los datos de la provincia sobre la cantidad de lana sucia, lana limpia, lana sucia vellón, lana sucia no vellón. Como resultado se obtuvo un nuevo DataFrame con 260 filas y 10 columnas, en este punto todavía manteníamos algunas columnas para guiarnos, pero más adelante en la limpieza se iban a eliminar. Para ver el análisis en detalle, se puede consultar el Anexo I (Sección 1.5).

FLA

A pesar de que no fue posible descargar archivos de datos desde la página web de la Federación Lanera Argentina, creamos un archivo CSV a partir de los archivos PDF que se encontraban allí. En la web de la FLA, se puede encontrar un listado de estadísticas anuales de la producción de lana de Argentina. Cada archivo PDF contenía información sobre existencias, consumo, exportaciones y gráficos estadísticos de la producción de lana en el país. Nos enfocamos en extraer los datos que nos interesaban para la producción de lana en la provincia de Chubut y para la zafra actual del año de cada archivo PDF. El conjunto de datos que creamos consta de 11 columnas: actividad_producto (en este caso, siempre es lana sucia), indicador (en este caso, siempre es producción), unidad de medida (en este caso, siempre es tonelada), frecuencia (siempre anual), alcance tipo (a nivel provincial), alcance nombre (siempre Chubut), índice de tiempo (desde el año 2000 hasta 2022), cantidad de toneladas de lana fina, cantidad de toneladas de lana mediana y el total de lana producida cada año. Por último, tenemos una columna de existencias ovinas que nos da una aproximación de la cantidad de ovinos en la provincia desde 2000 hasta 2022. Para algunos años, la información sobre las existencias ovinas no estaba disponible en la Federación Lanera Argentina, por lo que completamos estos valores con NaN. En total, el conjunto de datos consta de 23 filas.

Datos sobre áreas geográficas

Durante los entrenamientos de los modelos, nos surgió la duda de si incluir la superficie territorial de los departamentos como una variable adicional podría mejorar los resultados. Para explorar esta posibilidad, decidimos obtener el tamaño de cada departamento y convertirlo a hectáreas, ya que es la unidad de medida más comúnmente utilizada para áreas de campo. Para ello, buscamos manualmente la información sobre el área total de cada

departamento en internet y lo transformamos a hectáreas. Este proceso de obtención de datos fue completamente manual.

4.3.3 Limpieza y depuración de datos

La calidad y confiabilidad de los datos son elementos fundamentales en cualquier proyecto de aprendizaje automático. Si los datos que se utilizan no están limpios y no son confiables, pueden existir graves consecuencias, desde el desperdicio de recursos hasta la pérdida de vidas humanas. Como mencionamos anteriormente, se estima que la preparación de datos en un proyecto de aprendizaje automático representa cerca del 80% del tiempo dedicado al proyecto, mientras que el modelado de datos y los resultados obtenidos a través de técnicas de aprendizaje automático solo representan el 20% restante. Por lo tanto, la limpieza de datos es un paso importante en el proceso de preparación de datos, ya que ayuda a corregir errores y a rellenar espacios en caso de datos perdidos, lo que garantiza la calidad de los datos.

Después de la limpieza de datos, es necesario transformarlos a un formato consistente y legible. Este proceso puede incluir la conversión de campos como fechas y divisas, la modificación de convenciones de nomenclatura y la corrección de valores y unidades de medida para que sean consistentes. La limpieza de datos implica procesos de detección, así como la corrección de errores, omisiones e inconsistencias dentro de un conjunto de datos. Este proceso consta de dos procesos independientes: la depuración de datos y el llenado de datos faltantes [40][143].

Depuración de datos

La depuración de datos tiene como finalidad cumplir con diversos objetivos, entre los que se encuentran: identificar datos que sean atípicos, inexactos, incompletos, irrelevantes o corruptos, para posteriormente eliminarlos antes de procesarlos; analizar los datos y extraer información relevante, así como validar si los datos se encuentran en un formato aceptable; y transformar los datos a un formato de escala de tiempo o rango normalizado para facilitar su manejo y análisis. Al realizar una depuración de datos rigurosa, se asegura que los datos utilizados para el análisis y la toma de decisiones sean de alta calidad y confiabilidad.

Llenado de valores perdidos

Los algoritmos del aprendizaje máquina, generalmente, no funcionan bien si se tienen valores faltantes y es muy importante entender por qué falta un valor. Algunas causas pueden ser errores aleatorios, fallas sistemáticas o ruido en el sensor, es decir, que el valor, o valores, puede faltar por muchas razones; por lo tanto, resulta importante entender por qué falta, está ausente o dañado para poder llenarlo correctamente. Una vez que se identifica la causa, existen diversas formas de lidiar con los valores faltantes, tales como:

- ❖ Eliminar la observación: si se dispone de una cantidad suficiente de datos y solo un pequeño número de observaciones irrelevantes tienen valores faltantes, es recomendable eliminar esas observaciones.

- ❖ Eliminar el atributo: eliminar un atributo puede ser una opción viable cuando la mayoría de los valores están ausentes, los valores son constantes o el atributo está altamente correlacionado con otro atributo.
- ❖ Asignar un valor especial N/D: una forma de manejar valores faltantes es asignarles un valor especial como «N/D» (no disponible) o «N/A» (no aplicable). Esto puede ser apropiado cuando la falta de valor se debe a factores como estar fuera del alcance, la falta de definición de un valor de atributo discreto o la imposibilidad de obtener o medir el valor en cuestión. Por ejemplo, si una persona nunca ha calificado una película, la calificación para esa película puede ser considerada como «N/D». Sin embargo, es importante tener en cuenta que asignar un valor especial puede no ser adecuado en todas las situaciones y se debe considerar cuidadosamente según el contexto y los requisitos del análisis de datos.
- ❖ Tomar el valor de atributo promedio: en caso de que se cuente con un número limitado de observaciones, no resulta viable permitirse eliminar observaciones o atributos. En este caso, se pueden estimar los valores perdidos; por ejemplo, asignando el valor de atributo promedio o el valor promedio sobre observaciones similares.
- ❖ Predecir el valor de otros atributos: se puede predecir el valor anterior de los datos de entrada si el atributo posee dependencias de tiempo.

En cuanto a los datos atípicos, son aquellos que representan valores extremos presentes en ciertas observaciones de los datos. En algunos casos, es posible eliminar las observaciones con valores atípicos si se cuenta con suficientes observaciones; sin embargo, cuando las observaciones son limitadas, es necesario transformar los datos atípicos para evitar reducir el conjunto de observaciones [40].

En resumen, es importante abordar tanto los valores perdidos como los datos atípicos en cualquier proyecto de aprendizaje de máquina para garantizar la calidad de los resultados.

Datos meteorológicos

Después de recolectar datos del sitio web del SIPAS, obtuvimos cinco archivos CSV dado que ya se habían depurado a un formato adecuado. En cada uno de ellos, eliminamos todas las observaciones que no contenían datos, lo que significa que no se habían recolectado datos para ningún mes del año en esa sonda y las filas de cierta manera estaban vacías. Como resultado, el número de filas en los archivos CSV disminuyó de 613 a un número menor. Por ejemplo, para la humedad quedaron 363 filas, para la lluvia quedaron 513 filas, para la temperatura media quedaron 522 filas, para la temperatura máxima quedaron 520 filas y para la temperatura mínima quedaron 522 filas.

Después de eliminar las observaciones vacías, se planteó la siguiente interrogante: ¿estos datos son aptos para su uso? La respuesta fue «no», ya que se requerían datos climáticos por departamento, mientras que los datos recolectados eran por estación meteorológica, razón por la cual era necesario completar este dato faltante. Para determinar el departamento de Chubut al que pertenecía cada estación meteorológica, era necesario primero buscar esta información. Para hacerlo, se debía ubicar la latitud y longitud de la estación meteorológica

proporcionadas por la página del SIPAS y determinar en qué departamento de Chubut caía la sonda según su ubicación geográfica y asociar el nombre del departamento con cada estación. De esta manera, se eliminaron las estaciones que no pertenecían a Chubut, recordando que también había sondas de Santa Cruz disponibles.

Una vez determinado el departamento al que pertenecía cada estación meteorológica, era necesario añadir esta información en los archivos CSV correspondientes. Para ello, se agregó un nuevo atributo denominado «departamento» y se completó con el nombre del departamento correspondiente a cada sonda. Además, se procedió a ordenar el dataset por nombre de departamentos, seguido del nombre de las sondas y finalmente, por año en orden ascendente.

Posteriormente, se recuperaron los cinco archivos CSV actualizados con la información del departamento, y se cargaron a través de la librería Pandas. Luego, se convirtió cada CSV en un DataFrame, y se agrupó la información de cada dataset por departamento y año. De esta forma, se generaron cinco DataFrames, cada uno con los promedios totales de los datos para los departamentos de Chubut agrupados por año. A continuación, se juntaron estos cinco DataFrames para crear el conjunto de datos del clima por departamento en Chubut. Se creó una lista con los cinco DataFrames y se recorrió con una función reduce de Python, lo que generó un DataFrame agrupado por departamento y año, con cinco columnas que contenían los promedios de la humedad, la lluvia, la temperatura máxima, la temperatura mínima y la temperatura media. Posteriormente, se reiniciaron los índices y se renombraron las columnas, y se convirtió el año en tipo de fecha para facilitar su unión con el dataset de producción. De esta manera, se obtuvo un DataFrame con la columna departamento, la columna año y las cinco columnas mencionadas anteriormente, con un total de 209 filas. Finalmente, se guardó este DataFrame generado en un archivo CSV denominado «clima_chubut». Para obtener más información sobre el proceso de limpieza, se puede consultar el Anexo I (Sección 1.6).

Hay algunos aspectos importantes a tener en cuenta en el procesamiento de los datos meteorológicos de Chubut. En primer lugar, es necesario considerar que existen huecos en los datos, lo que significa que en algunos casos no se tienen registros de todos los meses del año para una determinada estación meteorológica. Por ejemplo, para la Sonda de Trelew en el año 2007, solo se registraron datos en los meses de marzo, agosto y noviembre. Por lo tanto, el promedio anual se calculó como la suma de estos tres valores dividida por tres, aunque esto puede no representar una imagen precisa del clima en ese año. Otro aspecto importante es que, al agrupar los datos por departamento y por año, se utilizó la media de los valores de las diferentes estaciones meteorológicas, por lo tanto, si para un departamento hubo tres sondas que midieron la lluvia en un año determinado, se calculó el promedio de estas tres mediciones para obtener el promedio total de lluvia para ese departamento y año. En algunos casos, solo se contaba con una sonda para un departamento, mientras que en otros no había datos disponibles, como es el caso de Mártires, Escalante y Gastre. Es importante tener en cuenta estas limitaciones al analizar los datos meteorológicos de Chubut y considerar que los resultados obtenidos pueden no reflejar con precisión el clima de la región en todos los casos.

En el caso del campo El Mirabueno, no se contaba con datos climáticos asociados a él. Desafortunadamente, este campo se encuentra dentro del departamento de Mártires, para el cual el SIPAS no tenía sondas que recopilaran datos meteorológicos de esa zona. Por lo tanto,

para completar los datos meteorológicos del campo que abarcaba de 1985 a 2020, se buscó la sonda meteorológica más cercana al campo que tuviera datos para ese año. Por ejemplo, desde 1985 a 2011, se tomaron los datos de la sonda de Trelew y, a partir de entonces, se combinaron con los datos de las sondas de Telsen y Laguna Fría, entre otras. Entonces es importante tener en cuenta que los datos meteorológicos de este campo no son 100% precisos debido a la falta de una sonda climática del SIPAS cercana y la combinación de datos de diferentes sondas.

Datos sobre existencias ovinas

Para que los datos recolectados sobre existencias ovinas fueran útiles era necesario limpiarlos. A continuación veremos el proceso de depuración que se llevó a cabo en dos conjuntos de datos.

Existencias ovinas 2008-2020

Anteriormente, mencionamos que el dataset de SENASA estaba compuesto por cuatro archivos `xlsx`, los cuales fueron convertidos en doce archivos `CSV`, uno para cada año. Antes de empezar el análisis, creamos una lista de las provincias argentinas en mayúsculas, tal como aparecían en el dataset, pero encontramos que los departamentos Formosa y la Ciudad Autónoma de Buenos Aires tenían el mismo nombre que sus respectivas provincias. Por ello, renombramos manualmente en el dataset la Ciudad Autónoma de Buenos Aires a «Capital Federal» y el departamento Formosa a «Formosa_d» para facilitar la búsqueda.

Luego, cargamos cada archivo `CSV` y creamos una función llamada «separar_en_provincias» que separaba el atributo de provincia/departamento en dos columnas distintas: una para la provincia y otra para el departamento. Además, agregamos dos atributos más llamados «establecimientos» y «unidades productivas» para aquellos archivos del 2020 que contenían esas columnas y asignamos valor «NaN» a las que no las tenían. También nos encontramos con datasets con nombres de columnas que contenían espacios que dificultaban la búsqueda, por lo que creamos una lista con los nombres deseados y renombramos las columnas para evitar problemas. Una vez que se separaron los datos por provincia y departamento y se corrigieron los nombres de las columnas, fue sencillo filtrar las filas correspondientes a la provincia de Chubut. Como resultado, se obtuvieron doce archivos `CSV`, uno para cada año, con información detallada sobre las existencias ovinas en la provincia. Finalmente, se combinaron estos archivos en un único `DataFrame`, que se ordenó por año y por departamento, y se guardó en un nuevo archivo `CSV` que contenía los datos de existencias ovinas desde 2008 hasta 2020, categorizados por tipo de ovino (oveja, carnero, borrego, capón, etc.). Para conocer los detalles sobre la construcción de este dataset, consultar el Anexo I (Sección 1.7).

Existencias ovinas 1965-2017

Teniendo en cuenta que nuestro objetivo era obtener las existencias ovinas desde 2000 hasta 2020, se encontró que el dataset de SENASA no contenía los primeros siete años necesarios. Por lo tanto, se decidió extraer las existencias ovinas de un dataset más amplio que abarcaba 48 años y contenía la información que necesitábamos. En primer lugar, se eliminaron los

atributos que no eran necesarios, como provincia, hasta_año y fuente, y se conservaron solo las columnas de cantidad y la de desde_año. La columna desde_año se convirtió en formato fecha y se utilizó como índice para filtrar los datos de 2000 a 2007. Para este dataset, la cantidad se expresaba en miles de cabezas, y era necesario hacer una transformación de datos, por lo que se multiplicó por 1.000 para obtener el número real de cabezas. Por ejemplo, el valor 4.017,0 en este dataset representaba 4.017 millones de cabezas de ganado. Es decir, se multiplicó 4.017 por 1.000 para obtener un total de 4.017.000 cabezas de ganado.

Con el fin de completar las filas faltantes, era necesario obtener el porcentaje de cada tipo de ovinos (corderos, carneros, ovejas, borregos, etc.) que cada departamento tenía en relación al total de ovinos de cada año. Para ello, utilizamos el dataset de existencias de SENASA obtenido previamente para obtener los totales anuales de ovinos. Luego, se generó una lista de objetos que contenía el año y el total de ovinos para ese año. El siguiente paso fue recorrer todos los departamentos de Chubut y, por cada departamento, recorrer los totales obtenidos anteriormente. En base al total de ovinos del departamento, calculamos el porcentaje de cada tipo de ovinos (corderos, carneros, borregos, ovejas, etc.). Luego, en base al total de ovinos de Chubut, calculamos el porcentaje de ovinos que representaba cada tipo de ovinos para el departamento. De esta manera, obtuvimos un dataset intermedio de porcentajes de Chubut.

Para poder generar los valores faltantes de manera aleatoria, pero con cierta estructura, fue necesario obtener las métricas estadísticas de los porcentajes obtenidos anteriormente. Se creó otro dataset intermedio que contenía el departamento, el tipo de ovinos y el total, que incluía la media, el desvío estándar, el mínimo, el máximo y una lista de valores normales aleatorios. Al utilizar valores aleatorios, la suma de los porcentajes generados no daba siempre 100, por lo que fue necesario hacer ajustes. De esta manera, se realizó el ajuste y se obtuvieron las categorías para cada departamento, lo que dio como resultado un dataset de 128 filas y nueve columnas donde las columnas son el año, la provincia, el departamento, carneros, ovejas, borregos, capones, corderos y el total de ovinos desde el año 2000 hasta el año 2007 para la provincia de Chubut. Para una explicación más detallada de este análisis, se puede hacer referencia al Anexo I (Sección 1.13).

Datos sobre lana ovina

Llegado a este punto teníamos tres dataset de producción de lana ovina en Chubut para limpiar. Es importante destacar que la depuración de datos es un proceso iterativo y puede requerir varios ciclos de revisión y corrección para obtener un dataset limpio y confiable, como veremos sobre uno de los dataset en donde se realizaron varias iteraciones producto de sucesos posteriores.

Movimiento de ovinos

Después de obtener la lana proveniente de Chubut, se encontró que existían diversas unidades de medida utilizadas: kilogramos, bultos y unidad. Desafortunadamente, esto requirió elegir una de ellas y realizar la correspondiente conversión para homogeneizar la información. Dado que la medida del peso del vellón es la forma tradicional de valorar el potencial lanero de la oveja, y como no había documentación sobre las unidades de medida para este conjunto de datos, se decidió que un vellón era equivalente a un bulto (como se encontró en la

referencia [144]) y a una unidad, y que tanto el bulto como la unidad se medían en kilogramos. Para generar los pesos de vellón aleatorios basados en la media global, necesitábamos primero conocer la media de peso del vellón de la región. Tras investigar, descubrimos que en la Patagonia las razas más comunes de ovinos son el Merino y el Corriedale, siendo el Merino criado en zonas áridas y secas de Río Negro, Chubut y norte de Santa Cruz para obtener lana fina y super fina de alto valor en el mercado internacional. Por tanto, nos centramos en esta raza, dado que nos interesaba Chubut, donde se encontró que el peso promedio del vellón es de $4,18 \pm 0,054$ kilogramos, con un rango de variación de 4,20 kg (de 6,20 kg a 2,00 kg) y un coeficiente de variación del 20,41%. No obstante, es importante tener en cuenta que este valor puede variar según la edad, el sexo y la ganadería de nacimiento de los animales, y que los animales jóvenes suelen tener un peso de vellón mayor que los adultos, y el peso del vellón varía según el sexo. Asimismo, se debe tener en cuenta que, idealmente, el 2% al 5% de la majada son machos y el resto hembras. Con esta información, pudimos generar los pesos de vellón aleatorios en base a la media global siguiendo una distribución normal, y multiplicarlos por la cantidad de bultos y/o unidades que tenía cada productor en el dataset [145][146][147].

Entonces, se diseñó una función llamada «convertir_a_kilogramos» que cumplía con la tarea de verificar la unidad de medida en cada fila. Si la unidad de medida era «bulto» o «unidad», la función realizaba la conversión correspondiente a kilogramos utilizando datos como el porcentaje de animales macho y hembra, la media del vellón macho-hembra y el coeficiente de variación. Por otro lado, si la fila no contenía estas unidades de medida, simplemente se retorna el valor en kilogramos sin realizar ningún cálculo ni conversión adicional. Después de haber creado la función «convertir_a_kilogramos», procedimos a recorrer el DataFrame y aplicar dicha función para convertir todas las medidas de lana a kilogramos. De esta forma, logramos homogeneizar los datos y continuar con el análisis de manera más precisa. Además, transformamos el atributo «fecha» en un dato de tipo fecha para poder agrupar los datos de manera anual y obtener la cantidad total de lana producida en Chubut en cada año. Para finalizar, se procedió a agrupar los datos por año y eliminar las columnas que ya no eran necesarias. De esta manera, obtuvimos la cantidad anual de lana producida en Chubut desde el año 2013 hasta el 2018. Estos datos fueron guardados en un archivo CSV para su posterior análisis y utilización. Para más información revisar Anexo I (Sección 1.5).

Indicadores de ovinos

Durante el análisis del dataset de indicadores ovinos, se obtuvieron cuatro archivos CSV relacionados con la producción de lana en Chubut. Al revisar el archivo correspondiente a la lana gruesa, se encontró que todos sus valores eran cero, por lo que se descartó como posible dataset para utilizar. Se procedió a analizar los otros tres archivos, y se descubrió que uno de ellos representaba el total de los otros dos. Como parte del proceso de limpieza, se decidió concatenar los tres archivos para obtener una información más clara y permitir una comparación más fácil con otros datasets. De esta forma, se obtuvo un nuevo dataset con 22 filas y 10 columnas, que incluía datos desde el año 2000 hasta el 2022. Para completar aquellos valores que quedaron vacíos al unir los archivos, se asignaron valores NaN. El Anexo I (Sección 1.2) contiene una descripción detallada de este análisis para que pueda ser revisado en su totalidad

FLA

Anteriormente, mencionamos que uno de los datasets sobre la producción de lana ovina se construyó a partir de los datos recolectados en la página web del FLA. Sin embargo, para que este dataset sea considerado como candidato para entrenar nuestros modelos, necesitábamos contar con información específica sobre la producción de lana dividida por departamentos. En la provincia de Chubut, la Ley III N° 17 establece la obligatoriedad para todos los productores ganaderos de la provincia de presentar una declaración jurada anual. Estos datos son procesados anualmente por el Ministerio de la Producción, a través de su dirección de marcas y señales [21].

Primera Iteración

Entonces, a través de la mencionada encuesta, era posible conseguir la información departamental que necesitábamos. En la sección de estadísticas y censos del SIPAS, se encontraban disponibles los resultados de las encuestas realizadas para los años 2004-2005, 2006-2007, 2008-2009, 2009-2010, 2010-2011, 2012, 2013, 2014. Lamentablemente, solo pudimos acceder a la encuesta del 2014 [148], las demás daban un error 500. Si bien supusimos que el formato de la encuesta del 2014 era similar al de las anteriores, realmente nos hubiera sido de mucha utilidad poder acceder a las demás encuestas, ya que nos permitiría conocer la cantidad de lana producida por departamento en Chubut, que es exactamente lo que estábamos buscando. Se invirtieron muchas horas tratando de conseguir acceso a las demás encuestas en otras páginas web, pero no se obtuvieron resultados positivos. Dado que nosotros necesitábamos esa información y no contábamos con los datos para todos los años, tomamos la decisión de generalizar la cantidad de lana producida por cada departamento en el año 2014 para los años restantes, reconociendo que esta aproximación podría no ser totalmente precisa.

En la encuesta del 2014 se incluyeron dos grandes resúmenes que contenían información sobre las existencias ganaderas y la producción de fibras por departamento, así como las unidades ganaderas para ovinos, bovinos, caprinos y porcinos. Dentro de la sección de ovinos, que era la de nuestro interés, había una tabla que proporcionaba el total de ovinos para cada departamento, desglosado por categorías como ovejas, carneros, capones, borregos y corderos señalados, así como información sobre la lana sucia, unidades ganaderas declaradas en el operativo censal, el engorde de ovinos, los productores y el porcentaje de ovinos en relación al total de ovinos en la provincia. Nos interesamos en los totales de ovinos, ovejas, carneros, capones, borregos, corderos y la cantidad de lana sucia. Otra tabla relevante para nosotros fue la de lanas e indicadores productivos por departamento, en la que se incluía información sobre la cantidad de ovinos esquilados, la cantidad de lana sucia, los kilogramos de lana por cabeza, la finura promedio, el rinde promedio y el porcentaje de lana sucia. De esta tabla, nos interesó la cantidad de ovinos esquilados, la cantidad de lana producida (que era la misma que en la tabla anterior), los kilogramos de lana por cabeza, la finura promedio, el rinde promedio y el porcentaje de lana sucia. Además, agregamos un porcentaje de ovinos esquilados, ya que teníamos información sobre la cantidad de ovinos esquilados y la cantidad de ovinos por cada departamento. Por último, había una tercera tabla que incluía información

sobre las ovejas en servicio, los corderos señalados y el porcentaje señalado. Aunque el porcentaje señalado era un dato que existía en el dataset chiquitito que nos pasaron sobre un campo de la región del Chubut, era sabido que no íbamos a poder obtener este dato para el resto de los años y departamentos, por lo que decidimos dejarlo afuera y no nos quedamos con información de esta tabla de señalada por departamento.

A partir de los datos anteriores, se creó un archivo CSV de 15 columnas y 16 filas, 15 para cada uno de los departamentos de Chubut y una fila adicional para el total. Luego, se creó una lista de objetos que contenía el nombre de cada departamento, el porcentaje de lana producido en el año 2014, la finura y el rinde. Cada objeto de la lista tenía un formato similar al siguiente: *nombre: Biedma; porcentaje de lana: 4,67; finura: 19,22; rinde 58,86*. Posteriormente, se recorrió el conjunto de datos de lana obtenidos de la Federación Lanera Argentina, que solo incluía los totales anuales de lana producidos. Para convertir los totales de toneladas a kilogramos, se multiplicaron los valores por 1.000. Luego, se recorrió la lista de objetos creada anteriormente y se calculó el porcentaje de lana producido por cada departamento en función del total de lana producida. Para calcular la cantidad de lana producida en cada departamento desde el año 2000 al año 2022, se multiplicó el porcentaje de lana por el total de lana y se dividió por 100. Para obtener valores aleatorios para la finura y el rinde de la producción de lana en cada departamento, se usó una función de Python llamada «random.normalvariate(mu, sigma)», donde el promedio de los valores de finura y rinde de cada departamento eran «mu» para la distribución normal, y se eligió 0,95 como «sigma» para determinar cuánto varían los valores. Esto significa que los valores aleatorios generados iban a estar cerca de la media con una alta probabilidad (alrededor del 95%), pero también pueden variar en ambas direcciones. Finalmente, se creó un DataFrame con cinco columnas: año, departamento, kilogramos de lana, finura y rinde. Este DataFrame se almacenó en un archivo CSV y se documentó el proceso de limpieza en el Anexo I (Sección 1.8).

Segunda Iteración

Durante la redacción de esta tesina, realizamos un último intento en la sección de estadísticas y censos del SIPAS para acceder a las encuestas que no fue posible obtener anteriormente. Para nuestra sorpresa, pudimos acceder a todas las encuestas desde 2004 hasta 2014, lo que nos brindó información valiosa sobre la producción de lana en cada departamento de Chubut. Nos dimos cuenta de que, aunque la encuesta del año 2014 era la más completa, las anteriores también eran útiles, aunque algunas contenían sólo una tabla y otras solo cubrían cuatro departamentos. Sin embargo, tener al menos diez años de datos reales fue mucho mejor que haber tenido que generalizar la información a partir de un solo año. Aunque nuestro objetivo era obtener datos productivos desde el año 2000 hasta el 2022, la disponibilidad de esta información nos permitió ver qué tan alejada estaba nuestra generalización anterior de la realidad. Aunque aún debimos generalizar los datos para los años faltantes, pero tener diez años de datos reales nos permitió hacer una estimación mucho más precisa. En definitiva, esta nueva información cambió la forma en que construimos los datos y nos brindó una mejor base para nuestro análisis. El detalle completo de este análisis se encuentra disponible en las secciones 1.9 a 1.12 del Anexo I.

4.3.4 Comparar datasets obtenidos

Luego de llevar a cabo la depuración, limpieza y filtrado de los conjuntos de datos obtenidos de diversas fuentes, ha llegado el momento de realizar su comparación.

Datos meteorológicos

Respecto a los datos del clima, solo disponíamos de un único conjunto de datos obtenido de la página web del SIPAS, el cual se recolectó a partir de las 60 estaciones meteorológicas distribuidas por toda la Patagonia. Después de realizar una exhaustiva limpieza de los datos, este es el conjunto de datos definitivo que utilizaríamos en nuestro análisis.

Datos sobre existencias ovinas

Después de recolectar y limpiar los datos sobre existencias ovinas en la provincia de Chubut, teníamos a disposición cuatro conjuntos de datos potenciales para usar en el análisis. El primer conjunto de datos provenía de los indicadores ovinos y proporcionaba información sobre el número total de cabezas anuales por departamento en la provincia de Chubut desde 2008 hasta 2019. El segundo conjunto de datos era el de existencias ovinas de SENASA, que ofrecía información sobre el número total de ovinos y su categoría desde 2008 hasta 2020. Por último, contábamos con otro dataset de existencias ovinas para el periodo 2008-2019, cuyos datos originales provenían del MAGyC, pero se obtuvieron a través de «Datos Argentina» y al igual que los anteriores presentaba los ovinos separados por categoría, aunque a diferencia del dataset anterior, no se encontraba totalizado el número total de ovinos.

Para hacer una comparación lo más justa posible entre los tres dataset, lo que se hizo fue seleccionar un departamento y un año para hacer la comparación. Se seleccionó el departamento de Biedma y el año 2008.

Tabla 4.3.1. Comparación datasets existencias ovinas.

fuelle	año	depto	carneros	ovejas	borregos	capones	cordero/as	total ovinos
1 - indicadores ovinos	2008	BIEDMA	Nan	Nan	Nan	Nan	Nan	291.332
2 - existencias senasa	2008	BIEDMA	7.686	121.561	58.576	69.633	33.876	291.332
3 - existencias MAGyC	2008	BIEDMA	7.686	121.561	58.576	69.633	33.876	Nan

Después de comparar los tres datasets de existencias ovinas para Biedma en el año 2008, se encontró que el primer y el segundo dataset coinciden en el total de ovinos, 291.332. Además, el segundo dataset coincide en todas las columnas de categorización de ovinos (cordero, oveja, carnero, etc.) con respecto al tercer dataset. Sin embargo, el tercer dataset no cuenta con la totalización de ovinos. Es importante mencionar que el primer y tercer datasets contienen datos desde el año 2008 al 2019, mientras que el segundo dataset incluye datos hasta el año 2020. Por lo tanto, se seleccionó el dataset de SENASA como el más adecuado para las existencias ovinas en Chubut. Cabe destacar que existía un «cuarto» dataset con

datos desde 2000 hasta 2007 que se creó a partir de las existencias de 1965 a 2017, categorizadas a partir de los datos de existencias ovinas de SENASA, y por ende eran una precuela de los mismos.

Datos sobre lana ovina

Tenemos tres posibles datasets para utilizar en relación a la producción de lana ovina en la provincia de Chubut. El primer dataset se obtuvo de los indicadores ovinos, el cual proporciona información sobre la cantidad de lana sucia producida en la provincia desde el año 2000 hasta el 2022, con un total de 22 filas. El segundo dataset se obtuvo de la Federación Lanera Argentina, también con 22 filas que representan la cantidad de lana sucia producida en Chubut en el mismo periodo de tiempo. El tercer dataset se extrajo de los movimientos ovinos y contiene seis filas que representan diferentes tipos de lana sucia y limpia producida entre el 2013 y el 2018, expresados en kilos, por lo que se transformó la columna de toneladas para facilitar la comparación con los otros dos datasets. Es importante mencionar que el tercer dataset no cuenta con información completa para todos los años y que los primeros dos datasets contienen la información completa desde el año 2000 hasta el 2022.

Para hacer una comparación lo más justa posible lo que se hizo fue seleccionar un año para hacer la comparación. Se seleccionó el año 2017.

Tabla 4.3.2. Comparación datasets producción de lana.

fuelle	año	unidad	lana sucia fina	lana sucia mediana	lana sucia	lana limpia	lana sucia vellón	lana sucia no vellon	total lana
1 - indicadores ovinos	2017	tn	13.700	300	14.000	Nan	Nan	Nan	14.000
2 - FLA	2017	tn	13.700	300	14.000	Nan	Nan	Nan	14.000
3 - movimientos ovinos	2017	tn	Nan	Nan	4.605,71	24.37	8.237,71	456,84	13.324,65

Los primeros dos datasets contienen la misma información. Al comparar en toneladas con el total de lana, se observa que el tercer dataset estuvo muy cerca de los otros dos, lo cual es valioso, ya que fue construido a partir de algunas suposiciones y estudios científicos, esto nos da la pauta de que nuestro trabajo estuvo bien hecho. Sin embargo, se decidió trabajar con el dataset obtenido de FLA.

4.3.5 Combinar datasets seleccionados

Luego de recopilar, depurar y comparar los distintos conjuntos de datos, obtuvimos cuatro datasets separados: uno sobre la producción de lana, dos sobre el stock ovino y otro más sobre las condiciones climáticas en Chubut. Lo siguiente fue analizar cada uno de estos sets,

encontrar los puntos en común y finalmente unirlos en un único conjunto de datos para ser utilizado en el entrenamiento de modelos de machine learning.

En primer lugar, se trabajó con los dos datasets de existencias ovinas, uno con datos desde el 2000 hasta el 2007 y el otro con datos desde el 2008 al 2020. Al tener las mismas columnas, se pudo unir ambos datasets sin complejidad. Luego, se procedió a fusionar el DataFrame resultante con el dataset de producción de lana. Para ello, se utilizó la función merge de Pandas y se establecieron como puntos en común el departamento y el año. El resultado fue un DataFrame con 336 filas y 12 columnas, incluyendo información sobre el número de carneros, ovejas, borregos, capones, corderos y corderas, así como el total de ovinos, los kilogramos de lana producidos, la finura y el rendimiento. Este DataFrame se guardó en un archivo CSV llamado «produccion_chubut_2000_2020». Para obtener más información acerca de esta unión de dataset, consultar la sección 1.13 del Anexo I, donde se encuentra el detalle completo.

Lo siguiente fue combinar los datasets de producción ovina y clima en Chubut, para tener una visión completa de la relación entre los factores climáticos y la producción ovina. Para realizar esta tarea, se utilizó la librería pandas de Python y se partió de dos dataframes diferentes: «produccion_chubut_2000_2020» y «clima_chubut». El primero tenía 336 filas y 12 columnas, mientras que el segundo tenía 209 filas y 7 columnas, y ambos compartían la columna «departamento» y «año». Para unir ambos dataframes, se utilizó la función «merge» de pandas, y se especificó que la columna en común eran «departamento» y «año». Se eligió un inner merge, que eliminó las observaciones que no tenían datos en ambos datasets, lo que ocasionó una pérdida de 171 observaciones de datos productivos y 44 observaciones de datos climáticos. La combinación de los datasets resultó en un total de 165 filas, las cuales se guardaron en un archivo CSV. El archivo resultante incluye 17 columnas, que proporcionan información detallada sobre los datos climáticos y la producción ovina en Chubut, dando una imagen más completa. Estas columnas incluyen información sobre el número de carneros, ovejas, borregos, capones, corderos y corderas, el total de ovinos, los kilogramos de lana producidos, la finura y el rinde de la lana, el promedio de humedad, el total de lluvia, y las temperaturas máximas, medias y mínimas. Para conocer más acerca de cómo se unieron los datasets, consultar la sección 1.14 del Anexo I, donde se presenta el detalle completo de este proceso.

Después de combinar los datos productivos y climáticos a nivel departamental, se decidió incluir los datos obtenidos del campo El Mirabueno en el dataset. Esto resultó en un conjunto de datos de 201 filas y 17 columnas que proporcionó información tanto a nivel departamental como a nivel de campo. No obstante, es crucial tener en cuenta que los valores en el conjunto de datos del campo son significativamente menores que los del nivel departamental, es decir, más magnitudes son distintas debido a que en un campo se manejan miles de cabezas y kilogramos de lana, mientras que en un departamento se manejan millones. Es esencial considerar esta diferencia de escalas al analizar los resultados del conjunto de datos combinado. Para obtener información adicional sobre la unión de los datasets, puede revisar la sección 1.15 del Anexo I, donde se describe de manera detallada todo el proceso.

Por último, sobre el dataset de datos productivos y climáticos que se obtuvo anteriormente, se hizo una actualización teniendo en cuenta los nuevos datos que se consiguieron a través de las encuestas ganaderas, lo cual se detalla en el Anexo I (Sección 1.16).

Después de un riguroso proceso de limpieza y fusión de datos, se generaron tres conjuntos de datos que iban a ser utilizados para el análisis y entrenamiento de modelos. El primer conjunto de datos contiene únicamente información productiva y climática de los departamentos de la provincia de Chubut, y para referirse a él en el resto de la tesis se utilizará el nombre «Dataset 1». El segundo conjunto combina los datos anteriores con los provenientes del campo de ganadería ovina El Mirabueno, incluido el dato de las hectáreas, y será llamado «Dataset 2». Por último, el tercer conjunto de datos incluye también información productiva y climática de los departamentos, pero ha sido actualizado con los datos obtenidos a través de encuestas, y este conjunto será referido como «Dataset 3». De esta manera, se cuenta con tres conjuntos de datos que permitirán un análisis exhaustivo y una mayor precisión en los resultados obtenidos por los modelos que se utilicen.

4.4 Análisis del dataset

Una vez terminada la limpieza, llegó el momento de conocer nuestros datos. Antes de crear cualquier modelo de aprendizaje automático, es importante examinar los datos para verificar si la información necesaria se encuentra presente en ellos, para asegurarse de que sean precisos y estén listos para utilizarlos en el aprendizaje automático. Por otro lado, inspeccionar la información es una buena manera de detectar anomalías y peculiaridades como mediciones inesperadas, debido a que es común encontrar incoherencias en los datos del mundo real [41].

En la siguiente sección, se trabajará con los tres datasets obtenidos anteriormente de información ovina de la provincia de Chubut, donde las principales categorías de interés son ovejas, corderos, capones y carneros, dependiendo del momento en el ciclo de vida del animal y otras variables como kilogramos de lana obtenidos en los subsiguientes procesos de esquila acompañados de información sobre el rinde y la finura de la lana obtenida. El ciclo de vida de una oveja tiene un gran impacto en las existencias ovinas. Dicho ciclo comienza con el nacimiento de un cordero, el cual, a medida que crece y se desarrolla, se convierte en un borrego o en una oveja, según su género. Los borregos, si son castrados, se convierten en capones y se utilizan para la producción de carne. Las ovejas, por otro lado, se crían para la producción de lana y carne. Finalmente, los carneros, que son machos adultos, se utilizan para la reproducción y son capaces de fertilizar a varias ovejas durante la temporada de apareamiento. Los carneros pueden ser sacrificados para la producción de carne cuando ya no son útiles para la reproducción.

En primer lugar, se realizará un análisis estadístico básico de cada columna que permita comprender la distribución y variabilidad de los datos. Luego, se elaborará una matriz de correlación para evaluar las interrelaciones entre las diferentes variables, y, por último, se visualizarán los datos mediante gráficos y diagramas, evidenciando sus patrones y tendencias de manera más clara. Esto permitirá conocer mejor la distribución y característica de los

datos y entender la relación entre las diferentes variables de interés, lo que es esencial para crear un modelo de aprendizaje automático preciso y efectivo.

4.4.1 Análisis estadístico

La realización de análisis estadísticos de datos es una práctica fundamental en la mayoría de las áreas del conocimiento, ya sea en la investigación científica, el análisis de datos empresariales, o en el campo de la salud pública, por nombrar algunos ejemplos. La importancia del análisis estadístico radica en su capacidad para descubrir patrones, relaciones y tendencias en los datos, lo que permite extraer información valiosa y tomar decisiones basadas en datos objetivos. Además, el análisis estadístico permite evaluar la confiabilidad de los datos y las conclusiones obtenidas a partir de ellos, lo que es crucial para garantizar la validez y la credibilidad de los resultados. En resumen, la realización de análisis estadísticos de datos es esencial para comprender, interpretar y utilizar los datos de manera efectiva en cualquier campo del conocimiento. A continuación vamos a ver los resultados del análisis estadístico realizado sobre los tres datasets obtenidos.

Dataset 1: Datos a nivel departamental

Al realizar el análisis estadístico de este dataset, encontramos 165 observaciones (es decir, 165 filas) y 17 características diferentes, que incluyen seis relacionadas con las existencias ovinas, tres relacionadas con la producción de lana, seis relacionadas con los datos meteorológicos, una columna de fecha que nos indica el tiempo, y dos columnas relacionadas con el lugar geográfico donde se tomaron las muestras (provincia y departamento). Pudimos observar que, dado que construimos el dataset, no contenía ningún dato faltante, por lo que estaba sumamente completo y pesaba 22 kb.

Al realizar un análisis de los datos, utilizando la función `describe()` de pandas, se pudieron observar una serie de estadísticas relevantes. Respecto a las existencias, encontramos que la media de ovejas es de 142.959, lo que sugiere que son el tipo de animal más prevalente en la producción ovina de la provincia. Por otro lado, la media de borregos es de 48.585, mientras que la media de capones es de 51.529 y la de corderos es de 42.996. Estos datos reflejan la dinámica del ciclo de vida de la oveja, en la que los animales pasan por diferentes etapas antes de alcanzar la madurez sexual y ser utilizados para diferentes fines. En cuanto a los carneros, se puede observar que su media es de 8.588, lo que representa sólo el 2,92% del total de ovinos (294.659), esta cifra refleja la función principal de los carneros en la producción ovina, que es la reproducción. La mayoría de los carneros son sacrificados para la producción de carne una vez que ya no son útiles para la reproducción, lo que explica por qué su media es relativamente baja en comparación con otros tipos de ovinos. Por otro lado, las ovejas y los borregos son criados principalmente para la producción de lana y carne, lo que explica su mayor número en comparación con los carneros. La producción de lana en Chubut para este dataset, tiene una media de 1.086.844 kilos, con una finura media de 19,79 micrones y un rinde medio de 58,20, lo que indica que en general la provincia produce lana de buena calidad con lanas superfinas y un rendimiento al lavado aceptable. En cuanto a los registros meteorológicos, se observa que la humedad promedio es de 53,57, mientras que la temperatura máxima, media y mínima presentan una media de 18,45, 11,59 y 4,91,

respectivamente. Con respecto a las precipitaciones, factor clave en la producción ovina de la región, se encuentra una media de 213,75 mm, con un mínimo de 7,1 y un máximo de 1.041,16. Esto coincide con lo señalado previamente acerca de la clasificación climática de Köppen-Geiger, ya que en Chubut se presentan diferentes tipos de clima, con lluvias muy variadas según la región.

Luego se utilizó la función «skew()» de pandas, y se pudo observar que ninguno de los datos seguían una distribución normal. Para las existencias, la mayoría de los coeficientes de asimetría eran superiores a 1,41. En cuanto a los kilos de lana, la asimetría era de 1,69, mientras que tanto el rinde como la finura tenían un coeficiente de asimetría negativo. En cuanto a los datos climáticos, algunas variables tenían una asimetría positiva y otras una asimetría negativa.

Finalmente, se observó que el departamento Rawson es el que tenía más observaciones, con un total de 21, mientras que los departamentos de Biedma y Languiño son los que tenían menos observaciones, con 9 y 8 respectivamente. Además, se notó que a partir del año 2013, se cuenta con doce observaciones para cada uno de los años restantes en el conjunto de datos. Para ver el análisis completo, consultar la sección 2.1 del Anexo I.

Dataset 2: Datos a nivel departamental más El Mirabueno

Durante el análisis estadístico del conjunto de datos, se descubrieron 201 filas y 17 características diferentes. Entre estas características, seis estaban relacionadas con las existencias ovinas, tres con la producción de lana, seis con datos meteorológicos, una columna de fecha y dos columnas relacionadas con la ubicación geográfica (hectáreas y departamento). Fue posible determinar que el conjunto de datos estaba completo, ya que no había ninguna entrada faltante, y tenía un tamaño de 26,8 kb.

Al llevar a cabo el análisis de los datos mediante la función describe() de pandas, se obtuvieron una serie de estadísticas de importancia. En relación a las existencias, se encontró que la media de ovejas es de 117.623, lo que sugiere que son los animales más dominantes en la producción ovina de la provincia. Asimismo, la media de borregos fue de 39.972, mientras que la media de capones y corderos fue de 42.412 y 35.389, respectivamente. Estos datos reflejan el proceso de crecimiento y desarrollo de las ovejas, el cual implica diferentes etapas antes de llegar a la madurez sexual y ser destinados a diversos fines. En cuanto a los carneros, se observó que su media fue de 7.059, lo que representa solo el 2,91% del total de ovinos (242.456). Esto pone de manifiesto su función principal en la producción ovina, que es la reproducción. La mayoría de los carneros son sacrificados para la producción de carne una vez que no son útiles para la reproducción, lo que explica por qué su media es relativamente baja en comparación con otros tipos de ovinos. En contraste, las ovejas y los borregos son criados principalmente para la producción de carne y lana, lo que justifica su mayor cantidad en comparación con los carneros. Para este dataset, la provincia de Chubut tiene una producción de lana promedio de 894.301,4 kilos, con una calidad buena debido a la finura media de 19,75 micrones y un rinde medio de 58,26. Los registros meteorológicos muestran una humedad media de 53,43 y temperaturas máximas, medias y mínimas con valores promedio de 18,93, 11,92 y 5,07, respectivamente. Las precipitaciones, un factor crucial para la producción ovina en la zona, tienen una media de 208,02 mm, con variaciones importantes

dependiendo de la región, lo que está en línea con la clasificación climática de Köppen-Geiger en Chubut, que presenta diversos tipos de clima y diferentes niveles de lluvia. Posteriormente, se aplicó la función «skew()» de pandas, la cual reveló que ninguno de los datos presentaba una distribución normal. En el caso de las existencias, la mayoría de los coeficientes de asimetría fueron superiores a 1,21. En cuanto a la cantidad de lana producida, se observó una asimetría de 1,50, mientras que tanto el rendimiento como la finura presentaron un coeficiente de asimetría negativo. Por otro lado, en cuanto a los datos climáticos, algunas variables presentaron una asimetría positiva, mientras que otras presentaron una asimetría negativa.

Por último, se pudo observar que el campo El Mirabueno y el departamento Rawson eran los lugares con mayor cantidad de observaciones, con 36 y 21 respectivamente, mientras que los departamentos de Biedma y Languiño presentaban la menor cantidad de datos, con 9 y 8 observaciones respectivamente. Además, se destacó que a partir del año 2013, se registraron trece observaciones para cada uno de los años siguientes en el conjunto de datos. Para acceder a la totalidad del análisis, revisar la sección 2.2 del Anexo I.

Dataset 3: Datos a nivel departamental actualizado

Al analizar el conjunto de datos, se encontraron 164 observaciones y 17 características diferentes, que incluyen información sobre existencias ovinas, producción de lana, datos meteorológicos, tiempo y ubicación geográfica. No se encontró ningún dato faltante, lo que indica que el conjunto de datos es completo (21,9 kb). Al utilizar la función describe() de pandas, se encontraron estadísticas relevantes, como que la media de ovejas en la provincia es de 109.195, lo que sugiere que son el tipo de animal más común en la producción ovina. La media de borregos es de 40.317, la de capones es de 39.883 y la de corderos es de 54.915, lo que refleja la dinámica del ciclo de vida de la oveja. La media de carneros es de 6.795, lo que representa sólo el 2,70% del total de ovinos, y su función principal es la reproducción, mientras que las ovejas y borregos son criados para la producción de lana y carne, lo que explica su mayor número. La producción de lana en Chubut para este dataset, en promedio, alcanza los 894.000,5 kilos y se caracteriza por tener una finura media de 20,08 micrones y un rinde medio de 57,11. Esto sugiere que, en general, la calidad de la lana producida en la provincia es buena, con fibras finas y un rendimiento al lavado aceptable. Respecto a los datos climáticos, se puede observar que la humedad promedio en la región es de 53,52, mientras que las temperaturas máxima, media y mínima promedio son de 18,45, 11,60 y 4,92, respectivamente. En lo que se refiere a las precipitaciones, un factor clave en la producción ovina, se encontró una media de 214,45 mm, con un mínimo de 7,1 y un máximo de 1.041,16. Esto está en línea con la clasificación climática de Köppen-Geiger, que indica que en Chubut se presentan diversos tipos de clima, con lluvias muy variables según la región.

Al analizar la distribución de los datos mediante la función «skew()» de pandas, se pudo observar que ninguno de los datos seguían una distribución normal. Las existencias presentaron una distribución sesgada hacia la derecha, con la mayoría de los coeficientes de asimetría superando 1,24. En cuanto a los kilos de lana, se encontró una asimetría positiva de 1,62, mientras que el rinde mostró una asimetría negativa y la finura una asimetría positiva. Por su parte, los datos meteorológicos presentaron tanto asimetría positiva como negativa en

distintas variables. La temperatura máxima fue la variable más simétrica del conjunto de datos, con un coeficiente de asimetría de -0,008.

Para terminar, se detectó que el departamento de Rawson presentó la mayor cantidad de observaciones en el conjunto de datos, con un total de 21, mientras que los departamentos de Biedma y Languiño presentaron la menor cantidad de observaciones, con 9 y 7 respectivamente. Además, se observó que a partir del año 2013 se registraron doce observaciones para cada uno de los años restantes en el conjunto de datos. Consultar la sección 2.3 del Anexo I, para conocer todos los detalles del análisis.

4.4.2. Visualización de datos

Después de realizar un análisis estadístico de los datos, llegamos a la conclusión de que los tres conjuntos de datos tenían características similares, lo cual era de esperar dado que los tres se referían a departamentos de Chubut, y en uno de ellos solo se disponía de datos de un campo de la región. Para lograr una comprensión más clara de los datos, procedimos a realizar un análisis visual, ya que es una de las mejores formas de inspeccionar los datos. La visualización nos ayudó a simplificar la información compleja para entenderla mejor, inspeccionarla y obtener una comprensión más profunda de los datos. Utilizamos la visualización de datos en casi todos los pasos del proceso de construcción del modelo, desde la recopilación y exploración de datos hasta la evaluación de los resultados finales. En particular, las herramientas de visualización, como histogramas, gráficos de dispersión, gráficos de caja, gráficos de línea y gráficos de barras, nos permitieron confirmar la precisión de los datos, ponerlos en contexto, encontrar la ausencia/presencia de tendencias y entender mejor la información disponible. Las visualizaciones fueron fundamentales para realizar análisis exploratorios de datos, donde pudimos buscar patrones, encontrar anomalías, probar hipótesis y verificar supuestos sin necesidad de un modelado formal. En definitiva, utilizamos las visualizaciones como una herramienta clave para desentrañar los datos de manera más efectiva y obtener resultados significativos [40][143].

Gráficos

A continuación, describiremos cada uno de los gráficos que se utilizaron para la visualización de datos de nuestros datasets, y explicaremos qué información nos brindan sobre los datos, así como la forma correcta de utilizarlos y otras características relevantes.

Histograma

Un histograma es una herramienta gráfica utilizada para visualizar la distribución de datos cuantitativos continuos agrupados en intervalos. Consiste en un conjunto de rectángulos adyacentes, donde el ancho de cada rectángulo corresponde al tamaño del intervalo y la altura representa la frecuencia o el número de observaciones dentro del intervalo. Si las clases son del mismo ancho, las barras tendrán el mismo ancho. Además, el área de cada rectángulo es proporcional a la frecuencia absoluta de la clase correspondiente en relación con todas las observaciones. En resumen, un histograma permite analizar la distribución de los datos y observar su frecuencia y concentración en cada intervalo [40][150].

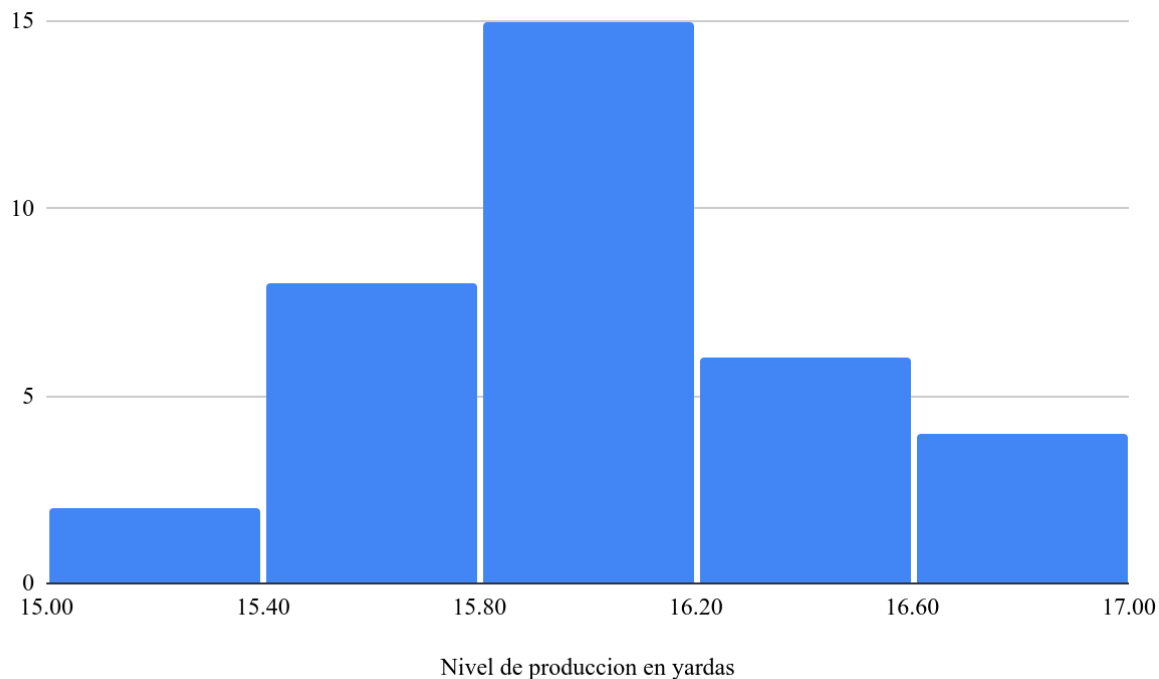


Figura 4.4.1. Distribución de frecuencia producción de telares.

A partir de los datos [150] que muestran la producción diaria de 30 telares para alfombra con intervalos de clase de 0,3 yardas, se construyó un histograma de frecuencia, que se observa en la figura 4.4.1. En la misma, se puede observar que las frecuencias de las clases con ancho de 0,3 yardas siguen una secuencia regular: el número de datos puntuales aumenta desde dos en la primera clase hasta ocho en la segunda, alcanzando un máximo de 15 en la tercera clase, para luego disminuir a seis y caer a cuatro en la quinta clase. Es importante destacar que cuanto más ancho sea el intervalo de clase, más suave será la progresión. Sin embargo, si las clases son demasiado anchas, podemos perder mucha información y la gráfica carecerá de significado [150].

Diagrama de caja

Un diagrama de caja o diagrama de caja y bigotes, es una representación de la distribución de un conjunto de datos utilizando cuartiles [40].

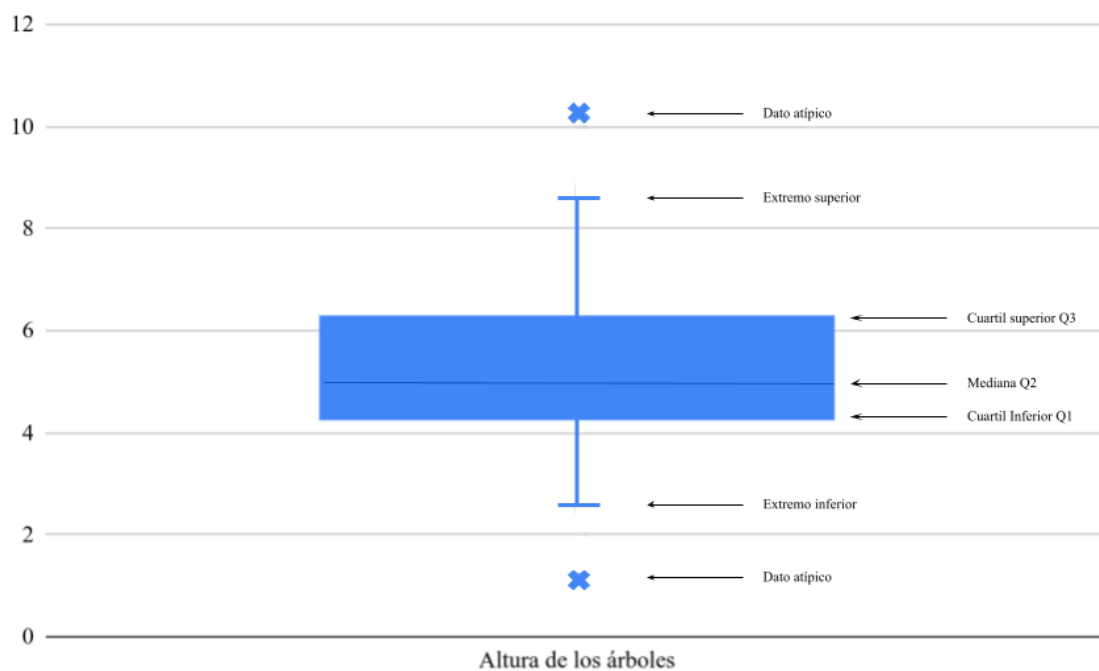


Figura 4.4.2. Diagrama de caja para altura de árboles.

El diagrama de caja es una herramienta gráfica útil para representar datos divididos en cuatro cuartiles iguales. En la figura 4.4.2 se muestra un ejemplo, donde se pueden identificar los tres cuartiles (Q1, Q2 y Q3) que separan el conjunto de datos en cuatro subgrupos iguales, desde el valor mínimo hasta el máximo. La mediana, que es el valor central, corresponde al cuartil Q2. En el gráfico, el rango intercuartil (RIC) se representa mediante una caja que abarca el 50% de los datos centrales. Los bigotes o brazos son las líneas que se extienden desde la caja hasta el valor mínimo y máximo, respectivamente, sin incluir los valores atípicos, que son los puntos fuera del rango del diagrama. El diagrama de caja no muestra la frecuencia ni las estadísticas individuales, pero es útil para identificar la simetría o asimetría en los datos. Es similar a los histogramas en términos de identificar la concentración de los valores y su distribución [40].

Diagrama de dispersión

Un diagrama de dispersión representa los datos colocando una característica en el eje horizontal (x) y otra en el eje vertical (y), y dibuja un punto para cada punto de datos. Este tipo de gráfico se utiliza para analizar la correlación y distribución de los datos, lo que permite detectar valores atípicos de manera temprana. Además, es útil para identificar la relación entre dos variables y determinar si una de ellas afecta a la otra. Al representar visualmente la relación entre dos variables en un formato compacto, se pueden comparar grandes cantidades de datos de manera efectiva. Cuanto más datos se incluyan en la gráfica, mejor será la comparación [40][151]. La figura 4.4.3 ilustra claramente la correlación positiva entre la temperatura y las ventas de helados. Se puede observar un aumento progresivo en las ventas a medida que la temperatura aumenta.

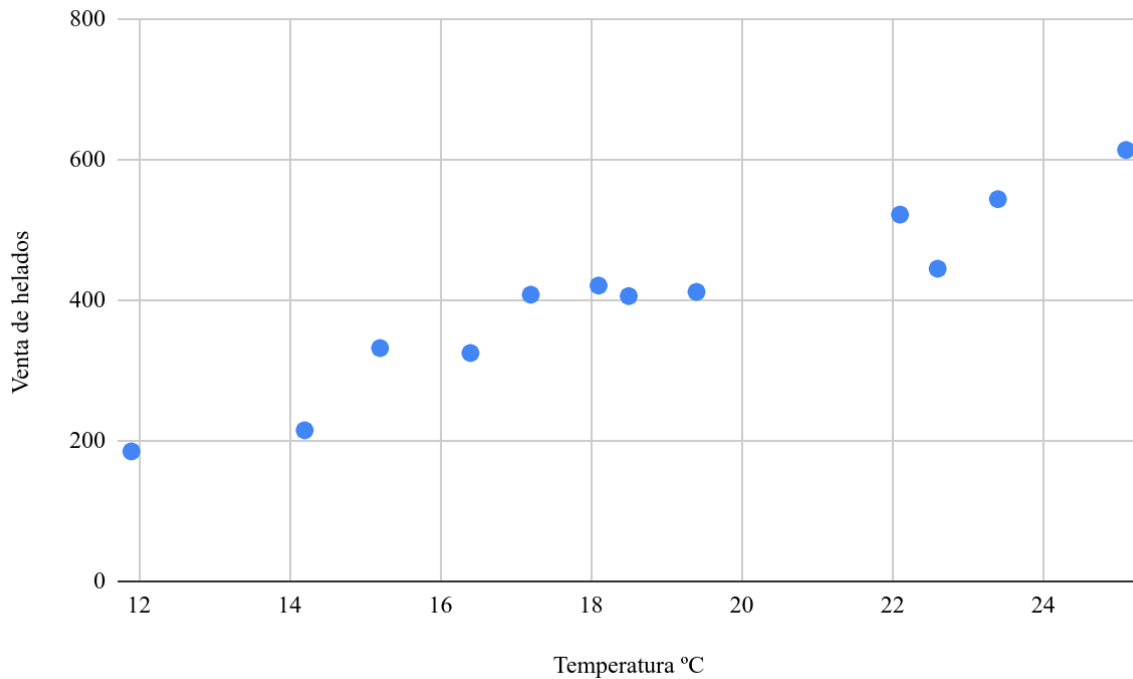


Figura 4.4.3. Diagrama de dispersión temperatura vs. venta de helados.

Lamentablemente, la representación de conjuntos de datos con más de tres características resulta difícil debido a la limitación de los medios de visualización con los que contamos. Para resolver este problema, se puede utilizar un gráfico de pares, que examina todas las posibles combinaciones de características. No obstante, es importante tener en cuenta que este tipo de gráfico no muestra la interacción de todas las características al mismo tiempo, lo que puede ocultar aspectos relevantes de los datos.

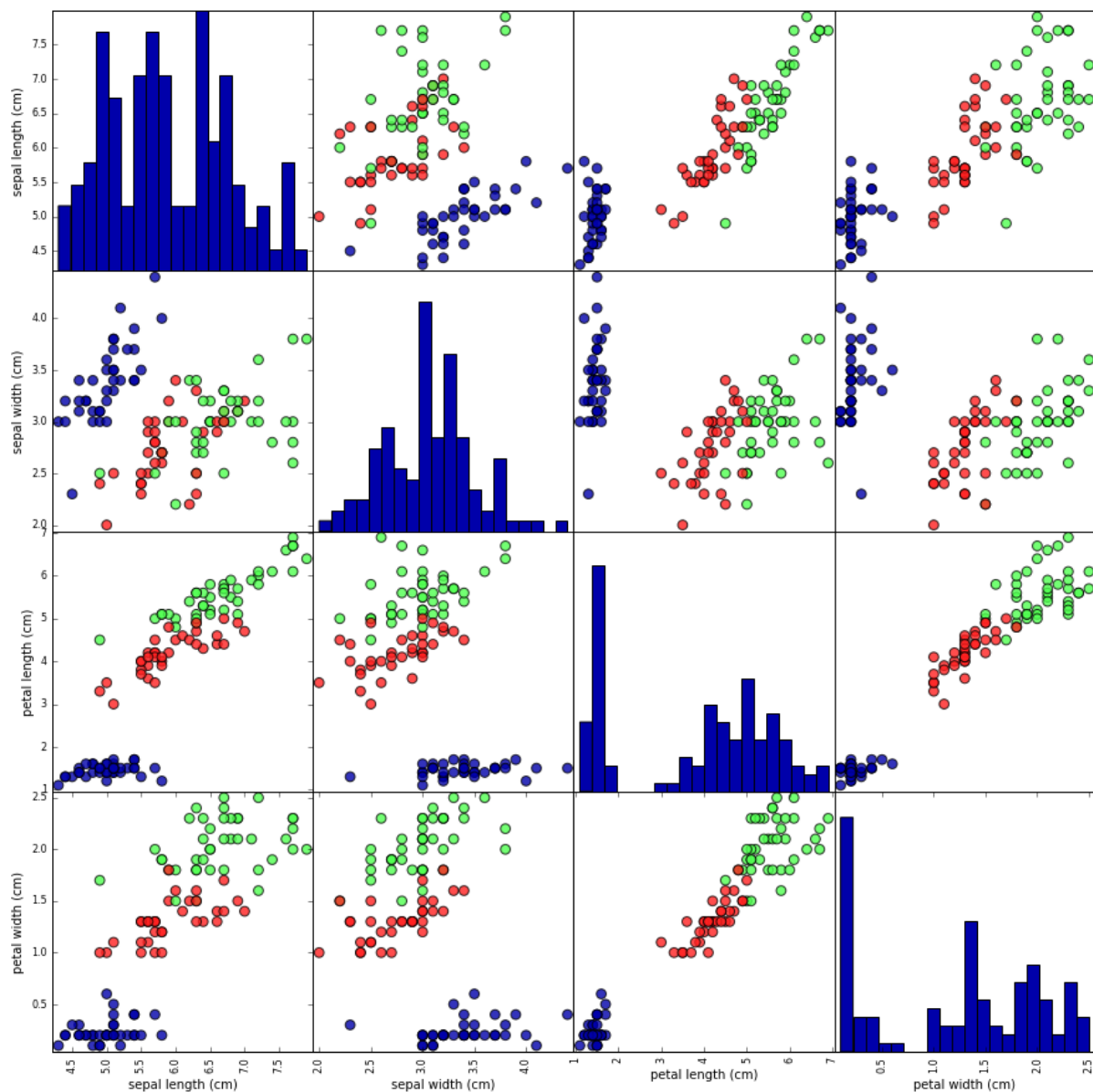


Figura 4.4.4. Diagrama de dispersión iris.

La figura 4.4.4 muestra el diagrama de dispersión que explora cómo se relacionan las longitudes de los sépalos y pétalos de las plantas de iris [41]. Este conjunto de datos se presenta mediante un gráfico de pares, en el cual se representan todas las posibles combinaciones de características del conjunto de datos. Cada punto de datos se colorea según la especie de iris a la que pertenece. Para crear este gráfico, se utilizó una función llamada «scatter_matrix», la cual forma parte de la biblioteca de Python denominada pandas [41]. Esta función permite crear gráficos de pares para visualizar la relación entre las características de un conjunto de datos. De esta manera, es posible identificar patrones y relaciones que no se pueden percibir en una representación de dos dimensiones.

Diagrama de barras

Un diagrama de barras es una herramienta de visualización de datos en la que se utilizan rectángulos de igual base para representar los datos sobre el eje de conceptos, y la longitud del otro lado corresponde al valor del dato, según la escala utilizada en el eje de valores. Cuando se grafican varias categorías, existen diferentes modalidades de presentación, siendo una de ellas el gráfico de barras verticales, también conocido como «gráfico de columnas». Este tipo de gráfico se distribuye sobre ejes cartesianos, en el que se muestra en el eje X los conceptos y sobre ellos se levantan barras o rectángulos de igual base cuya altura sea proporcional a sus frecuencias, mientras que en el eje Y se encuentra la escala de valores. La gráfica de barras permite comparar diferentes valores y resumirlos en categorías. Cada barra tiene un ancho idéntico y su altura se mide por la variable del valor total. Este tipo de gráfico se utiliza cuando los datos a analizar no son continuos o no tienen un único valor asociado, y cuando se necesitan comparar valores diferentes de dos o más categorías, analizar partes de un conjunto de datos, o hallar la relación entre varios conjuntos de datos con similitudes [151].

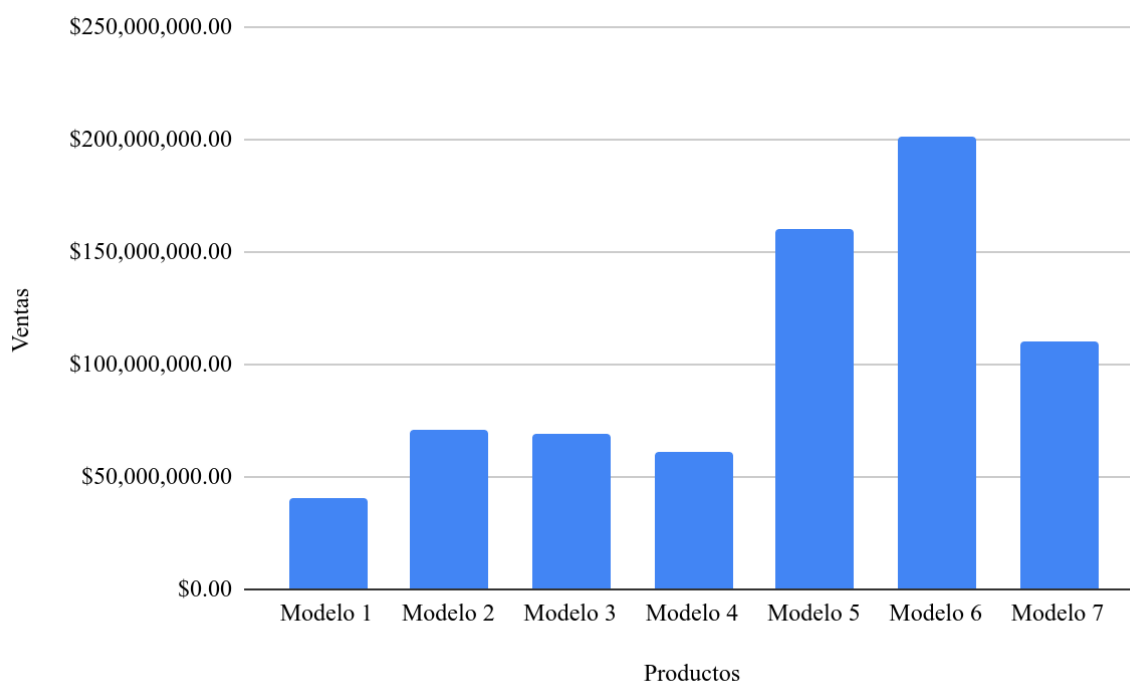


Figura 4.4.5. Diagrama de barras ventas en diferentes modelos.

En la figura 4.4.5 se puede observar un gráfico de barras que permite comparar las ventas de varios modelos de producción. Los modelos 5 y 6 se destacan por producir la mayor cantidad de ganancias, mientras que los modelos 1 y 4 presentan un rendimiento más bajo en términos de ganancias generadas. Este tipo de gráfico es útil para mostrar claramente las diferencias en el rendimiento de diferentes opciones y es comúnmente utilizado en informes de ventas y análisis de mercado.

Diagrama de líneas

Un diagrama se compone de una serie de puntos que se unen mediante líneas rectas. Cada línea representa un conjunto de datos que se compara con otro, y se establece un color

diferente para cada línea en la misma gráfica para facilitar su identificación. El gráfico de líneas es una herramienta útil para analizar tendencias en la producción o en cualquier otra actividad a lo largo del tiempo, para mostrar comparativas secuenciales de grupos de datos y para visualizar pequeños cambios en un grupo de datos. Existen gráficos de línea simple y los gráficos multilíneas.

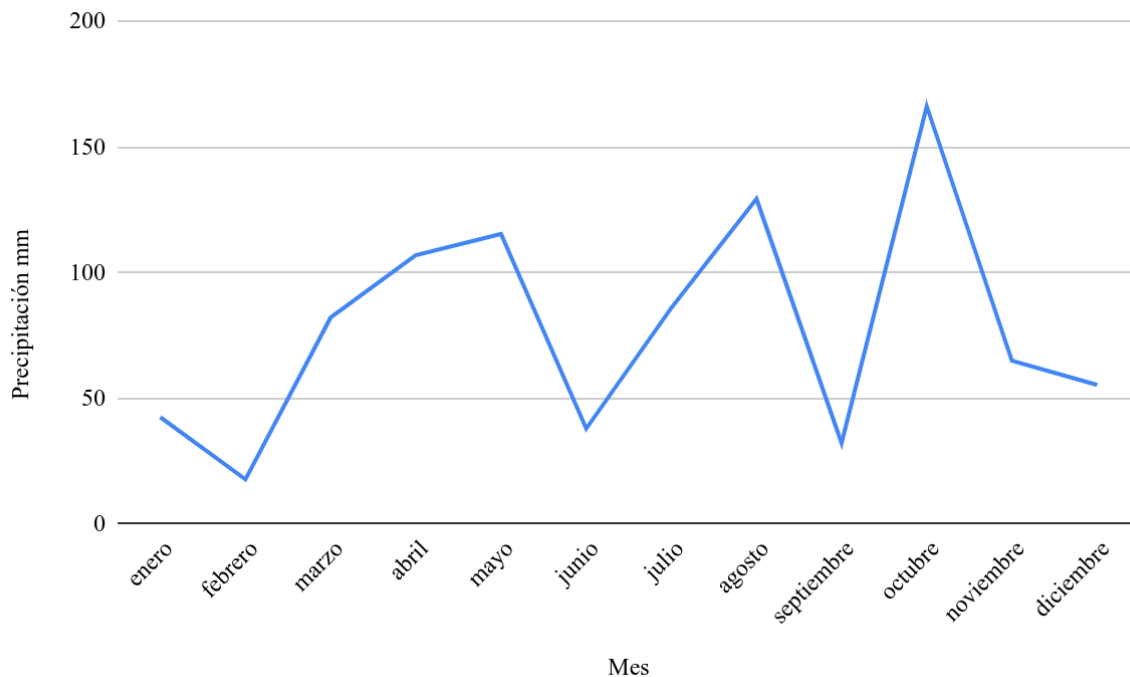


Figura 4.4.6. Diagrama de barras ventas en diferentes modelos.

Se puede observar en la figura 4.4.6 un gráfico de líneas simples que representa la variación en la cantidad de precipitaciones a lo largo de un año. El eje horizontal muestra los 12 meses del año, mientras que el eje vertical indica la cantidad de precipitaciones en milímetros.

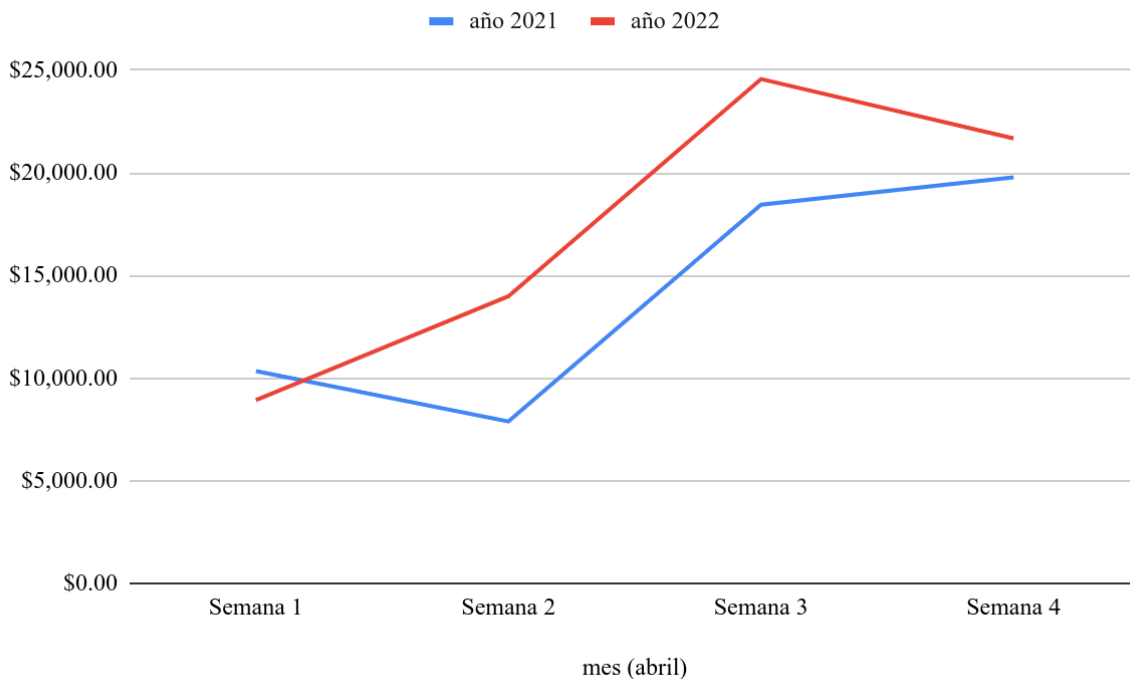


Figura 4.4.7. Diagrama de barras ventas en diferentes modelos.

En la figura 4.4.7, se puede observar una comparación entre las ventas del mes de septiembre del 2022 y las ventas del mismo mes del año 2021, utilizando los colores azul y rojo, respectivamente. Esta representación permite identificar de manera sencilla las diferencias y similitudes entre los datos de ambos años en el período de tiempo establecido. Es importante tener en cuenta que el uso de diferentes colores para cada año en el gráfico de multilíneas ayuda a visualizar de manera clara la evolución de los datos a lo largo del tiempo.

Dataset 1: Datos a nivel departamental

Durante nuestro análisis visual de datos, llevamos a cabo un gráfico de dispersión de pares y descubrimos una correlación lineal entre las existencias ovinas y las temperaturas, lo cual confirmamos posteriormente mediante una matriz de correlación. Sin embargo, en general, encontramos una relación negativa muy débil entre los datos productivos y los meteorológicos. Nos enfocamos luego en algunas variables productivas del conjunto de datos y las examinamos individualmente, en conjunto con la fecha y la lluvia acumulada, empleando gráficos de dispersión, de barras y de líneas. En algunos casos, dimos una dimensión adicional al identificar los departamentos por medio de un color para los puntos o las líneas. Para obtener una visión detallada del análisis llevado a cabo, por favor consulte el Anexo I (Sección 2.4).

Dataset 2: Datos a nivel departamental más El Mirabueno

Al inicio del análisis visual de datos, creamos un gráfico de barras que no nos proporcionó información útil, ya que cada columna del dataset tenía 165 observaciones y todas las barras eran iguales. Luego, al realizar un gráfico de dispersión de pares, descubrimos que las existencias ovinas y las temperaturas estaban linealmente correlacionadas entre ellas mismas, pero no se observó una relación entre los datos productivos y meteorológicos. Posteriormente, creamos una matriz de correlación para examinar más detalladamente la relación entre los datos, y descubrimos una fuerte correlación entre las existencias ovinas, como la cantidad de ovejas y la cantidad de corderos, así como la correlación entre las existencias ovinas y la producción de lana. Sin embargo, en la mayoría de los casos, se observó una débil correlación negativa al intentar establecer relaciones entre la finura y una o más variables, el rinde y una o más variables, la cantidad de lluvia acumulada y la producción de lana, así como entre la lluvia acumulada y la producción de corderos. Se prosiguió con el análisis considerando que la correlación no implicaba necesariamente causalidad y que había otras variables influyentes que debían ser exploradas. Nos enfocamos en ciertas variables y se buscó identificar otros elementos que pudieran afectar los resultados, con el objetivo de alcanzar una comprensión más completa de su interacción. Se puede consultar el análisis detallado en el Anexo I (Sección 2.4). Dado que este dataset se utilizó para entrenar a dos modelos finales, en particular al modelo que predice la finura y al modelo que predice el rinde de la lana, nos vamos a centrar en esas dos variables para describir visualmente lo que se ha observado en los datos.

Al analizar la finura de la lana, descubrimos que la gran mayoría de los datos se encontraban dentro del rango de 18 a 21 micrones, con excepción de dos datos atípicos, uno que superaba el límite inferior y otro que superaba el límite superior de la distribución. Al utilizar la función «swek()» en el análisis estadístico previo, se determinó que la distribución de la finura era casi simétrica, lo que se corroboró al examinar los correspondientes histograma y diagrama de caja. Sin embargo, al graficar solo las observaciones de la finura en un diagrama de dispersión y un diagrama de líneas, no se encontró un patrón significativo. Posteriormente, se intentó relacionar la finura con el año y los diferentes departamentos mediante la creación de un diagrama de dispersión, un diagrama de líneas y un diagrama de barras, pero no se observó ninguna tendencia clara en las diferentes regiones. También se intentó relacionar la finura con la cantidad de lluvia a través de diagramas de dispersión y de líneas, sin embargo, nuevamente no se pudo identificar una tendencia o patrón claro. No obstante, se observó que en algunos casos la lana se volvía más gruesa a medida que aumentaba la cantidad de lluvia, como se puede apreciar en la figura 4.4.8, donde se evidencia que a pesar de los picos muy pronunciados que presenta el gráfico, en el caso de los departamentos de Biedma, Florentino Ameghino y Rawson, las líneas tienden a subir a medida que aumenta la lluvia.



Figura 4.4.8. Gráfico multilíneas para finura de departamentos de la meseta

Al analizar el rendimiento de la lana, encontramos una distribución asimétrica negativa, lo que coincide con el análisis estadístico previo. Aunque el histograma presentaba cierta dificultad para interpretarse debido a la altura de las barras, daba la sensación de tener una cola hacia la izquierda, el gráfico de caja confirmó que no había ningún dato atípico afectando la distribución hacia ese lado, y además nos proporcionó el rango intercuartil de los datos que era entre 54,64 y 61,77. Al graficar las observaciones del rendimiento en un diagrama de dispersión y de líneas, se detectaron algunas agrupaciones difusas, pero no se encontró un patrón claro. Para ver si existía alguna tendencia temporal, se combinaron los datos del rendimiento con la fecha y se realizaron gráficos de dispersión, líneas y barras. Se encontró que los departamentos de Cushamen y Futaleufú tenían los valores más altos de rendimiento de la lana, superiores a 62,5, y que el departamento de Sarmiento presentaba los valores más bajos. Además, en el gráfico de líneas se observó que el rendimiento se mantenía relativamente constante entre 54 y 61,5 de rinde en todos los departamentos de Chubut. Luego, al combinar el rendimiento con la cantidad de lluvia, se descubrió que en algunos casos, a medida que aumentaba la cantidad de lluvia, también aumentaba el rendimiento de la lana. Esto podría deberse a que la lana estaba menos sucia y, por lo tanto, rendía más al lavarla. Una vez más, se encontró que los departamentos de Cushamen y Futaleufú tenían el mejor rinde, como se puede observar en la figura 4.4.9.

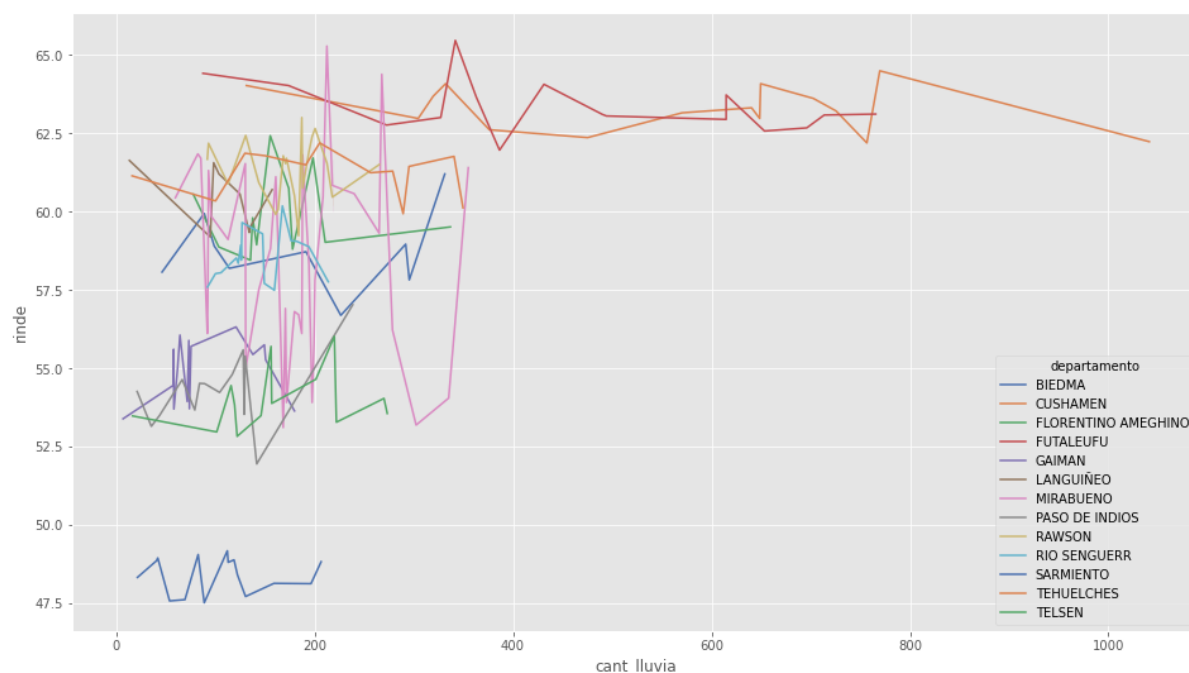


Figura 4.4.9. Gráfico multilíneas para rinde de departamentos

En ambos casos, no se observaron patrones claros o tendencias fuertes en la finura o el rendimiento de la lana. Sin embargo, el análisis visual nos proporcionó una guía sobre cómo abordar los modelos. Iniciamos con la exploración de modelos de regresión lineal y considerando que existía cierta linealidad temporal en los datos, también evaluamos modelos que tuvieron en cuenta el factor tiempo.

Dataset 3: Datos a nivel departamental actualizado

Al comienzo de nuestra exploración visual de datos, generamos un gráfico de barras que resultó ser poco informativo, ya que simplemente mostraba que cada columna del conjunto de datos tenía 165 observaciones. Luego, realizamos un diagrama de dispersión de pares, que nos reveló una correlación lineal entre las existencias ovinas y las temperaturas, pero, inicialmente, no encontramos ninguna relación entre los datos de producción y los datos meteorológicos. A continuación, creamos una matriz de correlación para examinar más detalladamente la relación entre los datos, donde confirmamos la fuerte correlación entre las existencias ovinas consigo mismas, como por ejemplo la cantidad de ovejas y la cantidad de corderos, que estaban altamente correlacionados con un valor de 0,99. También encontramos una correlación alta entre las existencias ovinas y la producción de lana, con un valor de 0,99 para los ovinos totales y los kilos de lana, lo cual tiene sentido, ya que la lana se obtiene a partir de la esquila de los ovinos. Además, encontramos una correlación significativa entre las temperaturas, donde la temperatura media estaba 0,95 relacionada con la temperatura máxima y 0,91 relacionada con la temperatura mínima. Había dos relaciones que estábamos buscando, una relación entre la cantidad de lluvia acumulada y la cantidad de kilos de lana producidos, y una relación entre la lluvia acumulada y la cantidad de corderos producidos. Pero en la mayoría de los casos, encontramos una relación negativa muy débil con valores entre -0,13 y -0,27, lo que indicaba que la lluvia no estaba relacionada de manera significativa con la

producción de lana o corderos. Aunque tuvimos en cuenta que la correlación no implicaba causalidad y que podían existir otros factores que influyeran en las relaciones, y que no estaban claros en la matriz de correlación, por ende continuamos con el análisis. En este proceso, nos enfocamos en variables específicas que tenían el potencial de servir como predictores en un modelo, examinamos detenidamente las mismas para determinar si existía alguna relación significativa entre ellas y obtener una comprensión más completa de su interacción, consultar Anexo I (sección 2.5). Además, buscamos otros factores que pudieran estar afectando los resultados y que no estuvieran claros en los datos iniciales para obtener una visión más precisa del panorama general.

Al examinar las existencias, se encontró que la mayoría de los datos estaban dentro del rango de 100 mil a 500 mil ovinos, aunque había quince datos atípicos que superaban el límite superior. Al aplicar la función «swek()» en el análisis estadístico previo, se determinó que la distribución de las existencias era asimétrica positiva, lo que se confirmó al observar el histograma y el diagrama de caja correspondientes. Sin embargo, al graficar solo las observaciones de las existencias en un diagrama de dispersión y un diagrama de líneas, no se encontró un patrón significativo. Posteriormente, se intentó establecer una relación entre las existencias de ovinos y el año, así como con los diferentes departamentos, mediante la creación de un diagrama de dispersión, un diagrama de líneas y un diagrama de barras. Se observó que a medida que un departamento tenía más ovinos, su producción se volvía más variable a lo largo de los años. Los departamentos de Rawson y Futaleufú se destacaron por su estabilidad y por tener la menor cantidad de ovinos, mientras que Río Senguer y Tehuelches se destacaron por tener la mayor cantidad de ovinos y, por ende, más variabilidad en la producción ovina a lo largo de los años. También se intentó establecer una relación entre las existencias y la cantidad de lluvia mediante diagramas de dispersión y de líneas, pero no se encontró un patrón claro. En la mayoría de los casos, las existencias se mantuvieron constantes a medida que aumentaba la cantidad de lluvia, como se puede apreciar en la figura 4.4.10. Aunque se observaron algunos picos pronunciados en ciertos departamentos, como Río Senguer, Paso de Indios y Rawson, en general, las líneas tendieron a permanecer constantes a medida que aumentaba la lluvia.

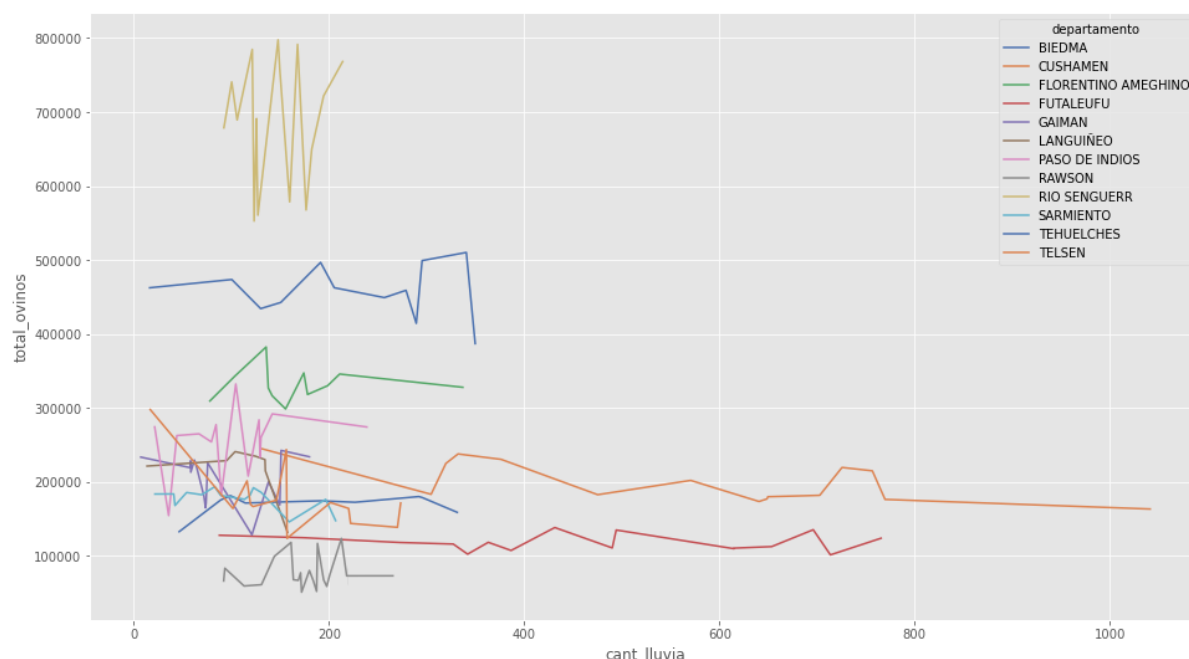


Figura 4.4.10. Gráfico multilíneas para existencia de departamentos

Al analizar el resto de las variables, se encontraron valores atípicos en los datos de ovejas (15), corderos (16) y kilos de lana (22), que superaban los límites superiores de la distribución. La mayoría de los datos de ovejas se encontraban en el rango de 50 mil a 150 mil, mientras que la mayoría de los datos de corderos se encontraban en el rango de 25 mil a 80 mil. En cuanto a los kilos de lana, la mayoría de los datos se encontraban en el rango de media tonelada a tonelada y media. Al utilizar la función «swek()» en el análisis estadístico previo, se determinó que las distribuciones de las ovejas, corderos y kilos de lana eran asimétricas positivas, lo cual se corroboró al examinar los histogramas y los diagramas de caja correspondientes. Los diagramas de dispersión y líneas correspondientes a las observaciones exclusivamente de ovejas, corderos y kilos de lana no arrojaron un patrón significativo. Después, se intentó establecer una relación entre las variables de interés (ovejas, corderos, kilos de lana) y el año y los diferentes departamentos, a través de diagramas de dispersión, líneas y barras. Se pudo observar que a medida que la cantidad de ovejas, corderos y kilos de lana aumentaba en un departamento, la producción se volvía más variable a lo largo de los años. En cuanto a los departamentos, Rawson y Futaleufú presentaron la menor cantidad de animales y mayor estabilidad en los tres casos analizados, mientras que Río Senguer y Tehuelches se destacaron por tener la mayor cantidad de animales y, por ende, mayor variabilidad en la producción ovina a lo largo de los años. Por último, se intentó establecer una relación entre las variables de interés y la cantidad de lluvia a través de diagramas de dispersión y de líneas, sin embargo, no se encontró un patrón claro. En su mayoría, las observaciones de ovejas, corderos y kilos de lana se mantuvieron constantes a medida que aumentaba la cantidad de lluvia, como se muestra en las figuras 4.4.11, 4.4.12 y 4.4.13 respectivamente. Aunque hubo algunos picos pronunciados en ciertos departamentos como Río Senguer, Paso de Indios y Rawson, en general, las líneas tendieron a permanecer constantes a medida que aumentaba la lluvia.

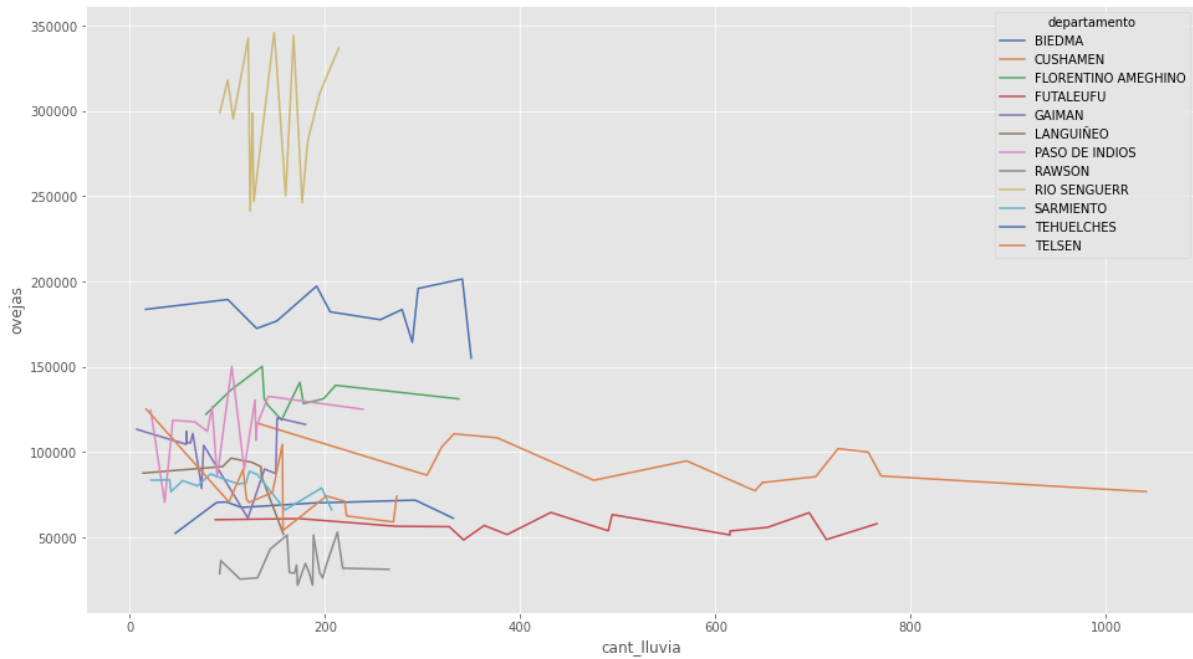


Figura 4.4.11. Gráfico multilíneas para ovejas de departamentos

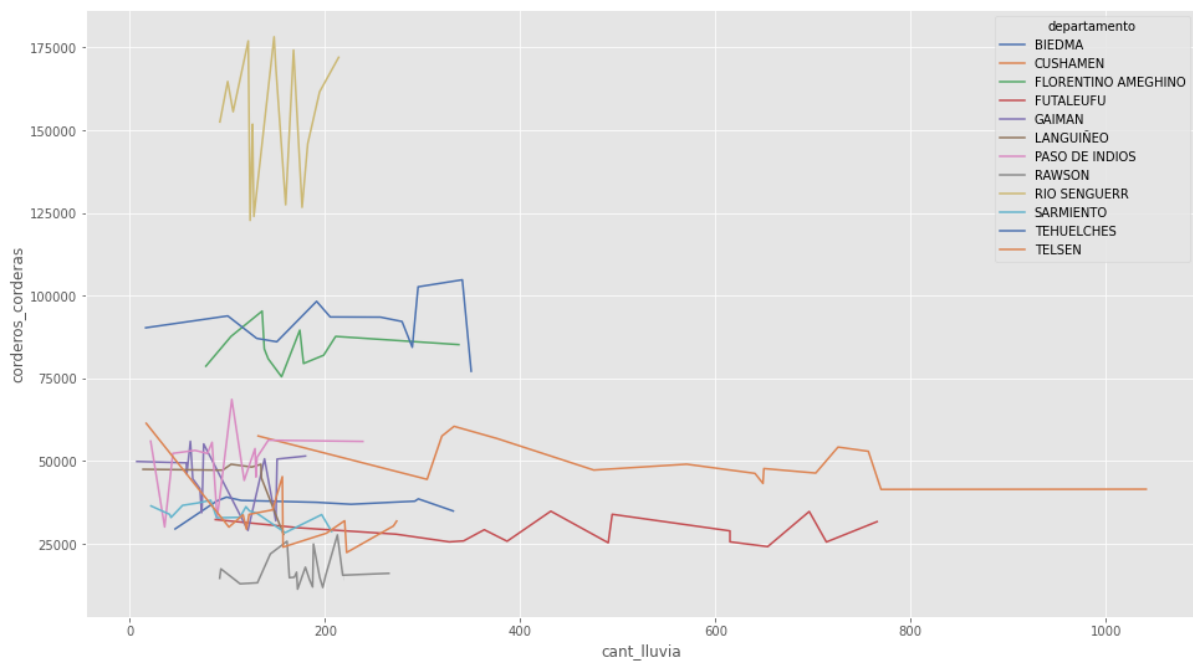


Figura 4.4.12. Gráfico multilíneas para corderos de departamentos

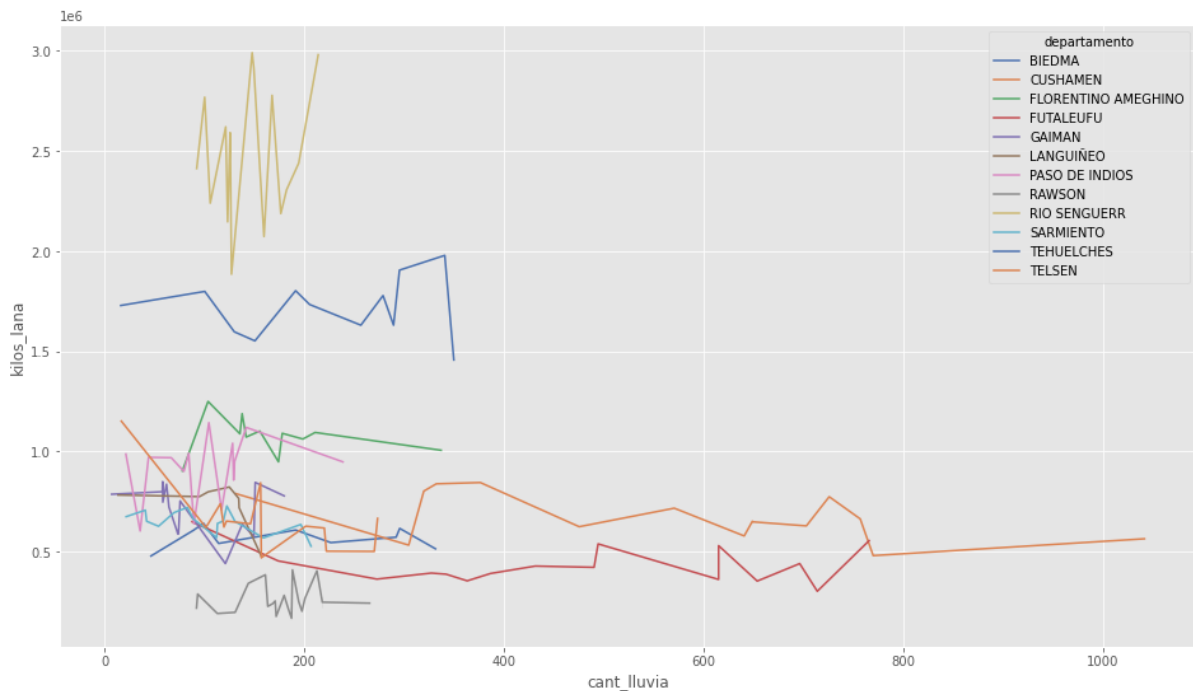


Figura 4.4.13. Gráfico multilíneas para kilos de lana de departamentos

Mientras realizábamos los gráficos, notamos que las variables de existencias ovinas, cantidad de ovejas, corderos y kilos de lana esquilados presentaban patrones de forma similares, variando únicamente en su escala. Esto sugería una fuerte correlación entre estas variables, tal como lo indicaba la matriz de correlación que habíamos obtenido previamente.

4.5 Entrenamiento de modelos

A continuación, presentaremos un resumen del entrenamiento llevado a cabo para cada uno de los modelos utilizados en la aplicación web.

Modelos regresión lineal

Inicialmente, realizamos pruebas con una regresión simple para determinar si era posible predecir la cantidad de kilos de lana que un productor pudiera obtener a partir de la variable de lluvia, y también para evaluar la posibilidad de predecir la cantidad de corderos que podrían nacer únicamente en función de la lluvia. A pesar de que nuestros análisis previos indicaban que estas variables no estaban relacionadas, quisimos comprobarlo y, lamentablemente, los resultados obtenidos fueron desastrosos.

Después de la prueba fallida con la regresión simple basada en la variable de lluvia, nos enfocamos en determinar qué otra variable podría ayudar en la estimación de los kilogramos de lana y que otra variable podría contribuir en la estimación de corderos. Según nuestro análisis previo, identificamos que el número total de ovinos tenía una relación significativa con la producción de lana, por lo que decidimos utilizarlo como segunda variable en nuestro modelo predictivo. En el caso de los corderos, observamos que la variable más influyente era el número de ovejas. Con este enfoque, buscamos predecir la cantidad de kilogramos de lana tomando en cuenta tanto el acumulado de lluvia en milímetros en la zona como el número de

ovinos que poseía el productor, y predecir la cantidad de corderos que podrían nacer teniendo en cuenta la lluvia acumulada y el número de ovejas presentes. Con esta idea en mente, tomamos los tres datasets previamente depurados y analizados para crear modelos de regresión lineal múltiple. Para el entrenamiento de los modelos, utilizamos los datos en su formato original sin modificar. Sin embargo, en el caso del dataset 2, donde encontramos valores muy distintos, además de utilizar los datos originales, también decidimos escalarlos para ver si esto mejoraba el proceso de entrenamiento.

Durante nuestros entrenamientos, dividimos el dataset en datos de prueba y entrenamiento utilizando el 20% y el 80%, respectivamente. Luego, utilizamos un objeto k-fold para realizar una validación cruzada en el conjunto de datos, dividiéndolo en cinco partes iguales. Este enfoque consistió en iterar cinco veces los datos, separando un porcentaje de muestras en cada iteración para utilizarlas como conjunto de prueba, mientras que las restantes se usaron para entrenar el modelo. Al final, medimos el rendimiento y obtuvimos cinco resultados para comparar. Esta técnica, descrita en el capítulo 2 de nuestra tesina, nos permitió determinar si el modelo tenía una buena capacidad de predicción al verificar si las cinco estimaciones resultantes eran similares y si la media de ellas era alta.

Luego de realizar validación cruzada, utilizamos el modelo entrenado para hacer predicciones sobre los datos de prueba y evaluar su rendimiento con diferentes métricas. Dado que estábamos trabajando con modelos de regresión, prestamos especial atención a tres métricas específicas: la varianza explicada, el coeficiente de determinación (R^2) y el R^2 ajustado. En los tres datasets, los modelos de predicción de lana y corderos tuvieron un desempeño muy bueno, ya que los valores de estas métricas oscilaban entre el 70% y el 99%. Al finalizar la evaluación, representamos los modelos en un gráfico 3D para determinar el plano que mejor se ajustaba a los datos, y seguidamente, en un gráfico 2D, donde comparábamos la predicción con los valores reales. Para una descripción completa del proceso de entrenamiento, se recomienda revisar las secciones 3.1 a 3.13 del Anexo I.

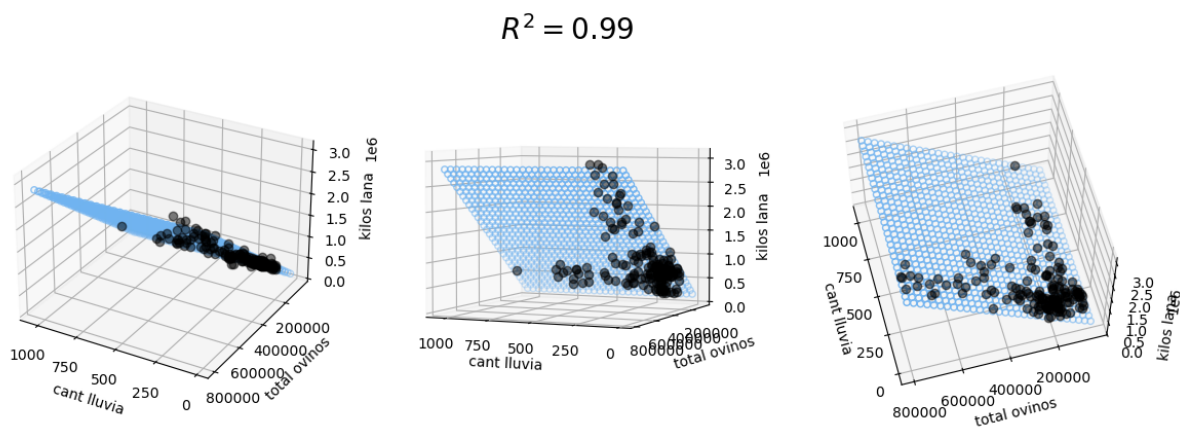


Figura 4.5.1. Gráfico 3D, predicción de lana para dataset 3.

$$R^2 = 0.98$$

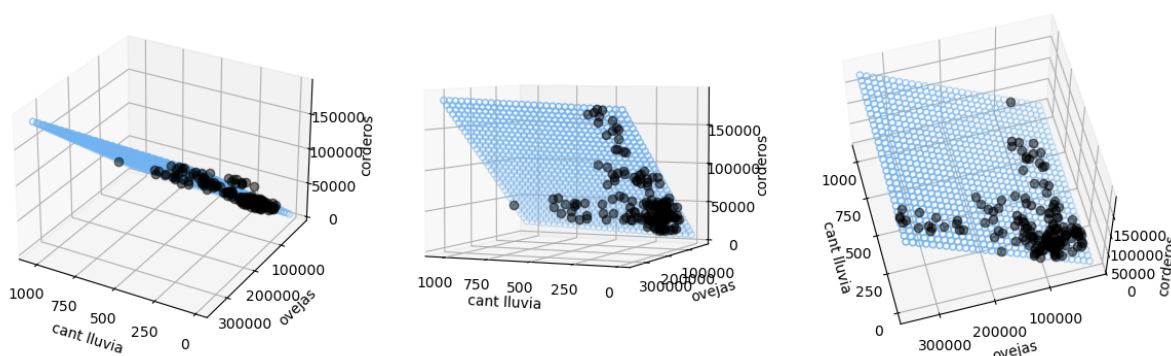


Figura 4.5.2. Gráfico 3D, predicción de corderos para dataset 3.

Los gráficos presentados muestran diversas perspectivas de la relación entre la cantidad de lluvia, la cantidad de ovejas y corderos en la figura 4.5.2, y la relación entre la cantidad de lluvia, la cantidad de ovinos y los kilogramos de lana en la figura 4.5.1. Cada gráfico presenta tres perspectivas diferentes: una vista desde arriba, otra vista de frente y la última observada desde el eje de las ovejas o el total de ovinos. En los gráficos, podemos observar cómo los datos reales representados por los puntos negros se van ajustando al plano. No obstante, el ajuste no es perfecto, lo cual es un buen indicativo y concuerda con las métricas obtenidas.

Sin embargo, para no basarnos únicamente en las métricas para determinar cuál modelo funcionaba mejor, lo pusimos a prueba de otra manera. Recordemos que teníamos tres datasets: el Dataset 1 y el Dataset 2 eran similares, con la diferencia de que el Dataset 2 contenía además los datos del campo de El Mirabueno. Por su parte, el Dataset 3 era distinto a los anteriores, ya que contenía todos los datos de los departamentos únicamente actualizados y/o ajustados con los valores de las encuestas. Entonces lo que hicimos fue crear un sistema de puntuación que incentivaba o penalizaba al modelo en función de su capacidad de predicción. Por ejemplo, al evaluar el modelo 1, que fue entrenado con el Dataset 1, se le otorgaría una penalización debido a la posibilidad de que conociera los datos de entrenamiento. Cabe destacar que en este punto cada modelo solo conocía el 80% de su dataset de entrenamiento. Para considerar un modelo como bueno, se requería una precisión de predicción superior al 60%, lo que otorgaba un punto a favor. A continuación, procedimos a tomar valores completamente aleatorios de los conjuntos de datos y a poner a prueba cada uno de los modelos. Para ello, realizamos predicciones con estos datos aleatorios y medimos su puntuación. Además, comparamos visualmente las predicciones y, según los desempeños de los modelos, comenzamos a asignar puntajes positivos y negativos. En cuanto a la predicción de la cantidad de lana y de corderos, el modelo que mejor se desempeñó en ambos casos fue el entrenado con el Dataset 3, ya que logró una mejor generalización de los datos para todos los conjuntos de datos utilizados en el estudio.

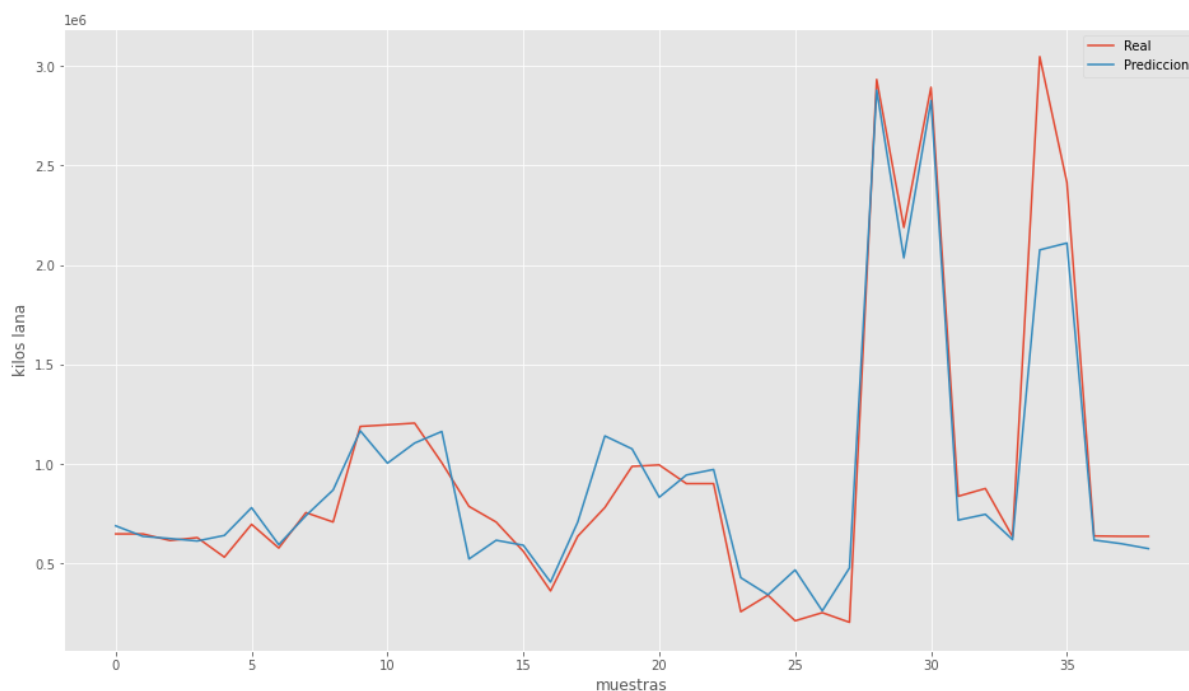


Figura 4.5.3. Mejor modelo de predicción de lana.

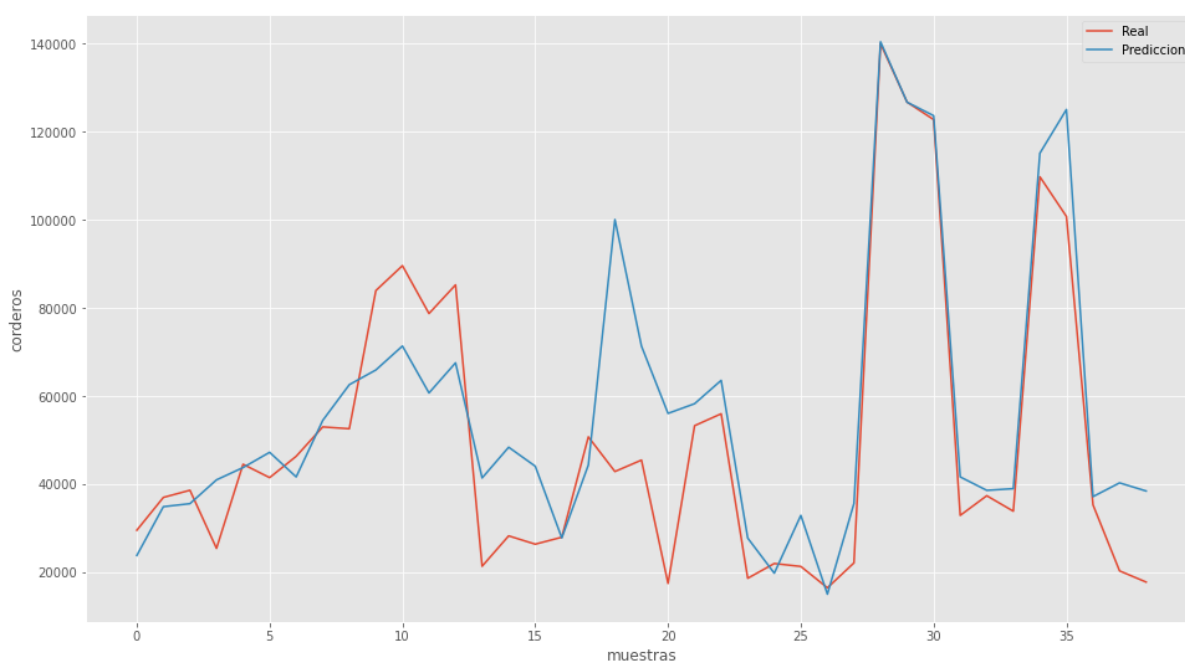


Figura 4.5.4. Mejor modelo de predicción de corderos.

Mediante el entrenamiento de estos modelos de regresión lineal múltiple, pudimos descubrir las «relaciones complejas» entre las variables de lluvia y producción que no eran evidentes en el análisis inicial de los datasets. Y estos modelos fueron altamente precisos en la predicción de los resultados. La figura 4.5.3 presenta el mejor modelo para la predicción de lanas. La línea roja representa los datos reales y la línea azul representa los datos de predicción. Podemos observar que ambas líneas tienen una similitud bastante elevada, aunque no son exactamente iguales. En el eje x se encuentran las muestras y en el eje y se representa la

cantidad de lana. Por otro lado, en la figura 4.5.4 podemos ver el mejor modelo para la predicción de corderos. En este caso, las líneas se ajustan un poco menos que en el gráfico anterior, aunque aún son muy similares. En ambos casos, se puede afirmar que los modelos tendrán una capacidad de predicción elevada. Durante el proceso de entrenamiento, en ciertos momentos, se planteó la posibilidad de que nuestros modelos estuvieran sobreajustados. Para abordar esta cuestión, se probó con las técnicas de regresión de Lasso y regresión de Ridge para regularizar nuestros modelos. Los resultados obtenidos fueron similares a los obtenidos en el entrenamiento sin regularización, lo cual sugería que nuestros modelos podrían generalizar bien.

Modelos de bosques aleatorios

Al intentar predecir las características de la lana, incluyendo la finura y el rinde, inicialmente utilizamos un modelo de regresión lineal simple basado en variables meteorológicas como la lluvia, la temperatura y la humedad. Como habíamos previsto en función de nuestro análisis previo de datos, los resultados no fueron satisfactorios ya que no se encontró relación alguna entre las características de la lana y los datos meteorológicos. Por lo tanto, decidimos probar con modelos de regresión lineal múltiple para la predicción de la lana y corderos, lo que permitió encontrar relaciones significativas entre las variables. Sin embargo, al tratar de predecir el rinde y la finura de la lana, a pesar de haber combinado varias variables, no pudimos lograr una precisión del modelo superior al 35%. Por lo tanto, decidimos explorar otras opciones y buscar algoritmos que pudieran ayudarnos a predecir estas variables de manera más efectiva. Fue entonces cuando descubrimos los Bosques Aleatorios, los cuales son capaces de obtener una precisión aceptable en casos donde las relaciones entre las variables no son lineales. Decidimos utilizar el Dataset 2, seleccionando únicamente los datos correspondientes a la lana del Mirabueno, es decir, los datos de finura, rinde y fecha. Nos enfocamos en tres métricas principales: el error medio absoluto, el error cuadrático medio y la raíz cuadrada del error cuadrático medio. Los resultados obtenidos fueron aceptables, ya que en el caso de la finura de la lana, el modelo presentó un error cuadrático medio de sólo 0,05 micras, mientras que en el caso del rinde el error fue de 4,16. Para ver el proceso consultar las secciones 3.14 y 3.15 del Anexo I.

Otros modelos probados

Durante el desarrollo de esta tesina además de los modelos mencionados, se realizaron pruebas con otros modelos, entre los cuales destacan los modelos basados en árboles y el algoritmo KNN. Si bien no se encontró un patrón 100% claro en los datos, se identificaron casos en los que parecía haber cierta agrupación de los mismos. En este sentido, se evaluó el algoritmo KNN, pero no se obtuvieron buenos resultados. Posteriormente, se entrenaron modelos de regresión basados en árboles, un ejemplo de este proceso de entrenamiento puede encontrarse en la sección 3.16 del Anexo I. Para los árboles, seleccionamos variables para que, aunque no presentaban una correlación alta, mostraban una relación significativa según la matriz de correlación. Utilizamos estas variables como predictores del árbol y establecimos una profundidad inicial aleatoria entre 3 y 6. Después de entrenar el modelo inicial, lo diagramamos, validamos la importancia de los predictores y realizamos un pruning. Para el

pruning, realizamos una búsqueda Grid para el valor óptimo del parámetro «ccp_alpha», que indicaba cuánto se tenía que podar el árbol. Luego entrenamos un segundo modelo con el árbol podado y comparamos los resultados del modelo inicial sin podado y del modelo final con podado. Aunque obtuvimos resultados prometedores en algunos casos, optamos por utilizar modelos de regresión lineal múltiple debido a su mayor facilidad de interpretación y menor tendencia al sobreajuste en comparación con los modelos basados en árboles.

Capítulo 5: Desarrollo realizado

En este capítulo se discutirá el desarrollo llevado a cabo como parte de la presente tesina, que consiste en la realización de un sistema para apoyar la gestión de un campo de producción ganadera ovina, como se mencionó en el primer capítulo.

El objetivo principal del sistema es contribuir a las nuevas tecnologías ganaderas, proporcionando información climática y productiva resumida que permita a los productores tomar decisiones más informadas y anticipar problemas futuros a fin de prevenirlos.

El sistema desarrollado procesa una amplia gama de información productiva y climática, incluyendo variables como la temperatura, el viento, la humedad y la cantidad de lluvia. En particular, el conocimiento detallado de la cantidad de lluvia y el momento en que llueve puede ser un dato de gran importancia para el productor en zonas donde no predominan las lluvias. Además, al contar con información meteorológica y de producción organizada en el tiempo, el productor puede analizar el comportamiento de sus variables de producción en años anteriores y realizar comparaciones entre diferentes años, como también correcciones cuando los datos presentan tendencias no deseadas. Esto le permite encontrar patrones, como un año en el que las precipitaciones acumularon la misma cantidad de milímetros o en el que se obtuvo el mayor kilaje de lana esquila, entre otros. Con esta información, el productor puede tomar decisiones más informadas y anticiparse a situaciones adversas, mejorando la gestión de su producción. En resumen, el sistema desarrollado proporciona al productor una herramienta valiosa para tomar decisiones más efectivas y mejorar su producción ganadera. Este conjunto de información le permitirá al productor conocer:

- ❖ Si es necesario adquirir suplementos y, en caso afirmativo, hacerlo con antelación a la temporada de mayor demanda, logrando así un precio más rentable.
- ❖ Si conviene o no soltar a los carneros en la majada.
- ❖ Si es necesario llamar a un veterinario que le sugiera medidas sanitarias a tomar con sus animales.
- ❖ Si es necesario reducir o aumentar la carga animal presente en el campo en un determinado momento.
- ❖ Si es conveniente adquirir una mayor cantidad de animales.

Con esta aplicación el productor puede tomar más y mejores decisiones teniendo en consideración las variaciones productivas y climáticas de la zona. Se presentará en detalle el caso de aplicación, incluyendo su arquitectura y la plataforma de software implementada, y se proporcionará una breve reseña de las tecnologías utilizadas en el desarrollo del sistema, destacando las decisiones de diseño más importantes tomadas oportunamente.

5.1 Aplicación Web

La aplicación web creada en Django es una herramienta valiosa para la gestión y análisis de datos en la producción ganadera. Entre sus principales funcionalidades, destacan la carga de datos productivos y climáticos en tiempo real, la visualización gráfica de estos datos para su rápida interpretación y la posibilidad de desarrollar y entrenar modelos de inteligencia artificial para realizar estimaciones sobre distintos datos de interés para el productor del campo. Para el funcionamiento de la plataforma son requeridos datos productivos y climáticos de un campo en particular, que son obtenidos de manera diaria tanto por el productor del campo como por tareas automatizadas. Estos datos se almacenan en una base de datos que generan registros históricos consultados para tomar decisiones estratégicas de desarrollo, como también son utilizados como pieza fundamental para el entrenamiento de algoritmos de machine learning que funcionan como herramientas de gestión productiva brindando previsibilidad a los productores.

Además, es posible automatizar la carga manual de datos productivos mediante la utilización de un bot de Telegram que posee funciones específicas que permiten tanto la carga de información como conocer el estado actual de todas las variables productivas de un campo en particular.

5.1.1 Modelo arquitectónico

El modelo arquitectónico de la aplicación puede verse como una arquitectura cliente-servidor. Un esquema cliente-servidor es un modelo de arquitectura de red en el que los recursos y servicios de una red de computadoras se distribuyen entre dos tipos de entidades: el cliente y el servidor. En el caso particular de la aplicación desarrollada podemos identificar los componentes o recursos como se muestran en la siguiente imagen:

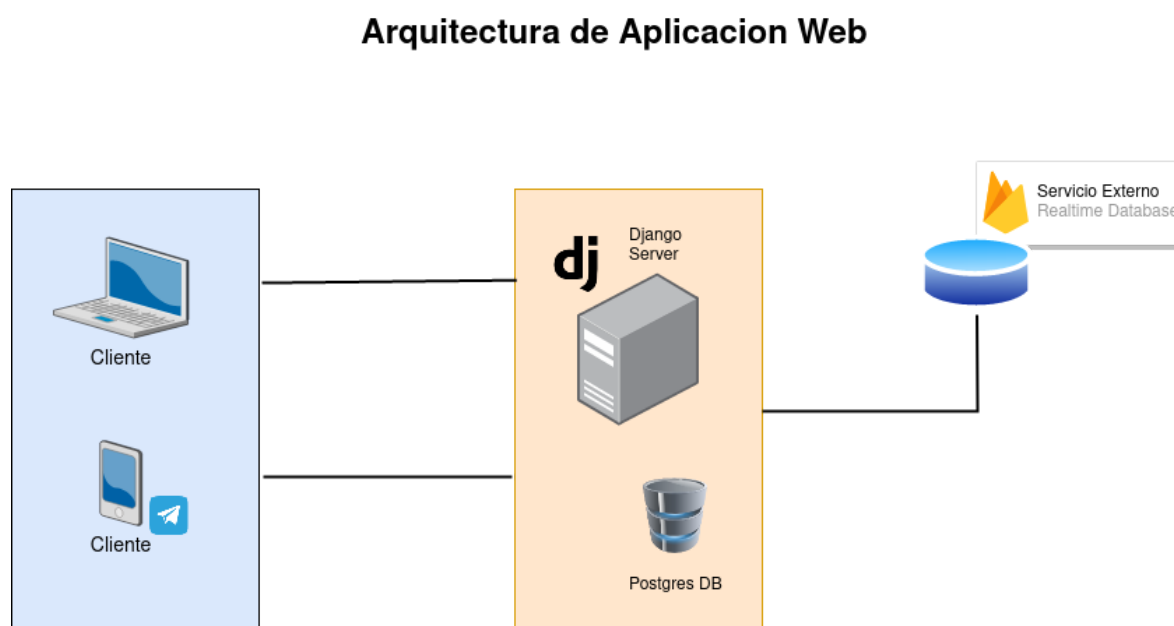


Figura 5.1.1. Modelo arquitectónico de la aplicación.

Cliente: Este componente es el punto de entrada para los usuarios que utilizan la aplicación, ya sea a través de una computadora o un dispositivo móvil. El cliente envía solicitudes al servidor para obtener datos o realizar acciones.

Servidor: En este caso, el servidor contiene la aplicación hecha en Django y es responsable de administrar y resolver las solicitudes del cliente. Incluye la lógica de negocio, la capa de acceso a datos y la comunicación con la base de datos y el servicio externo de base de datos en tiempo real. El servidor utiliza una base de datos en PostgreSQL para almacenar y recuperar información, y, a su vez, se comunica con un servicio de datos en tiempo real (denominado «Firebase», que se presentará en el apartado siguiente) para recibir actualizaciones de la información en tiempo real.

Servicio Externo: Es un servicio de almacenamiento de datos en tiempo real provisto por Firebase y utilizado por el servidor para recibir actualizaciones de datos en tiempo real. El servidor puede enviar actualizaciones a este servicio y estas se propagan automáticamente a los clientes conectados y viceversa.

5.2 Tecnologías utilizadas

Python

Python es un lenguaje de programación interpretado con una sintaxis limpia que favorece un código de programación legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, multiplataforma, usa tipado dinámico y conteo de referencias para la administración de memoria [109]. Algunas de sus características más importantes son [109]:

- ❖ Es de fácil lectura e interpretación debido a su sintaxis básica similar a la del inglés.
- ❖ Permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con otros lenguajes.
- ❖ Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- ❖ La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.
- ❖ Es utilizado para una gran variedad de desarrollos incluyendo: Data analytics y big data, Data mining, Data science, Inteligencia artificial, Blockchain, Machine learning, Juegos y gráficos 3D, Desarrollo web, entre otras.

Django

Django es un framework web de código abierto escrito en Python, para el desarrollo de aplicaciones web, el cual impulsa el desarrollo rápido y el diseño limpio y pragmático, permitiendo construir aplicaciones con menos código. Django se centra en automatizar todo

el desarrollo posible y se adhiere al principio *Don't Repeat Yourself (DRY)*. Basado en el patrón arquitectónico *Model View Controller (MVC)*, Django permite el desarrollo de aplicaciones web de modo simple y con completa integración a todo tipo de tecnologías web [110].

Firestore

Firestore es una plataforma digital diseñada para facilitar el desarrollo de aplicaciones web y móviles de calidad de una forma rápida y eficiente, con el objetivo de mejorar el rendimiento de las mismas a través de la implementación de sus distintos módulos que harán que la aplicación sea mucho más manejable, segura y fácil de utilizar para los usuarios [111].

Esta plataforma se encuentra alojada en la nube, y está disponible para Android, iOS y web entre otras plataformas de desarrollo. Entre sus principales módulos de trabajo se encuentran: Cloud Firestore, Realtime Database, Kit de Machine learning y Aprendizaje supervisado, Cloud Functions, Firestore Authentication, Hosting, Cloud Storage, entre otros [111].

Celery

Celery es una biblioteca de Python utilizada para crear tareas asincrónicas, programadas o en tiempo real. Permite a los desarrolladores crear tareas en segundo plano, procesamiento distribuido, colas de tareas, así como implementar patrones de mensajería y planificación de eventos [112]. Algunas de sus características principales son [113]:

- ❖ Tareas asincrónicas: Permite la creación y ejecución de tareas en segundo plano o de manera asíncrona, lo que significa que se pueden ejecutar tareas en el fondo mientras el usuario interactúa con la aplicación.
- ❖ Distribución de tareas: Celery permite distribuir tareas a través de múltiples procesos o incluso múltiples servidores, lo que ayuda a manejar grandes cargas de trabajo y aumentar la escalabilidad de una aplicación.
- ❖ Colas de tareas: Las tareas se pueden encolar para ser ejecutadas en un momento posterior. Esto ayuda a controlar el flujo de trabajo y priorizar tareas según su importancia.
- ❖ Planificación de tareas: Las tareas se pueden programar para que se ejecuten en momentos específicos, lo que ayuda a automatizar procesos y eventos en una aplicación.
- ❖ Integración de diferentes sistemas de mensajería: Celery se puede integrar con diferentes sistemas de mensajería como RabbitMQ, Redis, y Amazon SQS, lo que permite la creación de aplicaciones más flexibles y escalables.

Además, el desarrollo también está complementado por las siguientes librerías:

- ❖ Telegram Login: La librería telegram-login de Python es una herramienta que permite integrar la autenticación de usuarios de Telegram en una aplicación hecha con python a través de la API oficial de Telegram. Con esta librería, es posible implementar un sistema de login seguro y personalizado que utiliza las credenciales de Telegram del usuario para autenticarlo en la aplicación. Además, esta librería también ofrece funciones para administrar sesiones de usuario y manejar errores de autenticación. Esta librería es muy útil para los desarrolladores que deseen integrar la autenticación

de usuarios de Telegram en aplicaciones hechas con python de forma fácil y rápida [114].

- ❖ Redis: Es una base de datos de código abierto en memoria, utilizada principalmente como almacén de estructuras de datos en caché y como intermediario de mensajes en tiempo real. Redis se caracteriza por ser una base de datos extremadamente rápida y escalable, lo que la hace popular en aplicaciones web que requieren alta velocidad y rendimiento. Redis admite una variedad de estructuras de datos, como cadenas, listas, conjuntos, mapas ordenados y otros tipos de datos complejos. También proporciona características avanzadas como transacciones atómicas, scripting, replicación y clústeres, que permiten una mayor escalabilidad y confiabilidad [115].
- ❖ Memcached: Es una herramienta de almacenamiento de caché en memoria de código abierto que se utiliza principalmente para acelerar aplicaciones web mediante el almacenamiento temporal de datos en memoria RAM en lugar de acceder a ellos repetidamente desde una base de datos u otro origen de almacenamiento permanente. Sus principales ventajas son su capacidad para reducir la carga en la base de datos, mejorar la escalabilidad y reducir los tiempos de latencia en aplicaciones web de alta demanda [116].
- ❖ Bootstrap: Es un framework CSS de código abierto que favorece el desarrollo web de un modo más sencillo y rápido. Incluye plantillas de diseño basadas en HTML y CSS con la que es posible modificar tipografías, formularios, botones, tablas, navegaciones, menús desplegables, etc. También existe la posibilidad de utilizar extensiones de Javascript adicionales. Fue desarrollado inicialmente por Twitter en 2011 y permite crear interfaces de usuario limpias y compatibles con todo tipo de dispositivos. Entre las ventajas que tiene Bootstrap es que favorece el diseño responsivo, el cual se utiliza para mejorar la experiencia de los usuarios en el sitio web y en consecuencia el posicionamiento. Algunas de sus características más importantes son [117]:
 - Es de código abierto, y todo su código actualizado se encuentra en un repositorio de Github.
 - Está mantenido y actualizado por Twitter.
 - Es compatible con la mayoría de navegadores (Chrome, Safari, Mozilla, etc.).
 - Dispone de gran cantidad de documentación, tanto en su portal como en páginas web especializadas.
 - Utiliza componentes vitales para los desarrolladores (HTML5, CSS3, jQuery o GitHub, entre otros).
 - Incluye Grid system para maquetar por columnas.
 - Utiliza Less: Un lenguaje de las hojas de estilo CSS preparado para enriquecer los estilos de la web.
- ❖ Javascript: Es un lenguaje de programación que los desarrolladores utilizan para hacer páginas web interactivas. Desde actualizar fuentes de redes sociales a mostrar animaciones y mapas interactivos, las funciones de JavaScript pueden mejorar la experiencia del usuario de un sitio web. Como lenguaje de scripting del lado del servidor, se trata de una de las principales tecnologías de la *World Wide Web (Web)*. Es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos,

imperativo, débilmente tipado y dinámico. Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas [118].

- ❖ JQuery: Es una librería desarrollada en 2006 por John Resig que permite añadir una capa de interacción AJAX entre la web y las aplicaciones que desarrollemos controlando eventos, creando animaciones y diferentes efectos para enriquecer la experiencia de usuario. Cuenta con un diseño que facilita la navegación a través de un documento HTML y permite seleccionar elementos DOM proporcionando a los desarrolladores de aplicaciones web complementos que agilizan el desarrollo de proyectos. Esto permite a los desarrolladores centrarse en lo importante y crear abstracciones para interacción y animación de bajo nivel, efectos avanzados y widgets temáticos de alto nivel sin invertir tiempo en desarrollar complejos algoritmos y métodos que los controlen desde cero y generando menos código que las aplicaciones hechas con JS puro [119].

5.3 Funcionamiento de la Aplicación

La aplicación web es un sistema completo que incluye una variedad de pantallas y funcionalidades que ofrecen una experiencia de gestión integral del campo del usuario. La plataforma cuenta con un proceso de inicio de sesión y registro como punto de partida inicial de la cual se desprenden una serie de pantallas conteniendo las funcionalidades principales.

- ❖ La sección «Pantalla principal» ofrece una visión general de la aplicación.
- ❖ La sección «Mi campo» permite al usuario ingresar los datos productivos y climáticos diarios e históricos.
- ❖ La sección «Datos cargados» muestra un resumen gráfico de todos los datos ingresados y almacenados en la base de datos.
- ❖ La sección «Perfil» permite al usuario gestionar su información personal y del campo así como ajustar las preferencias de la aplicación.
- ❖ La sección «Estimaciones Machine Learning» ofrece la posibilidad de interactuar con modelos de machine learning pre entrenados para hacer estimaciones y mejorar la toma de decisiones del usuario.

Además, se ha incluido un bot de Telegram para automatizar y facilitar la carga de datos productivos aumentando la eficiencia del sistema en general.

A continuación se dará una descripción detallada del conjunto de pantallas y funcionalidades de la aplicación desarrollada.

5.3.1 Registro e Inicio de sesión

En esta pantalla se realiza el registro de nuevos usuarios dentro de la aplicación. Es el punto de partida inicial para comenzar a utilizar el sitio.



Nuevo usuario

Datos Personales

Nombre

Apellido

Nombre de usuario Email

Contraseña Confirmación contraseña

Datos del Campo

Nombre del Campo Cantidad de Hectareas

Latitud Longitud [ver mapa](#)

[Registrarse](#)

[Ya tiene una cuenta? Inicie Sesión!](#)

Figura 5.3.1. Registro de usuario de la aplicación.

Los datos requeridos para el registro de nuevos usuarios son los siguientes:

- ❖ Datos personales como nombre, apellido, nombre de usuario y correo electrónico, los cuales permitirán autenticar unívocamente a un usuario dentro de la aplicación.
- ❖ Datos del campo como el nombre del campo, que permitirá identificar al campo por su nombre o alias, cantidad de hectáreas que serán utilizadas para mostrar información relacionada a la ocupación animal del campo y por último la latitud y longitud del campo que se utilizarán no solo para conocer la ubicación geográfica específica, sino también para brindar información climática actualizada.

Una vez realizado el registro en el sistema, el usuario es redireccionado a la pantalla de inicio de sesión para luego ingresar en el sistema.



Bienvenido!

Usuario:

Contraseña: [👁](#)

[Ingresar](#)

[Crea una cuenta nueva!](#)

Figura 5.3.2. Inicio de sesión de la aplicación.

5.3.2 Pantalla Principal

Una vez iniciada la sesión, el usuario es redirigido a la pantalla principal del sistema.

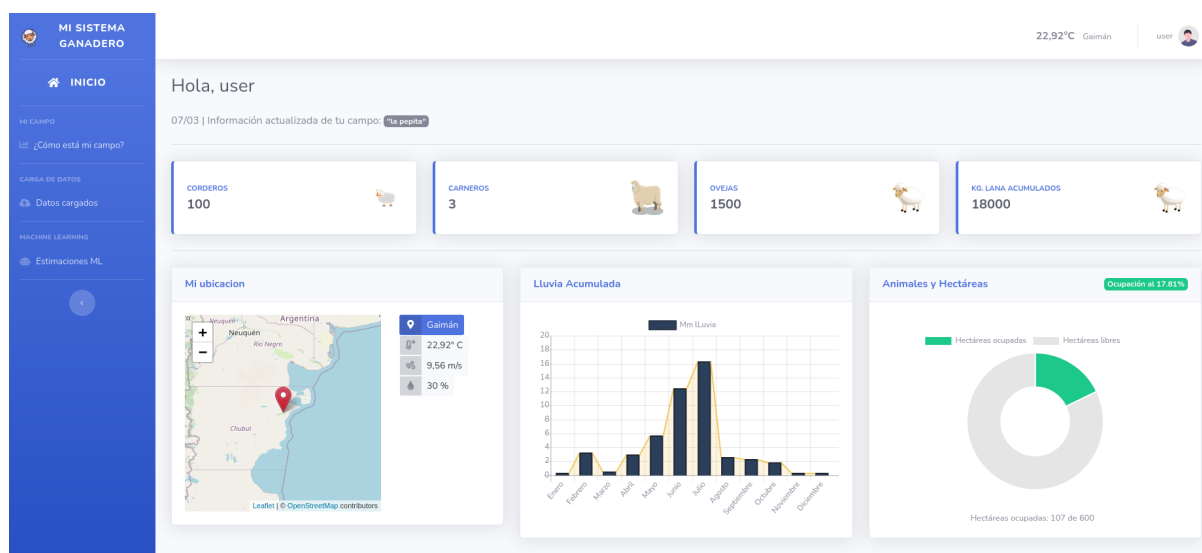


Figura 5.3.3. Pantalla principal.

Dentro de esta pantalla es posible visualizar todas las opciones de utilización de la aplicación. En la barra lateral se observan todas las opciones disponibles de consulta en el sistema, en la navbar el clima actual del campo y una opción para acceder a la configuración de perfil; mientras que en la pantalla principal se observa un conjunto agrupado de las principales variables de interés del campo con sus datos y su respectivo gráfico de visualización. En la primer fila de datos tenemos:

- ❖ Cantidad de ovejas actuales: Indica la cantidad de ovejas actualmente presentes en el campo.
- ❖ Cantidad de corderos actuales: Indica la cantidad de corderos actualmente presentes en el campo.
- ❖ Cantidad de carneros actuales: Indica la cantidad de carneros actualmente presentes en el campo.
- ❖ Kilos de lana acumulados: Indica la cantidad de kilos de lana obtenidos de la esquila del año en curso. Este valor es obtenido de la sumatoria de kilogramos de lana obtenidos durante todos los meses del año.

Además, de manera gráfica, se muestran:

- ❖ Mapa de ubicación actual del campo: Un mapa que muestra el punto geográfico específico donde se ubica el campo acompañado de las variables climáticas de ese momento como son la temperatura, velocidad del viento, humedad y la localidad.
- ❖ Lluvia acumulada: Dado que la lluvia es una de las variables climáticas más importantes para el productor, se muestra como central un gráfico de barras indicando los milímetros de lluvia acumulados en los meses transcurridos del año actual.
- ❖ Ocupación del campo: Es un gráfico de torta que indica, con distintas gamas de colores, el porcentaje de hectáreas ocupadas por animales y el porcentaje de hectáreas libres del campo. El rango de colores varía según el porcentaje de ocupación que puede ser:
 - Verde: de 0 a 25% de ocupación de hectáreas.

- Azul: de 25 a 50% de ocupación de hectáreas.
- Amarillo: de 50 a 75% de ocupación de hectáreas.
- Rojo: porcentaje de ocupación de hectáreas mayor a 75%.

Todas estas variables agrupadas indican, en conjunto, el estado actual del campo teniendo en cuenta los datos que pueden tener mayor relevancia para el productor.

Aquí la información meteorológica se obtiene consultando un servicio web externo que proporciona datos climáticos específicos según la ubicación geográfica del campo en formato de API REST [120]. Esta información una vez obtenida es almacenada en caché y renovada cada cierto tiempo para evitar los retardos propios del consumo de servicios externos por parte de la aplicación.

5.3.3 Pantalla Mi Campo

En esta pantalla se proporciona un resumen detallado de todas las variables climáticas y productivas del campo, organizadas en una serie de filtros y gráficos que permiten al productor determinar y conocer el mejor año del campo según diversos criterios de interés. Su objetivo es dar al productor una respuesta a la pregunta: ¿Cuál ha sido mi mejor año en términos de «alguna variable»?

Mediante esta funcionalidad, es posible determinar, por ejemplo, el mejor año en términos de la cantidad total de animales que tuvo el campo, el año con la mayor cantidad de lluvia acumulada durante los meses transcurridos o el año donde se obtuvo la mayor cantidad de lana esquilada. En resumen, esta pantalla ofrece una herramienta valiosa para el productor que desea establecer comparaciones entre diferentes períodos de tiempo, teniendo en cuenta las variables productivas específicas de cada momento. A través de esta pantalla, se puede seleccionar información histórica del campo mediante una serie de filtros que son:

MEJOR AÑO SEGÚN:
Mejor Rinde
Mejor Finura
Mayor cantidad de mm de lluvia
Mayor temperatura en verano
Kg de Lana Producida
Kg de Carne Producida
Cantidad de Ovejas

Figura 5.3.4. Filtros de consulta disponibles.

- ❖ **Mejor rinde:** Permite seleccionar el año en que se ha obtenido el mayor rinde de lana durante la esquila, teniendo en cuenta el rinde obtenido de lana durante cada período de esquila en un año específico.

- ❖ Mejor finura: Permite seleccionar el año en que se ha obtenido la lana más fina durante la esquila, teniendo en cuenta la revisión de finura de lana obtenida en cada período de esquila que es llevada a cabo durante el año.
- ❖ Mayor cantidad de mm de lluvia: Permite seleccionar el año en que se ha registrado el promedio mensual de lluvia más alto durante el periodo correspondiente.
- ❖ Mayor temperatura en verano: Permite seleccionar el año en que se ha registrado la temperatura más alta durante los meses de enero, febrero y marzo.
- ❖ Kilogramos de lana producidos: Permite seleccionar el año en que se ha obtenido la mayor cantidad de kilos de lana durante la esquila, teniendo en cuenta la sumatoria de kilos de lana obtenida en cada período de esquila que es llevada a cabo durante el año.
- ❖ Kilogramos de carne producidos: Permite seleccionar el año en que se ha producido la mayor cantidad de carne en el campo.
- ❖ Cantidad de ovejas: Permite seleccionar el año en que se ha alojado la mayor cantidad de animales en la totalidad de hectáreas del campo.

Cada uno de estos filtros permite obtener información detallada sobre el rendimiento, la productividad y las condiciones climáticas del campo a lo largo de un año determinado, presentándose en forma de gráficos y tablas como se observan a continuación:

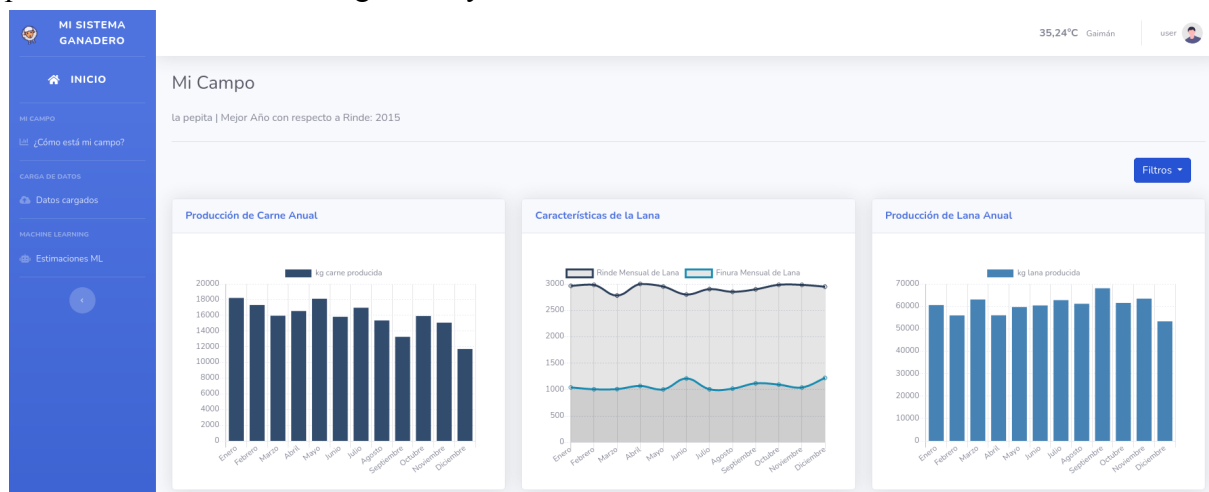


Figura 5.3.5. Pantalla mi campo con gráficos de producción.



Figura 5.3.6. Pantalla mi campo con gráficos climáticos.

Luego de la selección de algún filtro en particular se pueden visualizar diferentes gráficos y tablas cuya información es la siguiente:

En una primera fila de gráficos se pueden observar:

- ❖ Un diagrama de barras que muestra la cantidad de kilos de carne obtenidos durante cada mes del año. Este gráfico permite visualizar la evolución de la producción de carne a lo largo del tiempo y comparar los valores entre distintos meses.
- ❖ Un gráfico de líneas que representa la relación entre el rinde y la finura de la lana obtenida en cada período de esquila de un año particular. Este gráfico muestra cómo varían la finura y el rinde obtenidos de la lana y establecer una relación visual teniendo en cuenta la cantidad de animales presentes en ese año, la raza, etc.
- ❖ Por último, un diagrama de barras que indica la cantidad de kilos de lana obtenidos de la esquila durante cada mes del año. Este gráfico permite observar la producción de lana a lo largo del tiempo y comparar los diferentes meses.

Luego es posible observar distintos gráficos en formato carrusel con cada mes del año seleccionado, indicando en detalle cuáles han sido los valores climáticos y productivos del campo durante el transcurso esos meses. Para ello se visualizan los siguientes gráficos:

- ❖ Un gráfico de líneas que representa la temperatura máxima y mínima registrada en cada mes del año, permitiendo visualizar la variación mensual de esas temperaturas y comparar los diferentes meses.
- ❖ Un diagrama de barras que representa la cantidad de milímetros de lluvia acumulados durante cada mes del año que permite observar los patrones de lluvia a lo largo de los meses y determinar cuáles son los meses con mayor o menor cantidad de precipitaciones.

- ❖ Por último, se muestra diagrama de torta que indica la cantidad de ovejas, corderos y carneros presentes en el campo en un mes específico, acompañado de una tabla que muestra información detallada sobre la temperatura máxima y mínima, la humedad y la velocidad del viento. Esta información permite analizar las condiciones climáticas del campo y determinar cómo afectaron a la producción de carne y lana.

En conjunto, estos gráficos proporcionan una visión completa de las variables más importantes del campo, permitiendo identificar patrones y tendencias a lo largo del tiempo.

5.3.4 Pantalla Datos Cargados

En esta pantalla, se presentan los datos diarios generados por el productor del campo y también todos los registros históricos que hayan sido almacenados desde el primer uso de la aplicación.

Además, esta pantalla permite la carga, borrado y actualización toda la información disponible del campo, funcionalidad que también es llevada a cabo por un bot de Telegram que será mencionada en apartados posteriores.

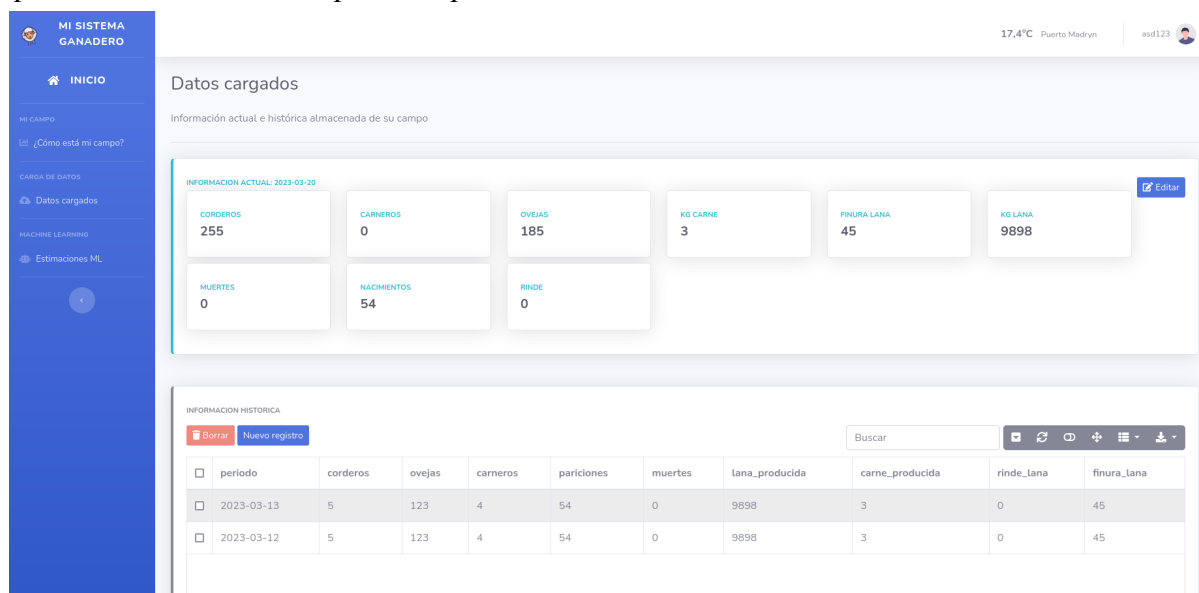


Figura 5.3.7. Pantalla datos cargados de la aplicación.

Los cambios en la información productiva del campo son recolectados diariamente por el productor, teniendo en cuenta los distintos momentos por los que puede pasar la producción ganadera de un campo, como por ejemplo: pariciones de corderos, ventas o compras de carneros, muertes de ovejas, etc. Estos cambios pueden ser cargados en la aplicación de manera manual o con un proceso de carga automatizado mediante comandos manuales o comandos de voz, de un bot de Telegram específico para el uso de la aplicación.

Al final de un día de trabajo, todos los cambios productivos que hayan sido reflejados durante el día se almacenan de manera histórica mediante un proceso de recolección automática que obtiene tanto los datos productivos, como la información climática del campo de un día particular. Además, es posible editar, borrar y cargar datos históricos que no hayan sido registrados de manera automática o por descuido u omisión del productor. De esta manera se

genera un registro histórico de todas las variables productivas y climáticas presentes en un campo en un momento dado.

La funcionalidad principal de esta pantalla es llevar un registro detallado y actualizado de los datos obtenidos a lo largo del tiempo, con el fin de poder analizarlos en distintos aspectos y tomar decisiones estratégicas en base a toda la información recolectada. Además de llevar un conteo específico de la producción a lo largo del tiempo y de brindar la posibilidad de registrar los cambios por los que atraviesa un campo, todo el conjunto de datos reunidos son posteriormente utilizados como fuente de alimentación para algoritmos de inteligencia artificial que pueden mejorar las capacidades productivas del campo, por esta razón es que esta pantalla es una pieza fundamental dentro de las opciones que se le brindan al productor para gestionar un campo productivo.

Por último, los datos históricos son organizados en formato de tabla de manera clara y consistente. Este formato permite visualizar la información relevante sobre la producción, como también filtrar y ordenar los datos de acuerdo a distintos criterios, facilitar la búsqueda de información específica y además contar con la posibilidad de exportar los datos a diferentes formatos que resulta de gran utilidad para su análisis y posterior uso en informes y reportes.

En resumen, esta pantalla es una herramienta esencial para el registro y análisis de todos los datos producidos en el campo de la cual se desprenden las principales funcionalidades brindadas al productor, cuya relevancia aumenta a medida que transcurre el tiempo debido al crecimiento de las bases de información generadas diariamente.

5.3.5 Pantalla Perfil

En esta pantalla se muestra toda la información relacionada a la configuración del usuario y su campo. Aquí se pueden editar los datos personales del usuario como el nombre y apellido, y los datos del campo como el nombre, la cantidad de hectáreas y los puntos geográficos donde se encuentra ubicado. Además de permitir modificar los datos del usuario y de su campo, es posible vincular una cuenta de Telegram asociada al usuario para interactuar con la carga de datos de la aplicación a través de un bot, el cual fue desarrollado con el principal objetivo facilitar la carga y recolección diaria de datos del campo, así como consultar información del estado actual del mismo en cualquier momento.

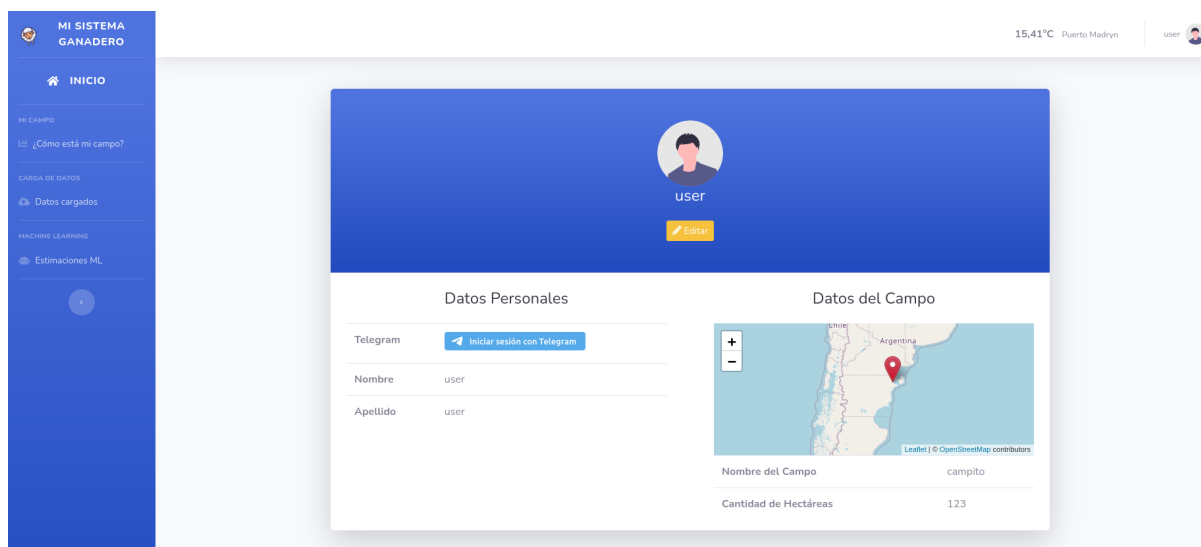


Figura 5.3.8. Pantalla mi perfil de la aplicación.

Figura 5.3.9. Solicitud de datos para login Telegram.

Para vincular la cuenta de Telegram del usuario, el sistema le solicita que se registre proporcionando su número de teléfono. Una vez completado con éxito el registro, el usuario estará habilitado para interactuar de manera directa con el bot, para ello, el usuario puede elegir escanear el código QR del bot o establecer directamente a una nueva conversación mediante un enlace como muestra se observa en la imagen siguiente.

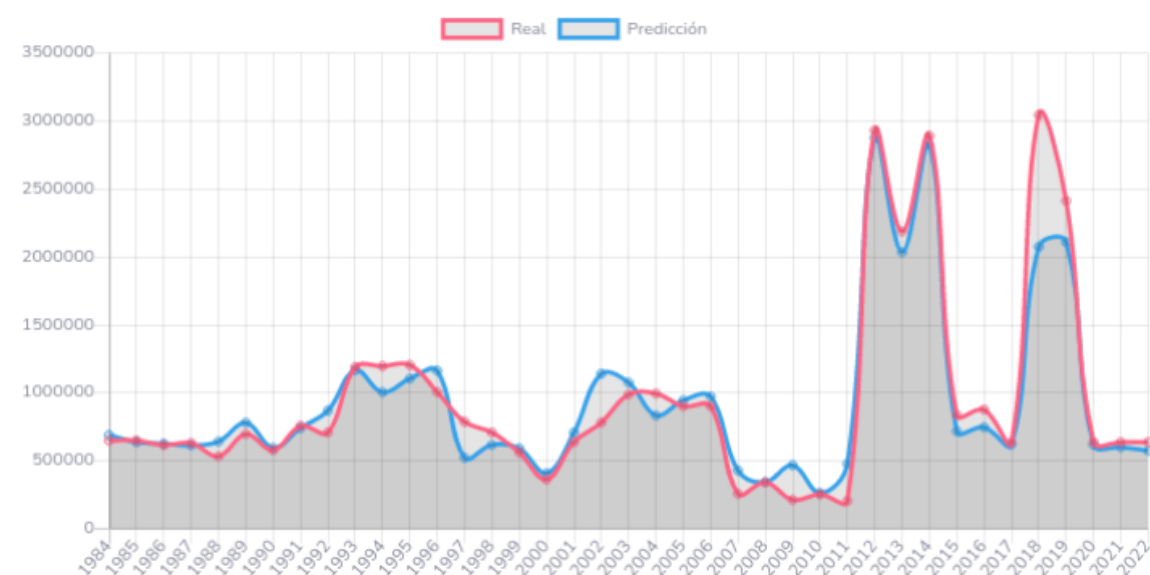


Figura 5.3.10. Sección Telegram de la aplicación.

5.3.6 Estimaciones Machine Learning

En esta pantalla se presentan cuatro gráficos de estimaciones productivas, cada uno de los cuales representa un modelo de Machine Learning diseñado para realizar una estimación específica. El proceso específico de entrenamiento y de selección de variables de interés de estos modelos se detalla en el capítulo 4 de esta tesina. El primer modelo visualizado en la figura 5.3.11, estima la cantidad de kilogramos de lana que el productor podrá obtener en el año actual, en función de los kilogramos de lana producidos en años anteriores, la cantidad de ovinos que se encuentren en el campo y la lluvia acumulada para ese año en particular. Este modelo utiliza una regresión lineal múltiple y tiene una precisión de estimación del 91.91%.

Modelo para Estimación de Kg de Lana

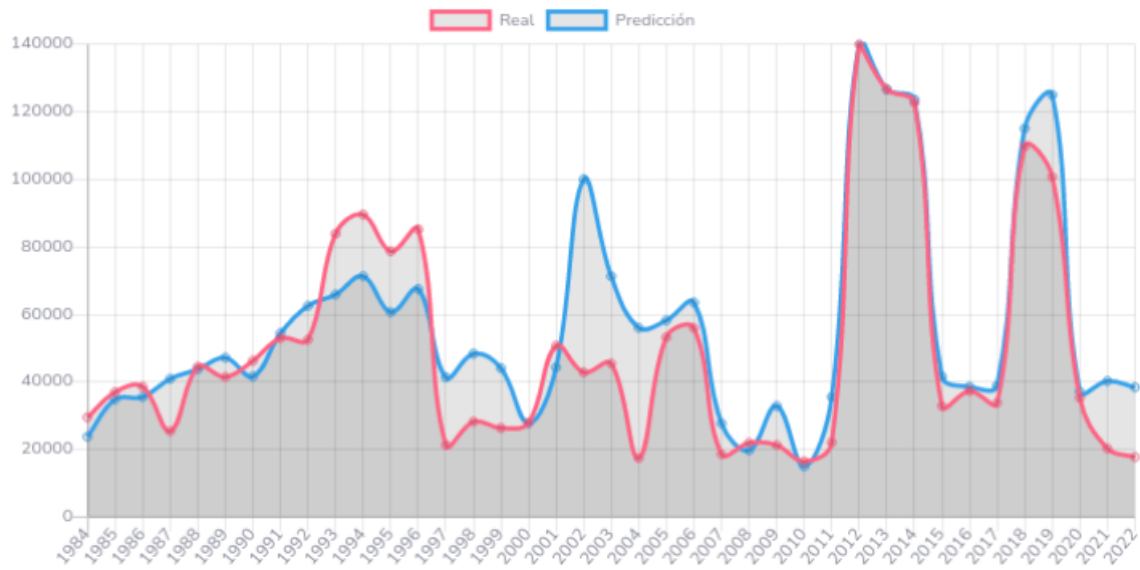


Exactitud de la predicción (R2): 91.91%

Figura 5.3.11. Modelo de estimación de kg. de lana.

Del mismo modo, el modelo de corderos visualizado en la figura 5.3.12, utiliza un enfoque similar basado en los nacimientos de corderos en años anteriores, la cantidad de ovinos que se encuentren en el campo y la lluvia acumulada registrada en un año específico. Para este modelo el porcentaje de confiabilidad es del 75.60%.

Modelo para Estimación de Corderos

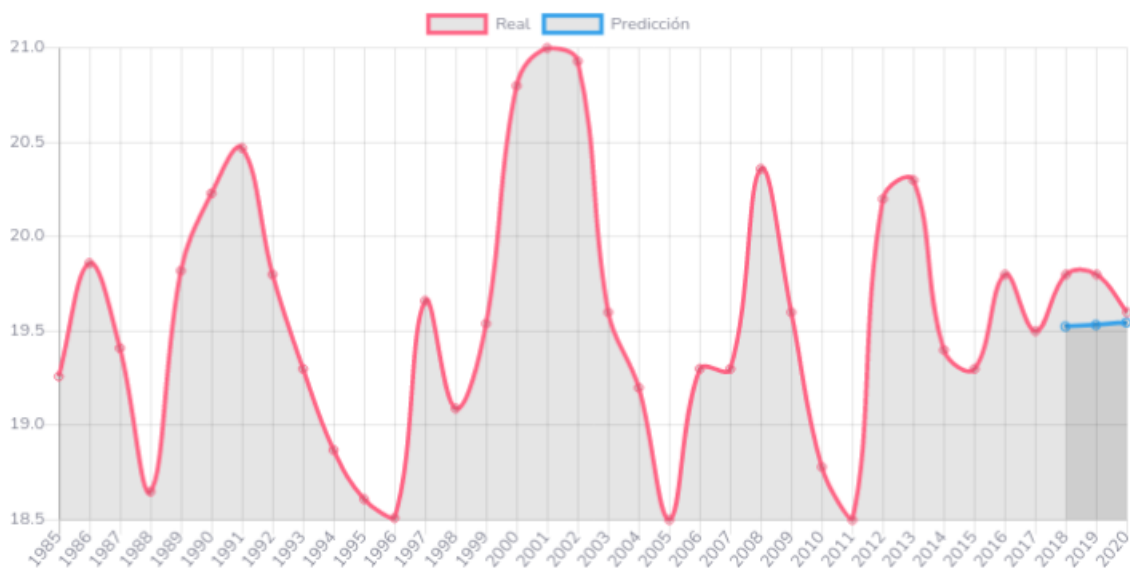


Exactitud de la predicción (R^2): 75.60%

Figura 5.3.12. Modelo de estimación de corderos.

Los dos modelos restantes utilizan un enfoque de regresor forecast para estimar el rendimiento y la finura de la lana para la zafra del año actual. En la figura 5.3.13 se muestra la estimación de la finura de la lana con un error de 0.05 micrones, mientras que en la figura 5.3.14 se muestra la estimación del rendimiento de la lana con un error de 4.16 de rinde.

Modelo para Estimación de Finura



Error (MSE): 0.05 micras

Figura 5.3.13. Modelo de estimación de finura de lana.

Modelo para Estimación de Rinde

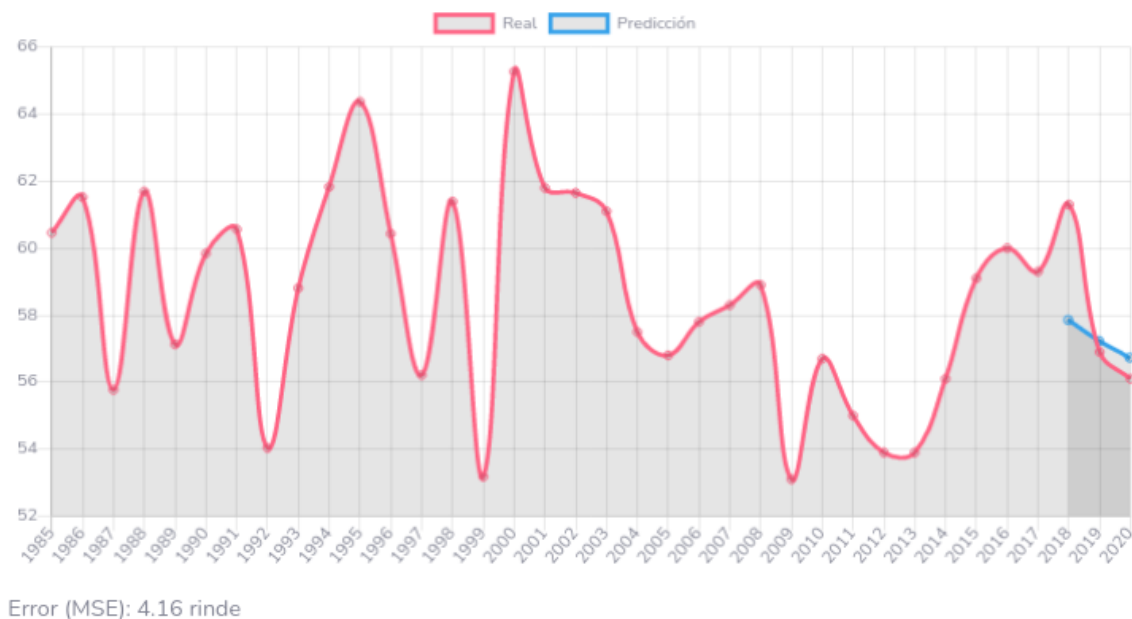


Figura 5.3.14. Modelo de estimación de rinde de lana.

En conjunto estos modelos de Inteligencia Artificial combinan cuatro estimaciones de las principales variables de interés productivo, lo que proporciona al usuario final (un productor de campo) una herramienta gráfica de rápida interpretación.

A través de esta representación visual, el productor obtiene información valiosa sobre la producción de lana y corderos, así como una visualización clara y rápida de la fluctuación de sus datos y de posibles desviaciones de los mismos. En resumen, estos modelos de IA brindan previsibilidad productiva, lo que ayuda al productor a tomar decisiones más informadas y mejorar la eficiencia de sus procesos.

5.3.7 Bot Telegram

Como se mencionó anteriormente, es posible interactuar con la aplicación mediante la utilización de un bot de Telegram. Dicho bot tiene funciones específicas que permiten tanto automatizar las tareas de carga de información, como también consultar el estado actual de las variables productivas de interés de un campo en particular. Además de las funciones implementadas en código, la utilización de un bot de Telegram supone algunas ventajas intrínsecas como pueden ser[108]:

- ❖ **Accesibilidad:** Telegram es una aplicación ampliamente utilizada y disponible en la mayoría de los dispositivos, lo que significa que un bot de Telegram puede hacer que la webapp sea fácilmente alcanzada por un público más amplio.
- ❖ **Interacción directa:** Los usuarios pueden tener una experiencia más rápida y personalizada al interactuar directamente con la webapp a través del bot de Telegram.

- ❖ Interfaz de usuario simplificada: El bot de Telegram puede proporcionar una interfaz de usuario simple y fácil de usar, lo que facilita a los usuarios el uso de la webapp.
- ❖ Comunicación instantánea: La comunicación a través de un bot de Telegram es instantánea, lo que permite a los usuarios recibir información en tiempo real sobre los cambios en la webapp.

El bot diseñado tiene como función principal facilitar la carga de datos del campo y permitir la consulta del estado de diversas variables productivas en cualquier momento.

Para realizar la carga de datos, se utiliza un comando específico denominado «actualizar datos» que permiten ingresar una serie de acciones que suceden sobre alguna de las categorías de animales presentes en el campo. Por ejemplo, dentro de las categorías disponibles de animales que muestra el bot se encuentran: Ovejas, corderos y carneros. Por otra parte, dentro de las acciones disponibles para utilizar sobre cada categoría animal se encuentran: Ventas, Compras, Muertes, Particiones (solo corderos). Este esquema brinda la posibilidad de ingresar por cada animal de 3 a 4 opciones de registro. Al ingresar este comando, se desencadenan una serie de eventos por parte del bot, tales como:

- ❖ Selección de la categoría animal deseada: El usuario debe ingresar la categoría de animales que desea cargar: ovejas, carneros o corderos.
- ❖ Selección de la acción: Las acciones determinan el tipo de información que se cargará para la categoría animal seleccionada. Las opciones disponibles son: particiones (en caso de corderos), muertos, ventas y compras de animales.
- ❖ Por último, el bot solicita el ingreso del valor y procede a actualizar la base de datos del campo con la información recolectada.

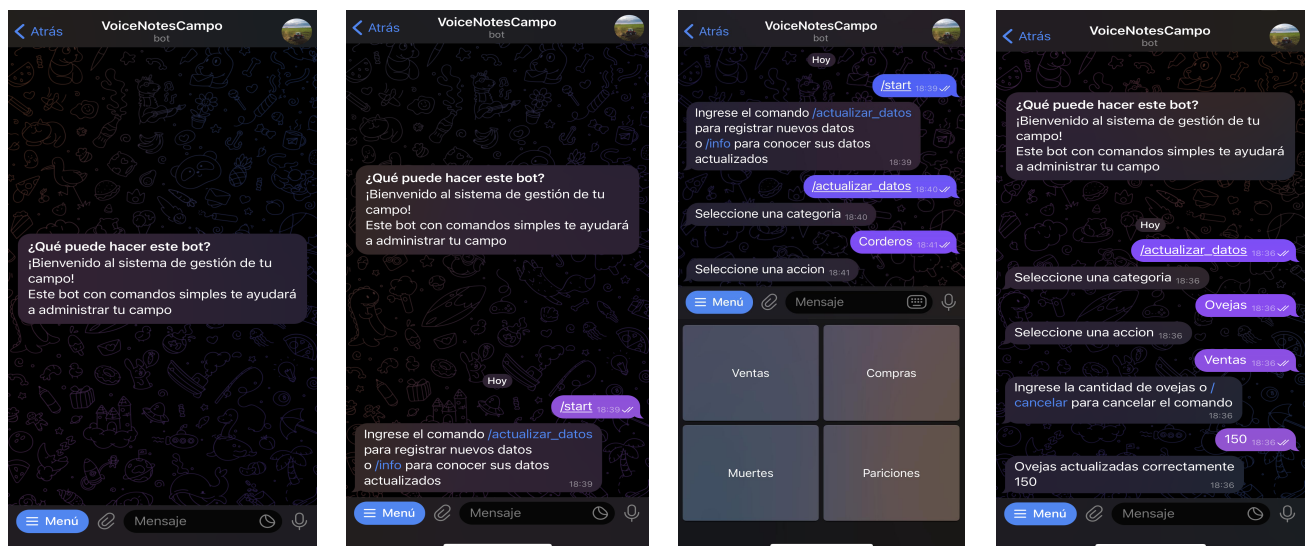


Figura 5.3.15. Chat de Telegram.

De esta manera es posible actualizar los datos productivos del campo que involucran variación en la cantidad total de animales.

Por otro lado, el bot también cuenta con un comando para consultar el estado actual del campo teniendo en cuenta el último registro de información cargado que permite visualizar

tanto los datos climáticos como productivos. Para realizar esta consulta se utiliza el comando **/info** como se visualiza en la siguiente imagen:

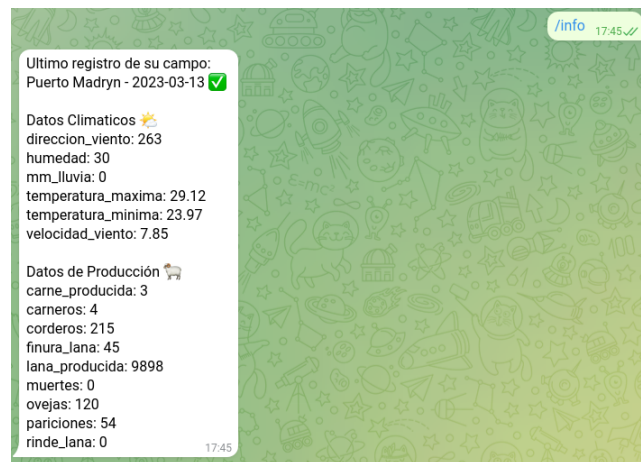


Figura 5.3.16. Comando info Telegram.

También es posible automatizar el proceso de carga manual de datos mediante el envío de mensajes de voz con un formato preestablecido, informando las mismas variables de interés disponibles con los comandos manuales, pero esta vez ingresando un comando de voz.

Para llevar a cabo esta funcionalidad utilizamos la combinación de dos bibliotecas que permiten interactuar con archivos de audio y realizar análisis y reconocimiento de texto en voz que son:

- ❖ **speech_recognition**: La librería speech recognition de Python es una biblioteca de reconocimiento de voz que permite transcribir audio a texto de manera fácil, rápida y que ofrece una interfaz para trabajar con distintas APIs de reconocimiento de voz, como Google Cloud Speech API, IBM Speech to Text, Microsoft Azure Speech API, entre otras. Esta herramienta permite obtener audio de diferentes fuentes, como micrófonos o archivos de audio, y convertir el audio en texto utilizando técnicas de reconocimiento de voz avanzadas [121].
- ❖ **spacy**: La librería spaCy es una herramienta de procesamiento del lenguaje natural (NLP) escrita en Python que se utiliza para realizar tareas de procesamiento del lenguaje natural, como tokenización, etiquetado de partes de la oración, análisis sintáctico y reconocimiento de entidades. Esta biblioteca posee modelos presentados de procesamiento de lenguaje natural, de los cuales utilizamos el modelo `es_core_news_sm` que es un modelo diseñado para procesar texto en español. Este modelo incluye un conjunto de reglas y patrones que permiten analizar la estructura gramatical de un texto y etiquetar sus componentes (por ejemplo, sustantivos, verbos, adjetivos, etc.), así como reconocer entidades [122].

En particular, utilizamos la función **Matcher** de spaCy para buscar patrones específicos en un texto, aplicando reglas gramaticales y semánticas. Por ejemplo, podemos buscar un patrón particular en un texto como «compré 1500 ovejas» o «vendí 500 corderos», y establecer las variables de interés y las cantidades adecuadas a cargar mediante una regla gramatical

específica. Una regla gramatical posee características similares al armado de una expresión regular donde es posible indicar las partes y características que debe poseer una cadena de texto para poder extraer información de interés. Un ejemplo de una regla gramatical utilizada con la función matcher puede ser la siguiente:

[{'POS': 'VERB', 'LEMMA': 'comprar'}, {'POS': 'NUM'}, {'POS': 'NOUN', 'LEMMA': 'oveja'}]

Donde:

- {'POS': 'VERB', 'LEMMA': 'comprar'}: Buscamos un elemento verbo en alguna posición de la cadena ingresada que sea el lema de «comprar». Ej: Compraré, compré, comprado, etc.
- {'POS': 'NUM'}: El siguiente token a identificar, luego del lema anterior, será un número.
- {'POS': 'NOUN', 'LEMMA': 'oveja'}: Por último, luego del número ingresado buscamos el token en la posición siguiente, que sea un sustantivo lema de «oveja» Ej: Ovejas.

Con esta regla, podríamos buscar ocurrencias de la secuencia de tokens «compré X ovejas» o «nacieron X corderos» en un texto presente en un comando de voz enviado por el productor del campo mediante el chat del bot.

Por ejemplo, al enviar un comando de voz diciendo: “*Ayer nacieron 20 corderos y además compré 5 ovejas y murieron 10 carneros*” es posible reconocer patrones de interés con reglas gramaticales preestablecidas y extraer la información necesaria para actualizar los datos de producción del campo:

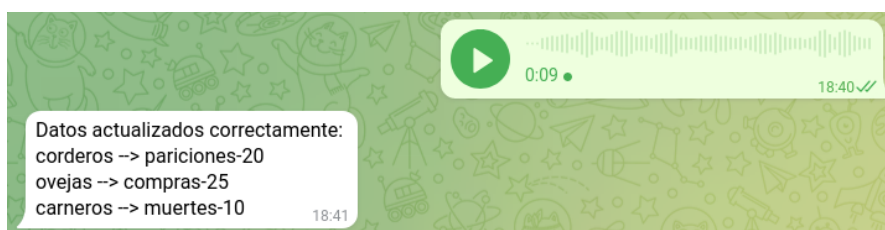


Figura 5.3.17. Comando por voz telegram.

Capítulo 6: Comentarios finales

A continuación, se expondrá una revisión detallada acerca de lo que representó el proceso de investigación, desarrollo y elaboración de la presente tesina. Este capítulo marca el cierre de un ciclo de intenso aprendizaje y desarrollo personal y profesional.

La realización de esta tesina ha sido una experiencia sumamente valiosa que nos ha brindado la oportunidad de profundizar una gran cantidad de campos de estudio y, en el transcurso, adquirir nuevas habilidades, herramientas, técnicas y conocimientos que, de seguro, nos serán de enorme utilidad en nuestra futura carrera profesional y que independiente al resultado final

de este trabajo, han sido la guía principal para atravesar este camino de profundo enriquecimiento investigativo y personal.

A lo largo del proceso, nos hemos enfrentado a desafíos que nos han obligado a salir de la zona de confort y hemos sabido resolver de manera airosa experimentando la satisfacción de superarlos y lograr alcanzar los objetivos impuestos.

Dada la naturaleza de la tesina como trabajo de investigación acotado a plazos y objetivos específicos, creemos que la presente es solo una parte de múltiples líneas de investigación que, aunque no se han abordado en detalle, notamos oportuno destacar convencidos de que pueden ser utilizadas en investigaciones académicas más profundas y complejas en un futuro.

Observación de resultados y conclusiones

El desarrollo de la presente tesina estuvo enfocado en realizar una investigación en profundidad sobre la situación actual del concepto de Industria 4.0 y por qué es importante su implementación a nivel global y provincial. También se profundizó en las características productivas de la región chubutense y las problemáticas que enfrentan los productores agropecuarios, específicamente del sector ovino. Se mencionaron posibles herramientas de ayuda a la gestión productiva y cómo están siendo cada vez más utilizadas por los productores de campo. Además, se ahondó en el concepto de PLF y su implementación con sus ventajas y desventajas, y su aporte a la producción animal.

Se logró cumplir con el objetivo general de la tesis al construir una aplicación web que aplicara buenas prácticas de programación, empleando herramientas actuales y haciendo hincapié en la utilización, la practicidad y la disminución de la curva de aprendizaje para el usuario final. Esto permitiría al productor contar con una mayor previsibilidad productiva y la consecuente mejora de sus procesos agrícolas.

Durante el desarrollo del proyecto, además, se interactuó con distintas fuentes de información externas, como el SIPAS o aplicaciones que brindaran información climática actualizada para puntos específicos de la región chubutense. Además, se consultaron e investigaron una gran cantidad de datasets de información productiva ovina, sorteando uno de los principales obstáculos en el desarrollo del proyecto, que fue no solo la falta de datos, sino también la dificultad para encontrar registros productivos y climáticos de calidad.

Por ello, se decidió no solo desarrollar un DSS que permitiera explorar y analizar los datos de producción existentes de un campo, sino también brindar herramientas al productor que le permitieran registrar los datos diarios de su campo y sus fluctuaciones. En base a esta premisa se desarrolló, además, un bot estandarizado para el registro de datos productivos con el fin de mejorar la accesibilidad y facilidad en el registro de información histórica de un campo además de aumentar la cantidad y la calidad de sus datos y su categorización.

La falta de datos públicos sobre la existencia de lana y ovinos en la región es una limitación importante para la toma de decisiones en el sector ganadero. Por lo tanto, la existencia y

disponibilidad de esta información permitiría una mejor planificación y gestión de los recursos productivos.

Finalmente, se logró alcanzar el objetivo de utilizar tecnologías modernas como soluciones basadas en la implementación de algoritmos de inteligencia artificial, con la posibilidad de que el productor de campo pueda realizar una estimación de sus variables productivas de mayor interés, como kilogramos de lana, rinde y finura y también una estimación de la cantidad de corderos.

En resumen, creemos haber alcanzado los objetivos propuestos al inicio de este trabajo, logrando enriquecer profundamente nuestros caminos académicos y futuros como profesionales informáticos. Estamos convencidos de que la investigación llevada a cabo en este proyecto de tesina puede brindar valiosas herramientas de ayuda a la gestión productiva en la región chubutense y contribuir a la toma de decisiones más informadas y efectivas en la gestión diaria de los campos.

Líneas futuras de investigación

A continuación se presentarán posibles campos de estudio futuros y de complementación al desarrollo, que no han sido abordados en la presente tesina. Estas líneas de investigación no solo ampliarían el conocimiento en el tema abordado, sino que también podrían contribuir al desarrollo de nuevas áreas de investigación.

- ❖ Complementación al desarrollo: Complementariamente al desarrollo del sistema podrían implementarse funcionalidades específicas que por cuestiones técnicas y de objetivos acotados no han sido desarrolladas e incluidas en el desarrollo actual, como pueden ser:
 - Más de un campo por usuario: En un escenario de uso real del sistema, es posible que un mismo productor pueda administrar múltiples campos a la vez. En ese caso, resultaría conveniente que pudiera hacerlo desde la misma aplicación sin necesidad de realizar configuraciones adicionales y de manera transparente.
 - Más de un usuario de Telegram por campo: Actualmente, el sistema solo permite vincular una sola cuenta de Telegram al productor del campo. Sin embargo, en campos de gran extensión territorial y amplia cantidad de animales, puede resultar útil permitir que diferentes actores dentro de la gestión del campo actualicen los datos productivos en simultáneo, por lo que sería beneficioso poder permitir esta configuración dentro de la aplicación.
- ❖ Ampliar modelos de IA: En la actualidad, dentro del sistema cada usuario cuenta con cuatro modelos de inteligencia artificial asociados a su campo, cada uno especializado en un ámbito específico. Sin embargo, sería interesante y beneficioso ampliar la cantidad de modelos y complementar la información productiva y climática con datos sobre pasturas y forrajes presentes en las hectáreas del campo o también podrían

utilizarse imágenes satelitales que permitan analizar la calidad de los suelos del campo. De esta manera, se podrían investigar y determinar las relaciones existentes entre todo el conjunto de datos y cómo afectan la calidad productiva ganadera de un campo en particular.

Además, resultaría interesante desarrollar cuatro modelos genéricos que utilicen toda la información disponible de todos los campos en la base de datos. De este modo, los usuarios tendrían la opción de elegir el modelo más apropiado para sus necesidades específicas. Por ejemplo, podrían elegir el modelo de lana que mejor se ajuste a sus datos, al departamento donde se encuentran, o incluso a toda la provincia de Chubut. Al ampliar la cantidad de modelos disponibles, se mejoraría significativamente la capacidad de gestión de un campo, aumentando la utilidad final de la aplicación.

- ❖ Incluir más variables de interés para el productor: En cuanto a la producción de lana en el campo, resulta de gran importancia contar con una mayor cantidad de variables de interés para el productor. En este sentido, sería de gran importancia poder calcular el precio de venta de la lana acumulada en base al precio del dólar actual para poder conocer de manera aproximada el retorno económico que sería posible obtener.

Asimismo, sería oportuno investigar soluciones que permitan establecer cuáles son los momentos más oportunos para la venta de un lote de lana. Esta mejora resultaría de gran valor para el productor, ya que le brindaría la oportunidad de tomar decisiones más estratégicas en el mercado.

Es importante mencionar que la incorporación de estas variables podría resultar compleja, pero el esfuerzo y la inversión en este sentido se traducirían en un aumento significativo en la rentabilidad productiva del campo, maximizando los beneficios en la producción de lana.

- ❖ Mejorar IA de reconocimiento de voz: Actualmente, el bot desarrollado solo tiene la capacidad de procesar información en tiempo real, es decir, en el momento en que se emite un comando de voz específico, la información es procesada y cargada en la base de datos del campo en esa fecha específica, lo que limita su capacidad para la carga de información o actualización de datos históricos.

Una mejora significativa sobre el trabajo realizado con algoritmos de reconocimiento de voz podría ser el desarrollo y elaboración de nuevas y mejores reglas gramaticales que permitan al bot identificar fechas precisas o momentos específicos de tiempo en los comandos de voz y, de esta manera, brindar la capacidad de proporcionar información histórica o actual según corresponda.

Esta funcionalidad sería de gran utilidad y una mejora en el reconocimiento de voz que permitiría una interacción más fluida y natural entre el usuario y el bot, resultando en una experiencia de usuario más efectiva y un manejo de datos más preciso y con ello dando al usuario final respuestas más útiles.

Bibliografía

- [1] “¿Qué es la industria 4.0 y qué se necesita saber sobre este concepto?”. *Software ISO*, 2018. Recuperado de: <https://www.isotools.org/2018/07/12/industria-4-0-que-debemos-saber/> [06/03/2023]
- [2] “¿Qué es la Industria 4.0?”. *Deloitte Spain*, 2018. Recuperado de: <https://www2.deloitte.com/es/es/pages/manufacturing/articles/que-es-la-industria-4.0.html> [06/03/2023]
- [3] “Industry 4.0 and the fourth industrial revolution explained”. *i-SCOOP*, 2016. Recuperado de: <https://www.i-scoop.eu/industry-4-0/> [07/03/2023]
- [4] Recuperado de: <https://moretech.mx/industria-4-0-en-el-mundo/> [15/09/2020]
- [5] “La Industria 4.0 y sus orígenes”. *Aspromec*, 2019. Recuperado de: <https://aspromec.org/la-industria-4-0-y-sus-origenes/> [07/03/2023]
- [6] David García. “España ocupa el puesto 24 en la Industria 4.0”. *Infoplc.net*, 2018. Recuperado de: <https://www.infoplc.net/actualidad-industrial/item/105075-espana-puesto-24-industria-4-0> [07/03/2023]
- [7] “Readiness for the Future of Production Report 2018”. World Economic Forum, 2018. ISBN: 978-1-944835-16-3. Recuperado de: http://www3.weforum.org/docs/FOP_Readiness_Report_2018.pdf [10/03/2023]
- [8] Ramiro Albrieu & Martín Rapetti. “¿Robots en las pampas? Futuros alternativos para el mercado de trabajo argentino en la Cuarta Revolución Industrial”. *Cippec*, 2018. Recuperado de: <https://www.cippec.org/wp-content/uploads/2018/11/210-DPP-ADE-Robots-en-las-pampas-Rapetti-y-Albrieu-diciembre-2018-vf.pdf> [11/03/2023]
- [9] https://www.estadistica.chubut.gov.ar/index.php?option=com_content&view=article&id=503&Itemid=368
- [10] <https://politicachubut.com.ar/noticias/3/economia> [11/03/2023]
- [11] S. Morrone, C. Dimauro, F. Gambella & M. G. Cappai. “Industry 4.0 and precision livestock farming (PLF): An up to date overview across animal productions”. *Sensors (Basel)*, vol. 22, num. 12, 2022. DOI: 10.3390/s22124319. Páginas: 25. Recuperado de: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9228240/> [06/03/2023]
- [12] “Vacas conectadas, la ganadería 4.0 llega a granjas españolas”. *Contexto ganadero*, 2019. Recuperado de: <https://www.contextoganadero.com/internacional/vacas-conectadas-la-ganaderia-40-llega-granjas-espanolas> [12/03/2023]
- [13] L. O. Tedeschi, P. L. Greenwood & I. Halachmi. “Advancements in sensor technology and decision support intelligent tools to assist smart livestock farming”. *Journal of Animal Science*, vol. 99, num. 2, 2021. DOI: 10.1093/jas/skab038. Páginas: 11. Recuperado de: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7896629/pdf/skab038.pdf> [06/03/2023]
- [14] R. García, J. Aguilar, M. Toro, A. Pinto & P. Rodríguez. “A systematic literature review on the use of machine learning in precision livestock farming”. *Computers and Electronics in*

- Agriculture*, vol. 179, num 105826, 2020. DOI: 10.1016/j.compag.2020.105826. Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S0168169920317099> [11/03/2023]
- [15] R. Gauthier, C. Largouët & J. Dourmad. “Prediction of litter performance in lactating sows using machine learning, for precision livestock farming”. *Computers and Electronics in Agriculture*, vol. 196, num 106876, 2022. DOI: 10.1016/j.compag.2022.106876. Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S0168169922001934> [11/03/2023]
- [16] A. L. Chaves Gurgel et al. “Aspectos relacionados con la importancia del uso de modelos predictivos en la producción ovina. Revisión”. *Revista Mexicana de Ciencias Pecuarias*, vol. 14, núm. 1, 2022. Recuperado de: <https://cienciaspecuarias.inifap.gob.mx/index.php/Pecuarias/article/view/6126/5021> [11/03/2023]
- [17] Wikipedia contributors. “Provincia del Chubut”. *Wikipedia, The Free Encyclopedia*. Recuperado de: https://es.wikipedia.org/wiki/Provincia_del_Chubut [11/03/2023]
- [18] Censo Nacional de Población, Hogares y Vivienda. *INDEC*, 2022. Recuperado de: https://www.censo.gob.ar/index.php/mapa_poblacion1/ [10/03/2023]
- [19] Censo Nacional de Población, Hogares y Vivienda. *INDEC*, 2010. Recuperado de: <https://www.indec.gob.ar/indec/web/Nivel4-CensoProvincia-999-999-26-000-2010> [10/03/2023]
- [20] Hermosilla Rivera, Cristian. “Población Rural En Chubut: La Meseta Desde Principio De Siglo XX A La Actualidad”. *Párrafos Geográficos*, vol. 12, núm. 1, 2013. Recuperado de: <http://www.revistas.unp.edu.ar/index.php/parrafosgeograficos/article/view/265/233> [11/03/2023]
- [21] “Plan Ganadero de la Provincia del Chubut”. *Ministerio de la Producción de Chubut*, 2017. Recuperado de: https://www.magyp.gob.ar/sitio/areas/d_ovinos/informes/_nacionales/_archivos//000001_Chubut/000000_Plan%20Ganadero/000000_Plan%20Ganadero%20Chubut%202017.pdf [11/03/2023]
- [22] Barry D., Cortez J. M. & Ordinez L. “Kit tecnológico para la automatización de romaneos”. In *VIII Congreso Argentino de AgroInformática - JAIIO 45*, 2016. Recuperado de: http://sedici.unlp.edu.ar/bitstream/handle/10915/57679/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y [11/03/2023]
- [23] Marco Antonio Aceves Fernández. “Inteligencia artificial para programadores con prisa”. *Universo de Letras*, 2022. ISBN: 8418856726, 9788418856723. Páginas: 162.
- [24] María Virginia Sturzenbaum & Larry Andrade. “Los productores ganaderos ovino-extensivos y la adopción tecnológica en el sureste de la provincia de Santa Cruz”. *Instituto Nacional de Tecnología Agropecuaria*, 2012. Recuperado de: <https://inta.gob.ar/documentos/los-productores-ganaderos-ovino-extensivos-y-la-adopcion-tecnologica-en-el-sureste-de-la-provincia-de-santa-cruz> [12/03/2023]
- [25] Ejarque, Mercedes. “¿Solo ‘hay que esperar que llueva’? Interpretaciones y prácticas sobre una sequía de los agentes ‘laneros’ en Chubut”. *Estudios Rurales*, vol. 10, núm. 20, 2020. Recuperado de: <http://portal.amelica.org/ameli/jatsRepo/181/1811302004/html/index.html> [12/03/2023]

- [26] “Situación actual y perspectiva de la ganadería en Patagonia”. *Centro Regional Patagonia Sur – Instituto Nacional de Tecnología Agropecuaria*, 2015. Recuperado de: https://inta.gob.ar/sites/default/files/inta_situacion_actual_perspectivas_ganaderia_patagonia_sur.pdf [12/03/2023]
- [27] Briceño Díaz & Freddy Arturo. “Sistemas de Información como apoyo a la toma de decisiones”. *PROSPECTIVA*, vol. 4, núm. 1, 2006. Recuperado de: <https://www.redalyc.org/pdf/4962/496251107008.pdf> [12/03/2023]
- [28] U. E. Gómez-Prada, M. L. Orellana-Hernández & J. M. Salinas-Ibáñez. “Apropiación de Sistemas de Tecnologías de la Información para toma de Decisiones de Productores Agroindustriales Basada en Videojuegos Serios. Una Revisión”. *Información tecnológica*, vol. 30, núm. 5, 2019. Recuperado de: https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-07642019000500331&lng=es&nrm=iso [12/03/2023]
- [29] “Análisis del Ambiente”. *Consulplan, Gestión ambiental*, 2017. Recuperado de: <http://www.ambiente.chubut.gov.ar/wp-content/uploads/2017/01/IAP-Planta-de-Corte-Intermedio-GBK-Parte-2.pdf> [12/03/2023]
- [30] Allice_Hunter. “Chubut Köppen”. *Wikimedia*. Recuperado de: https://commons.wikimedia.org/wiki/File:Chubut_K%C3%B6ppen.svg#/media/File:Chubut_K%C3%B6ppen.svg [25/03/2023]
- [31] D. Soubié, M. C. Gonzalez & Z. Trinidad. “Chubut - Clima”. *todo-argentina.net*. Recuperado de: <https://www.todo-argentina.net/geografia/provincias/chubut/clima.html> [25/03/2023]
- [32] Wikipedia contributors. “Valle inferior del río Chubut”. *Wikipedia, The Free Encyclopedia*. Recuperado de: https://es.wikipedia.org/wiki/Valle_inferior_del_r%C3%ADo_Chubut [12/03/2023]
- [33] Pablo Rimoldi. “Producción Ovina en Chubut”. *Revista Idia 21, Ediciones INTA*. Recuperado de: https://www.produccion-animal.com.ar/produccion_ovina/produccion_ovina/19-chubut.pdf [07/03/2023]
- [34] “Manual del Ovejero Mesopotámico”. *Ediciones INTA, Capítulo 8, 1987*. Recuperado de: https://www.produccion-animal.com.ar/libros_on_line/21-ovejero_mesopotamico/09-capitulo_8.pdf [07/03/2023]
- [35] J. C. P. Gallegos, A. T. Soto, S. Aguilera, et al. “Inteligencia Artificial - 1a ed”. *Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn)*, 2014. Páginas: 225. Recuperado de: https://rephip.unr.edu.ar/bitstream/handle/2133/17686/1520250496_Inteligencia-Artificial-C-C-BY-SA-3.0-86.pdf [07/03/2023]
- [36] “Definición de inteligencia artificial”. *Definición.de*. Recuperado de: <https://definicion.de/inteligencia-artificial/> [05/03/2023]
- [37] María Del Carmen Sosa Sierra. “Inteligencia artificial en la gestión financiera empresarial”. *Pensamiento & Gestión*, núm. 23, 2007. Recuperado de: https://drive.google.com/file/d/1Xjj9_780Ewo9uiPBbPbVKgAQqh42NZNR/view [12/03/2023]

- [38] “¿Qué es la inteligencia artificial (IA)?”. *Oracle*. Recuperado de: <https://www.oracle.com/ar/artificial-intelligence/what-is-ai/> [12/03/2023]
- [39] “Libro Online de la comunidad de Inteligencia Artificial Argentina”. *IARR*. Recuperado de: <https://iaarbook.github.io/> [07/03/2023]
- [40] E. V. Cuevas, O. Avalos, E. Primitivo, A. Valdivia & M. A. Pérez. “Introducción al Machine Learning con MATLAB”. *Marcombo*, 2021. ISBN: 978-84-267-3282-8. Páginas: 260. Recuperado de: <https://es.scribd.com/read/514471048/Introduccion-al-Machine-Learning-con-MATLAB> [05/03/2023]
- [41] S. Guido & A. C. Müller. Introduction to machine learning with python: A guide for data scientists. *O’Reilly Media*, 2016. ISBN: 978-1-449-36941-5. Páginas: 392. Recuperado de: https://drive.google.com/file/d/1WTfEFN_BXkN2QI0giUYVI5XGJvquOfby [05/03/2023]
- [42] Álex Rayón. “Guía para comenzar con algoritmos de Machine Learning”. *Deusto Data*, 2017. Recuperado de: <https://blogs.deusto.es/bigdata/guia-para-comenzar-con-algoritmos-de-machine-learning/> [12/03/2023]
- [43] A. C. Vasquez, H. V. Huerta & J. P. Quispe. “Procesamiento De Lenguaje Natural.” *Revista De Investigación De Sistemas e Informática*, vol 6, núm 2, 2009. Recuperado de: https://www.academia.edu/66213908/Procesamiento_de_lenguaje_natural [12/03/2023]
- [44] E. Kendall Kenneth & E. Kendall Julie. “Análisis y diseño de sistemas. Sexta edición.” *Pearson Educación*, 2005. ISBN: 970-26-0577-6. Páginas: 752.
- [45] Postigo Palacios Antonio. “Bases de datos”. *Ediciones Paraninfo*, 2021. ISBN: 8413660769, 9788413660769. Páginas: 460.
- [46] G. Santiago De Gea. "La producción ovina ante un nuevo escenario. Pasado, presente y futuro de la actividad". *Producción Animal*, 2003. Páginas: 4. Recuperado de: https://www.produccion-animal.com.ar/produccion_ovina/produccion_ovina/07-ovino_aapa_mza.pdf [07/03/2023]
- [47] Jorge Manazza. “¿Sabe usted cuántos corderos pierde en cada parición?”. *Producción Animal*, 2005. Páginas: 4. Recuperado de: https://www.produccion-animal.com.ar/produccion_ovina/produccion_ovina/04-perdida_corderos_en_la_paricion.pdf [07/03/2023]
- [48] “Manual del Ovejero Mesopotámico”. *Ediciones INTA*, Capítulo 1, 1987. Recuperado de: https://www.produccion-animal.com.ar/libros_on_line/21-ovejero_mesopotamico/02-capitulo_1.pdf [07/03/2023]
- [49] Pedro Ponce Cruz. “Inteligencia artificial: con aplicaciones a la ingeniería”. *Alpha Editorial*, 2010. ISBN: 607707358X, 9786077073581. Páginas: 372.
- [50] J. Han, M. Kamber & J. Pei. “Data Mining: Concepts and Techniques, 3.a ed”. *Morgan Kaufmann*, 2011. ISBN: 978-0-12-381479-1. Páginas: 744. Recuperado de: <http://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concept-s-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf> [12/03/2023]

- [51] R. S. Sutton & A. G. Barto. "Reinforcement Learning: An Introduction, 2.a ed". *Bradford Books*, 2018. ISBN: 9780262039246. Páginas: 552. Recuperado de: <http://incompleteideas.net/book/RLbook2020.pdf> [12/03/2023]
- [52] "Informes de tendencias en ciencia y tecnología. Agricultura y Ganadería inteligente". *Programa Nacional de Vigilancia Tecnológica e Inteligencia Competitiva*, 2019. Recuperado de: https://www.argentina.gob.ar/sites/default/files/reporte_de_smart_farming30-10_vf_vintec.pdf [12/03/2023]
- [53] "Ganadería de precisión: seis ejemplos concretos que facilitan la toma de decisiones". *Agrofy News*, 2020. Recuperado de: <https://news.agrofy.com.ar/noticia/190232/ganaderia-precision-seis-ejemplos-concretos-que-facilitan-toma-decisiones> [12/03/2023]
- [54] Andres Costamagna. "Ganadería 4.0: La producción de carne con animales enfrenta desafíos". *Ganados y Carnes*, 2018. Recuperado de: <https://ganadosycarnes.com/editorial/ganaderia-4-0-la-produccion-de-carne-con-animales-enfrenta-desafios/> [13/03/2023]
- [55] Sergio Osvaldo Pena. "Industria 4.0, Ambiente 4.0, ¿Ganadería 4.0?". *Agrosito*, 2019. Recuperado de: <https://www.agrosito.com.ar/noticia/201632-industria-40-ambiente-40-ganaderia-40-por-sergio-pena-inta> [13/03/2023]
- [56] Equipo de Expertos en Ciencia y Tecnología. "Smart farming. Conoce qué es y sus características". *Universidad Internacional de Valencia*, 2021. Recuperado de: <https://www.universidadviu.com/es/actualidad/nuestros-expertos/smart-farming-conoce-que-es-y-sus-caracteristicas> [15/03/2023]
- [57] F. Zhang, Y. Zhang, W. Lu, Y. Gao, Y. Gong & J. Cao. "6G-Enabled Smart Agriculture: A Review and Prospect". *Electronics*, vol. 11, núm. 18, 2022. DOI: 10.3390/electronics11182845. Recuperado de: <https://www.mdpi.com/2079-9292/11/18/2845> [16/03/2023]
- [58] Chrysanthos Maraveas. "Incorporating Artificial Intelligence Technology in Smart Greenhouses: Current State of the Art". *Applied Sciences*, vol 13, num. 14, 2022. DOI: 10.3390/app13010014. Recuperado de: <https://www.mdpi.com/2076-3417/13/1/14> [16/03/2023]
- [59] M. Dhanaraju, P. Chenniappan, K. Ramalingam, S. Pazhanivelan & R. Kaliaperumal. "Smart Farming: Internet of Things (IoT)-Based Sustainable Agriculture". *Agriculture*, vol. 12, núm. 10, 2022. DOI: 10.3390/agriculture12101745. Recuperado de: <https://www.mdpi.com/2077-0472/12/10/1745> [16/03/2023]
- [60] Z. Wang, S. Shadpour, E. Chan, V. Rotondo, K. M. Wood & D. Tulpan. "ASAS-NANP SYMPOSIUM: Applications of machine learning for livestock body weight prediction from digital images". *Journal of Animal Science*, vol. 99, num. 2, 2021. DOI: 10.1093/jas/skab022. Recuperado de: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7904040/pdf/skab022.pdf> [16/03/2023]
- [61] "Plan de Desarrollo Productivo Argentina 4.0". *Ministro de Desarrollo Productivo*, 2021. Recuperado de:

https://www.argentina.gob.ar/sites/default/files/plan_de_desarrollo_productivo_argentina_4.0.vf_1.pdf [18/03/2023]

[62] L. Domingo, Rivero, M. D. Beccaria & V. H. Preisz. “Ganadería de precisión aplicada a los rodeos de cría de Corrientes”. *Instituto Nacional de Tecnología Agropecuaria*, 2022. Recuperado de:

https://inta.gob.ar/sites/default/files/inta_ganaderia_precision_rodeos_de_cria.pdf [18/03/2023]

[63] G. A. Tallarico, O. A. H. Conegliano & R. J. Garro. “Tecnología inalámbrica para más puntos conectados”. *Instituto Nacional de Tecnología Agropecuaria*, 2021. Recuperado de: <https://inta.gob.ar/documentos/tecnologia-inalambrica-para-mas-puntos-conectados> [18/03/2023]

[64] “Merino Anuario 2020”. *Asociación Argentina Criadores de Merino*, 2020. Recuperado de: https://www.merino.org.ar/ice/wp-content/uploads/anuario_merino_2020_web.pdf [19/03/2023]

[65] Monique Éloit. “El bienestar animal: una ventaja para la industria ganadera”. *Bulletin de l’OIE*, vol. 2017, num. 1, 2017. DOI: 10.20506/bull.2017.1.2587. Recuperado de: <https://www.woah.org/app/uploads/2021/03/bull-2017-1-esp.pdf> [20/03/2023]

[66] Rebeca Garcia. “Un solo bienestar : marco de trabajo para apoyar la implementación de las normas de bienestar animal de la OIE”. *Bulletin de l’OIE*, vol. 2017, num. 1, 2017. DOI: 10.20506/bull.2017.1.2588. Recuperado de: <https://www.woah.org/app/uploads/2021/03/bull-2017-1-esp.pdf> [20/03/2023]

[67] M. Battini, E. Dalla Costa, F. Dai et al. “Aplicaciones móviles basadas en los protocolos AWIN para evaluar el bienestar animal en la granja”. *Bulletin de l’OIE*, vol. 2017, num. 1, 2017. DOI: 10.20506/bull.2017.1.2590. Recuperado de: <https://www.woah.org/app/uploads/2021/03/bull-2017-1-esp.pdf> [20/03/2023]

[68] Finnegans GO CERES. Gestión integral que simplifica la administración de los agronegocios. Recuperado de: <https://www.finneg.com/ar/site/industrias/agronegocios/> [20/03/2023]

[69] VISUAL Systemas. VISUAL Gestión Agro. Recuperado de: <https://www.visualsystemas.com.ar/visual-gestion-agro.php> [20/03/2023]

[70] Innovación Ganadera. SOFTWARE PARA OVINO. Recuperado de: <https://www.ganaderosonline.com/ovino-y-caprino> [20/03/2023]

[71] Agritec Software. Ovitec. Recuperado de: <https://www.agritecsoft.com/ovitec/es/> [20/03/2023]

[72] R. Gauthier, C. Largouët & J.-Y. Dourmad. “Prediction of litter performance in lactating sows using machine learning, for precision livestock farming”. *Computers and Electronics in Agriculture*, vol 196, num. 106876, 2022. DOI: 10.1016/j.compag.2022.106876. Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S0168169922001934> [20/03/2023]

[73] “¿Qué es el Machine learning y cómo se ha empleado en investigaciones en ganadería?”. *Contexto Ganadero*, 2020. Recuperado de: <https://www.contextoganadero.com/ganaderia-sostenible/que-es-el-machine-learning-y-como-se-ha-empleado-en-investigaciones-en> [20/03/2023]

[74] S. G. Naughton, C. B. Gleason, C. M. Leeth & R. R. White. “Brief research report: Photoplethysmography pulse sensors designed to detect human heart rates are ineffective at

- measuring horse heart rates”. *Frontiers in Animal Science*, vol. 4, 2023. Recuperado de: <https://www.frontiersin.org/articles/10.3389/fanim.2023.1103812/full> [21/03/2023]
- [75] J. C. Ascough II et al. “GPFARM: An integrated decision support system for sustainable great plains agriculture”. *Purdue University Indianapolis*, 2002. Recuperado de: <https://topsoil.nserl.purdue.edu/nserlweb-old/isco99/pdf/ISCOdisc/SustainingTheGlobalFarm/P039-Ascough.pdf> [25/03/2023]
- [76] J. R. Donnelly, A. D. Moore & M. Freer. “GRAZPLAN: Decision support systems for Australian grazing enterprises — I. Overview of the GRAZPLAN project, and a description of the MetAccess and LambAlive DSS». *Agricultural Systems*, vol. 54, núm. 1, 1997. DOI: 10.1016/S0308-521X(96)00046-7 Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S0308521X96000467> [25/03/2023]
- [77] “GrazPlan – DSS for agriculture”. *Csiro*. Recuperado de: <https://grazplan.csiro.au/> [25/03/2023]
- [78] I. R. Johnson et al. “DairyMod and EcoMod: biophysical pasture-simulation models for Australia and New Zealand”. *Australian Journal of Experimental Agriculture*, vol. 48, núm. 5, 2008. Recuperado de: <https://www.publish.csiro.au/an/ea07133> [22/03/2023]
- [79] “What is APSIM?”. *APSIM*, 2019. Recuperado de: <https://www.apsim.info/apsim-model/> [22/03/2023]
- [80] Harald Volden. “NorFor: The Nordic feed evaluation system”. *Springer Science & Business Media*, 2011. ISBN: 9086867189, 9789086867189. Páginas: 180.
- [81] “NorFor”. Recuperado de: <https://www.norfor.info/> [22/03/2023]
- [82] “Breedplan”. *University of New England*. Recuperado de: <https://breedplan.une.edu.au/products/breedplan/> [23/03/2023]
- [83] “Ranch Manager”. *White Mountains Livestock Company LLC*. Recuperado de: <https://ranchmanageropen.com/features.html> [23/03/2023]
- [84] <https://ovinapp.com/que-es-ovinapp/> [23/03/2023]
- [85] S. Shilpashree, R. R. Patil & C. Parvathi. “Cloud computing an overview”. *International Journal of Engineering & Technology*, vol. 7, núm. 4, 2018. DOI: 10.14419/ijet.v7i4.10904. Recuperado de: <https://www.sciencepubco.com/index.php/ijet/article/view/10904> [24/03/2023]
- [86] “Software Ovinca”. *USATI LTDA*. Recuperado de: <https://www.ovinca.com/Ovinca.aspx> [24/03/2023]
- [87] “Huella 4.0”. *Huella Software*. Recuperado de: <http://www.softhuella.com.ar/especificaciones/> [24/03/2023]
- [88] “Gestión Ganadería”. *Physis*. Recuperado de: <https://www.physis.com.ar/aplicacion.php?id=6> [24/03/2023]
- [89] “Software Campo GI”. *AgroGi*. Recuperado de: <https://agrogi.com.ar/software-campo-gi/> [24/03/2023]
- [90] Sebastián Li. “Las redes de diálogo en los sistemas de producción de conocimiento de las poblaciones rurales dispersas de Gualjaina (Chubut)”. *Universidad Nacional Del Sur*, 2016. Recuperado de: <https://repositoriodigital.uns.edu.ar/bitstream/handle/123456789/3436/Tesis%20LI%20SEBASTIAN%20%20%28BC%20UNS%29.pdf> [25/03/2023]

- [91] “SIF Móvil”. *Secretaría de Agricultura, Ganadería y Pesca*. Recuperado de: <https://www.magyp.gob.ar/sitio/areas/gestion/sif/> [25/03/2023]
- [92] “CaprinApp - Manual de usuario”. *Instituto Nacional de Tecnología Agropecuaria*, 2023. Recuperado de: <https://inta.gob.ar/documentos/caprinapp-manual-de-usuario> [25/03/2023]
- [93] “Utilización de drones en las actividades agropecuarias y científicas”. *INTA Chubut*, 2022. Recuperado de: https://www.youtube.com/watch?v=yB3lZfnZHC4&ab_channel=INTAChubut [25/03/2023]
- [94] AGTECH. “Trazabilidad y digitalización, herramientas clave para una ganadería sustentable”. *INTA Informa*, 2022. Recuperado de: <https://intainforma.inta.gob.ar/trazabilidad-y-digitalizacion-herramientas-clave-para-una-ganaderia-sustentable/> [25/03/2023]
- [95] Andrea Maria Lagomarsino & Gabriela Adriana Tallarico. “La Ganadería de Precisión es aporte y crecimiento para el futuro ganadero”. *Instituto Nacional de Tecnología Agropecuaria*, 2020. Recuperado de: <https://inta.gob.ar/documentos/la-ganaderia-de-precision-es-aporte-y-crecimiento-para-el-futuro-ganadero> [25/03/2023]
- [96] J. Lachman & A. López. “Innovation obstacles in an emerging high tech sector: The case of precision agriculture in Argentina”. *Management Research*, vol. 17, num. 4, 2019. DOI: 10.1108/MRJIAM-11-2018-0883. Recuperado de: <https://ri.conicet.gov.ar/handle/11336/119892> [26/03/2023]
- [97] J. Rodríguez Alvarez et al. “Estimating Body Condition Score in dairy cows from depth images using Convolutional Neural Networks, transfer learning and model ensembling techniques”. *Agronomy (Basel)*, vol. 9, núm. 2, 2019. DOI: 10.3390/agronomy9020090. Recuperado de: <https://www.mdpi.com/2073-4395/9/2/90> [26/03/2023]
- [98] A. L. Diedrichs, F. Bromberg, D. Dujovne, K. Brun-Laguna & T. Watteyne. “Prediction of frost events using machine learning and IoT sensing devices”. *IEEE Internet Things Journal*, vol. 5, num. 6, 2018. DOI: 10.1109/JIOT.2018.2867333. Recuperado de: <https://ieeexplore.ieee.org/document/8447058> [26/03/2023]
- [99] J. Rodríguez Alvarez et al. “Body condition estimation on cows from depth images using Convolutional Neural Network”. *Computers and Electronics in Agriculture*, vol. 155, 2018. DOI: 10.1016/j.compag.2018.09.039. Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S0168169917315788> [26/03/2023]
- [100] L. Casali, G. Rubio & J. M. Herrera. “Drought and temperature limit tropical and temperate maize hybrids differently in a subtropical region”. *Agronomy for Sustainable Development*, vol. 38, num. 5, 2018. DOI: 10.1007/s13593-018-0516-4. Recuperado de: <https://link.springer.com/article/10.1007/s13593-018-0516-4> [26/03/2023]
- [101] G. R. Chantre, M. R. Vigna, J. P. Renzi & A. M. Blanco. “A flexible and practical approach for real-time weed emergence prediction based on Artificial Neural Networks”. *Biosystems Engineering*, vol. 170, 2018. DOI: 10.1016/j.biosystemseng.2018.03.014. Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S1537511017306335> [26/03/2023]
- [102] K. Brun-Laguna et al. “Using SmartMesh IP in smart agriculture and smart building applications”. *Computer Communications*, vol. 121, 2018. DOI:

- 10.1016/j.comcom.2018.03.010. Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S0140366417305212> [26/03/2023]
- [103] J. O. Chelotti, S. R. Vanrell, J. R. Galli, L. L. Giovanini & H. L. Rufiner. “A pattern recognition approach for detecting and classifying jaw movements in grazing cattle”. *Computers and Electronics in Agriculture*, vol. 145, 2018. DOI: 10.1016/j.compag.2017.12.013. Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S0168169917302752> [26/03/2023]
- [104] A. Moussaid et al. “Machine learning applied to tree crop yield prediction using field data and satellite imagery: A case study in a citrus orchard”. *Informatics (MDPI)*, vol. 9, núm. 4, 2022. DOI: 10.3390/informatics9040080. Recuperado de: <https://www.mdpi.com/2227-9709/9/4/80> [26/03/2023]
- [105] M. L. Lima, A. Romanelli & H. E. Massone. “Assessing groundwater pollution hazard changes under different socio-economic and environmental scenarios in an agricultural watershed”. *Science of The Total Environment*, vol. 530-531, 2015. DOI: 10.1016/j.scitotenv.2015.05.026. Recuperado de: <https://www.sciencedirect.com/science/article/abs/pii/S004896971530084X> [26/03/2023]
- [106] Raul Jaime Freixa & Cristian Ramon Nuñez. “Uso de drones como herramienta para el sector agropecuario”. *Instituto Nacional de Tecnología Agropecuaria*, 2019. Recuperado de: <https://inta.gob.ar/noticias/uso-de-drones-como-herramienta-para-el-sector-agropecuario> [27/03/2023]
- [107] Ricardo Garro & Gabriela Tallarico. “Ganadería de precisión: innovaciones tecnológicas que agregan valor a la ganadería”. *Revista Idia 21, Ediciones INTA*, 2022. Recuperado de: https://inta.gob.ar/sites/default/files/pubidia22_ano2_n1mayo_v4_p.21-27.pdf [27/03/2023]
- [108] “Beneficios la inteligencia artificial bot Telegram”. *Aunoa*, 2021. Recuperado de: <https://aunoa.ai/beneficios-de-activar-un-chatbot-en-telegram/> [28/03/2023]
- [109] “¿Qué es Python?”. *Amazon*. Recuperado de: <https://aws.amazon.com/es/what-is/python/> [28/03/2023]
- [110] Wikipedia contributors. “Django (framework)”. *Wikipedia, The Free Encyclopedia*. Recuperado de: [https://es.wikipedia.org/wiki/Django_\(framework\)](https://es.wikipedia.org/wiki/Django_(framework)) [28/03/2023]
- [111] Sara López Mora. “Firebase: qué es, para qué sirve, funcionalidades y ventajas”. *DIGITAL55*, 2020. Recuperado de: <https://digital55.com/blog/que-es-firebase-funcionalidades-ventajas-conclusiones/> [28/03/2023]
- [112] “Introduction to celery — celery 5.2.7 documentation”. *Celeryq*. Recuperado de: <https://docs.celeryq.dev/en/stable/index.html> [28/03/2023]
- [113] Héctor Canto. “Qué es Celery: Introducción y primeros pasos”. *Openwebinars*, 2021. Recuperado de: <https://openwebinars.net/blog/que-es-celery-introduccion-y-primeros-pasos/> [28/03/2023]
- [114] Dmytro Striletskyi. “to django-telegram-login”. *Dmytro Striletskyi*, 2018. Recuperado de: <https://django-telegram-login.readthedocs.io/en/latest/index.html> [28/03/2023]
- [115] “¿Qué es Redis?”. *Amazon*. Recuperado de: <https://aws.amazon.com/es/redis/> [28/03/2023]

- [116] “Memcached: explicación sencilla de la memoria en caché”. *IONOS Digital Guide*, 2021. Recuperado de: <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/que-es-memcached/> [28/03/2023]
- [117] Daniel Urrutia. “Qué es Bootstrap”. *Arimetrics*, 2020. Recuperado de: <https://www.arimetrics.com/glosario-digital/bootstrap> [28/03/2023]
- [118] Wikipedia contributors. “JavaScript”. *Wikipedia, The Free Encyclopedia*. Recuperado de: <https://es.wikipedia.org/wiki/JavaScript> [28/03/2023]
- [119] Miguel Parada. “Qué es jQuery”. *Openwebinars*, 2019. Recuperado de: <https://openwebinars.net/blog/que-es-jquery/> [28/03/2023]
- [120] “OpenWeather”. *Openweathermap*. Recuperado de: <https://openweathermap.org/> [28/03/2023]
- [121] “The Ultimate Guide To Speech Recognition With Python”. Recuperado de: <https://realpython.com/python-speech-recognition/> [01/04/2023]
- [122] “Introducción a spaCy (Industrial-Strength Natural Language Processing en Python)”. Recuperado de: <https://todobi.com/introduccion-a-spacy/> [01/04/2023]
- [123] A. Naser y G. Concha. “Datos abiertos: un nuevo desafío para los gobiernos de la región”. *CEPAL*, 2012. Recuperado de: <https://www.cepal.org/es/publicaciones/7331-datos-abiertos-un-nuevo-desafio-gobiernos-la-region> [03/04/2023]
- [124] Dirección Nacional de Datos e Información Pública y Academia de Diseño de Políticas Públicas. “La apertura de datos en gobiernos provinciales y locales”. *LABgobar, Subsecretaría de Innovación Pública y Gobierno Abierto, Secretaría de Modernización*, 2019. Recuperado de: https://www.argentina.gob.ar/sites/default/files/datos_-_guia_de_apertura_para_gobiernos_provinciales_y_locales.pdf [03/04/2023]
- [125] “Guía para la apertura de datos”. *Conocimiento abierto*. Recuperado de: <https://www.conocimientoabierto.org/guia-para-la-apertura-de-datos.pdf> [03/04/2023]
- [126] Verónica Xhardez. “Datos abiertos en la Argentina: desafíos para la apertura y reutilización de datos públicos de gobierno”. *Centro interdisciplinario de estudios en ciencia, tecnología e innovación*, 2020. Recuperado de: http://www.ciecti.org.ar/wp-content/uploads/2021/04/DT22-Datos-abiertos_FINAL.pdf [03/04/2023]
- [127] “Carta internacional de datos abiertos - Principios”. *International Open Data Charter*, 2015. Recuperado de: <https://opendatacharter.net/principles-es/> [03/04/2023]
- [128] <https://datos.gob.ar/> [03/04/2023]
- [129] “Movimientos ovinos 2013 a 2018”. *Datos Argentina*. Recuperado de: https://www.datos.gob.ar/dataset/agroindustria-senasa---movimientos-ovinos/archivo/agroindustria_ca6d28b3-058a-4618-9ac0-741eff5d62c6 [03/04/2023]
- [130] “Existencias ovinas 1875-1964”. *Datos Argentina*. Recuperado de: https://www.datos.gob.ar/dataset/agroindustria-ganaderia---existencias-ovinas-desde-1875-al-2017/archivo/agroindustria_fb8dbe9d-28b7-4feb-ad0b-c45d24e1d358 [03/04/2023]

- [131] “Existencias ovinas 1965-2017”. *Datos Argentina*. Recuperado de: https://www.datos.gob.ar/dataset/agroindustria-ganaderia---existencias-ovinas-desde-1875-al-2017/archivo/agroindustria_2c7ef5ff-a570-43aa-a61a-ff7d2bb6a8e0 [03/04/2023]
- [132] “Serie de Tiempo de Existencias ovinas desde el año 1875 al 2017”. *Datos Argentina*. Recuperado de: https://www.datos.gob.ar/dataset/agroindustria-ganaderia---existencias-ovinas-desde-1875-al-2017/archivo/agroindustria_704fc523-7f74-4c30-8e0e-fb6338ae6731 [03/04/2023]
- [133] “Ovinos por provincia y EAPs - CNA-02”. *Datos Argentina*. Recuperado de: https://www.datos.gob.ar/dataset/agroindustria-censo---ganaderia-cna-02/archivo/agroindustria_d158d3bf-b09c-436c-8d99-bf7a8923e811 [03/04/2023]
- [134] “Indicadores de Ovinos - Lana y Carne en valores anuales, trimestrales y mensuales”. *Datos Argentina*. Recuperado de: https://www.datos.gob.ar/dataset/ssprys-indicadores-sectoriales-provinciales/archivo/ssprys_01.27 [03/04/2023]
- [135] “Existencias ovinas”. *Datos Argentina*. Recuperado de: https://datos.gob.ar/dataset/agroindustria-senasa---existencias-ovinas/archivo/agroindustria_5dfb3c49-7260-4dc4-afa3-95f3459754a6 [03/04/2023]
- [136] “Ovinos Sector primario”. *Argentina.gob.ar*, 2018. Recuperado de: <https://www.argentina.gob.ar/senasa/ovinos-sector-primario> [03/04/2023]
- [137] “Movimiento de subproductos animales 2013 a 2018”. *Datos Argentina*. Recuperado de: https://datos.gob.ar/he/dataset/agroindustria-senasa---movimientos-subproductos-animales/archivo/agroindustria_7d068012-5e8d-4f5f-8f48-e0281cd28230 [04/04/2023]
- [138] “Sistema de Información Patagonia Sur”. *Instituto Nacional de Tecnología Agropecuaria*. Recuperado de: <https://sipas.inta.gob.ar/> [04/04/2023]
- [139] <http://www.funbapa.org.ar/funbapa/servicios-e-informacion/estadisticas-de-ingresoegreso-de-productos/> [04/04/2023]
- [140] “Estadísticas de Ingreso/Egreso de Productos”. *Fundación Barrera Zoofitosanitaria Patagónica*. Recuperado de: <https://www.flasite.com/index.php/es/publicaciones/estadistica-anual-2018> [04/04/2023]
- [141] “Seaborn: Statistical data visualization — seaborn 0.12.2 documentation”. *Seaborn*. Recuperado de: <https://seaborn.pydata.org/> [04/04/2023]
- [142] “The selenium browser automation project”. *Selenium*. Recuperado de: <https://www.selenium.dev/documentation/> [04/04/2023]
- [143] “¿Qué es la preparación de datos?”. *Amazon*. Recuperado de: <https://aws.amazon.com/es/what-is/data-preparation/> [04/04/2023]
- [144] S. M. López & S. Li. “Artesanía textil mapuche en el NO de Chubut. Prácticas de intercambio que aportan al fortalecimiento de la cadena de valor”. *Asociación Latinoamericana de Sociología, Buenos Aires*, 2009. Recuperado de: <https://cdsa.aacademica.org/000-062/1467.pdf> [05/04/2023]
- [145] Ginés Santiago de Gea. “El Ganado Lanar en la Argentina”. *Edit. U.N.R.C, Capítulo 1*, 2007. Recuperado de:

- https://www.produccion-animal.com.ar/produccion_ovina/produccion_ovina/000-ganado_lan_ar_en_argentina_libro/05-captitulo_1.pdf [05/04/2023]
- [146] F. A. A. Molina, M. V. Córdoba & A. M. Alcalá. “Caracterización de la lana del merino autóctono español”. *Junta de Andalucía, Consejería de Agricultura y Pesca*, 2004. Recuperado de: https://www.juntadeandalucia.es/export/drupaljda/1337165058lana_merino.pdf [05/04/2023]
- [147] “Manual de manejo ovino”. *INIA*, 2017. Recuperado de: https://www.produccion-animal.com.ar/produccion_ovina/produccion_ovina/276-Manual_manejo.pdf [05/04/2023]
- [148] “Estadísticas ganaderas 2014”. *Subsecretaría de Ganadería y Agricultura Chubut*, 2014. Recuperado de: <https://sipas.inta.gob.ar/sites/default/files/archivos/ESTADISTICAS%20GANADERAS%202014%20CHUBUT.pdf> [06/04/2023]
- [149] Wikipedia contributors. “Sistemas de soporte a decisiones”. *Wikipedia, The Free Encyclopedia*. Recuperado de: https://es.wikipedia.org/wiki/Sistemas_de_soporte_a_decisiones [14/04/2023]
- [150] Richard I. Levin & David S. Rubin. “Estadística para administración y economía”. Pearson Education, 2004. ISBN: 0-13-476292-4. Recuperado de: https://www.economicas.unsa.edu.ar/afinan/informacion_general/book/ebooks/Estadistica%20para%20administracion%20y%20economia%20Levin%20et%20al.pdf [23/04/2023]
- [151] F. Cardona. “Gráficos Estadísticos”. *Excel Para Todos*, 2022. Recuperado de: <https://excelparatodos.com/graficos-estadisticos/> [23/04/2023]
- [152] “Big Data: ¿En qué consiste? Su importancia, desafíos y gobernabilidad”. *Powerdata*. Recuperado de: <https://www.powerdata.es/big-data> [25/04/2023]
- [153] “¿Qué es el 5G?”. Recuperado de: https://enacom.gob.ar/-que-es-el-5g-_p4903 [25/04/2023]

Anexo I

En este anexo se presentarán algunas secciones que complementan y enriquecen el contenido de la presente tesina. Las secciones son documentos que contienen información adicional que puede resultar de interés para los lectores, pero que no fue incluida en el cuerpo principal del trabajo. Estas secciones están compuestas por archivos Jupyter Notebook los cuales se han desarrollado durante el proceso de investigación y contienen información detallada sobre la metodología utilizada, el análisis de datos y los resultados obtenidos. Es importante destacar que los notebooks han sido diseñados para poder leerse de manera independiente a la tesis y ofrecen una continuidad de lectura para aquellos lectores que quieran profundizar en los detalles técnicos y metodológicos del estudio. En las secciones se incluyen también ejemplos de código y visualizaciones que permiten ilustrar de manera práctica algunos de los conceptos y resultados presentados en la tesina.

1.1. Scrapers estaciones meteorológicas

La siguiente celda importa varios módulos y bibliotecas de Python necesarios para llevar a cabo tareas de web scraping y manejo de datos. En resumen:

- **selenium** : una biblioteca de Python para automatizar la navegación web.
- **pandas** : una biblioteca de Python para la manipulación de datos.
- **unicodedata** : una biblioteca de Python para trabajar con caracteres Unicode.
- **logging** : una biblioteca de Python para la generación de registros (logs).
- **time** : una biblioteca de Python para el manejo del tiempo.
- **os** : una biblioteca de Python para interactuar con el sistema operativo.

Además, se define una excepción personalizada para detectar cuándo un elemento no existe en la página web (**NoSuchElementException**), se configura el driver de Chrome para que se descargue automáticamente (**ChromeDriverManager**), se establecen opciones de configuración para el driver de Chrome (**Options**), se importa el servicio para ejecutar el driver de Chrome (**Service**) y se define el formato de fecha y hora utilizado en los registros (**datetime**).

In [1]:

```
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import NoSuchElementException
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium import webdriver
from datetime import datetime

import pandas as pd
import unicodedata
import logging
import time
import os
```

La siguiente función **log_file_name()** retorna una cadena de texto que representa la ruta del archivo de registro (log) de la información producida por el proceso de scraping. La ruta es construida utilizando el módulo **os** para acceder a la variable de entorno **HOME** del sistema operativo y se concatenan diferentes partes de la ruta para crear la ubicación del archivo de registro. El nombre del archivo incluye la fecha y hora actual en el formato "YYYY-MM-DD_HH:MM:SS" para identificar de forma única cada archivo generado.

In [2]:

```
def log_file_name():
    timestamp = datetime.now().strftime("%Y-%m-%d_%H:%M:%S")
```

```
        return f"
{os.getenv('HOME')}/Escritorio/Tesina/Analisis_4.0/datos_clima/logs/
sipas-{timestamp}.log"
```

La función `pasar_a_texto_elemento` toma como entrada un objeto `elemento` y utiliza el método `get_attribute` para obtener el texto contenido en el atributo `innerText`. A continuación, utiliza la función `unicodedata.normalize` con el parámetro `'NFKC'` para normalizar el texto, es decir, para eliminar cualquier carácter especial o símbolo que pueda aparecer en el texto, como espacios en blanco adicionales o caracteres unicode no reconocidos. Finalmente, devuelve el texto normalizado como una cadena de caracteres.

In [3]:

```
def pasar_a_texto_elemento(elemento):
    elemento_texto = elemento.get_attribute('innerText')
    return unicodedata.normalize('NFKC', elemento_texto)
```

La función `configurar_web_driver` configura el objeto `webdriver.Chrome` para usarlo en la automatización del navegador web Google Chrome.

Primero, se crea un objeto `Options` para definir las opciones de configuración de Chrome. En este caso, se establece `options.headless = True` para ejecutar el navegador en modo sin cabeza (headless), es decir, sin abrir la interfaz gráfica del navegador, y se establece la dimensión de la ventana con `options.add_argument("--window-size=1920,1200")`.

Luego, se define un objeto `handlers` que contiene un `logging.FileHandler` para escribir los mensajes de registro en un archivo de registro en disco y un `logging.StreamHandler` para mostrar los mensajes en la consola.

Por último, se llama al constructor de `webdriver.Chrome`, se instala el driver de Chrome mediante `ChromeDriverManager().install()`, se pasan las opciones definidas anteriormente y se retorna el objeto `webdriver.Chrome` configurado.

In [4]:

```
def configurar_web_driver():
    options = Options()
    options.headless = True
    options.add_argument("--window-size=1920,1200")
    handlers = [logging.FileHandler(log_file_name()),
logging.StreamHandler()]
    logging.basicConfig(
        format='%(asctime)s - %(message)s',
        level=logging.INFO, handlers=handlers
    )
    return webdriver.Chrome(
        service=Service(ChromeDriverManager().install()),
        options=options
    )
```

Existen 60 estaciones sin embargo cada sonda tenía una ID que no necesariamente se correspondía con el número de sondas disponibles, lo que resultó en algunas sondas con números de ID mayores al total de 60 sondas. Para abordar este problema, se recorrió un rango de números que iba del 1 al 72 para recopilar los datos de cada una de las 60 sondas disponibles.

```
In [5]: numero_estaciones = 71 # De 1-71
```

La función `obtener_datos_para_url` recibe un objeto `driver` y una URL. Luego, itera sobre una serie de números de estación y reemplaza el identificador "idNum" en la URL con el número de estación actual para acceder a la página web correspondiente a cada estación meteorológica.

Una vez en la página web, espera hasta que el elemento de la tabla esté cargado y luego obtiene la cabecera de la tabla si aún no se ha obtenido. Luego, busca la estación meteorológica y los datos correspondientes en la tabla y los agrega a un DataFrame que contiene los datos para todas las estaciones.

La función utiliza la librería `logging` para imprimir información sobre el progreso de la iteración. Además, utiliza la función `pasar_a_texto_elemento` para obtener el texto de cada elemento HTML y normalizarlo en Unicode.

```
In [12]: def obtener_datos_para_url(driver, url):
cabecera_df=['Estación']
datos_df=[]
cantidad_elementos_cabecera = 15
for numero_estacion in range(1,numero_estaciones):
    logging.info(f"Número estación: {numero_estacion}")
    url_estacion = url['url'].replace('idNum',
str(numero_estacion))
    driver.get(url_estacion)
    time.sleep(120)
    if(numero_estacion==1):
        cabecera = driver.find_element(By.XPATH, '//*[@id="Contenido"]/div/table/thead/tr')
        elementos_cabecera = cabecera.find_elements(By.XPATH,
'.//th')
        for elemento in elementos_cabecera:
cabecera_df.append(pasar_a_texto_elemento(elemento))
        try:
            cuerpo = driver.find_element(By.XPATH, '//*[@id="Contenido"]/div/table/tbody[2]')
            estacion =
pasar_a_texto_elemento(driver.find_element(By.XPATH, '//*[@id="Contenido"]/p/a[2]/span'))
            estacion = estacion.replace('Estación meteorologica:
```

```

', ''')

        logging.info(f"Estación: {estacion}")
    except NoSuchElementException:
        logging.info(f"No hay datos")
        continue
    anios_elementos = cuerpo.find_elements(By.XPATH, '.*tr')
    for anio_elemento in anios_elementos:
        logging.info(f"Entre/Cambie de año")
        datos_anio = [estacion]
        for indice in range(1, cantidad_elementos_cabecera):

datos_anio.append(pasar_a_texto_elemento(anio_elemento.find_element(
f"../td[{indice}]")))
        datos_df.append(datos_anio)
        time.sleep(30)

    return pd.DataFrame(datos_df, columns=cabecera_df)

```

Este código define una lista llamada `urls` que contiene cinco diccionarios. Cada diccionario contiene dos claves: `'url'` y `'filename'`. La clave `'url'` contiene una URL que se utilizará más adelante en el código para obtener datos meteorológicos de una estación específica. La clave `'filename'` contiene el nombre de archivo que se utilizará para guardar los datos obtenidos de esa URL.

In [14]:

```

urls = [
{
'url': 'https://sipas.inta.gob.ar/?q=agrometeorologia-estadist-
temp-min&idEstacion=idNum',
'filename': 'temperatura_minima_estaciones.csv'},
{
'url': 'https://sipas.inta.gob.ar/?q=agrometeorologia-estadist-
temp-med&idEstacion=idNum',
'filename': 'temperatura_media_estaciones.csv'},
{
'url': 'https://sipas.inta.gob.ar/?q=agrometeorologia-estadist-
temp-max&idEstacion=idNum',
'filename': 'temperatura_maxima_estaciones.csv'},
{
'url': 'https://sipas.inta.gob.ar/?q=agrometeorologia-estadist-
hum&idEstacion=idNum',
'filename': 'humedad_estaciones.csv'},
{

```

```
'url': 'https://sipas.inta.gob.ar/?q=agrometeorologia-estadist-
lluvia&idEstacion=idNum',
'filename': 'lluvia_estaciones.csv'
}]
```

El siguiente código utiliza un webdriver para automatizar la descarga de datos de varias estaciones meteorológicas y guardar los datos en archivos CSV.

En primer lugar, se llama a la función `configurar_web_driver()` para configurar el webdriver. Luego, se itera sobre una lista de diccionarios `urls`, donde cada elemento del diccionario tiene una URL y un nombre de archivo CSV asociado.

Para cada URL en la lista, se llama a la función `obtener_datos_para_url(driver, url)` para extraer los datos de la página web correspondiente utilizando el webdriver. Los datos se almacenan en un DataFrame de pandas y luego se guardan en un archivo CSV con el nombre especificado en el diccionario.

Después de procesar todas las URL en la lista, el webdriver se cierra llamando a `driver.quit()`.

```
In [13]: driver = configurar_web_driver()

for url in urls:
    df_final = obtener_datos_para_url(driver, url)
    df_final.to_csv(url['filename'], index=None)

driver.quit()
```

```
[WDM] - ===== WebDriver manager =====
2023-04-05 09:09:26,895 - ===== WebDriver manager =====
[WDM] - Current google-chrome version is 111.0.5563
2023-04-05 09:09:26,926 - Current google-chrome version is 111.0.5563
[WDM] - Get LATEST chromedriver version for 111.0.5563 google-chrome
2023-04-05 09:09:26,927 - Get LATEST chromedriver version for 111.0.5563 goog
le-chrome
[WDM] - Driver [/home/lucila/.wdm/drivers/chromedriver/linux64/111.0.5563.64/
chromedriver] found in cache
2023-04-05 09:09:27,062 - Driver [/home/lucila/.wdm/drivers/chromedriver/linu
x64/111.0.5563.64/chromedriver] found in cache
2023-04-05 09:09:27,683 - Número estación: 1
2023-04-05 09:11:32,582 - Estación: Trelew
2023-04-05 09:11:32,590 - Entre/Cambie de año
2023-04-05 09:11:32,730 - Entre/Cambie de año
2023-04-05 09:11:32,871 - Entre/Cambie de año
2023-04-05 09:11:33,015 - Entre/Cambie de año
2023-04-05 09:11:33,157 - Entre/Cambie de año
2023-04-05 09:11:33,302 - Entre/Cambie de año
2023-04-05 09:11:33,449 - Entre/Cambie de año
2023-04-05 09:11:33,605 - Entre/Cambie de año
2023-04-05 09:11:33,764 - Entre/Cambie de año
2023-04-05 09:11:33,925 - Entre/Cambie de año
```

1.2. Analizando dataset: Indicadores de Ovinos

Introducción

Este dataset se obtuvo de [Datos Argentina](#). Se trata de un dataset muy poblado que contiene información sobre existencias ovinas, producción de lana y producción de carne tomadas de distintas fuentes como [MAGyP](#), [FAO](#), [FLA](#) y [SENASA](#).

Nos encontramos con indicadores anuales, trimestrales y mensuales. No existe documentación que nos proporcione información sobre el dataset, así que vamos a cargarlo para ver su contenido y comenzar el análisis.

Análisis

Importar librerías

Importamos las librerías de Python que necesitamos para analizar el dataset.

```
In [1]: import pandas as pd
```

Cargar dataset

Cargamos el dataset y observamos su contenido.

```
In [2]: df = pd.read_csv('datos/ovinos-lana-y-carne.csv', sep=";",  
encoding='latin-1')  
df
```

```
Out[2]:
```

	sector_id	sector_nombre	variable_id	actividad_producto_nombre	indicador	unidad_de_medida	fu
0	1	Ovinos - Lana y Carne	1	Carne ovina	Consumo aparente	tn res con hueso	MA
1	1	Ovinos - Lana y Carne	1	Carne ovina	Consumo aparente	tn res con hueso	MA
2	1	Ovinos - Lana y Carne	1	Carne ovina	Consumo aparente	tn res con hueso	MA
3	1	Ovinos - Lana y Carne	1	Carne ovina	Consumo aparente	tn res con hueso	MA
4	1	Ovinos - Lana y Carne	1	Carne ovina	Consumo aparente	tn res con hueso	MA
...	...	...	...	...	...	...	...
8050	1	Ovinos - Lana y Carne	18	Ovino	Stock al 31-03	cabezas	SEN
8051	1	Ovinos - Lana y Carne	18	Ovino	Stock al 31-03	cabezas	SEN
8052	1	Ovinos - Lana y Carne	18	Ovino	Stock al 31-03	cabezas	SEN

	sector_id	sector_nombre	variable_id	actividad_producto_nombre	indicador	unidad_de_medida	fu
8053	1	Ovinos - Lana y Carne	18	Ovino	Stock al 31-03	cabezas	SEN
8054	1	Ovinos - Lana y Carne	18	Ovino	Stock al 31-03	cabezas	SEN

8055 rows × 14 columns

Como se mencionó anteriormente se trata de un conjunto de datos un tanto mezclado, y será necesario analizarlo en detenimiento para encontrar los datos que necesitamos. Es un dataset de 8055 filas y 14 columnas, miremos las columnas que trae:

In [3]:

```
df.columns
```

Out[3]:

```
Index(['sector_id', 'sector_nombre', 'variable_id',
      'actividad_producto_nombre', 'indicador', 'unidad_de_medida', 'fuente',
      'frecuencia_nombre', 'cobertura_nombre', 'alcance_tipo', 'alcance_id',
      'alcance_nombre', 'indice_tiempo', 'valor'],
      dtype='object')
```

La columna que parece decirnos el producto es "actividad_producto_nombre", por lo tanto veamos si en efecto es así:

In [4]:

```
df['actividad_producto_nombre'].unique()
```

Out[4]:

```
array(['Carne ovina', 'Lana fina_ 20 micrones Rinde peine 55%',
      'Lana mediana_ 27 micrones Rinde peine 55%', 'Lana sucia',
      'Lana sucia cruza fina', 'Lana sucia fina', 'Lana sucia gruesa',
      'Lana sucia mediana', 'Ovino'], dtype=object)
```

Tenemos 9 "productos":

- Carne ovina
- Lana fina_ 20 micrones Rinde peine 55%
- Lana mediana_ 27 micrones Rinde peine 55%
- Lana sucia
- Lana sucia cruza fina
- Lana sucia fina
- Lana sucia gruesa
- Lana sucia mediana
- Ovino: Representa las existencias ovinas

Lo siguiente es buscar los productos que nos interesan pero que sean de la provincia del Chubut. Empezamos por la producción de carne.

In [5]:

```
resultado_carne = pd.DataFrame(columns=
['actividad_producto_nombre', 'indicador', 'unidad_de_medida',
'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
'indice_tiempo', 'valor'])
for i in df.index:
```

```

if "Carne ovina" == df['actividad_producto_nombre'][i]:
    resultado_carne = resultado_carne.append({

'actividad_producto_nombre': df['actividad_producto_nombre'][i],
        'indicador':
df['indicador'][i],

'unidad_de_medida':df['unidad_de_medida'][i],

'frecuencia_nombre':df['frecuencia_nombre'][i],

'alcance_tipo':df['alcance_tipo'][i],

'alcance_nombre':df['alcance_nombre'][i],

'indice_tiempo':df['indice_tiempo'][i],
        'valor':df['valor'][i]},),
        ignore_index=True)

resultado_carne

```

```

Out[5]:

```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nor
0	Carne ovina	Consumo aparente	tn res con hueso	Anual	PAIS	Arge
1	Carne ovina	Consumo aparente	tn res con hueso	Anual	PAIS	Arge
2	Carne ovina	Consumo aparente	tn res con hueso	Anual	PAIS	Arge
3	Carne ovina	Consumo aparente	tn res con hueso	Anual	PAIS	Arge
4	Carne ovina	Consumo aparente	tn res con hueso	Anual	PAIS	Arge
...	...	...	...	...	...	...
61	Carne ovina	Producción	tn res con hueso	Anual	PAIS	Arge
62	Carne ovina	Producción	tn res con hueso	Anual	PAIS	Arge
63	Carne ovina	Producción	tn res con hueso	Anual	PAIS	Arge
64	Carne ovina	Producción	tn res con hueso	Anual	PAIS	Arge
65	Carne ovina	Producción	tn res con hueso	Anual	PAIS	Arge

66 rows × 8 columns

Son 66 filas, entonces lo que hacemos es buscar si existen datos a nivel provincia o departamento

```

In [6]: resultado_carne['alcance_tipo'].unique()

```



```
Out[6]: array(['PAIS'], dtype=object)
```

No hay datos especificados por provincia ni por departamento, por lo tanto este dataset no tiene información sobre la producción de carne de Chubut. Pasemos al siguiente producto que sería la lana fina de 20 micrones.

```
In [7]: resultado_lana_20 = pd.DataFrame(columns=[
    'actividad_producto_nombre', 'indicador', 'unidad_de_medida',
    'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
    'indice_tiempo', 'valor'])
for i in df.index:
    if "Lana fina_ 20 micrones Rinde peine 55%" ==
df['actividad_producto_nombre'][i]:
        resultado_lana_20 = resultado_lana_20.append({

    'actividad_producto_nombre': df['actividad_producto_nombre'][i],
        'indicador':
df['indicador'][i],

    'unidad_de_medida':df['unidad_de_medida'][i],

    'frecuencia_nombre':df['frecuencia_nombre'][i],

    'alcance_tipo':df['alcance_tipo'][i],

    'alcance_nombre':df['alcance_nombre'][i],

    'indice_tiempo':df['indice_tiempo'][i],
        'valor':df['valor'][i],
        },
        ignore_index=True)

resultado_lana_20
```

```
Out[7]:
```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_no
0	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
1	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
2	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
3	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_no
4	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
...	...	...	...	...	...	...
210	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
211	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
212	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
213	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
214	Lana fina_ 20 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen

215 rows × 8 columns

Son 215 filas, entonces lo que hacemos es buscar si existen datos a nivel provincia o departamento

```
In [8]: resultado_lana_20['alcance_tipo'].unique()
```

```
Out[8]: array(['PAIS'], dtype=object)
```

No hay datos especificados por provincia ni por departamento, por lo tanto este dataset no tiene información sobre la lana fina de 20 micrones de Chubut. Pasemos al siguiente producto que sería la lana mediana de 27 micrones.

```
In [9]: resultado_lana_27 = pd.DataFrame(columns=
['actividad_producto_nombre', 'indicador', 'unidad_de_medida',
'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
'indice_tiempo', 'valor'])
for i in df.index:
    if "Lana mediana_ 27 micrones Rinde peine 55%" ==
df['actividad_producto_nombre'][i]:
        resultado_lana_27 = resultado_lana_27.append({
'actividad_producto_nombre': df['actividad_producto_nombre'][i],
'indicador':
df['indicador'][i],
'unidad_de_medida':df['unidad_de_medida'][i],
```

```

'frecuencia_nombre':df['frecuencia_nombre'][i],

'alcance_tipo':df['alcance_tipo'][i],

'alcance_nombre':df['alcance_nombre'][i],

'indice_tiempo':df['indice_tiempo'][i],

                                'valor':df['valor'][i],
                                },
                                ignore_index=True)

resultado_lana_27

```

```

Out[9]:

```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_no
0	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
1	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
2	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
3	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
4	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
...	...	...	...	...	...	...
210	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
211	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
212	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
213	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen
214	Lana mediana_ 27 micrones Rinde peine 55%	Precio interno_de referencia	US\$	Mensual	PAIS	Argen

215 rows × 8 columns

Son 215 filas, entonces lo que hacemos es buscar si existen datos a nivel provincia o departamento

```

In [10]: resultado_lana_27['alcance_tipo'].unique()

```

```
Out[10]: array(['PAIS'], dtype=object)
```

No hay datos especificados por provincia ni por departamento, por lo tanto este dataset no tiene información sobre la lana mediana de 27 micrones de Chubut. Pasemos al siguiente producto que sería la lana sucia.

```
In [11]: resultado_lana_sucia = pd.DataFrame(columns=[
    'actividad_producto_nombre', 'indicador', 'unidad_de_medida',
    'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
    'indice_tiempo', 'valor'])
for i in df.index:
    if "Lana sucia" == df['actividad_producto_nombre'][i]:
        resultado_lana_sucia = resultado_lana_sucia.append({
            'actividad_producto_nombre': df['actividad_producto_nombre'][i],
            'indicador':
df['indicador'][i],
            'unidad_de_medida':df['unidad_de_medida'][i],
            'frecuencia_nombre':df['frecuencia_nombre'][i],
            'alcance_tipo':df['alcance_tipo'][i],
            'alcance_nombre':df['alcance_nombre'][i],
            'indice_tiempo':df['indice_tiempo'][i],
            'valor':df['valor'][i],
        },
        ignore_index=True)
resultado_lana_sucia
```

```
Out[11]:
```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
0	Lana sucia	Producción	tn	Anual	MUNDO	
1	Lana sucia	Producción	tn	Anual	MUNDO	
2	Lana sucia	Producción	tn	Anual	MUNDO	
3	Lana sucia	Producción	tn	Anual	MUNDO	
4	Lana sucia	Producción	tn	Anual	MUNDO	
...	...	...	...	...	...	...
477	Lana sucia	Producción	tn	Anual	PROVINCIA	INDETERMINADO
478	Lana sucia	Producción	tn	Anual	PROVINCIA	INDETERMINADO
479	Lana sucia	Producción	tn	Anual	PROVINCIA	INDETERMINADO

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
480	Lana sucia	Producción	tn	Anual	PROVINCIA	INDETERMINADA
481	Lana sucia	Producción	tn	Anual	PROVINCIA	INDETERMINADA

482 rows × 8 columns

Son 482 filas, entonces lo que hacemos es buscar si existen datos a nivel provincia o departamento

```
In [12]: resultado_lana_sucia['alcance_tipo'].unique()
```

```
Out[12]: array(['MUNDO', 'PAIS', 'PROVINCIA'], dtype=object)
```

Hay datos especificados por provincia, por lo tanto buscamos si hay de Chubut.

```
In [13]: resultado_lana_sucia['alcance_nombre'].unique()
```

```
Out[13]: array(['MUNDO', 'Argentina', 'BUENOS AIRES', 'CATAMARCA', 'CORDOBA',
                'CORRIENTES', 'CHACO', 'CHUBUT', 'ENTRE RIOS', 'FORMOSA', 'JUJUY',
                'LA PAMPA', 'LA RIOJA', 'MENDOZA', 'MISSIONES', 'NEUQUEN',
                'RIO NEGRO', 'SALTA', 'SAN JUAN', 'SAN LUIS', 'SANTA CRUZ',
                'SANTA FE', 'SANTIAGO DEL ESTERO', 'TUCUMAN', 'TIERRA DEL FUEGO',
                'INDETERMINADA'], dtype=object)
```

Efectivamente hay datos sobre la producción de lana sucia en Chubut, por lo tanto filtramos esos datos y los guardamos en un archivo csv.

```
In [14]: resultado_lana_sucia_chubut = pd.DataFrame(columns=
            ['actividad_producto_nombre', 'indicador', 'unidad_de_medida',
            'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
            'indice_tiempo', 'valor'])
            for i in resultado_lana_sucia.index:
                if "CHUBUT" == resultado_lana_sucia['alcance_nombre'][i]:
                    resultado_lana_sucia_chubut =
                    resultado_lana_sucia_chubut.append({

                        'actividad_producto_nombre':
                        resultado_lana_sucia['actividad_producto_nombre'][i],
                                                'indicador':
                        resultado_lana_sucia['indicador'][i],

                        'unidad_de_medida': resultado_lana_sucia['unidad_de_medida'][i],

                        'frecuencia_nombre': resultado_lana_sucia['frecuencia_nombre'][i],

                        'alcance_tipo': resultado_lana_sucia['alcance_tipo'][i],

                        'alcance_nombre': resultado_lana_sucia['alcance_nombre'][i],
```

```

'indice_tiempo':resultado_lana_sucia['indice_tiempo'][i],

'valor':resultado_lana_sucia['valor'][i],},

                                ignore_index=True)

resultado_lana_sucia_chubut

```

```

Out[14]:

```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
0	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
1	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
2	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
3	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
4	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
5	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
6	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
7	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
8	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
9	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
10	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
11	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
12	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
13	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
14	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
15	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
16	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
17	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
18	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
19	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
20	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
21	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
22	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU

```

In [15]: resultado_lana_sucia_chubut.to_csv('datos_producidos/indicadores_ovini
2022.csv', index=False)

```

Pasamos al siguiente producto que sería la lana sucia cruza fina.

```

In [16]: resultado_lana_sucia_cruza_fina = pd.DataFrame(columns=
['actividad_producto_nombre', 'indicador', 'unidad_de_medida',
'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',

```

```

'indice_tiempo', 'valor'])
for i in df.index:
    if "Lana sucia cruza fina" == df['actividad_producto_nombre']
[i]:
        resultado_lana_sucia_cruza_fina =
resultado_lana_sucia_cruza_fina.append({

'actividad_producto_nombre': df['actividad_producto_nombre'][i],
        'indicador':
df['indicador'][i],

'unidad_de_medida':df['unidad_de_medida'][i],

'frecuencia_nombre':df['frecuencia_nombre'][i],

'alcance_tipo':df['alcance_tipo'][i],

'alcance_nombre':df['alcance_nombre'][i],

'indice_tiempo':df['indice_tiempo'][i],
        'valor':df['valor'][i],
        },
        ignore_index=True)

resultado_lana_sucia_cruza_fina

```

Out[16]:

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nor
0	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
1	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
2	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
3	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
4	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
5	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
6	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
7	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
8	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nor
9	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
10	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
11	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
12	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
13	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
14	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge
15	Lana sucia cruza fina	Producción por finura	tn	Anual	PAIS	Arge

Son 15 filas por lo tanto ya se observa que solo tiene alcance a nivel país. Pasamos al siguiente producto que sería la lana sucia fina.

```
In [17]: resultado_lana_sucia_fina = pd.DataFrame(columns=[
    'actividad_producto_nombre', 'indicador', 'unidad_de_medida',
    'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
    'indice_tiempo', 'valor'])
for i in df.index:
    if "Lana sucia fina" == df['actividad_producto_nombre'][i]:
        resultado_lana_sucia_fina =
resultado_lana_sucia_fina.append({

    'actividad_producto_nombre': df['actividad_producto_nombre'][i],
        'indicador':
df['indicador'][i],

    'unidad_de_medida':df['unidad_de_medida'][i],

    'frecuencia_nombre':df['frecuencia_nombre'][i],

    'alcance_tipo':df['alcance_tipo'][i],

    'alcance_nombre':df['alcance_nombre'][i],

    'indice_tiempo':df['indice_tiempo'][i],
        'valor':df['valor'][i],
    },
```



```
ignore_index=True)
```

```
resultado_lana_sucia_fina
```

```
Out[17]:
```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
0	Lana sucia fina	Producción por finura	tn	Anual	PAIS	Arg
1	Lana sucia fina	Producción por finura	tn	Anual	PAIS	Arg
2	Lana sucia fina	Producción por finura	tn	Anual	PAIS	Arg
3	Lana sucia fina	Producción por finura	tn	Anual	PAIS	Arg
4	Lana sucia fina	Producción por finura	tn	Anual	PAIS	Arg
...	...	...	...	...	...	...
226	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
227	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
228	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
229	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
230	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO

231 rows × 8 columns

Son 231 filas, entonces lo que hacemos es buscar si existen datos a nivel provincia o departamento

```
In [18]: resultado_lana_sucia_fina['alcance_tipo'].unique()
```

```
Out[18]: array(['PAIS', 'PROVINCIA'], dtype=object)
```

Hay datos especificados por provincia, por lo tanto busquemos si hay de Chubut.

```
In [19]: resultado_lana_sucia_fina['alcance_nombre'].unique()
```

```
Out[19]: array(['Argentina', 'BUENOS AIRES', 'CATAMARCA', 'CORDOBA', 'CORRIENTES',  
                'CHACO', 'CHUBUT', 'ENTRE RIOS', 'FORMOSA', 'JUJUY', 'LA PAMPA',  
                'LA RIOJA', 'MENDOZA', 'MISSIONES', 'NEUQUEN', 'RIO NEGRO', 'SALTA',  
                'SAN JUAN', 'SAN LUIS', 'SANTA CRUZ', 'SANTA FE',  
                'SANTIAGO DEL ESTERO', 'TUCUMAN', 'TIERRA DEL FUEGO'], dtype=object)
```

Efectivamente hay datos sobre la producción de lana sucia fina en Chubut, por lo tanto filtramos esos datos y los guardamos en un archivo csv.

```
In [20]: resultado_lana_sucia_fina_chubut = pd.DataFrame(columns=[  
    'actividad_producto_nombre', 'indicador', 'unidad_de_medida',  
    'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
```

```

'indice_tiempo', 'valor'])
for i in resultado_lana_sucia_fina.index:
    if "CHUBUT" == resultado_lana_sucia_fina['alcance_nombre'][i]:
        resultado_lana_sucia_fina_chubut =
resultado_lana_sucia_fina_chubut.append({

'actividad_producto_nombre':
resultado_lana_sucia_fina['actividad_producto_nombre'][i],
                                'indicador':
resultado_lana_sucia_fina['indicador'][i],

'unidad_de_medida':resultado_lana_sucia_fina['unidad_de_medida']
[i],

'frecuencia_nombre':resultado_lana_sucia_fina['frecuencia_nombre']
[i],

'alcance_tipo':resultado_lana_sucia_fina['alcance_tipo'][i],

'alcance_nombre':resultado_lana_sucia_fina['alcance_nombre'][i],

'indice_tiempo':resultado_lana_sucia_fina['indice_tiempo'][i],

'valor':resultado_lana_sucia_fina['valor'][i],},
                                ignore_index=True)
resultado_lana_sucia_fina_chubut

```

Out[20]:

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nom
0	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUE
1	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUE
2	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUE
3	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUE
4	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUE
5	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUE
6	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUE
7	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUE

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
8	Lana sucia fina	Producción por finura	tn	Anual	PROVINCIA	CHUBUT

```
In [21]: resultado_lana_sucia_fina_chubut.to_csv('datos_producidos/indicadores_2022.csv', index=False)
```

Pasamos al siguiente producto que seria la lana sucia gruesa.

```
In [22]: resultado_lana_sucia_gruesa = pd.DataFrame(columns=[
    'actividad_producto_nombre', 'indicador', 'unidad_de_medida',
    'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
    'indice_tiempo', 'valor'])
for i in df.index:
    if "Lana sucia gruesa" == df['actividad_producto_nombre'][i]:
        resultado_lana_sucia_gruesa =
resultado_lana_sucia_gruesa.append({
    'actividad_producto_nombre': df['actividad_producto_nombre'][i],
    'indicador':
df['indicador'][i],
    'unidad_de_medida':df['unidad_de_medida'][i],
    'frecuencia_nombre':df['frecuencia_nombre'][i],
    'alcance_tipo':df['alcance_tipo'][i],
    'alcance_nombre':df['alcance_nombre'][i],
    'indice_tiempo':df['indice_tiempo'][i],
    'valor':df['valor'][i],
    },
    ignore_index=True)
resultado_lana_sucia_gruesa
```

```
Out[22]:
```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
0	Lana sucia gruesa	Producción por finura	tn	Anual	PAIS	Arg
1	Lana sucia gruesa	Producción por finura	tn	Anual	PAIS	Arg
2	Lana sucia gruesa	Producción por finura	tn	Anual	PAIS	Arg

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
3	Lana sucia gruesa	Producción por finura	tn	Anual	PAIS	Arg
4	Lana sucia gruesa	Producción por finura	tn	Anual	PAIS	Arg
...	...	...	...	...	...	...
226	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
227	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
228	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
229	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
230	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO

231 rows × 8 columns

Son 231 filas, entonces lo que hacemos es buscar si existen datos a nivel provincia o departamento

```
In [23]: resultado_lana_sucia_gruesa['alcance_tipo'].unique()
```

```
Out[23]: array(['PAIS', 'PROVINCIA'], dtype=object)
```

Hay datos especificados por provincia, por lo tanto buscamos si hay de Chubut.

```
In [24]: resultado_lana_sucia_gruesa['alcance_nombre'].unique()
```

```
Out[24]: array(['Argentina', 'BUENOS AIRES', 'CATAMARCA', 'CORDOBA', 'CORRIENTES',
                'CHACO', 'CHUBUT', 'ENTRE RIOS', 'FORMOSA', 'JUJUY', 'LA PAMPA',
                'LA RIOJA', 'MENDOZA', 'MISIONES', 'NEUQUEN', 'RIO NEGRO', 'SALTA',
                'SAN JUAN', 'SAN LUIS', 'SANTA CRUZ', 'SANTA FE',
                'SANTIAGO DEL ESTERO', 'TUCUMAN', 'TIERRA DEL FUEGO'], dtype=object)
```

Efectivamente hay datos sobre la producción de lana sucia gruesa en Chubut, por lo tanto filtramos esos datos y los guardamos en un archivo csv.

```
In [25]: resultado_lana_sucia_gruesa_chubut = pd.DataFrame(columns=[
    'actividad_producto_nombre', 'indicador', 'unidad_de_medida',
    'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
    'indice_tiempo', 'valor'])
for i in resultado_lana_sucia_gruesa.index:
    if "CHUBUT" == resultado_lana_sucia_gruesa['alcance_nombre']:
        resultado_lana_sucia_gruesa_chubut =
resultado_lana_sucia_gruesa_chubut.append({
    'actividad_producto_nombre':
```

```

resultado_lana_sucia_gruesa['actividad_producto_nombre'][i],
                                'indicador':
resultado_lana_sucia_gruesa['indicador'][i],

'unidad_de_medida':resultado_lana_sucia_gruesa['unidad_de_medida']
[i],

'frecuencia_nombre':resultado_lana_sucia_gruesa['frecuencia_nombre']
[i],

'alcance_tipo':resultado_lana_sucia_gruesa['alcance_tipo'][i],

'alcance_nombre':resultado_lana_sucia_gruesa['alcance_nombre'][i],

'indice_tiempo':resultado_lana_sucia_gruesa['indice_tiempo'][i],

'valor':resultado_lana_sucia_gruesa['valor'][i],},
                                ignore_index=True)
resultado_lana_sucia_gruesa_chubut

```

Out[25]:

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nom
0	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE
1	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE
2	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE
3	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE
4	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE
5	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE
6	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE
7	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE
8	Lana sucia gruesa	Producción por finura	tn	Anual	PROVINCIA	CHUE

In [26]:

```

resultado_lana_sucia_gruesa_chubut.to_csv('datos_producidos/indicados
2022.csv', index=False)

```

Pasamos al siguiente producto que sería la lana sucia mediana.

```
In [27]: resultado_lana_sucia_mediana = pd.DataFrame(columns=
['actividad_producto_nombre', 'indicador', 'unidad_de_medida',
'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
'indice_tiempo', 'valor'])
for i in df.index:
    if "Lana sucia mediana" == df['actividad_producto_nombre'][i]:
        resultado_lana_sucia_mediana =
resultado_lana_sucia_mediana.append({

'actividad_producto_nombre': df['actividad_producto_nombre'][i],
                                'indicador':
df['indicador'][i],

'unidad_de_medida':df['unidad_de_medida'][i],

'frecuencia_nombre':df['frecuencia_nombre'][i],

'alcance_tipo':df['alcance_tipo'][i],

'alcance_nombre':df['alcance_nombre'][i],

'indice_tiempo':df['indice_tiempo'][i],
                                'valor':df['valor'][i],
                                },
                                ignore_index=True)

resultado_lana_sucia_mediana
```

Out[27]:

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
0	Lana sucia mediana	Producción por finura	tn	Anual	PAIS	Arg
1	Lana sucia mediana	Producción por finura	tn	Anual	PAIS	Arg
2	Lana sucia mediana	Producción por finura	tn	Anual	PAIS	Arg
3	Lana sucia mediana	Producción por finura	tn	Anual	PAIS	Arg
4	Lana sucia mediana	Producción por finura	tn	Anual	PAIS	Arg
...	...	...	...	...	...	...
226	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	TIERRA FI
227	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	TIERRA FI

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
228	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
229	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO
230	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	TIERRA DEL FUEGO

231 rows × 8 columns

Son 231 filas, entonces lo que hacemos es buscar si existen datos a nivel provincia o departamento

```
In [28]: resultado_lana_sucia_mediana['alcance_tipo'].unique()
```

```
Out[28]: array(['PAIS', 'PROVINCIA'], dtype=object)
```

Hay datos especificados por provincia, por lo tanto buscamos si hay de Chubut.

```
In [29]: resultado_lana_sucia_mediana['alcance_nombre'].unique()
```

```
Out[29]: array(['Argentina', 'BUENOS AIRES', 'CATAMARCA', 'CORDOBA', 'CORRIENTES',
                'CHACO', 'CHUBUT', 'ENTRE RIOS', 'FORMOSA', 'JUJUY', 'LA PAMPA',
                'LA RIOJA', 'MENDOZA', 'MISSIONES', 'NEUQUEN', 'RIO NEGRO', 'SALTA',
                'SAN JUAN', 'SAN LUIS', 'SANTA CRUZ', 'SANTA FE',
                'SANTIAGO DEL ESTERO', 'TUCUMAN', 'TIERRA DEL FUEGO'], dtype=object)
```

Efectivamente hay datos sobre la producción de lana sucia mediana en Chubut, por lo tanto filtramos esos datos y los guardamos en un archivo csv.

```
In [30]: resultado_lana_sucia_mediana_chubut = pd.DataFrame(columns=
                ['actividad_producto_nombre', 'indicador', 'unidad_de_medida',
                'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
                'indice_tiempo', 'valor'])
for i in resultado_lana_sucia_gruesa.index:
    if "CHUBUT" == resultado_lana_sucia_gruesa['alcance_nombre']:
        resultado_lana_sucia_mediana_chubut =
        resultado_lana_sucia_mediana_chubut.append({
            'actividad_producto_nombre':
            resultado_lana_sucia_mediana['actividad_producto_nombre'][i],
            'indicador':
            resultado_lana_sucia_mediana['indicador'][i],
            'unidad_de_medida': resultado_lana_sucia_mediana['unidad_de_medida']
            [i],
            'frecuencia_nombre': resultado_lana_sucia_mediana['frecuencia_nombre']
```

```
[i],

'alcance_tipo':resultado_lana_sucia_mediana['alcance_tipo'][i],

'alcance_nombre':resultado_lana_sucia_mediana['alcance_nombre']
[i],

'indice_tiempo':resultado_lana_sucia_mediana['indice_tiempo'][i],

'valor':resultado_lana_sucia_mediana['valor'][i],},
                                ignore_index=True)
resultado_lana_sucia_mediana_chubut
```

```
Out[30]:
```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nom
0	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE
1	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE
2	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE
3	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE
4	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE
5	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE
6	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE
7	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE
8	Lana sucia mediana	Producción por finura	tn	Anual	PROVINCIA	CHUE

```
In [31]: resultado_lana_sucia_mediana_chubut.to_csv('datos_producidos/indicac
2022.csv', index=False)
```

Pasamos al siguiente producto que sería la existencia de ovinos.

```
In [32]: resultado_ovinos = pd.DataFrame(columns=
['actividad_producto_nombre','indicador', 'unidad_de_medida',
'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
'indice_tiempo', 'valor'])
for i in df.index:
    if "Ovino" == df['actividad_producto_nombre'][i]:
        resultado_ovinos = resultado_ovinos.append({
```



```

'actividad_producto_nombre': df['actividad_producto_nombre'][i],
                                'indicador':
df['indicador'][i],

'unidad_de_medida':df['unidad_de_medida'][i],

'frecuencia_nombre':df['frecuencia_nombre'][i],

'alcance_tipo':df['alcance_tipo'][i],

'alcance_nombre':df['alcance_nombre'][i],

'indice_tiempo':df['indice_tiempo'][i],
                                'valor':df['valor'][i],
                                },
                                ignore_index=True)

resultado_ovinos

```

Out[32]:

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
0	Ovino	Faena estimada	cabezas	Anual	PAIS	
1	Ovino	Faena estimada	cabezas	Anual	PAIS	
2	Ovino	Faena estimada	cabezas	Anual	PAIS	
3	Ovino	Faena estimada	cabezas	Anual	PAIS	
4	Ovino	Faena estimada	cabezas	Anual	PAIS	
...	...	...	...	...	...	...
6363	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
6364	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
6365	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
6366	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
6367	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	

6368 rows × 8 columns

Son 6368 filas, entonces lo que hacemos es buscar si existen datos a nivel provincia o departamento

```
In [33]: resultado_ovinos['alcance_tipo'].unique()
```

```
Out[33]: array(['PAIS', 'DEPARTAMENTO'], dtype=object)
```

Hay datos especificados por departamento, por lo tanto buscamos si estan los de Chubut.

```
In [34]: departamentos =
resultado_ovinos['alcance_nombre'].unique().tolist()
departamentos_chubut = ["BIEDMA", "CUSHAMEN",
"ESCALANTE", "FUTALEUFU", "GAIMAN", "GASTRE", "LANGUIÑEO",
"MARTIRES", "PASO DE INDIOS", "RAWSON", "RIO SENGUER", "SARMIENTO",
"TEHUELCHES", "TELSEN"]

def compare(lst1, lst2, ini=0):
    n = len(lst2)
    return lst1[ini:ini+n] == lst2

def indexes(lst, x):
    return (i for (i,y) in enumerate(lst) if x==y)

any(compare(departamentos, departamentos_chubut, i) for i in
indexes(departamentos, departamentos_chubut[0]))
```

```
Out[34]: True
```

Segun la función anterior están los departamentos de Chubut por lo tanto hay datos sobre las existencias ovinas en la provincia. Lo siguiente es filtrar esos datos y guardalos en un archivo csv.

```
In [35]: departamentos_chubut = ["BIEDMA", "CUSHAMEN",
"ESCALANTE", "FUTALEUFU", "GAIMAN", "GASTRE", "LANGUIÑEO",
"MARTIRES", "PASO DE INDIOS", "RAWSON", "RIO SENGUER", "SARMIENTO",
"TEHUELCHES", "TELSEN"]

resultado_ovinos_chubut = pd.DataFrame(columns=
['actividad_producto_nombre', 'indicador', 'unidad_de_medida',
'frecuencia_nombre', 'alcance_tipo', 'alcance_nombre',
'indice_tiempo', 'valor'])
for i in resultado_ovinos.index:
    if resultado_ovinos['alcance_nombre'][i] in
departamentos_chubut:
        resultado_ovinos_chubut =
resultado_ovinos_chubut.append({

'actividad_producto_nombre':
```

```

resultado_ovinos['actividad_producto_nombre'][i],
                    'indicador':
resultado_ovinos['indicador'][i],

'unidad_de_medida':resultado_ovinos['unidad_de_medida'][i],

'frecuencia_nombre':resultado_ovinos['frecuencia_nombre'][i],

'alcance_tipo':resultado_ovinos['alcance_tipo'][i],

'alcance_nombre':resultado_ovinos['alcance_nombre'][i],

'indice_tiempo':resultado_ovinos['indice_tiempo'][i],

'valor':resultado_ovinos['valor'][i],},
                    ignore_index=True)
resultado_ovinos_chubut

```

```

Out[35]:

```

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_nombre
0	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
1	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
2	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
3	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
4	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	
...	...	...	...	...	...	...
216	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	SA
217	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	SA
218	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	SA
219	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	SA
220	Ovino	Stock al 31-03	cabezas	Anual	DEPARTAMENTO	SA

221 rows × 8 columns

Son 221 filas, por lo tanto verificamos que esten todos los departamentos de Chubut.

```

In [36]: resultado_ovinos_chubut['alcance_nombre'].unique()

```

```
Out[36]: array(['BIEDMA', 'CUSHAMEN', 'ESCALANTE', 'FUTALEUFU', 'GAIMAN', 'GASTRE',  
              'LANGUIÑEO', 'MARTIRES', 'PASO DE INDIOS', 'RAWSON', 'RIO SENGUER',  
              'SARMIENTO', 'TEHUELCHES', 'TELSEN'], dtype=object)
```

Una vez comprobado que los departamentos estan, los guardamos.

```
In [37]: resultado_ovinos_chubut.to_csv('datos_producidos/indicadores_ovinos/  
2020.csv', index=False)
```

1.3. Analizando dataset existencias ovinas 1965-2017

Introducción

Este dataset se obtuvo de [Datos Argentina](#); y contiene información sobre existencias ovinas dentro del país. Son datos anuales, provinciales, en el período 1965-2017. Y los años faltantes no tienen registros asentados.

Existe una pequeña documentación que nos da información sobre el dataset. Los campos serían los siguientes:

Título de la columna	Tipo de dato	Descripción
fuelle	Texto (string)	fuelle
pais_id	Número entero (integer)	pais_id
pais	Texto (string)	pais
provincia_id	Número entero (integer)	provincia_id
provincia	Texto (string)	provincia
desde_año	Número entero (integer)	desde_año
hasta_año	Número entero (integer)	hasta_año
cantidad (miles de cabezas)	Número decimal (number)	cantidad (miles de cabezas)
fuelle	Texto (string)	fuelle

Y además de los campos también existe información del dataset en sí:

- Responsable: Dirección de Ovinos, Caprinos y Camélidos
- Formato: csv
- Temas: Agroganadería, pesca y forestación
- Frecuencia de actualización: Anualmente
- Fecha de creación: 26 de octubre de 2018
- Último cambio: 30 de abril de 2019

Lo anterior nos da una idea genérica sobre el dataset, de todas formas vamos a cargarlo para ver su contenido y empezar el análisis.

Análisis

Importamos las librerías de python que necesitamos para analizar el dataset

```
In [1]: import pandas as pd
```

La librería pandas nos ayudará a leer y escribir datos en diferentes formatos, seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas, a fusionar y unir datos, entre muchas cosas más.

Cargamos el dataset y observamos su contenido

```
In [2]: df = pd.read_csv('datos/existencias-ovinas-1965-2017.csv',
sep=",", encoding='latin-1')
df
```

```
Out[2]:
```

	pais_id	pais	provincia_id	provincia	desde_a	ño	hasta_año	cantidad	fuentes
0	32	Argentina	6	Buenos Aires	1965	1965	18473.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...	
1	32	Argentina	6	Buenos Aires	1966	1966	17898.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...	
2	32	Argentina	6	Buenos Aires	1967	1967	17322.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...	
3	32	Argentina	6	Buenos Aires	1968	1968	16747.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...	
4	32	Argentina	6	Buenos Aires	1969	1970	16172.0	Censos INDEC	
...	...	...	...	...	...	...	...	...	
1132	32	Argentina	90	Tucuman	2013	2013	15.0	SIGSA - Dirección de Control de Gestión y Prog...	
1133	32	Argentina	90	Tucuman	2014	2014	17.0	SIGSA - Dirección de Control de Gestión y Prog...	
1134	32	Argentina	90	Tucuman	2015	2015	18.0	SIGSA - Dirección de Control de Gestión y Prog...	
1135	32	Argentina	90	Tucuman	2016	2016	17.0	SIGSA - Dirección de Control de Gestión y Prog...	
1136	32	Argentina	90	Tucuman	2017	2017	20.0	SIGSA - Dirección de Control de Gestión y Prog...	

1137 rows × 8 columns

Es un dataset de 1137 filas y 8 columnas, será necesario analizarlo en detenimiento para saber si contiene datos que nos sirvan, miremos las columnas que trae:

```
In [3]: df.columns
```

```
Out[3]: Index(['pais_id', 'pais', 'provincia_id', 'provincia', 'desde_año',
'hasta_año', 'cantidad', 'fuentes'],
dtype='object')
```

Comprobemos si este dataset tiene datos de Chubut.

```
In [4]: df['provincia'].unique()
```

```
Out[4]: array(['Buenos Aires', 'Catamarca', 'Chaco', 'Chubut', 'Cordoba',
        'Corrientes', 'Entre Rios', 'Formosa', 'Jujuy', 'La Pampa',
        'La Rioja', 'Mendoza', 'Misiones', 'Neuquen', 'Rio Negro', 'Salta',
        'San Juan', 'San Luis', 'Santa Cruz', 'Santa Fe',
        'Santiago Del Estero', 'Santiago del Estero', 'Tierra Del Fuego',
        'Tierra del Fuego', 'Tucuman'], dtype=object)
```

Efectivamente hay datos de Chubut, vamos a filtrarlos para verlos mejor:

```
In [5]: resultado_existencias_chubut = pd.DataFrame(columns=
        ['provincia', 'desde_año', 'hasta_año', 'cantidad', 'fuente'])
        for i in df.index:
            if "Chubut" == df['provincia'][i]:
                resultado_existencias_chubut =
                resultado_existencias_chubut.append({
                    'provincia':
df['provincia'][i],
                    'desde_año':
df['desde_año'][i],
                    'hasta_año':df['hasta_año']
[i],
                    'cantidad':df['cantidad']
[i],
                    'fuente':df['fuente'][i],
                    },
                ignore_index=True)
        resultado_existencias_chubut
```

```
Out[5]:
```

	provincia	desde_año	hasta_año	cantidad	fuente
0	Chubut	1965	1965	5736.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
1	Chubut	1966	1966	5789.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
2	Chubut	1967	1967	5841.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
3	Chubut	1968	1968	5894.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
4	Chubut	1969	1970	5947.0	Censos INDEC
5	Chubut	1971	1971	5887.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
6	Chubut	1972	1972	5826.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
7	Chubut	1973	1973	5766.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
8	Chubut	1974	1975	5705.0	Censos INDEC
9	Chubut	1976	1976	6063.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
10	Chubut	1977	1978	6421.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
11	Chubut	1979	1979	6287.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
12	Chubut	1980	1980	6153.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
13	Chubut	1981	1981	6020.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...

	provincia	desde_año	hasta_año	cantidad	fuerce
14	Chubut	1982	1982	5886.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
15	Chubut	1983	1983	5752.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
16	Chubut	1984	1984	5618.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
17	Chubut	1985	1985	5484.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
18	Chubut	1986	1986	5351.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
19	Chubut	1987	1987	5217.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
20	Chubut	1988	1988	5083.0	Censos INDEC
21	Chubut	1989	1989	5034.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
22	Chubut	1990	1990	4986.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
23	Chubut	1991	1991	4937.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
24	Chubut	1992	1992	4889.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
25	Chubut	1993	1993	4840.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
26	Chubut	1994	1994	4647.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
27	Chubut	1995	1995	4031.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
28	Chubut	1996	1996	4044.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
29	Chubut	1997	1997	3222.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
30	Chubut	1998	1998	3650.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
31	Chubut	1999	1999	4078.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
32	Chubut	2000	2000	4017.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
33	Chubut	2001	2001	3944.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
34	Chubut	2002	2002	3871.0	Censos INDEC
35	Chubut	2003	2003	4118.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
36	Chubut	2004	2004	4365.0	Estimación Dto. Ovinos y Lanas - SAGPyA en bas...
37	Chubut	2005	2005	4456.0	SIGSA - Dirección de Control de Gestión y Prog...
38	Chubut	2006	2006	4548.0	SIGSA - Dirección de Control de Gestión y Prog...
39	Chubut	2007	2007	4640.0	SIGSA - Dirección de Control de Gestión y Prog...
40	Chubut	2008	2008	4879.0	SIGSA - Dirección de Control de Gestión y Prog...
41	Chubut	2009	2009	4742.0	SIGSA - Dirección de Control de Gestión y Prog...
42	Chubut	2010	2010	4603.0	SIGSA - Dirección de Control de Gestión y Prog...
43	Chubut	2011	2011	4439.0	SIGSA - Dirección de Control de Gestión y Prog...
44	Chubut	2012	2012	4163.0	SIGSA - Dirección de Control de Gestión y Prog...
45	Chubut	2013	2013	4032.0	SIGSA - Dirección de Control de Gestión y Prog...
46	Chubut	2014	2014	4024.0	SIGSA - Dirección de Control de Gestión y Prog...
47	Chubut	2015	2015	3994.0	SIGSA - Dirección de Control de Gestión y Prog...
48	Chubut	2016	2016	4002.0	SIGSA - Dirección de Control de Gestión y Prog...
49	Chubut	2017	2017	3973.0	SIGSA - Dirección de Control de Gestión y Prog...

Estos son los datos que podemos obtener de dataset, así que los vamos a almacenar en formato csv:


```
In [6]: resultado_existencias_chubut.to_csv('datos_producidos/existencias_chubut_2017.csv', index=False)
```

Una columna interesante es la fuente de datos, estos nos podría dar un indicio de otro lugar en donde buscar datos de Chubut.

```
In [7]: resultado_existencias_chubut['fuente'].unique()
```

```
Out[7]: array(['Estimación Dto. Ovinos y Lanas - SAGPyA matematica con datos existentes',  
              'Censos INDEC',  
              'Encuesta Nacional Agropecuaria 93- 94- 95 y 01- SAPGyA e INDEC',  
              'Estimación Dto. Ovinos y Lanas - SAGPyA en base a datos objetivos',  
              'SIGSA - Dirección de Control de Gestión y Programas Especiales - Dirección Nacional de Sanidad Animal \x96 SENASA'],  
            dtype=object)
```

Cuando buscas en google *Estimación Dto. Ovinos y Lanas - SAGPyA matematica con datos existentes* o *Estimación Dto. Ovinos y Lanas - SAGPyA en base a datos objetivos* terminas llegando a la sección de ovinos de la página de [MAGyP](#) o a la sección de [Datos MAGyP](#). Y estos últimos son los mismos que están en [Datos Argentina](#).

La búsqueda de *Censos INDEC* o *Encuesta Nacional Agropecuaria 93- 94- 95 y 01- SAPGyA e INDEC* te lleva a difentes secciones de la pagiana del [INDEC](#) y lo unico que se consigue sobre ovinos es la cantiad que se exporta y consume en el pais, pero no hay informacion desmenuzada sobre Chubut.

Por último en *SIGSA - Dirección de Control de Gestión y Programas Especiales* y *Dirección Nacional de Sanidad Animal - SENASA* nos encontramos con la información proporcionada por senasa. Quizás acá haya algo interesante. Lo investigaremos y en caso de haber haremos un análisis.

1.4. Analizando dataset existencias ovinas 2008-2020 "MAGyC"

Introducción

Este dataset se obtuvo de [Datos MAGyC](#); y contiene información sobre existencias ovinas dentro del país. Son datos totales anuales desde Marzo 2008 a la última actualización, por provincia y partido / departamento, clasificadas por categorías.

Existe una pequeña documentación que nos da información sobre el dataset. Los campos serían los siguientes:

Título de la columna	Tipo de dato	Descripción
anio	Fecha ISO-8601 (date)	Año correspondiente al relevamiento.
provincia	Texto (string)	Nombre de la provincia.
provincia_id	Número entero (integer)	Código de la provincia.
departamento	Texto (string)	Nombre del partido / departamento.
departamento_id	Número entero (integer)	Código del partido / departamento.
carneros	Número entero (integer)	Cantidad de machos enteros adultos en edad reproductiva.
ovejas	Número entero (integer)	Cantidad de hembras adultas en edad reproductiva.
borregos	Número entero (integer)	Cantidad de machos jóvenes de reposición.
capones	Número entero (integer)	Cantidad de machos castrados.
corderos_corderas	Número entero (integer)	Cantidad de crías machos y hembras.

Y además de los campos también existe información del dataset en sí:

- Responsable: Dirección Nacional de Sanidad Animal
- Formato: csv
- Frecuencia de actualización: Anualmente
- Fecha de creación: 12 de septiembre de 2019
- Último cambio: 28 de noviembre de 2019

Lo anterior nos da una idea generica sobre el dataset, de todas formas vamos a cargarlo para ver su contenido y empezar el analisis.

Análisis

Importamos las librerías de python que necesitamos para analizar el dataset

```
In [1]: import pandas as pd
```

La librería pandas nos ayudará a leer y escribir datos en diferentes formatos, seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas, a fusionar y unir datos, entre muchas cosas más.

Cargamos el dataset y observamos su contenido

```
In [2]: df = pd.read_csv('datos/existencias-ovinas-provincia-departamento-2008-2019.csv', sep=",", encoding='latin-1')
df
```

```
Out[2]:
```

	anio	provincia	provincia_id	departamento	departamento_id	carneros	ovejas	borregos	capones
0	2008	Buenos Aires	6	Adolfo Alsina	6007.0	696.0	13238.0	868.0	195.0
1	2009	Buenos Aires	6	Adolfo Alsina	6007.0	721.0	11593.0	981.0	175.0
2	2010	Buenos Aires	6	Adolfo Alsina	6007.0	619.0	11016.0	860.0	236.0
3	2011	Buenos Aires	6	Adolfo Alsina	6007.0	670.0	10633.0	849.0	282.0
4	2012	Buenos Aires	6	Adolfo Alsina	6007.0	871.0	11617.0	1182.0	296.0
...	...	...	...	...	...	...	...	...	...
5790	2015	Tucumán	90	Yerba Buena	90119.0	11.0	31.0	NaN	NaN
5791	2016	Tucumán	90	Yerba Buena	90119.0	1.0	2.0	NaN	NaN
5792	2017	Tucumán	90	Yerba Buena	90119.0	1.0	2.0	NaN	NaN
5793	2018	Tucumán	90	Yerba Buena	90119.0	2.0	3.0	NaN	NaN
5794	2019	Tucumán	90	Yerba Buena	90119.0	2.0	3.0	NaN	NaN

5795 rows × 13 columns

Es un dataset de 5795 filas y 13 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [3]: df.columns
```

```
Out[3]: Index(['anio', 'provincia', 'provincia_id', 'departamento', 'departamento_id',
             'carneros', 'ovejas', 'borregos', 'capones', 'corderos_corderas',
             'Unnamed: 10', 'Unnamed: 11', 'Unnamed: 12'],
             dtype='object')
```

Comprobemos si este dataset tiene datos de Chubut.

```
In [4]: df['provincia'].unique()
```

```
Out[4]: array(['Buenos Aires', 'Catamarca', 'Chaco', 'Chubut',
             'Ciudad Autónoma de Buenos Aires', 'Córdoba', 'Corrientes',
             'Entre Ríos', 'Formosa', 'Jujuy', 'La Pampa', 'La Rioja',
             'Mendoza', 'Misiones', 'Neuquén', 'Río Negro', 'Salta', 'San Juan',
             'San Luis', 'Santa Cruz', 'Santa Fe', 'Santiago del Estero',
             'Tierra del Fuego', 'Tucumán'], dtype=object)
```

Efectivamente hay datos de Chubut, vamos a filtrarlos para verlos mejor:

In [5]:

```
resultado_existencias_chubut = pd.DataFrame(columns=
['anio','provincia', 'departamento', 'carneros', 'ovejas',
'borregos', 'capones', 'corderos_corderas'])
for i in df.index:
    if "Chubut" == df['provincia'][i]:
        resultado_existencias_chubut =
resultado_existencias_chubut.append({
                                'anio': df['anio'][i],
                                'provincia':
df['provincia'][i],
                                'departamento':df['departamento'][i],
                                'carneros':df['carneros']
[i],
                                'ovejas':df['ovejas'][i],
                                'borregos':df['borregos']
[i],
                                'capones':df['capones'][i],
                                'corderos_corderas':df['corderos_corderas'][i],},
                                ignore_index=True)
resultado_existencias_chubut
```

Out[5]:

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas
0	2008	Chubut	Biedma	7686.0	121561.0	58576.0	69633.0	33876.0
1	2009	Chubut	Biedma	7991.0	119248.0	55498.0	65328.0	23503.0
2	2010	Chubut	Biedma	6756.0	110469.0	38769.0	61853.0	23416.0
3	2011	Chubut	Biedma	6309.0	97816.0	36529.0	54338.0	21831.0
4	2012	Chubut	Biedma	5785.0	93378.0	30168.0	48447.0	16071.0
...	...	...	...	...	...	...	...	...
175	2015	Chubut	Telsen	4766.0	81572.0	24643.0	42758.0	21277.0
176	2016	Chubut	Telsen	4896.0	80262.0	26471.0	39243.0	20290.0
177	2017	Chubut	Telsen	4640.0	77556.0	27296.0	36570.0	17750.0
178	2018	Chubut	Telsen	4832.0	79117.0	28366.0	37558.0	21933.0
179	2019	Chubut	Telsen	4781.0	80109.0	28428.0	38109.0	17731.0

180 rows × 8 columns

```
In [7]: resultado_existencias_chubut =
resultado_existencias_chubut.sort_values(by=
['anio', 'departamento'], ignore_index=True)
resultado_existencias_chubut
```

```
Out[7]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas
0	2008	Chubut	Biedma	7686.0	121561.0	58576.0	69633.0	33876.0
1	2008	Chubut	Cushamen	8180.0	149189.0	41669.0	51128.0	37589.0
2	2008	Chubut	Escalante	7863.0	109225.0	35581.0	40786.0	41516.0
3	2008	Chubut	Florentino Ameghino	13875.0	164382.0	79863.0	63856.0	48809.0
4	2008	Chubut	Futaleufú	5639.0	90494.0	30119.0	24634.0	28237.0
...	...	...	...	...	...	...	...	...
175	2019	Chubut	Rawson	3628.0	74009.0	19514.0	20800.0	18895.0
176	2019	Chubut	Río Senguer	25733.0	384841.0	138981.0	122343.0	125892.0
177	2019	Chubut	Sarmiento	6077.0	99175.0	28287.0	31325.0	28339.0
178	2019	Chubut	Tehuelches	15264.0	196118.0	89141.0	87696.0	74093.0
179	2019	Chubut	Telsen	4781.0	80109.0	28428.0	38109.0	17731.0

180 rows × 8 columns

Estos son los datos que necesitamos, por lo tanto los guardamos en un archivo csv.

```
In [8]: resultado_existencias_chubut.to_csv('datos_producidos/existencias_ch
index=False)
```

1.5. Analizando dataset: movimientos ovinos

Introducción

Este dataset se obtuvo de [Datos Argentina](#); y contiene información sobre el movimiento de los subproductos ovinos dentro del país.

Existe una pequeña documentación que nos da información sobre el dataset. Los campos serían los siguientes:

Título de la columna	Tipo de dato	Descripción
fecha	Fecha ISO-8601 (date)	
salido_de	Texto (string)	Lugar del que sale.
ingresado_a	Texto (string)	Lugar al que ingresa.
provincia_origen	Texto (string)	Nombre de la provincia de origen.
provincia_origen_id	Número entero (integer)	Código de la provincia de origen.
provincia_destino	Texto (string)	Nombre de la provincia de destino.
provincia_destino_id	Número entero (integer)	Código de la provincia de destino.
cueros_pieles_brutas	Número entero (integer)	Cantidad de cueros y pieles
lana_sucia	Número entero (integer)	Cantidad de lana sucia.
lana_limpia	Número entero (integer)	Cantidad de lana limpia.
lana_sucia_vellon	Número entero (integer)	Cantidad de vellón de lana sucia.
lana_sucia_no_vellon	Número entero (integer)	Cantidad de lana sucia, que no corresponde a vellón.
pelos_cerdas	Número entero (integer)	Cantidad de pelos y cerdas.
estiercol	Número entero (integer)	Cantidad de estiércol.
guano_aviar	Número entero (integer)	Cantidad de guano aviar.
guano_cap	Número entero (integer)	Cantidad de guano.
cama_pollo	Número entero (integer)	Cantidad de cama de pollo.
aves_muertas	Número entero (integer)	Cantidad de aves muertas.
pasto_fresco_heno_TdF	Número entero (integer)	Cantidad de pasto.
huesos_despojos_faena	Número entero (integer)	Cantidad de huesos y despojos de faena.
trofeo_caza	Número entero (integer)	Cantidad de trofeos de caza.
astas_volteo_ciervos	Número entero (integer)	Cantidad de astas por volteo de ciervos.

Y además de los campos también existe información del dataset en sí:

- Responsable: Servicio Nacional de Sanidad y Calidad Agroalimentaria
- Formato: csv
- Temas: Agroganadería, pesca y forestación
- Frecuencia de actualización: Mensualmente
- Fecha de creación: 27 de septiembre de 2019
- Último cambio: 5 de diciembre de 2019

Lo anterior nos da una idea genérica sobre el dataset, de todas formas vamos a cargarlo para ver su contenido y empezar el análisis.

Análisis

Importar librerías

Importamos las librerías de python que necesitamos para analizar el dataset

```
In [1]: import pandas as pd
import numpy as np
```

La librería pandas nos ayudará a leer y escribir datos en diferentes formatos, seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas, a fusionar y unir datos, entre muchas cosas más.

Numpy lo utilizaremos para hacer cálculos.

Cargar dataset

Cargamos el dataset y observamos su contenido

```
In [2]: df = pd.read_csv('datos/movimiento-subproductos-animales-
provincia-2013-2018.csv', sep=",", encoding='latin-1')
df
```

```
Out[2]:
```

	fecha	salido_de	ingresado_a	provincia_origen	provincia_origen_id	provincia_destino	provin
0	2013-01	Establecimiento	Establecimiento	Buenos Aires	6	Buenos Aires	
1	2013-01	Establecimiento	Establecimiento	Buenos Aires	6	Buenos Aires	
2	2013-01	Establecimiento	Establecimiento	Buenos Aires	6	Buenos Aires	
3	2013-01	Establecimiento	Establecimiento	Buenos Aires	6	Buenos Aires	
4	2013-01	Establecimiento	Establecimiento	Buenos Aires	6	Chubut	
...	...	...	...	...	...	...	...
8252	2018-12	Establecimiento	Frigorífico	Santiago del Estero	86	Buenos Aires	
8253	2018-12	Establecimiento	Frigorífico	Santiago del Estero	86	Santa Fe	
8254	2018-12	Frigorífico	Frigorífico	Tucumán	90	Buenos Aires	
8255	2018-12	Frigorífico	Frigorífico	Tucumán	90	La Rioja	
8256	2018-12	Frigorífico	Frigorífico	Tucumán	90	Santa Fe	

8257 rows × 23 columns

Es un dataset de 8257 filas y 23 columnas, será necesario analizarlo en detenimiento para saber si contiene datos que nos sirvan, miremos las columnas que trae:

```
In [3]: df.columns
```

```
Out[3]: Index(['fecha', 'salido_de', 'ingresado_a', 'provincia_origen',  
             'provincia_origen_id', 'provincia_destino', 'provincia_destino_id',  
             'unidad_medida', 'cueros_pieles_brutas', 'lana_sucia', 'lana_limpia',  
             'lana_sucia_vellon', 'lana_sucia_no_vellon', 'pelos_cerdas',  
             'estiercol', 'guano_aviar', 'guano_cap', 'cama_pollo', 'aves_muertas',  
             'pasto_fresco_heno_TdF', 'huesos_despojos_faena', 'trofeo_caza',  
             'astas_volteo_ciervos'],  
            dtype='object')
```

De todas las columnas anteriores las que nos pueden interesar son las referentes a la provincia, la unidad de medida y la lana. Comprobemos si este dataset tiene datos de Chubut.

```
In [4]: df['provincia_origen'].unique()
```

```
Out[4]: array(['Buenos Aires', 'Chaco', 'Chubut', 'Córdoba', 'Corrientes',  
             'Entre Ríos', 'Jujuy', 'La Pampa', 'La Rioja', 'Mendoza',  
             'Misiones', 'Neuquén', 'Río Negro', 'Salta', 'San Juan',  
             'San Luis', 'Santa Cruz', 'Santa Fe', 'Santiago del Estero',  
             'Catamarca', 'Formosa', 'Tierra del Fuego', 'Tucumán'],  
            dtype=object)
```

```
In [5]: df['provincia_destino'].unique()
```

```
Out[5]: array(['Buenos Aires', 'Chubut', 'Ciudad Autónoma de Buenos Aires',  
             'Corrientes', 'Mendoza', 'Río Negro', 'Santa Fe', 'Catamarca',  
             'Córdoba', 'Entre Ríos', 'San Luis', 'La Rioja', 'Salta',  
             'La Pampa', 'Formosa', 'Santiago Del Estero', 'Misiones',  
             'Neuquén', 'San Juan', 'Jujuy', 'Chaco', 'Tierra Del Fuego',  
             'Tucumán', 'Santa Cruz', 'Santiago del Estero', 'Tierra del Fuego'],  
            dtype=object)
```

Efectivamente hay movimientos de subproductos que involucran a la provincia, vamos a suponer que la salida de productos, aquellas filas que tienen como origen a Chubut, corresponden a la producción de lana de la misma. Entonces comencemos a filtrar.

Obtengamos los datos de la provincia de Chubut

```
In [6]: movimientos_chubut = pd.DataFrame()  
movimientos_chubut = df.loc[(df['provincia_origen'] == 'Chubut') |  
                             (df['provincia_destino'] == 'Chubut')]  
movimientos_chubut
```

```
Out[6]:
```

	fecha	salido_de	ingresado_a	provincia_origen	provincia_origen_id	provincia_destino	provin
4	2013-01	Establecimiento	Establecimiento	Buenos Aires	6	Chubut	

	fecha	salido_de	ingresado_a	provincia_origen	provincia_origen_id	provincia_destino	provin
5	2013-01	Establecimiento	Establecimiento	Buenos Aires	6	Chubut	
6	2013-01	Establecimiento	Establecimiento	Buenos Aires	6	Chubut	
14	2013-01	Establecimiento	Establecimiento	Chubut	26	Buenos Aires	
24	2013-01	Establecimiento	Establecimiento	Corrientes	18	Chubut	
...	...	...	...	...	...	...	...
8217	2018-12	Establecimiento	Frigorífico	Neuquén	58	Chubut	
8221	2018-12	Establecimiento	Frigorífico	Río Negro	62	Chubut	
8222	2018-12	Establecimiento	Frigorífico	Río Negro	62	Chubut	
8242	2018-12	Establecimiento	Frigorífico	Santa Cruz	78	Chubut	
8246	2018-12	Establecimiento	Frigorífico	Santa Fe	82	Chubut	

1664 rows × 23 columns

Lo que se hizo fue filtrar por provincia origen y provincia destino

Obtengamos los datos referentes a la lana

```
In [7]: movimientos_lana_chubut = pd.DataFrame()
movimientos_lana_chubut = movimientos_chubut.loc[:, ['fecha',
                                                    'salido_de',
                                                    'ingresado_a',

                                                    'provincia_origen',

                                                    'provincia_destino',

                                                    'unidad_medida',
                                                    'lana_sucia',
                                                    'lana_limpia',

                                                    'lana_sucia_vellon',

                                                    'lana_sucia_no_vellon'],]

movimientos_lana_chubut
```

Out[7]:		fecha	salido_de	ingresado_a	provincia_origen	provincia_destino	unidad_medida	lana_sucia
	4	2013-01	Establecimiento	Establecimiento	Buenos Aires	Chubut	Kg	417063
	5	2013-01	Establecimiento	Establecimiento	Buenos Aires	Chubut	Unidad/es	63552
	6	2013-01	Establecimiento	Establecimiento	Buenos Aires	Chubut	Bulto/s	25211
	14	2013-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Unidad/es	0
	24	2013-01	Establecimiento	Establecimiento	Corrientes	Chubut	Kg	109693
	...	...	...	...	...	...	...	...
	8217	2018-12	Establecimiento	Frigorífico	Neuquén	Chubut	Unidad/es	0
	8221	2018-12	Establecimiento	Frigorífico	Río Negro	Chubut	Kg	0
	8222	2018-12	Establecimiento	Frigorífico	Río Negro	Chubut	Unidad/es	0
	8242	2018-12	Establecimiento	Frigorífico	Santa Cruz	Chubut	Kg	0
	8246	2018-12	Establecimiento	Frigorífico	Santa Fe	Chubut	Kg	0

1664 rows × 10 columns

Lo que se hizo fue quedarnos solo con aquellas columnas que son de nuestro interés, como la provincia, la unidad de medida y los diferentes tipos de lana.

En nuestro caso solo estamos interesados en la lana producida en Chubut, que estaría representado por provincia origen.

```
In [8]: salida_lana_chubut = pd.DataFrame()
salida_lana_chubut =
movimientos_lana_chubut.loc[(df['provincia_origen'] == 'Chubut')]
salida_lana_chubut
```

Out[8]:		fecha	salido_de	ingresado_a	provincia_origen	provincia_destino	unidad_medida	lana_sucia
	14	2013-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Unidad/es	0
	107	2013-02	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	4000
	108	2013-02	Establecimiento	Establecimiento	Chubut	Buenos Aires	Unidad/es	0
	109	2013-02	Establecimiento	Establecimiento	Chubut	Chubut	Kg	9600
	110	2013-02	Establecimiento	Establecimiento	Chubut	Chubut	Unidad/es	30
	...	...	...	...	...	...	...	...

	fecha	salido_de	ingresado_a	provincia_origen	provincia_destino	unidad_medida	lana_sucia
8077	2018-11	Establecimiento	Establecimiento	Chubut	Rio Negro	Kg	0
8175	2018-12	Establecimiento	Frigorífico	Chubut	Buenos Aires	Unidad/es	0
8176	2018-12	Establecimiento	Frigorífico	Chubut	Chubut	Kg	0
8177	2018-12	Establecimiento	Establecimiento	Chubut	Chubut	Kg	0
8178	2018-12	Establecimiento	Establecimiento	Chubut	Chubut	Unidad/es	0

472 rows × 10 columns

Otra forma de obtener los movimientos de lana de chubut sería así:

```
In [9]: resultado_movimientos_chubut = pd.DataFrame(columns=
['fecha', 'salido_de', 'ingresado_a', 'provincia_origen',
'provincia_destino', 'lana_sucia', 'lana_limpia',
'lana_sucia_vellon', 'lana_sucia_no_vellon'])
for i in df.index:
    if "Chubut" == df['provincia_origen'][i]:
        resultado_movimientos_chubut =
resultado_movimientos_chubut.append({
                                'fecha': df['fecha'][i],
                                'salido_de':
df['salido_de'][i],

                                'ingresado_a':df['ingresado_a'][i],

                                'provincia_origen':df['provincia_origen'][i],

                                'provincia_destino':df['provincia_destino'][i],

                                'unidad_medida':df['unidad_medida'][i],

                                'lana_sucia':df['lana_sucia'][i],

                                'lana_limpia':df['lana_limpia'][i],

                                'lana_sucia_vellon':df['lana_sucia_vellon'][i],

                                'lana_sucia_no_vellon':df['lana_sucia_no_vellon'][i],
                                },
```

```
ignore_index=True)
```

```
resultado_movimientos_chubut
```

```
Out[9]:
```

	fecha	salido_de	ingresado_a	provincia_origen	provincia_destino	lana_sucia	lana_limpia	lan
0	2013-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	0	0	
1	2013-02	Establecimiento	Establecimiento	Chubut	Buenos Aires	4000	0	
2	2013-02	Establecimiento	Establecimiento	Chubut	Buenos Aires	0	0	
3	2013-02	Establecimiento	Establecimiento	Chubut	Chubut	9600	0	
4	2013-02	Establecimiento	Establecimiento	Chubut	Chubut	30	0	
...	...	...	...	...	...	...	...	...
467	2018-11	Establecimiento	Establecimiento	Chubut	Río Negro	0	0	
468	2018-12	Establecimiento	Frigorífico	Chubut	Buenos Aires	0	0	
469	2018-12	Establecimiento	Frigorífico	Chubut	Chubut	0	0	
470	2018-12	Establecimiento	Establecimiento	Chubut	Chubut	0	0	
471	2018-12	Establecimiento	Establecimiento	Chubut	Chubut	0	0	

472 rows × 10 columns

Una vez obtenida la lana que sale de chubut, veamos si todas comparten la misma unidad de medida:

```
In [10]: salida_lana_chubut['unidad_medida'].unique()
```

```
Out[10]: array(['Unidad/es', 'Kg', 'Bulto/s', 'Grupo', 'Mts3'], dtype=object)
```

Lamentablemente no, eso significa que debemos elegir una y sera necesario hacer la coversión del resto. Primero vamos a eliminar aquellos establecimientos de Chubut que no produjeron lana, es decir cuyas filas son cero en los distintos productos relacionados a la lana

```
In [11]: valor=0
salida_lana_chubut_filtrada =
salida_lana_chubut[~salida_lana_chubut.lana_sucia.isin([valor])|

~salida_lana_chubut.lana_limpia.isin([valor])|

~salida_lana_chubut.lana_sucia_vellon.isin([valor])|

~salida_lana_chubut.lana_sucia_no_vellon.isin([valor])]
```

```
salida_lana_chubut_filtrada
```

```
Out[11]:
```

	fecha	salido_de	ingresado_a	provincia_origen	provincia_destino	unidad_medida	lana_sucia
107	2013-02	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	4000
109	2013-02	Establecimiento	Establecimiento	Chubut	Chubut	Kg	9600
110	2013-02	Establecimiento	Establecimiento	Chubut	Chubut	Unidad/es	30
195	2013-03	Establecimiento	Establecimiento	Chubut	Buenos Aires	Unidad/es	22400
196	2013-03	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	7390
...	...	...	...	...	...	...	...
7969	2018-10	Establecimiento	Establecimiento	Chubut	Chubut	Kg	0
8072	2018-11	Establecimiento	Frigorífico	Chubut	Chubut	Kg	0
8073	2018-11	Establecimiento	Establecimiento	Chubut	Chubut	Kg	0
8176	2018-12	Establecimiento	Frigorífico	Chubut	Chubut	Kg	0
8177	2018-12	Establecimiento	Establecimiento	Chubut	Chubut	Kg	0

260 rows × 10 columns

Limpieza de datos

Pasamos de 462 filas a 260, ahora veamos la unidad de medidas para estas filas.

```
In [12]: salida_lana_chubut_filtrada['unidad_medida'].unique()
```

```
Out[12]: array(['Kg', 'Unidad/es', 'Bulto/s'], dtype=object)
```

La unidad de medida seleccionada es Kilogramos.

En [este artículo](#) se menciona que los productores comercializan la lana por vellón, es decir por bulto, y cada uno de ellos se identifica con datos del productor, zafra y peso. Por lo tanto da entender que un vellón es equivalente a un bulto. Dado que no existe documentación que explique las unidades de medida para este dataset, se decidió que un vellón es equivalente a un bulto y a una unidad, y tanto el bulto como la unidad se miden en kilogramos.

Prácticamente todos los ovinos en la Patagonia son de raza Merino o Corriedale, se cría Merino para lana en las zonas áridas y secas de Río Negro, Chubut y norte de Santa Cruz; y Corriedale para lana y carne en zonas más húmedas del sur de Santa Cruz y Tierra del Fuego [1](#). Chubut se caracteriza por ser la provincia que mas lana produce y de mayor precio; y su producción se sustenta sobre la raza Merino, con lanas finas y superfinas de alto valor en el mercado internacional [2](#).

La raza Merino es originaria de España, su vellón es característico por su densidad de fibras, finura, ondulaciones y uniformidad. Es un animal de cara y lana blanca, con capacidad de adaptación a distintos medios. Los corderos para matanza alcanzan un peso de venta de al menos 40 kg a los 4-6 meses de edad y las ovejas adultas 55 a 65 kg, además las ovejas producen entre 4 y 6 kg de vellón, con finuras entre 19 y 22 micras de alta calidad [3](#).

El peso del vellón ha sido la medida con que tradicionalmente se ha valorado el potencial lanero del ovino. En este [estudio](#) la media poblacional obtenida para la variable peso de vellón en la raza merino ha sido de $4,18 \pm 0,054$ kilogramos, con un intervalo de variación de 4,20 kg. (máx. de 6,20 kg. y mín. de 2,00 kg.) y un coeficiente de variación del 20,41%. Y se observó que esta condicionada por la ganadería de nacimiento, la edad y el sexo de los animales. Así, nos encontramos con medias en función del tipo que oscilan entre los 3,72 kg para las ovejas y los 4,51 kg de los machos (los animales jóvenes tienen mayor peso del vellón que los adultos). El histograma de frecuencias de la variable peso del vellón se ajusta a una distribución normal, donde aproximadamente el 99% de la población se encuentra entre 2,8 y 6 kg. de peso del vellón, estando la máxima frecuencia entre 3,5 y 4,5 kg. de peso del vellón, lo cual representa el 47% de las observaciones. La frecuencia máxima la encontramos entre 3,5 y 4 kg. de peso.

Cómo no tenemos los datos reales en kilos vamos a generar pesos de vellones aleatorios en base a la media global siguiendo una distribución normal, y multiplicarla por la cantidad de bultos y/o unidades que tenga cada productor en el dataset. Algo que hay que tener en cuenta es que idealmente el 5% de la majada son machos mientras que el resto son hembras, y cómo el peso del vellón varía según el sexo, esto también se tendrá en cuenta.

Veamos un ejemplo de la forma en que se van a crear los datos. Supongamos que un productor tiene 30 bultos o 30 unidades:

```
In [13]: # sacar el 5% de total que representa a los machos
# el 95% son ovejas hembras
cantidad_bultos_unidades = 30
porcentaje_macho = 5
porcentaje_hembra = 95
media_vellon_hembra = 3.72
media_vellon_macho = 4.51
coeficiente_variacion = 0.054
cantidad_machos = int(np.ceil(cantidad_bultos_unidades *
porcentaje_macho / 100))
cantidad_hembras = int(np.floor(cantidad_bultos_unidades *
porcentaje_hembra / 100))
lana_hembra = np.random.normal(media_vellon_hembra,
coeficiente_variacion, cantidad_hembras)
lana_macho = np.random.normal(media_vellon_macho,
coeficiente_variacion, cantidad_machos)
lana_ovinos = np.concatenate((lana_hembra, lana_macho), axis =
None).tolist()
```

```
lana_total = round(sum(lana_ovinos),2)
lana_total
```

Out[13]: 113.43

En este caso 30 bultos o unidades equivalen a 113.4 kilogramos.

Vamos a convertir este cálculo en una función para ser utilizada más adelante

```
In [14]: def convertir_a_kilogramos(x, columna):
    if (x['unidad_medida'] == 'Bulto/s') | (x['unidad_medida'] ==
    'Unidad/es'):
        cantidad_bultos_unidades = x[columna]
        porcentaje_macho = 5
        porcentaje_hembra = 95
        media_vellon_hembra = 3.72
        media_vellon_macho = 4.51
        coeficiente_variacion = 0.054
        cantidad_machos = int(np.ceil(cantidad_bultos_unidades *
        porcentaje_macho / 100))
        cantidad_hembras = int(np.floor(cantidad_bultos_unidades *
        porcentaje_hembra / 100))
        lana_hembra = np.random.normal(media_vellon_hembra,
        coeficiente_variacion, cantidad_hembras)
        lana_macho = np.random.normal(media_vellon_macho,
        coeficiente_variacion, cantidad_machos)
        lana_ovinos = np.concatenate((lana_hembra, lana_macho),
        axis = None).tolist()
        lana_total = round(sum(lana_ovinos),2)
        return lana_total
    else:
        return x[columna]
```

Lo siguiente es definir las columnas que queremos convertir, recorrer el dataframe y aplicar la función anterior, el resultado que se obtiene es el siguiente:

```
In [15]: columnas = ["lana_sucia", "lana_limpia", "lana_sucia_vellon",
    "lana_sucia_no_vellon"]
    for columna in columnas:
        salida_lana_chubut_filtrada[columna] =
        salida_lana_chubut_filtrada.apply(lambda x:
        convertir_a_kilogramos(x, columna), axis=1)
```

```
salida_lana_chubut_filtrada.loc[salida_lana_chubut_filtrada.unidad_m
salida_lana_chubut_filtrada
```

Out[15]:

	fecha	salido_de	ingresado_a	provincia_origen	provincia_destino	unidad_medida	lana_sucia
107	2013-02	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	4000.00
109	2013-02	Establecimiento	Establecimiento	Chubut	Chubut	Kg	9600.00
110	2013-02	Establecimiento	Establecimiento	Chubut	Chubut	Kg	113.17
195	2013-03	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	84213.38
196	2013-03	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	7390.00
...	...	...	...	...	...	...	...
7969	2018-10	Establecimiento	Establecimiento	Chubut	Chubut	Kg	0.00
8072	2018-11	Establecimiento	Frigorífico	Chubut	Chubut	Kg	0.00
8073	2018-11	Establecimiento	Establecimiento	Chubut	Chubut	Kg	0.00
8176	2018-12	Establecimiento	Frigorífico	Chubut	Chubut	Kg	0.00
8177	2018-12	Establecimiento	Establecimiento	Chubut	Chubut	Kg	0.00

260 rows × 10 columns

Ahora todos nuestros datos de la lana están en kilogramos, lo que debemos hacer es convertir la columna fecha en un dato fecha para empezar a agrupar los datos y así obtener la cantidad de lana anual de Chubut.

In [16]:

```
salida_lana_chubut_filtrada['fecha'] =
pd.to_datetime(salida_lana_chubut['fecha'], format='%Y-%m')
```

Una vez convertida, hagamos primero una pequeña prueba con un año:

In [17]:

```
salida_lana_2013 =
salida_lana_chubut_filtrada.loc[(salida_lana_chubut_filtrada['fecha']
>= '2013-01')
                                & (salida_lana_chubut_filtrada['fecha'] <=
'2013-12')]
salida_lana_2013
```

Out[17]:

	fecha	salido_de	ingresado_a	provincia_origen	provincia_destino	unidad_medida	lana_sucia
107	2013-02-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	4000.00
109	2013-	Establecimiento	Establecimiento	Chubut	Chubut	Kg	9600.00

	fecha	salido_de	ingresado_a	provincia_origen	provincia_destino	unidad_medida	lana_sucia
	02-01						
110	2013-02-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	113.17
195	2013-03-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	84213.38
196	2013-03-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	7390.00
197	2013-03-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	36566.00
198	2013-03-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	63.87
401	2013-05-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	5000.00
744	2013-08-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	5000.00
881	2013-09-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	313.32
1008	2013-10-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	6000.00
1010	2013-10-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	188.31
1011	2013-10-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	67.86
1145	2013-11-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	14800.00
1146	2013-11-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	90.71
1147	2013-11-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	48780.00
1150	2013-11-01	Establecimiento	Establecimiento	Chubut	Río Negro	Kg	1600.00
1287	2013-12-01	Establecimiento	Establecimiento	Chubut	Buenos Aires	Kg	11200.00
1289	2013-12-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	70568.00
1290	2013-12-01	Establecimiento	Establecimiento	Chubut	Chubut	Kg	499.15
1292	2013-12-01	Establecimiento	Establecimiento	Chubut	Río Negro	Kg	230.00

In [18]:

```
lana_sucia = salida_lana_2013['lana_sucia'].sum()
lana_limpia = salida_lana_2013['lana_limpia'].sum()
lana_sucia_vellon = salida_lana_2013['lana_sucia_vellon'].sum()
lana_sucia_no_vellon =
salida_lana_2013['lana_sucia_no_vellon'].sum()
```

```
df = pd.DataFrame()
df['año'] = ['2013']
df['lana_sucia']=[lana_sucia]
df['lana_limpia']=[lana_limpia]
df['lana_sucia_vellon']=[lana_sucia_vellon]
df['lana_sucia_no_vellon']=[lana_sucia_no_vellon]
df
```

```
Out[18]:
```

	año	lana_sucia	lana_limpia	lana_sucia_vellon	lana_sucia_no_vellon
0	2013	306283.77	25000.0	0.0	0.0

De esta manera se obtiene la cantidad de lana "producida" en el año 2013 en la provincia de Chubut.

Hay ciertas columnas que ya no nos son de importancia, por lo tanto vamos a eliminarlas del dataframe.

```
In [19]: salida_lana_chubut_filtrada.drop(['salido_de', 'ingresado_a',
'provincia_origen', 'provincia_destino', 'unidad_medida' ], axis =
'columns', inplace=True)
salida_lana_chubut_filtrada
```

```
Out[19]:
```

	fecha	lana_sucia	lana_limpia	lana_sucia_vellon	lana_sucia_no_vellon
107	2013-02-01	4000.00	0.0	0.0	0.0
109	2013-02-01	9600.00	0.0	0.0	0.0
110	2013-02-01	113.17	0.0	0.0	0.0
195	2013-03-01	84213.38	0.0	0.0	0.0
196	2013-03-01	7390.00	0.0	0.0	0.0
...	...	...	...	...	...
7969	2018-10-01	0.00	1000.0	154613.0	34649.0
8072	2018-11-01	0.00	20000.0	2199643.0	21386.0
8073	2018-11-01	0.00	1050.0	142754.0	25654.0
8176	2018-12-01	0.00	0.0	2048232.0	71598.0
8177	2018-12-01	0.00	0.0	117112.0	8400.0

260 rows × 5 columns

Vamos a generalizar lo que hicimos antes con el año 2013 al resto de los años:

```
In [20]: anio = 2013
anio_fin = 2018
salida_lana_chubut = pd.DataFrame(columns=['año', 'lana_sucia',
'lana_limpia', 'lana_sucia_vellon', 'lana_sucia_no_vellon',
'total_lana'])
while(anio<=anio_fin):
    inicio = str(anio)+'-01'
```

```

        fin = str(anio)+'-12'
        salida_lana_anual =
salida_lana_chubut_filtrada.loc[(salida_lana_chubut_filtrada['fecha'
>= inicio)
                                & (salida_lana_chubut_filtrada['fecha'] <= fin)]

        lana_sucia = salida_lana_anual['lana_sucia'].sum()
        lana_limpia = salida_lana_anual['lana_limpia'].sum()
        lana_sucia_vellon =
salida_lana_anual['lana_sucia_vellon'].sum()
        lana_sucia_no_vellon =
salida_lana_anual['lana_sucia_no_vellon'].sum()
        lana_total =
lana_sucia+lana_limpia+lana_sucia_vellon+lana_sucia_no_vellon

        filas = [(str(anio), lana_sucia, lana_limpia,
lana_sucia_vellon, lana_sucia_no_vellon, lana_total)]
        salida_lana_chubut_anual = pd.DataFrame(filas, columns=['año',
'lana_sucia', 'lana_limpia',
'lana_sucia_vellon', 'lana_sucia_no_vellon', 'total_lana'])
        salida_lana_chubut =
salida_lana_chubut.append(salida_lana_chubut_anual,
ignore_index=True)
        anio+=1

salida_lana_chubut

```

Out[20]:

	año	lana_sucia	lana_limpia	lana_sucia_vellon	lana_sucia_no_vellon	total_lana
0	2013	306283.77	25000.00	0.00	0.0	331283.77
1	2014	3757227.60	72900.00	0.00	0.0	3830127.60
2	2015	8021463.81	82321.00	0.00	0.0	8103784.81
3	2016	16268803.24	19407.00	0.00	0.0	16288210.24
4	2017	4605708.40	24375.83	8237716.67	456845.1	13324646.00
5	2018	0.00	46190.00	12965665.73	666827.0	13678682.73

Y así obtuvimos la cantidad de lana que Chubut produjo desde el año 2013 al 2019. Vamos a guardar estos datos en un archivo csv.

In [22]:

```

salida_lana_chubut.to_csv('datos_producidos/movimientos_ovinos/salida_lana_
2018.csv', index=False)

```

1.6. Armar dataset de clima por departamento

Los datos se obtuvieron de los scrapers que se corrieron en la página del SIPAS. Se obtuvieron 5 dataset: humedad, lluvia, temperatura mínima, temperatura media, y temperatura máxima. En cada uno de ellos se eliminaron las sondas que no eran de Chubut (unas 7 creo) y a cada sonda se le asignó el departamento al que pertenecía manualmente.

Importamos las librerías necesarias.

```
In [1]: import pandas as pd
```

Cargamos el dataset de humedad, lo agrupamos por dpto y año

```
In [2]: datos_humedad =  
pd.read_csv('datos_sipas/Agrupado_por_dptos/humedad_estaciones.csv',  
sep=",")  
datos_humedad
```

```
Out[2]:
```

	Estación	Departamento	Año	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV
0	Estancia La Adela - Península Valdés	BIEDMA	2012						65.5	58.9	69.5	59.2	60.4	47.1
1	Estancia La Adela - Península Valdés	BIEDMA	2013	53.6	59.7	70	73.9	69.2	66.5	76.1	69.1	74.4	63.6	52.5
2	Estancia La Adela - Península Valdés	BIEDMA	2014	44.7	53.8	58	70.3	72.8	74	74.2	66	65	64.3	51.5
3	Estancia La Adela - Península Valdés	BIEDMA	2015	50.6	57.6	57.1	64.8		60			66.5	64.5	
4	Estancia La Adela - Península Valdés	BIEDMA	2016	60.5	59.7	64.2	54.7							
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
357	Telsen	TELSEN	2010	29.5	37.5	47	45.3	44.9	56.7	57.4	41.2	41.8	40.1	35.9
358	Telsen	TELSEN	2011	31.4	36.6	39.4	43.2	44.9	51.1	51.9	51.8	39.3	47.7	33.1
359	Telsen	TELSEN	2012	36.6	44.1	50.6	45.4	45.7	54.6	46	47.9	42.1	39.1	31.2
360	Telsen	TELSEN	2013	40.3	43.9	49.4	53.2	57.2	52.1	59.5	52.3	54	42.4	34.5
361	Telsen	TELSEN	2014	30	36.6	37.9								

362 rows × 16 columns

In [3]:

```
datos_humedad_agrupados = datos_humedad.groupby(by=[
    'Departamento', 'Año']).mean().round(2)
datos_humedad_agrupados
```

Out[3]:

PROMEDIO_HUMEDAD		
Departamento	Año	
BIEDMA	2012	58.40
	2013	64.70
	2014	60.50
	2015	58.70
	2016	57.50
...	...	...
TELSEN	2018	47.93
	2019	47.57
	2020	46.63
	2021	47.03
	2022	37.15

211 rows × 1 columns

Cargamos el dataset de lluvia, lo agrupamos por dpto y año

In [4]:

```
datos_lluvia =
pd.read_csv('datos_sipas/Agrupado_por_dptos/lluvia_estaciones.csv',
sep=",")
datos_lluvia
```

Out[4]:

	Estación	Departamento	Año	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV
0	Estancia La Adela - Península Valdés	BIEDMA	2012						16	5.4	1.2	2.3	9.1	0.6
1	Estancia La Adela - Península Valdés	BIEDMA	2013	14	13.8	60.2	16.2	21.4	13	23.6	28.1	9	21.9	3.5
2	Estancia La Adela - Península Valdés	BIEDMA	2014	12.7	4.2	22.1	133	25.6	2.1	19.2	26.3	38.7	27.1	10
3	Estancia La Adela	BIEDMA	2015	23	19.4	10.5	6.2		6.6			4.3	14.3	

	Estación	Departamento	Año	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV
	-													
	Península													
	Valdés													
	Estancia													
	La Adela													
4	-	BIEDMA	2016	43.3	8	5.7	13.2							
	Península													
	Valdés													
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
507	Telsen	TELSEN	2010	10	11.6	11.4	6.8	2.6	9	18.4	NaN	20.7	8.4	2.5
508	Telsen	TELSEN	2011	0.8	17.4	10.3	3.3	5.3	6.5	5.5	10.9	17.9	30.2	1.1
509	Telsen	TELSEN	2012	24.3	31.9	25.5	2.4	0.5	20.4	3.2	14.9	5.1	23.3	5.5
510	Telsen	TELSEN	2013	12.4	36.7	28.4	26.5	8.4	6.4	36.3	10.7	20.8	17.4	5.1
511	Telsen	TELSEN	2014	6.1	3.8	4.1								

512 rows × 16 columns

```
In [5]: datos_lluvia_agrupados = datos_lluvia.groupby(by=
['Departamento', 'Año']).sum().round(2)
datos_lluvia_agrupados
```

Out[5]:

TOTAL_LLUVIA		
Departamento	Año	
BIEDMA	2012	46.4
	2013	226.5
	2014	331.3
	2015	114.2
	2016	88.9
...	...	...
TELSEN	2018	604.2
	2019	358.4
	2020	304.4
	2021	608.9
	2022	58.7

250 rows × 1 columns

Cargamos el dataset de temperatura máxima, lo agrupamos por dpto y año

```
In [6]: datos_tem_max =
pd.read_csv('datos_sipas/Agrupado_por_dptos/temperatura_maxima_estac
sep=",")
datos_tem_max
```

Out[6]:	Estación	Departamento	Año	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV
0	Estancia La Adela - Península Valdés	BIEDMA	2012						13.2	12	14.9	17.6	20	24.5
1	Estancia La Adela - Península Valdés	BIEDMA	2013	28.8	26.1	20.7	21.3	16.7	14.4	12	13	14.6	19.8	24.5
2	Estancia La Adela - Península Valdés	BIEDMA	2014	27.8	26	23	17.3	16.4	13.8	12.5	16.1	16.8	19.5	23.7
3	Estancia La Adela - Península Valdés	BIEDMA	2015	27.5	26.6	26.3	21		14.3			15.4	17.3	
4	Estancia La Adela - Península Valdés	BIEDMA	2016	27.2	27.3	25.1	18.5							
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
514	Telsen	TELSEN	2010	33.2	29	27.4	22.2	18.3	12.9	11.5	15.3	19.3	24.2	27.5
515	Telsen	TELSEN	2011	32.4	32.1	27.9	22.8	19.4	14.4	13.6	13.9	20	22.7	28.9
516	Telsen	TELSEN	2012	33.5	29.6	27.5	21.4	18.4	13	13.4	17	21.2	23.9	28.4
517	Telsen	TELSEN	2013	33.3	27.3	22.9	24.4	17.7	15	12.3	14.2	16.9	25.4	29.4
518	Telsen	TELSEN	2014	33.3	31.2	28.1								

519 rows × 16 columns

```
In [7]: datos_tem_max_agrupados = datos_tem_max.groupby(by=
['Departamento', 'Año']).mean().round(2)
datos_tem_max_agrupados
```

Out[7]:	PROMEDIO_T_MAX		
	Departamento	Año	
	BIEDMA	2012	18.10
		2013	19.90
		2014	20.60
		2015	22.20
		2016	25.20
	...	...	...
	TELSEN	2018	17.63

PROMEDIO_T_MAX

Departamento	Año
	2019
	2020
	2021
	2022

248 rows × 1 columns

Cargamos el dataset de temperatura media, lo agrupamos por dpto y año

```
In [8]: datos_tem_med =
pd.read_csv('datos_sipas/Agrupado_por_dptos/temperatura_media_estaci
sep=",")
datos_tem_med
```

```
Out[8]:
```

	Estación	Departamento	Año	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV
0	Estancia La Adela - Península Valdés	BIEDMA	2012						8.8	7.9	9.7	11.7	13.3	16.9
1	Estancia La Adela - Península Valdés	BIEDMA	2013	20.9	19.7	15.6	16	12.1	9.7	7.9	8.8	9.1	13.6	16.9
2	Estancia La Adela - Península Valdés	BIEDMA	2014	20.7	19.5	17.3	13	11.7	9.5	7.6	11	11.3	13.6	16.6
3	Estancia La Adela - Península Valdés	BIEDMA	2015	20.4	19.9	19.4	15.5		10			9.5	11.1	
4	Estancia La Adela - Península Valdés	BIEDMA	2016	20.1	20.6	18.4	13.8							
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
516	Telsen	TELSEN	2010	22.7	20	18.6	13.2	10.2	7.2	4.5	6.8	10.1	15.2	18.1
517	Telsen	TELSEN	2011	23.3	22.2	18.3	13.3	10.2	7	6.6	6.3	12	14.2	19.6
518	Telsen	TELSEN	2012	24.5	20	17.4	12	10.4	6	5.5	8.3	11	14.8	19.4
519	Telsen	TELSEN	2013	24.3	20.2	16.1	15.6	9.9	8.1	5.3	7.2	8.8	15.4	19
520	Telsen	TELSEN	2014	22.6	21	18.1								

521 rows × 16 columns

In [9]:

```
datos_tem_med_agrupados = datos_tem_med.groupby(by=[
    'Departamento', 'Año']).mean().round(2)
datos_tem_med_agrupados
```

Out[9]:

PROMEDIO_T_MED		
Departamento	Año	
BIEDMA	2012	12.40
	2013	14.20
	2014	14.70
	2015	15.80
	2016	18.60
...	...	...
TELSEN	2018	11.33
	2019	11.17
	2020	10.60
	2021	12.37
	2022	20.90

250 rows × 1 columns

Cargamos el dataset de temperatura mínima, lo agrupamos por dpto y año

In [10]:

```
datos_tem_min =
pd.read_csv('datos_sipas/Agrupado_por_dptos/temperatura_minima_estac
sep=",")
datos_tem_min
```

Out[10]:

	Estación	Departamento	Año	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV
0	Estancia La Adela - Península Valdés	BIEDMA	2012						3.3	2.8	4.6	6	7.4	9.6
1	Estancia La Adela - Península Valdés	BIEDMA	2013	14.1	14.2	11.1	12.2	7.2	4.2	3.3	4.4	4.1	7.8	10.7
2	Estancia La Adela - Península Valdés	BIEDMA	2014	15	13.5	11.9	8.1	6.2	4.4	1.5	5.8	6.5	8.5	10.4
3	Estancia La Adela	BIEDMA	2015	14.5	14	13.5	10.9		5.1			3.9	5.3	

	Estación	Departamento	Año	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV
	-													
	Península													
	Valdés													
	Estancia													
	La Adela													
4	-	BIEDMA	2016	14.2	15.3	12.7	8.9							
	Península													
	Valdés													
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
516	Telsen	TELSEN	2010	10.9	11	10.1	5.2	3.2	1.8	-1.7	-1.5	0.8	5.6	8.1
517	Telsen	TELSEN	2011	13	12.1	9.2	4.1	1.7	0.8	0	-0.4	4.5	5.1	9.1
518	Telsen	TELSEN	2012	14.9	10.3	7.6	3.2	3.1	-0.3	-2.4	0.8	1.8	5.3	9.1
519	Telsen	TELSEN	2013	14.1	13.2	9.6	7.7	3.4	1.4	-0.5	0.8	1.5	4.8	8.1
520	Telsen	TELSEN	2014	10.6	10.8	7.9								

521 rows × 16 columns

```
In [11]: datos_tem_min_agrupados = datos_tem_min.groupby(by=
['Departamento', 'Año']).mean().round(2)
datos_tem_min_agrupados
```

Out[11]:

PROMEDIO_T_MIN		
Departamento	Año	
BIEDMA	2012	6.70
	2013	8.80
	2014	9.20
	2015	10.20
	2016	12.90
...	...	...
TELSEN	2018	5.03
	2019	4.83
	2020	4.63
	2021	5.90
	2022	13.05

250 rows × 1 columns

Lo siguiente es juntar todos los anteriores dataframes para armar el dataset

```
In [20]: import functools

dfs = [datos_humedad_agrupados,
datos_lluvia_agrupados,
```

```

    datos_tem_max_agrupados,
    datos_tem_med_agrupados,
    datos_tem_min_agrupados]

df_final = functools.reduce(lambda left, right:
pd.merge(left, right, left_index=True, right_index=True), dfs)
df_final

```

```

Out[20]:

```

		PROMEDIO_HUMEDAD	TOTAL_LLUVIA	PROMEDIO_T_MAX	PROMEDIO_T_MI
Departamento	Año				
BIEDMA	2012	58.40	46.4	18.10	12
	2013	64.70	226.5	19.90	14
	2014	60.50	331.3	20.60	14
	2015	58.70	114.2	22.20	15
	2016	57.50	88.9	25.20	18
...	...	...	...	...	
TELSEN	2018	47.93	604.2	17.63	11
	2019	47.57	358.4	17.53	11
	2020	46.63	304.4	16.60	10
	2021	47.03	608.9	18.97	12
	2022	37.15	58.7	28.50	20

209 rows × 5 columns

Reiniciamos los índices, renombramos las columnas y convertimos el año en tipo fecha para que nos sea más fácil unirlo con el dataset de producción.

```

In [21]:

```

```

datos_clima = df_final.reset_index()
datos_clima

```

```

Out[21]:

```

	Departamento	Año	PROMEDIO_HUMEDAD	TOTAL_LLUVIA	PROMEDIO_T_MAX	PROMEDIO_T_MI
0	BIEDMA	2012	58.40	46.4	18.10	
1	BIEDMA	2013	64.70	226.5	19.90	
2	BIEDMA	2014	60.50	331.3	20.60	
3	BIEDMA	2015	58.70	114.2	22.20	
4	BIEDMA	2016	57.50	88.9	25.20	
...	...	...	...	...	...	
204	TELSEN	2018	47.93	604.2	17.63	
205	TELSEN	2019	47.57	358.4	17.53	
206	TELSEN	2020	46.63	304.4	16.60	
207	TELSEN	2021	47.03	608.9	18.97	

	Departamento	Año	PROMEDIO_HUMEDAD	TOTAL_LLUVIA	PROMEDIO_T_MAX	PROMEDIO_T_MIN
208	TELSEN	2022	37.15	58.7	28.50	

209 rows × 7 columns

```
In [22]: datos_clima.columns = ['departamento', 'anio', 'PROMEDIO_HUMEDAD',
                                'TOTAL_LLUVIA',
                                'PROMEDIO_T_MAX', 'PROMEDIO_T_MED', 'PROMEDIO_T_MIN']
datos_clima['anio'] = pd.to_datetime(datos_clima['anio'],
format="%Y")
datos_clima
```

```
Out[22]:
```

	departamento	anio	PROMEDIO_HUMEDAD	TOTAL_LLUVIA	PROMEDIO_T_MAX	PROMEDIO_T_MIN
0	BIEDMA	2012-01-01	58.40	46.4	18.10	
1	BIEDMA	2013-01-01	64.70	226.5	19.90	
2	BIEDMA	2014-01-01	60.50	331.3	20.60	
3	BIEDMA	2015-01-01	58.70	114.2	22.20	
4	BIEDMA	2016-01-01	57.50	88.9	25.20	
...	...	...	...	...	...	
204	TELSEN	2018-01-01	47.93	604.2	17.63	
205	TELSEN	2019-01-01	47.57	358.4	17.53	
206	TELSEN	2020-01-01	46.63	304.4	16.60	
207	TELSEN	2021-01-01	47.03	608.9	18.97	
208	TELSEN	2022-01-01	37.15	58.7	28.50	

209 rows × 7 columns

```
In [23]: datos_clima.to_csv('datos_producidos/clima_chubut.csv',
index=False)
```

1.7. Analizando dataset existencias ovinas 2008-2020 "SENASA"

Introducción

Los dataset de este jupyter se obtuvieron de [Datos SENASA](#); y contienen información sobre existencias ovinas dentro del país. Son datos totales anuales desde Marzo 2008 a la última actualización, por provincia y partido / departamento, clasificadas por categorías.

No existe documentación que nos dé información sobre el dataset. Son 4 archivos.xlsx:

- existencias_ovinas_por_categoria_y_departamento_2017_20081.xlsx
- existencias_ovinas_por_departamento_marzo_2018.xlsx
- existencias_ovinas_por_departamento_marzo_2019.xlsx
- existencias_ovinas_por_departamento_marzo_2020.xlsx

Lo que se hizo fue convertirlos a todos en archivos csv para trabajarlos mejor, y se desmenuzó el archivo que contiene datos de 2008 a 2017 anualmente. Por lo tanto se obtuvieron 12 archivos csv.

Así que vamos a cargarlos para ver su contenido y empezar el análisis.

Análisis

Importamos las librerías de python que necesitamos para analizar el dataset

```
In [1]: import pandas as pd
```

La librería pandas nos ayudará a leer y escribir datos en diferentes formatos, seleccionar y filtrar de manera sencilla tablas de datos en función de posición, valor o etiquetas, a fusionar y unir datos, entre muchas cosas más.

Acomodando el dataset

Solo existe una columna para depto y provincia, nosotros vamos a necesitar identificar a qué provincia pertenece un depto, así que vamos a crear un arreglo que contenga las provincias con las características que encontramos en el dataset:

```
In [2]: from unicode import unicode
provincias = ['Buenos Aires', 'Catamarca', 'Chaco', 'Chubut',
              'Ciudad Autónoma de Buenos Aires', 'Córdoba', 'Corrientes',
              'Entre Ríos', 'Formosa', 'Jujuy', 'La Pampa', 'La Rioja',
              'Mendoza', 'Misiones', 'Neuquén', 'Río Negro', 'Salta',
              'San Juan',
              'San Luis', 'Santa Cruz', 'Santa Fe', 'Santiago del
              Estero',
              'Tierra del Fuego', 'Tucumán']
```

```
provincias = [unicode(x) for x in provincias]
provincias = [x.upper() for x in provincias]

provincias
```

```
Out[2]: ['BUENOS AIRES',
        'CATAMARCA',
        'CHACO',
        'CHUBUT',
        'CIUDAD AUTONOMA DE BUENOS AIRES',
        'CORDOBA',
        'CORRIENTES',
        'ENTRE RIOS',
        'FORMOSA',
        'JUJUY',
        'LA PAMPA',
        'LA RIOJA',
        'MENDOZA',
        'MISSIONES',
        'NEUQUEN',
        'RIO NEGRO',
        'SALTA',
        'SAN JUAN',
        'SAN LUIS',
        'SANTA CRUZ',
        'SANTA FE',
        'SANTIAGO DEL ESTERO',
        'TIERRA DEL FUEGO',
        'TUCUMAN']
```

El siguiente inconveniente es que existen dos "deptos" que tienen el mismo nombre que sus "provincias":

- Formosa
- Ciudad Autónoma de Buenos Aires

Esto va a dificultar la búsqueda por lo tanto lo que se hizo fue cambiar en los dataset de manera manual la "provincia" Ciudad Autónoma de Buenos Aires por Capital Federal y el depto Formosa por Formosa_d. Así que vamos a reflejar este cambio en el arreglo creado anteriormente para las provincias:

```
In [3]: provincias.insert(1, 'CAPITAL FEDERAL')
del provincias[5]
provincias
```

```
Out[3]: ['BUENOS AIRES',
        'CAPITAL FEDERAL',
        'CATAMARCA',
        'CHACO',
        'CHUBUT',
        'CORDOBA',
        'CORRIENTES',
        'ENTRE RIOS',
        'FORMOSA',
        'JUJUY',
        'LA PAMPA',
```

```
'LA RIOJA',
'MENDOZA',
'MISIONES',
'NEUQUEN',
'RIO NEGRO',
'SALTA',
'SAN JUAN',
'SAN LUIS',
'SANTA CRUZ',
'SANTA FE',
'SANTIAGO DEL ESTERO',
'TIERRA DEL FUEGO',
'TUCUMAN']
```

Cargamos el dataset del 2008 y observamos su contenido

```
In [4]: existencias_2008 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2008.csv',
sep=",")
existencias_2008
```

```
Out[4]:
```

	Provincia/Partido	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	96623	1231573	114101	45574	238336	1726207
1	AYACUCHO	9615	58613	10743	9582	17351	105904
2	CAMPANA	52	1310	26	-	321	1709
3	ZARATE	86	795	40	39	288	1248
4	PUNTA INDIO	62	4142	211	2	36	4453
...	...	...	...	...	...	...	...
495	CHICLIGASTA	2	15	-	-	-	17
496	LULES	2	9	-	-	2	13
497	YERBA BUENA	2	30	2	-	-	34
498	SAN MIGUEL DE TUCUMAN	-	19	-	-	-	19
499	Total	603403	8898738	2231588	1608096	2646298	15988123

500 rows × 7 columns

Es un dataset de 500 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [5]: existencias_2008.columns
```

```
Out[5]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
               'CORDEROS/AS', 'TOTAL OVINOS'],
              dtype='object')
```

En este dataset no hay datos de la Ciudad Autónoma de Buenos Aires, por lo tanto se va a realizar una copia del arreglo de provincias donde se va a quitar esa "provincia" y se va a trabajar con esta copia para este

año.

```
In [6]: provincias_copia = provincias
del provincias_copia[1]
provincias_copia
```

```
Out[6]: ['BUENOS AIRES',
'CATAMARCA',
'CHACO',
'CHUBUT',
'CORDOBA',
'CORRIENTES',
'ENTRE RIOS',
'FORMOSA',
'JUJUY',
'LA PAMPA',
'LA RIOJA',
'MENDOZA',
'MISIONES',
'NEUQUEN',
'RIO NEGRO',
'SALTA',
'SAN JUAN',
'SAN LUIS',
'SANTA CRUZ',
'SANTA FE',
'SANTIAGO DEL ESTERO',
'TIERRA DEL FUEGO',
'TUCUMAN']
```

Para filtrar mejor los datos se creará un dataframe donde existan las columnas provincia y departamento por separado. Y aquellas filas donde las provincias y deptos tengan el mismo nombre representan el total de la provincia. La última fila del data frame representa el total del país (Dice provincia: Tucuman pero seria total). Además se agregaron dos columnas más llamadas *establecimientos* y *unidades_productivas* porque hay dataset que las contienen, y para aquellos que no las contengan sus valores serán Nan.

```
In [7]: df_separado_2008 = pd.DataFrame(columns=['anio', 'provincia',
'departamento', 'carneros', 'ovejas', 'borregos', 'caponos',
'corderos_corderas', 'total_ovinos'])
provincia_actual = -1
for i in existencias_2008.index:
    if existencias_2008['Provincia/Partido'][i] in
provincias_copia:
        provincia_actual+=1

    df_separado_2008 = df_separado_2008.append({
                                                'anio': 2008,
                                                'provincia':
provincias_copia[provincia_actual],
```



```

'departamento':existencias_2008['Provincia/Partido'][i],
                                'carneros':existencias_2008['
CARNEROS'][i],
                                'ovejas':existencias_2008['
OVEJAS'][i],
                                'borregos':existencias_2008['
BORREGOS'][i],
                                'capones':existencias_2008['
CAPONES'][i],

'corderos_corderas':existencias_2008[' CORDEROS/AS'][i],

'total_ovinos':existencias_2008[' TOTAL OVINOS'][i],
                                },
                                ignore_index=True)

df_separado_2008

```

```

Out[7]:

```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovin
0	2008	BUENOS AIRES	BUENOS AIRES	96623	1231573	114101	45574	238336	17262
1	2008	BUENOS AIRES	AYACUCHO	9615	58613	10743	9582	17351	1059
2	2008	BUENOS AIRES	CAMPANA	52	1310	26	-	321	17
3	2008	BUENOS AIRES	ZARATE	86	795	40	39	288	12
4	2008	BUENOS AIRES	PUNTA INDIO	62	4142	211	2	36	42
...	...	...	...	...	...	...	...	...	...
495	2008	TUCUMAN	CHICLIGASTA	2	15	-	-	-	-
496	2008	TUCUMAN	LULES	2	9	-	-	2	-
497	2008	TUCUMAN	YERBA BUENA	2	30	2	-	-	-
498	2008	TUCUMAN	SAN MIGUEL DE TUCUMAN	-	19	-	-	-	-
499	2008	TUCUMAN	Total	603403	8898738	2231588	1608096	2646298	159881

500 rows × 9 columns

Una vez identificadas las provincias ahora es mucho más sencillo filtrar por Chubut para obtener las existencias.

```

In [8]: resultado_existencias_2008_chubut = pd.DataFrame(columns=
['anio', 'provincia', 'carneros', 'ovejas', 'borregos', 'capones',

```

```

'corderos_corderas', 'total_ovinos'])
for i in df_separado_2008.index:
    if "CHUBUT" == df_separado_2008['provincia'][i]:
        resultado_existencias_2008_chubut =
resultado_existencias_2008_chubut.append({
            'anio':
df_separado_2008['anio'][i],
            'provincia':
df_separado_2008['provincia'][i],
            'departamento':df_separado_2008['departamento'][i],
            'carneros':df_separado_2008['carneros'][i],
            'ovejas':df_separado_2008['ovejas'][i],
            'borregos':df_separado_2008['borregos'][i],
            'capones':df_separado_2008['capones'][i],
            'corderos_corderas':df_separado_2008['corderos_corderas'][i],
            'total_ovinos':df_separado_2008['total_ovinos'][i],},
            ignore_index=True)
resultado_existencias_2008_chubut

```

Out[8]:

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
0	2008	CHUBUT	135075	2160564	885807	931734	766298	4879478	CHUBUT
1	2008	CHUBUT	2256	44272	18756	20551	14559	100394	RAWSON
2	2008	CHUBUT	5292	112497	32493	33649	28886	212817	GAIMAN
3	2008	CHUBUT	13875	164382	79863	63856	48809	370785	FLORENTINO AMEGHINO
4	2008	CHUBUT	9253	153127	68968	65652	42389	339389	PASO DE INDIOS
5	2008	CHUBUT	7388	75160	43779	32683	22104	181114	MARTIRES
6	2008	CHUBUT	14960	221763	107149	126501	109720	580093	TEHUELCHES
7	2008	CHUBUT	8180	149189	41669	51128	37589	287755	CUSHAMEN
8	2008	CHUBUT	7096	160400	53115	74594	89234	384439	LANGUIÑEO
9	2008	CHUBUT	21728	325347	136018	119163	147472	749728	RIO SENGUERR
10	2008	CHUBUT	7756	140502	68341	66391	45572	328562	GASTRE
11	2008	CHUBUT	9052	173966	76783	98535	39250	397586	TELSEN

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
12	2008	CHUBUT	5639	90494	30119	24634	28237	179123	FUTALEUFU
13	2008	CHUBUT	7686	121561	58576	69633	33876	291332	VIDEBA
14	2008	CHUBUT	7863	109225	35581	40786	41516	234971	ESCALANTE
15	2008	CHUBUT	7051	118679	34597	43978	37085	241390	SARMIENTO

De esta manera se obtuvo las existencias ovinas de Chubut del año 2008 divididas por departamento, guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [9]: resultado_existencias_2008_chubut.to_csv('datos_producidos/senasa/re
index=False)
```

Generalización para analizar datasets

Vamos a generalizar lo que hicimos antes creando dos funciones

Como se hizo antes para filtrar mejor los datos se creará una función que cree un dataframe donde existan las columnas provincia y departamento por separado. Y aquellas filas donde las provincias y deptos tengan el mismo nombre representan el total de la provincia. La última fila del data frame representa el total del país (Dice provincia: Tucuman pero seria total de Argentina, como no vamos a trabajar con ella no le vamos a dar importancia). Además se agregaron dos columnas más llamadas establecimientos y unidades_productivas porque hay dataset que las contienen, y para aquellos que no las contengan sus valores serán Nan.

```
In [10]: def separar_en_provincias(existencias, anio):
df_separado = pd.DataFrame(columns=[ 'anio',
                                     'provincia',
                                     'departamento',
                                     'carneros',
                                     'ovejas',
                                     'borregos',
                                     'capones',
                                     'corderos_corderas',
                                     'total_ovinos'])

provincia_actual = -1
for i in existencias.index:
    if existencias['Provincia/Partido'][i] in provincias:
        provincia_actual+=1

    df_separado = df_separado.append({
                                     'anio': anio,
                                     'provincia':
provincias[provincia_actual],
```

```

'departamento':existencias['Provincia/Partido'][i],

'carneros':existencias['CARNEROS'][i],

'ovejas':existencias['OVEJAS'][i],

'borregos':existencias['BORREGOS'][i],

'capones':existencias['CAPONES'][i],

'corderos_corderas':existencias['CORDEROS/AS'][i],

'total_ovinos':existencias['TOTAL OVINOS'][i],
                                },
                                ignore_index=True)

return df_separado

```

La siguiente función nos servirá para filtrar las existencias de Chubut, una vez identificadas las provincias.

In [11]:

```

def filtrar_existencias_chubut(df_seprado):
    existencias_chubut = pd.DataFrame(columns=['anio',
                                                'provincia',
                                                'departamento',
                                                'carneros',
                                                'ovejas',
                                                'borregos',
                                                'capones',

                                                'corderos_corderas',
                                                'total_ovinos'])

    for i in df_seprado.index:
        if "CHUBUT" == df_seprado['provincia'][i]:
            existencias_chubut = existencias_chubut.append({
                'anio':
df_seprado['anio'][i],
                'provincia':
df_seprado['provincia'][i],
                'departamento':df_seprado['departamento'][i],
                'carneros':df_seprado['carneros'][i],

```

```

'ovejas':df_seprado['ovejas'][i],

'borregos':df_seprado['borregos'][i],

'capones':df_seprado['capones'][i],

'corderos_corderas':df_seprado['corderos_corderas'][i],

'total_ovinos':df_seprado['total_ovinos'][i],},
                                ignore_index=True)

return existencias_chubut

```

Además de las funciones será necesario renombrar las columnas de los dataset cargados porque algunos contienen espacios demás que harían que la función falle. Para eso se va a crear una lista con los nombres de columnas deseados.

```

In [12]: columnas = ['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS',
                    'CAPONES',
                    'CORDEROS/AS', 'TOTAL OVINOS']

```

Cargamos el dataset del 2009 y observamos su contenido

```

In [13]: existencias_2009 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2009.csv',
sep=",")
existencias_2009

```

```

Out[13]:

```

	Provincia/Partido	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	96931	1142751	107580	45655	214226	1607143
1	ADOLFO ALSINA	721	11593	981	175	1615	15085
2	ALBERTI	113	1140	-	9	21	1283
3	ALMIRANTE BROWN	14	275	-	-	-	289
4	AMEGHINO	90	1058	118	33	72	1371
...	...	...	...	...	...	...	...
499	TAFI DEL VALLE	420	4520	1017	151	948	7056
500	TAFI VIEJO	73	648	52	3	25	801
501	TRANCAS	192	2437	128	48	233	3038
502	YERBA BUENA	2	30	2	-	-	34
503	Total	586930	8609142	2052680	1566723	2463026	15278501

504 rows × 7 columns

Es un dataset de 504 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [14]: existencias_2009.columns
```

```
Out[14]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',  
              'CORDEROS/AS', 'TOTAL OVINOS'],  
              dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [15]: existencias_2009.columns = columns  
         existencias_2009.columns
```

```
Out[15]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',  
              'CORDEROS/AS', 'TOTAL OVINOS'],  
              dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2009 divididas por departamento.

```
In [16]: df_separado_2009 = separar_en_provincias(existencias_2009, 2009)  
         resultado_existencias_2009_chubut =  
         filtrar_exitencias_chubut(df_separado_2009)  
         resultado_existencias_2009_chubut
```

```
Out[16]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2009	CHUBUT	CHUBUT	135414	2182913	857329	888470	678003	4742129
1	2009	CHUBUT	CUSHAMEN	7191	130124	30369	31453	44013	243150
2	2009	CHUBUT	ESCALANTE	7706	115049	31613	37919	29499	221786
3	2009	CHUBUT	FLORENTINO AMEGHINO	12056	150454	69972	59835	51674	343991
4	2009	CHUBUT	FUTALEUFU	5155	85752	28633	24140	26350	170030
5	2009	CHUBUT	GAIMAN	5816	123241	31526	33769	29740	224092
6	2009	CHUBUT	GASTRE	7816	147044	65197	70770	33487	324314
7	2009	CHUBUT	LANGUINEO	7422	150988	49788	53990	82899	345087
8	2009	CHUBUT	MARTIRES	6705	89223	38382	34714	16621	185645
9	2009	CHUBUT	PASO DE INDIOS	9665	161713	71004	70192	42575	355149
10	2009	CHUBUT	RAWSON	2520	52126	19419	22801	11899	108765
11	2009	CHUBUT	RIO SENGUERR	23858	349639	175695	126080	124871	800143
12	2009	CHUBUT	SARMIENTO	6509	108523	31668	38117	36111	220928
13	2009	CHUBUT	TEHUELCHES	17055	241352	99948	130175	100704	589234

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
14	2009	CHUBUT	TELSEN	7949	158437	58617	89187	24057	338247
15	2009	CHUBUT	VIEDMA	7991	119248	55498	65328	23503	271568

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [17]: resultado_existencias_2009_chubut.to_csv('datos_producidos/senasa/re
index=False)
```

Cargamos el dataset del 2010 y observamos su contenido

```
In [18]: existencias_2010 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2010.csv',
sep=",")
existencias_2010
```

```
Out[18]:
```

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	85926	1088099	94607	36517	204689	1509838
1	ADOLFO ALSINA	619	11016	860	236	1045	13776
2	ALBERTI	118	1050	4		124	1296
3	ALMIRANTE BROWN	17	272				289
4	AMEGHINO	155	1433	82	15	41	1726
...	...	...	...	...	...	...	...
499	TAFI DEL VALLE	347	4307	930	197	717	6498
500	TAFI VIEJO	57	498	31	3	16	605
501	TRANCAS	121	1872	81	27	132	2233
502	YERBA BUENA	2	30				32
503	Total	564332	8580056	1965516	1487688	2427154	15024746

504 rows × 7 columns

Es un dataset de 504 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [19]: existencias_2010.columns
```

```
Out[19]: Index(['Provincia/Departamento', ' CARNEROS', ' OVEJAS', ' BORREGOS',
' CAPONES', ' CORDEROS/AS', ' TOTAL OVINOS'],
dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [20]: existencias_2010.columns = columnas
         existencias_2010.columns
```

```
Out[20]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
               'CORDEROS/AS', 'TOTAL OVINOS'],
              dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2010 divididas por departamento.

```
In [21]: df_separado_2010 = separar_en_provincias(existencias_2010, 2010)
         resultado_existencias_2010_chubut =
         filtrar_existencias_chubut(df_separado_2010)
         resultado_existencias_2010_chubut
```

```
Out[21]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2010	CHUBUT	CHUBUT	131249	2202419	777797	879509	612109	4603083
1	2010	CHUBUT	CUSHAMEN	6649	128141	30782	31756	40774	238102
2	2010	CHUBUT	ESCALANTE	6911	103115	29139	41199	39746	220110
3	2010	CHUBUT	FLORENTINO AMEGHINO	11907	169295	65937	60837	42462	350438
4	2010	CHUBUT	FUTALEUFU	5781	91721	30329	24897	21252	173980
5	2010	CHUBUT	GAIMAN	5553	129160	28238	42120	29240	234311
6	2010	CHUBUT	GASTRE	7335	148405	53470	63719	31541	304470
7	2010	CHUBUT	LANGUINEO	9310	179649	68180	54826	70734	382699
8	2010	CHUBUT	MARTIRES	6281	101923	30022	30752	18785	187763
9	2010	CHUBUT	PASO DE INDIOS	10084	164352	65580	66501	39919	346436
10	2010	CHUBUT	RAWSON	2132	52746	17783	24333	14075	111069
11	2010	CHUBUT	RIO SENGUERR	22663	338907	151755	135432	105030	753787
12	2010	CHUBUT	SARMIENTO	6518	112088	31817	42015	27872	220310
13	2010	CHUBUT	TEHUELCHES	16062	235053	97593	123094	89000	560802
14	2010	CHUBUT	TELSEN	7307	137395	38403	76175	18263	277543
15	2010	CHUBUT	VIEDMA	6756	110469	38769	61853	23416	241263

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [22]: resultado_existencias_2010_chubut.to_csv('datos_producidos/senasa/re
         index=False)
```

Cargamos el dataset del 2011 y observamos su contenido


```
In [23]: existencias_2011 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2011.csv',
sep=",")
existencias_2011
```

```
Out[23]:
```

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	91517	1043715	98959	40301	258795	1533287
1	ADOLFO ALSINA	670	10633	849	282	1671	14105
2	ALBERTI	118	1118	5	51	451	1743
3	ALMIRANTE BROWN	13	172	-	-	-	185
4	AMEGHINO	152	1393	58	15	77	1695
...	...	...	...	...	...	...	...
500	TAFI DEL VALLE	387	4475	968	233	900	6963
501	TAFI VIEJO	104	892	33	4	18	1051
502	TRANCAS	158	2273	117	43	241	2832
503	YERBA BUENA	14	39	-	-	-	53
504	Total	576162	8436154	1938866	1417208	2363756	14732146

505 rows × 7 columns

Es un dataset de 505 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [24]: existencias_2011.columns
```

```
Out[24]: Index(['Provincia/Departamento', 'CARNEROS', 'OVEJAS', 'BORREGOS',
'CAPONES', 'CORDEROS/AS', 'TOTAL OVINOS'],
dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [25]: existencias_2011.columns = columns
existencias_2011.columns
```

```
Out[25]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
'CORDEROS/AS', 'TOTAL OVINOS'],
dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2011 divididas por departamento.

```
In [26]: df_separado_2011 = separar_en_provincias(existencias_2011, 2011)
resultado_existencias_2011_chubut =
```

```
filtrar_existencias_chubut(df_separado_2011)
resultado_existencias_2011_chubut
```

Out[26]:

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2011	CHUBUT	CHUBUT	132272	2119111	762246	844921	580441	4438991
1	2011	CHUBUT	CUSHAMEN	6937	134263	36860	32524	39295	249879
2	2011	CHUBUT	ESCALANTE	6993	97711	31108	37970	36293	210075
3	2011	CHUBUT	FLORENTINO AMEGHINO	11347	153875	60216	58589	38162	322189
4	2011	CHUBUT	FUTALEUFU	5405	94013	33600	22878	27973	183869
5	2011	CHUBUT	GAIMAN	5584	132221	27090	42712	27345	234952
6	2011	CHUBUT	GASTRE	7142	138313	48031	59875	32771	286132
7	2011	CHUBUT	LANGUINEO	9005	146047	58757	54040	50261	318110
8	2011	CHUBUT	MARTIRES	6936	102036	30312	30909	18902	189095
9	2011	CHUBUT	PASO DE INDIOS	10475	162623	57581	71825	38175	340679
10	2011	CHUBUT	RAWSON	2165	51998	17953	23373	13380	108869
11	2011	CHUBUT	RIO SENGUERR	24926	348657	160536	146116	105368	785603
12	2011	CHUBUT	SARMIENTO	6318	113645	29595	39374	27014	215946
13	2011	CHUBUT	TEHUELCHES	15804	221365	95119	100380	84103	516771
14	2011	CHUBUT	TELSEN	6926	124528	38959	70018	19568	259999
15	2011	CHUBUT	VIEDMA	6309	97816	36529	54338	21831	216823

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

In [27]:

```
resultado_existencias_2011_chubut.to_csv('datos_producidos/senasa/resultado_existencias_2011_chubut.csv',
index=False)
```

Cargamos el dataset del 2012 y observamos su contenido

In [28]:

```
existencias_2012 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2012.csv',
sep=",")
existencias_2012
```

Out[28]:

	Provincia/Departamento	Carnero	Oveja	Borrego/a	Capon	Cordero/a	Ovinos
0	BUENOS AIRES	91304	1144417	124521	53895	313958	1728095
1	ADOLFO ALSINA	871	11617	1182	296	2019	15985
2	ALBERTI	115	1212	1	51	412	1791
3	ALMIRANTE BROWN	9	188				197
4	AMEGHINO	157	1685	58	15	87	2002
...	...	...	...	...	...	...	...

	Provincia/Departamento	Carnero	Oveja	Borrego/a	Capon	Cordero/a	Ovinos
526	TAFI DEL VALLE	404	4425	1014	285	1022	7150
527	TAFI VIEJO	95	862	33	3	18	1011
528	TRANCAS	140	1870	101	34	200	2345
529	YERBA BUENA	14	39				53
530	Total	571740	8397582	1900922	1372168	2454491	14696903

531 rows × 7 columns

Es un dataset de 531 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [29]: existencias_2012.columns
```

```
Out[29]: Index(['Provincia/Departamento', 'Carnero', 'Oveja', 'Borrego/a',
              'Capon', 'Cordero/a', 'Ovinos'],
              dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [30]: existencias_2012.columns = columns
          existencias_2012.columns
```

```
Out[30]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
               'CORDEROS/AS', 'TOTAL OVINOS'],
               dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2012 divididas por departamento.

```
In [31]: df_separado_2012 = separar_en_provincias(existencias_2012, 2012)
          resultado_existencias_2012_chubut =
          filtrar_exitencias_chubut(df_separado_2012)
          resultado_existencias_2012_chubut
```

```
Out[31]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2012	CHUBUT	CHUBUT	126443	1998999	689806	792843	555331	4163422
1	2012	CHUBUT	CUSHAMEN	6758	126283	35504	27867	30258	226670
2	2012	CHUBUT	ESCALANTE	7159	90741	30302	36071	36469	200742
3	2012	CHUBUT	FLORENTINO AMEGHINO	10531	142512	56336	49490	52955	311824
4	2012	CHUBUT	FUTALEUFU	5713	91189	32426	23200	27788	180316
5	2012	CHUBUT	GAIMAN	5977	123977	20057	45528	23647	219186
6	2012	CHUBUT	GASTRE	7183	129262	46412	56339	23340	262536
7	2012	CHUBUT	LANGUIÑEO	8550	141153	52659	54014	45432	301808

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
8	2012	CHUBUT	MARTIRES	5381	90578	23501	31523	10211	161194
9	2012	CHUBUT	PASO DE INDIOS	8980	150846	37809	64028	32809	294472
10	2012	CHUBUT	RAWSON	2263	53025	15448	22120	11071	103927
11	2012	CHUBUT	RIO SENGUERR	25366	365526	159817	139353	140285	830347
12	2012	CHUBUT	SARMIENTO	6400	100390	29363	34584	24672	195409
13	2012	CHUBUT	TEHUELCHES	14493	195434	91941	98158	66160	466186
14	2012	CHUBUT	TELSEN	5904	104705	28063	62121	14163	214956
15	2012	CHUBUT	VIEDMA	5785	93378	30168	48447	16071	193849

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [32]: resultado_existencias_2012_chubut.to_csv('datos_producidos/senasa/re
index=False)
```

Cargamos el dataset del 2013 y observamos su contenido

```
In [33]: existencias_2013 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2013.csv',
sep=",")
existencias_2013
```

```
Out[33]:
```

	Provincia/Departamento	Carnero	Oveja	Borrego/a	Capon	Cordero/a	Ovinos
0	BUENOS AIRES	89675	1212290	132867	60615	343367	1838814
1	ADOLFO ALSINA	836	11951	1752	371	2858	17768
2	ALBERTI	127	1316	1	42	498	1984
3	ALMIRANTE BROWN	9	141	4			154
4	AMEGHINO	178	1960	52	15	91	2296
...	...	...	...	...	...	...	...
502	TAFI DEL VALLE	367	4212	1014	249	878	6720
503	TAFI VIEJO	99	869	33	4	18	1023
504	TRANCAS	140	1842	97	34	193	2306
505	YERBA BUENA	11	31				42
506	Total	585831	8325029	1887544	1312906	2463579	14574889

507 rows × 7 columns

Es un dataset de 507 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [34]: existencias_2013.columns
```

```
Out[34]: Index(['Provincia/Departamento', '    Carnero', '    Oveja', '    Borrego/a',
            '    Capon', '    Cordero/a', '    Ovinos'],
            dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [35]: existencias_2013.columns = columnas
         existencias_2013.columns
```

```
Out[35]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
            'CORDEROS/AS', 'TOTAL OVINOS'],
            dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2013 divididas por departamento.

```
In [36]: df_separado_2013 = separar_en_provincias(existencias_2013, 2013)
         resultado_existencias_2013_chubut =
         filtrar_exitencias_chubut(df_separado_2013)
         resultado_existencias_2013_chubut
```

```
Out[36]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2013	CHUBUT	CHUBUT	133292	1967599	649453	725939	555396	4031679
1	2013	CHUBUT	BIEDMA	5376	82998	25341	47350	26832	187897
2	2013	CHUBUT	CUSHAMEN	12516	125655	31127	22615	33934	225847
3	2013	CHUBUT	ESCALANTE	9153	97660	30550	38523	28023	203909
4	2013	CHUBUT	FLORENTINO AMEGHINO	9623	131947	59697	42035	41337	284639
5	2013	CHUBUT	FUTALEUFU	5210	89861	32255	21340	25789	174455
6	2013	CHUBUT	GAIMAN	5802	138573	20059	45098	20514	230046
7	2013	CHUBUT	GASTRE	7568	115666	42937	53335	24395	243901
8	2013	CHUBUT	LANGUIÑEO	8875	130228	45552	42406	21887	248948
9	2013	CHUBUT	MARTIRES	5489	88395	22760	28779	14587	160010
10	2013	CHUBUT	PASO DE INDIOS	8339	143441	36190	59153	28510	275633
11	2013	CHUBUT	RAWSON	2773	57599	14701	21929	15736	112738
12	2013	CHUBUT	RIO SENGUERR	25403	384024	144490	136655	143717	834289
13	2013	CHUBUT	SARMIENTO	6978	102420	31288	33644	27841	202171
14	2013	CHUBUT	TEHUELCHES	14577	187946	88690	80677	85841	457731
15	2013	CHUBUT	TELSEN	5610	91186	23816	52400	16453	189465
16	2013	CHUBUT	CAPITAL FEDERAL	7	45	87	4		143

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
17	2013	CHUBUT	CIUDAD AUTONOMA DE BUENOS AIRES	7	45	87	4		143

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [37]: resultado_existencias_2013_chubut.to_csv('datos_producidos/senasa/re
index=False)
```

Cargamos el dataset del 2014 y observamos su contenido

```
In [38]: existencias_2014 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2014.cs
sep=",")
existencias_2014
```

```
Out[38]:
```

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	84738	1207240	138449	57843	355912	1844182
1	ADOLFO ALSINA	856	12504	1537	351	2941	18189
2	ALBERTI	137	1223	1	51	514	1926
3	ALMIRANTE BROWN	7	149	4			160
4	AMEGHINO	180	2042	52	15	91	2380
...	...	...	...	...	...	...	...
504	TAFI DEL VALLE	401	4719	1659	268	1069	8116
505	TAFI VIEJO	102	872	53	7	28	1062
506	TRANCAS	151	1972	117	43	267	2550
507	YERBA BUENA	11	31				42
508	Total	581433	8214688	1883954	1290437	2563186	14533698

509 rows × 7 columns

Es un dataset de 509 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [39]: existencias_2014.columns
```

```
Out[39]: Index(['Provincia/Departamento', ' CARNEROS', ' OVEJAS', ' BORREGOS',
' CAPONES', ' CORDEROS/AS', ' TOTAL OVINOS'],
dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [40]: existencias_2014.columns = columnas
         existencias_2014.columns
```

```
Out[40]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
               'CORDEROS/AS', 'TOTAL OVINOS'],
              dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2014 divididas por departamento.

```
In [41]: df_separado_2014 = separar_en_provincias(existencias_2014, 2014)
         resultado_existencias_2014_chubut =
         filtrar_existencias_chubut(df_separado_2014)
         resultado_existencias_2014_chubut
```

```
Out[41]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2014	CHUBUT	CHUBUT	128053	1941873	643744	701889	608489	4024048
1	2014	CHUBUT	CUSHAMEN	6525	120803	33531	29759	40184	230802
2	2014	CHUBUT	ESCALANTE	8435	91157	30847	39547	32422	202408
3	2014	CHUBUT	FLORENTINO AMEGHINO	9623	128662	54128	39480	50249	282142
4	2014	CHUBUT	FUTALEUFU	5126	85641	26607	20764	26403	164541
5	2014	CHUBUT	GAIMAN	7647	143403	21683	44605	27900	245238
6	2014	CHUBUT	GASTRE	7077	104867	40127	49703	30458	232232
7	2014	CHUBUT	LANGUINEO	7649	145930	43258	45030	43245	285112
8	2014	CHUBUT	MARTIRES	5694	83628	25610	27084	18243	160259
9	2014	CHUBUT	PASO DE INDIOS	8820	136765	38061	52569	29853	266068
10	2014	CHUBUT	RAWSON	3051	61699	15436	22215	18436	120837
11	2014	CHUBUT	RIO SENGUERR	25282	384360	139502	131200	132538	812882
12	2014	CHUBUT	SARMIENTO	7348	106471	31146	34529	30598	210092
13	2014	CHUBUT	TEHUELCHES	15833	181194	89063	82015	80224	448329
14	2014	CHUBUT	TELSEN	5158	84587	25366	48234	17836	181181
15	2014	CHUBUT	VIEDMA	4785	82706	29379	35155	29900	181925

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [42]: resultado_existencias_2014_chubut.to_csv('datos_producidos/senasa/re
         index=False)
```

Cargamos el dataset del 2015 y observamos su contenido

```
In [43]: existencias_2015 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2015.csv',
sep=",")
existencias_2015
```

```
Out[43]:
```

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	83683	1232714	142024	57472	373336	1889229
1	ADOLFO ALSINA	814	12268	1635	351	3026	18094
2	ALBERTI	135	1128	1	51	540	1855
3	ALMIRANTE BROWN	9	135	4			148
4	AMEGHINO	189	2065	77	15	93	2439
...	...	...	...	...	...	...	...
505	TAFI DEL VALLE	420	4532	1206	291	1180	7629
506	TAFI VIEJO	122	1026	97	13	78	1336
507	TRANCAS	197	2190	276	77	410	3150
508	YERBA BUENA	11	31				42
509	Total	606227	8499019	1926727	1267144	2560369	14859486

510 rows × 7 columns

Es un dataset de 510 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [44]: existencias_2015.columns
```

```
Out[44]: Index(['Provincia/Departamento', ' CARNEROS', ' OVEJAS', ' BORREGOS',
' CAPONES', ' CORDEROS/AS', ' TOTAL OVINOS'],
dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [45]: existencias_2015.columns = columnas
existencias_2015.columns
```

```
Out[45]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
'CORDEROS/AS', 'TOTAL OVINOS'],
dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2015 divididas por departamento.

```
In [46]: df_separado_2015 = separar_en_provincias(existencias_2015, 2015)
resultado_existencias_2015_chubut =
```



```
filtrar_existencias_chubut(df_separado_2015)
resultado_existencias_2015_chubut
```

Out[46]:

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2015	CHUBUT	CHUBUT	131095	1939450	637096	694326	600189	4002156
1	2015	CHUBUT	CUSHAMEN	5764	115205	30865	26410	36446	214690
2	2015	CHUBUT	ESCALANTE	6699	85329	31991	37533	25581	187133
3	2015	CHUBUT	FLORENTINO AMEGHINO	12635	145253	52847	46691	46046	303472
4	2015	CHUBUT	FUTALEUFU	5218	86528	23929	19766	25085	160526
5	2015	CHUBUT	GAIMAN	8503	161928	25067	44956	28410	268864
6	2015	CHUBUT	GASTRE	6059	100048	38465	47449	26759	218780
7	2015	CHUBUT	LANGUINEO	9928	116399	44518	44570	42894	258309
8	2015	CHUBUT	MARTIRES	5628	88899	27039	26984	17985	166535
9	2015	CHUBUT	PASO DE INDIOS	9418	135182	40238	52997	25057	262892
10	2015	CHUBUT	RAWSON	3308	64914	14568	22110	20932	125832
11	2015	CHUBUT	RIO SENGUERR	24115	373103	140325	126419	127228	791190
12	2015	CHUBUT	SARMIENTO	7298	105229	30923	35616	36312	215378
13	2015	CHUBUT	TEHUELCHES	17316	190599	82015	84630	87834	462394
14	2015	CHUBUT	TELSEN	4766	81572	24643	42758	21277	175016
15	2015	CHUBUT	VIEDMA	4440	89262	29663	35437	32343	191145

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

In [47]:

```
resultado_existencias_2015_chubut.to_csv('datos_producidos/senasa/resultado_existencias_2015_chubut.csv',
index=False)
```

Cargamos el dataset del 2016 y observamos su contenido

In [48]:

```
existencias_2016 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2016.csv',
sep=",")
existencias_2016
```

Out[48]:

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	83047	1270657	148745	58681	382607	1943737
1	ADOLFO ALSINA	759	11743	1746	306	3512	18066
2	ALBERTI	124	1189	1	51	513	1878
3	ALMIRANTE BROWN	8	145	4		130	287
4	AMEGHINO	189	2188	76	15	93	2561

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
...	...	...	...	...	...	...	...
506	TAFI DEL VALLE	470	5255	1332	321	1260	8638
507	TAFI VIEJO	122	1189	193	33	143	1680
508	TRANCAS	196	2147	281	103	454	3181
509	YERBA BUENA	1	2				3
510	Total	573235	8420305	1988929	1296240	2585612	14864321

511 rows × 7 columns

Es un dataset de 511 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [49]: existencias_2016.columns
```

```
Out[49]: Index(['Provincia/Departamento', 'CARNEROS', 'OVEJAS', 'BORREGOS',
               'CAPONES', 'CORDEROS/AS', 'TOTAL OVINOS'],
              dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [50]: existencias_2016.columns = columns
         existencias_2016.columns
```

```
Out[50]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
               'CORDEROS/AS', 'TOTAL OVINOS'],
              dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2016 divididas por departamento.

```
In [51]: df_separado_2016 = separar_en_provincias(existencias_2016, 2016)
         resultado_existencias_2016_chubut =
         filtrar_exitencias_chubut(df_separado_2016)
         resultado_existencias_2016_chubut
```

```
Out[51]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2016	CHUBUT	CHUBUT	130635	1947672	658063	703023	554340	3993733
1	2016	CHUBUT	CUSHAMEN	5920	113136	30649	28942	46037	224684
2	2016	CHUBUT	ESCALANTE	6834	89014	33000	42032	32902	203782
3	2016	CHUBUT	FLORENTINO AMEGHINO	11974	159866	63418	47987	25964	309209
4	2016	CHUBUT	FUTALEUFU	4568	77084	23247	15616	17473	137988
5	2016	CHUBUT	GAIMAN	8368	166144	27932	47610	33646	283700

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
6	2016	CHUBUT	GASTRE	6374	104994	40322	47036	28354	227080
7	2016	CHUBUT	LANGUINEO	7588	112547	47178	49593	17445	234351
8	2016	CHUBUT	MARTIRES	5529	89598	31956	26627	15073	168783
9	2016	CHUBUT	PASO DE INDIOS	9181	129835	43632	50592	26582	259822
10	2016	CHUBUT	RAWSON	3672	66374	17693	21512	19079	128330
11	2016	CHUBUT	RIO SENGUERR	26957	368581	131766	122579	134887	784770
12	2016	CHUBUT	SARMIENTO	7249	107299	29077	37383	30378	211386
13	2016	CHUBUT	TEHUELCHES	17073	200643	78060	92801	84904	473481
14	2016	CHUBUT	TELSEN	4896	80262	26471	39243	20290	171162
15	2016	CHUBUT	VIEDMA	4452	82295	33662	33470	21326	175205

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [52]: resultado_existencias_2016_chubut.to_csv('datos_producidos/senasa/resultado_existencias_2016_chubut.csv', index=False)
```

Cargamos el dataset del 2017 y observamos su contenido

```
In [53]: existencias_2017 = pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2017.csv', sep=",")
existencias_2017
```

```
Out[53]:
```

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	85279	1304142	146093	58877	402816	1997207
1	ADOLFO ALSINA	739	11733	1782	306	3504	18064
2	ALBERTI	132	1406	1	51	547	2137
3	ALMIRANTE BROWN	7	136	4		130	277
4	AMEGHINO	210	2279	79	15	93	2676
...	...	...	...	...	...	...	...
506	TAFI DEL VALLE	473	5491	1253	347	1359	8923
507	TAFI VIEJO	114	1217	176	45	188	1740
508	TRANCAS	215	2548	316	127	486	3692
509	YERBA BUENA	1	2				3
510	Total	567391	8371694	1986130	1260896	2560455	14746566

511 rows × 7 columns

Es un dataset de 511 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [54]: existencias_2017.columns
```

```
Out[54]: Index(['Provincia/Departamento', ' CARNEROS', ' OVEJAS', ' BORREGOS',  
              ' CAPONES', ' CORDEROS/AS', ' TOTAL OVINOS'],  
              dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [55]: existencias_2017.columns = columns  
         existencias_2017.columns
```

```
Out[55]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',  
              'CORDEROS/AS', 'TOTAL OVINOS'],  
              dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2017 divididas por departamento.

```
In [56]: df_separado_2017 = separar_en_provincias(existencias_2017, 2017)  
         resultado_existencias_2017_chubut =  
         filtrar_exitencias_chubut(df_separado_2017)  
         resultado_existencias_2017_chubut
```

```
Out[56]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2017	CHUBUT	CHUBUT	124046	1917298	652423	672122	606862	3972751
1	2017	CHUBUT	CUSHAMEN	5198	106723	31589	28842	46907	219259
2	2017	CHUBUT	ESCALANTE	7650	87939	29194	36140	24834	185757
3	2017	CHUBUT	FLORENTINO AMEGHINO	12216	157416	59969	46044	52072	327717
4	2017	CHUBUT	FUTALEUFU	4313	74028	24502	15556	16234	134633
5	2017	CHUBUT	GAIMAN	8272	170790	28217	47586	37463	292328
6	2017	CHUBUT	GASTRE	6122	96368	37424	43738	29309	212961
7	2017	CHUBUT	LANGUINEO	6650	103552	38645	40482	40508	229837
8	2017	CHUBUT	MARTIRES	5150	88323	32027	24563	14832	164895
9	2017	CHUBUT	PASO DE INDIOS	7789	128838	43440	53838	30833	264738
10	2017	CHUBUT	RAWSON	4146	69722	19404	20173	19231	132676
11	2017	CHUBUT	RIO SENGUERR	24665	369734	134929	122073	140185	791586
12	2017	CHUBUT	SARMIENTO	5892	91496	22810	33128	22739	176065
13	2017	CHUBUT	TEHUELCHES	16655	207634	89829	89442	93065	496625
14	2017	CHUBUT	TELSEN	4640	77556	27296	36570	17750	163812

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
15	2017	CHUBUT	VIEDMA	4688	87179	33148	33947	20900	179862

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [57]: resultado_existencias_2017_chubut.to_csv('datos_producidos/senasa/re
index=False)
```

Cargamos el dataset del 2018 y observamos su contenido

```
In [58]: existencias_2018 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2018.cs
sep=",")
existencias_2018
```

```
Out[58]:
```

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	87282	1324290	146833	59627	401982	2020014
1	ADOLFO ALSINA	778	11907	1832	311	3548	18376
2	ALBERTI	140	1473	1	51	621	2286
3	ALMIRANTE BROWN	9	227	23		130	389
4	AMEGHINO	246	2258	79	15	95	2693
...	...	...	...	...	...	...	...
509	TAFI DEL VALLE	494	5471	1297	399	1549	9210
510	TAFI VIEJO	112	1177	171	46	199	1705
511	TRANCAS	229	2603	405	171	505	3913
512	YERBA BUENA	2	3			3	8
513	Total	551077	8153999	1885998	1217169	2531665	14339908

514 rows × 7 columns

Es un dataset de 514 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [59]: existencias_2018.columns
```

```
Out[59]: Index(['Provincia/Departamento', ' CARNEROS', ' OVEJAS', ' BORREGOS',
' CAPONES', ' CORDEROS/AS', ' TOTAL OVINOS'],
dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [60]: existencias_2018.columns = columnas
         existencias_2018.columns
```

```
Out[60]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
               'CORDEROS/AS', 'TOTAL OVINOS'],
              dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2018 divididas por departamento.

```
In [61]: df_separado_2018 = separar_en_provincias(existencias_2018, 2018)
         resultado_existencias_2018_chubut =
         filtrar_existencias_chubut(df_separado_2018)
         resultado_existencias_2018_chubut
```

```
Out[61]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2018	CHUBUT	CHUBUT	120807	1929004	643675	656187	595247	3944920
1	2018	CHUBUT	CUSHAMEN	6510	114304	33488	29440	46476	230218
2	2018	CHUBUT	ESCALANTE	6943	87141	29811	33749	39510	197154
3	2018	CHUBUT	FLORENTINO AMEGHINO	11465	152819	61672	43911	46348	316215
4	2018	CHUBUT	FUTALEUFU	4096	68134	22201	13799	16000	124230
5	2018	CHUBUT	GAIMAN	7991	178598	27915	46278	38481	299263
6	2018	CHUBUT	GASTRE	6499	102714	39417	43561	35386	227577
7	2018	CHUBUT	LANGUINEO	6411	107526	43471	36192	20992	214592
8	2018	CHUBUT	MARTIRES	5662	90476	30854	22658	19385	169035
9	2018	CHUBUT	PASO DE INDIOS	8198	128880	47244	57146	32537	274005
10	2018	CHUBUT	RAWSON	3558	68936	20065	19143	21294	132996
11	2018	CHUBUT	RIO SENGUERR	22376	374068	120941	121585	129376	768346
12	2018	CHUBUT	SARMIENTO	5494	90067	22437	31258	26686	175942
13	2018	CHUBUT	TEHUELCHES	16005	199751	79560	88562	78554	462432
14	2018	CHUBUT	TELSEN	4832	79117	28366	37558	21933	171806
15	2018	CHUBUT	VIEDMA	4767	86473	36233	31347	22289	181109

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [62]: resultado_existencias_2018_chubut.to_csv('datos_producidos/senasa/re
         index=False)
```

Cargamos el dataset del 2019 y observamos su contenido

```
In [63]: existencias_2019 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2019.csv',
sep=",")
existencias_2019
```

```
Out[63]:
```

	Provincia/Departamento	CARNEROS	OVEJAS	BORREGOS	CAPONES	CORDEROS/AS	TOTAL OVINOS
0	BUENOS AIRES	87855	1351731	154330	61672	423245	2078833
1	ADOLFO ALSINA	655	10845	1660	306	3166	16632
2	ALBERTI	139	1393	1	51	666	2250
3	ALMIRANTE BROWN	9	224	28		136	397
4	AMEGHINO	174	2154	79	15	96	2518
...	...	...	...	...	...	...	...
509	TAFI DEL VALLE	508	5291	1302	440	1570	9111
510	TAFI VIEJO	109	1186	170	42	210	1717
511	TRANCAS	241	2796	461	175	521	4194
512	YERBA BUENA	2	3			3	8
513	Total	560417	8302740	2039408	1296183	2575448	14774196

514 rows × 7 columns

Es un dataset de 514 filas y 7 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut, miremos las columnas que trae:

```
In [64]: existencias_2019.columns
```

```
Out[64]: Index(['Provincia/Departamento', ' CARNEROS', ' OVEJAS', ' BORREGOS',
' CAPONES', ' CORDEROS/AS', ' TOTAL OVINOS'],
dtype='object')
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [65]: existencias_2019.columns = columnas
existencias_2019.columns
```

```
Out[65]: Index(['Provincia/Partido', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
'CORDEROS/AS', 'TOTAL OVINOS'],
dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2019 divididas por departamento.

```
In [66]: df_separado_2019 = separar_en_provincias(existencias_2019, 2019)
resultado_existencias_2019_chubut =
```

```
filtrar_existencias_chubut(df_separado_2019)
resultado_existencias_2019_chubut
```

Out[66]:

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2019	CHUBUT	CHUBUT	127625	1996422	681206	673858	597226	4076337
1	2019	CHUBUT	BIEDMA	4887	84400	31784	34032	23424	178527
2	2019	CHUBUT	CUSHAMEN	8292	124242	32843	28734	43428	237539
3	2019	CHUBUT	ESCALANTE	7040	96309	28636	36375	39591	207951
4	2019	CHUBUT	FLORENTINO AMEGHINO	12092	163230	66071	47475	40992	329860
5	2019	CHUBUT	FUTALEUFU	3956	64916	20087	12651	16165	117775
6	2019	CHUBUT	GAIMAN	8309	193501	29047	48760	45825	325442
7	2019	CHUBUT	GASTRE	6797	105531	40287	44725	34228	231568
8	2019	CHUBUT	LANGUIÑEO	5879	102562	38569	39746	41717	228473
9	2019	CHUBUT	MARTIRES	5781	91491	35610	24758	18565	176205
10	2019	CHUBUT	PASO DE INDIOS	9109	135988	53921	56329	28341	283688
11	2019	CHUBUT	RAWSON	3628	74009	19514	20800	18895	136846
12	2019	CHUBUT	RIO SENGUERR	25733	384841	138981	122343	125892	797790
13	2019	CHUBUT	SARMIENTO	6077	99175	28287	31325	28339	193203
14	2019	CHUBUT	TEHUELCHES	15264	196118	89141	87696	74093	462312
15	2019	CHUBUT	TELSEN	4781	80109	28428	38109	17731	169158

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

In [67]:

```
resultado_existencias_2019_chubut.to_csv('datos_producidos/senasa/re
index=False)
```

Cargamos el dataset del 2020 y observamos su contenido

In [68]:

```
existencias_2020 =
pd.read_csv('datos/existencias_ovinas_por_departamento_marzo_2020.cs
sep=",")
existencias_2020
```

Out[68]:

	Provincia/Departamento	ESTABLECIMIENTOS	UNIDADES PRODUCTIVAS	CARNEROS	OVEJAS	BORREGOS
0	BUENOS AIRES	25152	28912	55651	1401711	158349
1	ADOLFO ALSINA	260	275	446	10725	1672
2	ALBERTI	112	118	86	1508	15
3	ALMIRANTE BROWN	13	14	6	226	22
4	AMEGHINO	87	106	158	2448	102

	Provincia/Departamento	ESTABLECIMIENTOS	UNIDADES PRODUCTIVAS	CARNEROS	OVEJAS	BORREGOS
...	...	...	...	...	...	...
511	TAFI DEL VALLE	89	260	302	5426	129
512	TAFI VIEJO	43	66	79	1077	17
513	TRANCAS	126	169	164	2814	50
514	YERBA BUENA	2	2	2	3	
515	Total	98130	134205	355037	8365934	200905

516 rows × 9 columns

Es un dataset de 516 filas y 9 columnas, será necesario analizarlo en detenimiento para saber si contiene los deptos de Chubut.

```
In [69]: existencias_2020.columns
```

```
Out[69]: Index(['Provincia/Departamento', 'ESTABLECIMIENTOS', 'UNIDADES PRODUCTIVAS',
              'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES', 'CORDEROS/AS',
              'TOTAL OVINOS'],
              dtype='object')
```

Dado que la cantidad de columnas es diferente a lo que veníamos trabajando, se va a redinifir la variable columnas.

```
In [70]: columnas = ['Provincia/Partido', 'ESTABLECIMIENTOS', 'UNIDADES
PRODUCTIVAS', 'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES',
                    'CORDEROS/AS', 'TOTAL OVINOS']
```

Lo que se observa es que por el nombre de las columnas los datos son acordes a los que buscamos, pero los nombre de las columnas tienen un detalle: contienen espacios antes de la primer palabra. Para evitar que esto ocasione problemas se las renombrara.

```
In [71]: existencias_2020.columns = columnas
existencias_2020.columns
```

```
Out[71]: Index(['Provincia/Partido', 'ESTABLECIMIENTOS', 'UNIDADES PRODUCTIVAS',
              'CARNEROS', 'OVEJAS', 'BORREGOS', 'CAPONES', 'CORDEROS/AS',
              'TOTAL OVINOS'],
              dtype='object')
```

Lo siguiente es utilizar las funciones anteriormente definidas para obtener las existencias ovinas de Chubut del año 2020 divididas por departamento.

```
In [72]: df_separado_2020 = separar_en_provincias(existencias_2020, 2020)
resultado_existencias_2020_chubut =
filtrar_exitencias_chubut(df_separado_2020)
resultado_existencias_2020_chubut
```

Out[72]:	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2020	CHUBUT	CHUBUT	79860	1992008	669777	666432	576856	3984933
1	2020	CHUBUT	BIEDMA	3084	82806	29134	33727	25394	174145
2	2020	CHUBUT	CUSHAMEN	5385	124862	33570	28086	52545	244448
3	2020	CHUBUT	ESCALANTE	6310	98903	29646	37741	39590	212190
4	2020	CHUBUT	FLORENTINO AMEGHINO	7324	165006	57199	43199	45253	317981
5	2020	CHUBUT	FUTALEUFU	2044	65375	22621	12477	15472	117989
6	2020	CHUBUT	GAIMAN	5295	197372	27275	46137	42834	318913
7	2020	CHUBUT	GASTRE	4308	109233	42684	46336	33388	235949
8	2020	CHUBUT	LANGUIÃ'EO	3968	107651	42621	37285	40238	231763
9	2020	CHUBUT	MARTIRES	4051	94735	33775	25100	23559	181220
10	2020	CHUBUT	PASO DE INDIOS	5587	135352	46140	56784	30279	274142
11	2020	CHUBUT	RAWSON	2273	74096	18058	19267	22102	135796
12	2020	CHUBUT	RIO SENGUERR	15609	360905	146961	123560	112227	759262
13	2020	CHUBUT	SARMIENTO	3681	99173	27021	32939	20498	183312
14	2020	CHUBUT	TEHUELCHES	8071	197382	87268	86893	54493	434107
15	2020	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163716

Por último guardaremos estos datos en un archivo csv para trabajar con ellos más adelante.

```
In [73]: resultado_existencias_2020_chubut.to_csv('datos_producidos/senasa/re
index=False)
```

Juntando datasets producidos de existencias

Se van a cargar los csv anteriormente producidos, para juntarlos en un solo archivo

```
In [74]: anio_2008 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2008_chub
sep=",")
anio_2009 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2009_chub
sep=",")
anio_2010 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2010_chub
sep=",")
anio_2011 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2011_chub
sep=",")
```

```

anio_2012 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2012_chub
sep=",")
anio_2013 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2013_chub
sep=",")
anio_2014 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2014_chub
sep=",")
anio_2015 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2015_chub
sep=",")
anio_2016 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2016_chub
sep=",")
anio_2017 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2017_chub
sep=",")
anio_2018 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2018_chub
sep=",")
anio_2019 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2019_chub
sep=",")
anio_2020 =
pd.read_csv('datos_producidos/senasa/resultado_existencias_2020_chub
sep=",")
existencias_chubut = pd.DataFrame()
existencias_chubut = pd.concat([anio_2008, anio_2009, anio_2010,
anio_2011, anio_2012, anio_2013, anio_2014, anio_2015, anio_2016,
anio_2017, anio_2018, anio_2019, anio_2020], ignore_index=True)
existencias_chubut

```

Out[74]:

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
0	2008	CHUBUT	135075	2160564	885807	931734	766298	4879478	CHUBUT
1	2008	CHUBUT	2256	44272	18756	20551	14559	100394	RAWSON
2	2008	CHUBUT	5292	112497	32493	33649	28886	212817	GAIMAN
3	2008	CHUBUT	13875	164382	79863	63856	48809	370785	FLORENTINO AMEGHINO
4	2008	CHUBUT	9253	153127	68968	65652	42389	339389	PASO DE INDIOS
...	...	...	...	...	...	...	...	...	...

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
205	2020	CHUBUT	2273	74096	18058	19267	22102	135796	RAWSON
206	2020	CHUBUT	15609	360905	146961	123560	112227	759262	RIO SENGUERR
207	2020	CHUBUT	3681	99173	27021	32939	20498	183312	SARMIENTO
208	2020	CHUBUT	8071	197382	87268	86893	54493	434107	TEHUELCHES
209	2020	CHUBUT	2870	79157	25804	36901	18984	163716	TELSEN

210 rows × 9 columns

Se realizaran las siguientes correcciones al nuevo dataset:

- El departamento Biedma esta mal escrito en algunas filas, dice "Viedma"
- El departamento Languiño esta mal escrito en todas las filas
- El "departamento" CHUBUT en realidad es el total provincial, se lo va reemplazar por TOTAL.

```
In [75]: existencias_chubut.loc[existencias_chubut['departamento'] ==
'VIEDMA', 'departamento'] = 'BIEDMA'
existencias_chubut.loc[existencias_chubut['departamento'] ==
'LANGUIÑEO', 'departamento'] = 'LANGUIÑO'
existencias_chubut.loc[existencias_chubut['departamento'] ==
'CHUBUT', 'departamento'] = 'TOTAL'
```

Luego se ordenaran los datos por año, y luego por departamento.

```
In [76]: existencias_chubut = existencias_chubut.sort_values(by=
['anio', 'departamento'], ignore_index=True)
existencias_chubut
```

```
Out[76]:
```

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
0	2008	CHUBUT	7686	121561	58576	69633	33876	291332	BIEDMA
1	2008	CHUBUT	8180	149189	41669	51128	37589	287755	CUSHAMEN
2	2008	CHUBUT	7863	109225	35581	40786	41516	234971	ESCALANTE
3	2008	CHUBUT	13875	164382	79863	63856	48809	370785	FLORENTINO AMEGHINO
4	2008	CHUBUT	5639	90494	30119	24634	28237	179123	FUTALEUFU
...	...	...	...	...	...	...	...	...	...
205	2020	CHUBUT	15609	360905	146961	123560	112227	759262	RIO SENGUERR
206	2020	CHUBUT	3681	99173	27021	32939	20498	183312	SARMIENTO
207	2020	CHUBUT	8071	197382	87268	86893	54493	434107	TEHUELCHES
208	2020	CHUBUT	2870	79157	25804	36901	18984	163716	TELSEN
209	2020	CHUBUT	79860	1992008	669777	666432	576856	3984933	TOTAL

210 rows × 9 columns

Una vez realizados los cambios, se guardara todo en un archivo csv.

```
In [77]: existencias_chubut.to_csv('datos_producidos/existencias_chubut_2008_
index=False)
```

1.8 Dataset de lana FLA

Anteriormente se seleccionó este dataset para trabajar, ahora lo que necesitamos es conseguir la información de cada departamento de Chubut. Existe una encuesta ganadera que se realiza en la provincia, esta tiene una [sección](#) donde aparecen los resultados de las encuestas de los siguientes años:

- 2004-2005
- 2006-2007
- 2008-2009
- 2009-2010
- 2010-2011
- 2012
- 2013
- 2014

Lamentablemente solo se puede acceder a la encuesta del 2014. Suponiendo que el formato de la encuesta 2014 es el mismo que las anteriores realmente nos hubiese sido de mucha utilidad poder acceder a las demás, porque nos dice la cantidad de lana que produjo cada departamento de Chubut, que es exactamente lo que estamos buscando. Se invirtieron muchas horas tratando de conseguir acceso a las demás encuestas pero no se llegaron a buenos resultados.

Entonces lo que se va a hacer es generalizar la lana que produjo cada departamento en el año 2014 al resto de los años, los datos van a ser bastantes inexactos pero es lo único que tenemos.

Se tomo de la [encuesta](#) algunos datos y se armo el siguiente archivo csv

```
In [10]: import pandas as pd
import random
```

```
In [12]: encuesta_ganadera =
pd.read_csv('datos_sipas/encuesta_ganadera_2014.csv', sep=",")
encuesta_ganadera
```

```
Out[12]:
```

	DEPARTAMENTO	OVINOS	OVINOS_%	OVEJAS	CARNEROS	CAPONES	BORREGAS	CORDE
0	BIEDMA	158523	4,8	61195	3223	29688	29491	
1	CUSHAMEN	173238	5,3	77263	4198	18975	26512	
2	ESCALANTE	202476	6,2	80819	6921	35212	33497	
3	FLORENTINO AMEGHINO	347112	10,6	140928	8893	51459	56282	
4	FUTALEUFU	109363	3,3	51384	3436	11841	13783	
5	GAIMAN	199824	6,1	90039	4936	26376	27772	
6	GASTRE	153194	4,7	63706	3791	29293	21054	
7	LANGUIÑEO	221111	6,7	87640	6151	40929	38874	
8	MARTIRES	136857	4,2	62066	3788	21770	18961	

	DEPARTAMENTO	OVINOS	OVINOS_%	OVEJAS	CARNEROS	CAPONES	BORREGAS	CORDE
9	PASO DE INDIOS	207630	6,3	89998	5398	37250	30804	
10	RAWSON	51630	1,6	21913	1446	8839	7450	
11	RIO SENGUERR	560455	17,1	246882	15571	83406	90683	
12	SARMIENTO	182358	5,6	80123	5269	32012	27608	
13	TEHUELCHES	442496	13,5	176838	13819	90371	75419	
14	TELSEN	138288	4,2	59018	3495	22489	22886	
15	TOTAL	3284555	100	1389812	90335	539910	521076	

Cambiamos la coma por punto para que tomará el decimal.

In [5]:

```
lana_sucia_chubut =
pd.read_csv('datos_producidos/lana_sucia_chubut_2000-
2022_FLA.csv', sep=",")
lana_sucia_chubut
```

Out[5]:

	actividad_producto_nombre	indicador	unidad_de_medida	frecuencia_nombre	alcance_tipo	alcance_noi
0	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
1	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
2	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
3	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
4	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
5	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
6	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
7	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
8	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
9	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
10	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
11	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
12	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
13	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
14	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
15	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
16	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
17	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
18	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
19	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
20	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
21	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU
22	Lana sucia	Producción	tn	Anual	PROVINCIA	CHU

Miremos las columnas que tiene el dataset

```
In [22]: encuesta_ganadera.columns
```

```
Out[22]: Index(['DEPARTAMENTO', 'OVINOS', 'OVINOS_%', 'OVEJAS', 'CARNEROS', 'CAPONES',  
              'BORREGAS', 'CORDEROS_SEÑALADOS', 'OVINOS_ESQUILADOS',  
              'OVINOS_ESQUILADOS_%', 'LANA_SUCIA_PRODUCIDA', 'KG_LANA_CABEZA',  
              'FINURA', 'RINDE', 'LANA_SUCIA_%'],  
              dtype='object')
```

Se va a crear una lista de objetos, donde cada objeto va a tener el nombre del departamento y el porcentaje de lana que ese departamento produjo en el año 2014.

```
In [13]: departamentos = []  
for i in encuesta_ganadera.index:  
    departamentos.append({'nombre':  
encuesta_ganadera["DEPARTAMENTO"][i],  
                          'lana_%':  
encuesta_ganadera["LANA_SUCIA_%"][i],  
                          'finura': encuesta_ganadera["FINURA"]  
[i],  
                          'rinde': encuesta_ganadera["RINDE"][i]})  
departamentos
```

```
Out[13]: [{'nombre': 'BIEDMA', 'lana_%': 4.67, 'finura': 19.22, 'rinde': 58.86},  
          {'nombre': 'CUSHAMEN', 'lana_%': 5.25, 'finura': 19.74, 'rinde': 63.35},  
          {'nombre': 'ESCALANTE', 'lana_%': 6.39, 'finura': 19.83, 'rinde': 50.91},  
          {'nombre': 'FLORENTINO AMEGHINO',  
            'lana_%': 8.61,  
            'finura': 19.53,  
            'rinde': 60.37},  
          {'nombre': 'FUTALEUFU', 'lana_%': 3.28, 'finura': 20.27, 'rinde': 63.13},  
          {'nombre': 'GAIMAN', 'lana_%': 5.79, 'finura': 20.39, 'rinde': 55.06},  
          {'nombre': 'GASTRE', 'lana_%': 4.34, 'finura': 19.8, 'rinde': 54.38},  
          {'nombre': 'LANGUIÑEO', 'lana_%': 7.11, 'finura': 19.86, 'rinde': 60.94},  
          {'nombre': 'MARTIRES', 'lana_%': 4.53, 'finura': 20.1, 'rinde': 49.07},  
          {'nombre': 'PASO DE INDIOS', 'lana_%': 6.44, 'finura': 19.92, 'rinde': 54.1  
9},  
          {'nombre': 'RAWSON', 'lana_%': 1.52, 'finura': 19.78, 'rinde': 61.22},  
          {'nombre': 'RIO SENGUERR', 'lana_%': 17.12, 'finura': 20.1, 'rinde': 59.01},  
          {'nombre': 'SARMIENTO', 'lana_%': 6.31, 'finura': 19.92, 'rinde': 48.21},  
          {'nombre': 'TEHUELCHES', 'lana_%': 14.1, 'finura': 19.41, 'rinde': 61.21},  
          {'nombre': 'TELSEN', 'lana_%': 4.55, 'finura': 19.79, 'rinde': 54.01},  
          {'nombre': 'TOTAL', 'lana_%': 100.0, 'finura': 19.83, 'rinde': 57.35}]
```

Se va a recorrer el dataset de FLA que solo tiene los totales anuales de lana producidos, como se va a trabajar en kilos a las toneladas se las multiplica por mil para hacer la transformación. Luego se va a recorrer la lista creada anteriormente y se va a calcular el porcentaje de lana que produjo cada departamento a partir del total de lana producido.

In [14]:

```
datos=[]
for i in lana_sucia_chubut.index:
    fecha = lana_sucia_chubut['indice_tiempo'][i]
    total_lana_kg = lana_sucia_chubut['total'][i]*1000
    for dpto in departamentos:
        kg_lana = round((dpto['lana_%']*total_lana_kg)/100,2)
        finura = round(random.normalvariate(dpto['finura'],
0.95),2)
        rinde = round(random.normalvariate(dpto['rinde'], 0.95),2)
        datos.append([fecha, dpto['nombre'],kg_lana,finura,rinde])
datos
```

Out[14]:

```
[['2000-01-01', 'BIEDMA', 793900.0, 20.39, 59.21],
 ['2000-01-01', 'CUSHAMEN', 892500.0, 19.1, 61.65],
 ['2000-01-01', 'ESCALANTE', 1086300.0, 21.9, 49.99],
 ['2000-01-01', 'FLORENTINO AMEGHINO', 1463700.0, 19.61, 60.32],
 ['2000-01-01', 'FUTALEUFU', 557600.0, 20.57, 63.37],
 ['2000-01-01', 'GAIMAN', 984300.0, 20.76, 54.52],
 ['2000-01-01', 'GASTRE', 737800.0, 19.27, 55.73],
 ['2000-01-01', 'LANGUIÑO', 1208700.0, 19.0, 60.69],
 ['2000-01-01', 'MARTIRES', 770100.0, 20.64, 47.37],
 ['2000-01-01', 'PASO DE INDIOS', 1094800.0, 20.11, 54.48],
 ['2000-01-01', 'RAWSON', 258400.0, 17.66, 61.53],
 ['2000-01-01', 'RIO SENGUERR', 2910400.0, 20.89, 58.82],
 ['2000-01-01', 'SARMIENTO', 1072700.0, 19.47, 46.59],
 ['2000-01-01', 'TEHUELCHES', 2397000.0, 18.32, 60.39],
 ['2000-01-01', 'TELSEN', 773500.0, 18.22, 54.67],
 ['2000-01-01', 'TOTAL', 17000000.0, 21.26, 57.38],
 ['2001-01-01', 'BIEDMA', 793900.0, 19.01, 59.03],
 ['2001-01-01', 'CUSHAMEN', 892500.0, 19.32, 63.65],
 ['2001-01-01', 'ESCALANTE', 1086300.0, 20.38, 52.18],
 ['2001-01-01', 'FLORENTINO AMEGHINO', 1463700.0, 18.18, 60.91],
 ['2001-01-01', 'FUTALEUFU', 557600.0, 21.43, 63.23],
 ['2001-01-01', 'GAIMAN', 984300.0, 21.07, 54.4],
 ['2001-01-01', 'GASTRE', 737800.0, 20.04, 54.42],
 ['2001-01-01', 'LANGUIÑO', 1208700.0, 18.15, 61.33],
 ['2001-01-01', 'MARTIRES', 770100.0, 20.72, 49.65],
 ['2001-01-01', 'PASO DE INDIOS', 1094800.0, 19.87, 53.35],
 ['2001-01-01', 'RAWSON', 258400.0, 18.15, 60.74],
 ['2001-01-01', 'RIO SENGUERR', 2910400.0, 19.57, 59.72],
 ['2001-01-01', 'SARMIENTO', 1072700.0, 20.64, 47.92],
 ['2001-01-01', 'TEHUELCHES', 2397000.0, 17.89, 62.33],
 ['2001-01-01', 'TELSEN', 773500.0, 19.6, 53.86],
 ['2001-01-01', 'TOTAL', 17000000.0, 22.01, 56.04],
 ['2002-01-01', 'BIEDMA', 789230.0, 20.53, 57.17],
 ['2002-01-01', 'CUSHAMEN', 887250.0, 20.58, 62.79],
 ['2002-01-01', 'ESCALANTE', 1079910.0, 19.6, 50.15],
 ['2002-01-01', 'FLORENTINO AMEGHINO', 1455090.0, 19.1, 62.38],
 ['2002-01-01', 'FUTALEUFU', 554320.0, 20.85, 63.49],
 ['2002-01-01', 'GAIMAN', 978510.0, 21.48, 53.42],
 ['2002-01-01', 'GASTRE', 733460.0, 17.98, 54.65],
 ['2002-01-01', 'LANGUIÑO', 1201590.0, 20.2, 60.13],
```

['2002-01-01', 'MARTIRES', 765570.0, 19.48, 48.53],
 ['2002-01-01', 'PASO DE INDIOS', 1088360.0, 20.11, 53.7],
 ['2002-01-01', 'RAWSON', 256880.0, 19.99, 60.19],
 ['2002-01-01', 'RIO SENGUERR', 2893280.0, 20.39, 58.27],
 ['2002-01-01', 'SARMIENTO', 1066390.0, 19.04, 49.36],
 ['2002-01-01', 'TEHUELCHES', 2382900.0, 18.11, 61.26],
 ['2002-01-01', 'TELSEN', 768950.0, 21.55, 53.43],
 ['2002-01-01', 'TOTAL', 16900000.0, 20.41, 57.96],
 ['2003-01-01', 'BIEDMA', 994710.0, 20.57, 58.14],
 ['2003-01-01', 'CUSHAMEN', 1118250.0, 20.41, 63.57],
 ['2003-01-01', 'ESCALANTE', 1361070.0, 19.93, 51.24],
 ['2003-01-01', 'FLORENTINO AMEGHINO', 1833930.0, 19.18, 60.0],
 ['2003-01-01', 'FUTALEUFU', 698640.0, 20.62, 64.2],
 ['2003-01-01', 'GAIMAN', 1233270.0, 20.01, 54.27],
 ['2003-01-01', 'GASTRE', 924420.0, 21.21, 55.49],
 ['2003-01-01', 'LANGUINEO', 1514430.0, 21.39, 60.41],
 ['2003-01-01', 'MARTIRES', 964890.0, 19.38, 47.98],
 ['2003-01-01', 'PASO DE INDIOS', 1371720.0, 20.79, 54.07],
 ['2003-01-01', 'RAWSON', 323760.0, 20.9, 59.9],
 ['2003-01-01', 'RIO SENGUERR', 3646560.0, 20.54, 56.98],
 ['2003-01-01', 'SARMIENTO', 1344030.0, 19.29, 51.39],
 ['2003-01-01', 'TEHUELCHES', 3003300.0, 20.43, 61.76],
 ['2003-01-01', 'TELSEN', 969150.0, 20.79, 55.24],
 ['2003-01-01', 'TOTAL', 21300000.0, 21.02, 57.72],
 ['2004-01-01', 'BIEDMA', 1022730.0, 20.68, 57.63],
 ['2004-01-01', 'CUSHAMEN', 1149750.0, 20.95, 63.45],
 ['2004-01-01', 'ESCALANTE', 1399410.0, 19.62, 50.91],
 ['2004-01-01', 'FLORENTINO AMEGHINO', 1885590.0, 17.83, 60.21],
 ['2004-01-01', 'FUTALEUFU', 718320.0, 20.47, 61.87],
 ['2004-01-01', 'GAIMAN', 1268010.0, 21.94, 54.93],
 ['2004-01-01', 'GASTRE', 950460.0, 19.12, 53.18],
 ['2004-01-01', 'LANGUINEO', 1557090.0, 18.16, 61.33],
 ['2004-01-01', 'MARTIRES', 992070.0, 19.49, 48.73],
 ['2004-01-01', 'PASO DE INDIOS', 1410360.0, 18.29, 53.25],
 ['2004-01-01', 'RAWSON', 332880.0, 19.02, 60.91],
 ['2004-01-01', 'RIO SENGUERR', 3749280.0, 21.08, 59.42],
 ['2004-01-01', 'SARMIENTO', 1381890.0, 18.74, 47.59],
 ['2004-01-01', 'TEHUELCHES', 3087900.0, 19.6, 61.51],
 ['2004-01-01', 'TELSEN', 996450.0, 18.31, 53.55],
 ['2004-01-01', 'TOTAL', 21900000.0, 19.91, 57.92],
 ['2005-01-01', 'BIEDMA', 1083440.0, 18.5, 59.49],
 ['2005-01-01', 'CUSHAMEN', 1218000.0, 19.32, 64.71],
 ['2005-01-01', 'ESCALANTE', 1482480.0, 21.08, 49.98],
 ['2005-01-01', 'FLORENTINO AMEGHINO', 1997520.0, 20.73, 60.14],
 ['2005-01-01', 'FUTALEUFU', 760960.0, 21.45, 64.4],
 ['2005-01-01', 'GAIMAN', 1343280.0, 20.46, 57.39],
 ['2005-01-01', 'GASTRE', 1006880.0, 20.72, 54.14],
 ['2005-01-01', 'LANGUINEO', 1649520.0, 20.32, 60.05],
 ['2005-01-01', 'MARTIRES', 1050960.0, 21.44, 49.43],
 ['2005-01-01', 'PASO DE INDIOS', 1494080.0, 19.31, 54.22],
 ['2005-01-01', 'RAWSON', 352640.0, 18.92, 60.5],
 ['2005-01-01', 'RIO SENGUERR', 3971840.0, 21.11, 58.01],
 ['2005-01-01', 'SARMIENTO', 1463920.0, 20.13, 48.29],
 ['2005-01-01', 'TEHUELCHES', 3271200.0, 20.67, 61.67],
 ['2005-01-01', 'TELSEN', 1055600.0, 19.26, 53.64],
 ['2005-01-01', 'TOTAL', 23200000.0, 19.55, 56.43],

['2006-01-01', 'BIEDMA', 1027400.0, 17.51, 59.67],
 ['2006-01-01', 'CUSHAMEN', 1155000.0, 19.83, 63.14],
 ['2006-01-01', 'ESCALANTE', 1405800.0, 21.21, 52.08],
 ['2006-01-01', 'FLORENTINO AMEGHINO', 1894200.0, 20.67, 59.16],
 ['2006-01-01', 'FUTALEUFU', 721600.0, 19.92, 62.66],
 ['2006-01-01', 'GAIMAN', 1273800.0, 19.98, 53.38],
 ['2006-01-01', 'GASTRE', 954800.0, 18.63, 57.14],
 ['2006-01-01', 'LANGUIÑEO', 1564200.0, 19.02, 61.92],
 ['2006-01-01', 'MARTIRES', 996600.0, 18.46, 47.69],
 ['2006-01-01', 'PASO DE INDIOS', 1416800.0, 19.36, 51.94],
 ['2006-01-01', 'RAWSON', 334400.0, 19.64, 62.64],
 ['2006-01-01', 'RIO SENGUERR', 3766400.0, 21.78, 58.87],
 ['2006-01-01', 'SARMIENTO', 1388200.0, 20.01, 49.17],
 ['2006-01-01', 'TEHUELCHES', 3102000.0, 19.37, 60.72],
 ['2006-01-01', 'TELSEN', 1001000.0, 21.96, 55.02],
 ['2006-01-01', 'TOTAL', 22000000.0, 20.54, 58.07],
 ['2007-01-01', 'BIEDMA', 1120800.0, 18.95, 57.14],
 ['2007-01-01', 'CUSHAMEN', 1260000.0, 20.49, 62.35],
 ['2007-01-01', 'ESCALANTE', 1533600.0, 20.03, 49.84],
 ['2007-01-01', 'FLORENTINO AMEGHINO', 2066400.0, 19.95, 61.02],
 ['2007-01-01', 'FUTALEUFU', 787200.0, 20.82, 62.99],
 ['2007-01-01', 'GAIMAN', 1389600.0, 20.71, 55.6],
 ['2007-01-01', 'GASTRE', 1041600.0, 19.23, 54.29],
 ['2007-01-01', 'LANGUIÑEO', 1706400.0, 19.28, 60.04],
 ['2007-01-01', 'MARTIRES', 1087200.0, 19.77, 50.2],
 ['2007-01-01', 'PASO DE INDIOS', 1545600.0, 18.75, 53.66],
 ['2007-01-01', 'RAWSON', 364800.0, 19.67, 61.9],
 ['2007-01-01', 'RIO SENGUERR', 4108800.0, 20.93, 57.58],
 ['2007-01-01', 'SARMIENTO', 1514400.0, 19.62, 48.85],
 ['2007-01-01', 'TEHUELCHES', 3384000.0, 17.57, 60.12],
 ['2007-01-01', 'TELSEN', 1092000.0, 21.11, 55.35],
 ['2007-01-01', 'TOTAL', 24000000.0, 21.1, 56.56],
 ['2008-01-01', 'BIEDMA', 1008720.0, 18.88, 58.4],
 ['2008-01-01', 'CUSHAMEN', 1134000.0, 17.95, 62.96],
 ['2008-01-01', 'ESCALANTE', 1380240.0, 19.73, 52.29],
 ['2008-01-01', 'FLORENTINO AMEGHINO', 1859760.0, 19.55, 60.77],
 ['2008-01-01', 'FUTALEUFU', 708480.0, 21.22, 63.71],
 ['2008-01-01', 'GAIMAN', 1250640.0, 20.76, 53.69],
 ['2008-01-01', 'GASTRE', 937440.0, 20.29, 53.49],
 ['2008-01-01', 'LANGUIÑEO', 1535760.0, 20.94, 59.61],
 ['2008-01-01', 'MARTIRES', 978480.0, 21.09, 48.73],
 ['2008-01-01', 'PASO DE INDIOS', 1391040.0, 21.08, 53.49],
 ['2008-01-01', 'RAWSON', 328320.0, 19.79, 60.06],
 ['2008-01-01', 'RIO SENGUERR', 3697920.0, 21.34, 59.04],
 ['2008-01-01', 'SARMIENTO', 1362960.0, 20.96, 47.57],
 ['2008-01-01', 'TEHUELCHES', 3045600.0, 19.15, 61.75],
 ['2008-01-01', 'TELSEN', 982800.0, 21.55, 53.47],
 ['2008-01-01', 'TOTAL', 21600000.0, 19.82, 56.94],
 ['2009-01-01', 'BIEDMA', 799504.0, 19.52, 59.8],
 ['2009-01-01', 'CUSHAMEN', 898800.0, 19.83, 63.6],
 ['2009-01-01', 'ESCALANTE', 1093968.0, 20.77, 50.61],
 ['2009-01-01', 'FLORENTINO AMEGHINO', 1474032.0, 19.47, 59.79],
 ['2009-01-01', 'FUTALEUFU', 561536.0, 21.2, 61.95],
 ['2009-01-01', 'GAIMAN', 991248.0, 21.1, 53.93],
 ['2009-01-01', 'GASTRE', 743008.0, 20.89, 53.84],
 ['2009-01-01', 'LANGUIÑEO', 1217232.0, 20.68, 59.6],

['2009-01-01', 'MARTIRES', 775536.0, 20.41, 50.56],
 ['2009-01-01', 'PASO DE INDIOS', 1102528.0, 19.47, 54.51],
 ['2009-01-01', 'RAWSON', 260224.0, 19.28, 61.77],
 ['2009-01-01', 'RIO SENGUERR', 2930944.0, 19.62, 58.44],
 ['2009-01-01', 'SARMIENTO', 1080272.0, 20.96, 48.4],
 ['2009-01-01', 'TEHUELCHES', 2413920.0, 18.62, 61.43],
 ['2009-01-01', 'TELSEN', 778960.0, 19.63, 55.69],
 ['2009-01-01', 'TOTAL', 17120000.0, 20.94, 56.5],
 ['2010-01-01', 'BIEDMA', 799504.0, 18.02, 58.37],
 ['2010-01-01', 'CUSHAMEN', 898800.0, 20.36, 64.07],
 ['2010-01-01', 'ESCALANTE', 1093968.0, 19.19, 51.72],
 ['2010-01-01', 'FLORENTINO AMEGHINO', 1474032.0, 19.93, 62.41],
 ['2010-01-01', 'FUTALEUFU', 561536.0, 20.37, 63.09],
 ['2010-01-01', 'GAIMAN', 991248.0, 19.8, 53.7],
 ['2010-01-01', 'GASTRE', 743008.0, 19.17, 54.71],
 ['2010-01-01', 'LANGUINEO', 1217232.0, 21.14, 60.08],
 ['2010-01-01', 'MARTIRES', 775536.0, 20.11, 49.0],
 ['2010-01-01', 'PASO DE INDIOS', 1102528.0, 21.62, 53.89],
 ['2010-01-01', 'RAWSON', 260224.0, 18.0, 59.22],
 ['2010-01-01', 'RIO SENGUERR', 2930944.0, 21.13, 59.06],
 ['2010-01-01', 'SARMIENTO', 1080272.0, 19.06, 48.94],
 ['2010-01-01', 'TEHUELCHES', 2413920.0, 20.43, 61.28],
 ['2010-01-01', 'TELSEN', 778960.0, 20.39, 54.44],
 ['2010-01-01', 'TOTAL', 17120000.0, 19.28, 56.3],
 ['2011-01-01', 'BIEDMA', 788763.0, 19.17, 57.54],
 ['2011-01-01', 'CUSHAMEN', 886725.0, 19.03, 62.96],
 ['2011-01-01', 'ESCALANTE', 1079271.0, 20.96, 50.2],
 ['2011-01-01', 'FLORENTINO AMEGHINO', 1454229.0, 19.6, 58.86],
 ['2011-01-01', 'FUTALEUFU', 553992.0, 19.61, 65.45],
 ['2011-01-01', 'GAIMAN', 977931.0, 21.82, 55.88],
 ['2011-01-01', 'GASTRE', 733026.0, 20.18, 55.64],
 ['2011-01-01', 'LANGUINEO', 1200879.0, 19.5, 60.92],
 ['2011-01-01', 'MARTIRES', 765117.0, 19.78, 49.11],
 ['2011-01-01', 'PASO DE INDIOS', 1087716.0, 20.3, 56.36],
 ['2011-01-01', 'RAWSON', 256728.0, 20.53, 62.42],
 ['2011-01-01', 'RIO SENGUERR', 2891568.0, 20.47, 58.34],
 ['2011-01-01', 'SARMIENTO', 1065759.0, 19.58, 48.13],
 ['2011-01-01', 'TEHUELCHES', 2381490.0, 17.85, 59.92],
 ['2011-01-01', 'TELSEN', 768495.0, 18.66, 52.82],
 ['2011-01-01', 'TOTAL', 16890000.0, 19.89, 57.27],
 ['2012-01-01', 'BIEDMA', 649130.0, 19.8, 58.06],
 ['2012-01-01', 'CUSHAMEN', 729750.0, 18.61, 62.22],
 ['2012-01-01', 'ESCALANTE', 888210.0, 19.08, 50.87],
 ['2012-01-01', 'FLORENTINO AMEGHINO', 1196790.0, 19.07, 59.01],
 ['2012-01-01', 'FUTALEUFU', 455920.0, 18.14, 63.07],
 ['2012-01-01', 'GAIMAN', 804810.0, 18.47, 56.31],
 ['2012-01-01', 'GASTRE', 603260.0, 16.94, 54.86],
 ['2012-01-01', 'LANGUINEO', 988290.0, 20.15, 60.69],
 ['2012-01-01', 'MARTIRES', 629670.0, 20.53, 48.0],
 ['2012-01-01', 'PASO DE INDIOS', 895160.0, 19.07, 53.14],
 ['2012-01-01', 'RAWSON', 211280.0, 20.52, 61.69],
 ['2012-01-01', 'RIO SENGUERR', 2379680.0, 20.69, 57.48],
 ['2012-01-01', 'SARMIENTO', 877090.0, 18.74, 48.82],
 ['2012-01-01', 'TEHUELCHES', 1959900.0, 18.02, 60.11],
 ['2012-01-01', 'TELSEN', 632450.0, 20.94, 53.87],
 ['2012-01-01', 'TOTAL', 13900000.0, 18.86, 55.45],

['2013-01-01', 'BIEDMA', 620176.0, 19.6, 56.68],
 ['2013-01-01', 'CUSHAMEN', 697200.0, 21.0, 64.48],
 ['2013-01-01', 'ESCALANTE', 848592.0, 19.87, 50.74],
 ['2013-01-01', 'FLORENTINO AMEGHINO', 1143408.0, 19.81, 58.44],
 ['2013-01-01', 'FUTALEUFU', 435584.0, 20.16, 62.56],
 ['2013-01-01', 'GAIMAN', 768912.0, 18.89, 55.74],
 ['2013-01-01', 'GASTRE', 576352.0, 19.05, 53.33],
 ['2013-01-01', 'LANGUIÑEO', 944208.0, 19.94, 61.19],
 ['2013-01-01', 'MARTIRES', 601584.0, 20.53, 47.1],
 ['2013-01-01', 'PASO DE INDIOS', 855232.0, 19.79, 54.5],
 ['2013-01-01', 'RAWSON', 201856.0, 20.95, 62.38],
 ['2013-01-01', 'RIO SENGUERR', 2273536.0, 19.61, 58.04],
 ['2013-01-01', 'SARMIENTO', 837968.0, 19.29, 47.51],
 ['2013-01-01', 'TEHUELCHES', 1872480.0, 18.72, 61.23],
 ['2013-01-01', 'TELSEN', 604240.0, 19.97, 53.27],
 ['2013-01-01', 'TOTAL', 13280000.0, 18.84, 57.42],
 ['2014-01-01', 'BIEDMA', 649130.0, 20.1, 61.19],
 ['2014-01-01', 'CUSHAMEN', 729750.0, 20.48, 63.3],
 ['2014-01-01', 'ESCALANTE', 888210.0, 19.33, 51.4],
 ['2014-01-01', 'FLORENTINO AMEGHINO', 1196790.0, 20.45, 60.73],
 ['2014-01-01', 'FUTALEUFU', 455920.0, 21.55, 62.93],
 ['2014-01-01', 'GAIMAN', 804810.0, 21.14, 55.43],
 ['2014-01-01', 'GASTRE', 603260.0, 19.89, 54.03],
 ['2014-01-01', 'LANGUIÑEO', 988290.0, 19.38, 62.03],
 ['2014-01-01', 'MARTIRES', 629670.0, 18.83, 48.45],
 ['2014-01-01', 'PASO DE INDIOS', 895160.0, 20.9, 54.79],
 ['2014-01-01', 'RAWSON', 211280.0, 19.55, 62.99],
 ['2014-01-01', 'RIO SENGUERR', 2379680.0, 19.16, 59.64],
 ['2014-01-01', 'SARMIENTO', 877090.0, 22.22, 47.61],
 ['2014-01-01', 'TEHUELCHES', 1959900.0, 17.74, 61.77],
 ['2014-01-01', 'TELSEN', 632450.0, 19.77, 54.03],
 ['2014-01-01', 'TOTAL', 13900000.0, 22.2, 57.43],
 ['2015-01-01', 'BIEDMA', 672480.0, 18.58, 58.18],
 ['2015-01-01', 'CUSHAMEN', 756000.0, 18.33, 62.18],
 ['2015-01-01', 'ESCALANTE', 920160.0, 19.76, 50.93],
 ['2015-01-01', 'FLORENTINO AMEGHINO', 1239840.0, 20.27, 60.14],
 ['2015-01-01', 'FUTALEUFU', 472320.0, 20.94, 63.1],
 ['2015-01-01', 'GAIMAN', 833760.0, 20.58, 56.05],
 ['2015-01-01', 'GASTRE', 624960.0, 20.46, 52.71],
 ['2015-01-01', 'LANGUIÑEO', 1023840.0, 22.48, 61.62],
 ['2015-01-01', 'MARTIRES', 652320.0, 20.79, 48.45],
 ['2015-01-01', 'PASO DE INDIOS', 927360.0, 19.79, 53.52],
 ['2015-01-01', 'RAWSON', 218880.0, 18.35, 60.87],
 ['2015-01-01', 'RIO SENGUERR', 2465280.0, 17.93, 58.92],
 ['2015-01-01', 'SARMIENTO', 908640.0, 19.09, 47.71],
 ['2015-01-01', 'TEHUELCHES', 2030400.0, 18.48, 62.18],
 ['2015-01-01', 'TELSEN', 655200.0, 17.81, 53.48],
 ['2015-01-01', 'TOTAL', 14400000.0, 18.87, 57.29],
 ['2016-01-01', 'BIEDMA', 653800.0, 17.99, 59.93],
 ['2016-01-01', 'CUSHAMEN', 735000.0, 19.86, 63.67],
 ['2016-01-01', 'ESCALANTE', 894600.0, 20.18, 50.32],
 ['2016-01-01', 'FLORENTINO AMEGHINO', 1205400.0, 18.75, 60.54],
 ['2016-01-01', 'FUTALEUFU', 459200.0, 18.75, 64.05],
 ['2016-01-01', 'GAIMAN', 810600.0, 21.47, 53.63],
 ['2016-01-01', 'GASTRE', 607600.0, 20.84, 54.51],
 ['2016-01-01', 'LANGUIÑEO', 995400.0, 21.27, 60.55],

['2016-01-01', 'MARTIRES', 634200.0, 21.24, 48.68],
 ['2016-01-01', 'PASO DE INDIOS', 901600.0, 19.7, 55.38],
 ['2016-01-01', 'RAWSON', 212800.0, 20.16, 60.71],
 ['2016-01-01', 'RIO SENGUERR', 2396800.0, 20.59, 58.51],
 ['2016-01-01', 'SARMIENTO', 883400.0, 19.62, 48.88],
 ['2016-01-01', 'TEHUELCHES', 1974000.0, 17.12, 60.33],
 ['2016-01-01', 'TELSEN', 637000.0, 20.2, 53.55],
 ['2016-01-01', 'TOTAL', 14000000.0, 20.2, 57.48],
 ['2017-01-01', 'BIEDMA', 653800.0, 19.73, 58.95],
 ['2017-01-01', 'CUSHAMEN', 735000.0, 19.67, 63.2],
 ['2017-01-01', 'ESCALANTE', 894600.0, 19.5, 51.98],
 ['2017-01-01', 'FLORENTINO AMEGHINO', 1205400.0, 19.84, 59.5],
 ['2017-01-01', 'FUTALEUFU', 459200.0, 19.73, 63.04],
 ['2017-01-01', 'GAIMAN', 810600.0, 20.85, 55.26],
 ['2017-01-01', 'GASTRE', 607600.0, 18.81, 54.94],
 ['2017-01-01', 'LANGUINEO', 995400.0, 19.57, 59.32],
 ['2017-01-01', 'MARTIRES', 634200.0, 21.41, 50.2],
 ['2017-01-01', 'PASO DE INDIOS', 901600.0, 19.33, 54.63],
 ['2017-01-01', 'RAWSON', 212800.0, 21.05, 61.49],
 ['2017-01-01', 'RIO SENGUERR', 2396800.0, 21.1, 60.17],
 ['2017-01-01', 'SARMIENTO', 883400.0, 19.5, 48.12],
 ['2017-01-01', 'TEHUELCHES', 1974000.0, 17.15, 61.48],
 ['2017-01-01', 'TELSEN', 637000.0, 19.11, 56.01],
 ['2017-01-01', 'TOTAL', 14000000.0, 21.39, 58.88],
 ['2018-01-01', 'BIEDMA', 653800.0, 18.86, 58.89],
 ['2018-01-01', 'CUSHAMEN', 735000.0, 18.59, 62.6],
 ['2018-01-01', 'ESCALANTE', 894600.0, 19.66, 49.72],
 ['2018-01-01', 'FLORENTINO AMEGHINO', 1205400.0, 19.04, 58.93],
 ['2018-01-01', 'FUTALEUFU', 459200.0, 21.2, 64.01],
 ['2018-01-01', 'GAIMAN', 810600.0, 20.43, 54.97],
 ['2018-01-01', 'GASTRE', 607600.0, 18.89, 53.7],
 ['2018-01-01', 'LANGUINEO', 995400.0, 19.47, 59.47],
 ['2018-01-01', 'MARTIRES', 634200.0, 19.49, 51.3],
 ['2018-01-01', 'PASO DE INDIOS', 901600.0, 20.11, 57.02],
 ['2018-01-01', 'RAWSON', 212800.0, 19.84, 62.17],
 ['2018-01-01', 'RIO SENGUERR', 2396800.0, 20.45, 57.75],
 ['2018-01-01', 'SARMIENTO', 883400.0, 20.03, 48.8],
 ['2018-01-01', 'TEHUELCHES', 1974000.0, 17.83, 61.71],
 ['2018-01-01', 'TELSEN', 637000.0, 20.38, 54.64],
 ['2018-01-01', 'TOTAL', 14000000.0, 19.78, 58.82],
 ['2019-01-01', 'BIEDMA', 629189.1, 18.95, 57.81],
 ['2019-01-01', 'CUSHAMEN', 707332.5, 19.99, 64.07],
 ['2019-01-01', 'ESCALANTE', 860924.7, 18.99, 49.86],
 ['2019-01-01', 'FLORENTINO AMEGHINO', 1160025.3, 18.92, 61.7],
 ['2019-01-01', 'FUTALEUFU', 441914.4, 19.67, 62.75],
 ['2019-01-01', 'GAIMAN', 780086.7, 21.01, 55.69],
 ['2019-01-01', 'GASTRE', 584728.2, 17.95, 54.45],
 ['2019-01-01', 'LANGUINEO', 957930.3, 20.58, 59.18],
 ['2019-01-01', 'MARTIRES', 610326.9, 18.65, 49.59],
 ['2019-01-01', 'PASO DE INDIOS', 867661.2, 19.99, 55.58],
 ['2019-01-01', 'RAWSON', 204789.6, 20.42, 61.52],
 ['2019-01-01', 'RIO SENGUERR', 2306577.6, 18.8, 59.28],
 ['2019-01-01', 'SARMIENTO', 850146.3, 18.15, 49.05],
 ['2019-01-01', 'TEHUELCHES', 1899693.0, 20.09, 61.13],
 ['2019-01-01', 'TELSEN', 613021.5, 16.88, 53.83],
 ['2019-01-01', 'TOTAL', 13473000.0, 19.16, 56.04],

```
[ '2020-01-01', 'BIEDMA', 630450.0, 19.1, 58.71],
[ '2020-01-01', 'CUSHAMEN', 708750.0, 19.5, 64.01],
[ '2020-01-01', 'ESCALANTE', 862650.0, 21.6, 51.33],
[ '2020-01-01', 'FLORENTINO AMEGHINO', 1162350.0, 19.94, 58.79],
[ '2020-01-01', 'FUTALEUFU', 442800.0, 19.45, 63.64],
[ '2020-01-01', 'GAIMAN', 781650.0, 20.88, 54.45],
[ '2020-01-01', 'GASTRE', 585900.0, 21.12, 54.46],
[ '2020-01-01', 'LANGUIÑEO', 959850.0, 20.53, 61.55],
[ '2020-01-01', 'MARTIRES', 611550.0, 20.72, 49.83],
[ '2020-01-01', 'PASO DE INDIOS', 869400.0, 19.4, 54.25],
[ '2020-01-01', 'RAWSON', 205200.0, 19.57, 61.66],
[ '2020-01-01', 'RIO SENGUERR', 2311200.0, 20.28, 57.7],
[ '2020-01-01', 'SARMIENTO', 851850.0, 19.37, 48.32],
[ '2020-01-01', 'TEHUELCHES', 1903500.0, 20.02, 61.85],
[ '2020-01-01', 'TELSEN', 614250.0, 21.55, 52.96],
[ '2020-01-01', 'TOTAL', 13500000.0, 20.36, 56.57],
[ '2021-01-01', 'BIEDMA', 622931.3, 19.65, 59.0],
[ '2021-01-01', 'CUSHAMEN', 700297.5, 19.18, 64.27],
[ '2021-01-01', 'ESCALANTE', 852362.1, 19.84, 50.54],
[ '2021-01-01', 'FLORENTINO AMEGHINO', 1148487.9, 19.7, 60.16],
[ '2021-01-01', 'FUTALEUFU', 437519.2, 18.8, 64.04],
[ '2021-01-01', 'GAIMAN', 772328.1, 20.84, 56.29],
[ '2021-01-01', 'GASTRE', 578912.6, 19.64, 54.25],
[ '2021-01-01', 'LANGUIÑEO', 948402.9, 18.55, 58.17],
[ '2021-01-01', 'MARTIRES', 604256.7, 20.34, 48.5],
[ '2021-01-01', 'PASO DE INDIOS', 859031.6, 19.27, 53.44],
[ '2021-01-01', 'RAWSON', 202752.8, 19.34, 60.31],
[ '2021-01-01', 'RIO SENGUERR', 2283636.8, 20.83, 60.86],
[ '2021-01-01', 'SARMIENTO', 841690.9, 18.03, 47.41],
[ '2021-01-01', 'TEHUELCHES', 1880799.0, 17.98, 61.18],
[ '2021-01-01', 'TELSEN', 606924.5, 20.04, 55.92],
[ '2021-01-01', 'TOTAL', 13339000.0, 20.52, 58.11],
[ '2022-01-01', 'BIEDMA', 539447.58, 18.15, 60.3],
[ '2022-01-01', 'CUSHAMEN', 606445.35, 19.72, 64.17],
[ '2022-01-01', 'ESCALANTE', 738130.63, 21.4, 49.84],
[ '2022-01-01', 'FLORENTINO AMEGHINO', 994570.37, 18.62, 60.68],
[ '2022-01-01', 'FUTALEUFU', 378883.95, 20.89, 64.02],
[ '2022-01-01', 'GAIMAN', 668822.59, 18.96, 54.96],
[ '2022-01-01', 'GASTRE', 501328.16, 20.32, 54.16],
[ '2022-01-01', 'LANGUIÑEO', 821300.27, 22.22, 60.73],
[ '2022-01-01', 'MARTIRES', 523275.7, 19.72, 48.2],
[ '2022-01-01', 'PASO DE INDIOS', 743906.3, 20.87, 55.13],
[ '2022-01-01', 'RAWSON', 175580.37, 18.59, 60.95],
[ '2022-01-01', 'RIO SENGUERR', 1977589.41, 19.72, 58.07],
[ '2022-01-01', 'SARMIENTO', 728889.55, 20.94, 49.2],
[ '2022-01-01', 'TEHUELCHES', 1628738.94, 18.47, 60.56],
[ '2022-01-01', 'TELSEN', 525585.97, 19.11, 55.35],
[ '2022-01-01', 'TOTAL', 11551340.0, 21.82, 58.79]]
```

Por último se va a utilizar la lista de datos obtenidas para conseguir el siguiente dataframe:

In [15]:

```
columnas = ['anio', 'departamento', 'kg_lana', 'finura', 'rinde']
lana_chubut = pd.DataFrame(datos, columns=columnas)
lana_chubut
```

```
Out[15]:
```

	anio	departamento	kg_lana	finura	rinde
0	2000-01-01	BIEDMA	793900.00	20.39	59.21
1	2000-01-01	CUSHAMEN	892500.00	19.10	61.65
2	2000-01-01	ESCALANTE	1086300.00	21.90	49.99
3	2000-01-01	FLORENTINO AMEGHINO	1463700.00	19.61	60.32
4	2000-01-01	FUTALEUFU	557600.00	20.57	63.37
...	...	...	...	...	...
363	2022-01-01	RIO SENGUERR	1977589.41	19.72	58.07
364	2022-01-01	SARMIENTO	728889.55	20.94	49.20
365	2022-01-01	TEHUELCHES	1628738.94	18.47	60.56
366	2022-01-01	TELSEN	525585.97	19.11	55.35
367	2022-01-01	TOTAL	11551340.00	21.82	58.79

368 rows × 5 columns

Guardamos los datos en un archivo csv

```
In [16]: lana_chubut.to_csv('datos_producidos/lana_chubut_2000-2022.csv',
index=False)
```

Cada fila va a tener la siguiente información:

```
In [17]: lana_chubut.loc[128]
```

```
Out[17]: anio          2008-01-01
departamento    BIEDMA
kg_lana          1008720.0
finura           18.88
rinde            58.4
Name: 128, dtype: object
```


1.9. Generar dataset de metricas

La idea consiste en generar conjuntos de datos intermedios que contengan métricas para las existencias ovinas, los kilogramos de lana, la finura y el rinde. Estos datos intermedios nos permitirán completar el conjunto de datos final de manera más estructurada utilizando estas métricas.

```
In [1]: import numpy as np
import pandas as pd
```

Vamos a empezar con los datos para las existencias.

```
In [11]: datos = pd.read_csv("actualizando_dataset -
dataset_nuevos_porcentajes_existencias.csv")
datos
```

```
Out[11]:
```

	fecha	provincia	departamento	%_carneros	%_ovejas	%_borregos	%_capones	%_corderos_cordera
0	2013-12-31	CHUBUT	BIEDMA	2.28	41.05	15.81	19.39	21.4
1	2014-12-31	CHUBUT	BIEDMA	2.03	38.60	18.60	18.73	22.0
2	2013-12-31	CHUBUT	CUSHAMEN	2.70	48.83	14.92	10.01	23.5
3	2014-12-31	CHUBUT	CUSHAMEN	2.42	44.60	15.30	10.95	26.7
4	2013-12-31	CHUBUT	FLORENTINO AMEGHINO	2.38	39.30	18.77	14.61	24.9
5	2014-12-31	CHUBUT	FLORENTINO AMEGHINO	2.56	40.60	16.21	14.82	25.8
6	2013-12-31	CHUBUT	FUTALEUFU	3.20	49.78	12.70	12.78	21.5
7	2014-12-31	CHUBUT	FUTALEUFU	3.14	46.98	12.60	10.83	26.4
8	2013-12-31	CHUBUT	GAIMAN	2.43	51.81	11.90	14.89	18.9
9	2014-12-31	CHUBUT	GAIMAN	2.47	45.06	13.90	13.20	25.3
10	2013-12-31	CHUBUT	LANGUIÑEO	3.02	40.05	18.93	17.62	20.3
11	2015-12-31	CHUBUT	LANGUIÑEO	2.78	39.64	17.58	18.51	21.4
12	2013-12-31	CHUBUT	PASO DE INDIOS	2.57	47.14	14.45	17.79	18.0
13	2014-12-31	CHUBUT	PASO DE INDIOS	2.60	43.35	14.84	17.94	21.2
14	2013-12-31	CHUBUT	RAWSON	2.87	44.78	14.90	17.18	20.2

	fecha	provincia	departamento	%_carneros	%_ovejas	%_borregos	%_capones	%_corderos_cordera
15	2014-12-31	CHUBUT	RAWSON	2.80	42.44	14.43	17.12	23.2
16	2013-12-31	CHUBUT	RIO SENGUERR	2.52	42.83	18.06	14.02	22.5
17	2014-12-31	CHUBUT	RIO SENGUERR	2.78	44.05	16.18	14.88	22.1
18	2013-12-31	CHUBUT	SARMIENTO	3.30	47.20	14.49	16.89	18.1
19	2014-12-31	CHUBUT	SARMIENTO	2.89	43.94	15.14	17.55	20.4
20	2013-12-31	CHUBUT	TEHUELCHES	3.02	39.55	17.52	19.08	20.8
21	2014-12-31	CHUBUT	TEHUELCHES	3.12	39.96	17.04	20.42	19.4
22	2013-12-31	CHUBUT	TELSEN	2.72	43.55	16.59	21.55	15.5
23	2014-12-31	CHUBUT	TELSEN	2.53	42.68	16.55	16.26	21.9

In [13]:

```
datos_agrupado = datos.groupby("departamento").agg({
    "%_carneros": ["mean", "std", "min", "max"],
    "%_ovejas": ["mean", "std", "min", "max"],
    "%_borregos": ["mean", "std", "min", "max"],
    "%_capones": ["mean", "std", "min", "max"],
    "%_corderos_corderas": ["mean", "std", "min", "max"],
})
```

In [14]:

```
datos_agrupado.stack()
```

Out[14]:

			%_carneros	%_ovejas	%_borregos	%_capones	%_corderos_corderas
departamento							
BIEDMA	mean		2.155000	39.825000	17.205000	19.060000	21.750000
	std		0.176777	1.732412	1.972828	0.466690	0.395980
	min		2.030000	38.600000	15.810000	18.730000	21.470000
	max		2.280000	41.050000	18.600000	19.390000	22.030000
CUSHAMEN	mean		2.560000	46.715000	15.110000	10.480000	25.135000
	std		0.197990	2.991062	0.268701	0.664680	2.241528
	min		2.420000	44.600000	14.920000	10.010000	23.550000
	max		2.700000	48.830000	15.300000	10.950000	26.720000
FLORENTINO AMEGHINO	mean		2.470000	39.950000	17.490000	14.715000	25.370000
	std		0.127279	0.919239	1.810193	0.148492	0.608112
	min		2.380000	39.300000	16.210000	14.610000	24.940000

		%_carneros	%_ovejas	%_borregos	%_capones	%_corderos_corderas
departamento						
FUTALEUFU	max	2.560000	40.600000	18.770000	14.820000	25.800000
	mean	3.170000	48.380000	12.650000	11.805000	23.985000
	std	0.042426	1.979899	0.070711	1.378858	3.471894
	min	3.140000	46.980000	12.600000	10.830000	21.530000
GAIMAN	max	3.200000	49.780000	12.700000	12.780000	26.440000
	mean	2.450000	48.435000	12.900000	14.045000	22.170000
	std	0.028284	4.772971	1.414214	1.195010	4.525483
	min	2.430000	45.060000	11.900000	13.200000	18.970000
LANGUIÑO	max	2.470000	51.810000	13.900000	14.890000	25.370000
	mean	2.900000	39.845000	18.255000	18.065000	20.935000
	std	0.169706	0.289914	0.954594	0.629325	0.784889
	min	2.780000	39.640000	17.580000	17.620000	20.380000
PASO DE INDIOS	max	3.020000	40.050000	18.930000	18.510000	21.490000
	mean	2.585000	45.245000	14.645000	17.865000	19.660000
	std	0.021213	2.679935	0.275772	0.106066	2.291026
	min	2.570000	43.350000	14.450000	17.790000	18.040000
RAWSON	max	2.600000	47.140000	14.840000	17.940000	21.280000
	mean	2.835000	43.610000	14.665000	17.150000	21.740000
	std	0.049497	1.654630	0.332340	0.042426	2.078894
	min	2.800000	42.440000	14.430000	17.120000	20.270000
RIO SENGUERR	max	2.870000	44.780000	14.900000	17.180000	23.210000
	mean	2.650000	43.440000	17.120000	14.450000	22.335000
	std	0.183848	0.862670	1.329361	0.608112	0.318198
	min	2.520000	42.830000	16.180000	14.020000	22.110000
SARMIENTO	max	2.780000	44.050000	18.060000	14.880000	22.560000
	mean	3.095000	45.570000	14.815000	17.220000	19.300000
	std	0.289914	2.305168	0.459619	0.466690	1.668772
	min	2.890000	43.940000	14.490000	16.890000	18.120000
TEHUELCHES	max	3.300000	47.200000	15.140000	17.550000	20.480000
	mean	3.070000	39.755000	17.280000	19.750000	20.135000
	std	0.070711	0.289914	0.339411	0.947523	0.968736
	min	3.020000	39.550000	17.040000	19.080000	19.450000
TELSEN	max	3.120000	39.960000	17.520000	20.420000	20.820000
	mean	2.625000	43.115000	16.570000	18.905000	18.785000
	std	0.134350	0.615183	0.028284	3.740595	4.518412
	min	2.530000	42.680000	16.550000	16.260000	15.590000

	%_carneros	%_ovejas	%_borregos	%_capones	%_corderos_corderas
departamento					
max	2.720000	43.550000	16.590000	21.550000	21.980000

```
In [16]: df_resultante = datos_agrupado.stack().reset_index()
df_resultante.rename(columns={'level_1': 'metrica',
                              '%_carneros': 'carneros',
                              '%_ovejas': 'ovejas',
                              '%_borregos': 'borregos',
                              '%_capones': 'capones',
                              '%_corderos_corderas': 'corderos_corderas'}, inplace=True)

df_resultante
```

	departamento	metrica	carneros	ovejas	borregos	capones	corderos_corderas
0	BIEDMA	mean	2.155000	39.825000	17.205000	19.060000	21.750000
1	BIEDMA	std	0.176777	1.732412	1.972828	0.466690	0.395980
2	BIEDMA	min	2.030000	38.600000	15.810000	18.730000	21.470000
3	BIEDMA	max	2.280000	41.050000	18.600000	19.390000	22.030000
4	CUSHAMEN	mean	2.560000	46.715000	15.110000	10.480000	25.135000
5	CUSHAMEN	std	0.197990	2.991062	0.268701	0.664680	2.241528
6	CUSHAMEN	min	2.420000	44.600000	14.920000	10.010000	23.550000
7	CUSHAMEN	max	2.700000	48.830000	15.300000	10.950000	26.720000
8	FLORENTINO AMEGHINO	mean	2.470000	39.950000	17.490000	14.715000	25.370000
9	FLORENTINO AMEGHINO	std	0.127279	0.919239	1.810193	0.148492	0.608112
10	FLORENTINO AMEGHINO	min	2.380000	39.300000	16.210000	14.610000	24.940000
11	FLORENTINO AMEGHINO	max	2.560000	40.600000	18.770000	14.820000	25.800000
12	FUTALEUFU	mean	3.170000	48.380000	12.650000	11.805000	23.985000
13	FUTALEUFU	std	0.042426	1.979899	0.070711	1.378858	3.471894
14	FUTALEUFU	min	3.140000	46.980000	12.600000	10.830000	21.530000
15	FUTALEUFU	max	3.200000	49.780000	12.700000	12.780000	26.440000
16	GAIMAN	mean	2.450000	48.435000	12.900000	14.045000	22.170000
17	GAIMAN	std	0.028284	4.772971	1.414214	1.195010	4.525483
18	GAIMAN	min	2.430000	45.060000	11.900000	13.200000	18.970000
19	GAIMAN	max	2.470000	51.810000	13.900000	14.890000	25.370000
20	LANGUIÑEO	mean	2.900000	39.845000	18.255000	18.065000	20.935000
21	LANGUIÑEO	std	0.169706	0.289914	0.954594	0.629325	0.784889
22	LANGUIÑEO	min	2.780000	39.640000	17.580000	17.620000	20.380000
23	LANGUIÑEO	max	3.020000	40.050000	18.930000	18.510000	21.490000
24	PASO DE INDIOS	mean	2.585000	45.245000	14.645000	17.865000	19.660000

	departamento	metrica	carneros	ovejas	borregos	capones	corderos_corderas
25	PASO DE INDIOS	std	0.021213	2.679935	0.275772	0.106066	2.291026
26	PASO DE INDIOS	min	2.570000	43.350000	14.450000	17.790000	18.040000
27	PASO DE INDIOS	max	2.600000	47.140000	14.840000	17.940000	21.280000
28	RAWSON	mean	2.835000	43.610000	14.665000	17.150000	21.740000
29	RAWSON	std	0.049497	1.654630	0.332340	0.042426	2.078894
30	RAWSON	min	2.800000	42.440000	14.430000	17.120000	20.270000
31	RAWSON	max	2.870000	44.780000	14.900000	17.180000	23.210000
32	RIO SENGUERR	mean	2.650000	43.440000	17.120000	14.450000	22.335000
33	RIO SENGUERR	std	0.183848	0.862670	1.329361	0.608112	0.318198
34	RIO SENGUERR	min	2.520000	42.830000	16.180000	14.020000	22.110000
35	RIO SENGUERR	max	2.780000	44.050000	18.060000	14.880000	22.560000
36	SARMIENTO	mean	3.095000	45.570000	14.815000	17.220000	19.300000
37	SARMIENTO	std	0.289914	2.305168	0.459619	0.466690	1.668772
38	SARMIENTO	min	2.890000	43.940000	14.490000	16.890000	18.120000
39	SARMIENTO	max	3.300000	47.200000	15.140000	17.550000	20.480000
40	TEHUELCHES	mean	3.070000	39.755000	17.280000	19.750000	20.135000
41	TEHUELCHES	std	0.070711	0.289914	0.339411	0.947523	0.968736
42	TEHUELCHES	min	3.020000	39.550000	17.040000	19.080000	19.450000
43	TEHUELCHES	max	3.120000	39.960000	17.520000	20.420000	20.820000
44	TELSEN	mean	2.625000	43.115000	16.570000	18.905000	18.785000
45	TELSEN	std	0.134350	0.615183	0.028284	3.740595	4.518412
46	TELSEN	min	2.530000	42.680000	16.550000	16.260000	15.590000
47	TELSEN	max	2.720000	43.550000	16.590000	21.550000	21.980000

Por último, guardamos el DataFrame como archivo CSV.

```
In [ ]: df_resultante.to_csv("nuevo_porcentajes_existencias.csv",
index=False)
```

Continuamos con los datos para la lana.

```
In [2]: datos = pd.read_csv("actualizando_dataset -
dataset_nuevos_%_lana.csv")
datos
```

```
Out[2]:
```

	fecha	provincia	departamento	kilos_animal
0	2012-12-31	CHUBUT	BIEDMA	3.61
1	2013-12-31	CHUBUT	BIEDMA	3.16
2	2014-12-31	CHUBUT	BIEDMA	3.24
3	2006-12-31	CHUBUT	CUSHAMEN	3.55

	fecha	provincia	departamento	kilos_animal
4	2007-12-31	CHUBUT	CUSHAMEN	3.42
...	...	...	...	...
73	2010-12-31	CHUBUT	TELSEN	3.67
74	2011-12-31	CHUBUT	TELSEN	3.92
75	2012-12-31	CHUBUT	TELSEN	3.81
76	2013-12-31	CHUBUT	TELSEN	3.50
77	2014-12-31	CHUBUT	TELSEN	3.62

78 rows × 4 columns

In [3]:

```
datos_agrupado = datos.groupby("departamento").agg({
    "kilos_animal": ["mean", "std", "min", "max"],
})
```

In [4]:

```
datos_agrupado.stack()
```

Out[4]:

		kilos_animal	
departamento			
FLORENTINO AMEGHINO	BIEDMA	mean	3.336667
		std	0.240069
		min	3.160000
		max	3.610000
	CUSHAMEN	mean	3.407500
		std	0.295284
		min	2.730000
		max	3.690000
		mean	3.286667
		std	0.430473
		min	2.730000
		max	3.690000
FUTALEUFU	mean	3.648750	
	std	0.655274	
	min	2.990000	
	max	5.090000	
GAIMAN	mean	3.440000	
	std	0.153716	
	min	3.190000	
	max	3.720000	
LANGUIÑEO	mean	3.483333	

departamento		kilos_animal
	std	0.143643
	min	3.320000
	max	3.590000
	mean	3.568333
PASO DE INDIOS	std	0.179490
	min	3.410000
	max	3.910000
	mean	3.441250
RAWSON	std	0.178920
	min	3.230000
	max	3.710000
	mean	3.625000
RIO SENGUERR	std	0.230403
	min	3.250000
	max	3.880000
	mean	3.713750
SARMIENTO	std	0.224750
	min	3.230000
	max	3.900000
	mean	3.758333
TEHUELCHES	std	0.161173
	min	3.510000
	max	3.940000
	mean	3.665000
TELSEN	std	0.175129
	min	3.470000
	max	3.920000
	mean	3.920000

```
In [6]: df_resultante = datos_agrupado.stack().reset_index()
df_resultante.rename(columns={'level_1': 'metrica'}, inplace=True)
df_resultante
```

```
Out[6]:
```

	departamento	metrica	kilos_animal
0	BIEDMA	mean	3.336667
1	BIEDMA	std	0.240069
2	BIEDMA	min	3.160000
3	BIEDMA	max	3.610000

	departamento	metrica	kilos_animal
4	CUSHAMEN	mean	3.407500
5	CUSHAMEN	std	0.295284
6	CUSHAMEN	min	2.730000
7	CUSHAMEN	max	3.690000
8	FLORENTINO AMEGHINO	mean	3.286667
9	FLORENTINO AMEGHINO	std	0.430473
10	FLORENTINO AMEGHINO	min	2.730000
11	FLORENTINO AMEGHINO	max	3.690000
12	FUTALEUFU	mean	3.648750
13	FUTALEUFU	std	0.655274
14	FUTALEUFU	min	2.990000
15	FUTALEUFU	max	5.090000
16	GAIMAN	mean	3.440000
17	GAIMAN	std	0.153716
18	GAIMAN	min	3.190000
19	GAIMAN	max	3.720000
20	LANGUIÑEO	mean	3.483333
21	LANGUIÑEO	std	0.143643
22	LANGUIÑEO	min	3.320000
23	LANGUIÑEO	max	3.590000
24	PASO DE INDIOS	mean	3.568333
25	PASO DE INDIOS	std	0.179490
26	PASO DE INDIOS	min	3.410000
27	PASO DE INDIOS	max	3.910000
28	RAWSON	mean	3.441250
29	RAWSON	std	0.178920
30	RAWSON	min	3.230000
31	RAWSON	max	3.710000
32	RIO SENGUERR	mean	3.625000
33	RIO SENGUERR	std	0.230403
34	RIO SENGUERR	min	3.250000
35	RIO SENGUERR	max	3.880000
36	SARMIENTO	mean	3.713750
37	SARMIENTO	std	0.224750
38	SARMIENTO	min	3.230000
39	SARMIENTO	max	3.900000
40	TEHUELCHES	mean	3.758333

	departamento	metrica	kilos_animal
41	TEHUELCHES	std	0.161173
42	TEHUELCHES	min	3.510000
43	TEHUELCHES	max	3.940000
44	TELSEN	mean	3.665000
45	TELSEN	std	0.175129
46	TELSEN	min	3.470000
47	TELSEN	max	3.920000

Por último, guardamos el DataFrame como archivo CSV.

```
In [ ]: df_resultante.to_csv("nuevo_porcentajes_lana.csv", index=False)
```

Ahora toca hacer lo mismo para la finura y el rinde.

```
In [7]: datos = pd.read_csv("actualizando_dataset -
dataset_nuevos_%_f_r.csv")
datos
```

```
Out[7]:
```

	fecha	provincia	departamento	finura	rinde
0	2013-12-31	CHUBUT	BIEDMA	19.80	55.00
1	2014-12-31	CHUBUT	BIEDMA	19.22	58.86
2	2013-12-31	CHUBUT	CUSHAMEN	20.04	57.47
3	2014-12-31	CHUBUT	CUSHAMEN	19.74	63.35
4	2013-12-31	CHUBUT	FLORENTINO AMEGHINO	19.30	57.75
5	2014-12-31	CHUBUT	FLORENTINO AMEGHINO	19.53	60.37
6	2013-12-31	CHUBUT	FUTALEUFU	21.60	63.90
7	2014-12-31	CHUBUT	FUTALEUFU	20.27	63.13
8	2013-12-31	CHUBUT	GAIMAN	20.80	56.20
9	2014-12-31	CHUBUT	GAIMAN	20.39	55.06
10	2013-12-31	CHUBUT	LANGUIÑEO	20.00	55.40
11	2015-12-31	CHUBUT	LANGUIÑEO	19.86	60.94
12	2013-12-31	CHUBUT	PASO DE INDIOS	20.20	52.60
13	2014-12-31	CHUBUT	PASO DE INDIOS	19.92	54.19
14	2013-12-31	CHUBUT	RAWSON	20.30	55.67
15	2014-12-31	CHUBUT	RAWSON	19.78	61.22
16	2013-12-31	CHUBUT	RIO SENGUERR	20.25	55.86
17	2014-12-31	CHUBUT	RIO SENGUERR	20.10	59.01
18	2013-12-31	CHUBUT	SARMIENTO	20.73	49.13
19	2014-12-31	CHUBUT	SARMIENTO	19.92	48.21
20	2013-12-31	CHUBUT	TEHUELCHES	20.05	60.07

	fecha	provincia	departamento	finura	rinde
21	2014-12-31	CHUBUT	TEHUELCHES	19.41	61.21
22	2013-12-31	CHUBUT	TELSEN	20.01	51.90
23	2014-12-31	CHUBUT	TELSEN	19.79	54.01

```
In [8]: datos_agrupado = datos.groupby("departamento").agg({
    "finura": ["mean", "std", "min", "max"],
    "rinde": ["mean", "std", "min", "max"],
})
```

```
In [9]: datos_agrupado.stack()
```

```
Out[9]:
```

			finura	rinde
	departamento			
	BIEDMA	mean	19.510000	56.930000
		std	0.410122	2.729432
		min	19.220000	55.000000
		max	19.800000	58.860000
	CUSHAMEN	mean	19.890000	60.410000
		std	0.212132	4.157788
		min	19.740000	57.470000
		max	20.040000	63.350000
	FLORENTINO AMEGHINO	mean	19.415000	59.060000
		std	0.162635	1.852620
		min	19.300000	57.750000
		max	19.530000	60.370000
	FUTALEUFU	mean	20.935000	63.515000
		std	0.940452	0.544472
		min	20.270000	63.130000
		max	21.600000	63.900000
	GAIMAN	mean	20.595000	55.630000
		std	0.289914	0.806102
		min	20.390000	55.060000
		max	20.800000	56.200000
	LANGUIÑEO	mean	19.930000	58.170000
		std	0.098995	3.917372
		min	19.860000	55.400000
		max	20.000000	60.940000
	PASO DE INDIOS	mean	20.060000	53.395000

		finura	rinde
departamento			
	std	0.197990	1.124300
	min	19.920000	52.600000
	max	20.200000	54.190000
	mean	20.040000	58.445000
RAWSON	std	0.367696	3.924443
	min	19.780000	55.670000
	max	20.300000	61.220000
	mean	20.175000	57.435000
RIO SENGUERR	std	0.106066	2.227386
	min	20.100000	55.860000
	max	20.250000	59.010000
	mean	20.325000	48.670000
SARMIENTO	std	0.572756	0.650538
	min	19.920000	48.210000
	max	20.730000	49.130000
	mean	19.730000	60.640000
TEHUELCHES	std	0.452548	0.806102
	min	19.410000	60.070000
	max	20.050000	61.210000
	mean	19.900000	52.955000
TELSEN	std	0.155563	1.491995
	min	19.790000	51.900000
	max	20.010000	54.010000
	mean	19.900000	52.955000

```
In [10]: df_resultante = datos_agrupado.stack().reset_index()
df_resultante.rename(columns={'level_1':'metrica'}, inplace=True)
df_resultante
```

	departamento	metrica	finura	rinde
0	BIEDMA	mean	19.510000	56.930000
1	BIEDMA	std	0.410122	2.729432
2	BIEDMA	min	19.220000	55.000000
3	BIEDMA	max	19.800000	58.860000
4	CUSHAMEN	mean	19.890000	60.410000
5	CUSHAMEN	std	0.212132	4.157788
6	CUSHAMEN	min	19.740000	57.470000
7	CUSHAMEN	max	20.040000	63.350000

	departamento	metrica	finura	rinde
8	FLORENTINO AMEGHINO	mean	19.415000	59.060000
9	FLORENTINO AMEGHINO	std	0.162635	1.852620
10	FLORENTINO AMEGHINO	min	19.300000	57.750000
11	FLORENTINO AMEGHINO	max	19.530000	60.370000
12	FUTALEUFU	mean	20.935000	63.515000
13	FUTALEUFU	std	0.940452	0.544472
14	FUTALEUFU	min	20.270000	63.130000
15	FUTALEUFU	max	21.600000	63.900000
16	GAIMAN	mean	20.595000	55.630000
17	GAIMAN	std	0.289914	0.806102
18	GAIMAN	min	20.390000	55.060000
19	GAIMAN	max	20.800000	56.200000
20	LANGUIÑEO	mean	19.930000	58.170000
21	LANGUIÑEO	std	0.098995	3.917372
22	LANGUIÑEO	min	19.860000	55.400000
23	LANGUIÑEO	max	20.000000	60.940000
24	PASO DE INDIOS	mean	20.060000	53.395000
25	PASO DE INDIOS	std	0.197990	1.124300
26	PASO DE INDIOS	min	19.920000	52.600000
27	PASO DE INDIOS	max	20.200000	54.190000
28	RAWSON	mean	20.040000	58.445000
29	RAWSON	std	0.367696	3.924443
30	RAWSON	min	19.780000	55.670000
31	RAWSON	max	20.300000	61.220000
32	RIO SENGUERR	mean	20.175000	57.435000
33	RIO SENGUERR	std	0.106066	2.227386
34	RIO SENGUERR	min	20.100000	55.860000
35	RIO SENGUERR	max	20.250000	59.010000
36	SARMIENTO	mean	20.325000	48.670000
37	SARMIENTO	std	0.572756	0.650538
38	SARMIENTO	min	19.920000	48.210000
39	SARMIENTO	max	20.730000	49.130000
40	TEHUELCHES	mean	19.730000	60.640000
41	TEHUELCHES	std	0.452548	0.806102
42	TEHUELCHES	min	19.410000	60.070000
43	TEHUELCHES	max	20.050000	61.210000
44	TELSEN	mean	19.900000	52.955000

	departamento	metrica	finura	rinde
45	TELSEN	std	0.155563	1.491995
46	TELSEN	min	19.790000	51.900000
47	TELSEN	max	20.010000	54.010000

Por último, guardamos el DataFrame como archivo CSV.

```
In [ ]: df_resultante.to_csv("nuevo_porcentajes_f_r.csv", index=False)
```

1.10. Aplicar métricas de existencias

In [12]:

```
import numpy as np
import pandas as pd
import random
```

Después de generar el nuevo archivo con las nuevas métricas para generar los valores, cargamos este archivo csv para convertirlo en un DataFrame y trabajarlo en combinación con el otro más adelante.

In [13]:

```
metricas = pd.read_csv("nuevo_porcentajes_existencias.csv")
metricas
```

Out[13]:

	departamento	metrica	carneros	ovejas	borregos	capones	corderos_corderas
0	BIEDMA	mean	2.155000	39.825000	17.205000	19.060000	21.750000
1	BIEDMA	std	0.176777	1.732412	1.972828	0.466690	0.395980
2	BIEDMA	min	2.030000	38.600000	15.810000	18.730000	21.470000
3	BIEDMA	max	2.280000	41.050000	18.600000	19.390000	22.030000
4	CUSHAMEN	mean	2.560000	46.715000	15.110000	10.480000	25.135000
5	CUSHAMEN	std	0.197990	2.991062	0.268701	0.664680	2.241528
6	CUSHAMEN	min	2.420000	44.600000	14.920000	10.010000	23.550000
7	CUSHAMEN	max	2.700000	48.830000	15.300000	10.950000	26.720000
8	FLORENTINO AMEGHINO	mean	2.470000	39.950000	17.490000	14.715000	25.370000
9	FLORENTINO AMEGHINO	std	0.127279	0.919239	1.810193	0.148492	0.608112
10	FLORENTINO AMEGHINO	min	2.380000	39.300000	16.210000	14.610000	24.940000
11	FLORENTINO AMEGHINO	max	2.560000	40.600000	18.770000	14.820000	25.800000
12	FUTALEUFU	mean	3.170000	48.380000	12.650000	11.805000	23.985000
13	FUTALEUFU	std	0.042426	1.979899	0.070711	1.378858	3.471894
14	FUTALEUFU	min	3.140000	46.980000	12.600000	10.830000	21.530000
15	FUTALEUFU	max	3.200000	49.780000	12.700000	12.780000	26.440000
16	GAIMAN	mean	2.450000	48.435000	12.900000	14.045000	22.170000
17	GAIMAN	std	0.028284	4.772971	1.414214	1.195010	4.525483
18	GAIMAN	min	2.430000	45.060000	11.900000	13.200000	18.970000
19	GAIMAN	max	2.470000	51.810000	13.900000	14.890000	25.370000
20	LANGUIÑEO	mean	2.900000	39.845000	18.255000	18.065000	20.935000
21	LANGUIÑEO	std	0.169706	0.289914	0.954594	0.629325	0.784889
22	LANGUIÑEO	min	2.780000	39.640000	17.580000	17.620000	20.380000
23	LANGUIÑEO	max	3.020000	40.050000	18.930000	18.510000	21.490000
24	PASO DE INDIOS	mean	2.585000	45.245000	14.645000	17.865000	19.660000
25	PASO DE INDIOS	std	0.021213	2.679935	0.275772	0.106066	2.291026

	departamento	metrica	carneros	ovejas	borregos	capones	corderos_corderas
26	PASO DE INDIOS	min	2.570000	43.350000	14.450000	17.790000	18.040000
27	PASO DE INDIOS	max	2.600000	47.140000	14.840000	17.940000	21.280000
28	RAWSON	mean	2.835000	43.610000	14.665000	17.150000	21.740000
29	RAWSON	std	0.049497	1.654630	0.332340	0.042426	2.078894
30	RAWSON	min	2.800000	42.440000	14.430000	17.120000	20.270000
31	RAWSON	max	2.870000	44.780000	14.900000	17.180000	23.210000
32	RIO SENGUERR	mean	2.650000	43.440000	17.120000	14.450000	22.335000
33	RIO SENGUERR	std	0.183848	0.862670	1.329361	0.608112	0.318198
34	RIO SENGUERR	min	2.520000	42.830000	16.180000	14.020000	22.110000
35	RIO SENGUERR	max	2.780000	44.050000	18.060000	14.880000	22.560000
36	SARMIENTO	mean	3.095000	45.570000	14.815000	17.220000	19.300000
37	SARMIENTO	std	0.289914	2.305168	0.459619	0.466690	1.668772
38	SARMIENTO	min	2.890000	43.940000	14.490000	16.890000	18.120000
39	SARMIENTO	max	3.300000	47.200000	15.140000	17.550000	20.480000
40	TEHUELCHES	mean	3.070000	39.755000	17.280000	19.750000	20.135000
41	TEHUELCHES	std	0.070711	0.289914	0.339411	0.947523	0.968736
42	TEHUELCHES	min	3.020000	39.550000	17.040000	19.080000	19.450000
43	TEHUELCHES	max	3.120000	39.960000	17.520000	20.420000	20.820000
44	TELSEN	mean	2.625000	43.115000	16.570000	18.905000	18.785000
45	TELSEN	std	0.134350	0.615183	0.028284	3.740595	4.518412
46	TELSEN	min	2.530000	42.680000	16.550000	16.260000	15.590000
47	TELSEN	max	2.720000	43.550000	16.590000	21.550000	21.980000

Lo siguiente es cargar el nuevo Dataset de datos productivos y el clima ajustados con los nuevos valores obtenidos de las estadísticas ganaderas de Chubut, y con una nueva columna de estado para saber cuáles son las filas que deben ser modificadas.

In [14]:

```
datos = pd.read_csv("actualizando_dataset - Existencias.csv")
datos
```

Out[14]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	5785	93378	30168	48447	1601
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	3690
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	3492
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	4440	89262	29663	35437	3234
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	4452	82295	33662	33470	2132
...	...	...	...	...	...	...	...	...	...

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4896	80262	26471	39243	2029
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4640	77556	27296	36570	1774
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4832	79117	28366	37558	2193
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4781	80109	28428	38109	1774
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	2870	79157	25804	36901	1898

164 rows × 10 columns

Ahora vamos a separar este DataFrame en las filas que son modificables y las que no, obteniendo como resultado un DataFrame de datos modificables y un DataFrame de datos reales, que no tienen que ser modificados.

In [15]:

```
df_real = datos[datos['estado'] == 'REAL']
df_real
```

Out[15]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	36961	
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	34926	
16	2013-12-31	CHUBUT	REAL	CUSHAMEN	4751	85946	26255	17619	41451	
17	2014-12-31	CHUBUT	REAL	CUSHAMEN	4198	77263	26512	18975	46290	
28	2013-12-31	CHUBUT	REAL	FLORENTINO AMEGHINO	9114	150199	71752	55841	95320	
29	2014-12-31	CHUBUT	REAL	FLORENTINO AMEGHINO	8893	140928	56282	51459	89550	
43	2013-12-31	CHUBUT	REAL	FUTALEUFU	3594	55861	14255	14340	24160	
44	2014-12-31	CHUBUT	REAL	FUTALEUFU	3436	51384	13783	11841	28919	
58	2013-12-31	CHUBUT	REAL	GAIMAN	4102	87337	20059	25098	31970	
59	2014-12-31	CHUBUT	REAL	GAIMAN	4936	90039	27772	26376	50701	
67	2013-12-31	CHUBUT	REAL	LANGUIÑEO	7275	96402	45552	42406	49046	
68	2015-12-31	CHUBUT	REAL	LANGUIÑEO	6151	87640	38874	40929	47517	
79	2013-12-31	CHUBUT	REAL	PASO DE INDIOS	4653	85441	26190	32246	32703	

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas	tota
80	2014-12-31	CHUBUT	REAL	PASO DE INDIOS	5398	89998	30804	37250		44180
100	2013-12-31	CHUBUT	REAL	RAWSON	1673	26143	8701	10029		11832
101	2014-12-31	CHUBUT	REAL	RAWSON	1446	21913	7450	8839		11982
116	2013-12-31	CHUBUT	REAL	RIO SENGUERR	17403	295275	124490	96655		155537
117	2014-12-31	CHUBUT	REAL	RIO SENGUERR	15571	246882	90683	83406		123913
131	2013-12-31	CHUBUT	REAL	SARMIENTO	5978	85638	26288	30644		32877
132	2014-12-31	CHUBUT	REAL	SARMIENTO	5269	80123	27608	32012		37346
144	2013-12-31	CHUBUT	REAL	TEHUELCHES	13577	177588	78690	85677		93487
145	2014-12-31	CHUBUT	REAL	TEHUELCHES	13819	176838	75419	90371		86049
156	2013-12-31	CHUBUT	REAL	TELSEN	3899	62509	23816	30940		22378
157	2014-12-31	CHUBUT	REAL	TELSEN	3495	59018	22886	22489		30400

In [16]:

```
df_modificable = datos[datos['estado'] == 'MODIFICABLE']
df_modificable
```

Out[16]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas	tota
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	5785	93378	30168	48447		1605
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	4440	89262	29663	35437		3234
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	4452	82295	33662	33470		2132
5	2017-12-31	CHUBUT	MODIFICABLE	BIEDMA	4688	87179	33148	33947		2090
6	2018-12-31	CHUBUT	MODIFICABLE	BIEDMA	4767	86473	36233	31347		2228
...	...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4896	80262	26471	39243		2029
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4640	77556	27296	36570		1775
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4832	79117	28366	37558		2195
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4781	80109	28428	38109		1775

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	2870	79157	25804	36901	1898

140 rows × 18 columns

Creamos una lista con los nombres de las categorías de ovinos

```
In [17]: categorias_ovinos = ['carneros', 'ovejas', 'borregos', 'capones',
                              'corderos_corderas']
```

Después creamos una función llamada "aleatorio_rango", que toma como entrada una media, una desviación estándar, un valor mínimo y un valor máximo. La función utiliza la función "random.normalvariate" de Python para generar un valor aleatorio con la media y la desviación estándar dadas, y redondea el resultado a dos decimales. Si el valor generado está fuera del rango mínimo y máximo especificado, o si la suma acumulada de los valores generados excede el total de ovinos, la función vuelve a generar otro valor. Esto nos permite ajustar los valores aleatorios generados para que la suma total de los mismos sea exactamente igual al número total de ovinos.

```
In [ ]: def aleatorio_rango(media, desvio, minimo, maximo, total_ovino,
                             sum_calculado):
    maximo_sum = total_ovino - sum_calculado # El máximo permitido
    para esta categoría depende de la suma de las categorías
    anteriores
    valor_aleatorio = round(random.normalvariate(media, desvio),
                             2)
    while valor_aleatorio < minimo or valor_aleatorio > maximo or
    sum_calculado + valor_aleatorio > total_ovino:
        valor_aleatorio = round(random.normalvariate(media,
        desvio), 2)
    return valor_aleatorio
```

El siguiente paso consiste en recorrer el DataFrame modificable de manera iterativa, extrayendo los departamentos uno por uno y comparándolos con los departamentos presentes en el DataFrame de métricas. A continuación, se recorren las diferentes categorías correspondientes a los ovinos y se obtienen las métricas asociadas, tales como la media, el desvío, el mínimo y el máximo. Luego, se genera un valor aleatorio para cada categoría y se aplica la fórmula correspondiente para obtener el valor calculado. Finalmente, se almacena el valor obtenido en el DataFrame modificable.

```
In [ ]: for index, row in df_modificable.iterrows():
    departamento = row['departamento']
    df1_departamento = metricas[metricas['departamento'] ==
    departamento]
    total_ovino = row['total_ovinos']
    sum_calculado = 0
```

```

# Generamos los valores aleatorios para cada categoría de
ovinos
valores = []
for categoria in categorias_ovinos:
    media = df1_departamento[categoria].iloc[0]
    desvio = df1_departamento[categoria].iloc[1]
    minimo = df1_departamento[categoria].iloc[2]
    maximo = df1_departamento[categoria].iloc[3]
    valor_aleatorio = aleatorio_rango(media, desvio, minimo,
maximo, total_ovino, sum_calculado)
    sum_calculado += valor_aleatorio
    valores.append(valor_aleatorio)

# Ajustamos los valores calculados para que la suma sea igual
a total_ovino
factor_ajuste = total_ovino / sum(valores)
valores_ajustados = [round(valor * factor_ajuste) for valor in
valores]

# Ajustamos la última categoría para que la suma total sea
exactamente igual al total de ovinos
diferencia = total_ovino - sum(valores_ajustados)
valores_ajustados[-1] += diferencia

# Almacenamos los valores calculados en las nuevas columnas
del segundo DataFrame
for i, categoria in enumerate(categorias_ovinos):
    df_modificable.loc[index, categoria] =
valores_ajustados[i]

```

Como resultado obtenemos un nuevo DataFrame con los nuevos valores para cada categoría ovina en base al total de ovinos.

In [30]:

```
df_modificable
```

Out[30]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	2740	52320	22319	25489	2951
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
5	2017-12-31	CHUBUT	MODIFICABLE	BIEDMA	3900	71790	32259	34024	3788
6	2018-12-31	CHUBUT	MODIFICABLE	BIEDMA	4070	70696	32886	34339	3911
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

140 rows × 10 columns

Lo siguiente es concatenar los dos dataset: el de datos reales y el de datos modificables para obtener un nuevo Dataset actualizado.

```
In [31]: df_actualizado = pd.concat([df_real, df_modificable])
df_actualizado
```

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	369
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	349
16	2013-12-31	CHUBUT	REAL	CUSHAMEN	4751	85946	26255	17619	419
17	2014-12-31	CHUBUT	REAL	CUSHAMEN	4198	77263	26512	18975	469
28	2013-12-31	CHUBUT	REAL	FLORENTINO AMEGHINO	9114	150199	71752	55841	959
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

164 rows × 10 columns

Luego ordenamos el Data frame por el índice.

```
In [32]: df_ordenado = df_actualizado.sort_index()  
df_ordenado
```

```
Out[32]:
```

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	2740	52320	22319	25489	2951
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	3696
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	3492
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

164 rows × 10 columns

Y finalmente guardamos el resultado en un archivo csv.

```
In [33]: df_ordenado.to_csv("dataset_aplicado_%_existencias.csv",  
index=False)
```

1.11. Aplicar métricas de lana

```
In [1]: import numpy as np
import pandas as pd
import random
```

Después de generar el nuevo archivo con las nuevas métricas para generar los valores, cargamos este archivo csv para convertirlo en un DataFrame y trabajarlo en combinación con el otro más adelante.

```
In [2]: metricas = pd.read_csv("nuevo_porcentajes_lana.csv")
metricas
```

```
Out[2]:
```

	departamento	metrica	kilos_animal
0	BIEDMA	mean	3.336667
1	BIEDMA	std	0.240069
2	BIEDMA	min	3.160000
3	BIEDMA	max	3.610000
4	CUSHAMEN	mean	3.407500
5	CUSHAMEN	std	0.295284
6	CUSHAMEN	min	2.730000
7	CUSHAMEN	max	3.690000
8	FLORENTINO AMEGHINO	mean	3.286667
9	FLORENTINO AMEGHINO	std	0.430473
10	FLORENTINO AMEGHINO	min	2.730000
11	FLORENTINO AMEGHINO	max	3.690000
12	FUTALEUFU	mean	3.648750
13	FUTALEUFU	std	0.655274
14	FUTALEUFU	min	2.990000
15	FUTALEUFU	max	5.090000
16	GAIMAN	mean	3.440000
17	GAIMAN	std	0.153716
18	GAIMAN	min	3.190000
19	GAIMAN	max	3.720000
20	LANGUIÑEO	mean	3.483333
21	LANGUIÑEO	std	0.143643
22	LANGUIÑEO	min	3.320000
23	LANGUIÑEO	max	3.590000
24	PASO DE INDIOS	mean	3.568333
25	PASO DE INDIOS	std	0.179490

	departamento	metrica	kilos_animal
26	PASO DE INDIOS	min	3.410000
27	PASO DE INDIOS	max	3.910000
28	RAWSON	mean	3.441250
29	RAWSON	std	0.178920
30	RAWSON	min	3.230000
31	RAWSON	max	3.710000
32	RIO SENGUERR	mean	3.625000
33	RIO SENGUERR	std	0.230403
34	RIO SENGUERR	min	3.250000
35	RIO SENGUERR	max	3.880000
36	SARMIENTO	mean	3.713750
37	SARMIENTO	std	0.224750
38	SARMIENTO	min	3.230000
39	SARMIENTO	max	3.900000
40	TEHUELCHES	mean	3.758333
41	TEHUELCHES	std	0.161173
42	TEHUELCHES	min	3.510000
43	TEHUELCHES	max	3.940000
44	TELSEN	mean	3.665000
45	TELSEN	std	0.175129
46	TELSEN	min	3.470000
47	TELSEN	max	3.920000

Lo siguiente es cargar el nuevo Dataset de datos productivos y el clima ajustados con los nuevos valores obtenidos de las estadísticas ganaderas de Chubut, y con una nueva columna de estado para saber cuáles son las filas que deben ser modificadas.

In [10]:

```
datos = pd.read_csv("actualizando_dataset - Lana.csv")
datos
```

Out[10]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
0	2012-12-31	CHUBUT	REAL	BIEDMA	2740	52320	22319	25489	2951
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	3690
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	3492
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792
...	...	...	...	...	...	...	...	...	...

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

164 rows × 19 columns

Ahora vamos a separar este DataFrame en las filas que son modificables y las que no, obteniendo como resultado un DataFrame de datos modificables y un DataFrame de datos reales, que no tienen que ser modificados.

In [11]:

```
df_real = datos[datos['estado'] == 'REAL']
df_real
```

Out[11]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total
0	2012-12-31	CHUBUT	REAL	BIEDMA	2740	52320	22319	25489	29517	109685
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	36961	142180
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	34926	128523
9	2006-12-31	CHUBUT	REAL	CUSHAMEN	5218	94823	31143	21497	49064	151745
10	2007-12-31	CHUBUT	REAL	CUSHAMEN	4603	83411	27453	19643	47297	132807
...	...	...	...	...	...	...	...	...	...	...
153	2010-12-31	CHUBUT	REAL	TELSEN	5466	89712	34742	37485	33778	151183
154	2011-12-31	CHUBUT	REAL	TELSEN	4186	70414	27322	30519	33896	136337
155	2012-12-31	CHUBUT	REAL	TELSEN	3250	53993	20715	21365	24014	102337
156	2013-12-31	CHUBUT	REAL	TELSEN	3899	62509	23816	30940	22378	113532
157	2014-12-31	CHUBUT	REAL	TELSEN	3495	59018	22886	22489	30400	107828

78 rows × 19 columns

In [12]:

```
df_modificable = datos[datos['estado'] == 'MODIFICABLE']
df_modificable
```

Out[12]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792
5	2017-12-31	CHUBUT	MODIFICABLE	BIEDMA	3900	71790	32259	34024	3788
6	2018-12-31	CHUBUT	MODIFICABLE	BIEDMA	4070	70696	32886	34339	3911
7	2019-12-31	CHUBUT	MODIFICABLE	BIEDMA	3606	70938	32085	33311	3858
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

86 rows × 19 columns

Luego creamos una función `aleatorio_rango` que recibe una media, un desvío, un mínimo y un máximo. Lo que hacemos es utilizar el `random.normalvariate` de python y le pasamos una media y un desvío y lo redondeamos a dos; pero mientras ese valor aleatorio que generamos sea mayor al máximo o menor al mínimo vamos a seguir generando otro.

In [13]:

```
def aleatorio_rango(media, desvio, minimo, maximo):
    valor_aleatorio = round(random.normalvariate(media, desvio),
2)
    while valor_aleatorio < minimo or valor_aleatorio > maximo:
        valor_aleatorio = round(random.normalvariate(media,
desvio), 2)
    return valor_aleatorio
```

El siguiente paso consiste en recorrer el DataFrame modificable de manera iterativa, extrayendo los departamentos uno por uno y comparándolos con los departamentos presentes en el DataFrame de métricas y se obtienen las métricas asociadas, tales como la media, el desvío, el mínimo y el máximo. Luego, se genera un valor aleatorio para cada uno y se aplica la fórmula correspondiente para obtener el valor calculado. Finalmente, se almacena el valor obtenido en el DataFrame modificable.

```
In [ ]: for index, row in df_modificable.iterrows():
    departamento = row['departamento']
    df1_departamento = metricas[metricas['departamento'] ==
departamento]
    media = df1_departamento['kilos_animal'].iloc[0]
    desvio = df1_departamento['kilos_animal'].iloc[1]
    minimo = df1_departamento['kilos_animal'].iloc[2]
    maximo = df1_departamento['kilos_animal'].iloc[3]
    valor_aleatorio = round(aleatorio_rango(media, desvio, minimo,
maximo), 2)
    total_ovino = row['total_ovinos']
    valor_calculado = round(valor_aleatorio*total_ovino,2)
    df_modificable.loc[index, 'kilos_lana'] = valor_calculado
```

Como resultado obtenemos un nuevo Data Frame con los nuevos valores para cada los kilos de lana.

```
In [15]: df_modificable
```

```
Out[15]:
```

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792
5	2017-12-31	CHUBUT	MODIFICABLE	BIEDMA	3900	71790	32259	34024	3788
6	2018-12-31	CHUBUT	MODIFICABLE	BIEDMA	4070	70696	32886	34339	3911
7	2019-12-31	CHUBUT	MODIFICABLE	BIEDMA	3606	70938	32085	33311	3858
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

86 rows × 19 columns

Lo siguiente es concatenar los dos dataset: el de datos reales y el de datos modificables para obtener un nuevo Dataset actualizado

```
In [16]: df_actualizado = pd.concat([df_real, df_modificable])
df_actualizado
```

```
Out[16]:
```

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
0	2012-12-31	CHUBUT	REAL	BIEDMA	2740	52320	22319	25489	2951
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	3690
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	3492
9	2006-12-31	CHUBUT	REAL	CUSHAMEN	5218	94823	31143	21497	4900
10	2007-12-31	CHUBUT	REAL	CUSHAMEN	4603	83411	27453	19643	4729
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3000

164 rows × 19 columns

Luego ordenamos el Data frame por el índice.

```
In [17]: df_ordenado = df_actualizado.sort_index()
df_ordenado
```

```
Out[17]:
```

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
0	2012-12-31	CHUBUT	REAL	BIEDMA	2740	52320	22319	25489	2951
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	3690
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	3492
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

164 rows × 19 columns

Y finalmente guardamos el resultado en un archivo csv.

In [18]:

```
df_ordenado.to_csv("dataset_aplicado_%_lana.csv", index=False)
```

1.12. Aplicar métricas de lana: finura y rinde

In [1]:

```
import numpy as np
import pandas as pd
import random
```

Después de generar el nuevo archivo con las nuevas métricas para generar los valores, cargamos este archivo csv para convertirlo en un DataFrame y trabajarlo en combinación con el otro más adelante.

In [2]:

```
metricas = pd.read_csv("nuevo_porcentajes_f_r.csv")
metricas
```

Out[2]:

	departamento	metrica	finura	rinde
0	BIEDMA	mean	19.510000	56.930000
1	BIEDMA	std	0.410122	2.729432
2	BIEDMA	min	19.220000	55.000000
3	BIEDMA	max	19.800000	58.860000
4	CUSHAMEN	mean	19.890000	60.410000
5	CUSHAMEN	std	0.212132	4.157788
6	CUSHAMEN	min	19.740000	57.470000
7	CUSHAMEN	max	20.040000	63.350000
8	FLORENTINO AMEGHINO	mean	19.415000	59.060000
9	FLORENTINO AMEGHINO	std	0.162635	1.852620
10	FLORENTINO AMEGHINO	min	19.300000	57.750000
11	FLORENTINO AMEGHINO	max	19.530000	60.370000
12	FUTALEUFU	mean	20.935000	63.515000
13	FUTALEUFU	std	0.940452	0.544472
14	FUTALEUFU	min	20.270000	63.130000
15	FUTALEUFU	max	21.600000	63.900000
16	GAIMAN	mean	20.595000	55.630000
17	GAIMAN	std	0.289914	0.806102
18	GAIMAN	min	20.390000	55.060000
19	GAIMAN	max	20.800000	56.200000
20	LANGUIÑEO	mean	19.930000	58.170000
21	LANGUIÑEO	std	0.098995	3.917372
22	LANGUIÑEO	min	19.860000	55.400000
23	LANGUIÑEO	max	20.000000	60.940000
24	PASO DE INDIOS	mean	20.060000	53.395000
25	PASO DE INDIOS	std	0.197990	1.124300

	departamento	metrica	finura	rinde
26	PASO DE INDIOS	min	19.920000	52.600000
27	PASO DE INDIOS	max	20.200000	54.190000
28	RAWSON	mean	20.040000	58.445000
29	RAWSON	std	0.367696	3.924443
30	RAWSON	min	19.780000	55.670000
31	RAWSON	max	20.300000	61.220000
32	RIO SENGUERR	mean	20.175000	57.435000
33	RIO SENGUERR	std	0.106066	2.227386
34	RIO SENGUERR	min	20.100000	55.860000
35	RIO SENGUERR	max	20.250000	59.010000
36	SARMIENTO	mean	20.325000	48.670000
37	SARMIENTO	std	0.572756	0.650538
38	SARMIENTO	min	19.920000	48.210000
39	SARMIENTO	max	20.730000	49.130000
40	TEHUELCHES	mean	19.730000	60.640000
41	TEHUELCHES	std	0.452548	0.806102
42	TEHUELCHES	min	19.410000	60.070000
43	TEHUELCHES	max	20.050000	61.210000
44	TELSEN	mean	19.900000	52.955000
45	TELSEN	std	0.155563	1.491995
46	TELSEN	min	19.790000	51.900000
47	TELSEN	max	20.010000	54.010000

Lo siguiente es cargar el nuevo Dataset de datos productivos y el clima ajustados con los nuevos valores obtenidos de las estadísticas ganaderas de Chubut, y con una nueva columna de estado para saber cuáles son las filas que deben ser modificadas.

```
In [25]: datos = pd.read_csv("actualizando_dataset - Rinde - Finura.csv")
datos
```

```
Out[25]:
```

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	2740	52320	22319	25489	2951
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	3690
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	3492
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792
...	...	...	...	...	...	...	...	...	...

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

164 rows × 19 columns

Ahora vamos a separar este DataFrame en las filas que son modificables y las que no, obteniendo como resultado un DataFrame de datos modificables y un DataFrame de datos reales, que no tienen que ser modificados.

```
In [26]: df_real = datos[datos['estado'] == 'REAL']
df_real
```

Out[26]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	36961	
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	34926	
16	2013-12-31	CHUBUT	REAL	CUSHAMEN	4751	85946	26255	17619	41451	
17	2014-12-31	CHUBUT	REAL	CUSHAMEN	4198	77263	26512	18975	46290	
28	2013-12-31	CHUBUT	REAL	FLORENTINO AMEGHINO	9114	150199	71752	55841	95320	
29	2014-12-31	CHUBUT	REAL	FLORENTINO AMEGHINO	8893	140928	56282	51459	89550	
43	2013-12-31	CHUBUT	REAL	FUTALEUFU	3594	55861	14255	14340	24160	
44	2014-12-31	CHUBUT	REAL	FUTALEUFU	3436	51384	13783	11841	28919	
58	2013-12-31	CHUBUT	REAL	GAIMAN	4102	87337	20059	25098	31970	
59	2014-12-31	CHUBUT	REAL	GAIMAN	4936	90039	27772	26376	50701	
67	2013-12-31	CHUBUT	REAL	LANGUIÑEO	7275	96402	45552	42406	49046	
68	2015-12-31	CHUBUT	REAL	LANGUIÑEO	6151	87640	38874	40929	47517	
79	2013-12-31	CHUBUT	REAL	PASO DE INDIOS	4653	85441	26190	32246	32703	

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas	tota
80	2014-12-31	CHUBUT	REAL	PASO DE INDIOS	5398	89998	30804	37250		44180
100	2013-12-31	CHUBUT	REAL	RAWSON	1673	26143	8701	10029		11832
101	2014-12-31	CHUBUT	REAL	RAWSON	1446	21913	7450	8839		11982
116	2013-12-31	CHUBUT	REAL	RIO SENGUERR	17403	295275	124490	96655		155537
117	2014-12-31	CHUBUT	REAL	RIO SENGUERR	15571	246882	90683	83406		123913
131	2013-12-31	CHUBUT	REAL	SARMIENTO	5978	85638	26288	30644		32877
132	2014-12-31	CHUBUT	REAL	SARMIENTO	5269	80123	27608	32012		37346
144	2013-12-31	CHUBUT	REAL	TEHUELCHES	13577	177588	78690	85677		93487
145	2014-12-31	CHUBUT	REAL	TEHUELCHES	13819	176838	75419	90371		86049
156	2013-12-31	CHUBUT	REAL	TELSEN	3899	62509	23816	30940		22378
157	2014-12-31	CHUBUT	REAL	TELSEN	3495	59018	22886	22489		30400

In [27]:

```
df_modificable = datos[datos['estado'] == 'MODIFICABLE']
df_modificable
```

Out[27]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderas	tota
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	2740	52320	22319	25489		2951
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296		3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038		3792
5	2017-12-31	CHUBUT	MODIFICABLE	BIEDMA	3900	71790	32259	34024		3788
6	2018-12-31	CHUBUT	MODIFICABLE	BIEDMA	4070	70696	32886	34339		3911
...	...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728		3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606		3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979		2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273		2941

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

140 rows × 19 columns

Creamos una lista con los nombres de las categorías de lana.

```
In [7]: categorias_lana = ['finura', 'rinde']
```

Luego creamos una función `aleatorio_rango` que recibe una media, un desvío, un mínimo y un máximo lo que hacemos es utilizar el `random.normalvariate` de python y le pasamos una media y un desvío y lo redondeamos a dos; pero mientras ese valor aleatorio que generamos sea mayor al máximo o menor al mínimo vamos a seguir generando otro.

```
In [9]: def aleatorio_rango(media, desvio, minimo, maximo):
    valor_aleatorio = round(random.normalvariate(media, desvio),
2)
    while valor_aleatorio < minimo or valor_aleatorio > maximo:
        valor_aleatorio = round(random.normalvariate(media,
desvio), 2)
    return valor_aleatorio
```

El siguiente paso consiste en recorrer el DataFrame modificable de manera iterativa, extrayendo los departamentos uno por uno y comparándolos con los departamentos presentes en el DataFrame de métricas. A continuación, se recorren las diferentes categorías correspondientes a la finura y al rinde y se obtienen las métricas asociadas, tales como la media, el desvío, el mínimo y el máximo. Luego, se genera un valor aleatorio para cada categoría y se aplica la fórmula correspondiente para obtener el valor calculado. Finalmente, se almacena el valor obtenido en el DataFrame modificable.

```
In [ ]: for index, row in df_modificable.iterrows():
    departamento = row['departamento']
    df1_departamento = metricas[metricas['departamento'] ==
departamento]
    for categoria in categorias_lana:
        media = df1_departamento[categoria].iloc[0]
        desvio = df1_departamento[categoria].iloc[1]
        minimo = df1_departamento[categoria].iloc[2]
        maximo = df1_departamento[categoria].iloc[3]
        valor_aleatorio = aleatorio_rango(media, desvio, minimo,
maximo)
        df_modificable.loc[index, categoria] = valor_aleatorio
```

Como resultado obtenemos un nuevo Data Frame con los nuevos valores para la finura y el rinde.

```
In [29]: df_modificable
```

Out[29]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	2740	52320	22319	25489	2951
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792
5	2017-12-31	CHUBUT	MODIFICABLE	BIEDMA	3900	71790	32259	34024	3788
6	2018-12-31	CHUBUT	MODIFICABLE	BIEDMA	4070	70696	32886	34339	3911
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

140 rows × 19 columns

Lo siguiente es concatenar los dos dataset: el de datos reales y el de datos modificables para obtener un nuevo Dataset actualizado.

In [30]:

```
df_actualizado = pd.concat([df_real, df_modificable])
df_actualizado
```

Out[30]:

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	364
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	344
16	2013-12-31	CHUBUT	REAL	CUSHAMEN	4751	85946	26255	17619	414
17	2014-12-31	CHUBUT	REAL	CUSHAMEN	4198	77263	26512	18975	466
28	2013-12-31	CHUBUT	REAL	FLORENTINO AMEGHINO	9114	150199	71752	55841	951
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corde
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	28793
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	29418
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	30068

164 rows × 19 columns

Luego ordenamos el Data frame por el índice.

```
In [31]: df_ordenado = df_actualizado.sort_index()
df_ordenado
```

```
Out[31]:
```

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corde
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	2740	52320	22319	25489	29518
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	36906
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	34923
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	38111
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	37923
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	31888
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	31988
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	28223
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	29418
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	30068

164 rows × 19 columns

Y finalmente guardamos el resultado en un archivo csv.

```
In [32]: df_ordenado.to_csv("dataset_aplicado_%_lana_f_r.csv", index=False)
```

1.13 Relacionar dataset

Ahora es momento de relacionar los dataset, por lo tanto comencemos cargando ambos

In [16]:

```
import pandas as pd
import random
pd.options.mode.chained_assignment = None
```

In [52]:

```
lana_sucia_chubut =
pd.read_csv('datos_producidos/lana_chubut_2000-2022.csv', sep=",")
lana_sucia_chubut
```

Out[52]:

	anio	departamento	kg_lana	finura	rinde
0	2000-01-01	BIEDMA	793900.00	20.39	59.21
1	2000-01-01	CUSHAMEN	892500.00	19.10	61.65
2	2000-01-01	ESCALANTE	1086300.00	21.90	49.99
3	2000-01-01	FLORENTINO AMEGHINO	1463700.00	19.61	60.32
4	2000-01-01	FUTALEUFU	557600.00	20.57	63.37
...	...	...	...	...	...
363	2022-01-01	RIO SENGUERR	1977589.41	19.72	58.07
364	2022-01-01	SARMIENTO	728889.55	20.94	49.20
365	2022-01-01	TEHUELCHES	1628738.94	18.47	60.56
366	2022-01-01	TELSEN	525585.97	19.11	55.35
367	2022-01-01	TOTAL	11551340.00	21.82	58.79

368 rows × 5 columns

In [3]:

```
existencias_chubut =
pd.read_csv('datos_producidos/existencias_chubut_2008_2020_SENASA.csv',
sep=",")
existencias_chubut
```

Out[3]:

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
0	2008	CHUBUT	7686	121561	58576	69633	33876	291332	BIEDMA
1	2008	CHUBUT	8180	149189	41669	51128	37589	287755	CUSHAMEN
2	2008	CHUBUT	7863	109225	35581	40786	41516	234971	ESCALANTE
3	2008	CHUBUT	13875	164382	79863	63856	48809	370785	FLORENTINO AMEGHINO
4	2008	CHUBUT	5639	90494	30119	24634	28237	179123	FUTALEUFU
...	...	...	...	...	...	...	...	...	...

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
203	2020	CHUBUT	15609	360905	146961	123560	112227	759262	RIO SENGUERR
204	2020	CHUBUT	3681	99173	27021	32939	20498	183312	SARMIENTO
205	2020	CHUBUT	8071	197382	87268	86893	54493	434107	TEHUELCHES
206	2020	CHUBUT	2870	79157	25804	36901	18984	163716	TELSEN
207	2020	CHUBUT	79860	1992008	669777	666432	576856	3984933	TOTAL

208 rows × 9 columns

Se va a transformar las columnas que representan datos de tiempo a datos de tipo fecha

In [56]:

```
lana_sucia_chubut['anio'] =
pd.to_datetime(lana_sucia_chubut['anio'], format="%Y-%m-%d")
lana_sucia_chubut
```

Out[56]:

	anio	departamento	kg_lana	finura	rinde
0	2000-01-01	BIEDMA	793900.00	20.39	59.21
1	2000-01-01	CUSHAMEN	892500.00	19.10	61.65
2	2000-01-01	ESCALANTE	1086300.00	21.90	49.99
3	2000-01-01	FLORENTINO AMEGHINO	1463700.00	19.61	60.32
4	2000-01-01	FUTALEUFU	557600.00	20.57	63.37
...	...	...	...	...	...
363	2022-01-01	RIO SENGUERR	1977589.41	19.72	58.07
364	2022-01-01	SARMIENTO	728889.55	20.94	49.20
365	2022-01-01	TEHUELCHES	1628738.94	18.47	60.56
366	2022-01-01	TELSEN	525585.97	19.11	55.35
367	2022-01-01	TOTAL	11551340.00	21.82	58.79

368 rows × 5 columns

In [9]:

```
existencias_chubut['anio'] =
pd.to_datetime(existencias_chubut['anio'], format="%Y")
existencias_chubut
```

Out[9]:

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departament
0	2008-01-01	CHUBUT	7686	121561	58576	69633	33876	291332	BIEDMA
1	2008-01-01	CHUBUT	8180	149189	41669	51128	37589	287755	CUSHAMEN
2	2008-01-01	CHUBUT	7863	109225	35581	40786	41516	234971	ESCALANTI
3	2008-01-01	CHUBUT	13875	164382	79863	63856	48809	370785	FLORENTINO AMEGHINO

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departament
4	2008-01-01	CHUBUT	5639	90494	30119	24634	28237	179123	FUTALEUFU
...	...	...	...	...	...	...	...	...	...
203	2020-01-01	CHUBUT	15609	360905	146961	123560	112227	759262	RIC SENGUERI
204	2020-01-01	CHUBUT	3681	99173	27021	32939	20498	183312	SARMIENTO
205	2020-01-01	CHUBUT	8071	197382	87268	86893	54493	434107	TEHUELCHES
206	2020-01-01	CHUBUT	2870	79157	25804	36901	18984	163716	TELSEN
207	2020-01-01	CHUBUT	79860	1992008	669777	666432	576856	3984933	TOTAL

208 rows × 9 columns

Se van a juntar los dos dataset

In [10]:

```
merged_inner =
pd.merge(left=existencias_chubut,right=lana_sucia_chubut)
merged_inner
```

Out[10]:

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departament
0	2008-01-01	CHUBUT	7686	121561	58576	69633	33876	291332	BIEDMA
1	2008-01-01	CHUBUT	8180	149189	41669	51128	37589	287755	CUSHAMEN
2	2008-01-01	CHUBUT	7863	109225	35581	40786	41516	234971	ESCALANTI
3	2008-01-01	CHUBUT	13875	164382	79863	63856	48809	370785	FLORENTINO AMEGHINO
4	2008-01-01	CHUBUT	5639	90494	30119	24634	28237	179123	FUTALEUFU
...	...	...	...	...	...	...	...	...	...
203	2020-01-01	CHUBUT	15609	360905	146961	123560	112227	759262	RIC SENGUERI
204	2020-01-01	CHUBUT	3681	99173	27021	32939	20498	183312	SARMIENTO
205	2020-01-01	CHUBUT	8071	197382	87268	86893	54493	434107	TEHUELCHES
206	2020-01-01	CHUBUT	2870	79157	25804	36901	18984	163716	TELSEN
207	2020-01-01	CHUBUT	79860	1992008	669777	666432	576856	3984933	TOTAL

208 rows × 10 columns

En este punto nos parece una pena desperdiciar 8 años de datos laneros, por lo tanto se van a tomar las existencias desde el año 2000 al 2007 del dataset que posee datos de las existencias ovinas de Chubut desde el año 1965 al 2017.

Observamos el contenido del dataset de existencias 1965-2017

```
In [4]: existencias_chubut_1 =
pd.read_csv('datos_producidos/existencias_chubut_1965-2017.csv',
sep=",")
existencias_chubut_1
```

Out[4]:	provincia	desde_año	hasta_año	cantidad	fuerce
0	Chubut	1965	1965	5736.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
1	Chubut	1966	1966	5789.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
2	Chubut	1967	1967	5841.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
3	Chubut	1968	1968	5894.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
4	Chubut	1969	1970	5947.0	Censos INDEC
5	Chubut	1971	1971	5887.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
6	Chubut	1972	1972	5826.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
7	Chubut	1973	1973	5766.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
8	Chubut	1974	1975	5705.0	Censos INDEC
9	Chubut	1976	1976	6063.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
10	Chubut	1977	1978	6421.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
11	Chubut	1979	1979	6287.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
12	Chubut	1980	1980	6153.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
13	Chubut	1981	1981	6020.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
14	Chubut	1982	1982	5886.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
15	Chubut	1983	1983	5752.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
16	Chubut	1984	1984	5618.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
17	Chubut	1985	1985	5484.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
18	Chubut	1986	1986	5351.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
19	Chubut	1987	1987	5217.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
20	Chubut	1988	1988	5083.0	Censos INDEC
21	Chubut	1989	1989	5034.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
22	Chubut	1990	1990	4986.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
23	Chubut	1991	1991	4937.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
24	Chubut	1992	1992	4889.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
25	Chubut	1993	1993	4840.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
26	Chubut	1994	1994	4647.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
27	Chubut	1995	1995	4031.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
28	Chubut	1996	1996	4044.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...

	provincia	desde_año	hasta_año	cantidad	fuelle
29	Chubut	1997	1997	3222.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
30	Chubut	1998	1998	3650.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
31	Chubut	1999	1999	4078.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
32	Chubut	2000	2000	4017.0	Encuesta Nacional Agropecuaria 93- 94- 95 y 01...
33	Chubut	2001	2001	3944.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
34	Chubut	2002	2002	3871.0	Censos INDEC
35	Chubut	2003	2003	4118.0	Estimación Dto. Ovinos y Lanas - SAGPyA matema...
36	Chubut	2004	2004	4365.0	Estimación Dto. Ovinos y Lanas - SAGPyA en bas...
37	Chubut	2005	2005	4456.0	SIGSA - Dirección de Control de Gestión y Prog...
38	Chubut	2006	2006	4548.0	SIGSA - Dirección de Control de Gestión y Prog...
39	Chubut	2007	2007	4640.0	SIGSA - Dirección de Control de Gestión y Prog...
40	Chubut	2008	2008	4879.0	SIGSA - Dirección de Control de Gestión y Prog...
41	Chubut	2009	2009	4742.0	SIGSA - Dirección de Control de Gestión y Prog...
42	Chubut	2010	2010	4603.0	SIGSA - Dirección de Control de Gestión y Prog...
43	Chubut	2011	2011	4439.0	SIGSA - Dirección de Control de Gestión y Prog...
44	Chubut	2012	2012	4163.0	SIGSA - Dirección de Control de Gestión y Prog...
45	Chubut	2013	2013	4032.0	SIGSA - Dirección de Control de Gestión y Prog...
46	Chubut	2014	2014	4024.0	SIGSA - Dirección de Control de Gestión y Prog...
47	Chubut	2015	2015	3994.0	SIGSA - Dirección de Control de Gestión y Prog...
48	Chubut	2016	2016	4002.0	SIGSA - Dirección de Control de Gestión y Prog...
49	Chubut	2017	2017	3973.0	SIGSA - Dirección de Control de Gestión y Prog...

In [5]: `existencias_chubut_1.columns`

Out[5]: `Index(['provincia', 'desde_año', 'hasta_año', 'cantidad', 'fuente'], dtype='object')`

Eliminamos las columnas que no necesitamos, transformamos la del tiempo en el tipo de dato datetime, seleccionamos la columna de tiempo como índice y luego filtramos para quedarnos con los datos del 2000-2007.

```
In [6]: existencias_chubut_1.drop(['provincia', 'hasta_año', 'fuente'],
axis = 'columns', inplace=True)
existencias_chubut_1['desde_año'] =
pd.to_datetime(existencias_chubut_1['desde_año'],
format="%Y").dt.normalize()
existencias_chubut_1 =
existencias_chubut_1.set_index(['desde_año'])
existencias_chubut_filtrada=existencias_chubut_1.loc['2000-01-
01':'2007-01-01']
existencias_chubut_filtrada
```


Out[6]:

	cantidad
desde_año	
2000-01-01	4017.0
2001-01-01	3944.0
2002-01-01	3871.0
2003-01-01	4118.0
2004-01-01	4365.0
2005-01-01	4456.0
2006-01-01	4548.0
2007-01-01	4640.0

Se transformaron los datos de la cantidad de toneladas a kilogramos.

In [7]:

```
existencias_chubut_filtrada['cantidad'] =  
existencias_chubut_filtrada['cantidad'].apply(lambda x : x * 1000)  
existencias_chubut_filtrada
```

Out[7]:

	cantidad
desde_año	
2000-01-01	4017000.0
2001-01-01	3944000.0
2002-01-01	3871000.0
2003-01-01	4118000.0
2004-01-01	4365000.0
2005-01-01	4456000.0
2006-01-01	4548000.0
2007-01-01	4640000.0

Para poder completar bien la filas faltantes seria interesante saber el porcentaje de corderos, carneros, ovejas, etc, que cada departamento tenia en base a su total de ovinos cada año.

Entonces utilizamos el dataset de existencias de SENASA para obtener los totales anuales.

In [8]:

```
existencias_anuales =  
existencias_chubut[existencias_chubut.departamento.eq('TOTAL')]  
existencias_anuales
```

Out[8]:

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
15	2008	CHUBUT	135075	2160564	885807	931734	766298	4879478	TOTAL
31	2009	CHUBUT	135414	2182913	857329	888470	678003	4742129	TOTAL
47	2010	CHUBUT	131249	2202419	777797	879509	612109	4603083	TOTAL
63	2011	CHUBUT	132272	2119111	762246	844921	580441	4438991	TOTAL

	anio	provincia	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	departamento
79	2012	CHUBUT	126443	1998999	689806	792843	555331	4163422	TOTAL
95	2013	CHUBUT	133292	1967599	649453	725939	555396	4031679	TOTAL
111	2014	CHUBUT	128053	1941873	643744	701889	608489	4024048	TOTAL
127	2015	CHUBUT	131095	1939450	637096	694326	600189	4002156	TOTAL
143	2016	CHUBUT	130635	1947672	658063	703023	554340	3993733	TOTAL
159	2017	CHUBUT	124046	1917298	652423	672122	606862	3972751	TOTAL
175	2018	CHUBUT	120807	1929004	643675	656187	595247	3944920	TOTAL
191	2019	CHUBUT	127625	1996422	681206	673858	597226	4076337	TOTAL
207	2020	CHUBUT	79860	1992008	669777	666432	576856	3984933	TOTAL

```
In [9]: totales_chubut = []
for i in existencias_anuales.index:
    totales_chubut.append(
        {'anio': existencias_anuales["anio"][i],
         'total_ovinos': existencias_anuales["total_ovinos"][i]})

totales_chubut
```

```
Out[9]: [{'anio': 2008, 'total_ovinos': 4879478},
{'anio': 2009, 'total_ovinos': 4742129},
{'anio': 2010, 'total_ovinos': 4603083},
{'anio': 2011, 'total_ovinos': 4438991},
{'anio': 2012, 'total_ovinos': 4163422},
{'anio': 2013, 'total_ovinos': 4031679},
{'anio': 2014, 'total_ovinos': 4024048},
{'anio': 2015, 'total_ovinos': 4002156},
{'anio': 2016, 'total_ovinos': 3993733},
{'anio': 2017, 'total_ovinos': 3972751},
{'anio': 2018, 'total_ovinos': 3944920},
{'anio': 2019, 'total_ovinos': 4076337},
{'anio': 2020, 'total_ovinos': 3984933}]
```

Lo siguiente es recorrer todos los departamentos de chubut y por cada departamento recorrer los totales que obtuvimos antes, luego en base al total de ovinos de departamento (total_ovinos) calculamos el porcentaje de corderos, carneros, etc, y por ultimo en base al total de ovinos de chubut calculamos el porcentaje de ovinos que representa eso para el departamento. De esta manera obtenemos un dataset de porcentajes de Chubut.

```
In [10]: departamentos_chubut = ["BIEDMA", "CUSHAMEN", "ESCALANTE",
                                "FLORENTINO AMEGHINO", "FUTALEUFU", "GAIMAN", "GASTRE",
                                "LANGUIÑEO", "MARTIRES", "PASO DE INDIOS",
                                "RAWSON", "RIO SENGUERR",
                                "SARMIENTO", "TEHUELCHES", "TELSEN"]
```

```
In [11]: columnas = ['anio', 'provincia', 'departamento', 'carneros',  
                'ovejas', 'borregos', 'caponos',  
                'corderos_corderas', 'total_ovinos']
```

```
In [12]: def calcular_porcentaje(valor, total):  
        return round(valor*100/total, 2)  
  
def calcular_promedio(columna):  
    return round(columna.mean)
```

```
In [13]: porcentajes_dpto = []  
  
for dpto in departamentos_chubut:  
    datos_dpto =  
existencias_chubut[existencias_chubut.departamento.eq(dpto)]  
    for total in totales_chubut:  
        datos_dpto_anio =  
datos_dpto[datos_dpto.anio.eq(total['anio'])]  
        total_ovinos = datos_dpto_anio.iloc[0]['total_ovinos']  
        datos = datos_dpto_anio.iloc[0]  
        porcentaje_carneros = calcular_porcentaje(  
            datos['carneros'], total_ovinos)  
        porcentaje_ovejas = calcular_porcentaje(datos['ovejas'],  
total_ovinos)  
        porcentaje_borregos = calcular_porcentaje(  
            datos['borregos'], total_ovinos)  
        porcentaje_caponos = calcular_porcentaje(  
            datos['caponos'], total_ovinos)  
        porcentaje_corderos_corderas = calcular_porcentaje(  
            datos['corderos_corderas'], total_ovinos)  
        porcentaje_total = calcular_porcentaje(  
            datos['total_ovinos'], total['total_ovinos'])  
        datos_porcentaje = [datos['anio'],  
                             datos['provincia'],  
                             datos['departamento'],  
                             porcentaje_carneros,  
                             porcentaje_ovejas,  
                             porcentaje_borregos,  
                             porcentaje_caponos,  
                             porcentaje_corderos_corderas,  
                             porcentaje_total]
```

```

    ]
    porcentajes_dpto.append(datos_porcentaje)

chubut_porcentajes = pd.DataFrame(porcentajes_dpto,
columns=columnas)
chubut_porcentajes

```

```

Out[13]:

```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2008	CHUBUT	BIEDMA	2.64	41.73	20.11	23.90	11.63	5.97
1	2009	CHUBUT	BIEDMA	2.94	43.91	20.44	24.06	8.65	5.73
2	2010	CHUBUT	BIEDMA	2.80	45.79	16.07	25.64	9.71	5.24
3	2011	CHUBUT	BIEDMA	2.91	45.11	16.85	25.06	10.07	4.88
4	2012	CHUBUT	BIEDMA	2.98	48.17	15.56	24.99	8.29	4.66
...	...	...	...	...	...	...	...	...	...
190	2016	CHUBUT	TELSEN	2.86	46.89	15.47	22.93	11.85	4.29
191	2017	CHUBUT	TELSEN	2.83	47.34	16.66	22.32	10.84	4.12
192	2018	CHUBUT	TELSEN	2.81	46.05	16.51	21.86	12.77	4.36
193	2019	CHUBUT	TELSEN	2.83	47.36	16.81	22.53	10.48	4.15
194	2020	CHUBUT	TELSEN	1.75	48.35	15.76	22.54	11.60	4.11

195 rows × 9 columns

```

In [48]:
chubut_porcentajes.to_csv('datos_producidos/chubut_porcentajes.csv',
index=False)

```

Para poder generar los valores faltantes de manera aleatoria pero con cierta "estructura" es necesario obtener las medidas estadísticas de los porcentajes anteriores.

```

In [14]:
def generar_valores_normales(media, desvio):
    valores = []
    for i in range(1,9):
        valores.append(round(random.normalvariate(media_dpto,
desvio_dpto),2))
    return valores

```

```

In [17]:
columnas=['departamento', 'dato', 'media',
'desvio_estandar','min', 'max', 'valor_normal_random']
columnas_numericas = ['carneros', 'ovejas', 'borregos',
'capones','corderos_corderas', 'total_ovinos']
promedios_dptos = []

for dpto in departamentos_chubut:

```

```

for columna in columnas_numericas:
    datos_dpto =
chubut_porcentajes[chubut_porcentajes.departamento.eq(dpto)]
    media_dpto = round(datos_dpto[columna].mean(),2)
    desvio_dpto = round(datos_dpto[columna].std(),2)
    min_dpto = datos_dpto[columna].min()
    max_dpto = datos_dpto[columna].max()
    valor_normal = generar_valores_normales(media_dpto,
desvio_dpto)
    promedios_dptos.append([dpto,columna,media_dpto,
desvio_dpto, min_dpto, max_dpto, valor_normal])
chubut_metricas = pd.DataFrame(promedios_dptos, columns=columnas)
chubut_metricas

```

Out[17]:

	departamento	dato	media	desvio_estandar	min	max	valor_normal_random
0	BIEDMA	carneros	2.64	0.32	1.77	2.98	[2.2, 2.07, 3.18, 2.64, 2.87, 2.87, 2.5, 2.91]
1	BIEDMA	ovejas	46.08	1.97	41.73	48.47	[48.03, 46.51, 43.87, 45.15, 48.17, 49.4, 46.7...
2	BIEDMA	borregos	17.41	2.12	13.49	20.44	[18.91, 15.87, 23.51, 17.83, 17.11, 16.06, 16....
3	BIEDMA	capones	21.57	3.19	17.31	25.64	[19.48, 21.17, 22.2, 21.2, 19.1, 23.97, 16.87,...
4	BIEDMA	corderos_corderas	12.29	2.74	8.29	16.92	[14.75, 11.79, 14.1, 14.76, 11.24, 13.95, 14.2...
...	...	...	...	...	...	...	...
85	TELSEN	ovejas	47.24	1.42	43.76	49.50	[44.65, 47.42, 48.8, 47.08, 46.62, 47.13, 45.3...
86	TELSEN	borregos	15.41	1.91	12.57	19.31	[14.04, 16.07, 11.78, 16.46, 10.1, 15.17, 14.7...
87	TELSEN	capones	25.02	2.42	21.86	28.90	[26.97, 24.36, 23.82, 26.74, 25.75, 23.43, 26....
88	TELSEN	corderos_corderas	9.68	2.19	6.58	12.77	[6.92, 11.71, 8.6, 9.55, 6.08, 7.08, 9.29, 8.57]
89	TELSEN	total_ovinos	5.15	1.29	4.11	8.15	[3.47, 3.63, 4.0, 5.89, 7.89, 4.09, 6.27, 5.87]

90 rows × 7 columns

In [67]:

```

chubut_metricas.to_csv('datos_producidos/chubut_metricas.csv',
index=False)

```

Obviamente al usar valores aleatorios la suma de los porcentajes generados no da siempre 100, por lo tanto es necesario hacer un ajuste. Veamos el ejemplo con uno de los datos. Primero obtengamos los datos del total de ovinos de cada departamento.

```
In [18]: total_ovinos_dpto = chubut_metricas.loc[chubut_metricas.loc[:,
'dato'] == 'total_ovinos']
total_ovinos_dpto
```

```
Out[18]:
```

	departamento	dato	media	desvio_estandar	min	max	valor_normal_random
5	BIEDMA	total_ovinos	4.82	0.52	4.37	5.97	[4.89, 5.79, 5.04, 4.66, 3.63, 5.24, 5.77, 4.37]
11	CUSHAMEN	total_ovinos	5.61	0.29	5.13	6.13	[6.28, 5.99, 5.35, 5.52, 5.87, 5.74, 5.46, 5.49]
17	ESCALANTE	total_ovinos	4.91	0.21	4.68	5.32	[5.12, 5.01, 5.08, 4.67, 5.13, 4.99, 4.7, 5.59]
23	FLORENTINO AMEGHINO	total_ovinos	7.61	0.40	7.01	8.25	[6.88, 7.89, 7.06, 7.99, 7.04, 6.44, 7.7, 7.11]
29	FUTALEUFU	total_ovinos	3.68	0.49	2.89	4.33	[3.54, 3.69, 4.15, 3.04, 3.99, 3.29, 4.47, 3.89]
35	GAIMAN	total_ovinos	6.25	1.27	4.36	8.00	[5.58, 6.08, 6.56, 5.3, 7.46, 6.79, 6.37, 7.09]
41	GASTRE	total_ovinos	6.05	0.49	5.36	6.84	[7.01, 5.6, 7.08, 6.57, 6.14, 7.16, 5.03, 6.19]
47	LANGUINËO	total_ovinos	6.62	0.93	5.44	8.31	[6.53, 7.72, 6.99, 7.21, 4.78, 6.21, 6.98, 7.52]
53	MARTIRES	total_ovinos	4.11	0.22	3.71	4.55	[4.28, 4.27, 4.2, 3.91, 3.79, 3.93, 4.08, 4.19]
59	PASO DE INDIOS	total_ovinos	6.98	0.38	6.51	7.67	[6.77, 7.32, 6.76, 6.73, 6.86, 7.0, 6.89, 7.25]
65	RAWSON	total_ovinos	2.87	0.48	2.06	3.41	[3.03, 2.94, 3.23, 2.78, 3.5, 2.86, 3.02, 2.83]
71	RIO SENGUERR	total_ovinos	18.81	1.67	15.36	20.69	[18.45, 19.62, 19.22, 17.46, 21.19, 18.6, 18.0...
77	SARMIENTO	total_ovinos	4.85	0.30	4.43	5.38	[5.29, 4.66, 5.21, 4.89, 4.81, 4.52, 5.18, 4.63]
83	TEHUELCHES	total_ovinos	11.67	0.50	10.89	12.50	[11.48, 11.16, 11.63, 11.6, 12.31, 11.51, 12.1...
89	TELSEN	total_ovinos	5.15	1.29	4.11	8.15	[3.47, 3.63, 4.0, 5.89, 7.89, 4.09, 6.27, 5.87]

Ahora agrupamos los datos por año

```
In [42]: anio_2000 = []
anio_2001 = []
anio_2002 = []
anio_2003 = []
anio_2004 = []
anio_2005 = []
anio_2006 = []
anio_2007 = []
anios = []
```

```
for valor_normal in total_ovinos_dpto['valor_normal_random']:
    anio_2000.append(valor_normal[0])
    anio_2001.append(valor_normal[1])
    anio_2002.append(valor_normal[2])
    anio_2003.append(valor_normal[3])
    anio_2004.append(valor_normal[4])
    anio_2005.append(valor_normal[5])
    anio_2006.append(valor_normal[6])
    anio_2007.append(valor_normal[7])
```

Y miramos el ajuste del año 2001

In [44]:

```
suma = round(sum(anio_2001),2)
print("Antes del ajuste", suma)
resto=0
if suma>100:
    resto = suma - 100
elif suma<100:
    resto = -(100 - suma)
ajuste = round(resto / 15, 4)
anio_2001_ajustado = list(map(lambda x: round(x - ajuste,2),
    anio_2001))
suma = round(sum(anio_2001_ajustado),0)
print("Despues del ajuste", suma)
```

Antes del ajuste 101.37
Despues del ajuste 100.0

De esta manera se fue realizando el ajuste, pero a partir de acá se trabajo el dataset de manera manual con Excel y se obtuvo lo siguiente:

In [50]:

```
existencias_chubut_2 =
pd.read_csv('datos_producidos/existencias_chubut_2000_2007.csv',
    sep=",")
existencias_chubut_2
```

Out[50]:

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2000	CHUBUT	BIEDMA	4627	88905	31529	41328	22410	188799
1	2000	CHUBUT	CUSHAMEN	5483	112188	32417	25337	34665	210089
2	2000	CHUBUT	ESCALANTE	5115	99373	30290	39530	36183	210491
3	2000	CHUBUT	FLORENTINO AMEGHINO	11162	146253	58737	45221	40304	301677
4	2000	CHUBUT	FUTALEUFU	4342	69056	20947	14924	18069	127339
...	...	...	...	...	...	...	...	...	...

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
123	2007	CHUBUT	RIO SENGUERR	24501	428453	140769	142907	154311	890940
124	2007	CHUBUT	SARMIENTO	6357	107240	30052	39893	27644	211185
125	2007	CHUBUT	TEHUELCHES	12118	213731	95550	102023	94462	517884
126	2007	CHUBUT	TELSEN	5281	92040	29246	54972	18504	200044
127	2007	CHUBUT	TOTAL	119166	2254183	763187	824700	678761	4639997

128 rows $\times$ 9 columns

Ahora vamos a combinar los dos dataset de existencias, concatenandolos.

In [55]:

```
concat_existencias =
pd.concat([existencias_chubut_2, existencias_chubut], sort=False)
concat_existencias['anio'] =
pd.to_datetime(concat_existencias['anio'], format="%Y")
concat_existencias
```

Out[55]:

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovino
0	2000-01-01	CHUBUT	BIEDMA	4627	88905	31529	41328	22410	188799
1	2000-01-01	CHUBUT	CUSHAMEN	5483	112188	32417	25337	34665	210081
2	2000-01-01	CHUBUT	ESCALANTE	5115	99373	30290	39530	36183	210491
3	2000-01-01	CHUBUT	FLORENTINO AMEGHINO	11162	146253	58737	45221	40304	301671
4	2000-01-01	CHUBUT	FUTALEUFU	4342	69056	20947	14924	18069	127331
...	...	...	...	...	...	...	...	...	...
203	2020-01-01	CHUBUT	RIO SENGUERR	15609	360905	146961	123560	112227	759261
204	2020-01-01	CHUBUT	SARMIENTO	3681	99173	27021	32939	20498	183311
205	2020-01-01	CHUBUT	TEHUELCHES	8071	197382	87268	86893	54493	434101
206	2020-01-01	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163711
207	2020-01-01	CHUBUT	TOTAL	79860	1992008	669777	666432	576856	3984931

336 rows $\times$ 9 columns

Finalmente vamos a combinar los dos dataset, para obtener la produccion de chubut durante 20 años

In [57]:

```
produccion_chubut =  
pd.merge(left=concat_existencias,right=lana_sucia_chubut)  
produccion_chubut
```

Out[57]:

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovino
0	2000-01-01	CHUBUT	BIEDMA	4627	88905	31529	41328	22410	188799
1	2000-01-01	CHUBUT	CUSHAMEN	5483	112188	32417	25337	34665	210089
2	2000-01-01	CHUBUT	ESCALANTE	5115	99373	30290	39530	36183	210499
3	2000-01-01	CHUBUT	FLORENTINO AMEGHINO	11162	146253	58737	45221	40304	301679
4	2000-01-01	CHUBUT	FUTALEUFU	4342	69056	20947	14924	18069	127339
...	...	...	...	...	...	...	...	...	...
331	2020-01-01	CHUBUT	RIO SENGUERR	15609	360905	146961	123560	112227	759269
332	2020-01-01	CHUBUT	SARMIENTO	3681	99173	27021	32939	20498	183319
333	2020-01-01	CHUBUT	TEHUELCHES	8071	197382	87268	86893	54493	434109
334	2020-01-01	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163719
335	2020-01-01	CHUBUT	TOTAL	79860	1992008	669777	666432	576856	3984939

336 rows × 12 columns

In [58]:

```
produccion_chubut.to_csv('datos_producidos/produccion_chubut_2000_2020'  
index=False)
```

1.14 Armar dataset clima + producción

Anteriormente se obtuvieron los datos de clima de cada departamento de chubut, de la existencias ovinas y de la produccion de lana. Ahora vamos a combinar todo para poder empezar a entrenar a los modelos.

```
In [1]: import pandas as pd
```

```
In [2]: produccion_chubut =  
pd.read_csv('datos_producidos/produccion_chubut_2000_2020.csv',  
sep=",")  
produccion_chubut
```

```
Out[2]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovino
0	2000-01-01	CHUBUT	BIEDMA	4627	88905	31529	41328	22410	188799
1	2000-01-01	CHUBUT	CUSHAMEN	5483	112188	32417	25337	34665	210089
2	2000-01-01	CHUBUT	ESCALANTE	5115	99373	30290	39530	36183	210499
3	2000-01-01	CHUBUT	FLORENTINO AMEGHINO	11162	146253	58737	45221	40304	301679
4	2000-01-01	CHUBUT	FUTALEUFU	4342	69056	20947	14924	18069	127339
...	...	...	...	...	...	...	...	...	...
331	2020-01-01	CHUBUT	RIO SENGUERR	15609	360905	146961	123560	112227	759269
332	2020-01-01	CHUBUT	SARMIENTO	3681	99173	27021	32939	20498	183319
333	2020-01-01	CHUBUT	TEHUELCHES	8071	197382	87268	86893	54493	434109
334	2020-01-01	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163719
335	2020-01-01	CHUBUT	TOTAL	79860	1992008	669777	666432	576856	3984939

336 rows × 12 columns

```
In [8]: clima_chubut = pd.read_csv('datos_producidos/clima_chubut.csv',  
sep=",")  
clima_chubut
```

```
Out[8]:
```

	departamento	anio	PROMEDIO_HUMEDAD	TOTAL_LLUVIA	PROMEDIO_T_MAX	PROMEDIO_T_MIN
0	BIEDMA	2012-01-01	58.40	46.4	18.10	10.10

	departamento	anio	PROMEDIO_HUMEDAD	TOTAL_LLUVIA	PROMEDIO_T_MAX	PROMEDIO_
1	BIEDMA	2013-01-01	64.70	226.5	19.90	
2	BIEDMA	2014-01-01	60.50	331.3	20.60	
3	BIEDMA	2015-01-01	58.70	114.2	22.20	
4	BIEDMA	2016-01-01	57.50	88.9	25.20	
...	...	...	...	...	...	
204	TELSEN	2018-01-01	47.93	604.2	17.63	
205	TELSEN	2019-01-01	47.57	358.4	17.53	
206	TELSEN	2020-01-01	46.63	304.4	16.60	
207	TELSEN	2021-01-01	47.03	608.9	18.97	
208	TELSEN	2022-01-01	37.15	58.7	28.50	

209 rows × 7 columns

In [9]:

```
merged_inner = pd.merge(left=produccion_chubut,right=clima_chubut)
merged_inner
```

Out[9]:

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2000-01-01	CHUBUT	RAWSON	2355	57320	20632	24405	18609	123321
1	2001-01-01	CHUBUT	RAWSON	2226	54176	19500	23067	17588	116557
2	2002-01-01	CHUBUT	RAWSON	1584	38558	13879	16417	12518	82956
3	2003-01-01	CHUBUT	RAWSON	2252	54793	19722	23330	17789	117886
4	2004-01-01	CHUBUT	RAWSON	2463	59959	21582	25529	19466	128999
...	...	...	...	...	...	...	...	...	...
160	2020-01-01	CHUBUT	RAWSON	2273	74096	18058	19267	22102	135796
161	2020-01-01	CHUBUT	RIO SENGUERR	15609	360905	146961	123560	112227	759262
162	2020-01-01	CHUBUT	SARMIENTO	3681	99173	27021	32939	20498	183312
163	2020-01-01	CHUBUT	TEHUELCHES	8071	197382	87268	86893	54493	434107
164	2020-01-01	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163716

165 rows × 17 columns

```
In [10]: merged_inner.to_csv('datos_producidos/merged_inner.csv',  
                             index=False)
```

```
In [11]: merged_left_join =  
pd.merge(left=produccion_chubut, right=clima_chubut, how='left')  
merged_left_join
```

```
Out[11]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovino
0	2000-01-01	CHUBUT	BIEDMA	4627	88905	31529	41328	22410	188799
1	2000-01-01	CHUBUT	CUSHAMEN	5483	112188	32417	25337	34665	210089
2	2000-01-01	CHUBUT	ESCALANTE	5115	99373	30290	39530	36183	21049
3	2000-01-01	CHUBUT	FLORENTINO AMEGHINO	11162	146253	58737	45221	40304	30167
4	2000-01-01	CHUBUT	FUTALEUFU	4342	69056	20947	14924	18069	127339
...	...	...	...	...	...	...	...	...	...
331	2020-01-01	CHUBUT	RIO SENGUERR	15609	360905	146961	123560	112227	759269
332	2020-01-01	CHUBUT	SARMIENTO	3681	99173	27021	32939	20498	183319
333	2020-01-01	CHUBUT	TEHUELCHES	8071	197382	87268	86893	54493	434109
334	2020-01-01	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163719
335	2020-01-01	CHUBUT	TOTAL	79860	1992008	669777	666432	576856	3984939

336 rows × 17 columns

```
In [15]: merged_left_join.to_csv('datos_producidos/merged_left_join.csv',  
                                  index=False)
```

```
In [13]: merged_full_join =  
pd.merge(left=produccion_chubut, right=clima_chubut, how='outer')  
merged_full_join
```

```
Out[13]:
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovino
0	2000-01-01	CHUBUT	BIEDMA	4627.0	88905.0	31529.0	41328.0	22410.0	188799.0

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovino
1	2000-01-01	CHUBUT	CUSHAMEN	5483.0	112188.0	32417.0	25337.0	34665.0	210089.0
2	2000-01-01	CHUBUT	ESCALANTE	5115.0	99373.0	30290.0	39530.0	36183.0	210491.0
3	2000-01-01	CHUBUT	FLORENTINO AMEGHINO	11162.0	146253.0	58737.0	45221.0	40304.0	301677.0
4	2000-01-01	CHUBUT	FUTALEUFU	4342.0	69056.0	20947.0	14924.0	18069.0	127339.0
...	...	...	...	...	...	...	...	...	...
375	2022-01-01	NaN	SARMIENTO	NaN	NaN	NaN	NaN	NaN	NaN
376	2021-01-01	NaN	TEHUELCHES	NaN	NaN	NaN	NaN	NaN	NaN
377	2022-01-01	NaN	TEHUELCHES	NaN	NaN	NaN	NaN	NaN	NaN
378	2021-01-01	NaN	TELSEN	NaN	NaN	NaN	NaN	NaN	NaN
379	2022-01-01	NaN	TELSEN	NaN	NaN	NaN	NaN	NaN	NaN

380 rows × 17 columns

In [16]:

```
merged_full_join.to_csv('datos_producidos/merged_full_join.csv',
index=False)
```

1.15 Combinar dataset con mirabueno

```
In [1]: import pandas as pd
```

Este dataset representa los datos climaticos y de produccion de un campo de Chubut

```
In [9]: datos_mirabueno =  
pd.read_csv('datos/datos_produccion_clima_mirabueno.csv', sep=",")  
datos_mirabueno
```

```
Out[9]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos	esquila	total_ovino
0	31/12/1985	CHUBUT	MIRABUENO	50	1419	563	690	912	3514	363
1	31/12/1986	CHUBUT	MIRABUENO	53	1520	605	1045	680	3783	390
2	31/12/1987	CHUBUT	MIRABUENO	67	1928	637	964	512	3988	410
3	31/12/1988	CHUBUT	MIRABUENO	55	1560	575	1210	310	3590	371
4	31/12/1989	CHUBUT	MIRABUENO	42	1210	457	1091	150	2830	295
5	31/12/1990	CHUBUT	MIRABUENO	35	998	390	969	124	2396	251
6	31/12/1991	CHUBUT	MIRABUENO	39	1122	407	995	60	2503	262
7	31/12/1992	CHUBUT	MIRABUENO	39	1125	451	1225	70	2790	291
8	31/12/1993	CHUBUT	MIRABUENO	43	1240	447	1090	65	2765	288
9	31/12/1994	CHUBUT	MIRABUENO	32	925	428	1342	34	2641	276
10	31/12/1995	CHUBUT	MIRABUENO	41	1159	361	239	527	2207	232
11	31/12/1996	CHUBUT	MIRABUENO	30	849	354	643	410	2166	228
12	31/12/1997	CHUBUT	MIRABUENO	27	768	319	426	520	1940	206
13	31/12/1998	CHUBUT	MIRABUENO	39	1124	453	802	507	2805	292
14	31/12/1999	CHUBUT	MIRABUENO	39	1124	484	966	507	3000	312
15	31/12/2000	CHUBUT	MIRABUENO	57	1632	515	444	672	3200	332
16	31/12/2001	CHUBUT	MIRABUENO	48	1378	544	1089	450	3389	350
17	31/12/2002	CHUBUT	MIRABUENO	59	1688	561	488	824	3500	362
18	31/12/2003	CHUBUT	MIRABUENO	68	1952	615	855	480	3850	397
19	31/12/2004	CHUBUT	MIRABUENO	65	1860	580	423	816	3624	374
20	31/12/2005	CHUBUT	MIRABUENO	108	3083	763	147	820	4239	492
21	31/12/2006	CHUBUT	MIRABUENO	65	1850	582	702	555	3634	375
22	31/12/2007	CHUBUT	MIRABUENO	64	1842	506	775	79	3146	326
23	31/12/2008	CHUBUT	MIRABUENO	43	1232	351	136	500	1677	226
24	31/12/2009	CHUBUT	MIRABUENO	39	1101	318	193	400	1931	205
25	31/12/2010	CHUBUT	MIRABUENO	35	1009	363	855	79	2221	234
26	31/12/2011	CHUBUT	MIRABUENO	34	970	275	192	300	1651	177
27	31/12/2012	CHUBUT	MIRABUENO	31	896	283	186	432	1708	182

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos	esquila	total_ovino
28	31/12/2013	CHUBUT	MIRABUENO	32	923	325	234	582	1976	209
29	31/12/2014	CHUBUT	MIRABUENO	63	1800	451	394	200	2788	290
30	31/12/2015	CHUBUT	MIRABUENO	75	2130	654	159	1200	2738	421
31	31/12/2016	CHUBUT	MIRABUENO	81	2311	683	134	1200	3289	440
32	31/12/2017	CHUBUT	MIRABUENO	71	2032	701	264	1455	3484	452
33	31/12/2018	CHUBUT	MIRABUENO	42	1200	478	543	820	2963	308
34	31/12/2019	CHUBUT	MIRABUENO	103	2940	806	220	1129	3520	519
35	31/12/2020	CHUBUT	MIRABUENO	71	2021	533	385	430	3320	344

36 rows × 11 columns

Se lo va combinar con cada uno de los datasets de clima y produccion de los departamentos de chubut

```
In [5]: datos_chubut_1 = pd.read_csv('datos_producidos/merged_inner.csv',
sep=",")
datos_chubut_2 =
pd.read_csv('datos_producidos/merged_left_join.csv', sep=",")
datos_chubut_3 =
pd.read_csv('datos_producidos/merged_full_join.csv', sep=",")
datos_chubut_1
```

	anio	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2000-12-31	CHUBUT	RAWSON	2355	57320	20632	24405	18609	123321
1	2001-12-31	CHUBUT	RAWSON	2226	54176	19500	23067	17588	116557
2	2002-12-31	CHUBUT	RAWSON	1584	38558	13879	16417	12518	82956
3	2003-12-31	CHUBUT	RAWSON	2252	54793	19722	23330	17789	117886
4	2004-12-31	CHUBUT	RAWSON	2463	59959	21582	25529	19466	128999
...	...	...	...	...	...	...	...	...	...
160	2020-12-31	CHUBUT	RAWSON	2273	74096	18058	19267	22102	135796
161	2020-12-31	CHUBUT	RIO SENGUERR	15609	360905	146961	123560	112227	759262
162	2020-12-31	CHUBUT	SARMIENTO	3681	99173	27021	32939	20498	183312
163	2020-12-31	CHUBUT	TEHUELCHES	8071	197382	87268	86893	54493	434107
164	2020-12-31	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163716

165 rows × 10 columns

Para poder hacer la combinacion sera necesario a los datasets quitarles columnas, renombrarlas pra que la informacion del mirabueno coincida con la que pudimos conseguir para los departamentos.

Cambios en datos de mirabueno:

1. Se agregaron manualmente la columna provincia, departamento y total de ovinos. Vamos a considerar al campo "El mirabueno" como un departamento para tener mas datos.
2. Se elimaron columnas ya que no tenemos esa información para los departamentos.
3. Se renombraron las columnas
4. Se transformo el campo fecha en un datetime

```
In [11]: datos_mirabueno = datos_mirabueno.drop(
          columns=['esquila', 'kilo_lana_p/animal',
                  'porcentaje_paricion', 'rad_solar', 'acumulado_anual',
                  'acumulado_verano', 'vel_viento'])
datos_mirabueno['fecha'] =
pd.to_datetime(datos_mirabueno['fecha'])
datos_mirabueno.columns = ['fecha', 'provincia', 'departamento',
                            'carneros', 'ovejas',
                            'borregos', 'capones', 'cordero/as',
                            'total_ovinos',
                            'kilos_lana', 'rinde', 'finura',
                            'temp_media', 'temp_min',
                            'tem_max', 'humedad', 'cant_lluvia']
datos_mirabueno
```

```
Out[11]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lan
0	1985-12-31	CHUBUT	MIRABUENO	50	1419	563	690	912	3634	1560
1	1986-12-31	CHUBUT	MIRABUENO	53	1520	605	1045	680	3903	1816
2	1987-12-31	CHUBUT	MIRABUENO	67	1928	637	964	512	4108	1120
3	1988-12-31	CHUBUT	MIRABUENO	55	1560	575	1210	310	3710	1319
4	1989-12-31	CHUBUT	MIRABUENO	42	1210	457	1091	150	2950	1230
5	1990-12-31	CHUBUT	MIRABUENO	35	998	390	969	124	2516	1390
6	1991-12-31	CHUBUT	MIRABUENO	39	1122	407	995	60	2623	1580
7	1992-12-31	CHUBUT	MIRABUENO	39	1125	451	1225	70	2910	1160
8	1993-12-31	CHUBUT	MIRABUENO	43	1240	447	1090	65	2885	1030
9	1994-	CHUBUT	MIRABUENO	32	925	428	1342	34	2761	900

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lan
	12-31									
10	1995-12-31	CHUBUT	MIRABUENO	41	1159	361	239	527	2327	993
11	1996-12-31	CHUBUT	MIRABUENO	30	849	354	643	410	2286	973
12	1997-12-31	CHUBUT	MIRABUENO	27	768	319	426	520	2060	1176
13	1998-12-31	CHUBUT	MIRABUENO	39	1124	453	802	507	2925	1240
14	1999-12-31	CHUBUT	MIRABUENO	39	1124	484	966	507	3120	1370
15	2000-12-31	CHUBUT	MIRABUENO	57	1632	515	444	672	3320	1734
16	2001-12-31	CHUBUT	MIRABUENO	48	1378	544	1089	450	3509	1352
17	2002-12-31	CHUBUT	MIRABUENO	59	1688	561	488	824	3620	1735
18	2003-12-31	CHUBUT	MIRABUENO	68	1952	615	855	480	3970	1155
19	2004-12-31	CHUBUT	MIRABUENO	65	1860	580	423	816	3744	1126
20	2005-12-31	CHUBUT	MIRABUENO	108	3083	763	147	820	4921	1575
21	2006-12-31	CHUBUT	MIRABUENO	65	1850	582	702	555	3754	1378
22	2007-12-31	CHUBUT	MIRABUENO	64	1842	506	775	79	3266	1033
23	2008-12-31	CHUBUT	MIRABUENO	43	1232	351	136	500	2262	524
24	2009-12-31	CHUBUT	MIRABUENO	39	1101	318	193	400	2051	873
25	2010-12-31	CHUBUT	MIRABUENO	35	1009	363	855	79	2341	916
26	2011-12-31	CHUBUT	MIRABUENO	34	970	275	192	300	1771	651
27	2012-12-31	CHUBUT	MIRABUENO	31	896	283	186	432	1828	837
28	2013-12-31	CHUBUT	MIRABUENO	32	923	325	234	582	2096	989
29	2014-12-31	CHUBUT	MIRABUENO	63	1800	451	394	200	2908	1051
30	2015-12-31	CHUBUT	MIRABUENO	75	2130	654	159	1200	4218	1077
31	2016-12-31	CHUBUT	MIRABUENO	81	2311	683	134	1200	4409	1099
32	2017-12-31	CHUBUT	MIRABUENO	71	2032	701	264	1455	4523	1120

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lan
33	2018-12-31	CHUBUT	MIRABUENO	42	1200	478	543	820	3083	919
34	2019-12-31	CHUBUT	MIRABUENO	103	2940	806	220	1129	5198	1302
35	2020-12-31	CHUBUT	MIRABUENO	71	2021	533	385	430	3440	1213

Cambios en datos de produccion:

1. Se cambio la fecha 01-01 por 31-12 manualmente. Ya que tiene mucho sentido, es una manera de decir estos son los datos que se juntaron en el año.
2. Se renombraron las columnas

In [12]:

```
datos_chubut_1.columns = ['fecha', 'provincia', 'departamento',
                          'carneros', 'ovejas', 'borregos',
                          'capones', 'cordero/as', 'total_ovinos',
                          'kilos_lana', 'finura',
                          'rinde', 'humedad', 'cant_lluvia',
                          'tem_max', 'temp_media', 'temp_min']

datos_chubut_2.columns = ['fecha', 'provincia', 'departamento',
                          'carneros', 'ovejas', 'borregos',
                          'capones', 'cordero/as', 'total_ovinos',
                          'kilos_lana', 'finura',
                          'rinde', 'humedad', 'cant_lluvia',
                          'tem_max', 'temp_media', 'temp_min']

datos_chubut_3.columns = ['fecha', 'provincia', 'departamento',
                          'carneros', 'ovejas', 'borregos',
                          'capones', 'cordero/as', 'total_ovinos',
                          'kilos_lana', 'finura',
                          'rinde', 'humedad', 'cant_lluvia',
                          'tem_max', 'temp_media', 'temp_min']

datos_chubut_1
```

Out[12]:

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lan
0	2000-12-31	CHUBUT	RAWSON	2355	57320	20632	24405	18609	123321	2584
1	2001-12-31	CHUBUT	RAWSON	2226	54176	19500	23067	17588	116557	2584
2	2002-12-31	CHUBUT	RAWSON	1584	38558	13879	16417	12518	82956	2568

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_
3	2003-12-31	CHUBUT	RAWSON	2252	54793	19722	23330	17789	117886	3237
4	2004-12-31	CHUBUT	RAWSON	2463	59959	21582	25529	19466	128999	3328
...	...	...	...	...	...	...	...	...	...	...
160	2020-12-31	CHUBUT	RAWSON	2273	74096	18058	19267	22102	135796	2052
161	2020-12-31	CHUBUT	RIO SENGUERR	15609	360905	146961	123560	112227	759262	23112
162	2020-12-31	CHUBUT	SARMIENTO	3681	99173	27021	32939	20498	183312	8518
163	2020-12-31	CHUBUT	TEHUELCHES	8071	197382	87268	86893	54493	434107	19035
164	2020-12-31	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163716	6142

165 rows × 17 columns

Ahora se va a concatenar los datos del mirabueno a los otros tres dataset

In [13]:

```
datos_1 = pd.concat([datos_chubut_1, datos_mirabueno])
datos_1['fecha'] = pd.to_datetime(datos_1['fecha'])
datos_1 = datos_1.sort_values(['departamento', 'fecha'])
datos_1
```

Out[13]:

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_la
60	2012-12-31	CHUBUT	BIEDMA	5785	93378	30168	48447	16071	193849	649130
72	2013-12-31	CHUBUT	BIEDMA	5376	82998	25341	47350	26832	187897	620176
84	2014-12-31	CHUBUT	BIEDMA	4785	82706	29379	35155	29900	181925	649130
95	2015-12-31	CHUBUT	BIEDMA	4440	89262	29663	35437	32343	191145	672480
106	2016-12-31	CHUBUT	BIEDMA	4452	82295	33662	33470	21326	175205	653800
...	...	...	...	...	...	...	...	...	...	...
117	2016-12-31	CHUBUT	TELSEN	4896	80262	26471	39243	20290	171162	637000
129	2017-12-31	CHUBUT	TELSEN	4640	77556	27296	36570	17750	163812	637000
140	2018-12-31	CHUBUT	TELSEN	4832	79117	28366	37558	21933	171806	637000
152	2019-12-31	CHUBUT	TELSEN	4781	80109	28428	38109	17731	169158	613021
164	2020-12-31	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163716	614250

201 rows × 17 columns

```
In [14]: datos_1.to_csv('datos_producidos/datos_1.csv', index=False)
```

```
In [15]: datos_2 = pd.concat([datos_chubut_2, datos_mirabueno])
datos_2['fecha'] = pd.to_datetime(datos_2['fecha'])
datos_2 = datos_2.sort_values(['departamento', 'fecha'])
datos_2
```

```
Out[15]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos
0	2000-12-31	CHUBUT	BIEDMA	4627	88905	31529	41328	22410	188799	793
16	2001-12-31	CHUBUT	BIEDMA	4933	94817	33626	44076	23900	201352	793
32	2002-12-31	CHUBUT	BIEDMA	4506	86622	30720	40267	21835	183950	789
48	2003-12-31	CHUBUT	BIEDMA	5148	98950	35092	45997	24942	210129	994
64	2004-12-31	CHUBUT	BIEDMA	4764	91580	32478	42571	23085	194478	1022
...	...	...	...	...	...	...	...	...	...	...
271	2016-12-31	CHUBUT	TOTAL	130635	1947672	658063	703023	554340	3993733	14000
287	2017-12-31	CHUBUT	TOTAL	124046	1917298	652423	672122	606862	3972751	14000
303	2018-12-31	CHUBUT	TOTAL	120807	1929004	643675	656187	595247	3944920	14000
319	2019-12-31	CHUBUT	TOTAL	127625	1996422	681206	673858	597226	4076337	13473
335	2020-12-31	CHUBUT	TOTAL	79860	1992008	669777	666432	576856	3984933	13500

372 rows × 17 columns

```
In [16]: datos_2.to_csv('datos_producidos/datos_2.csv', index=False)
```

```
In [17]: datos_3 = pd.concat([datos_chubut_3, datos_mirabueno])
datos_3['fecha'] = pd.to_datetime(datos_3['fecha'])
datos_3 = datos_3.sort_values(['departamento', 'fecha'])
datos_3
```

```
Out[17]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	ki
0	2000-12-31	CHUBUT	BIEDMA	4627.0	88905.0	31529.0	41328.0	22410.0	188799.0	7

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kil
16	2001-12-31	CHUBUT	BIEDMA	4933.0	94817.0	33626.0	44076.0	23900.0	201352.0	7
32	2002-12-31	CHUBUT	BIEDMA	4506.0	86622.0	30720.0	40267.0	21835.0	183950.0	7
48	2003-12-31	CHUBUT	BIEDMA	5148.0	98950.0	35092.0	45997.0	24942.0	210129.0	9
64	2004-12-31	CHUBUT	BIEDMA	4764.0	91580.0	32478.0	42571.0	23085.0	194478.0	10
...	...	...	...	...	...	...	...	...	...	...
271	2016-12-31	CHUBUT	TOTAL	130635.0	1947672.0	658063.0	703023.0	554340.0	3993733.0	140
287	2017-12-31	CHUBUT	TOTAL	124046.0	1917298.0	652423.0	672122.0	606862.0	3972751.0	140
303	2018-12-31	CHUBUT	TOTAL	120807.0	1929004.0	643675.0	656187.0	595247.0	3944920.0	140
319	2019-12-31	CHUBUT	TOTAL	127625.0	1996422.0	681206.0	673858.0	597226.0	4076337.0	134
335	2020-12-31	CHUBUT	TOTAL	79860.0	1992008.0	669777.0	666432.0	576856.0	3984933.0	135

416 rows × 17 columns

In [18]:

```
datos_3.to_csv('datos_producidos/datos_3.csv', index=False)
```

Se saco el .0 de las columnas enteras, se eliminaron las filas TOTAL

1.1.15. Obtener dataset actualizado

Al tener acceso a las encuestas ganaderas realizadas en la provincia de Chubut se pudieron recolectaron nuevos datos, que nos permitieron actualizar las métricas relacionadas con las existencias ovinas, la cantidad de kilogramos de lana y la finura y el rinde de la misma. Este proceso se llevó a cabo en tres etapas: primero se actualizó la información sobre las existencias, luego se actualizó la cantidad de kilogramos de lana y finalmente se actualizó la información sobre la finura y el rinde de la lana.

Por lo tanto, hemos generado un nuevo conjunto de datos que se cargará y se guardará como archivo csv. Antes de guardar, se eliminaron las columnas intermedias que utilizamos para identificar y calcular la información actualizada, como el estado de los animales y los kilogramos por animal utilizados para calcular la cantidad de lana. Así, obtenemos un dataset más completo y preciso para el análisis y entrenamiento de modelos.

```
In [2]: import pandas as pd
```

```
In [3]: datos = pd.read_csv("datos_lana/dataset_aplicado_%_lana_f_r.csv")
datos
```

```
Out[3]:
```

	fecha	provincia	estado	departamento	carneros	ovejas	borregos	capones	corderos_corderos
0	2012-12-31	CHUBUT	MODIFICABLE	BIEDMA	2740	52320	22319	25489	2951
1	2013-12-31	CHUBUT	REAL	BIEDMA	3925	70677	27227	33389	3696
2	2014-12-31	CHUBUT	REAL	BIEDMA	3223	61195	29491	29688	3492
3	2015-12-31	CHUBUT	MODIFICABLE	BIEDMA	3885	67528	28319	33296	3811
4	2016-12-31	CHUBUT	MODIFICABLE	BIEDMA	3743	70451	30050	33038	3792
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	MODIFICABLE	TELSEN	4711	74125	28715	31728	3188
160	2017-12-31	CHUBUT	MODIFICABLE	TELSEN	4496	71149	27574	28606	3198
161	2018-12-31	CHUBUT	MODIFICABLE	TELSEN	4634	74179	28793	35979	2822
162	2019-12-31	CHUBUT	MODIFICABLE	TELSEN	4305	72515	27648	35273	2941
163	2020-12-31	CHUBUT	MODIFICABLE	TELSEN	4416	70736	27084	31418	3006

164 rows × 19 columns

```
In [5]: datos = datos.drop(['estado', 'kilos_animal'], axis=1)
```

datos

Out[5]:

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2012-12-31	CHUBUT	BIEDMA	2740	52320	22319	25489	29517	132385
1	2013-12-31	CHUBUT	BIEDMA	3925	70677	27227	33389	36961	172179
2	2014-12-31	CHUBUT	BIEDMA	3223	61195	29491	29688	34926	158523
3	2015-12-31	CHUBUT	BIEDMA	3885	67528	28319	33296	38117	171145
4	2016-12-31	CHUBUT	BIEDMA	3743	70451	30050	33038	37923	175205
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	TELSEN	4711	74125	28715	31728	31883	171162
160	2017-12-31	CHUBUT	TELSEN	4496	71149	27574	28606	31987	163812
161	2018-12-31	CHUBUT	TELSEN	4634	74179	28793	35979	28221	171806
162	2019-12-31	CHUBUT	TELSEN	4305	72515	27648	35273	29417	169158
163	2020-12-31	CHUBUT	TELSEN	4416	70736	27084	31418	30062	163716

164 rows × 10 columns

In [6]:

```
datos.to_csv("dataset_produccion_clima_chubut_actualizado.csv",  
index=False)
```

2.1. Análisis estadístico - Dataset 1

El análisis estadístico es una herramienta importante para extraer información útil de los datos. En Python, la biblioteca Pandas se utiliza comúnmente para el análisis de datos. Pandas proporciona estructuras de datos poderosas y eficientes para trabajar con datos tabulares, incluyendo la capacidad de leer y escribir diferentes formatos de archivo, un conjunto completo de herramientas para el procesamiento de datos, así como funcionalidades de visualización.

Con Pandas, podemos leer y cargar datos en varios formatos, realizar manipulaciones de datos, limpiar y transformar datos, hacer cálculos estadísticos y análisis exploratorios de datos,.

```
In [1]: import pandas as pd
```

En este caso vamos a explorar el dataset 1. Para ello, comenzaremos cargando los datos del archivo CSV correspondiente.

```
In [2]: datos =  
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut.csv")
```

La expresión `datos.shape` devuelve una tupla con dos elementos, (165, 17). Esto significa que el dataframe `datos` está compuesto por 165 registros, cada uno de ellos con información en 17 variables o características diferentes.

```
In [3]: datos.shape
```

```
Out[3]: (165, 17)
```

El atributo `columns` de un objeto `DataFrame` de pandas devuelve un objeto de tipo `Index` que contiene los nombres de las columnas del `DataFrame`. Esta propiedad permite acceder a las columnas del conjunto de datos y operar con ellas.

```
In [4]: datos.columns
```

```
Out[4]: Index(['fecha', 'provincia', 'departamento', 'carneros', 'ovejas', 'borregos',  
              'capones', 'cordero/as', 'total_ovinos', 'kilos_lana', 'finura',  
              'rinde', 'humedad', 'cant_lluvia', 'tem_max', 'temp_media', 'temp_min'],  
              dtype='object')
```

La variable "fecha" indica la fecha en que se tomó la medición. "Provincia" y "departamento" indican la ubicación geográfica donde se tomó la medición. Las siguientes columnas "carneros", "ovejas", "borregos", "capones" y "cordero/as" indican la cantidad de animales de cada tipo que se midieron. La columna "total_ovinos" es la suma de todas las columnas de animales y representa el total de ovinos medidos en cada registro. La columna "kilos_lana" indica la cantidad de lana medida en kilos. Las columnas "finura" y "rinde" son mediciones de la calidad de la lana. Las columnas "humedad", "cant_lluvia", "tem_max", "temp_media" y "temp_min" son mediciones meteorológicas.

La función `datos.head()` de Pandas permite visualizar las primeras filas de un DataFrame, por defecto muestra las primeras 5 filas. Nos da una idea general de los datos cargados y nos permite ver si hay algún problema con la carga de datos.

In [5]: `datos.head()`

Out[5]:

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lana
0	2012-12-31	CHUBUT	BIEDMA	5785	93378	30168	48447	16071	193849	649130.0
1	2013-12-31	CHUBUT	BIEDMA	5376	82998	25341	47350	26832	187897	620176.0
2	2014-12-31	CHUBUT	BIEDMA	4785	82706	29379	35155	29900	181925	649130.0
3	2015-12-31	CHUBUT	BIEDMA	4440	89262	29663	35437	32343	191145	672480.0
4	2016-12-31	CHUBUT	BIEDMA	4452	82295	33662	33470	21326	175205	653800.0

La función `tail()` de Pandas muestra las últimas filas del dataframe, por defecto las últimas 5. Esta función nos da una vista previa rápida de los últimos datos de un dataframe y nos permite comprobar que los datos se han cargado correctamente y se están procesando adecuadamente.

In [6]: `datos.tail()`

Out[6]:

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_la
160	2016-12-31	CHUBUT	TELSEN	4896	80262	26471	39243	20290	171162	637000
161	2017-12-31	CHUBUT	TELSEN	4640	77556	27296	36570	17750	163812	637000
162	2018-12-31	CHUBUT	TELSEN	4832	79117	28366	37558	21933	171806	637000
163	2019-12-31	CHUBUT	TELSEN	4781	80109	28428	38109	17731	169158	613021
164	2020-12-31	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163716	614250

El resultado de `datos.info()` muestra información sobre el conjunto de datos en cuestión. En este caso, se trata de un DataFrame de pandas con 165 entradas y 17 columnas. Cada columna representa una variable diferente, como la fecha, la provincia, el departamento, el número de carneros, el número de ovejas, la cantidad de lluvia, etc.

La tabla indica la cantidad de valores no nulos para cada columna y su tipo de dato. En este caso, todas las columnas tienen 165 valores no nulos y hay tres tipos de datos diferentes: float64, int64 y object. La información de memoria utilizada también se proporciona al final, indicando que el DataFrame ocupa 22.0+ KB de memoria.

In [7]: `datos.info()`

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165 entries, 0 to 164
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  -
0   fecha                  165 non-null    object
1   provincia              165 non-null    object
2   departamento           165 non-null    object
3   carneros               165 non-null    int64
4   ovejas                 165 non-null    int64
5   borregos               165 non-null    int64
6   capones               165 non-null    int64
7   cordero/as            165 non-null    int64
8   total_ovinos           165 non-null    int64
9   kilos_lana             165 non-null    float64
10  finura                 165 non-null    float64
11  rinde                  165 non-null    float64
12  humedad                165 non-null    float64
13  cant_lluvia            165 non-null    float64
14  tem_max                165 non-null    float64
15  temp_media             165 non-null    float64
16  temp_min               165 non-null    float64
dtypes: float64(8), int64(6), object(3)
memory usage: 22.0+ KB

```

La función `describe()` de Pandas proporciona un resumen estadístico de cada columna numérica en el DataFrame. Este resumen incluye el recuento de valores no nulos (count), la media (mean), la desviación estándar (std), los valores mínimo (min) y máximo (max) de cada columna, y los percentiles 25%, 50% y 75%.

Nos da una idea general del rango y la distribución de los datos en cada columna.

In [8]: `datos.describe()`

Out[8]:

	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_l
count	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	1.650000e
mean	8588.442424	142959.406061	48585.266667	51529.509091	42996.503030	294659.109091	1.086844e
std	6096.545968	87713.008151	37165.529596	34135.200210	34869.840309	195259.805974	7.772044e
min	1584.000000	38558.000000	13879.000000	12477.000000	11071.000000	82956.000000	2.018560e
25%	4896.000000	86473.000000	26471.000000	28734.000000	21294.000000	176065.000000	6.370000e
50%	6609.000000	123241.000000	32080.000000	40482.000000	29740.000000	230802.000000	8.770900e
75%	9410.000000	161713.000000	56336.000000	59153.000000	44013.000000	317981.000000	1.250640e
max	26957.000000	443288.000000	175695.000000	147855.000000	159654.000000	921789.000000	4.108800e

La siguiente celda de código calcula la mediana de cada columna del dataset. Esto es útil para obtener una idea de la tendencia central de los datos en cada variable. La mediana es una medida de posición que divide a un conjunto de datos ordenados en dos partes iguales, de modo que el 50% de los datos son mayores que la mediana y el 50% son menores que la mediana.

In [9]: `datos.median()`

```
Out[9]: carneros          6609.00
ovejas       123241.00
borregos      32080.00
capones       40482.00
cordero/as    29740.00
total_ovinos  230802.00
kilos_lana    877090.00
finura         19.80
rinde         59.18
humedad       54.10
cant_lluvia   155.40
tem_max       18.40
temp_media    11.70
temp_min       4.90
dtype: float64
```

Para evaluar si los datos tienen algún tipo de sesgo, es fundamental analizar la distribución de los mismos. Una distribución sesgada indica que los datos se concentran en un lado de la distribución, lo que puede afectar la interpretación de los resultados obtenidos.

La asimetría de una distribución es una medida que nos permite determinar si la distribución está sesgada hacia la derecha o hacia la izquierda. Una forma común de medir la asimetría es mediante el coeficiente de asimetría (skewness en inglés), que puede ser calculado utilizando la función `skew()` de Pandas. Si el valor del coeficiente de asimetría es cercano a cero, entonces la distribución es simétrica, mientras que si el valor es mayor a cero, indica una distribución sesgada hacia la derecha (positivo) y si es menor a cero, indica una distribución sesgada hacia la izquierda (negativo).

```
In [10]: datos.skew()
```

```
Out[10]: carneros          1.604009
ovejas       1.793906
borregos      1.745677
capones       1.412119
cordero/as    1.900176
total_ovinos  1.778671
kilos_lana    1.692090
finura       -0.213958
rinde        -0.743358
humedad      -1.592241
cant_lluvia   1.958729
tem_max      -0.001843
temp_media    0.115908
temp_min      0.455066
dtype: float64
```

Podemos observar que los valores del coeficiente de asimetría (skewness) para las existencias ovinas se encuentran en el rango de 1.412119 a 1.900176. Este rango indica que la distribución de los datos en todas estas columnas presenta sesgo positivo, es decir, una cola larga hacia la derecha. Esto significa que la mayor concentración de valores se encuentra en el lado izquierdo de la distribución, mientras que hay pocos valores en el lado derecho. En otras palabras, para estas columnas, la mayoría de los datos están agrupados en valores bajos o moderados y hay pocos valores atípicos en el extremo derecho de la distribución.

En relación a la variable "kilos de lana", se observa un coeficiente de asimetría (skewness) de 1.692090, lo que indica que la distribución de los datos está sesgada hacia la derecha. Esto implica que la mayoría de los valores se encuentran en el lado izquierdo de la distribución, mientras que hay pocos valores en el lado derecho. Por otro lado, las variables "rinde" (-0.743358) y "finura" (-0.213958) presentan una asimetría negativa, aunque en menor medida que la variable "lana", ya que sus coeficientes de asimetría están más cercanos a cero. Esto significa que la cola de la distribución se extiende más hacia la izquierda en relación con la media.

Al analizar los datos meteorológicos, se puede observar que la humedad y la temperatura máxima tienen una asimetría negativa de -1.592241 y -0.001843, respectivamente. La humedad presenta una gran asimetría, mientras que la temperatura máxima es casi simétrica. Por otro lado, la lluvia (1.958729) tiene una asimetría positiva más pronunciada y la temperatura mínima (0.455066) y la temperatura media (0.115908) presentan una ligera asimetría positiva. En estos últimos tres casos, se puede afirmar que la cola de la distribución se extiende más hacia la derecha que hacia la izquierda en relación con la media.

Conocer la cantidad de valores únicos en un dataset es fundamental para comprender la diversidad y distribución de los datos. En este sentido, empezaremos analizando tanto la fecha como el departamento de los registros.

```
In [11]: def ver_valores_unicos(col_names):  
         for col in col_names:  
             vc = datos[col].value_counts()  
             print(f"Valores únicos en la columna {col}: \n{vc}\n")
```

```
In [12]: col_names = ['fecha', 'departamento']  
         ver_valores_unicos(col_names)
```

Valores únicos en la columna fecha:

2012-12-31	12
2016-12-31	12
2017-12-31	12
2019-12-31	12
2020-12-31	12
2013-12-31	12
2014-12-31	11
2015-12-31	11
2018-12-31	11
2009-12-31	10
2010-12-31	9
2011-12-31	9
2008-12-31	9
2007-12-31	7
2006-12-31	7
2005-12-31	4
2000-12-31	1
2001-12-31	1
2002-12-31	1
2003-12-31	1
2004-12-31	1

Name: fecha, dtype: int64

Valores únicos en la columna departamento:

RAWSON	21
FUTALEUFU	16
RIO SENGUERR	16
CUSHAMEN	15
GAIMAN	15
SARMIENTO	15
PASO DE INDIOS	14
TELSEN	13
TEHUELCHES	12
FLORENTINO AMEGHINO	11
BIEDMA	9
LANGUIÑEO	8

Name: departamento, dtype: int64

Podemos observar que a partir del año 2009 hasta el 2020 se registra una cantidad de datos anual que varía entre 10 y 12. Sin embargo, para años anteriores a este periodo, se registra una disminución en la cantidad de datos debido quizás a la falta de recolección de información. En cuanto a los valores únicos de la columna "departamento", se puede notar que el departamento de Rawson cuenta con la mayor cantidad de registros, con un total de 21. Además, existen dos departamentos con 16 registros, tres con 15 registros, y varios departamentos con un número menor de registros, incluyendo Biedma con 9 registros y Languiñeo con 8 registros.

Lo siguiente es ver los valores únicos para las existencias.

In [13]:

```
col_names = ['carneros', 'ovejas',  
             'borregos', 'cordero/as', 'total_ovinos']  
ver_valores_unicos(col_names)
```

Valores únicos en la columna carneros:

6650	2
9623	2
5785	1
24926	1
2273	1
..	
5802	1
7647	1
8503	1
8368	1
2870	1

Name: carneros, Length: 163, dtype: int64

Valores únicos en la columna ovejas:

93378	1
66374	1
68936	1
74009	1
74096	1
..	
132221	1
123977	1
138573	1
143403	1

```

79157      1
Name: ovejas, Length: 165, dtype: int64

Valores únicos en la columna borregos:
30168      1
17693      1
20065      1
19514      1
18058      1
..
27090      1
20057      1
20059      1
21683      1
25804      1
Name: borregos, Length: 165, dtype: int64

```

```

Valores únicos en la columna cordero/as:
16071      1
19079      1
21294      1
18895      1
22102      1
..
27345      1
23647      1
20514      1
27900      1
18984      1
Name: cordero/as, Length: 165, dtype: int64

```

```

Valores únicos en la columna total_ovinos:
193849      1
128330      1
132996      1
136846      1
135796      1
..
234952      1
219186      1
230046      1
245238      1
163716      1
Name: total_ovinos, Length: 165, dtype: int64

```

Al analizar los datos de existencias ovinas, se puede observar que en su mayoría los valores no se repiten, ya que la mayoría de las veces sólo aparecen una vez y en algunos casos dos. Dado que una alta cantidad de valores únicos puede aumentar la complejidad de un modelo de aprendizaje automático y hacer que sea más difícil de entrenar y optimizar, esta bueno conocer esta información.

La siguiente celda se utiliza para contar la cantidad de valores faltantes en cada columna del conjunto de datos. Los valores faltantes, también conocidos como valores nulos o NA, son aquellos que no tienen un valor asignado en la tabla de datos. Saber la cantidad de valores faltantes en cada columna es

importante porque puede afectar la calidad de los análisis y modelos de aprendizaje automático que se realicen con los datos.

```
In [14]: datos.isnull().sum()
```

```
Out[14]: fecha            0
provincia            0
departamento        0
carneros             0
ovejas              0
borregos            0
capones             0
cordero/as          0
total_ovinos         0
kilos_lana           0
finura              0
rinde               0
humedad             0
cant_lluvia          0
tem_max             0
temp_media          0
temp_min            0
dtype: int64
```

En este caso, no falta ningún dato, ya que se trabajó arduamente con los scrapers y el procesamiento de datos para garantizar su integridad.

Con este primer análisis estadístico, estamos preparados para analizar el conjunto de datos de manera gráfica.

2.2. Análisis estadístico - Dataset 2

El análisis estadístico es una herramienta importante para extraer información útil de los datos. En Python, la biblioteca Pandas se utiliza comúnmente para el análisis de datos. Pandas proporciona estructuras de datos poderosas y eficientes para trabajar con datos tabulares, incluyendo la capacidad de leer y escribir diferentes formatos de archivo, un conjunto completo de herramientas para el procesamiento de datos, así como funcionalidades de visualización.

Con Pandas, podemos leer y cargar datos en varios formatos, realizar manipulaciones de datos, limpiar y transformar datos, hacer cálculos estadísticos y análisis exploratorios de datos,.

```
In [1]: import pandas as pd
```

En este caso vamos a explorar el dataset 2. Para ello, comenzaremos cargando los datos del archivo CSV correspondiente.

```
In [2]: datos =  
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_mirab
```

La expresión `datos.shape` devuelve una tupla con dos elementos, (201, 17). Esto significa que el dataframe `datos` está compuesto por 201 registros, cada uno de ellos con información en 17 variables o características diferentes.

```
In [3]: datos.shape
```

```
Out[3]: (201, 17)
```

El atributo `columns` de un objeto `DataFrame` de pandas devuelve un objeto de tipo `Index` que contiene los nombres de las columnas del `DataFrame`. Esta propiedad permite acceder a las columnas del conjunto de datos y operar con ellas.

```
In [4]: datos.columns
```

```
Out[4]: Index(['fecha', 'hectareas', 'departamento', 'carneros', 'ovejas', 'borregos',  
              'capones', 'cordero/as', 'total_ovinos', 'kilos_lana', 'finura',  
              'rinde', 'humedad', 'cant_lluvia', 'tem_max', 'temp_media', 'temp_min'],  
              dtype='object')
```

La columna "fecha" contiene la fecha en que se realizó la medición, mientras que la columna "departamento" indica la ubicación geográfica donde se llevó a cabo la medición. La columna "hectáreas" indica el área de tierra ocupada por cada departamento en hectáreas. Las columnas "carneros", "ovejas", "borregos", "capones" y "corderos/as" indican la cantidad de animales de cada tipo medidos en cada registro. La columna "total_ovinos" es la suma de todas estas columnas de animales y representa el total de ovinos medidos en cada registro. La columna "kilos_lana" indica la cantidad de lana medida en kilos para cada registro. Además, las columnas "finura" y "rinde" proporcionan mediciones de la calidad de la lana. Por otro lado, las columnas "humedad", "cant_lluvia", "tem_max", "temp_media" y "temp_min" indican mediciones meteorológicas relevantes para cada registro.

La función `datos.head()` de Pandas permite visualizar las primeras filas de un DataFrame, por defecto muestra las primeras 5 filas. Nos da una idea general de los datos cargados y nos permite ver si hay algún problema con la carga de datos.

In [5]:

```
datos.head()
```

Out[5]:

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lana
0	2012-12-31	1294000	BIEDMA	5785	93378	30168	48447	16071	193849	649130.0
1	2013-12-31	1294000	BIEDMA	5376	82998	25341	47350	26832	187897	620176.0
2	2014-12-31	1294000	BIEDMA	4785	82706	29379	35155	29900	181925	649130.0
3	2015-12-31	1294000	BIEDMA	4440	89262	29663	35437	32343	191145	672480.0
4	2016-12-31	1294000	BIEDMA	4452	82295	33662	33470	21326	175205	653800.0

La función `tail()` de Pandas muestra las últimas filas del dataframe, por defecto las últimas 5. Esta función nos da una vista previa rápida de los últimos datos de un dataframe y nos permite comprobar que los datos se han cargado correctamente y se están procesando adecuadamente.

In [6]:

```
datos.tail()
```

Out[6]:

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lana
196	2016-12-31	1989300	TELSEN	4896	80262	26471	39243	20290	171162	637000.0
197	2017-12-31	1989300	TELSEN	4640	77556	27296	36570	17750	163812	637000.0
198	2018-12-31	1989300	TELSEN	4832	79117	28366	37558	21933	171806	637000.0
199	2019-12-31	1989300	TELSEN	4781	80109	28428	38109	17731	169158	613021.0
200	2020-12-31	1989300	TELSEN	2870	79157	25804	36901	18984	163716	614250.0

El resultado de `datos.info()` muestra información sobre el conjunto de datos en cuestión. En este caso, se trata de un DataFrame de pandas con 201 entradas y 17 columnas. Cada columna representa una variable diferente, como la fecha, la provincia, el departamento, el número de carneros, el número de ovejas, la cantidad de lluvia, etc.

La tabla indica la cantidad de valores no nulos para cada columna y su tipo de dato. En este caso, todas las columnas tienen 201 valores no nulos y hay tres tipos de datos diferentes: float64, int64 y object. La información de memoria utilizada también se proporciona al final, indicando que el DataFrame ocupa 26.8+ KB de memoria.

In [7]:

```
datos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  -
0   fecha                 201 non-null    object
1   hectareas             201 non-null    int64
2   departamento         201 non-null    object
3   carneros              201 non-null    int64
4   ovejas                201 non-null    int64
5   borregos              201 non-null    int64
6   capones               201 non-null    int64
7   cordero/as           201 non-null    int64
8   total_ovinos          201 non-null    int64
9   kilos_lana            201 non-null    float64
10  finura                201 non-null    float64
11  rinde                 201 non-null    float64
12  humedad               201 non-null    float64
13  cant_lluvia           201 non-null    float64
14  tem_max               201 non-null    float64
15  temp_media            201 non-null    float64
16  temp_min              201 non-null    float64
dtypes: float64(8), int64(7), object(2)
memory usage: 26.8+ KB
```

La función `describe()` de Pandas proporciona un resumen estadístico de cada columna numérica en el DataFrame. Este resumen incluye el recuento de valores no nulos (count), la media (mean), la desviación estándar (std), los valores mínimo (min) y máximo (max) de cada columna, y los percentiles 25%, 50% y 75%.

Nos da una idea general del rango y la distribución de los datos en cada columna.

In [8]:

```
datos.describe()
```

Out[8]:

	hectareas	carneros	ovejas	borregos	capones	cordero/as	total_ov
count	2.010000e+02	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000
mean	1.185600e+06	7059.592040	117623.000000	39972.074627	42412.358209	35389.223881	242456.233
std	7.553617e+05	6422.172245	96258.454605	38397.581433	36583.899122	35547.605098	209323.795
min	2.500000e+02	27.000000	768.000000	275.000000	134.000000	34.000000	1771.000
25%	3.922000e+05	2520.000000	61699.000000	19514.000000	21340.000000	17144.000000	123321.000
50%	1.456300e+06	5785.000000	106471.000000	30119.000000	34584.000000	27014.000000	214592.000
75%	1.625000e+06	8550.000000	150454.000000	46140.000000	52400.000000	41337.000000	287755.000
max	2.233500e+06	26957.000000	443288.000000	175695.000000	147855.000000	159654.000000	921789.000

La siguiente celda de código calcula la mediana de cada columna del dataset. Esto es útil para obtener una idea de la tendencia central de los datos en cada variable. La mediana es una medida de posición que

divide a un conjunto de datos ordenados en dos partes iguales, de modo que el 50% de los datos son mayores que la mediana y el 50% son menores que la mediana.

```
In [9]: datos.median()
```

```
Out[9]: hectareas      1456300.00
carneros        5785.00
ovejas          106471.00
borregos         30119.00
capones          34584.00
cordero/as       27014.00
total_ovinos     214592.00
kilos_lana       768912.00
finura           19.77
rinde            59.06
humedad          54.30
cant_lluvia      161.00
tem_max          19.85
temp_media       12.50
temp_min         5.30
dtype: float64
```

Para evaluar si los datos tienen algún tipo de sesgo, es fundamental analizar la distribución de los mismos. Una distribución sesgada indica que los datos se concentran en un lado de la distribución, lo que puede afectar la interpretación de los resultados obtenidos.

La asimetría de una distribución es una medida que nos permite determinar si la distribución está sesgada hacia la derecha o hacia la izquierda. Una forma común de medir la asimetría es mediante el coeficiente de asimetría (skewness en inglés), que puede ser calculado utilizando la función `skew()` de Pandas. Si el valor del coeficiente de asimetría es cercano a cero, entonces la distribución es simétrica, mientras que si el valor es mayor a cero, indica una distribución sesgada hacia la derecha (positivo) y si es menor a cero, indica una distribución sesgada hacia la izquierda (negativo).

```
In [10]: datos.skew()
```

```
Out[10]: hectareas      -0.310836
carneros        1.430845
ovejas          1.349549
borregos         1.629180
capones          1.219575
cordero/as       1.815691
total_ovinos     1.461836
kilos_lana       1.508300
finura          -0.125210
rinde           -0.728973
humedad         -1.695929
cant_lluvia      2.131434
tem_max         -0.302842
temp_media      -0.198763
temp_min         0.257953
dtype: float64
```

Podemos observar que los valores del coeficiente de asimetría (skewness) para las existencias ovinas se encuentran en el rango de 1.219575 a 1.815691. Este rango indica que la distribución de los datos en

todas estas columnas presenta sesgo positivo, es decir, una cola larga hacia la derecha. Esto significa que la mayor concentración de valores se encuentra en el lado izquierdo de la distribución, mientras que hay pocos valores en el lado derecho. En otras palabras, para estas columnas, la mayoría de los datos están agrupados en valores bajos o moderados y hay pocos valores atípicos en el extremo derecho de la distribución.

En relación a la variable "kilos de lana", se observa un coeficiente de asimetría (skewness) de 1.508300, lo que indica que la distribución de los datos está sesgada hacia la derecha. Esto implica que la mayoría de los valores se encuentran en el lado izquierdo de la distribución, mientras que hay pocos valores en el lado derecho. Por otro lado, las variables "rinde" (-0.728973) y "finura" (-0.125210) presentan una asimetría negativa, aunque en menor medida que la variable "lana", ya que sus coeficientes de asimetría están más cercanos a cero. Esto significa que la cola de la distribución se extiende más hacia la izquierda en relación con la media.

Al analizar los datos meteorológicos, se puede observar que la cantidad de lluvia y la temperatura mínima tienen una asimetría positiva de 2.131434 y 0.257953, respectivamente. La lluvia presenta una gran asimetría, mientras que la temperatura máxima es mas simétrica. Por otro lado, la humedad (-1.695929) tiene una asimetría negativa más pronunciada y la temperatura máxima (-0.302842) y la temperatura media (-0.198763) presentan una ligera asimetría positiva. En estos últimos tres casos, se puede afirmar que la cola de la distribución se extiende más hacia la izquierda que hacia la derecha en relación con la media.

Conocer la cantidad de valores únicos en un dataset es fundamental para comprender la diversidad y distribución de los datos. En este sentido, empezaremos analizando tanto la fecha como el departamento de los registros.

```
In [11]: def ver_valores_unicos(col_names):  
         for col in col_names:  
             vc = datos[col].value_counts()  
             print(f"Valores únicos en la columna {col}: \n{vc}\n")
```

```
In [12]: col_names = ['fecha', 'departamento']  
         ver_valores_unicos(col_names)
```

Valores únicos en la columna fecha:

2012-12-31	13
2016-12-31	13
2017-12-31	13
2019-12-31	13
2020-12-31	13
2013-12-31	13
2014-12-31	12
2015-12-31	12
2018-12-31	12
2009-12-31	11
2010-12-31	10
2011-12-31	10
2008-12-31	10
2007-12-31	8
2006-12-31	8
2005-12-31	5

2000-12-31	2
2003-12-31	2
2002-12-31	2
2001-12-31	2
2004-12-31	2
1990-12-31	1
1991-12-31	1
1992-12-31	1
1993-12-31	1
1994-12-31	1
1995-12-31	1
1996-12-31	1
1997-12-31	1
1998-12-31	1
1999-12-31	1
1989-12-31	1
1988-12-31	1
1986-12-31	1
1985-12-31	1
1987-12-31	1

Name: fecha, dtype: int64

Valores únicos en la columna departamento:

MIRABUENO	36
RAWSON	21
FUTALEUFU	16
RIO SENGUERR	16
CUSHAMEN	15
GAIMAN	15
SARMIENTO	15
PASO DE INDIOS	14
TELSEN	13
TEHUELCHES	12
FLORENTINO AMEGHINO	11
BIEDMA	9
LANGUIÑEO	8

Name: departamento, dtype: int64

Podemos observar que entre los años 2008 y 2020, la cantidad de datos anuales registrados varía entre 10 y 13. No obstante, para los años anteriores a este periodo, se observa una disminución en la cantidad de datos, quizás debido a una falta de recolección de información. En cuanto a los valores únicos de la columna "departamento", se puede notar que el falso departamento "Mirabueno" cuenta con la mayor cantidad de registros, con un total de 36, seguido de Rawson con 21. Además, hay dos departamentos con 16 registros, tres con 15 registros, y varios departamentos con un número menor de registros, incluyendo Biedma con 9 registros y Languiñeo con 8 registros.

Lo siguiente es ver los valores únicos para las existencias.

```
In [13]: col_names = ['carneros', 'ovejas',
                  'borregos', 'cordero/as', 'total_ovinos']
          ver_valores_unicos(col_names)
```

Valores únicos en la columna carneros:

39	5
----	---

```

9623    2
71      2
42      2
35      2
..
8309    1
5295    1
8550    1
8875    1
2870    1
Name: carneros, Length: 189, dtype: int64

```

Valores únicos en la columna ovejas:

```

1124      2
93378     1
61212     1
52805     1
55886     1
..
112547     1
103552     1
107526     1
102562     1
79157      1
Name: ovejas, Length: 200, dtype: int64

```

Valores únicos en la columna borregos:

```

451      2
30168     1
15436     1
22032     1
19007     1
..
47178     1
38645     1
43471     1
38569     1
25804     1
Name: borregos, Length: 200, dtype: int64

```

Valores únicos en la columna cordero/as:

```

507      2
1200     2
79       2
820      2
19079     1
..
21887     1
42894     1
17445     1
40508     1
18984     1
Name: cordero/as, Length: 197, dtype: int64

```

Valores únicos en la columna total_ovinos:

```

193849    1

```

```

120837    1
128999    1
131693    1
113609    1
..
229837    1
214592    1
228473    1
231763    1
163716    1
Name: total_ovinos, Length: 201, dtype: int64

```

Al analizar los datos de existencias ovinas, se puede observar que en su mayoría los valores no se repiten, ya que la mayoría de las veces sólo aparecen una vez y en algunos casos dos. Dado que una alta cantidad de valores únicos puede aumentar la complejidad de un modelo de aprendizaje automático y hacer que sea más difícil de entrenar y optimizar, es bueno conocer esta información.

La siguiente celda se utiliza para contar la cantidad de valores faltantes en cada columna del conjunto de datos. Los valores faltantes, también conocidos como valores nulos o NA, son aquellos que no tienen un valor asignado en la tabla de datos. Saber la cantidad de valores faltantes en cada columna es importante porque puede afectar la calidad de los análisis y modelos de aprendizaje automático que se realicen con los datos.

In [14]:

```
datos.isnull().sum()
```

Out[14]:

```

fecha            0
hectareas        0
departamento    0
carneros         0
ovejas           0
borregos         0
capones          0
cordero/as       0
total_ovinos     0
kilos_lana       0
finura           0
rinde            0
humedad          0
cant_lluvia      0
tem_max          0
temp_media       0
temp_min         0
dtype: int64

```

En este caso, no falta ningún dato, ya que se trabajó arduamente con los scrapers y el procesamiento de datos para garantizar su integridad.

Con este primer análisis estadístico, estamos preparados para analizar el conjunto de datos de manera gráfica.

2.3. Análisis estadístico - Dataset 1

El análisis estadístico es una herramienta importante para extraer información útil de los datos. En Python, la biblioteca Pandas se utiliza comúnmente para el análisis de datos. Pandas proporciona estructuras de datos poderosas y eficientes para trabajar con datos tabulares, incluyendo la capacidad de leer y escribir diferentes formatos de archivo, un conjunto completo de herramientas para el procesamiento de datos, así como funcionalidades de visualización.

Con Pandas, podemos leer y cargar datos en varios formatos, realizar manipulaciones de datos, limpiar y transformar datos, hacer cálculos estadísticos y análisis exploratorios de datos,.

```
In [1]: import pandas as pd
```

En este caso vamos a explorar el dataset 1. Para ello, comenzaremos cargando los datos del archivo CSV correspondiente.

```
In [2]: datos =  
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_actua
```

La expresión `datos.shape` devuelve una tupla con dos elementos, (164, 17). Esto significa que el dataframe `datos` está compuesto por 164 registros, cada uno de ellos con información en 17 variables o características diferentes.

```
In [3]: datos.shape
```

```
Out[3]: (164, 17)
```

El atributo `columns` de un objeto `DataFrame` de pandas devuelve un objeto de tipo `Index` que contiene los nombres de las columnas del `DataFrame`. Esta propiedad permite acceder a las columnas del conjunto de datos y operar con ellas.

```
In [4]: datos.columns
```

```
Out[4]: Index(['fecha', 'provincia', 'departamento', 'carneros', 'ovejas', 'borregos',  
              'capones', 'corderos_corderas', 'total_ovinos', 'kilos_lana', 'finura',  
              'rinde', 'humedad', 'cant_lluvia', 'tem_max', 'temp_media', 'temp_min'],  
              dtype='object')
```

La variable "fecha" indica la fecha en que se tomó la medición. "Provincia" y "departamento" indican la ubicación geográfica donde se tomó la medición. Las siguientes columnas "carneros", "ovejas", "borregos", "capones" y "cordero/as" indican la cantidad de animales de cada tipo que se midieron. La columna "total_ovinos" es la suma de todas las columnas de animales y representa el total de ovinos medidos en cada registro. La columna "kilos_lana" indica la cantidad de lana medida en kilos. Las columnas "finura" y "rinde" son mediciones de la calidad de la lana. Las columnas "humedad", "cant_lluvia", "tem_max", "temp_media" y "temp_min" son mediciones meteorológicas.

La función `datos.head()` de Pandas permite visualizar las primeras filas de un DataFrame, por defecto muestra las primeras 5 filas. Nos da una idea general de los datos cargados y nos permite ver si hay algún problema con la carga de datos.

In [5]:

```
datos.head()
```

Out[5]:

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	ki
0	2012-12-31	CHUBUT	BIEDMA	2740	52320	22319	25489	29517	132385	4
1	2013-12-31	CHUBUT	BIEDMA	3925	70677	27227	33389	36961	172179	5
2	2014-12-31	CHUBUT	BIEDMA	3223	61195	29491	29688	34926	158523	5
3	2015-12-31	CHUBUT	BIEDMA	3885	67528	28319	33296	38117	171145	5
4	2016-12-31	CHUBUT	BIEDMA	3743	70451	30050	33038	37923	175205	6

La función `tail()` de Pandas muestra las últimas filas del dataframe, por defecto las últimas 5. Esta función nos da una vista previa rápida de los últimos datos de un dataframe y nos permite comprobar que los datos se han cargado correctamente y se están procesando adecuadamente.

In [6]:

```
datos.tail()
```

Out[6]:

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
159	2016-12-31	CHUBUT	TELSEN	4711	74125	28715	31728	31883	171162
160	2017-12-31	CHUBUT	TELSEN	4496	71149	27574	28606	31987	163812
161	2018-12-31	CHUBUT	TELSEN	4634	74179	28793	35979	28221	171806
162	2019-12-31	CHUBUT	TELSEN	4305	72515	27648	35273	29417	169158
163	2020-12-31	CHUBUT	TELSEN	4416	70736	27084	31418	30062	163716

El resultado de `datos.info()` muestra información sobre el conjunto de datos en cuestión. En este caso, se trata de un DataFrame de pandas con 164 entradas y 17 columnas. Cada columna representa una variable diferente, como la fecha, la provincia, el departamento, el número de carneros, el número de ovejas, la cantidad de lluvia, etc.

La tabla indica la cantidad de valores no nulos para cada columna y su tipo de dato. En este caso, todas las columnas tienen 164 valores no nulos y hay tres tipos de datos diferentes: float64, int64 y object. La información de memoria utilizada también se proporciona al final, indicando que el DataFrame ocupa 21.9+ KB de memoria.

In [7]:

```
datos.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 164 entries, 0 to 163
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  -
0   fecha                 164 non-null   object
1   provincia             164 non-null   object
2   departamento          164 non-null   object
3   carneros              164 non-null   int64
4   ovejas                164 non-null   int64
5   borregos              164 non-null   int64
6   capones               164 non-null   int64
7   corderos_corderas    164 non-null   int64
8   total_ovinos          164 non-null   int64
9   kilos_lana            164 non-null   float64
10  finura                164 non-null   float64
11  rinde                 164 non-null   float64
12  humedad               164 non-null   float64
13  cant_lluvia           164 non-null   float64
14  tem_max               164 non-null   float64
15  temp_media            164 non-null   float64
16  temp_min              164 non-null   float64
dtypes: float64(8), int64(6), object(3)
memory usage: 21.9+ KB

```

La función `describe()` de Pandas proporciona un resumen estadístico de cada columna numérica en el DataFrame. Este resumen incluye el recuento de valores no nulos (count), la media (mean), la desviación estándar (std), los valores mínimo (min) y máximo (max) de cada columna, y los percentiles 25%, 50% y 75%.

Nos da una idea general del rango y la distribución de los datos en cada columna.

```

In [8]: datos.describe()

```

	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos	kil
count	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	1.6400
mean	6795.079268	109195.097561	40317.048780	39883.079268	54915.945122	251106.250000	8.9400
std	4791.898721	74333.016127	31458.372487	28173.486877	39221.660324	175984.465887	6.4688
min	1421.000000	21913.000000	7264.000000	8605.000000	11300.000000	50513.000000	1.6690
25%	3875.500000	63116.500000	21414.500000	19209.750000	30004.000000	138213.000000	5.2390
50%	5374.500000	85986.000000	28544.000000	31487.000000	42456.500000	183361.000000	6.6338
75%	7508.750000	125565.250000	45959.250000	47709.500000	60730.000000	293195.500000	9.9478
max	21067.000000	345839.000000	139387.000000	114801.000000	178182.000000	797790.000000	2.9917

La siguiente celda de código calcula la mediana de cada columna del dataset. Esto es útil para obtener una idea de la tendencia central de los datos en cada variable. La mediana es una medida de posición que divide a un conjunto de datos ordenados en dos partes iguales, de modo que el 50% de los datos son mayores que la mediana y el 50% son menores que la mediana.

```

In [9]: datos.median()

```

```
Out[9]: carneros          5374.500
ovejas          85986.000
borregos        28544.000
capones         31487.000
corderos_corderas 42456.500
total_ovinos    183361.000
kilos_lana      663384.050
finura          20.010
rinde           57.465
humedad         54.050
cant_lluvia     155.800
tem_max         18.550
temp_media      11.750
temp_min        4.900
dtype: float64
```

Para evaluar si los datos tienen algún tipo de sesgo, es fundamental analizar la distribución de los mismos. Una distribución sesgada indica que los datos se concentran en un lado de la distribución, lo que puede afectar la interpretación de los resultados obtenidos.

La asimetría de una distribución es una medida que nos permite determinar si la distribución está sesgada hacia la derecha o hacia la izquierda. Una forma común de medir la asimetría es mediante el coeficiente de asimetría (skewness en inglés), que puede ser calculado utilizando la función `skew()` de Pandas. Si el valor del coeficiente de asimetría es cercano a cero, entonces la distribución es simétrica, mientras que si el valor es mayor a cero, indica una distribución sesgada hacia la derecha (positivo) y si es menor a cero, indica una distribución sesgada hacia la izquierda (negativo).

```
In [10]: datos.skew()
```

```
Out[10]: carneros          1.552531
ovejas          1.669065
borregos        1.588924
capones         1.245943
corderos_corderas 1.607333
total_ovinos    1.579539
kilos_lana      1.625766
finura          0.865873
rinde           -0.346975
humedad         -1.584182
cant_lluvia     1.949903
tem_max         -0.008854
temp_media      0.106191
temp_min        0.444767
dtype: float64
```

Podemos observar que los valores del coeficiente de asimetría (skewness) para las existencias ovinas se encuentran en el rango de 1.245943 a 1.669065. Este rango indica que la distribución de los datos en todas estas columnas presenta sesgo positivo, es decir, una cola larga hacia la derecha. Esto significa que la mayor concentración de valores se encuentra en el lado izquierdo de la distribución, mientras que hay pocos valores en el lado derecho. En otras palabras, para estas columnas, la mayoría de los datos están agrupados en valores bajos o moderados y hay pocos valores atípicos en el extremo derecho de la distribución.

En relación a la variable "kilos de lana", se puede observar un coeficiente de asimetría (skewness) de 1.625766, lo que indica que la distribución de los datos está sesgada hacia la derecha. Esto significa que la mayoría de los valores se concentran en el lado izquierdo de la distribución, mientras que hay pocos valores en el lado derecho. Por otro lado, las variables "rinde" (-0.346975) y "finura" (0.865873) presentan asimetría negativa y positiva, respectivamente, aunque en menor medida que la variable "lana", ya que sus coeficientes de asimetría están más cercanos a cero.

Al analizar los datos meteorológicos, se puede observar que la humedad y la temperatura máxima tienen una asimetría negativa de -1.584182 y -0.008854, respectivamente. La humedad presenta una gran asimetría, mientras que la temperatura máxima es casi simétrica. Por otro lado, la lluvia (1.949903) tiene una asimetría positiva más pronunciada y la temperatura mínima (0.444767) y la temperatura media (0.106191) presentan una ligera asimetría positiva. En estos últimos tres casos, se puede afirmar que la cola de la distribución se extiende más hacia la derecha que hacia la izquierda en relación con la media.

Conocer la cantidad de valores únicos en un dataset es fundamental para comprender la diversidad y distribución de los datos. En este sentido, empezaremos analizando tanto la fecha como el departamento de los registros.

```
In [11]: def ver_valores_unicos(col_names):  
         for col in col_names:  
             vc = datos[col].value_counts()  
             print(f"Valores únicos en la columna {col}: \n{vc}\n")
```

```
In [12]: col_names = ['fecha', 'departamento']  
         ver_valores_unicos(col_names)
```

Valores únicos en la columna fecha:

2012-12-31	12
2016-12-31	12
2017-12-31	12
2019-12-31	12
2013-12-31	12
2014-12-31	11
2015-12-31	11
2018-12-31	11
2020-12-31	11
2009-12-31	10
2010-12-31	9
2011-12-31	9
2008-12-31	9
2007-12-31	7
2006-12-31	7
2005-12-31	4
2000-12-31	1
2001-12-31	1
2002-12-31	1
2003-12-31	1
2004-12-31	1

Name: fecha, dtype: int64

Valores únicos en la columna departamento:

RAWSON	21
FUTALEUFU	16
RIO SENGUERR	16
CUSHAMEN	15
GAIMAN	15
SARMIENTO	15
PASO DE INDIOS	14
TELSEN	13
TEHUELCHES	12
FLORENTINO AMEGHINO	11
BIEDMA	9
LANGUIÑEO	7

Name: departamento, dtype: int64

Podemos observar que a partir del año 2009 hasta el 2020 se registra una cantidad de datos anual que varía entre 10 y 12. Sin embargo, para años anteriores a este periodo, se registra una disminución en la cantidad de datos debido quizás a la falta de recolección de información. En cuanto a los valores únicos de la columna "departamento", se puede notar que el departamento de Rawson cuenta con la mayor cantidad de registros, con un total de 21. Además, existen dos departamentos con 16 registros, tres con 15 registros, y varios departamentos con un número menor de registros, incluyendo Biedma con 9 registros y Languiño con 7 registros.

Lo siguiente es ver los valores únicos para las existencias.

```
In [13]: col_names = ['carneros', 'ovejas',
                  'borregos', 'corderos_corderas', 'total_ovinos']
          ver_valores_unicos(col_names)
```

Valores únicos en la columna carneros:

3885	2
6624	2
2740	1
14701	1
2198	1
..	
4286	1
3191	1
4102	1
4936	1
4416	1

Name: carneros, Length: 162, dtype: int64

Valores únicos en la columna ovejas:

104515	2
52320	1
282372	1
31221	1
36271	1
..	
85321	1
78610	1
61268	1
87337	1
70736	1

```
Name: ovejas, Length: 163, dtype: int64
```

```
Valores únicos en la columna borregos:
```

```
22319    1
8937     1
12300    1
11357    1
9613     1
```

```
..
```

```
21365    1
22475    1
16706    1
20059    1
27084    1
```

```
Name: borregos, Length: 164, dtype: int64
```

```
Valores únicos en la columna corderos_corderas:
```

```
33896     2
29517     1
139943    1
17463     1
16450     1
```

```
..
```

```
36282     1
34392     1
29065     1
31970     1
30062     1
```

```
Name: corderos_corderas, Length: 163, dtype: int64
```

```
Valores únicos en la columna total_ovinos:
```

```
132385    1
62330     1
82996     1
76846     1
65796     1
```

```
..
```

```
171408    1
164712    1
128210    1
168566    1
163716    1
```

```
Name: total_ovinos, Length: 164, dtype: int64
```

Al analizar los datos de existencias ovinas, se puede observar que en su mayoría los valores no se repiten, ya que la mayoría de las veces sólo aparecen una vez y en algunos casos dos. Dado que una alta cantidad de valores únicos puede aumentar la complejidad de un modelo de aprendizaje automático y hacer que sea más difícil de entrenar y optimizar, esta bueno conocer esta información.

La siguiente celda se utiliza para contar la cantidad de valores faltantes en cada columna del conjunto de datos. Los valores faltantes, también conocidos como valores nulos o NA, son aquellos que no tienen un valor asignado en la tabla de datos. Saber la cantidad de valores faltantes en cada columna es importante porque puede afectar la calidad de los análisis y modelos de aprendizaje automático que se realicen con los datos.

```
In [14]: datos.isnull().sum()
```

```
Out[14]: fecha                0  
provincia                0  
departamento            0  
carneros                 0  
ovejas                  0  
borregos                0  
capones                 0  
corderos_corderas       0  
total_ovinos             0  
kilos_lana               0  
finura                  0  
rinde                   0  
humedad                 0  
cant_lluvia              0  
tem_max                 0  
temp_media              0  
temp_min                0  
dtype: int64
```

En este caso, no falta ningún dato, ya que se trabajó arduamente con los scrapers y el procesamiento de datos para garantizar su integridad.

Con este primer análisis estadístico, estamos preparados para analizar el conjunto de datos de manera gráfica.

2.4. Análisis Gráfico - Dataset 1

A continuación, realizaremos un análisis visual del conjunto de datos con el objetivo de identificar patrones o relaciones entre las variables. Para ello, haremos uso de las librerías Pandas, Seaborn y Matplotlib.

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [2]: %matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

```
In [3]: datos =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut.csv")
```

Para que los graficos funcionen mejor es necesario que el atributo fecha sea de tipo date.

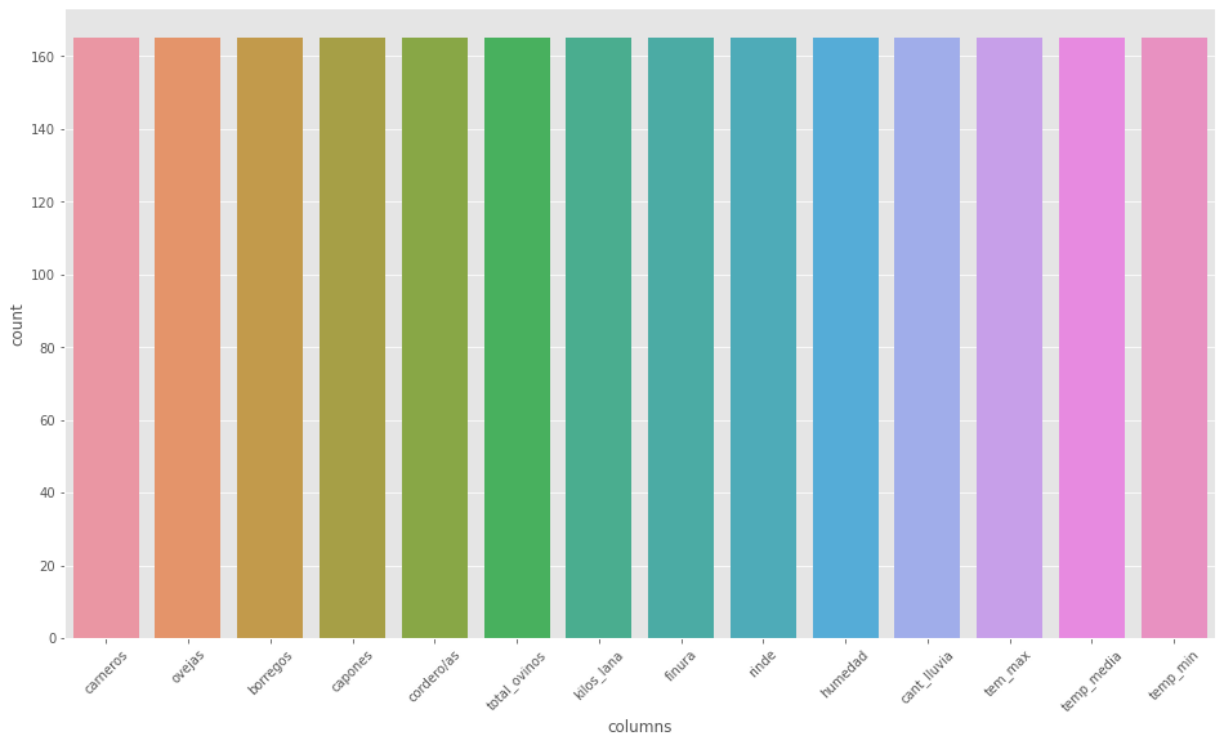
```
In [4]: datos['fecha'] = pd.to_datetime(datos['fecha'])
```

En primer lugar, realizaremos un análisis conjunto de las variables para tener una visión general del dataset. A continuación, procederemos a analizar las variables de forma individual para obtener una comprensión más detallada de cada una de ellas. Por último, relacionaremos los atributos del conjunto de datos para identificar patrones y posibles relaciones entre ellos.

Variables en conjunto

En primer lugar, realizaremos un gráfico que nos permitirá conocer la cantidad de datos disponibles para cada atributo del dataset, observando que cada columna cuenta con 165 datos.

```
In [5]: sns.countplot(data=datos)
plt.xticks(rotation=45)
plt.xlabel('columns')
plt.show()
```

Luego, utilizaremos la función `pairplot` de `seaborn` para obtener una matriz que muestra un gráfico de dispersión para cada combinación de atributos. Dicha matriz es bastante amplia debido a la cantidad de variables presentes en el dataset, pero podemos identificar algunas gráficas que parecen ajustarse a una regresión lineal en principio, principalmente aquellas relacionadas con las existencias de ovinos, como carneros, ovejas, borregos, capones, corderos, coderas y el total de ovinos.

```
In [6]: sns.pairplot(datos, hue="departamento")
```

```
Out[6]: <seaborn.axisgrid.PairGrid at 0x7f3b071d0c40>
```



Asimismo, se pueden apreciar variables relacionadas con la temperatura, como la temperatura máxima, media y mínima, que también muestran una tendencia lineal. Sin embargo, al combinar datos de producción con información climática, la mayoría de las columnas no parecen ajustarse a un modelo lineal en principio.

Para mejorar la visualización de las relaciones entre variables, vamos a crear una matriz de correlación y graficarla en un mapa de calor de seaborn. De esta forma, podremos identificar de manera más clara cuáles son las variables que tienen una mayor o menor correlación entre sí.

Una matriz de correlación es una tabla que muestra las correlaciones entre todas las variables de un conjunto de datos. Las correlaciones miden el grado en que dos variables están relacionadas entre sí, y pueden oscilar entre -1 y 1.

Un valor de correlación de 1 significa que dos variables están perfectamente correlacionadas positivamente, es decir, que cuando una aumenta, la otra también lo hace en la misma medida. Por otro lado, un valor de -1 indica una correlación perfectamente negativa, lo que significa que cuando una variable aumenta, la otra disminuye en la misma medida. Finalmente, un valor de 0 indica que no hay correlación entre dos variables.

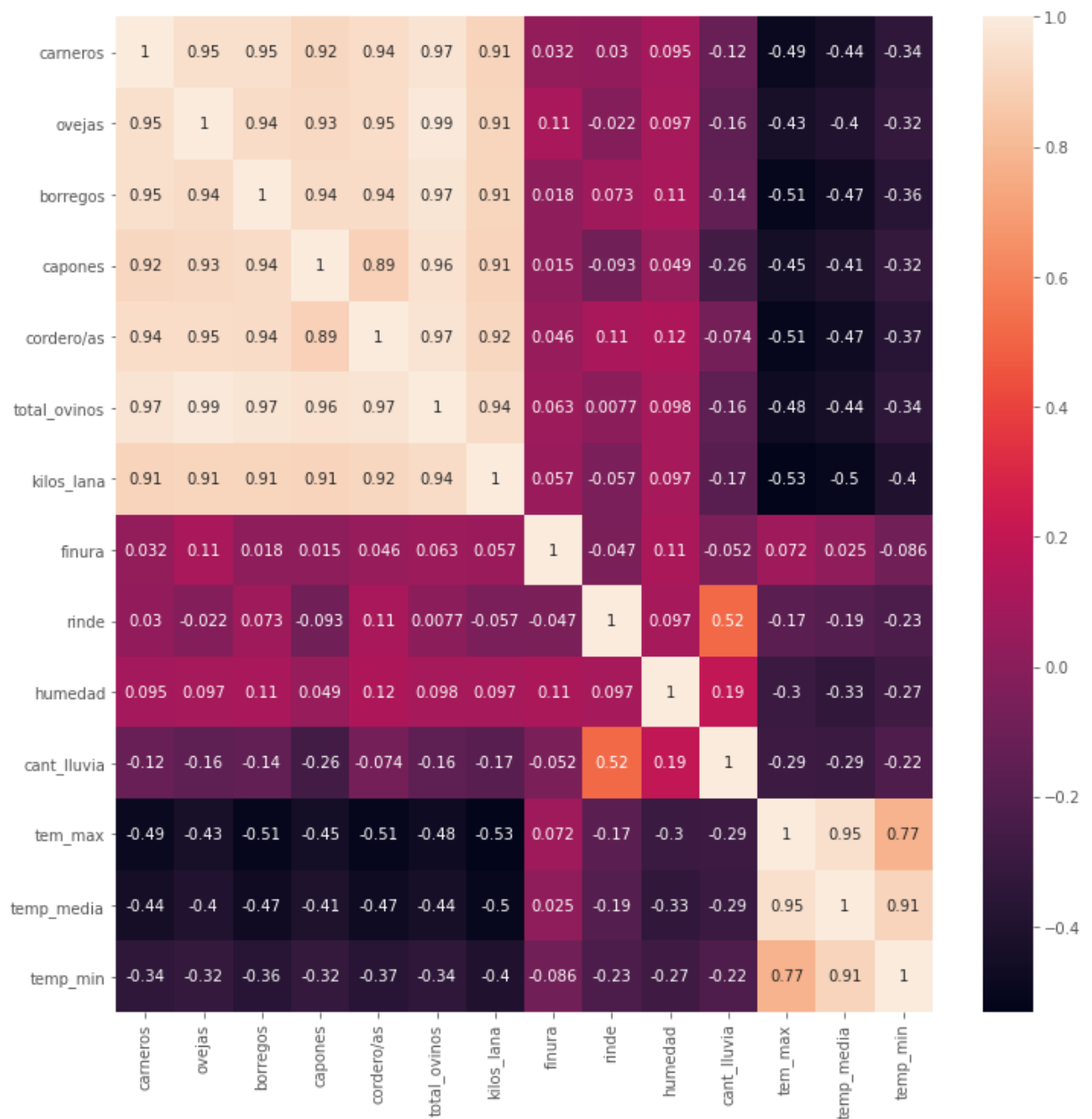
La matriz de correlación se representa como una tabla cuadrada, donde cada variable se encuentra tanto en la fila como en la columna de la matriz. En la diagonal principal, los valores son siempre 1, ya que representa la correlación de una variable consigo misma. Los valores fuera de la diagonal principal

indican la correlación entre dos variables diferentes, y su posición en la matriz es simétrica, es decir, el valor de la fila i y la columna j es el mismo que el valor de la fila j y la columna i .

En la matriz de correlación, representada por un gráfico de calor, se puede observar la intensidad de la correlación entre las diferentes variables del dataset. De acuerdo con la documentación de seaborn para el heatmap, las correlaciones negativas se muestran en violeta oscuro, las correlaciones positivas en naranja claro y las correlaciones cercanas a cero en rosado.

```
In [7]: correlation_mat = datos.corr()
fig, ax = plt.subplots(figsize = (12, 12))
sns.heatmap(correlation_mat, annot = True, ax = ax)
```

Out[7]: <AxesSubplot:>



En este caso, se puede observar que los atributos relacionados con las existencias ovinas tienen una correlación muy alta entre ellos, con valores que oscilan entre 0.89 y 0.97. Además, los atributos relacionados con la temperatura también presentan una alta correlación entre ellos, con valores que van desde 0.77 a 0.95. Estos atributos son los que previamente se habían identificado como los que seguían una forma lineal en el gráfico de dispersión.

Otra correlación interesante a destacar es la que se presenta entre la cantidad de kilos de lana producidos y las existencias ovinas, ya que se observa una alta correlación con valores que oscilan entre 0.91 y 0.94.

Teniendo en cuenta la bibliografía y la experiencia de algunos productores, se sabe que la lluvia es un factor determinante en la producción ganadera ovina de la región, debido a que se trata de una zona con bajos niveles de precipitaciones. Es por esto que resulta interesante analizar la correlación existente entre la lluvia acumulada en el año y las existencias ovinas, así como también con la producción de lana. Al observar la matriz de correlación, los valores obtenidos para estas relaciones están más cerca del color rosa que del violeta o naranja, lo que indica que no existe una correlación significativa entre estas variables. En detalle, se aprecia una correlación negativa débil entre la cantidad de lluvia y las existencias ovinas, así como también con la producción de lana. Los valores obtenidos oscilan entre -0.12 y -0.16 para carneros, ovejas y borregos, mientras que para los corderos la relación es de -0.074 y para los capones de -0.26. Los kilos de lana también muestran una correlación negativa débil de -0.17 con la cantidad de lluvia, lo que sugiere que a medida que aumenta la cantidad de lluvia en el año, la producción de lana y existencias tiende a disminuir ligeramente. En el caso de la finura se observa una correlación negativa con la cantidad de lluvia y para el rinde de la lana, se observa una correlación positiva con la cantidad de lluvia. La correlación entre la finura de la lana y la cantidad de lluvia es negativa muy débil, con un valor de -0.052, mientras que la correlación entre el rinde de la lana y la cantidad de lluvia es positiva moderada, con un valor de 0.52; esto significa que a medida que aumenta la cantidad de lluvia, el rinde también tienden a aumentar. Es importante tener en cuenta que la correlación no implica causalidad y que existen otros factores que pueden influir en esta relación.

Variables independientes

A continuación lo que se va a hacer es explicar el funcionamiento de los gráficos que se van a usar para el análisis y después vamos a graficar para cada una de las variables que se van a utilizar en los entrenamientos.

Gráficos

Histogramas

Para saber si los datos están sesgados, es necesario analizar la distribución de los datos. Una distribución sesgada indica que los datos se concentran en un lado de la distribución, lo que puede afectar la interpretación de los resultados.

Se pueden visualizar las distribuciones utilizando histogramas o gráficos de densidad para identificar visualmente si hay algún tipo de sesgo en los datos.

El comando `sns.histplot(data=datos['variable_X'])` te muestra un histograma de la columna 'variable_X' del dataframe 'datos'. En el eje `x` se encuentra el rango de valores de la variable y en el eje `y` se encuentra la frecuencia de ocurrencia de cada valor. Este tipo de gráfico es útil para visualizar la distribución de los datos y para detectar posibles valores atípicos o sesgos en la distribución.

Diagrama de caja

También, para detectar valores atípicos, se puede utilizar la visualización de un diagrama de caja (boxplot), donde se pueden identificar los valores que se encuentran por encima o por debajo de los

límites del diagrama. En un boxplot, los valores atípicos se representan como puntos más allá de los límites de los bigotes del diagrama.

El comando `sns.boxplot(data=datos['variable_X'])` muestra un diagrama de caja (boxplot) que resume la distribución de los valores en la columna 'variable_X' del DataFrame 'datos'. La línea horizontal dentro de la caja representa la mediana, mientras que la caja en sí representa el rango intercuartil (IQR), es decir, el rango de valores que se encuentran entre el primer y tercer cuartil de la distribución. Los puntos fuera de la caja, llamados valores atípicos, representan valores que están más allá del IQR. Este gráfico puede ser útil para identificar valores extremos o fuera de lo común en la distribución de los datos.

Diagrama de dispersión

Con `sns.scatterplot(data=datos['variable_X'])` se mostrará un gráfico de dispersión con el eje *X* representando los índices de las filas del dataframe y el eje *Y* representando los valores de la columna 'variable_X'. Cada punto en el gráfico representa un valor de 'variable_X' para una fila específica del dataframe. El gráfico puede ser útil para visualizar la distribución y la relación entre los valores de la columna 'variable_X'.

Diagrama de línea

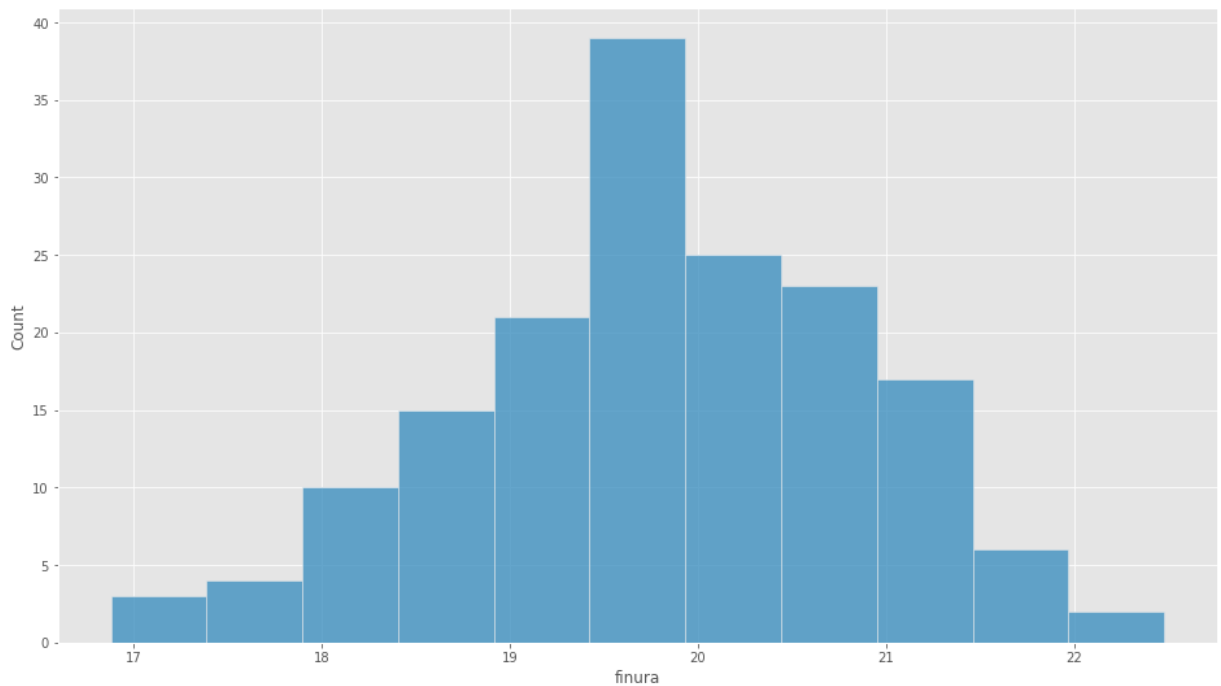
El código `sns.lineplot(data=datos['variable_X'])` genera un gráfico de línea de la columna 'variable_X' del DataFrame 'datos'. El eje *x* del gráfico representa el índice de la serie (es decir, el número de fila), y el eje *y* representa el valor de la columna 'variable_X' en cada fila. Por defecto, Seaborn dibuja una línea que conecta los puntos en orden, lo que ayuda a visualizar la tendencia a lo largo del tiempo (o cualquier otra dimensión representada por el índice).

Finura

El histograma presentado muestra que la variable `finura` presenta una distribución casi simétrica.

```
In [8]: sns.histplot(data=datos['finura'])
```

```
Out[8]: <AxesSubplot:xlabel='finura', ylabel='Count'>
```



Podemos observar en el histograma una leve asimetría hacia la izquierda, lo que se conoce como una distribución negativamente sesgada. Esta asimetría se caracteriza por la presencia de barras más altas en el lado derecho y barras más bajas en el lado izquierdo, lo que sugiere la presencia de valores extremos o atípicos que aumentan el promedio y desplazan la distribución hacia la izquierda. En otras palabras, la mayor concentración de valores se encuentra en el lado derecho, mientras que el lado izquierdo muestra una menor cantidad de valores.

En el siguiente boxplot, vemos más claramente los datos extremos para la `finura` que nos indicaba el histograma.

```
In [9]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['finura'], orient="h")

# Eliminar el label del eje Y
ax.set_yticks([])

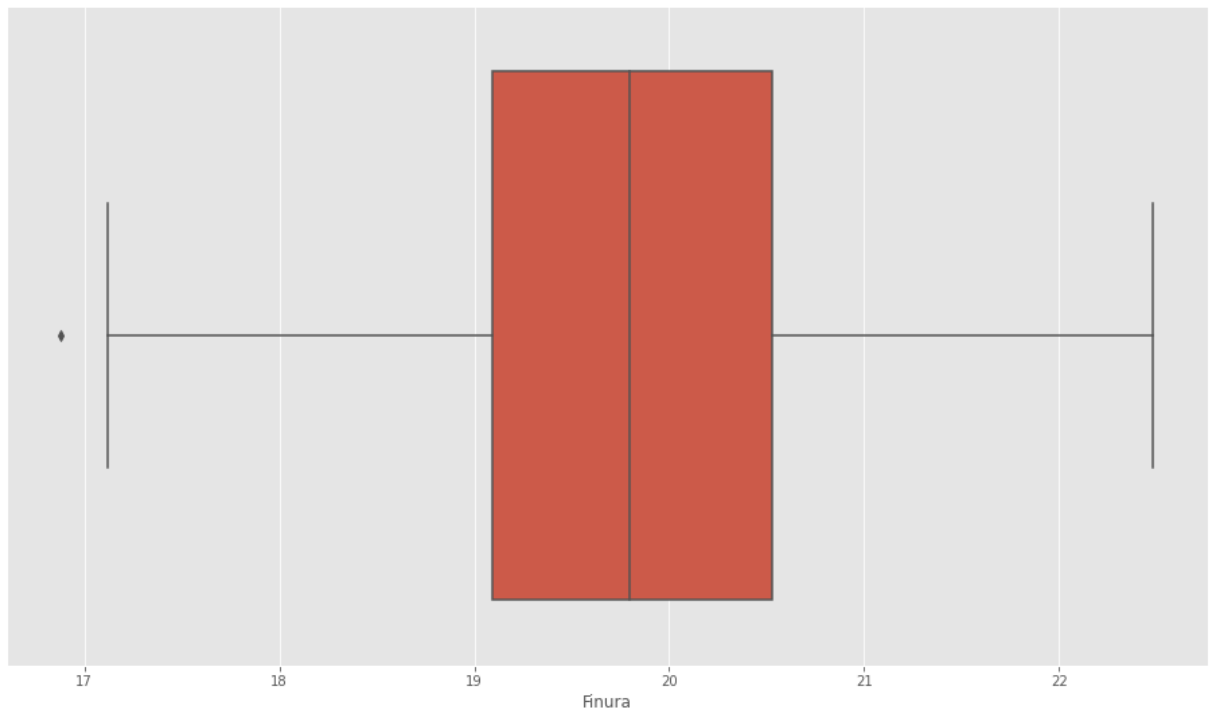
# Agregar un label al eje X
ax.set_xlabel('Finura')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 17.12

Valor del bigote superior: 22.48



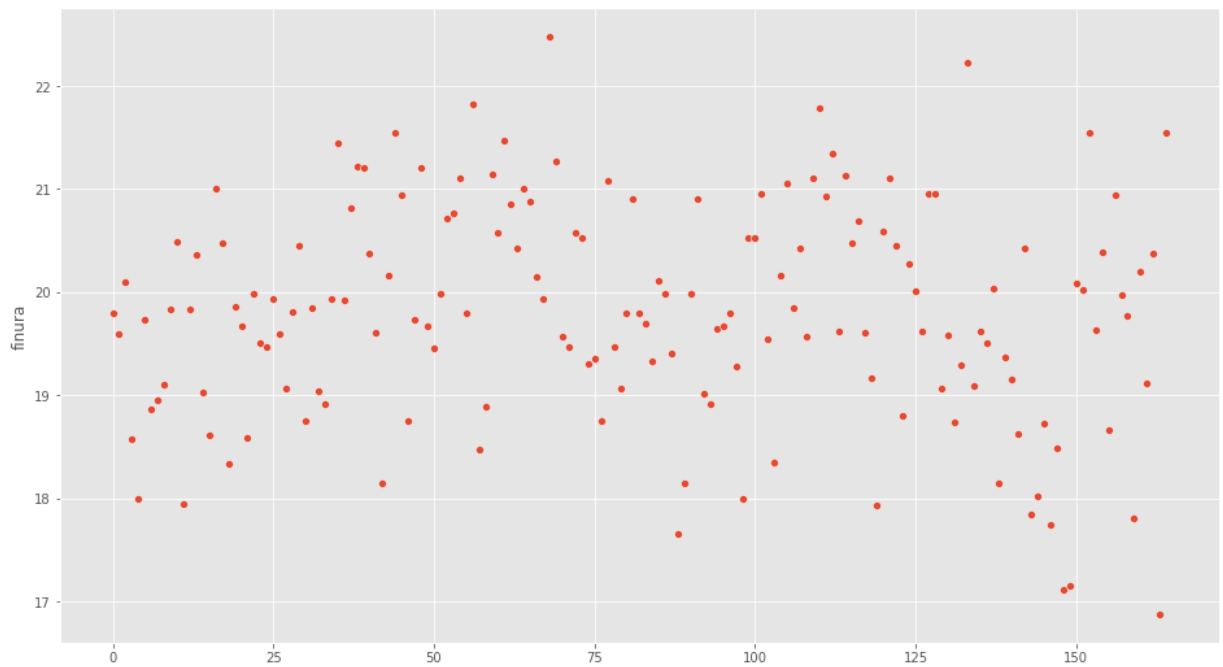
Explicación del gráfico de boxplot observado:

- Primer cuartil: el 25% de los valores son menores o iguales a 19.09 de finura.
- Segundo cuartil: el 50% de los valores son menores o iguales a 19.80 de finura.
- Tercer cuartil: el 75% de los valores son menores o iguales a 20.53 de finura.
- Mediana: a mediana de la finura de la lana es 19.80 y divide la caja en dos partes casi iguales. Sin embargo, debido a que la parte inferior de la caja es levemente más larga que la superior, se observa una asimetría negativa o sesgo hacia la izquierda. Esto indica que la mayoría de las observaciones se concentran en valores más altos de finura, y hay una pequeña cantidad de valores extremos más bajos que empujan la mediana hacia la izquierda.
- Bigotes: el límite inferior es de 17.12 y el límite superior es de 22.48.
- Valores atípicos: hay un valor que supera el límite inferior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el total de ovinos se distribuyen en un plano cartesiano.

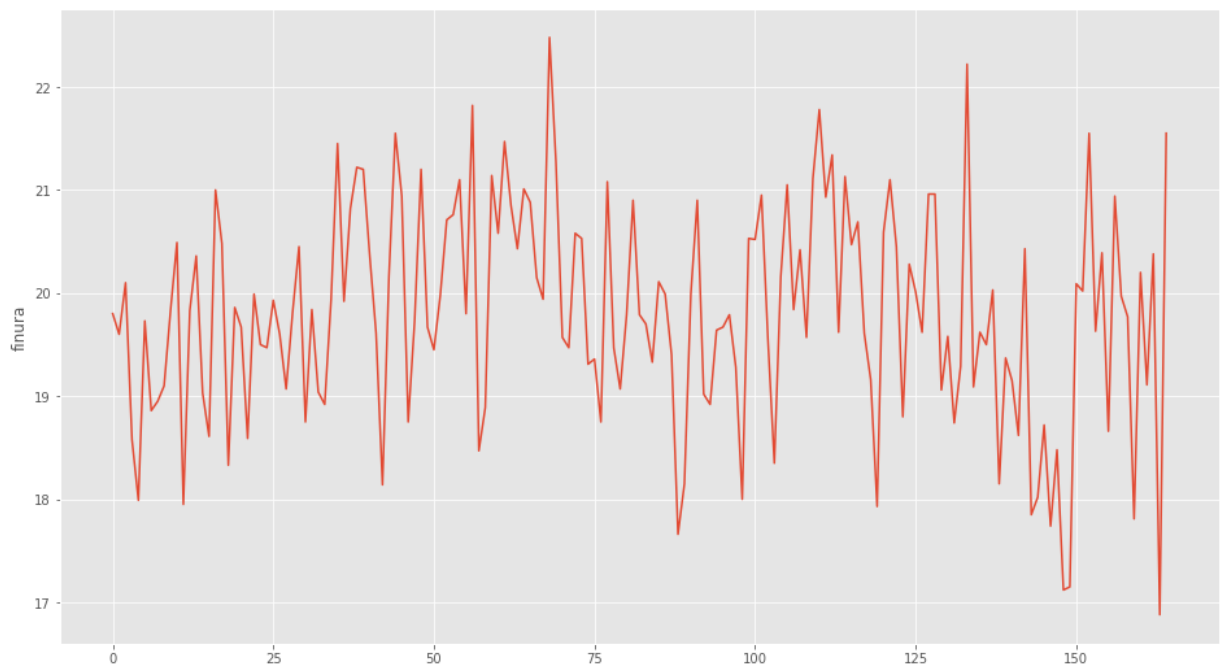
```
In [10]: sns.scatterplot(data=datos['finura'])
```

```
Out[10]: <AxesSubplot:ylabel='finura'>
```



```
In [11]: sns.lineplot(data=datos['finura'])
```

```
Out[11]: <AxesSubplot:ylabel='finura'>
```



En el gráfico de dispersión, se puede apreciar que no existe una clara relación entre las variables representadas en el plano cartesiano. La disposición de los puntos no sigue un patrón definido, como una línea recta o agrupaciones relevantes. Por otro lado, el gráfico de línea muestra cómo la variación en la finura de la lana cambia de una observación a otra, pero no se observa una tendencia clara. Es decir, no se puede afirmar que haya un aumento o disminución constante en la finura de la lana en las diferentes observaciones.

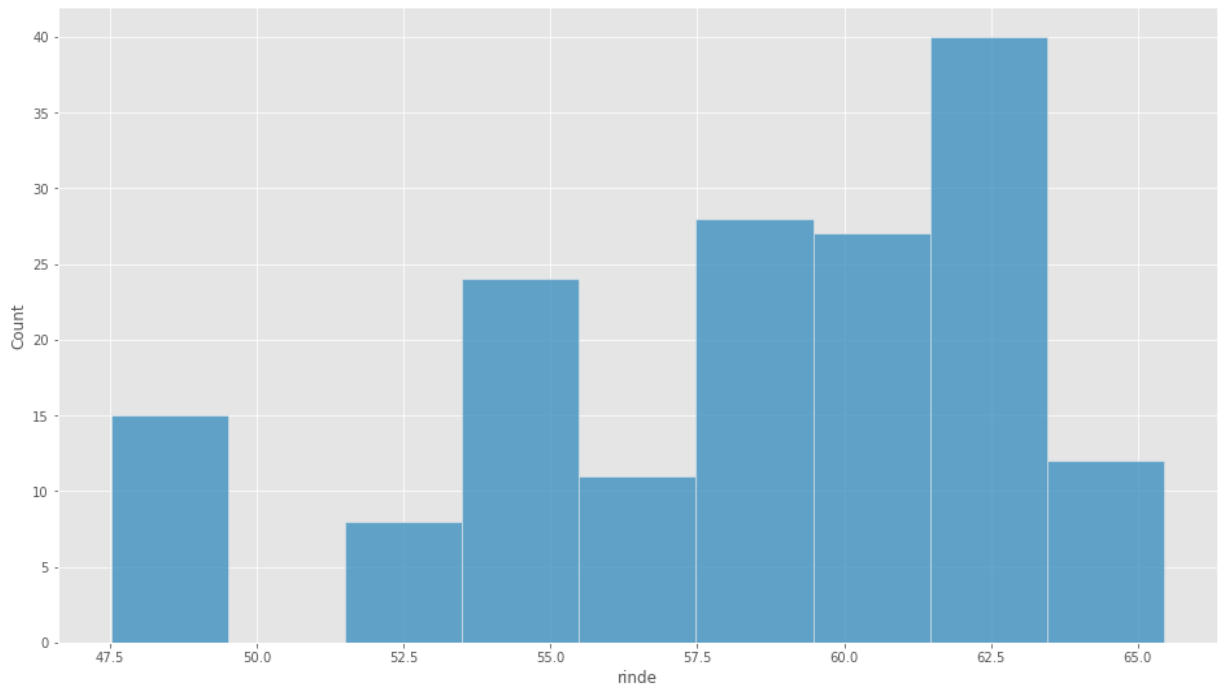
Es importante tener en cuenta que aunque no se detecte una tendencia lineal en los datos, podría existir una relación no lineal o compleja entre las variables que no es evidente en estos gráficos.

Rinde

El histograma presentado muestra que la variable `rinde` presenta una distribución asimétrica.

```
In [12]: sns.histplot(data=datos['rinde'])
```

```
Out[12]: <AxesSubplot:xlabel='rinde', ylabel='Count'>
```



Podemos hacer varias observaciones:

- La distribución tiene un sesgo hacia la izquierda.
- El valor máximo de la variable es 65.5 y el mínimo es 47.5.
- La mayoría de los valores se encuentran en el rango de 53 a 63 en el eje x, donde se concentran las barras más altas del histograma.
- En el extremo izquierdo del histograma hay una única barra correspondiente al intervalo 47.5-49.5 en el eje x, que tiene una frecuencia baja de observaciones.

En el siguiente boxplot, vemos más claramente la asimetría para el `rinde` que nos indicaba el histograma.

```
In [13]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['rinde'], orient="h")

# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Rinde')

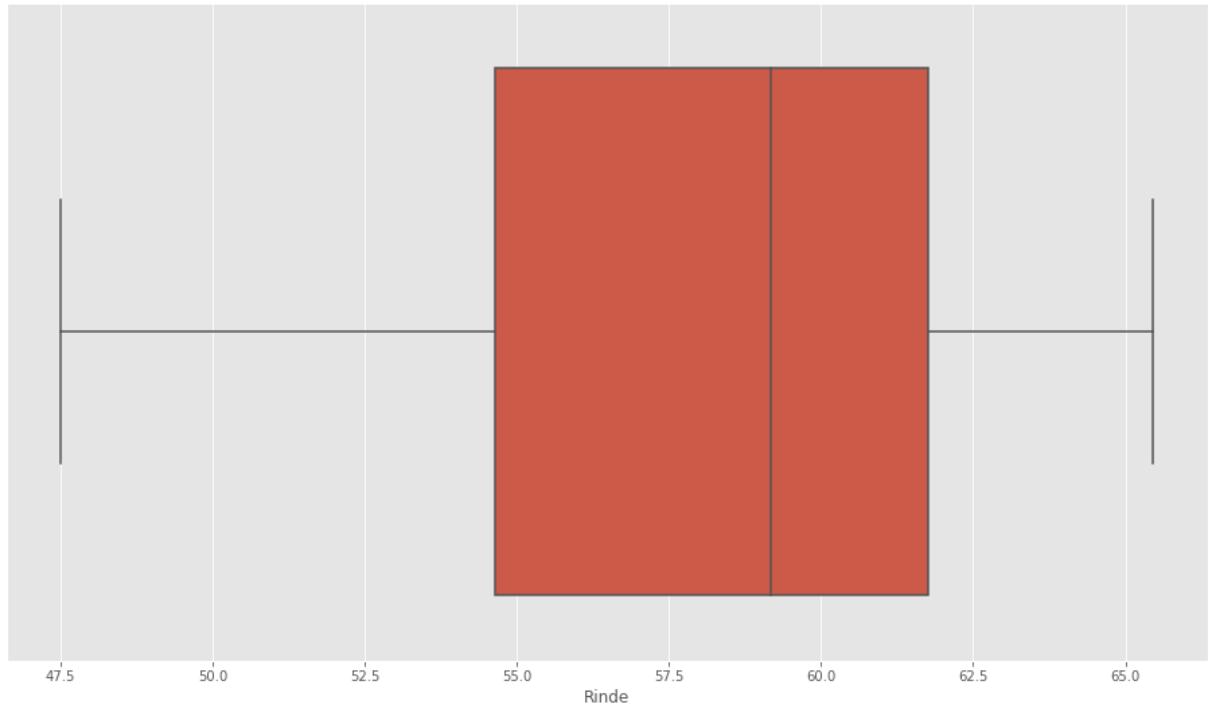
# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
```

```
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 47.51

Valor del bigote superior: 65.45



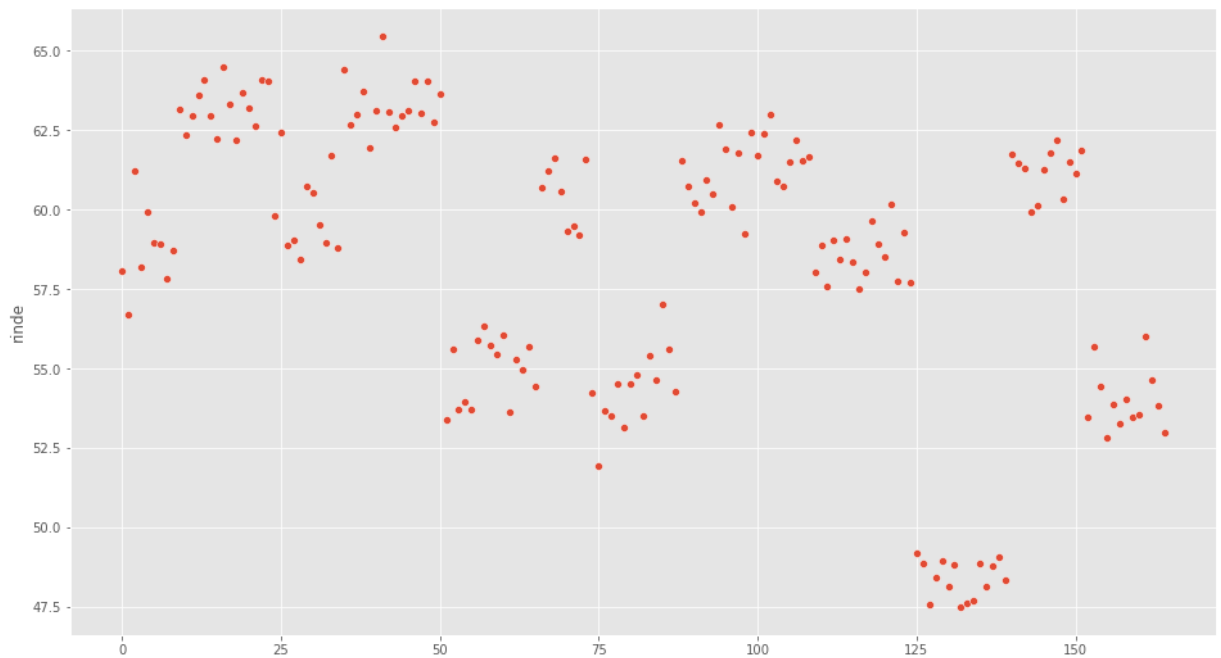
Explicación gráfico boxplot observado:

- Primer cuartil: el 25% de los valores son menores o igual a 54.64 de rinde.
- Segundo Cuartil: el 50% de los valores son menores o igual a 59.18 de rinde.
- Tercer cuartil: el 75% de los valores son menores o igual a 61.77 de rinde.
- Mediana: la mediana es 59.18 de rinde y corta la caja en dos lados desiguales, muestra una asimetría negativa o segada a la izquierda dado que la parte más larga de la caja es la parte inferior a la mediana, lo que indica que los valores más bajos son más frecuentes que los valores más altos.
- Bigotes: el limite inferior es de 47.51 y el limite superior es de 65.45.
- Valores atípicos: no hay valores que superen los limites.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el `rinde` se distribuyen en un plano cartesiano.

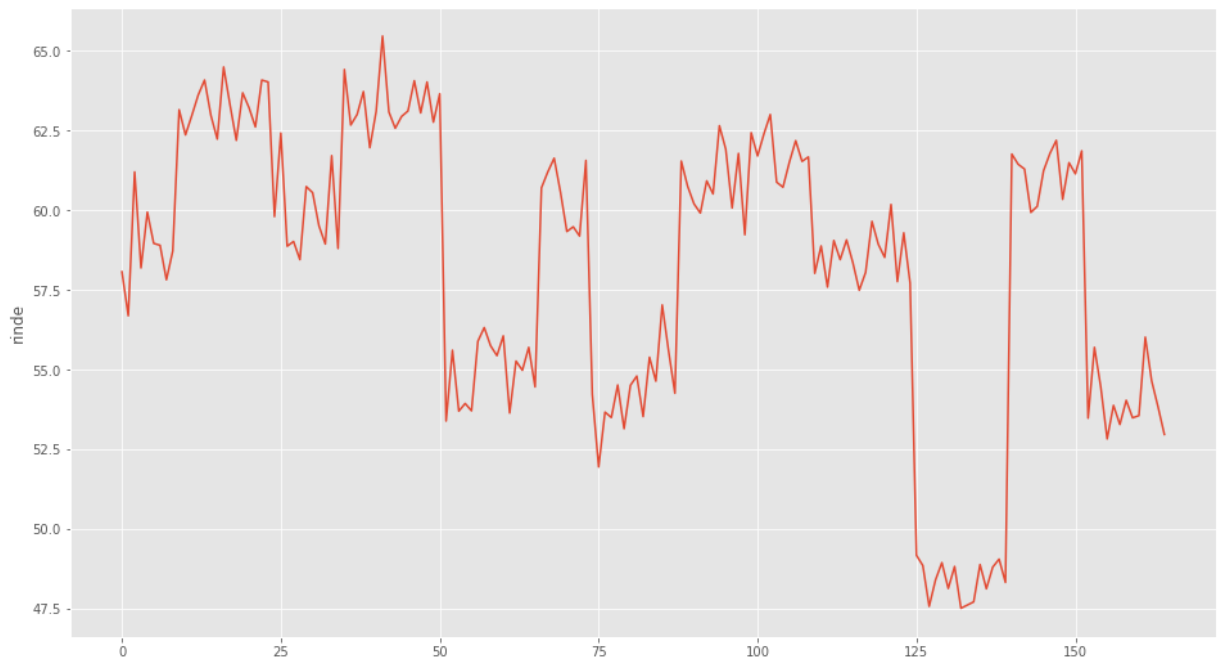
```
In [14]: sns.scatterplot(data=datos['rinde'])
```

```
Out[14]: <AxesSubplot:ylabel='rinde'>
```



```
In [15]: sns.lineplot(data=datos['rinde'])
```

```
Out[15]: <AxesSubplot:ylabel='rinde'>
```



Al examinar el diagrama de dispersión, se observa que no existe una relación evidente entre las variables representadas en el plano cartesiano. La ubicación de los puntos no sigue una pauta definida, como una línea recta, aunque se pueden identificar once pequeños conjuntos que podrían ser interesantes. En cambio, el gráfico de línea ilustra cómo el rinde de la lana varía de una observación a otra, cada vez que el rinde sube los picos son similares, pero en general no se distingue una tendencia definida. Por lo tanto, no se puede concluir que haya una tendencia constante al alza o a la baja en el rinde en las observaciones.

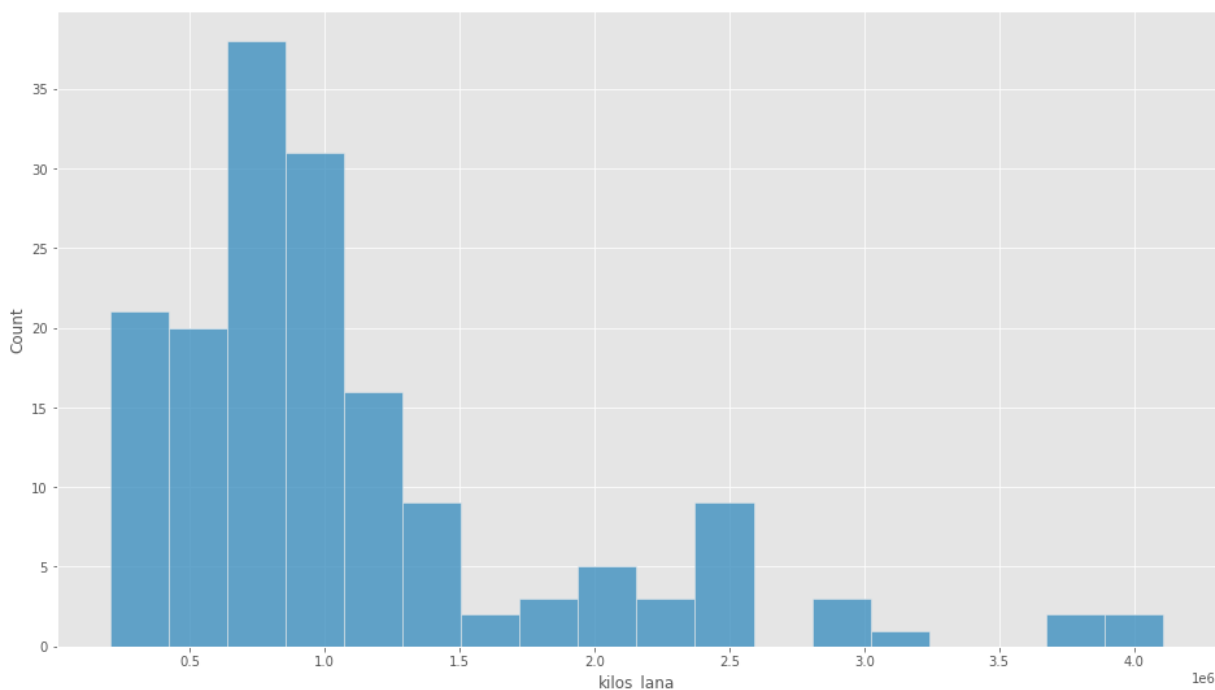
Es importante tener en cuenta que aunque no se detecte una tendencia lineal en los datos, podría existir una relación no lineal o compleja entre las variables que no es evidente en estos gráficos.

Kilogramos de lana

En el siguiente histograma vemos que los `kilos_lana` tiene una distribución asimétrica.

```
In [16]: sns.histplot(data=datos['kilos_lana'])
```

```
Out[16]: <AxesSubplot:xlabel='kilos_lana', ylabel='Count'>
```



En el histograma que se observa hay un total de 15 barras. De esas barras, 8 van desde el valor cero hasta cinco, y están todas situadas en el lado derecho de la distribución. Las siete barras restantes representan valores más altos, y se ubican en el lado izquierdo del gráfico a excepción de una. En este caso, la distribución está sesgada positivamente, lo que significa que hay una cola larga que se extiende hacia la derecha de la distribución, y que se deben considerar los valores atípicos en ese extremo. Esta asimetría se produce porque hay más valores en el lado izquierdo de la distribución que en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el **total de kilogramos de lana** que nos indicaba el histograma.

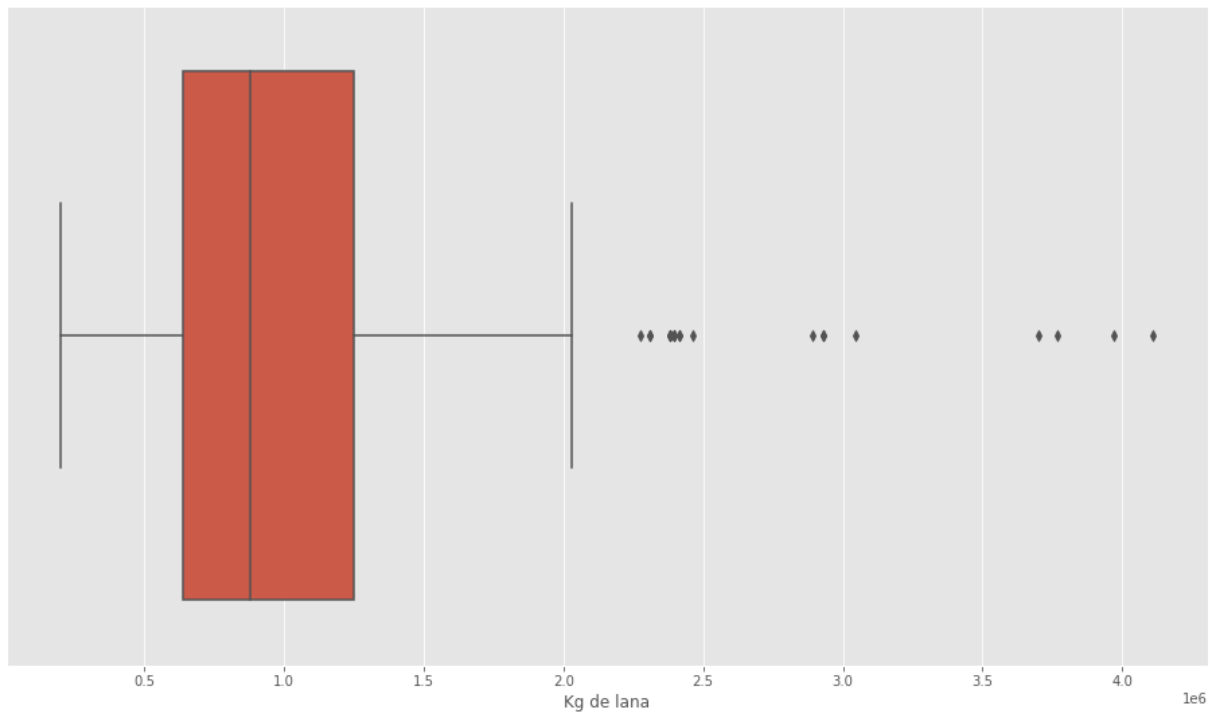
```
In [17]: ax = sns.boxplot(data=datos['kilos_lana'], orient="h")

# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Kg de lana')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)
```

Valor del bigote inferior: 201856.0
Valor del bigote superior: 2030400.0

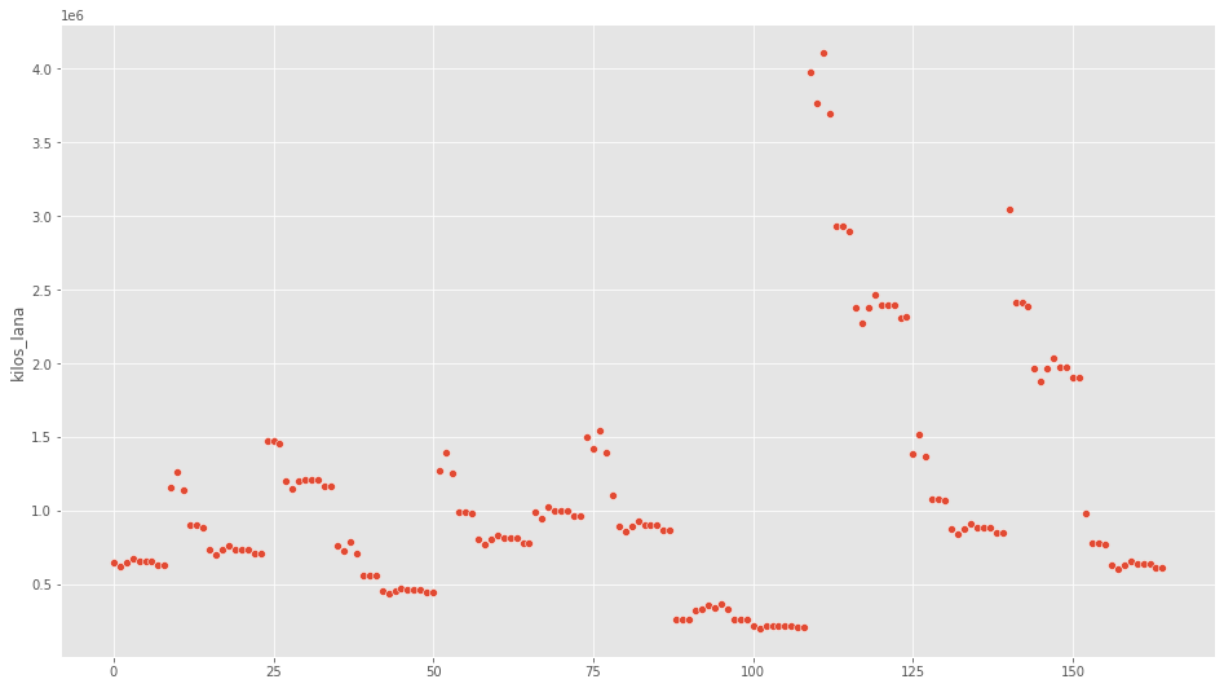


Se puede observar en el gráfico de boxplot que los datos están expresados en notación científica con una escala de millones, ya que los datos originales se transformaron de toneladas a kilogramos. El primer cuartil, que representa el 25% de los datos, tiene un valor de 637000 kg de lana o menos, mientras que el segundo cuartil, que representa el 50% de los datos, tiene un valor de 877090 kg de lana o menos. El tercer cuartil, que representa el 75% de los datos, tiene un valor de 1250640 kg de lana o menos. La mediana, tiene un valor de 877090 kg de lana y muestra una asimetría positiva o sesgada hacia la derecha, ya que la parte superior de la caja es más larga que la inferior, lo que indica que los datos se concentran en la parte superior de la distribución. Los bigotes, que representan el rango de los datos, tienen un límite inferior de 201856 y un límite superior de 2030400. Además, se identifican 14 valores atípicos que superan el límite superior, lo que indica la presencia de varios valores extremos en la distribución de los datos.

En los siguientes gráficos de dispersión y de línea, vemos como los datos para el `total de lana` se distribuyen en un plano cartesiano.

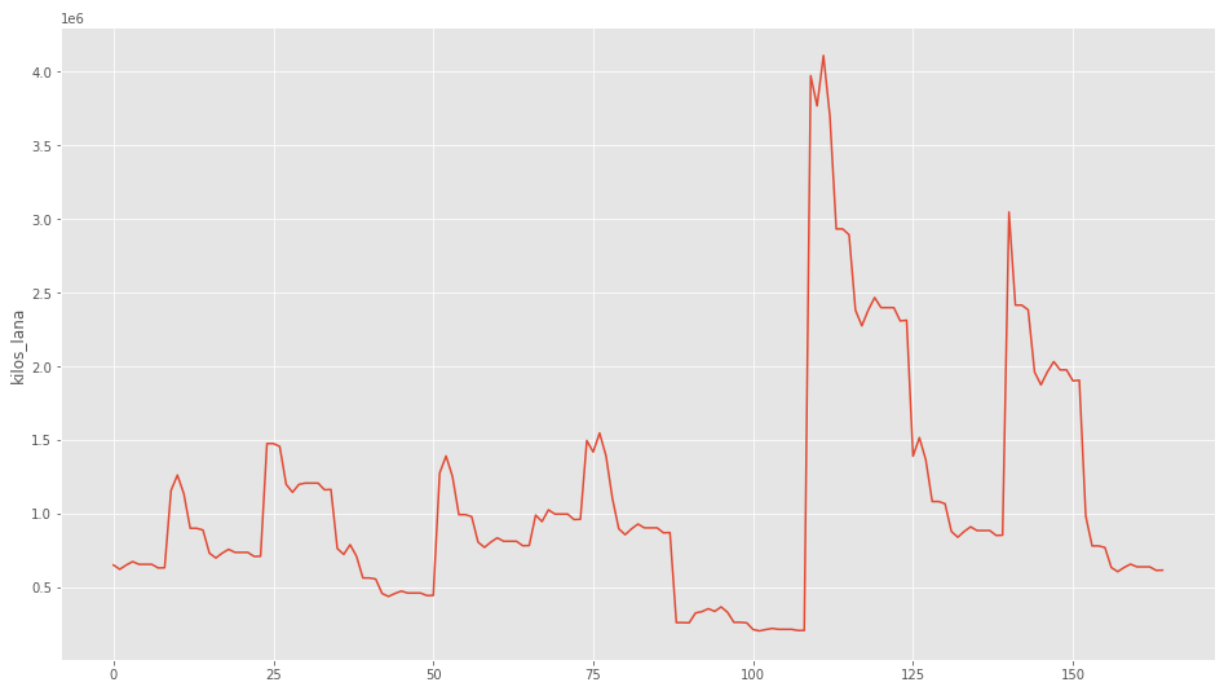
```
In [18]: sns.scatterplot(data=datos['kilos_lana'])
```

```
Out[18]: <AxesSubplot:ylabel='kilos_lana'>
```



```
In [19]: sns.lineplot(data=datos['kilos_lana'])
```

```
Out[19]: <AxesSubplot:ylabel='kilos_lana'>
```



Al analizar los datos representados en el gráfico de dispersión, se observa que los datos parecen tener una distribución dispersa sin un patrón o tendencia clara entre las variables. Sin embargo, se pueden identificar tres grupos grandes que podrían ser relevantes.

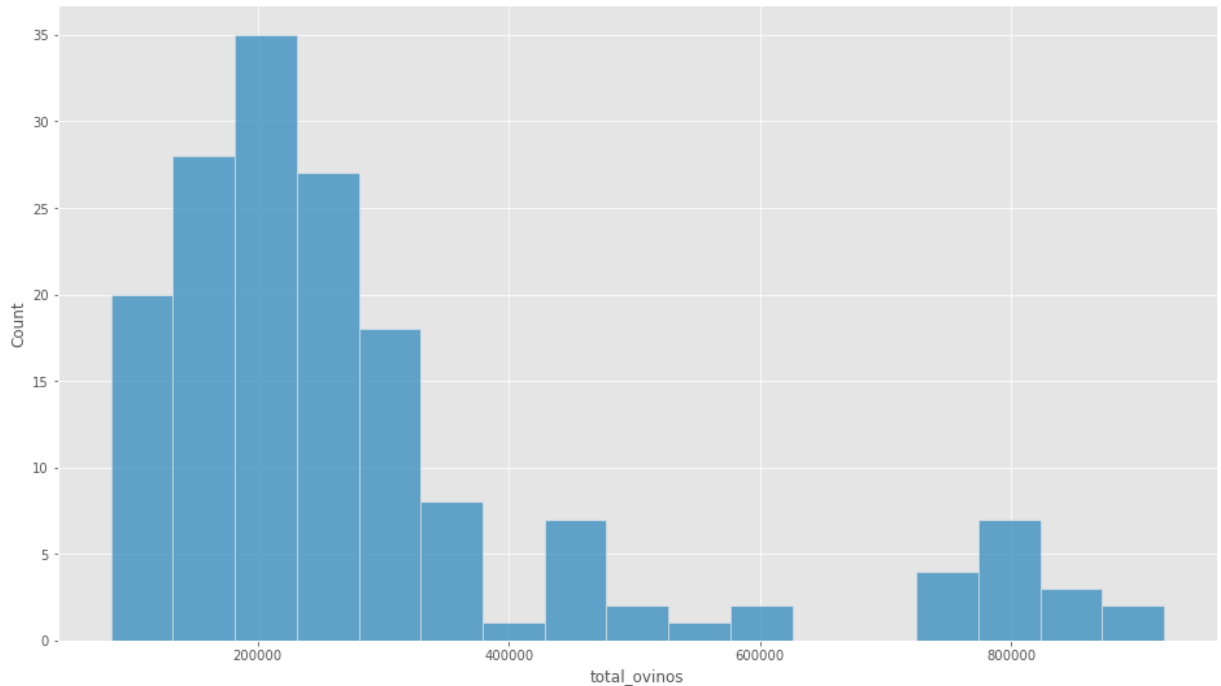
Al observar el gráfico de línea de los kilogramos de lana, se puede notar que hay una variabilidad en la producción, representada por los picos en los datos. Esto podría indicar una producción fluctuante o con una gran variabilidad. También se observa que cada vez que baja y sube la cantidad de lana los picos tienen una forma similar, sin embargo, no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Total de ovinos

En el siguiente histograma vemos que el `total_ovinos` tiene una distribución asimétrica.

```
In [20]: sns.histplot(data=datos['total_ovinos'])
```

```
Out[20]: <AxesSubplot:xlabel='total_ovinos', ylabel='Count'>
```



El gráfico representa un histograma con una distribución asimétrica derecha, lo que se puede ver debido a la presencia de barras más altas en el lado izquierdo y barras más bajas en el lado derecho. Esta distribución también se conoce como una distribución positivamente sesgada. El sesgo positivo sugiere que la cola de la distribución se extiende hacia la derecha, lo que indica la presencia de valores extremos o atípicos en el lado derecho de la distribución que aumentan el promedio y empujan la distribución hacia la derecha.

En el siguiente boxplot, vemos más claramente los datos extremos para el `total de ovinos` que nos indicaba el histograma.

```
In [21]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['total_ovinos'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 960000, 50000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

# Eliminar el label del eje Y
ax.set_yticks([])

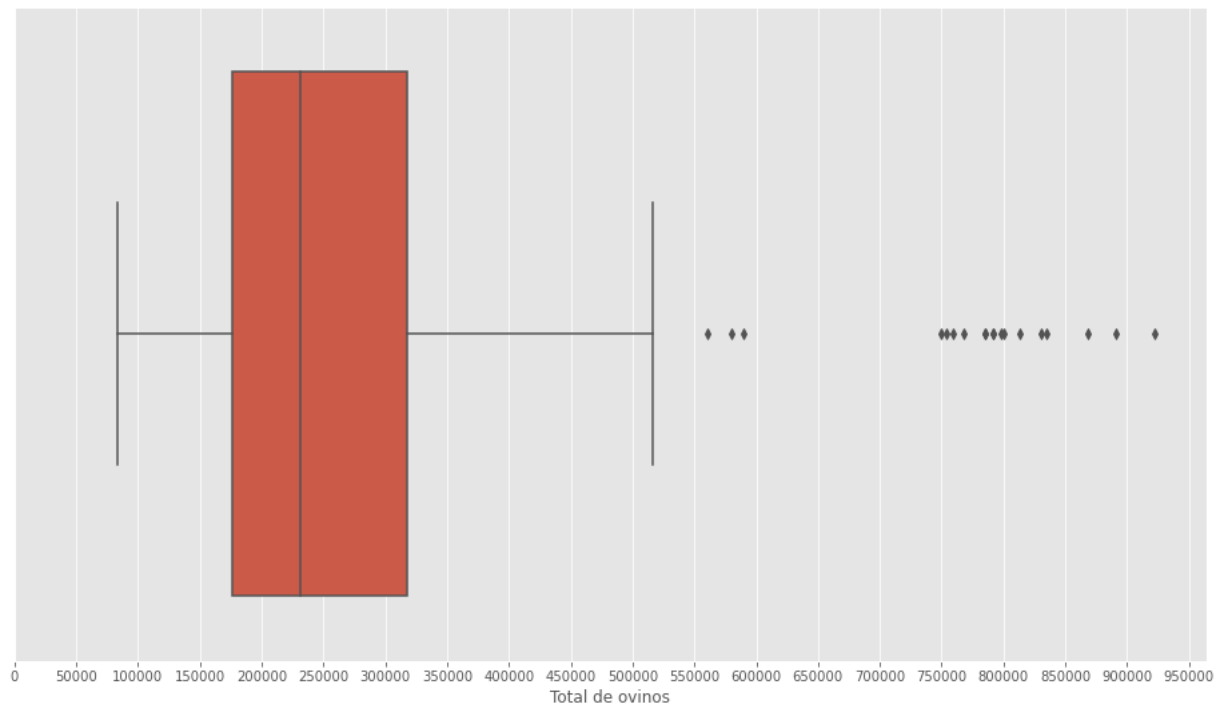
# Agregar un label al eje X
ax.set_xlabel('Total de ovinos')
```

```
# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 82956.0

Valor del bigote superior: 516771.0



El gráfico boxplot nos proporciona una representación visual de la distribución de los datos. Los valores del primer cuartil, el segundo cuartil (o mediana) y el tercer cuartil nos indican la posición relativa de los datos en la distribución. En este caso, el primer cuartil muestra que el 25% de los valores son menores o iguales a 176065 ovinos, mientras que el tercer cuartil nos indica que el 75% de los valores son menores o iguales a 317981 ovinos.

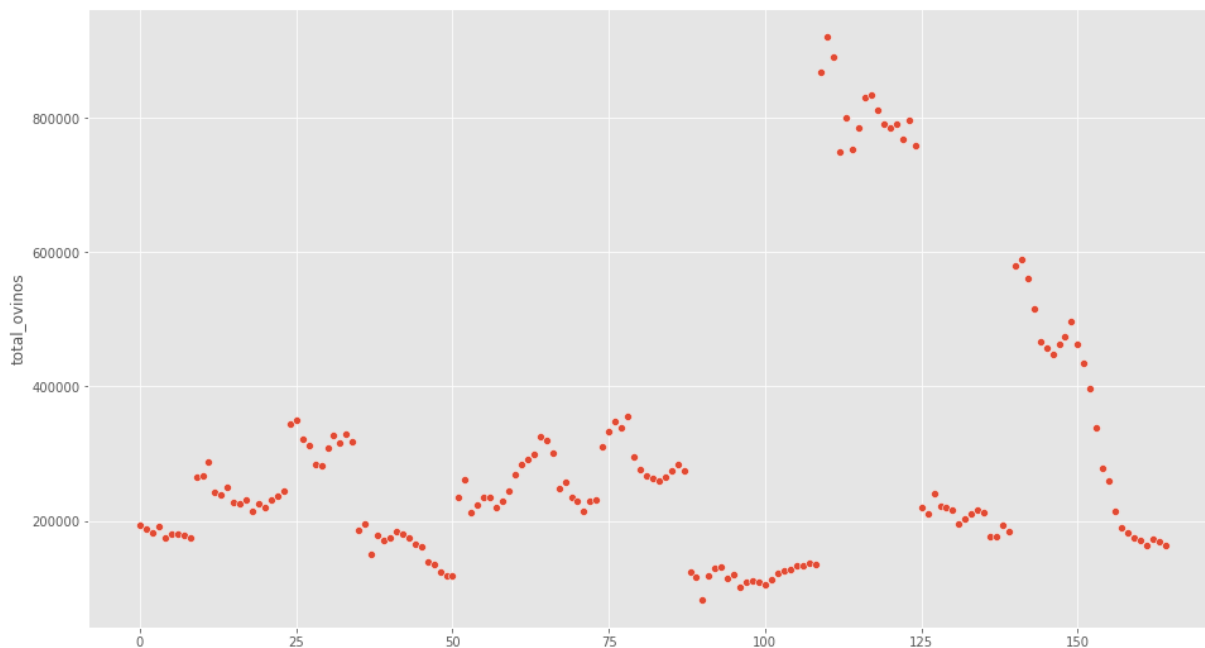
La mediana, que es 230802 ovinos, divide la caja en dos partes desiguales, mostrando una asimetría positiva o sesgada hacia la derecha, dado que la mayor parte de los datos se concentran en la parte superior de la distribución.

Los bigotes del gráfico indican los valores mínimo y máximo dentro de un rango aceptable, que se extiende desde el límite inferior de 82956 hasta el límite superior de 516771. Además, el gráfico muestra que hay 17 valores atípicos que superan el límite superior, lo que indica que hay algunas observaciones que están muy por encima de la mayoría de los datos y que podrían ser considerados como excepciones o datos extremos.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de ovinos** se distribuyen en un plano cartesiano.

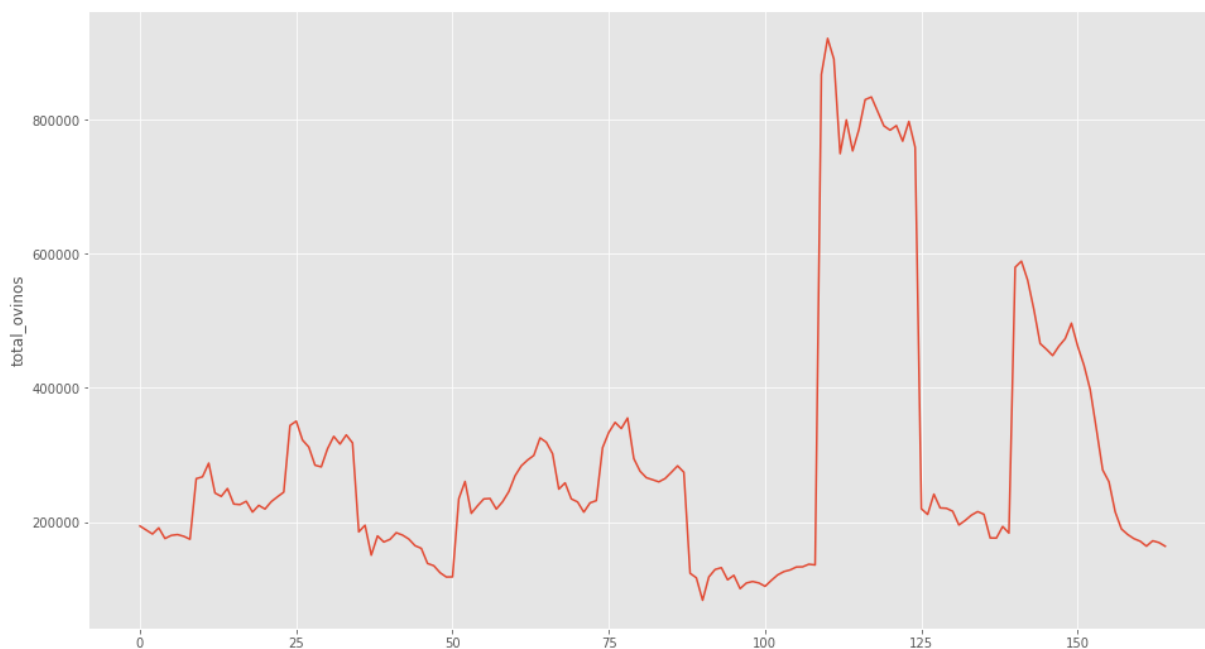

```
In [22]: sns.scatterplot(data=datos['total_ovinos'])
```

```
Out[22]: <AxesSubplot:ylabel='total_ovinos'>
```



```
In [23]: sns.lineplot(data=datos['total_ovinos'])
```

```
Out[23]: <AxesSubplot:ylabel='total_ovinos'>
```



Al examinar el diagrama de dispersión, no se puede distinguir ninguna tendencia o patrón definido entre las variables representadas, aunque se pueden identificar seis grupos grandes que podrían ser relevantes, no hay una disposición clara de los puntos, como una línea recta o un grupo bien definido.

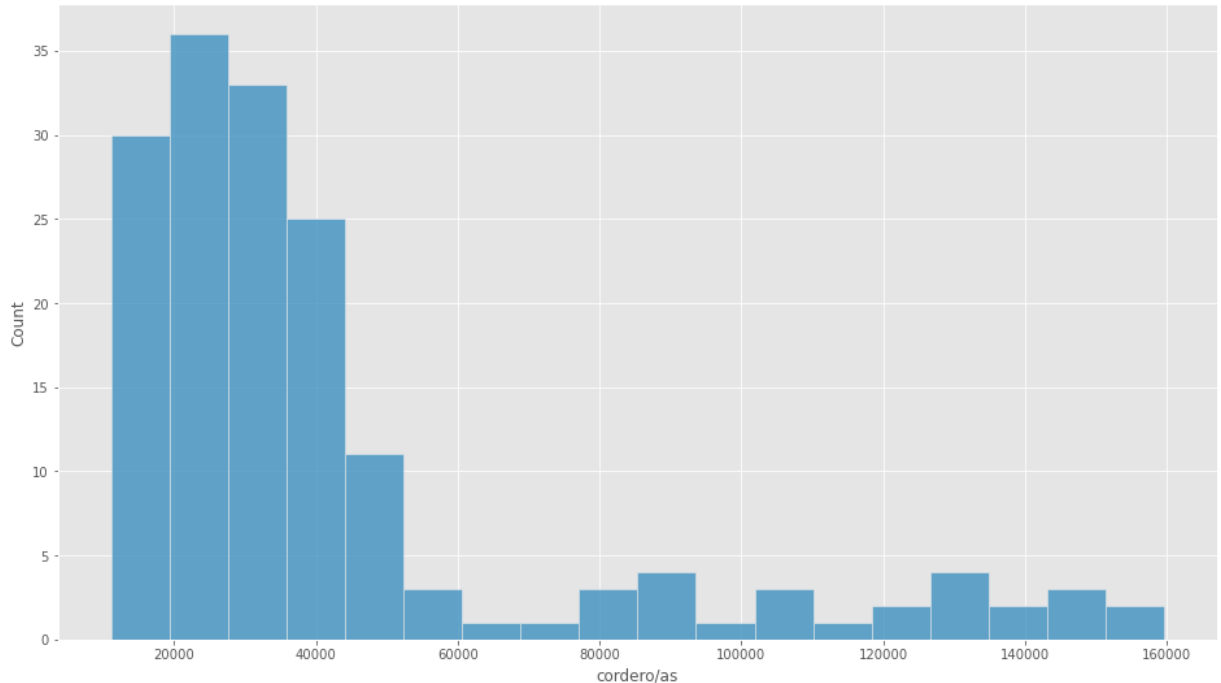
Por otro lado, al observar el gráfico de línea que representa la producción total de ovinos, se puede notar que existe una variabilidad en la producción, lo que se refleja en los picos de los datos. Esto podría indicar que la población ovina tiene fluctuaciones o una variabilidad relativamente alta. No se observa ninguna tendencia clara, por lo que no es posible afirmar que exista un aumento o disminución constante en las observaciones.

Corderos

En el siguiente histograma vemos que el `cordero/as` tiene una distribución asimétrica.

```
In [24]: sns.histplot(data=datos['cordero/as'])
```

```
Out[24]: <AxesSubplot:xlabel='cordero/as', ylabel='Count'>
```



En un histograma, una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho indica una distribución asimétrica derecha, también conocida como una distribución positivamente sesgada. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el `total de corderos` que nos indicaba el histograma.

```
In [25]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['cordero/as'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 170000, 10000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

# Eliminar el label del eje Y
ax.set_yticks([])
```

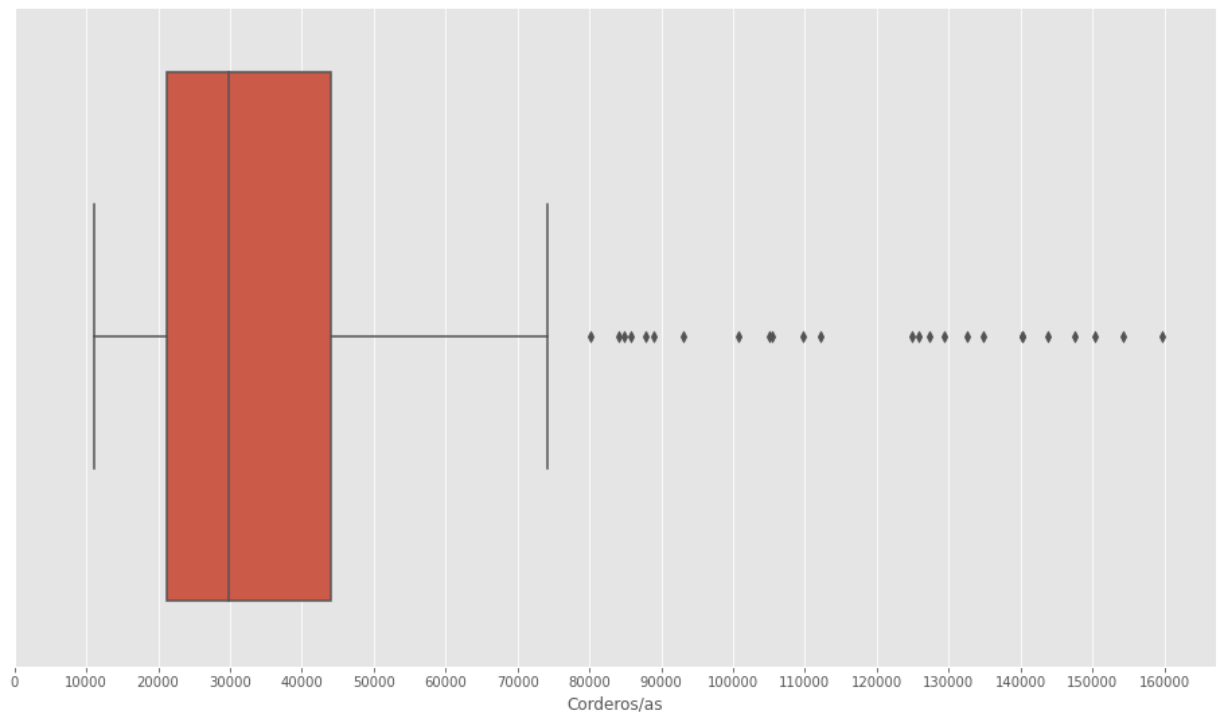
```
# Agregar un label al eje X
ax.set_xlabel('Corderos/as')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 11071.0

Valor del bigote superior: 74093.0



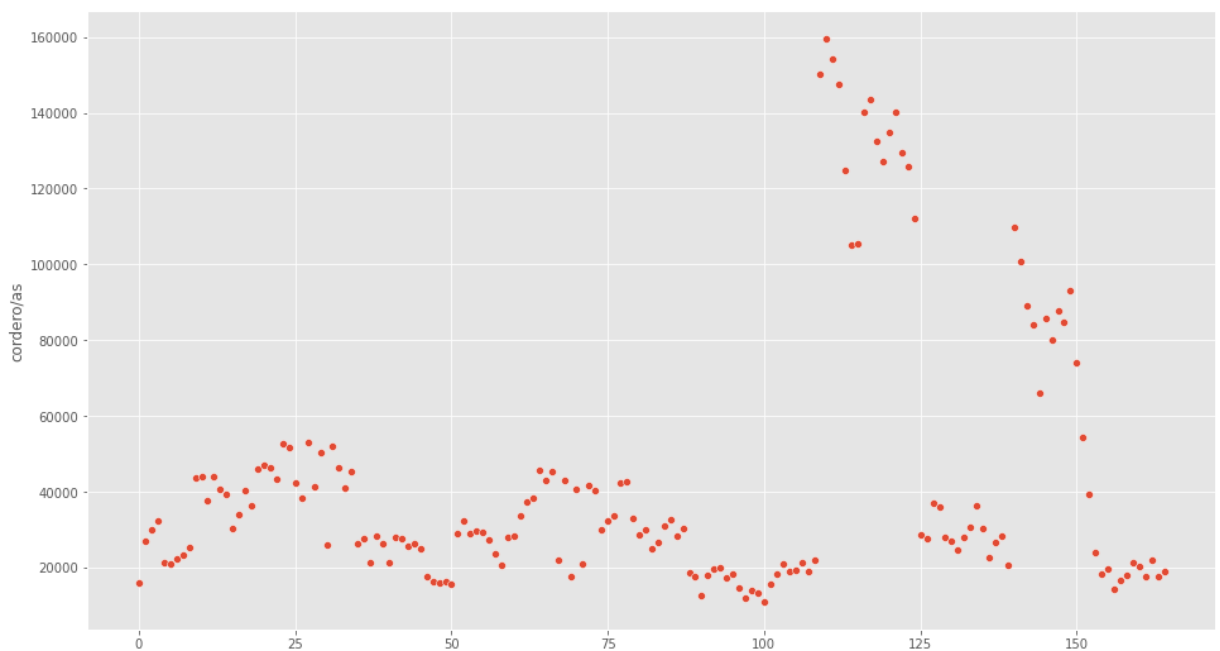
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 21294 corderos/as.
- Segundo Cuartil: el 50% de los valores son menores o igual a 29740 corderos/as.
- Tercer cuartil: el 75% de los valores son menores o igual a 44013 corderos/as.
- Mediana: la mediana es 29740 corderos/as y corta la caja en dos lados desiguales, por lo tanto muestra una asimetría positiva o segada a la derecha dado que la parte más larga de la caja es la parte superior a la mediana.
- Bigotes: el limite inferior es de 11071 y el limite superior es de 74093.
- Valores atípicos: hay 24 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de corderos** se distribuyen en un plano cartesiano.

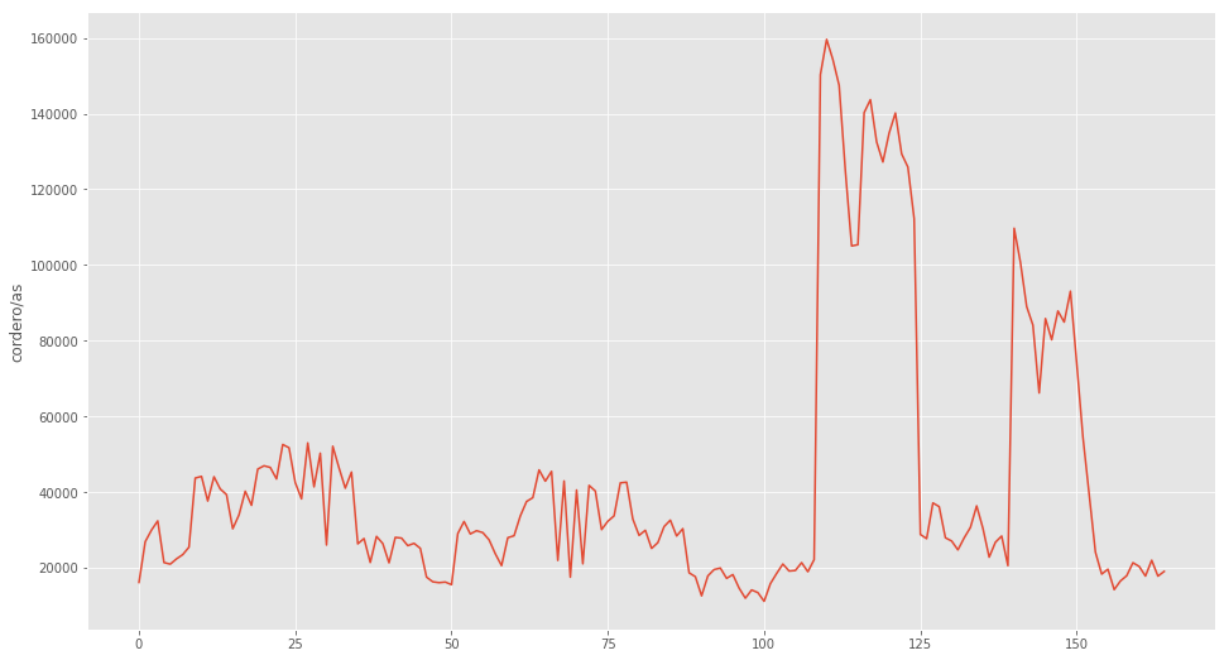
```
In [26]: sns.scatterplot(data=datos['cordero/as'])
```

```
Out[26]: <AxesSubplot:ylabel='cordero/as'>
```



```
In [27]: sns.lineplot(data=datos['cordero/as'])
```

```
Out[27]: <AxesSubplot:ylabel='cordero/as'>
```



Al analizar los datos representados en el gráfico de dispersión, no se puede identificar un patrón o tendencia clara entre las variables. Aunque se pueden distinguir tres grupos grandes que podrían ser relevantes, no hay una disposición definida de los puntos, como una línea recta o agrupaciones bien definidas.

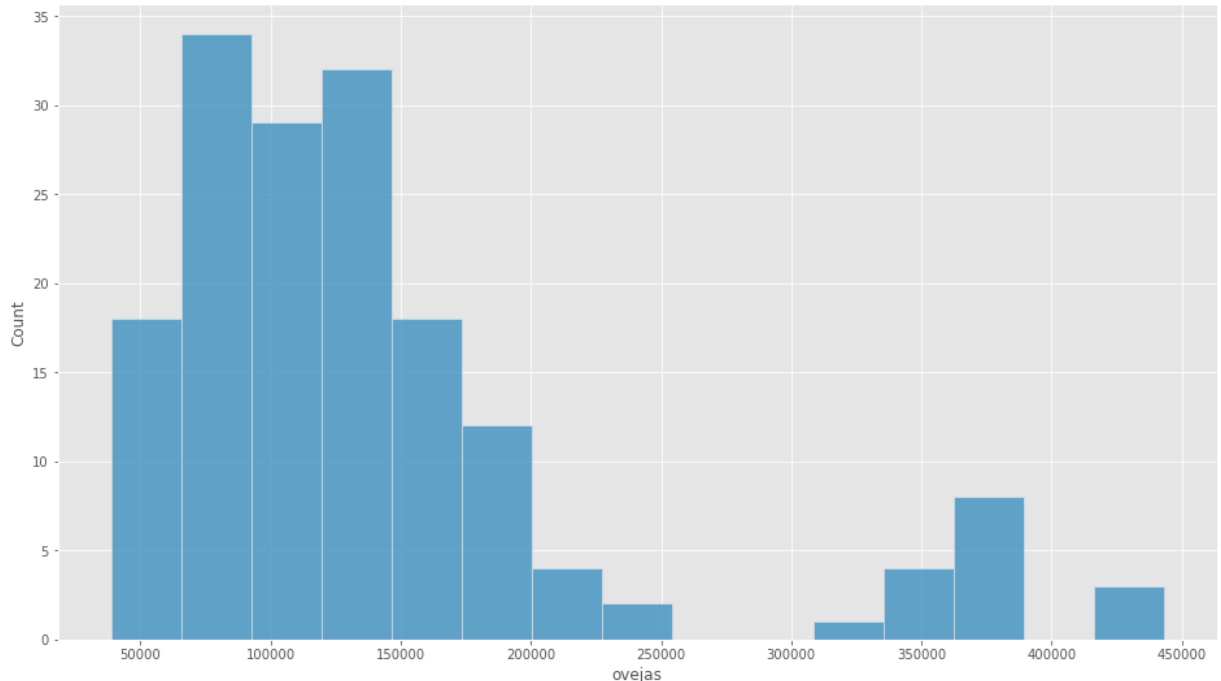
En cuanto al gráfico de línea del total de corderos/as, se puede observar una variabilidad en la producción, representada por los altos en los datos. Sin embargo, la mayoría de los datos se encuentran entre 20000 y 50000 corderos/as, lo que podría indicar una fluctuación moderada. Pero no se aprecia una tendencia clara, por lo que no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Ovejas

En el siguiente histograma vemos que la **ovejas** tienen una distribución asimétrica.

```
In [28]: sns.histplot(data=datos['ovejas'])
```

```
Out[28]: <AxesSubplot:xlabel='ovejas', ylabel='Count'>
```



El histograma muestra una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho, lo que indica una distribución asimétrica derecha. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el **total de ovejas** que nos indicaba el histograma.

```
In [29]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['ovejas'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 460000, 50000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

# Eliminar el label del eje Y
ax.set_yticks([])

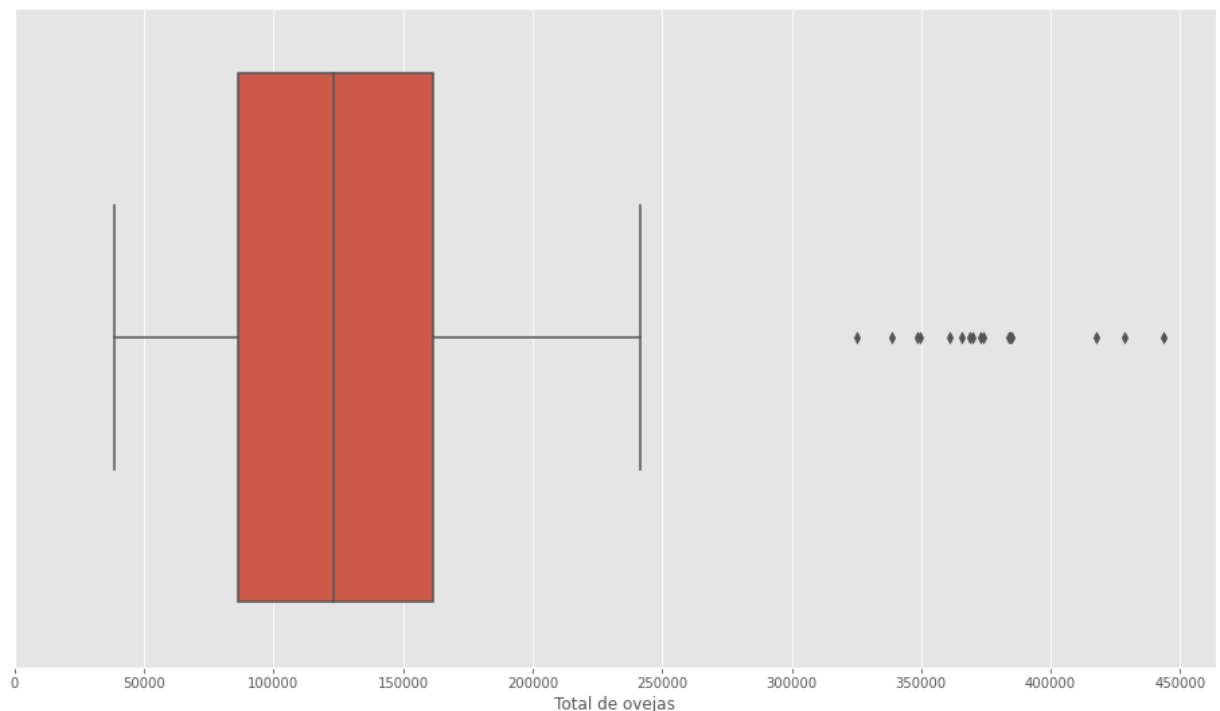
# Agregar un label al eje X
```

```
ax.set_xlabel('Total de ovejas')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 38558.0
Valor del bigote superior: 241352.0



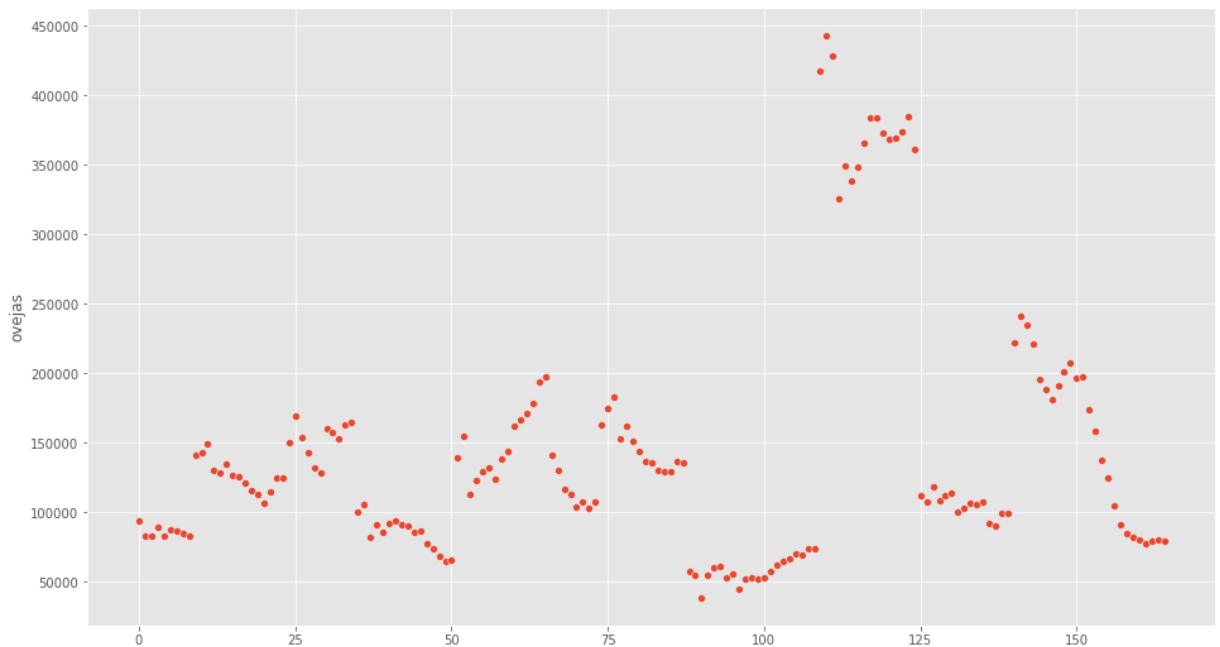
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 86473 ovejas.
- Segundo Cuartil: el 50% de los valores son menores o igual a 123241 ovejas.
- Tercer cuartil: el 75% de los valores son menores o igual a 161713 ovejas.
- Mediana: la mediana es 123241 de ovejas y corta la caja en dos lados casi iguales.
- Bigotes: el limite inferior es de 38558 y el limite superior es de 241352.
- Valores atípicos: hay 16 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de ovejas** se distribuyen en un plano cartesiano.

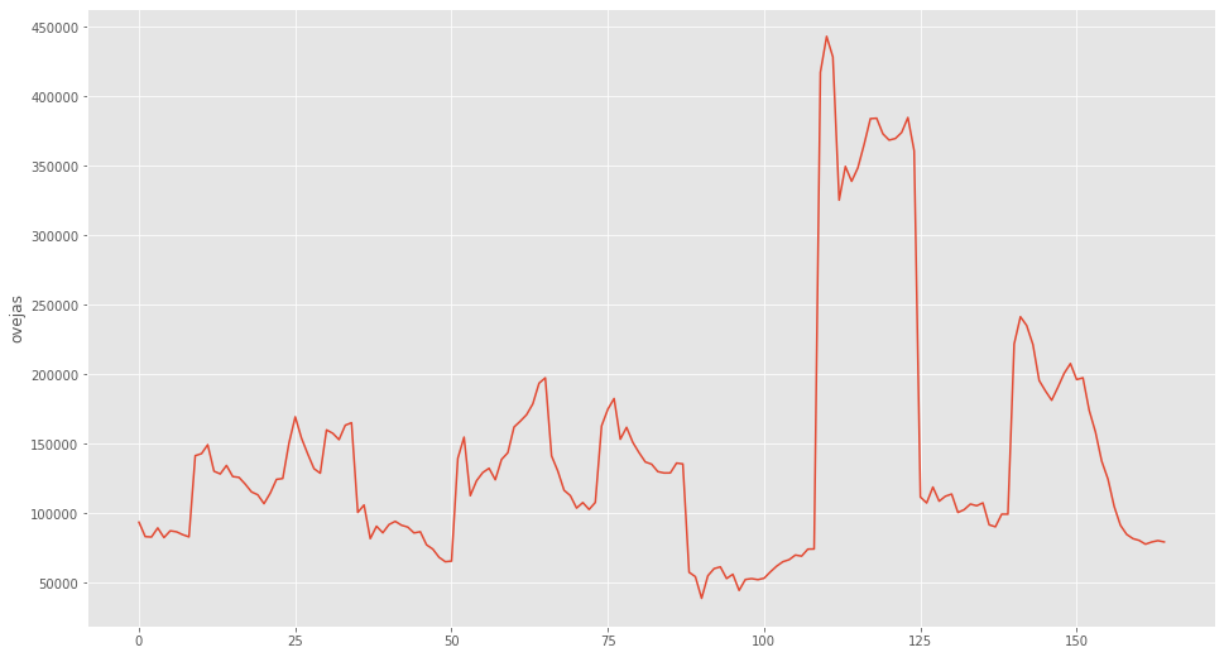
In [30]: `sns.scatterplot(data=datos['ovejas'])`

Out[30]: `<AxesSubplot:ylabel='ovejas'>`



```
In [31]: sns.lineplot(data=datos['ovejas'])
```

```
Out[31]: <AxesSubplot:ylabel='ovejas'>
```



Al analizar los datos representados en el gráfico de dispersión, no se puede identificar un patrón o tendencia clara entre las variables, aunque se pueden identificar alrededor de 3 o 4 grupos grandes en los datos.

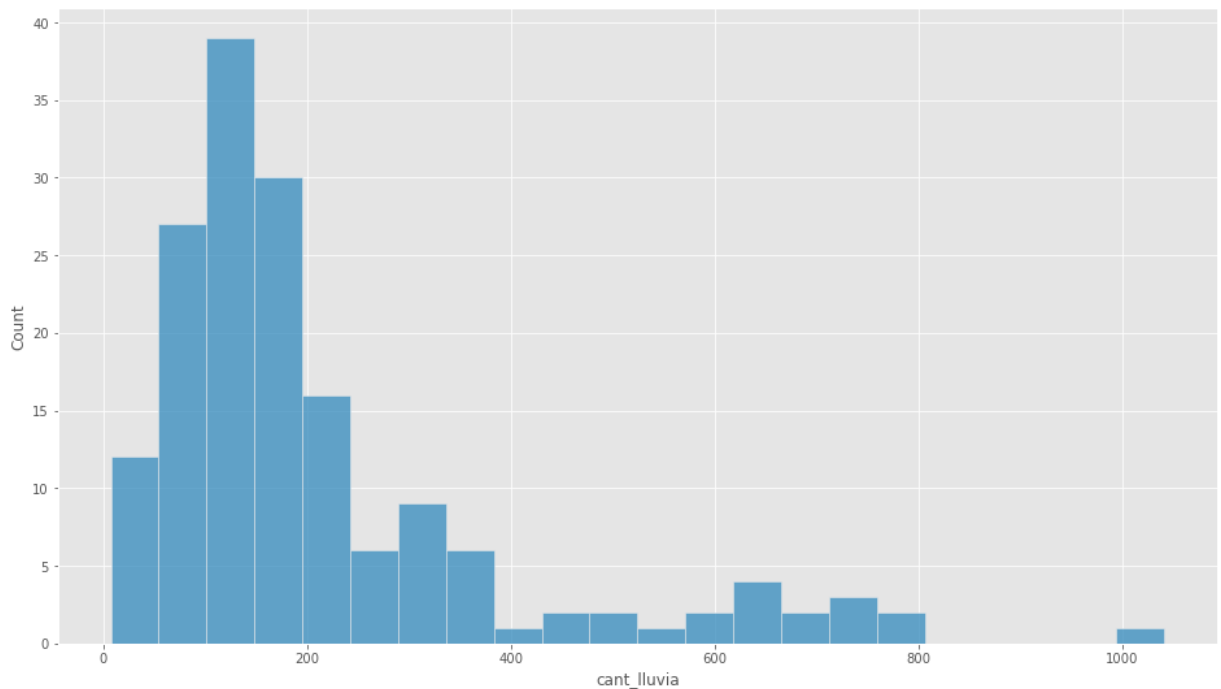
Por otro lado, al observar el gráfico de línea del total de ovejas, se puede notar que la mayoría de las observaciones se encuentran entre 50000 y 200000 ovejas. Sin embargo, no se aprecia una tendencia clara, por lo que no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Lluvia

En el siguiente histograma vemos que la `cant_lluvia` tiene una distribución asimétrica.

```
In [32]: sns.histplot(data=datos['cant_lluvia'])
```

Out[32]: <AxesSubplot:xlabel='cant_lluvia', ylabel='Count'>



Se ve una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho lo que indica una distribución asimétrica derecha. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el **total de mm lluvia** que nos indicaba el histograma.

```
In [33]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['cant_lluvia'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 1100, 100)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Mm de lluvia')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
```

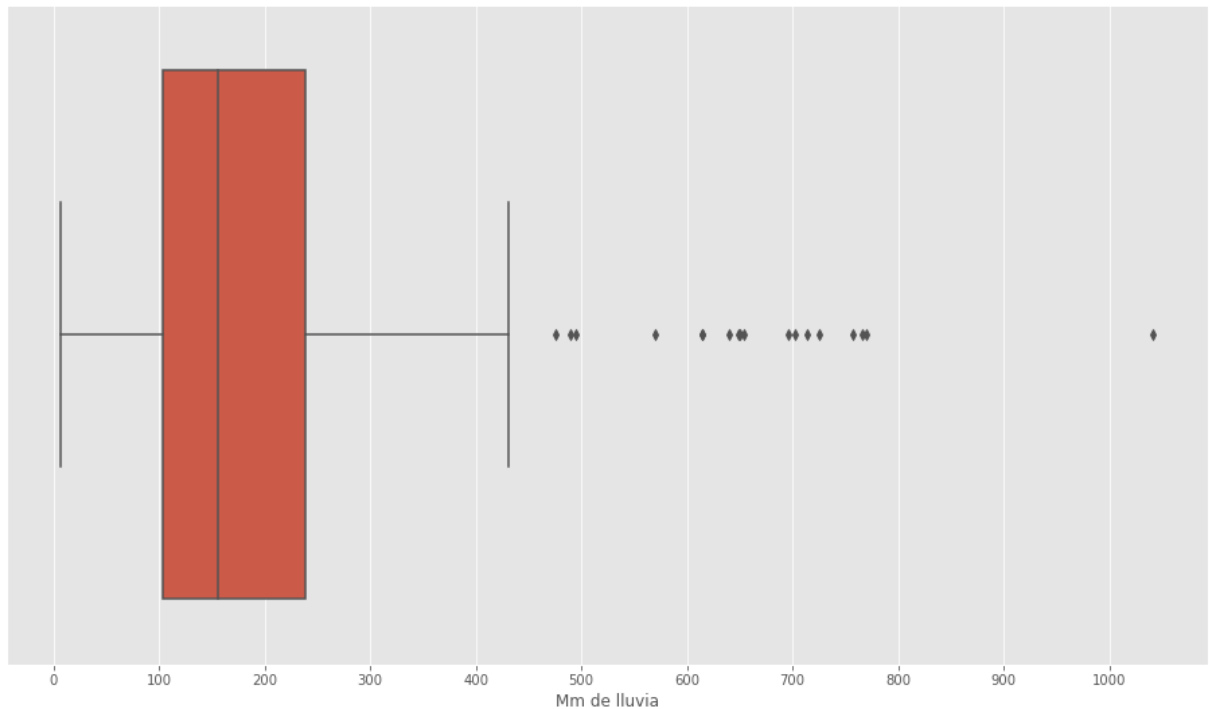


```
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 7.1

Valor del bigote superior: 431.23



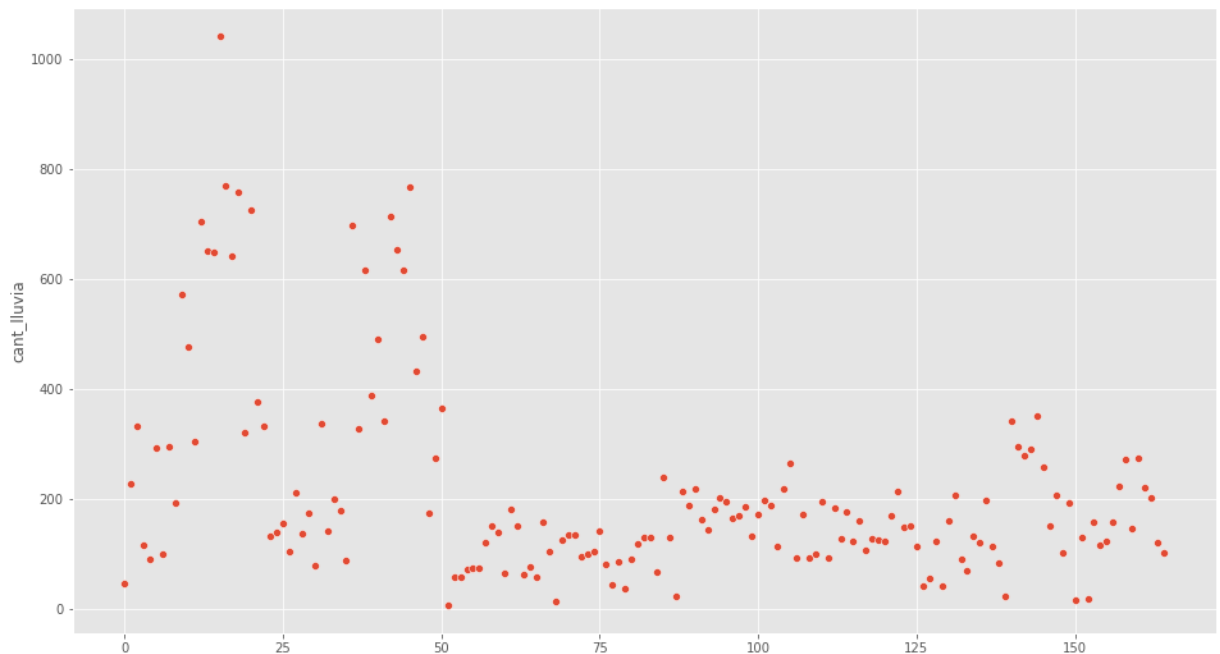
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 103.7 mm de lluvia.
- Segundo Cuartil: el 50% de los valores son menores o igual a 155.4 mm de lluvia.
- Tercer cuartil: el 75% de los valores son menores o igual a 238.6 mm de lluvia.
- Mediana: la mediana es 155.4 mm de lluvia y corta la caja en dos lados desiguales, por lo tanto muestra una asimetría positiva o segada a la derecha dado que la parte más larga de la caja es la parte superior a la mediana.
- Bigotes: el limite inferior es de 7.1 mm y el limite superior es de 431.23 mm.
- Valores atípicos: hay 16 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el total de lluvia se distribuyen en un plano cartesiano.

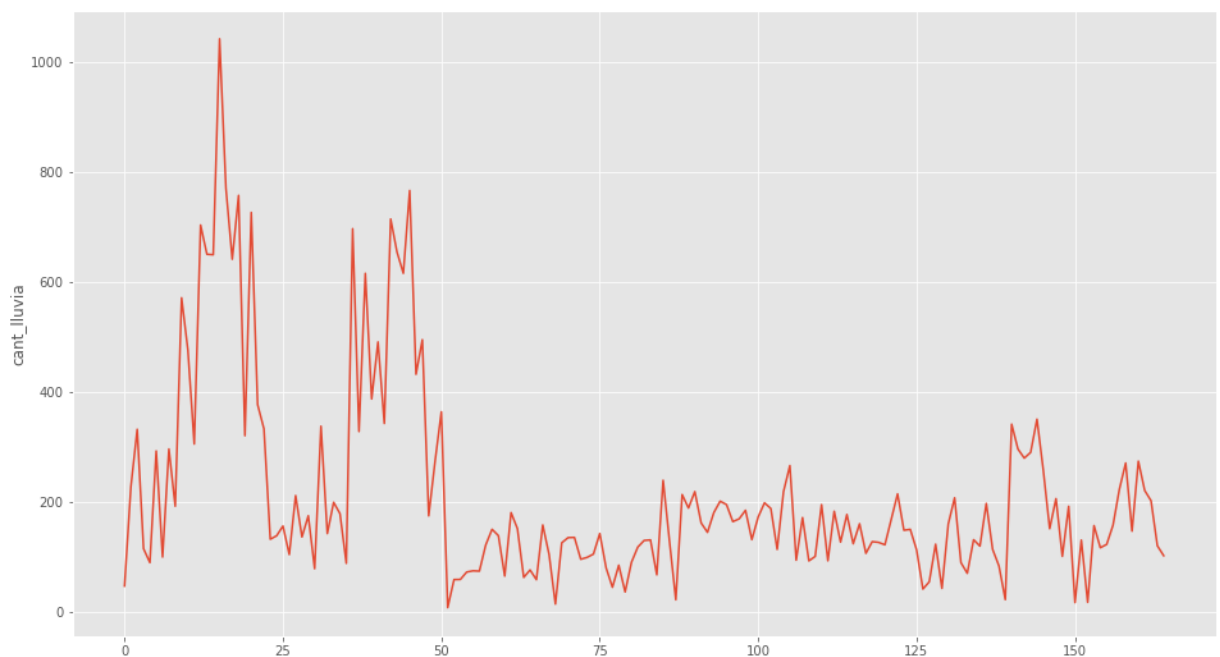
```
In [34]: sns.scatterplot(data=datos['cant_lluvia'])
```

```
Out[34]: <AxesSubplot:ylabel='cant_lluvia'>
```



```
In [35]: sns.lineplot(data=datos['cant_lluvia'])
```

```
Out[35]: <AxesSubplot:ylabel='cant_lluvia'>
```



Tras analizar los gráficos de dispersión y de línea, no se puede identificar un patrón o tendencia clara en los datos de lluvia en Chubut. En el gráfico de dispersión, no se observa ningún grupo o agrupación definida de los puntos. Por su parte, en el gráfico de línea se aprecian varios picos altos, lo que indica que existe una variabilidad significativa en los valores de lluvia. Sin embargo, se puede establecer que la precipitación en Chubut varía en un rango de 0 a 600 mm, tal como se refleja en ambos gráficos.

Como observación final de las variables independientes vemos que en los gráficos de dispersión y de línea, los datos de lana y de existencias son muy parecidos, casi que tienen la misma forma.

Variables combinadas

A continuación se va a mejorar la paleta de colores para que sea más fácil distinguir entre ellos. Para lograrlo, crearemos una paleta de colores personalizada que constará de 15 colores profundos ("deep") y

la estableceremos como la paleta por defecto para los gráficos de Seaborn.

```
In [5]: mi_paleta = sns.color_palette("deep", 15)
sns.set_palette(mi_paleta)
```

A continuación, se crearán dos dataframes más pequeños divididos por los departamentos que pertenecen a la cordillera y los que no, con el fin de facilitar el proceso de gráficos conjuntos y posiblemente obtener información adicional a partir de la división de los datos.

```
In [6]: departamentos_cordillera =
datos[datos['departamento'].isin(['CUSHAMEN', 'RIO SENGUERR',
'TEHUELCHES', 'LANGUIÑEO', 'FUTALEUFU'])]
departamentos_meseta =
datos[~datos['departamento'].isin(['CUSHAMEN', 'RIO SENGUERR',
'TEHUELCHES', 'LANGUIÑEO', 'FUTALEUFU'])]
```

Graficos

Diagrama de dispersión

El comando `sns.scatterplot(data=datos, x='variable_X', y='variable_Y', hue='departamento')` va a generar un gráfico de dispersión donde se muestra la relación entre dos variables: la `variable_X` en el eje X y la `variable_Y` en el eje Y. Además, se utiliza el argumento `hue` para agregar una tercera dimensión de información, que en este caso es el departamento. Cada punto en el gráfico representa una observación del dataset, donde la posición en el eje X corresponde al valor de `'variable_X'` de la observación, la posición en el eje Y corresponde al valor de `'variable_Y'` en esa observación y el color corresponde al departamento.

Diagrama de línea

El código `sns.lineplot(data=datos, x='variable_X', y='variable_Y')` genera un gráfico de línea que representa la evolución de la `'variable_X'` a lo largo de `'variable_Y'`. Cada punto en el gráfico representa la `'variable_X'` registrada para un valor específico de `'variable_Y'`, la línea que los une representa la tendencia general de la variable a lo largo de `'variable_Y'`, es decir, la evolución de `'variable_X'` a medida que cambia `'variable_Y'`, y el borde de color más claro indica el intervalo de confianza. Esto significa que el gráfico muestra un valor promedio de la variable `'variable_X'` para cada valor de `'variable_Y'`, junto con un intervalo de confianza alrededor de ese valor promedio. El intervalo de confianza indica la incertidumbre en la estimación del valor promedio y se calcula comúnmente como la desviación estándar de los datos dividido por la raíz cuadrada del número de observaciones.

Como el intervalo de confianza se calcula en función del número de observaciones que hay en los datos, en cada uno de los gráficos que vamos a ver a continuación vemos que comienzan en el año 2000 y el intervalo de confianza se muestra a partir del 2006, esto es porque hay pocas observaciones en los años anteriores a 2006, lo que hace que el intervalo de confianza no sea significativo o no pueda ser calculado con precisión.

Diagrama de barras

El comando `sns.barplot(data=datos, x='fecha', y='variable_Y')` genera un gráfico de barras que muestra la cantidad total de 'variable_Y' para cada año en el conjunto de datos. Cada año se mostraría en el eje x y la cantidad total 'variable_Y' correspondiente se mostraría en el eje y. La altura de cada barra representaría la cantidad la 'variable_Y' para una fecha determinada. La línea finita que se encuentra en la parte superior de cada barra representa el intervalo de confianza del 95% para la media de cada grupo de datos. Es decir, es una estimación de la incertidumbre en la medida de la media para cada grupo. La altura de la línea muestra el rango de valores dentro del cual se espera que la media real esté con una confianza del 95%.

Fecha con otras variables

Existencias ovinas

Con el fin de manejar adecuadamente la información de las fechas, se procedió a ordenar el conjunto de datos según este atributo y se guardó en un nuevo DataFrame. Posteriormente, se agregó una nueva columna que contiene únicamente el año, lo que permitirá trabajar de manera más eficiente con esta información en caso de ser necesario.

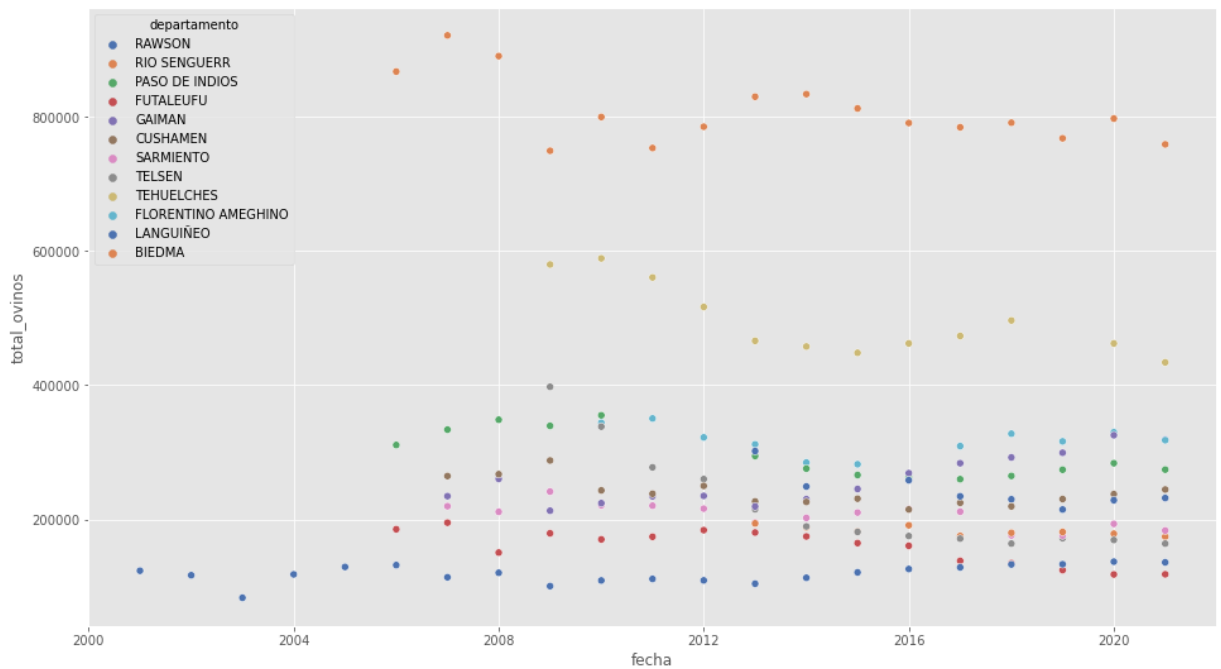
```
In [37]: datos_ordenados_fecha = datos.sort_values('fecha')
datos_ordenados_fecha['anio'] =
datos_ordenados_fecha['fecha'].dt.strftime('%Y')
```

Total de ovinos

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo `total_ovinos` para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de ovinos a lo largo del tiempo y cómo se distribuyen en los diferentes años.

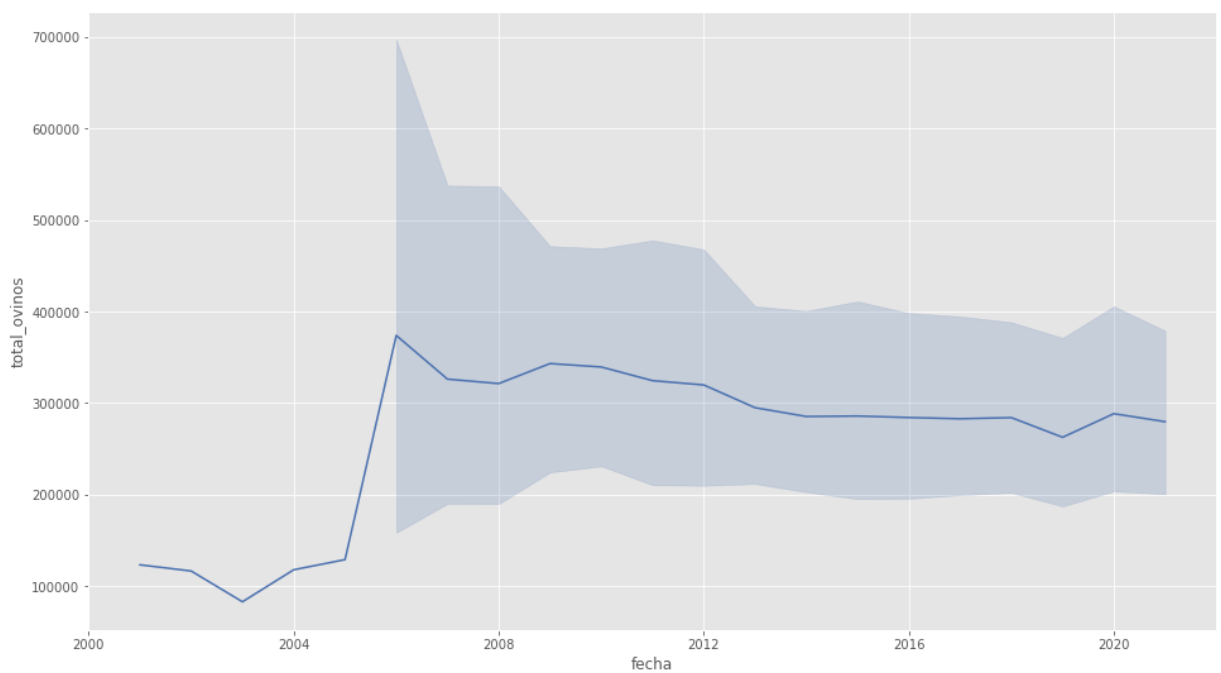
```
In [38]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',
y='total_ovinos', hue='departamento')
```

```
Out[38]: <AxesSubplot:xlabel='fecha', ylabel='total_ovinos'>
```



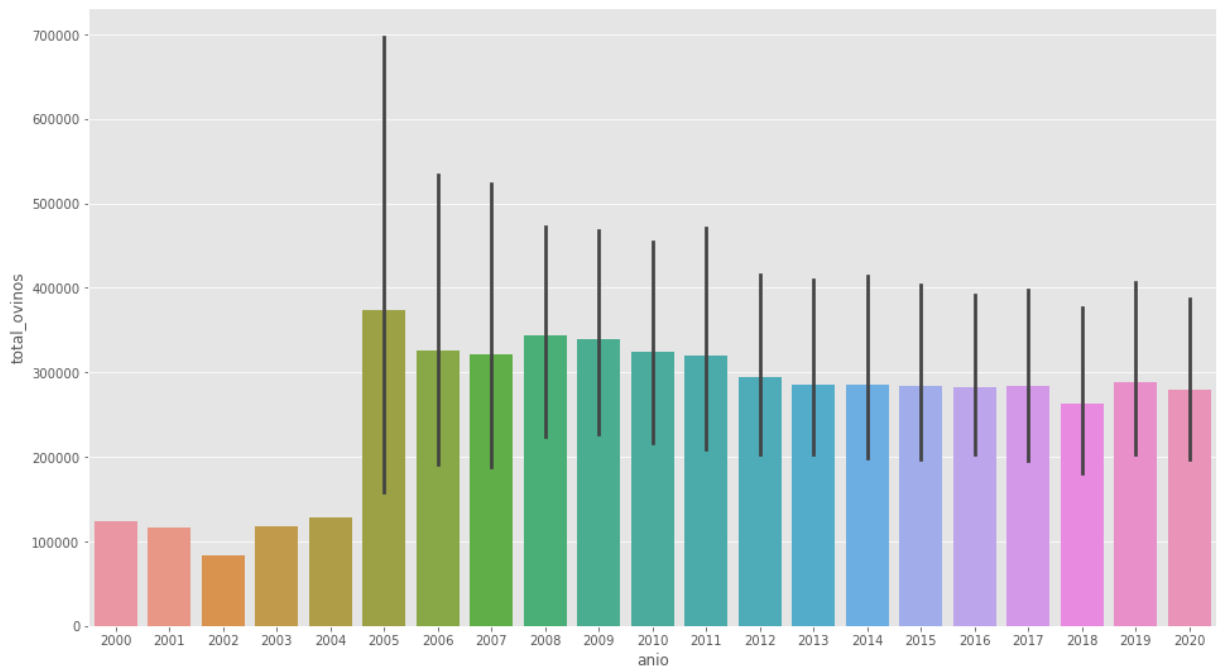
```
In [39]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',
y='total_ovinos')
```

```
Out[39]: <AxesSubplot:xlabel='fecha', ylabel='total_ovinos'>
```



```
In [40]: sns.barplot(data=datos_ordenados_fecha, x='anio',
y='total_ovinos')
```

```
Out[40]: <AxesSubplot:xlabel='anio', ylabel='total_ovinos'>
```



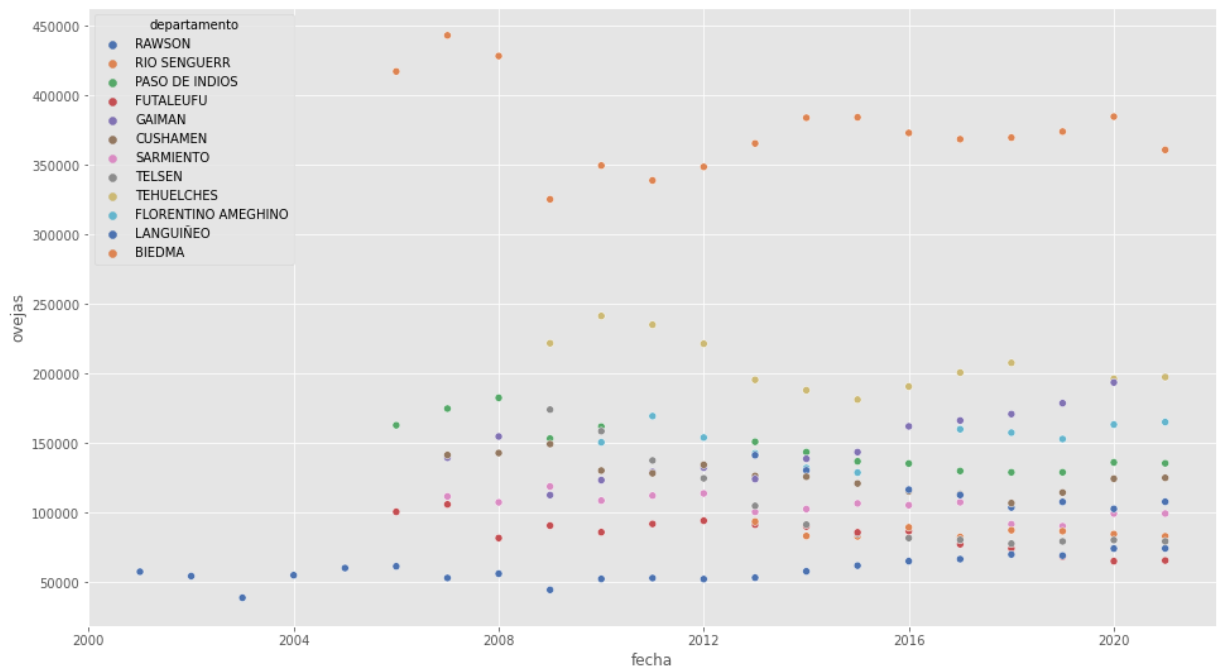
El análisis de estos tres gráficos proporciona información sobre la variación del número total de ovinos a lo largo del tiempo. En el gráfico de dispersión, cada punto representa un departamento en un plano cartesiano, donde su posición en los ejes X e Y representa la cantidad de ovinos en un año específico. Los puntos se colorean según el departamento, lo que facilita la comparación visual entre departamentos. Se observa que la mayoría de los departamentos mantienen una cantidad de ovinos relativamente constante, que no supera los 400 mil, excepto Tehuelches y Río Senger, que presentan valores más altos y variables. Cuando el número de ovinos supera los 400 mil, los puntos se dispersan mucho, lo que indica una gran variabilidad en la producción de ovinos a lo largo de los años. En cambio, cuando el número de ovinos es inferior a 400 mil, las filas de puntos son más uniformes, lo que sugiere una producción más estable a lo largo del tiempo. Este patrón también se refleja en el gráfico de línea y en el de barras, que incluyen intervalos de confianza para mostrar la variabilidad de los datos.

Ovejas

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo `ovejas` para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de ovejas a lo largo del tiempo y cómo se distribuyen en los diferentes años.

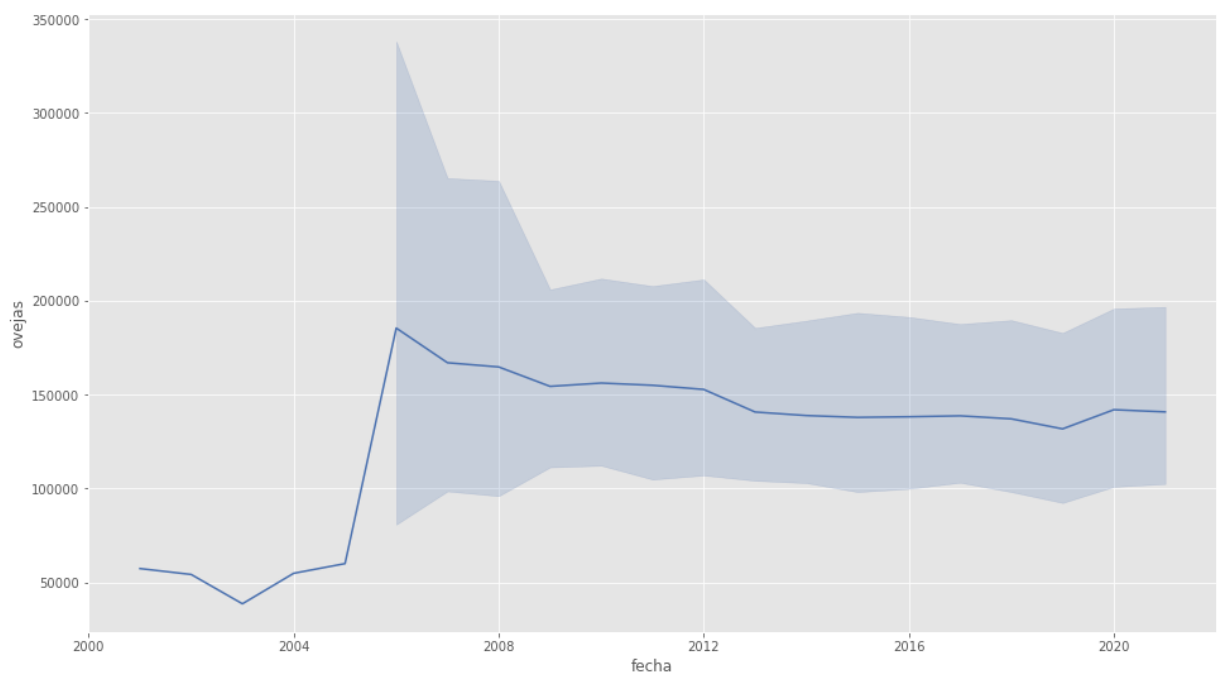
```
In [41]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='ovejas',
hue='departamento')
```

```
Out[41]: <AxesSubplot:xlabel='fecha', ylabel='ovejas'>
```



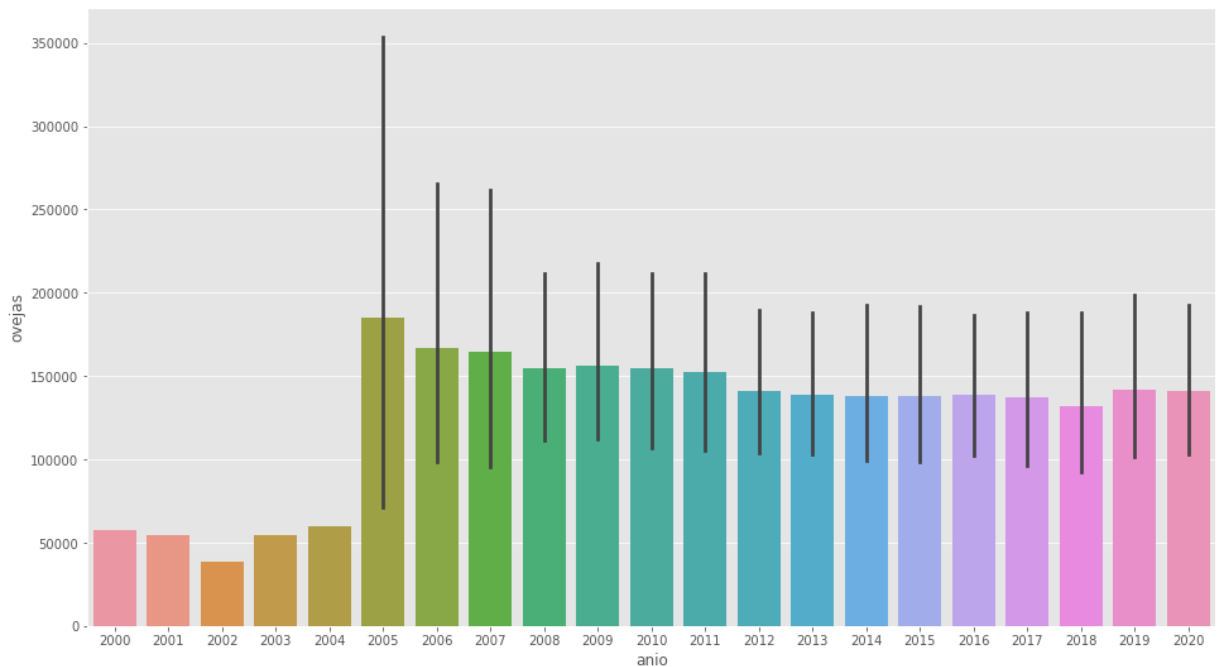
In [42]: `sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='ovejas')`

Out[42]: `<AxesSubplot:xlabel='fecha', ylabel='ovejas'>`



In [43]: `sns.barplot(data=datos_ordenados_fecha, x='anio', y='ovejas')`

Out[43]: `<AxesSubplot:xlabel='anio', ylabel='ovejas'>`



Los tres gráficos proporcionan datos acerca de la evolución del total de ovejas en diferentes departamentos a lo largo del tiempo. El gráfico de dispersión muestra cada departamento como un punto en un plano cartesiano, donde la posición de los puntos en los ejes X e Y representa la cantidad de ovejas en un año específico. El color de cada punto indica el departamento al que pertenece, lo que nos permite comparar visualmente la cantidad de ovejas entre departamentos.

Al analizar el gráfico de dispersión, se puede apreciar que el departamento de Gaiman experimentó una disminución en la cantidad de ovejas en el año 2009, pero a partir de 2010 logró aumentar progresivamente la cantidad de ovejas hasta superar los 190 mil en el año 2020. Además, se puede observar que los departamentos de Tehuelches y Río Senger presentan valores que superan los 190 mil, y muestran una mayor variabilidad en la producción de ovejas en comparación con el resto de los departamentos que están por debajo de este valor.

Cuando la cantidad de ovejas supera los 190 mil, se observa una dispersión de los puntos en el gráfico de dispersión, lo que sugiere una mayor variabilidad en la producción de ovejas a lo largo de los años. Por otro lado, cuando la cantidad de ovejas es inferior a 190 mil, se observa una mayor uniformidad en las filas de puntos, lo que indica una producción más estable a lo largo del tiempo. Este patrón se puede confirmar en los gráficos de línea y de barras, los cuales incluyen intervalos de confianza para mostrar la variabilidad de los datos.

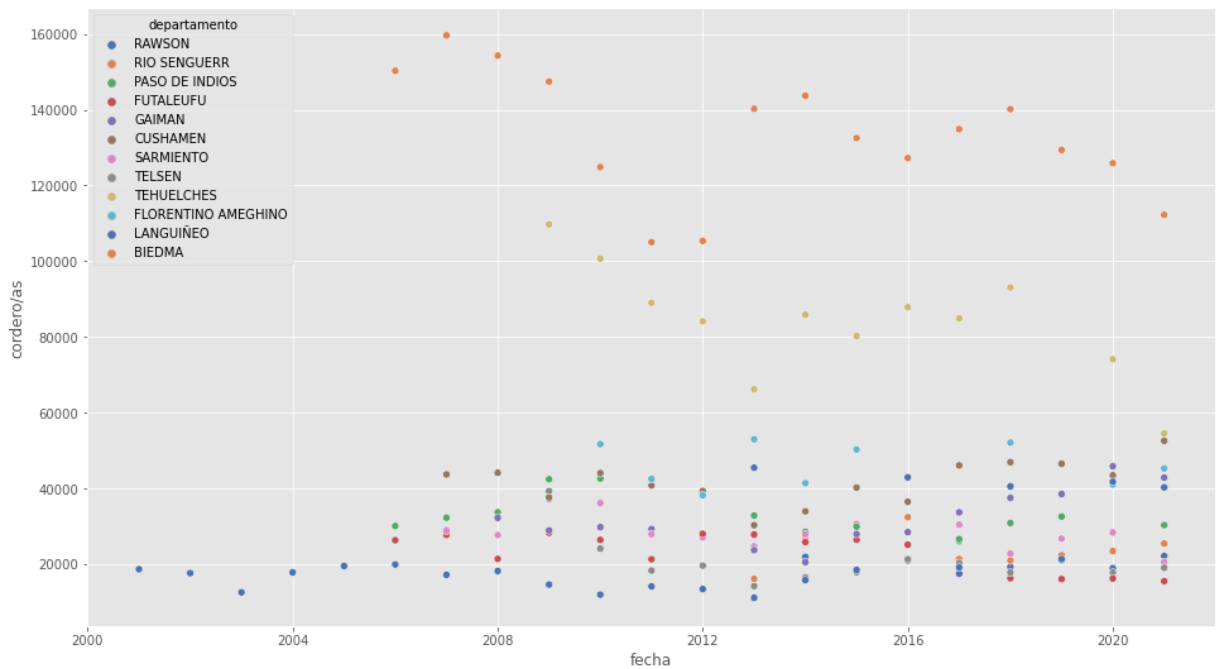
Corderos

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **corderos** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de corderos a lo largo del tiempo y cómo se distribuyen en los diferentes años.

In [44]:

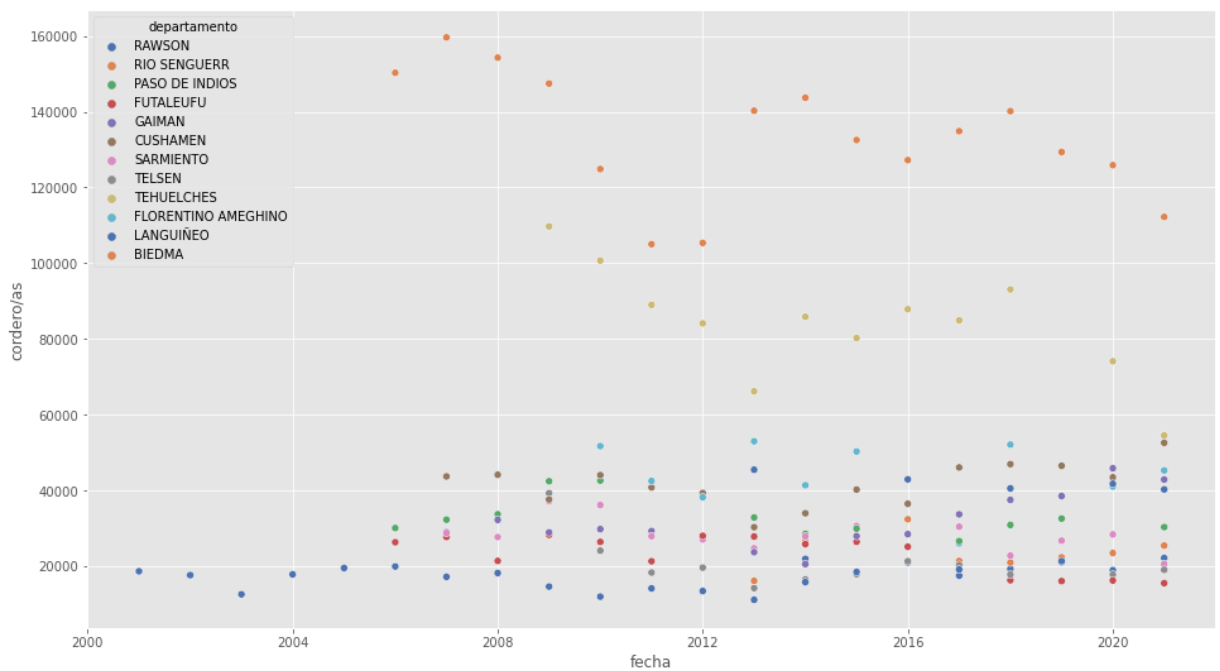
```
# Crear el scatterplot con la nueva paleta de colores
sns.scatterplot(data=datos_ordenados_fecha, x='fecha',
y='cordero/as', hue='departamento')
```

Out[44]: <AxesSubplot:xlabel='fecha', ylabel='cordero/as'>



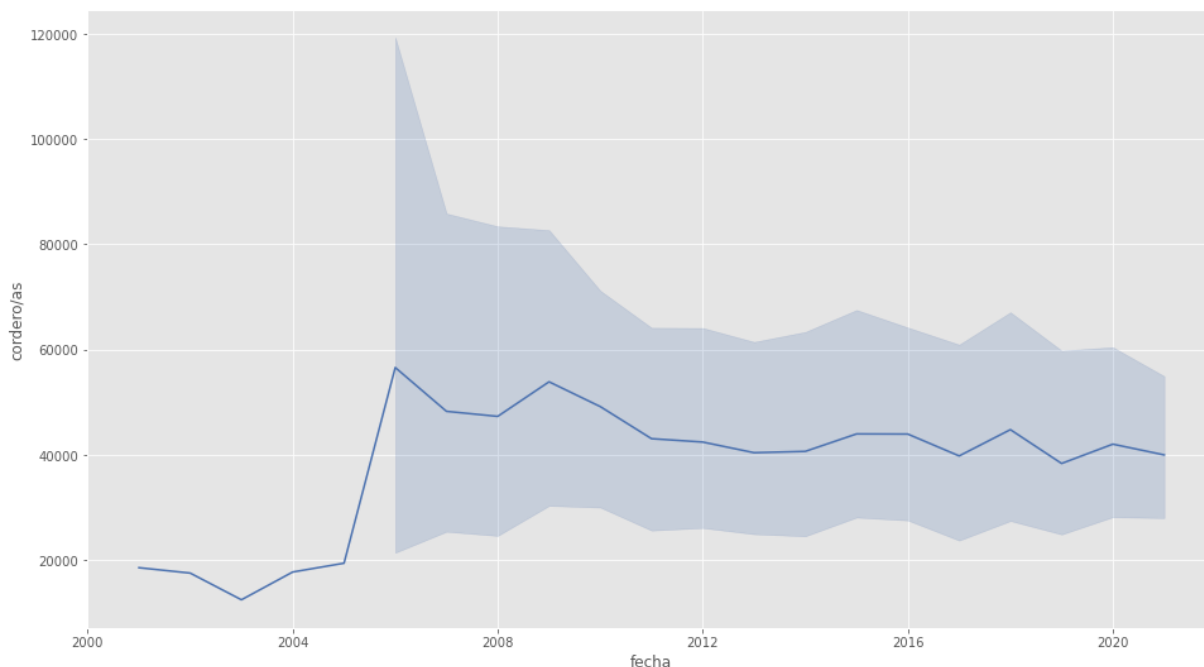
```
In [45]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',
                        y='cordero/as', hue='departamento')
```

```
Out[45]: <AxesSubplot:xlabel='fecha', ylabel='cordero/as'>
```



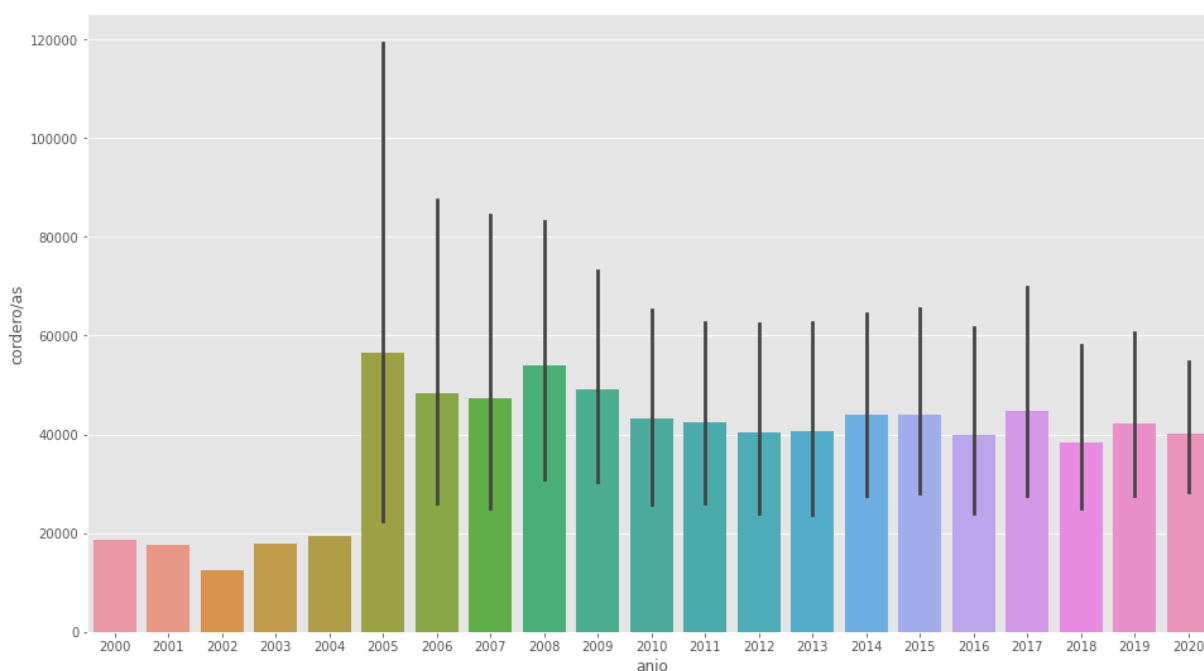
```
In [46]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',
                      y='cordero/as')
```

```
Out[46]: <AxesSubplot:xlabel='fecha', ylabel='cordero/as'>
```



```
In [47]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='cordero/as')
```

```
Out[47]: <AxesSubplot:xlabel='anio', ylabel='cordero/as'>
```



Se pueden obtener datos acerca de la variación en la producción de corderos a lo largo del tiempo mediante estos tres gráficos. El gráfico de dispersión muestra cada departamento representado por un punto en un plano cartesiano, donde la posición en los ejes X e Y indica la cantidad de corderos en un año determinado. Los puntos se codifican por colores según el departamento al que pertenecen, lo que permite comparar la cantidad de corderos entre departamentos. Se puede observar que los departamentos de Tehuelches y Río Senger presentan valores que superan los 60 mil, y los datos varían ampliamente. Además, los departamentos de Florentino Ameghino y Cushamen presentan un rango de corderos entre 44 mil y 60 mil, con una variabilidad moderada en comparación con otros departamentos que tienen valores inferiores. Cuando el total de corderos supera los 60 mil, los puntos están muy dispersos, lo que sugiere una mayor variabilidad en la producción a lo largo de los años. En cambio, cuando los totales de corderos son inferiores a 44 mil, las filas de puntos son más uniformes, lo que indica una producción más estable

en el transcurso de los años. Este patrón se refleja también en los gráficos de línea y barra, los cuales incluyen intervalos de confianza para mostrar la variabilidad de los datos.

Producción de lana

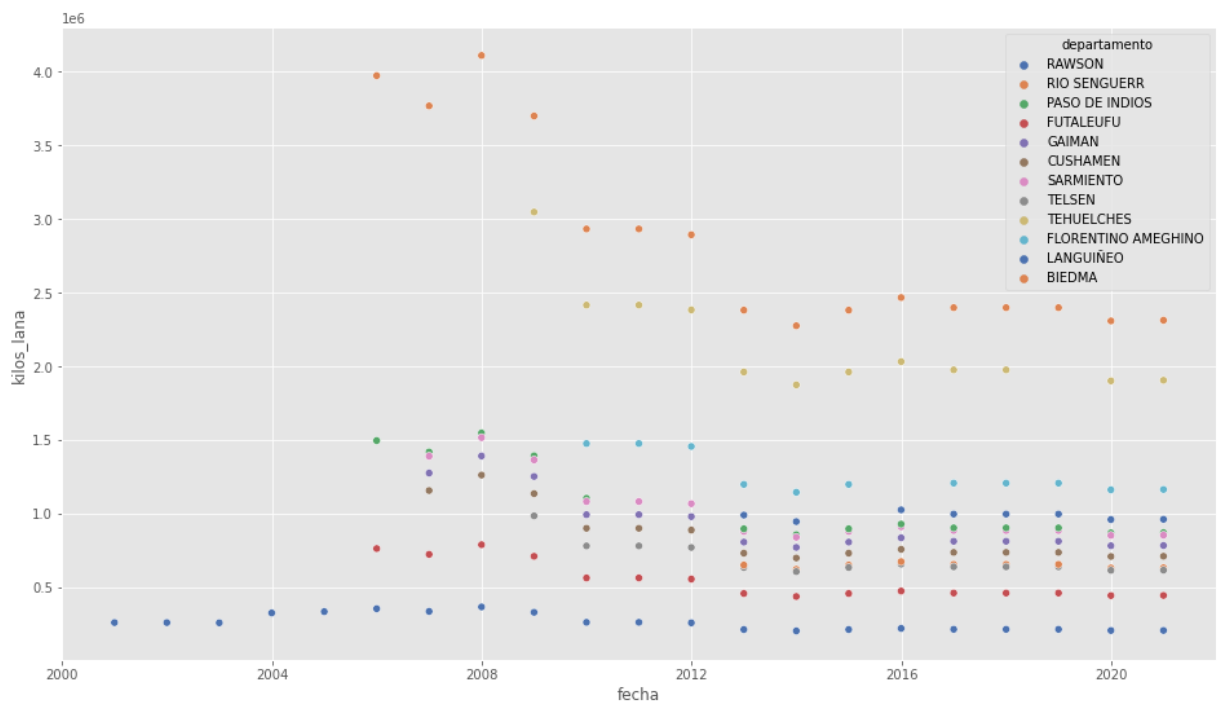
El objetivo primordial en la cría de ovejas para obtener lana es producir lanas de calidad cruzada fina y mediana, con un diámetro de entre 25 y 32 micras y un rendimiento de lavado superior al 75%. En cuanto a la situación en la provincia de Chubut, se puede observar que la producción de lana es en su mayoría de buena calidad, con un diámetro promedio en el rango deseado y un rendimiento de lavado adecuado, según lo indican los gráficos presentados a continuación.

Kilos de lana

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo `kilos de lana` para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

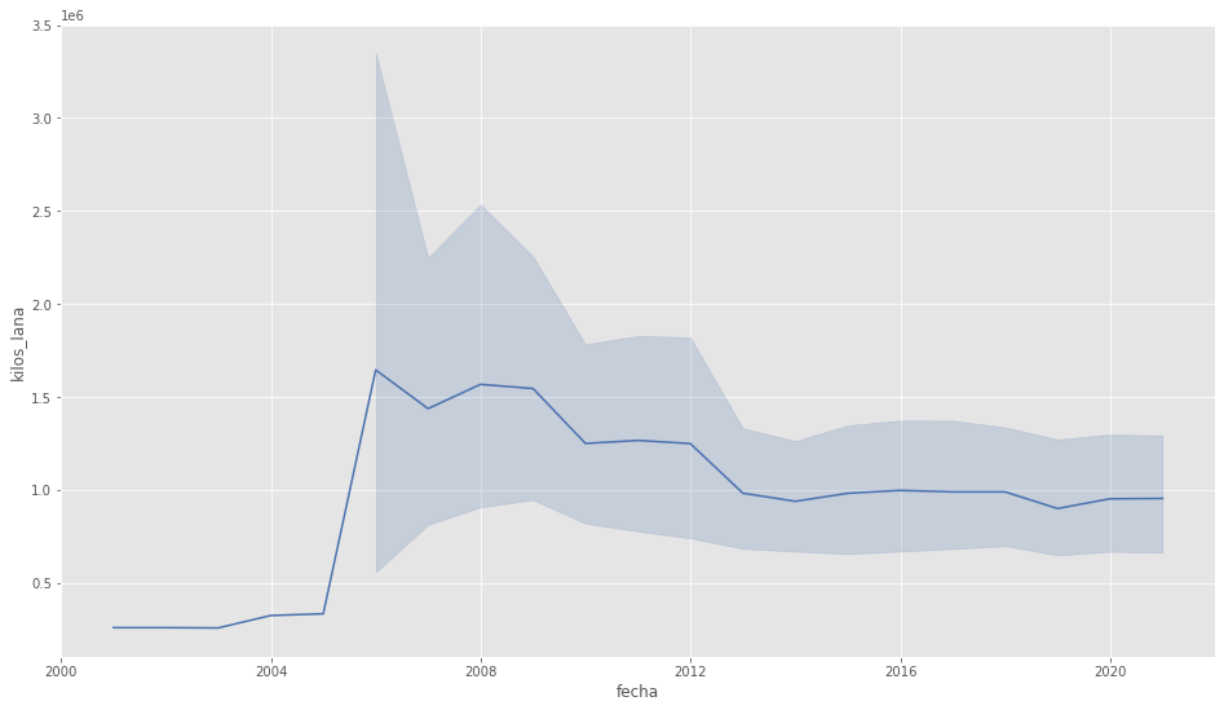
```
In [48]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',  
                        y='kilos_lana', hue='departamento')
```

```
Out[48]: <AxesSubplot:xlabel='fecha', ylabel='kilos_lana'>
```



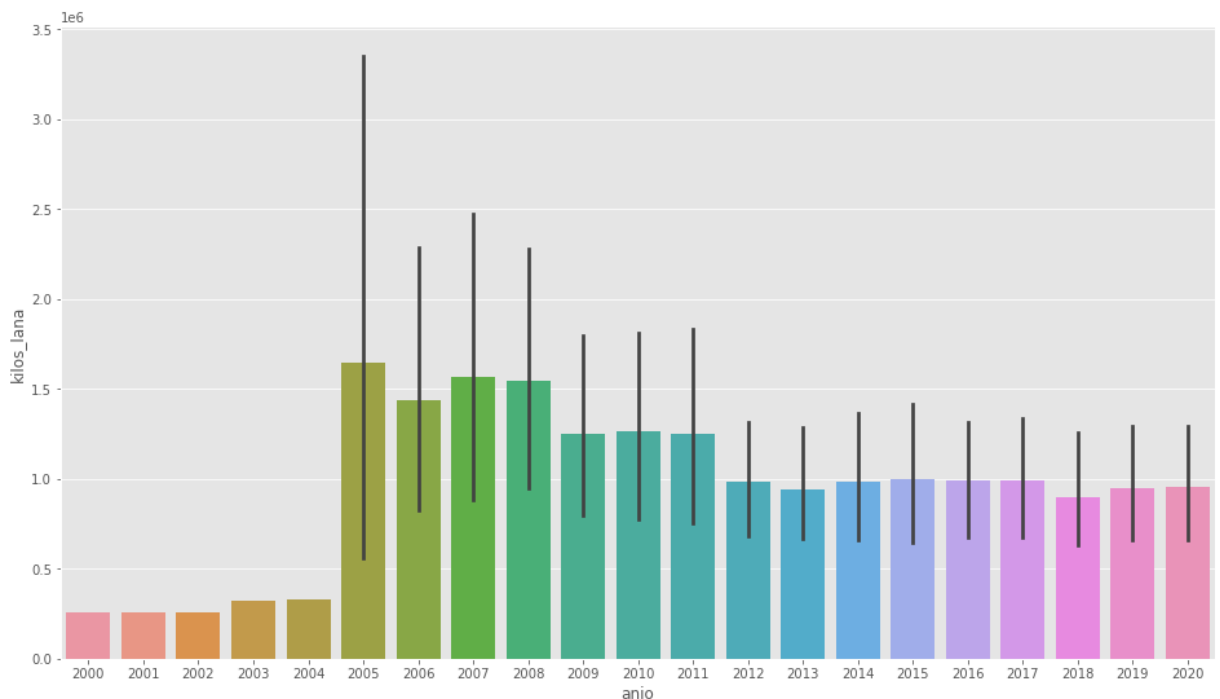
```
In [49]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',  
                     y='kilos_lana')
```

```
Out[49]: <AxesSubplot:xlabel='fecha', ylabel='kilos_lana'>
```



```
In [50]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='kilos_lana')
```

```
Out[50]: <AxesSubplot:xlabel='anio', ylabel='kilos_lana'>
```



Para obtener información sobre la variación en la producción de lana a lo largo del tiempo, se pueden examinar estos tres gráficos. El scatterplot muestra cada departamento representado por un punto en un plano cartesiano, con su ubicación en los ejes X e Y indicando la cantidad de lana en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que permite comparar visualmente la cantidad de lana entre departamentos. Se puede observar que los departamentos de Tehuelches y Río Senger presentan valores que superan la tonelada y media de lana, y son más variables en comparación con otros departamentos que presentan valores inferiores.

Entonces cuando el total de kilos de lana supera los 1500000, los puntos están muy dispersos, lo que sugiere una mayor variabilidad en la producción a lo largo de los años. En cambio, cuando la cantidad de lana es inferior a 1.5 toneladas, las filas de puntos son más uniformes, lo que indica una producción más

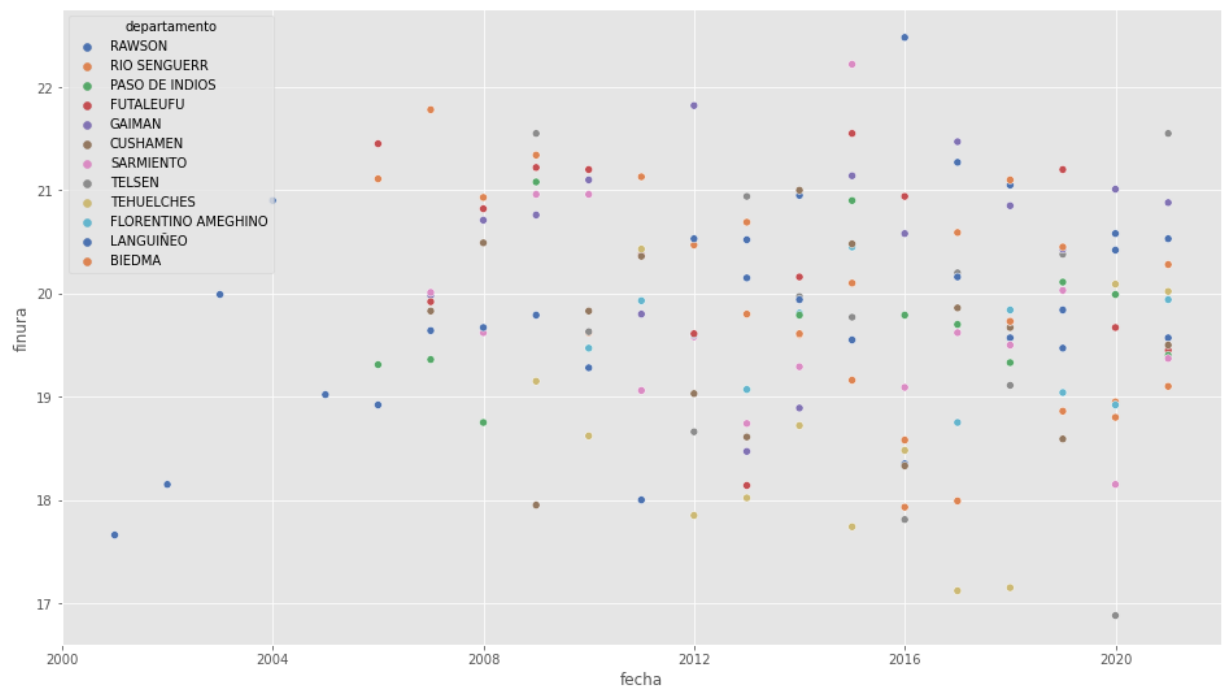
estable en el transcurso de los años. Este patrón también se refleja en los gráficos de línea y barra, los cuales incluyen intervalos de confianza para mostrar la variabilidad de los datos.

Finura

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **finura de la lana** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la finura de la lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

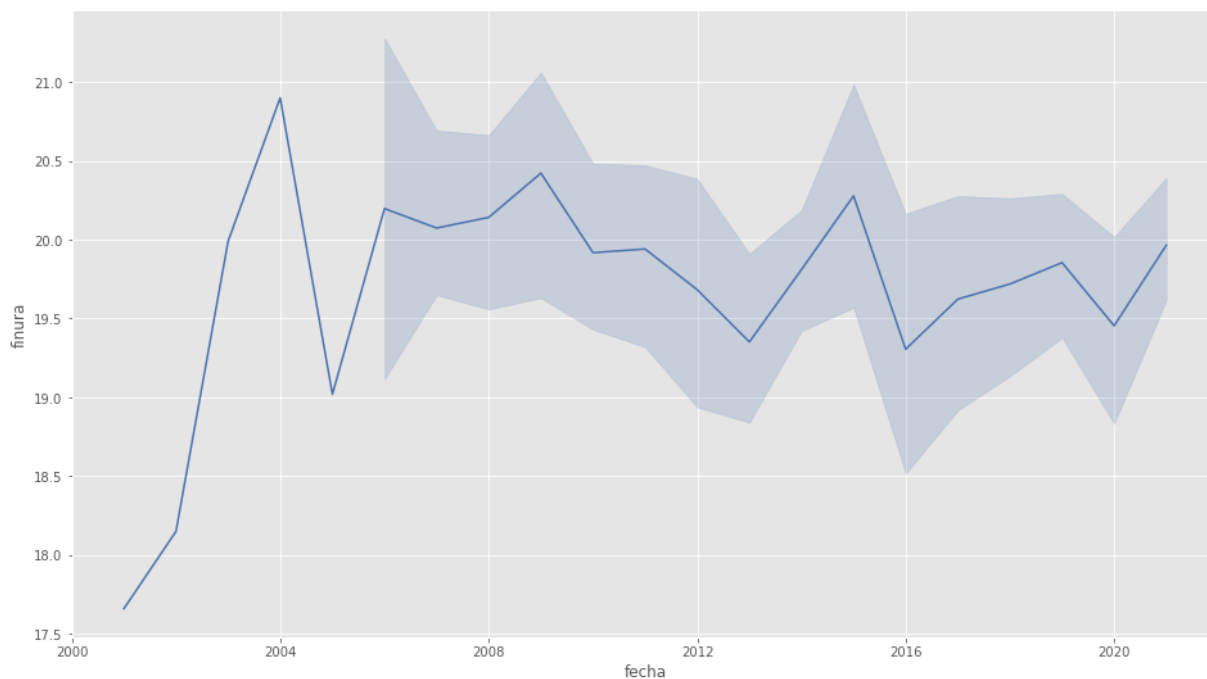
```
In [51]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='finura',  
hue='departamento')
```

```
Out[51]: <AxesSubplot:xlabel='fecha', ylabel='finura'>
```



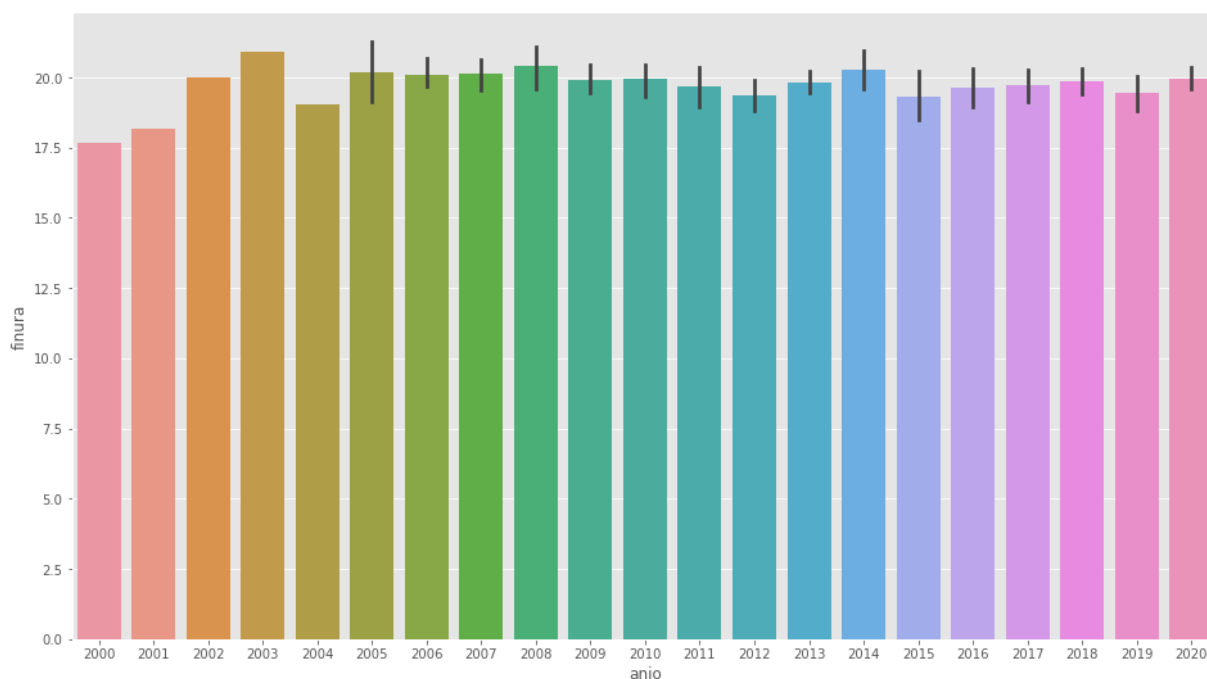
```
In [52]: sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='finura')
```

```
Out[52]: <AxesSubplot:xlabel='fecha', ylabel='finura'>
```



```
In [53]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='finura')
```

```
Out[53]: <AxesSubplot:xlabel='anio', ylabel='finura'>
```



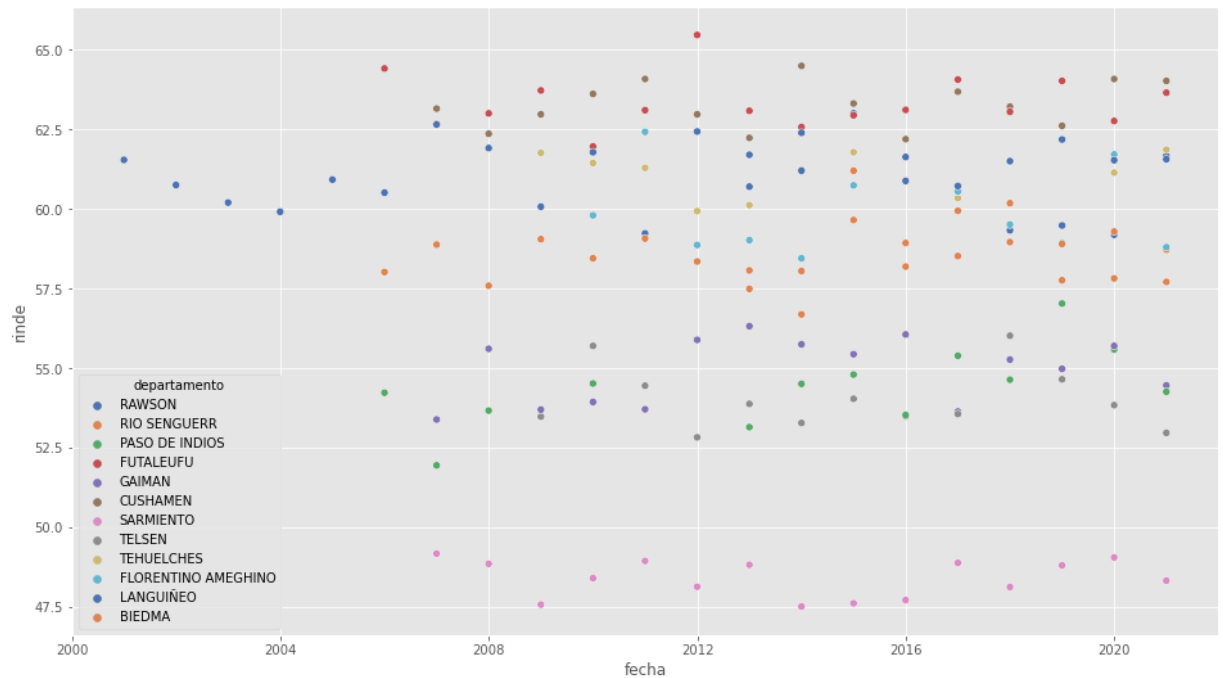
Para obtener información acerca de la variación en la finura de la lana a lo largo del tiempo, se pueden examinar estos tres gráficos. El scatterplot representa cada departamento mediante un punto en un plano cartesiano, y la posición del punto en los ejes X e Y indica la finura en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que permite una fácil comparación visual de la finura de la lana entre departamentos. Sin embargo, los datos están muy dispersos y no se puede observar ningún patrón general. A pesar de esto, es posible notar que en la provincia de Chubut, la finura de la lana se mantiene constante entre 18.5 y 21.0 en los gráficos de línea y de barras. Esto sugiere que Chubut mantiene un buen control sobre la finura de la lana, lo que es una señal positiva para la industria lanera en la región.

Rinde

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **rinde de la lana** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en el rinde de la lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

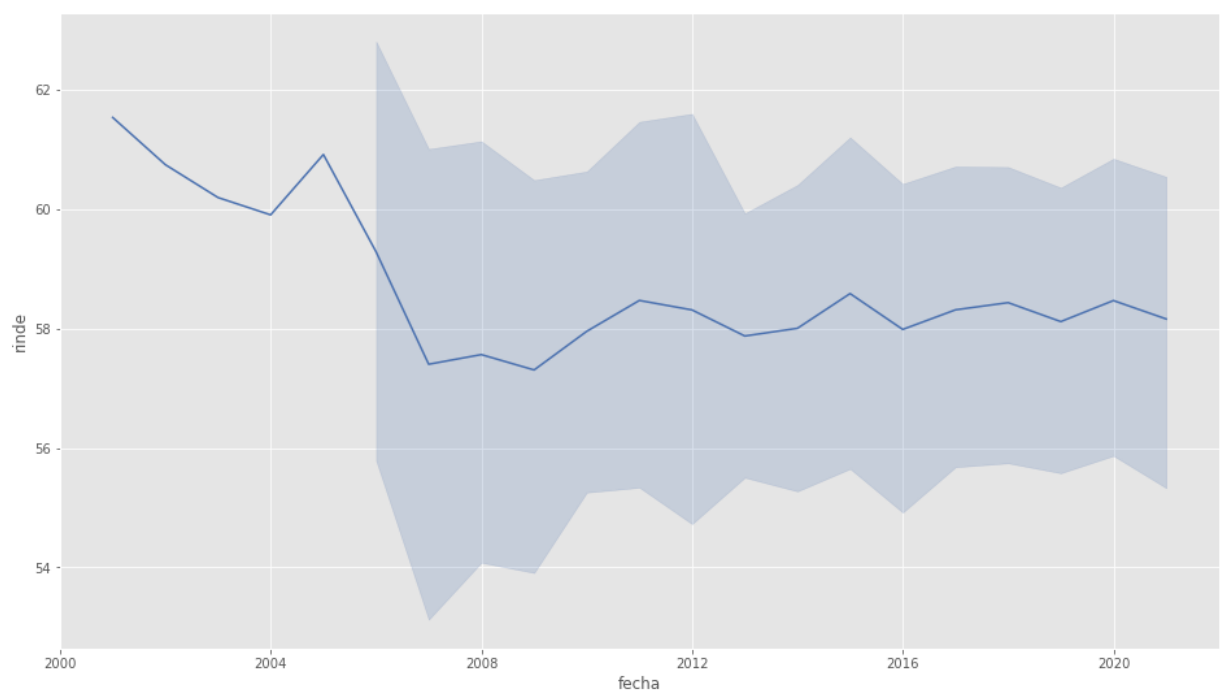
```
In [54]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='rinde',  
hue='departamento')
```

```
Out[54]: <AxesSubplot:xlabel='fecha', ylabel='rinde'>
```



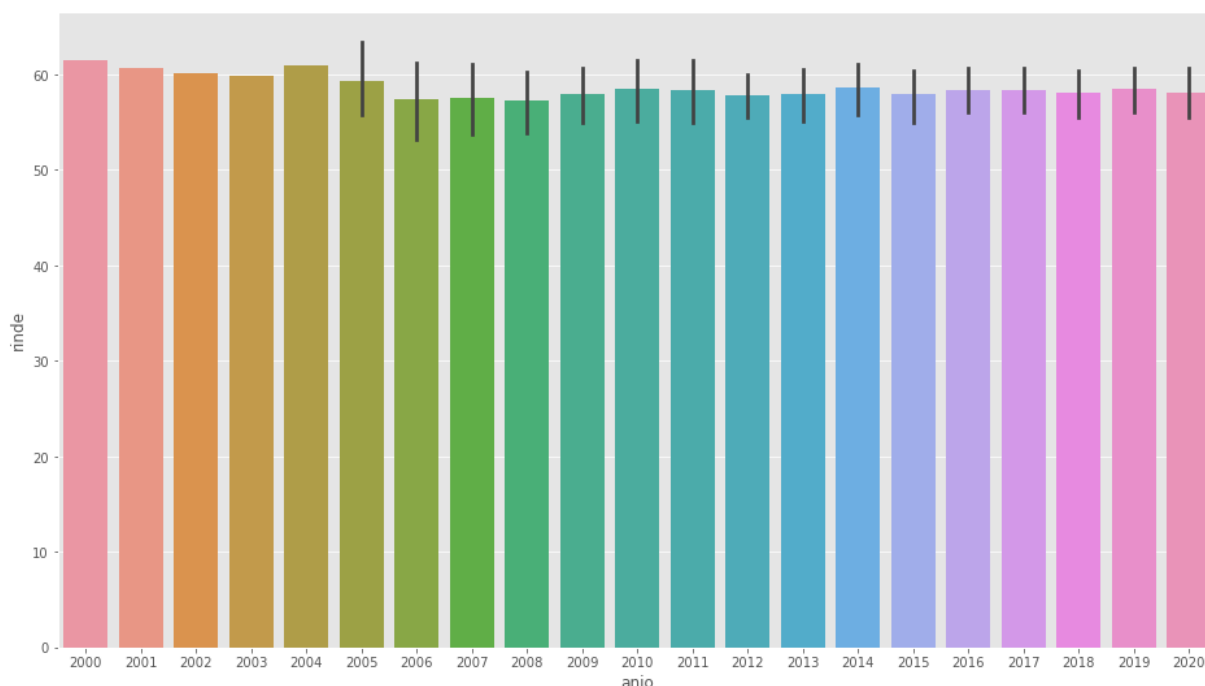
```
In [55]: sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='rinde')
```

```
Out[55]: <AxesSubplot:xlabel='fecha', ylabel='rinde'>
```



```
In [56]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='rinde')
```

```
Out[56]: <AxesSubplot:xlabel='anio', ylabel='rinde'>
```



Para obtener información sobre la variación en el rendimiento de lavado de la lana a lo largo del tiempo, se pueden analizar estos tres gráficos. El scatterplot representa cada departamento mediante un punto en un plano cartesiano, con la posición del punto en los ejes X e Y indicando el rendimiento en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que permite una fácil comparación visual del rendimiento de lavado de la lana entre departamentos.

Se puede observar que los departamentos de Cushamen y Futaleufú tienen los valores más altos de rendimiento de la lana, superiores a 62.5 para la mayoría de los años del dataset, lo que indica que la lana producida en esos departamentos tiene un alto valor en el mercado debido a su mayor rendimiento. Por otro lado, el departamento de Sarmiento presenta los valores más bajos de rendimiento de la lana, lo que indica que la lana producida en este departamento tiene un menor valor debido a su menor rendimiento.

Sin embargo, en los gráficos de línea y barra se puede observar que el rendimiento de la lana en Chubut se mantiene bastante estable entre 54 y 62, mientras que el ideal es 75. Esto sugiere que la provincia de Chubut tiene un buen rendimiento de lavado de la lana, lo que es una buena señal para la industria de la lana en la región. Un alto rendimiento de lavado significa que la lana puede ser lavada y procesada con mayor eficiencia, lo que a su vez se traduce en una mayor calidad del producto final y mayores beneficios económicos para los productores de lana en la región.

Lluvia con otras variables

Para entender la influencia de la lluvia en la producción de ganado ovino en la región, es importante examinar la relación que existe entre la cantidad de lluvia acumulada en un año y las variables relacionadas con las existencias de ovejas y la producción de lana. De esta manera, podremos obtener una idea más clara de cómo la cantidad de lluvia puede afectar a la producción ganadera en la región.

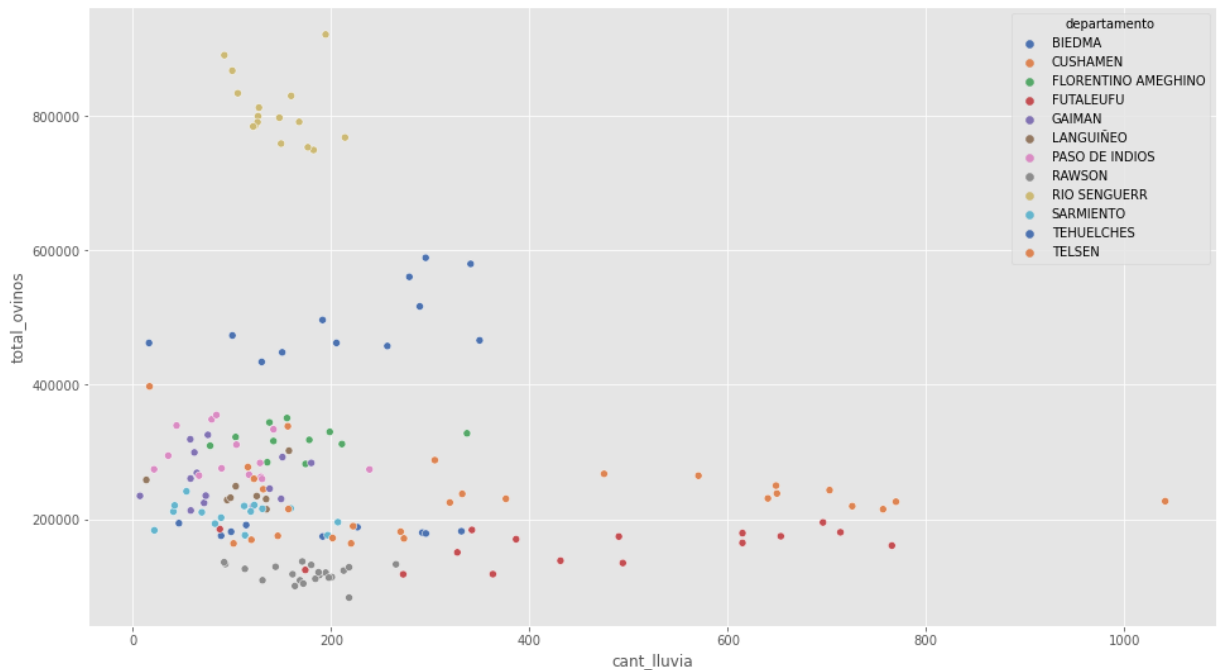
Total de ovinos

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `total_ovinos` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos

gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de ovinos y la cantidad de lluvia recibida en cada año.

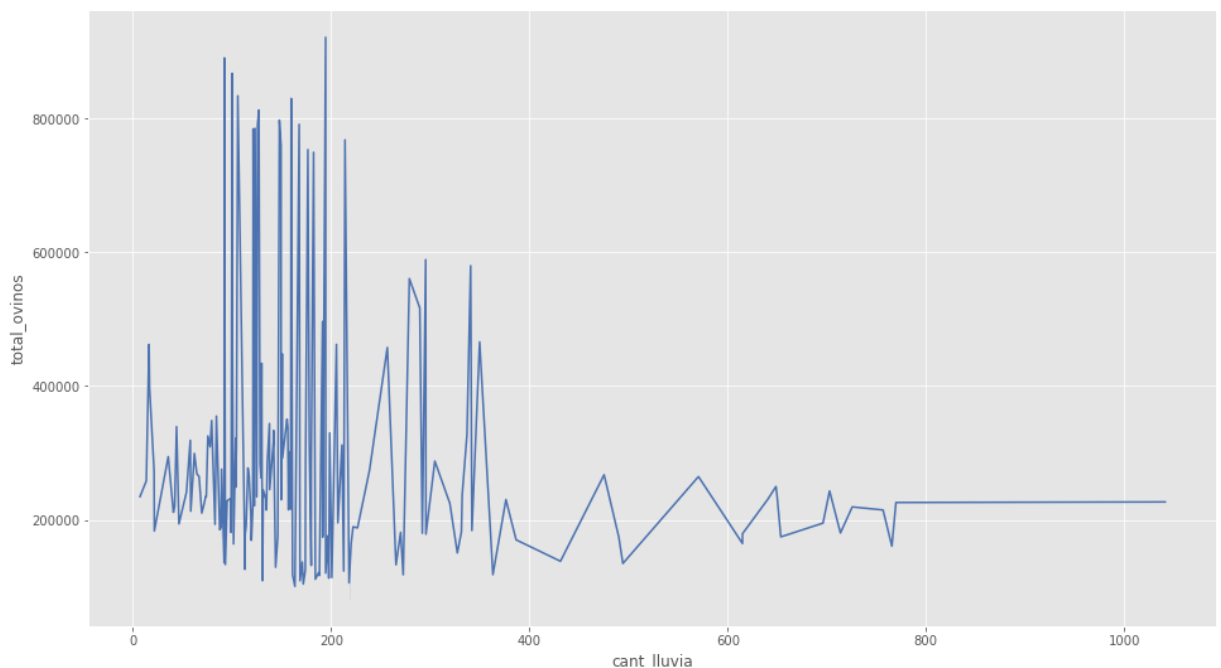
```
In [57]: sns.scatterplot(data=datos, x='cant_lluvia', y='total_ovinos',  
hue='departamento')
```

```
Out[57]: <AxesSubplot:xlabel='cant_lluvia', ylabel='total_ovinos'>
```



```
In [58]: sns.lineplot(data=datos, x='cant_lluvia', y='total_ovinos')
```

```
Out[58]: <AxesSubplot:xlabel='cant_lluvia', ylabel='total_ovinos'>
```



En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de ovinos para cada departamento. Sin embargo, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de ovinos, ya que al aumentar la lluvia, la cantidad de ovinos también se incrementa

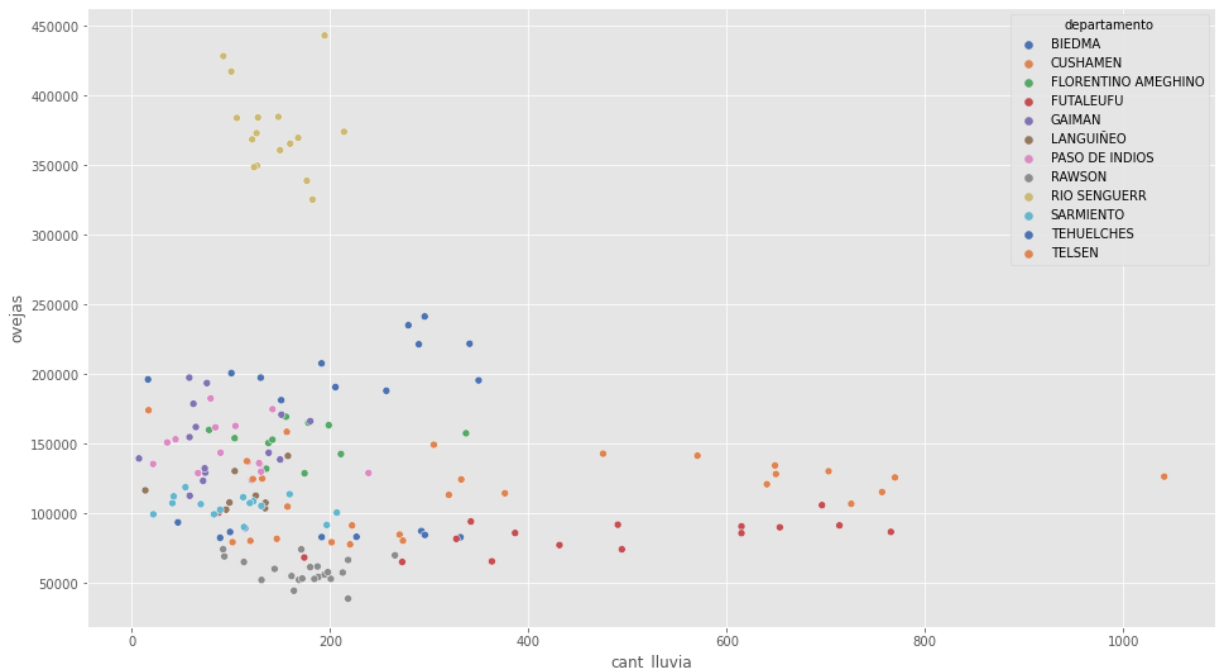
levemente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia excepcionalmente altos, pero no necesariamente tienen la mayor cantidad de ovinos. También se puede observar esta leve tendencia en el gráfico de líneas, aunque presenta muchos picos y valles que no tienen en cuenta el departamento.

Ovejas

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **ovejas** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de ovejas y la cantidad de lluvia recibida en cada año.

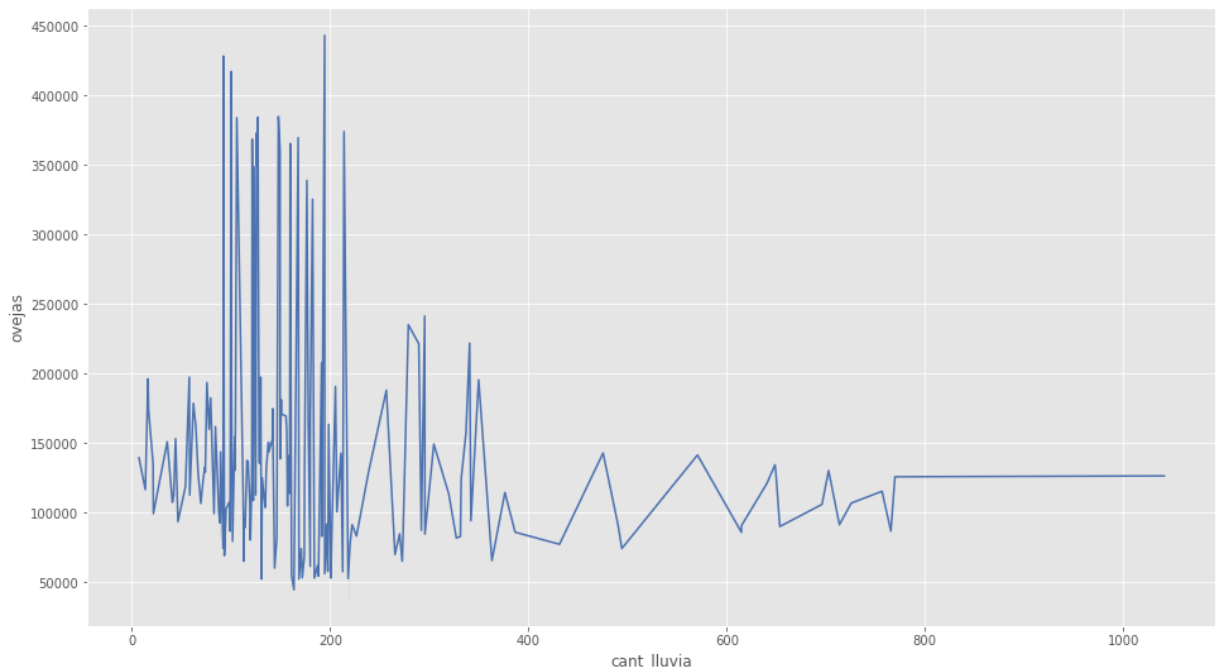
```
In [59]: sns.scatterplot(data=datos, x='cant_lluvia', y='ovejas',  
hue='departamento')
```

```
Out[59]: <AxesSubplot:xlabel='cant_lluvia', ylabel='ovejas'>
```



```
In [60]: sns.lineplot(data=datos, x='cant_lluvia', y='ovejas')
```

```
Out[60]: <AxesSubplot:xlabel='cant_lluvia', ylabel='ovejas'>
```



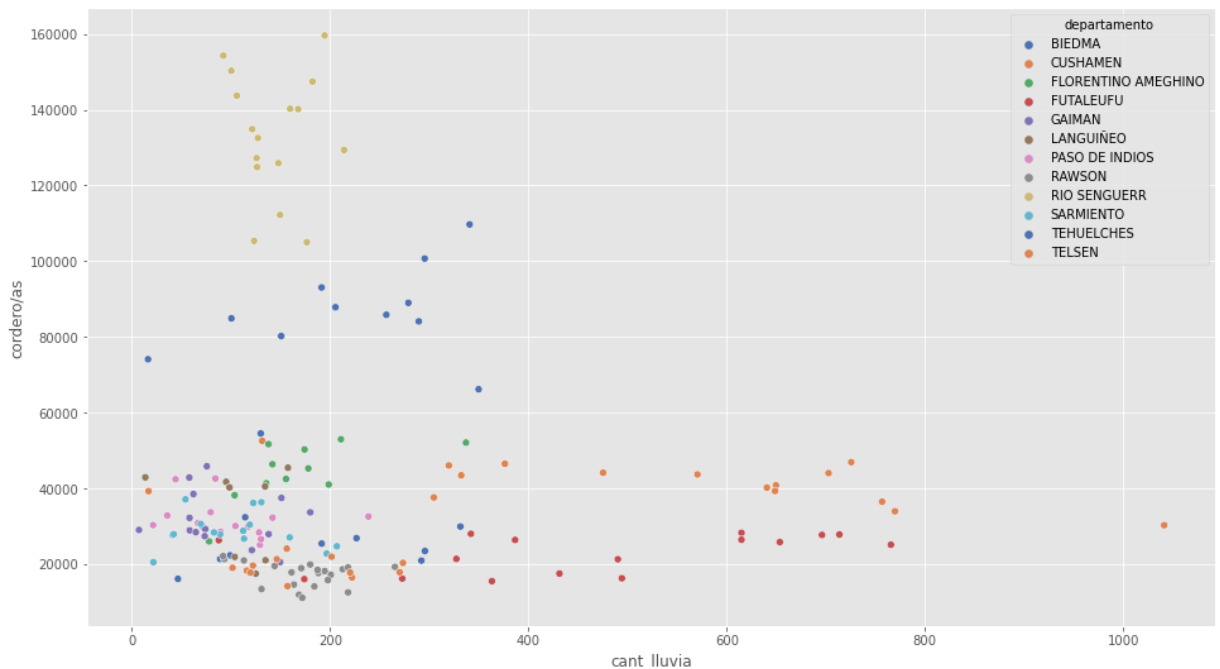
En los gráficos de dispersión y líneas que hemos analizado, no se puede apreciar una relación clara entre la cantidad de lluvia acumulada y la cantidad de ovejas en cada departamento. Sin embargo, si nos enfocamos en un departamento en particular, podemos observar una relación lineal leve entre la cantidad de lluvia y el número total de ovejas, lo que sugiere que cuando aumenta la cantidad de lluvia, también lo hace ligeramente el número de ovejas. Sin embargo, debemos tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que tienen datos atípicos de lluvia muy altos debido a su ubicación en la cordillera, pero no necesariamente tienen la mayor cantidad de ovejas. Además, el gráfico de líneas también muestra esta tendencia leve, pero con muchos picos y valles que dificultan su interpretación.

Corderos

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **corderos** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de corderos y la cantidad de lluvia recibida en cada año.

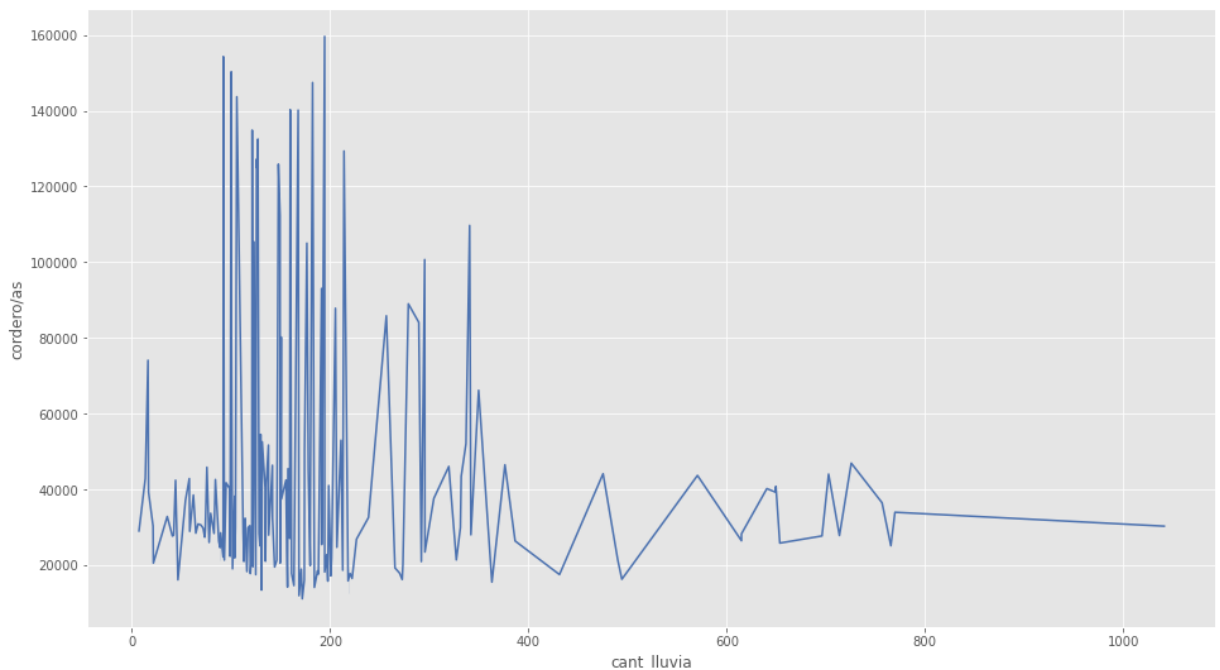
```
In [61]: sns.scatterplot(data=datos, x='cant_lluvia', y='cordero/as',
hue='departamento')
```

```
Out[61]: <AxesSubplot:xlabel='cant_lluvia', ylabel='cordero/as'>
```



```
In [62]: sns.lineplot(data=datos, x='cant_lluvia', y='cordero/as')
```

```
Out[62]: <AxesSubplot:xlabel='cant_lluvia', ylabel='cordero/as'>
```



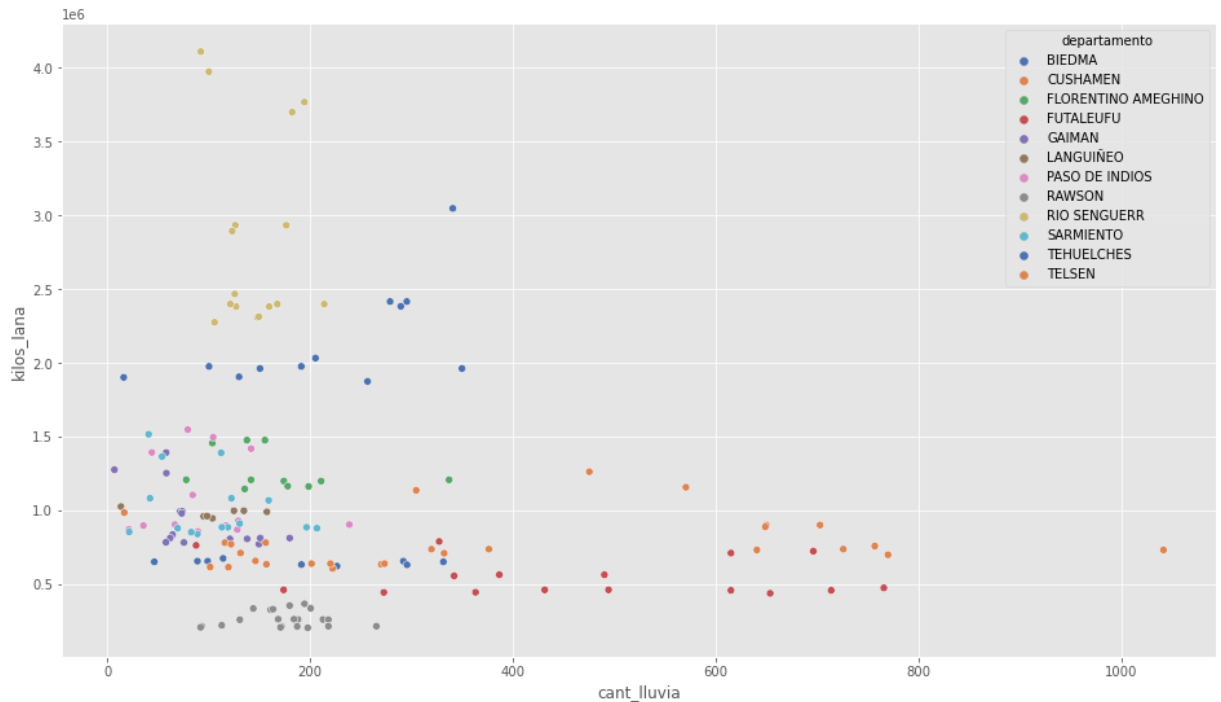
En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de corderos para cada departamento. Y sus formas son muy similares a los anteriores graficos del total de ovinos y ovejas, por lo tanto, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de corderos, ya que al aumentar la lluvia, la cantidad de corderos también se incrementa ligeramente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia, pero no necesariamente tienen la mayor cantidad de corderos. También se puede observar esta leve tendencia en el gráfico de líneas, aunque al presentar tantos picos y valles, hace difícil su interpretación.

Kilo de lana

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **kilos de lana** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de lana y la cantidad de lluvia recibida en cada año.

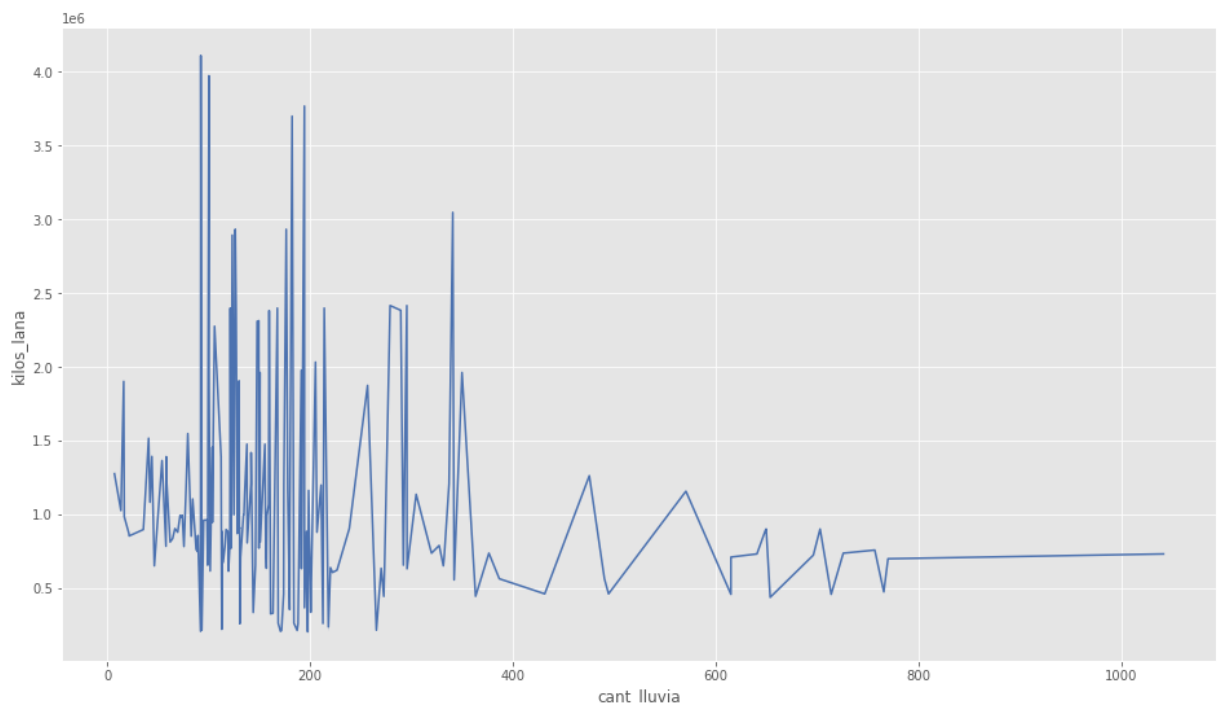
```
In [63]: sns.scatterplot(data=datos, x='cant_lluvia', y='kilos_lana',
hue='departamento')
```

```
Out[63]: <AxesSubplot:xlabel='cant_lluvia', ylabel='kilos_lana'>
```



```
In [64]: sns.lineplot(data=datos, x='cant_lluvia', y='kilos_lana')
```

```
Out[64]: <AxesSubplot:xlabel='cant_lluvia', ylabel='kilos_lana'>
```



En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de lana para cada departamento. Y sus formas son muy similares a los anteriores

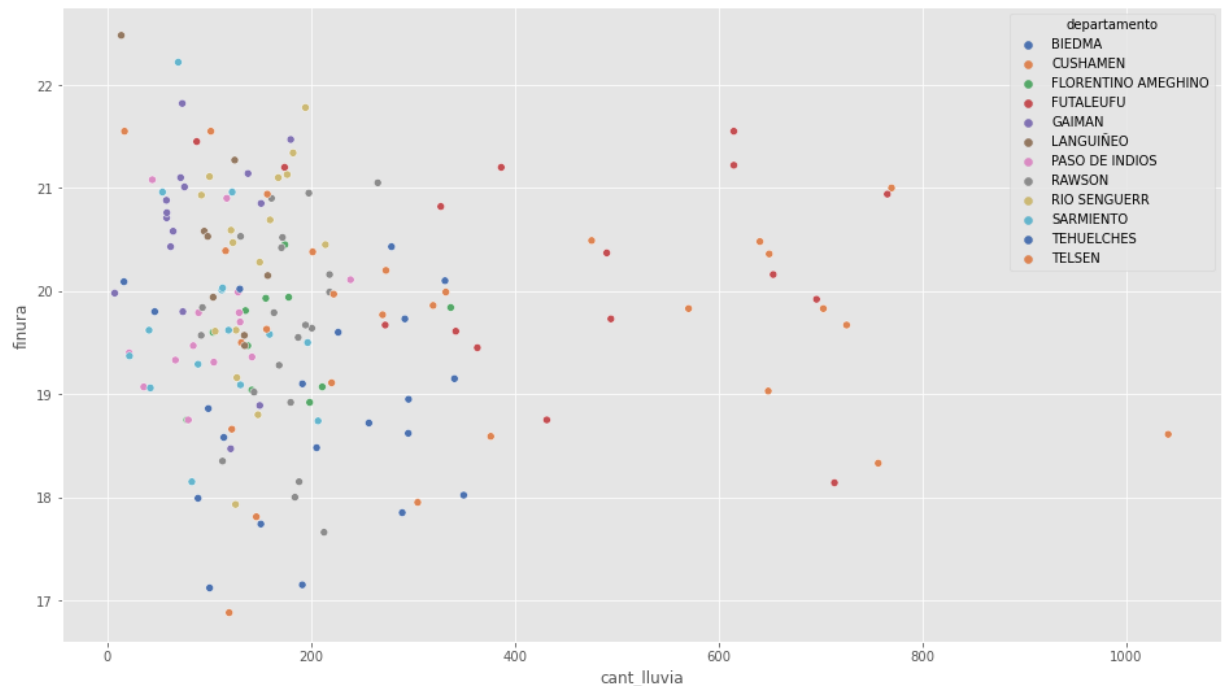
3 gráficos de existencias, por lo tanto, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de lana, ya que al aumentar la lluvia, la cantidad de lana también se incrementa ligeramente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia excepcionalmente altos, pero no necesariamente tienen la mayor cantidad de lana. También se puede observar esta leve tendencia en el gráfico de líneas, aunque presenta muchos picos y valles que no tienen en cuenta el departamento.

Finura

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `finura` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en el valor de la finura y la cantidad de lluvia recibida en cada año.

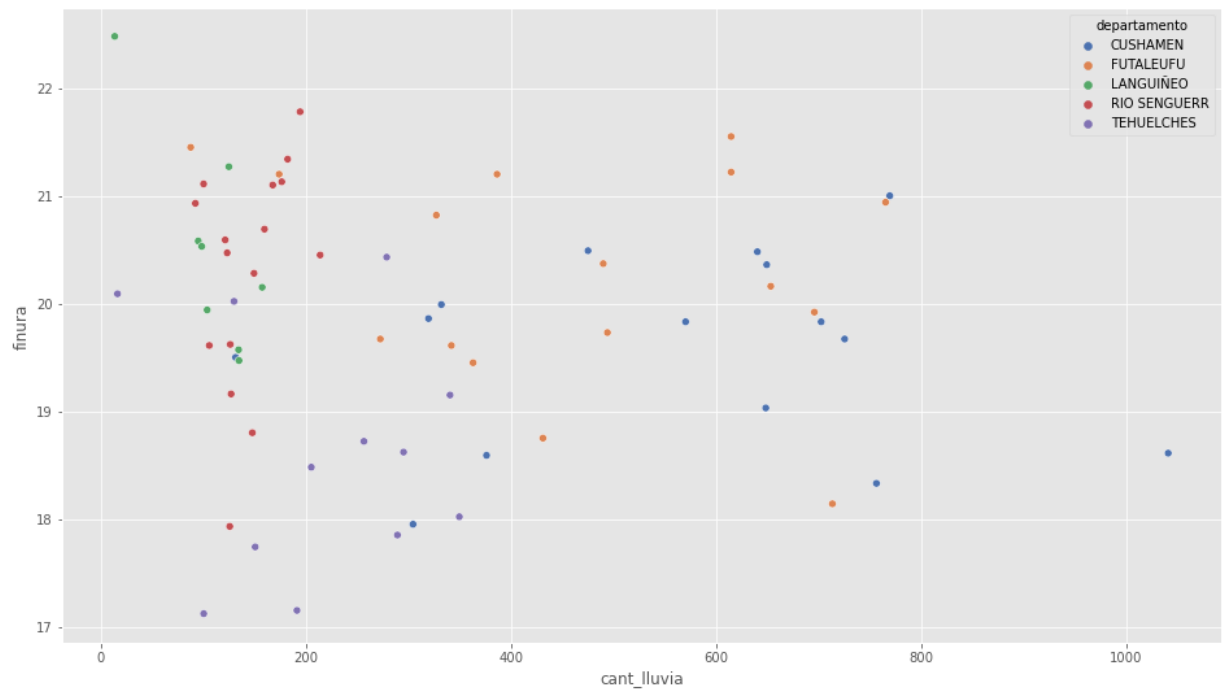
```
In [65]: sns.scatterplot(data=datos, x='cant_lluvia', y='finura',  
hue='departamento')
```

```
Out[65]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



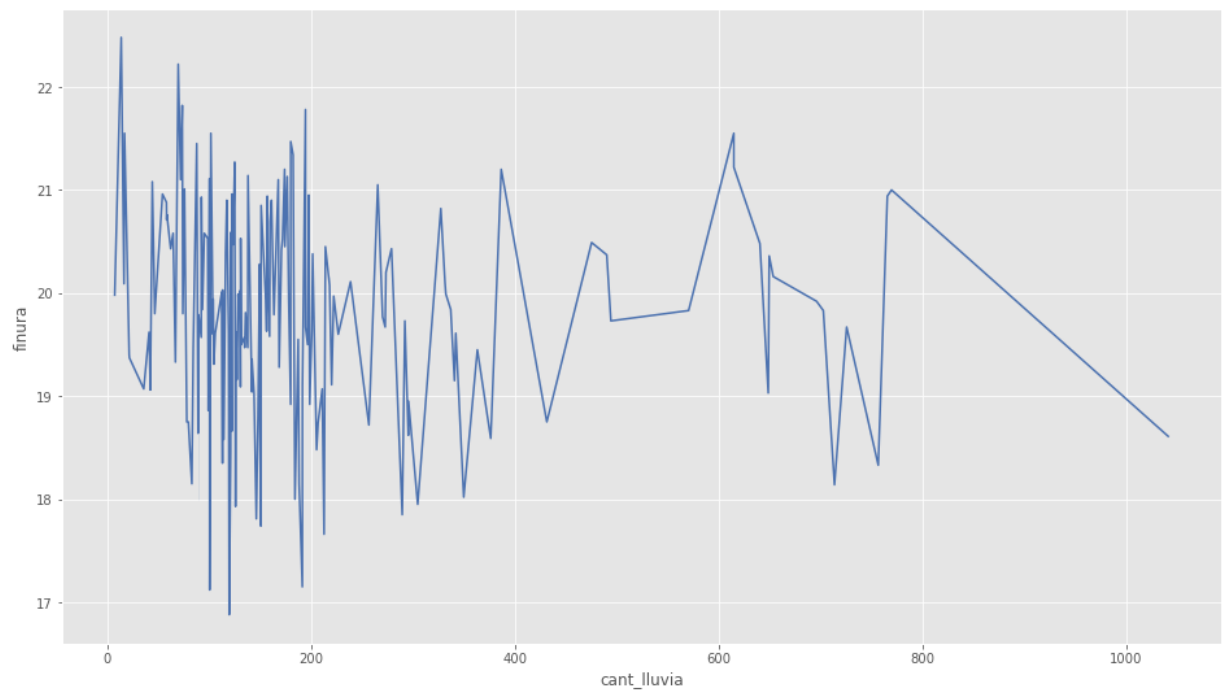
```
In [16]: sns.scatterplot(data=departamentos_cordillera, x='cant_lluvia',  
y='finura', hue='departamento')
```

```
Out[16]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



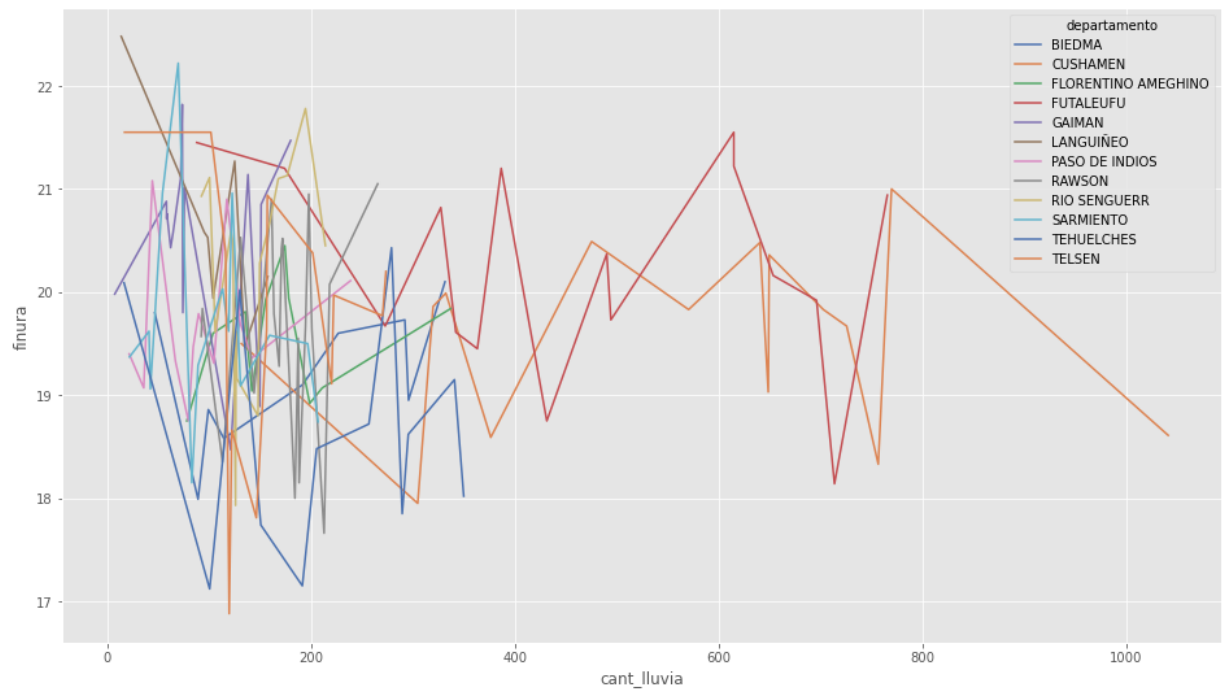
```
In [9]: sns.lineplot(data=datos, x='cant_lluvia', y='finura')
```

```
Out[9]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



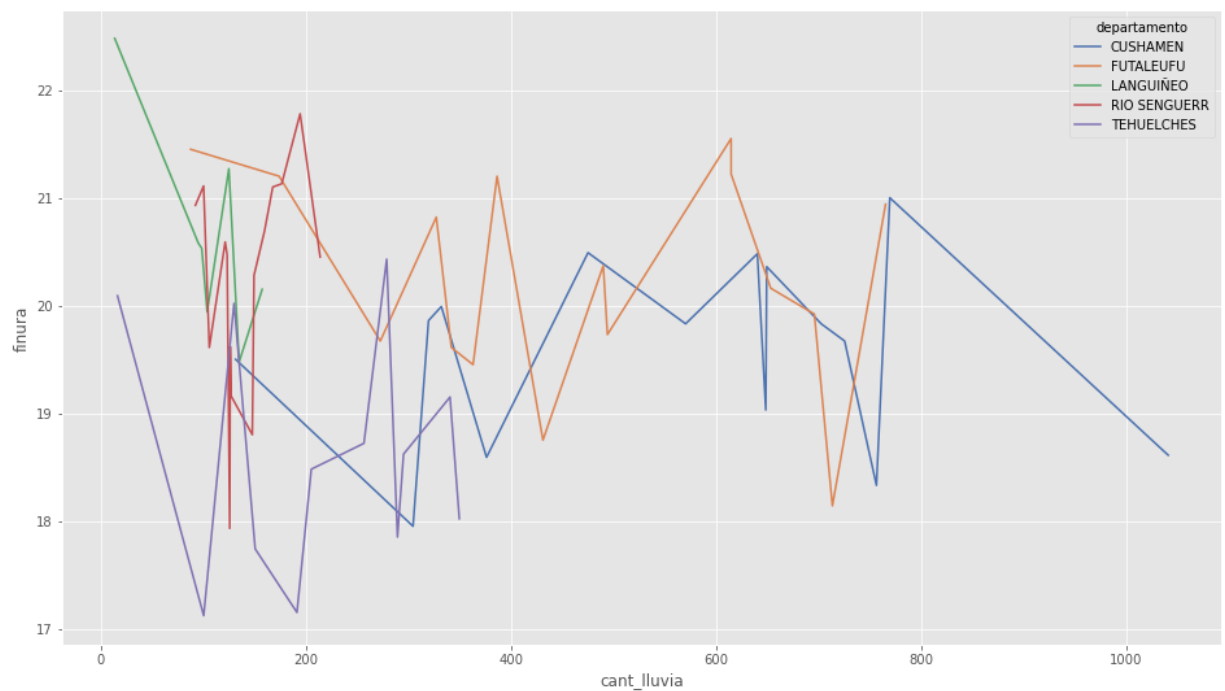
```
In [17]: sns.lineplot(data=datos, x='cant_lluvia', y='finura',  
hue='departamento')
```

```
Out[17]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



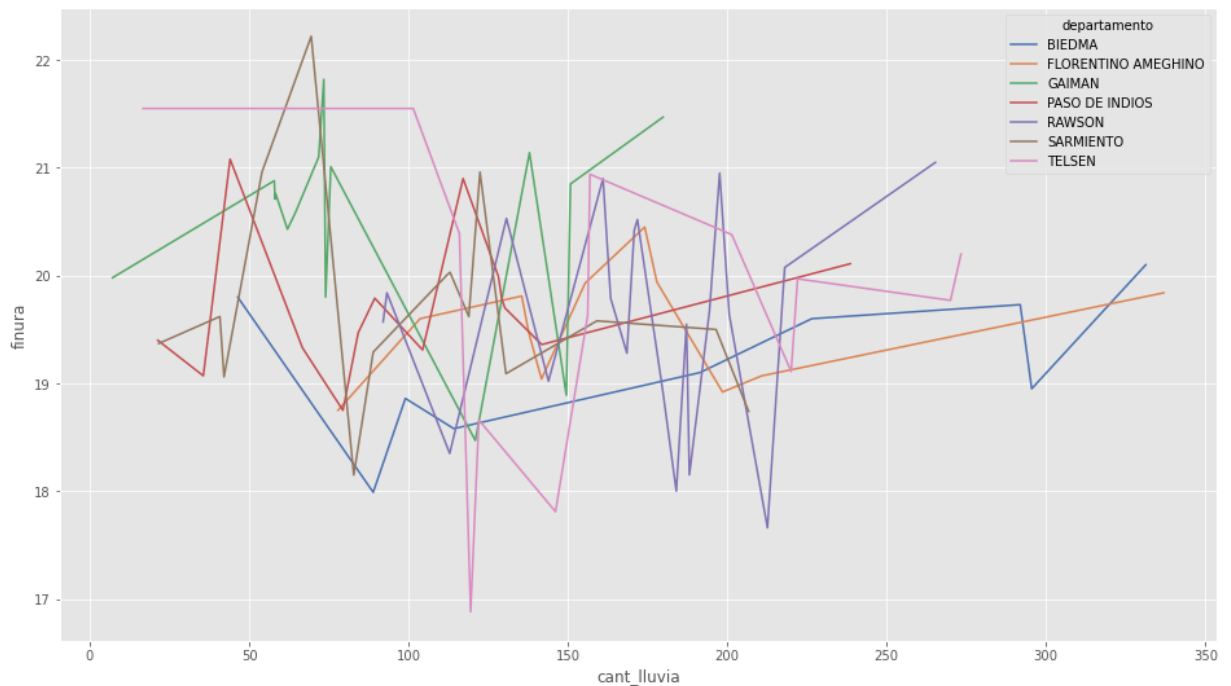
```
In [19]: sns.lineplot(data=departamentos_cordillera, x='cant_lluvia',
y='finura', hue='departamento')
```

```
Out[19]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



```
In [20]: sns.lineplot(data=departamentos_meseta, x='cant_lluvia',
y='finura', hue='departamento')
```

```
Out[20]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```

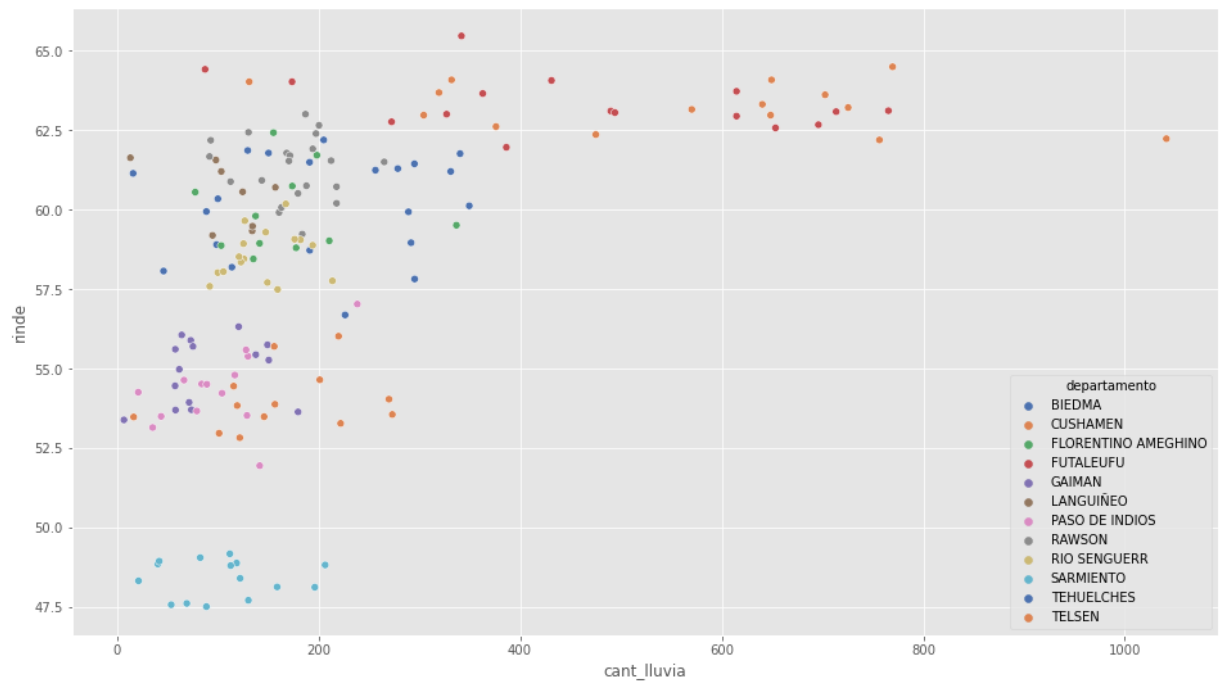
No se puede apreciar una relación clara entre la finura de la lana y la cantidad de lluvia acumulada en el año, y se observa que en algunos casos, a medida que aumenta la cantidad de lluvia, la lana se vuelve más gruesa, que no necesariamente es lo que buscan los productores ganaderos. En el gráfico de líneas no se observa ningún patrón discernible y en el gráfico de dispersión, la mayoría de los datos se concentran en un rango estrecho entre 0 y 400 mm de lluvia y una finura de lana entre 19.2 y 21.8 micrones, lo que sugiere que hay poca variabilidad en los datos. Sin embargo, existen valores atípicos en los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen una mayor cantidad de lluvia en promedio.

Rinde

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `rinde` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en el valor del rinde y la cantidad de lluvia recibida en cada año.

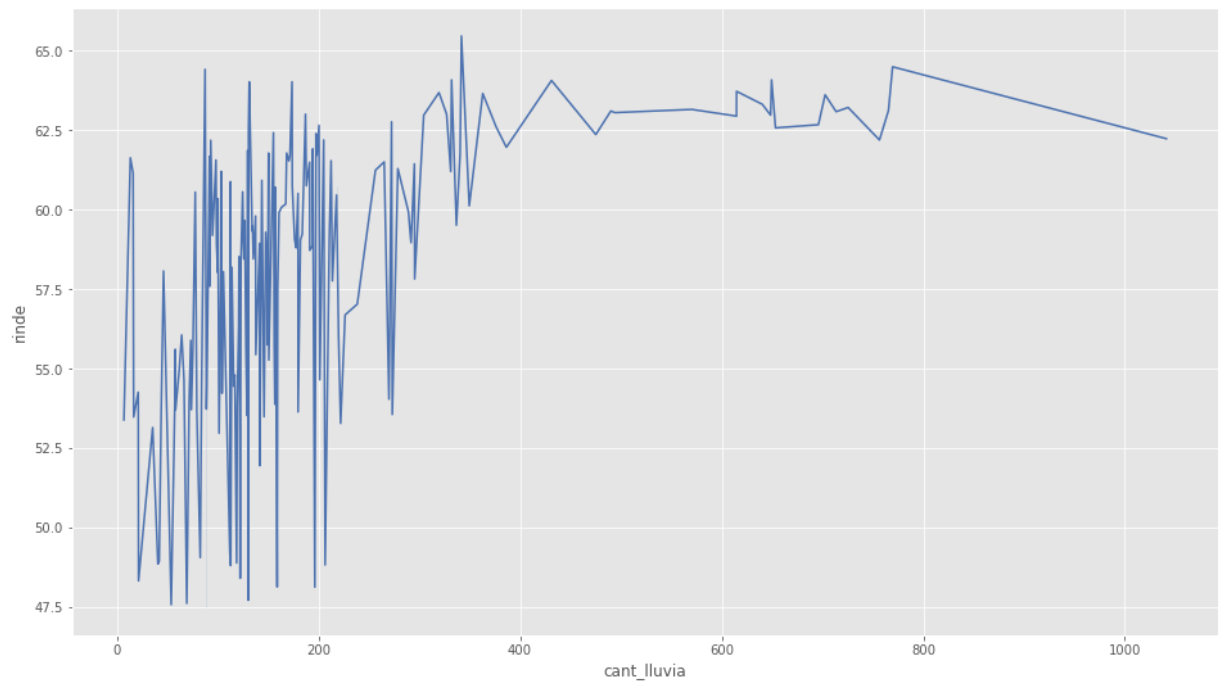
```
In [67]: sns.scatterplot(data=datos, x='cant_lluvia', y='rinde',
hue='departamento')
```

```
Out[67]: <AxesSubplot:xlabel='cant_lluvia', ylabel='rinde'>
```



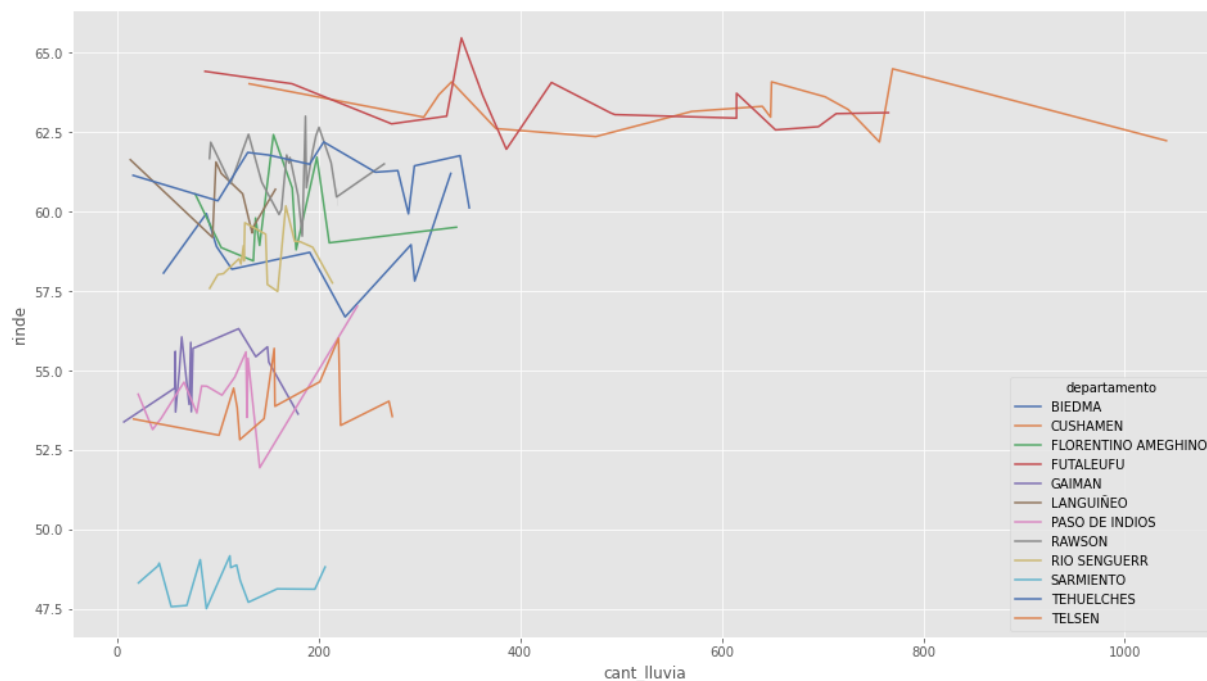
```
In [68]: sns.lineplot(data=datos, x='cant_lluvia', y='rinde')
```

```
Out[68]: <AxesSubplot:xlabel='cant_lluvia', ylabel='rinde'>
```



```
In [7]: sns.lineplot(data=datos, x='cant_lluvia', y='rinde',  
hue="departamento")
```

```
Out[7]: <AxesSubplot:xlabel='cant_lluvia', ylabel='rinde'>
```



En cuanto al rendimiento, tampoco se observa una relación clara con la cantidad de lluvia, aunque sí se puede ver una relación un poco más directa que en la finura de la lana. En este caso, cuando aumenta la cantidad de lluvia, también aumenta el rendimiento, lo cual es algo que los productores de ganadería ovina buscan. Esto podría significar que la lana está menos sucia y, por lo tanto, rinde más al lavarla. Sin embargo, los datos se mantienen constantes en la mayoría de los departamentos, independientemente de la cantidad de lluvia. En Cushamen y Futaleufú, que son los departamentos con la mayor cantidad de lluvia, se obtiene el mayor rendimiento, alcanzando un máximo de 64.

2.5. Análisis Gráfico - Dataset 2

A continuación, realizaremos un análisis visual del conjunto de datos con el objetivo de identificar patrones o relaciones entre las variables. Para ello, haremos uso de las librerías Pandas, Seaborn y Matplotlib.

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

```
In [2]: %matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

```
In [3]: datos =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_mirab
```

Para que los graficos funcionen mejor es necesario que el atributo fecha sea de tipo date.

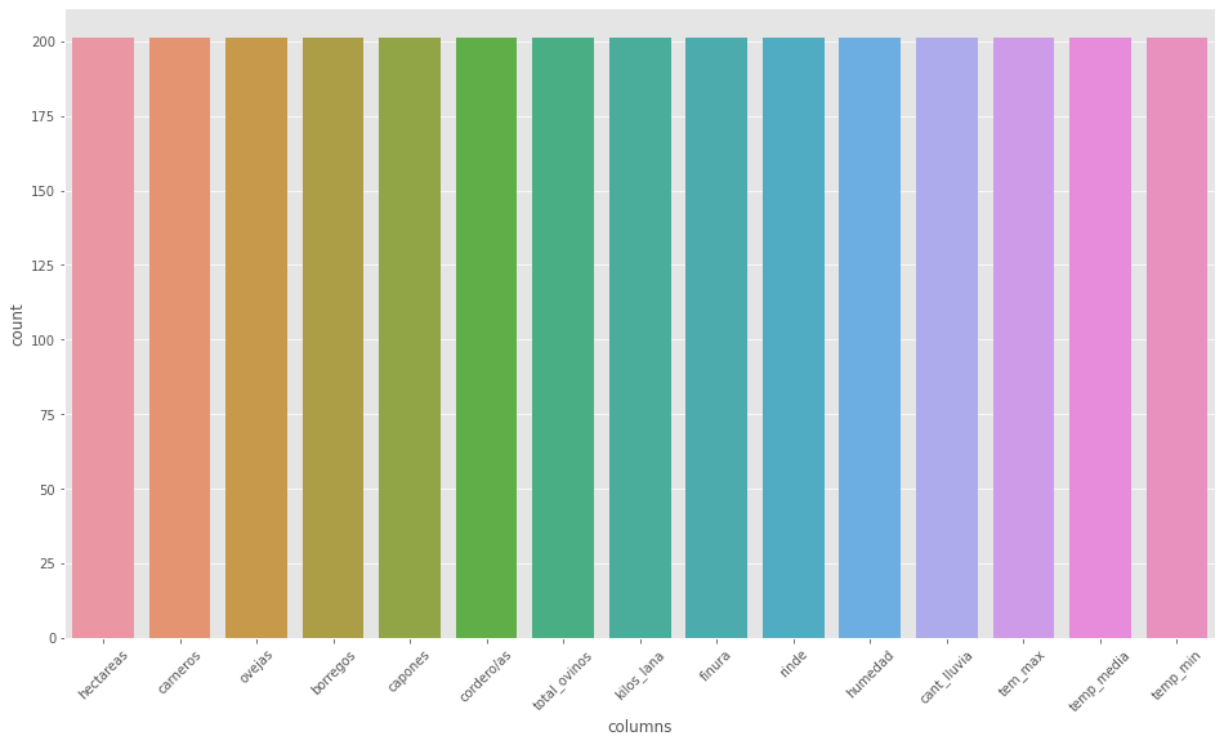
```
In [4]: datos['fecha'] = pd.to_datetime(datos['fecha'])
```

En primer lugar, realizaremos un análisis conjunto de las variables para tener una visión general del dataset. A continuación, procederemos a analizar las variables de forma individual para obtener una comprensión más detallada de cada una de ellas. Por último, relacionaremos los atributos del conjunto de datos para identificar patrones y posibles relaciones entre ellos.

Variables en conjunto

En primer lugar, realizaremos un gráfico que nos permitirá conocer la cantidad de datos disponibles para cada atributo del dataset, observando que cada columna cuenta con 201 datos.

```
In [5]: sns.countplot(data=datos)
plt.xticks(rotation=45)
plt.xlabel('columns')
plt.show()
```



Luego, utilizaremos la función `pairplot` de `seaborn` para obtener una matriz que muestra un gráfico de dispersión para cada combinación de atributos. Dicha matriz es bastante amplia debido a la cantidad de variables presentes en el dataset, pero podemos identificar algunas gráficas que parecen ajustarse a una regresión lineal en principio, principalmente aquellas relacionadas con las existencias de ovinos, como carneros, ovejas, borregos, capones, corderos, coderas y el total de ovinos.

```
In [6]: sns.pairplot(datos, hue="departamento")
```

```
Out[6]: <seaborn.axisgrid.PairGrid at 0x7f441536d100>
```



Asimismo, se pueden apreciar variables relacionadas con la temperatura, como la temperatura máxima, media y mínima, que también muestran una tendencia lineal. Sin embargo, al combinar datos de producción con información climática, la mayoría de las columnas no parecen ajustarse a un modelo lineal en principio.

Para mejorar la visualización de las relaciones entre variables, vamos a crear una matriz de correlación y graficarla en un mapa de calor de seaborn. De esta forma, podremos identificar de manera más clara cuáles son las variables que tienen una mayor o menor correlación entre sí.

Una matriz de correlación es una tabla que muestra las correlaciones entre todas las variables de un conjunto de datos. Las correlaciones miden el grado en que dos variables están relacionadas entre sí, y pueden oscilar entre -1 y 1.

Un valor de correlación de 1 significa que dos variables están perfectamente correlacionadas positivamente, es decir, que cuando una aumenta, la otra también lo hace en la misma medida. Por otro lado, un valor de -1 indica una correlación perfectamente negativa, lo que significa que cuando una variable aumenta, la otra disminuye en la misma medida. Finalmente, un valor de 0 indica que no hay correlación entre dos variables.

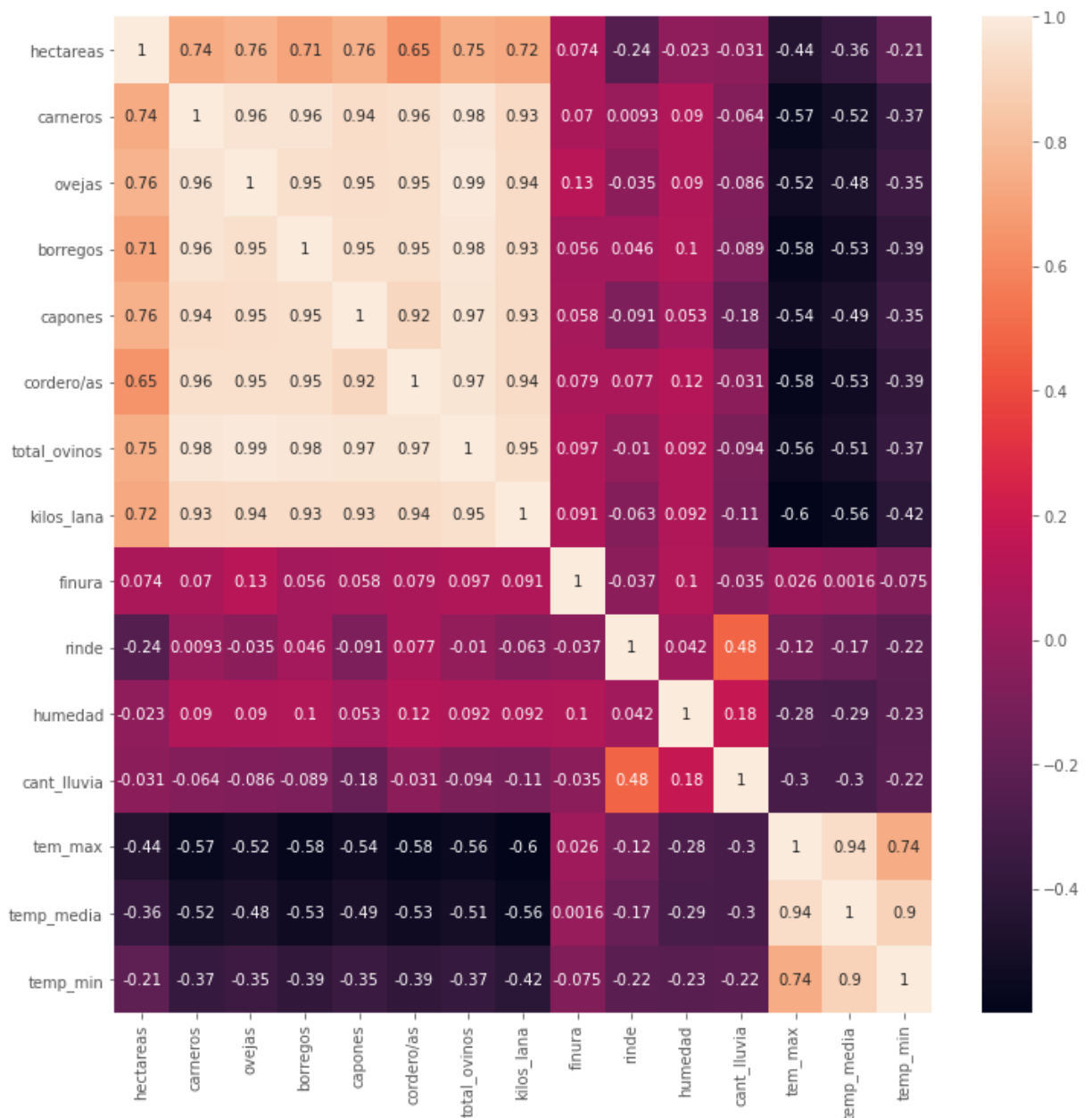
La matriz de correlación se representa como una tabla cuadrada, donde cada variable se encuentra tanto en la fila como en la columna de la matriz. En la diagonal principal, los valores son siempre 1, ya que representa la correlación de una variable consigo misma. Los valores fuera de la diagonal principal

indican la correlación entre dos variables diferentes, y su posición en la matriz es simétrica, es decir, el valor de la fila i y la columna j es el mismo que el valor de la fila j y la columna i .

En la matriz de correlación, representada por un gráfico de calor, se puede observar la intensidad de la correlación entre las diferentes variables del dataset. De acuerdo con la documentación de seaborn para el heatmap, las correlaciones negativas se muestran en violeta oscuro, las correlaciones positivas en naranja claro y las correlaciones cercanas a cero en rosado.

```
In [7]: correlation_mat = datos.corr()
fig, ax = plt.subplots(figsize = (12, 12))
sns.heatmap(correlation_mat, annot = True, ax = ax)
```

Out[7]: <AxesSubplot:>



En este caso, se puede observar que los atributos relacionados con las existencias ovinas tienen una correlación muy alta entre ellos, con valores que oscilan entre 0.93 y 0.99. Además, los atributos relacionados con la temperatura también presentan una alta correlación entre ellos, con valores que van desde 0.74 a 0.94. Estos atributos son los que previamente se habían identificado como los que seguían una forma lineal en el gráfico de dispersión.

Otra correlación interesante a destacar es la que se presenta entre la cantidad de kilos de lana producidos y las existencias ovinas, ya que se observa una alta correlación con valores que oscilan entre 0.93 y 0.95.

También las hectareas parecen tener una correlación positiva alta con las existencias ovina y la producción de lana, con valores entre 0.65 y 0.76

Teniendo en cuenta la bibliografía y la experiencia de algunos productores, se sabe que la lluvia es un factor determinante en la producción ganadera ovina de la región, debido a que se trata de una zona con bajos niveles de precipitaciones. Es por esto que resulta interesante analizar la correlación existente entre la lluvia acumulada en el año y las existencias ovinas, así como también con la producción de lana. Al observar la matriz de correlación, los valores obtenidos para estas relaciones están más cerca del color rosa que del violeta o naranja, lo que indica que no existe una correlación significativa entre estas variables. En detalle, se aprecia una correlación negativa débil entre la cantidad de lluvia y las existencias ovinas, así como también con la producción de lana. Los valores obtenidos oscilan entre -0.064 y -0.089 para carneros, ovejas y borregos, mientras que para los corderos la relación es de -0.031 y para los capones de -0.18. Los kilos de lana también muestran una correlación negativa débil de -0.11 con la cantidad de lluvia, lo que sugiere que a medida que aumenta la cantidad de lluvia en el año, la producción de lana y existencias tiende a disminuir ligeramente. En el caso de la finura se observa una correlación negativa con la cantidad de lluvia y para el rinde de la lana, se observa una correlación positiva con la cantidad de lluvia. La correlación entre la finura de la lana y la cantidad de lluvia es negativa muy débil, con un valor de -0.035, mientras que la correlación entre el rinde de la lana y la cantidad de lluvia es positiva moderada, con un valor de 0.48; esto significa que a medida que aumenta la cantidad de lluvia, el rinde también tienden a aumentar. Es importante tener en cuenta que la correlación no implica causalidad y que existen otros factores que pueden influir en esta relación.

Variables independientes

A continuación lo que se va a hacer es explicar el funcionamiento de los gráficos que se van a usar para el análisis y después vamos a graficar para cada una de las variables que se van a utilizar en los entrenamientos.

Gráficos

Histogramas

Para saber si los datos están sesgados, es necesario analizar la distribución de los datos. Una distribución sesgada indica que los datos se concentran en un lado de la distribución, lo que puede afectar la interpretación de los resultados.

Se pueden visualizar las distribuciones utilizando histogramas o gráficos de densidad para identificar visualmente si hay algún tipo de sesgo en los datos.

El comando `sns.histplot(data=datos['variable_X'])` te muestra un histograma de la columna 'variable_X' del dataframe 'datos'. En el eje `x` se encuentra el rango de valores de la variable y en el eje `y` se encuentra la frecuencia de ocurrencia de cada valor. Este tipo de gráfico es útil para visualizar la distribución de los datos y para detectar posibles valores atípicos o sesgos en la distribución.

Diagrama de caja

También, para detectar valores atípicos, se puede utilizar la visualización de un diagrama de caja (boxplot), donde se pueden identificar los valores que se encuentran por encima o por debajo de los límites del diagrama. En un boxplot, los valores atípicos se representan como puntos más allá de los límites de los bigotes del diagrama.

El comando `sns.boxplot(data=datos['variable_X'])` muestra un diagrama de caja (boxplot) que resume la distribución de los valores en la columna 'variable_X' del DataFrame 'datos'. La línea horizontal dentro de la caja representa la mediana, mientras que la caja en sí representa el rango intercuartil (IQR), es decir, el rango de valores que se encuentran entre el primer y tercer cuartil de la distribución. Los puntos fuera de la caja, llamados valores atípicos, representan valores que están más allá del IQR. Este gráfico puede ser útil para identificar valores extremos o fuera de lo común en la distribución de los datos.

Diagrama de dispersión

Con `sns.scatterplot(data=datos['variable_X'])` se mostrará un gráfico de dispersión con el eje *X* representando los índices de las filas del dataframe y el eje *Y* representando los valores de la columna 'variable_X'. Cada punto en el gráfico representa un valor de 'variable_X' para una fila específica del dataframe. El gráfico puede ser útil para visualizar la distribución y la relación entre los valores de la columna 'variable_X'.

Diagrama de línea

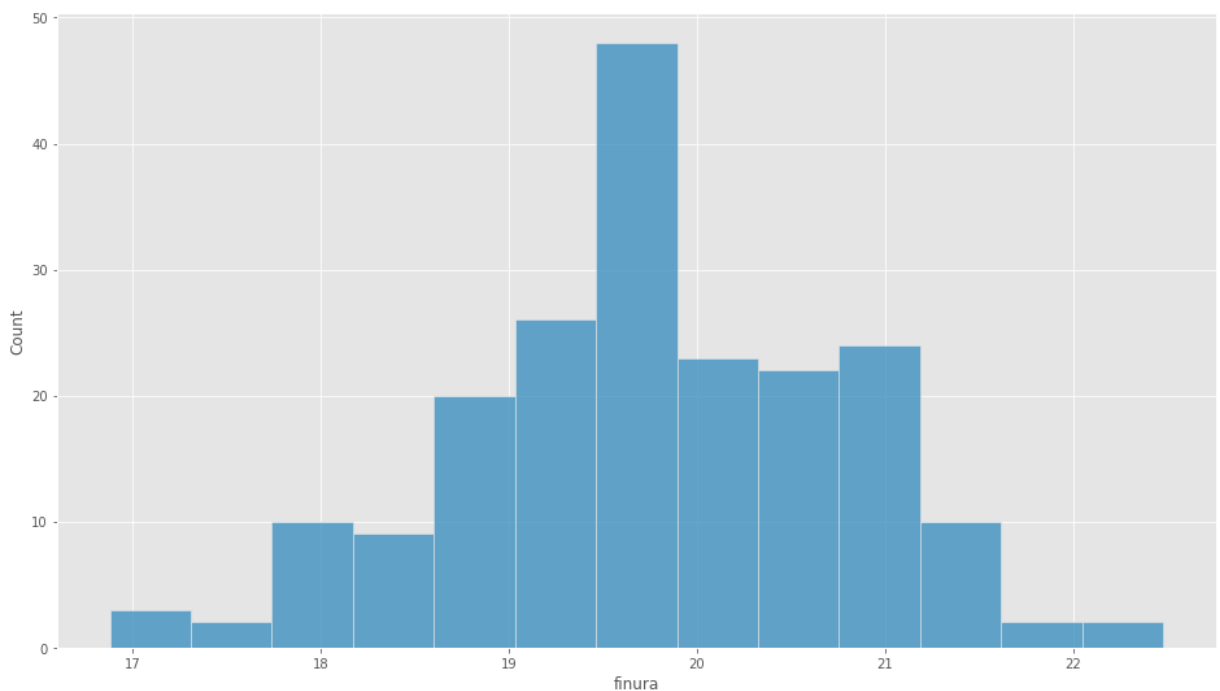
El código `sns.lineplot(data=datos['variable_X'])` genera un gráfico de línea de la columna 'variable_X' del DataFrame 'datos'. El eje *x* del gráfico representa el índice de la serie (es decir, el número de fila), y el eje *y* representa el valor de la columna 'variable_X' en cada fila. Por defecto, Seaborn dibuja una línea que conecta los puntos en orden, lo que ayuda a visualizar la tendencia a lo largo del tiempo (o cualquier otra dimensión representada por el índice).

Finura

El histograma presentado muestra que la variable `finura` presenta una distribución casi simétrica.

```
In [8]: sns.histplot(data=datos['finura'])
```

```
Out[8]: <AxesSubplot:xlabel='finura', ylabel='Count'>
```



Podemos observar en el histograma una leve asimetría hacia la izquierda, lo que se conoce como una distribución negativamente sesgada. Esta asimetría se caracteriza por la presencia de barras más altas en el lado derecho y barras más bajas en el lado izquierdo, lo que sugiere la presencia de valores extremos o atípicos que aumentan el promedio y desplazan la distribución hacia la izquierda. En otras palabras, parece que la mayor concentración de valores se encuentra en el lado derecho, mientras que el lado izquierdo muestra una menor cantidad de valores.

En el siguiente boxplot, vemos más la casi simetría para la `finura` que nos indicaba el histograma.

```
In [9]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['finura'], orient="h")

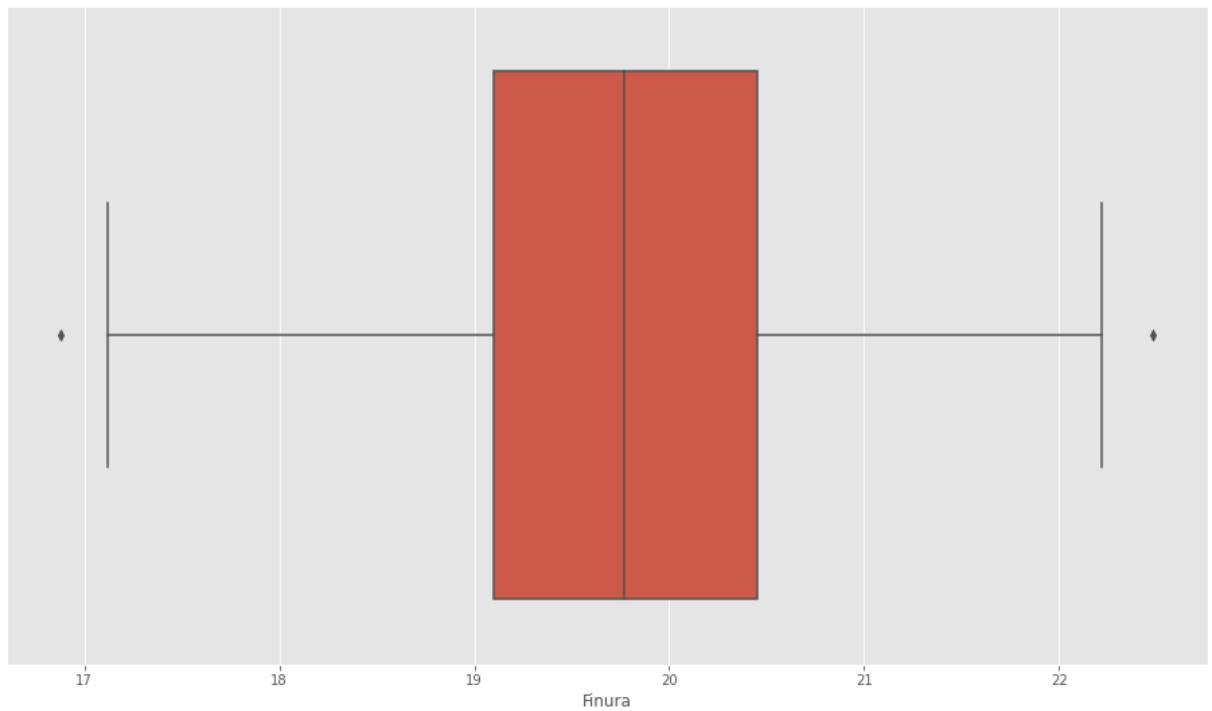
# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Finura')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

```
Valor del bigote inferior: 17.12
Valor del bigote superior: 22.22
```



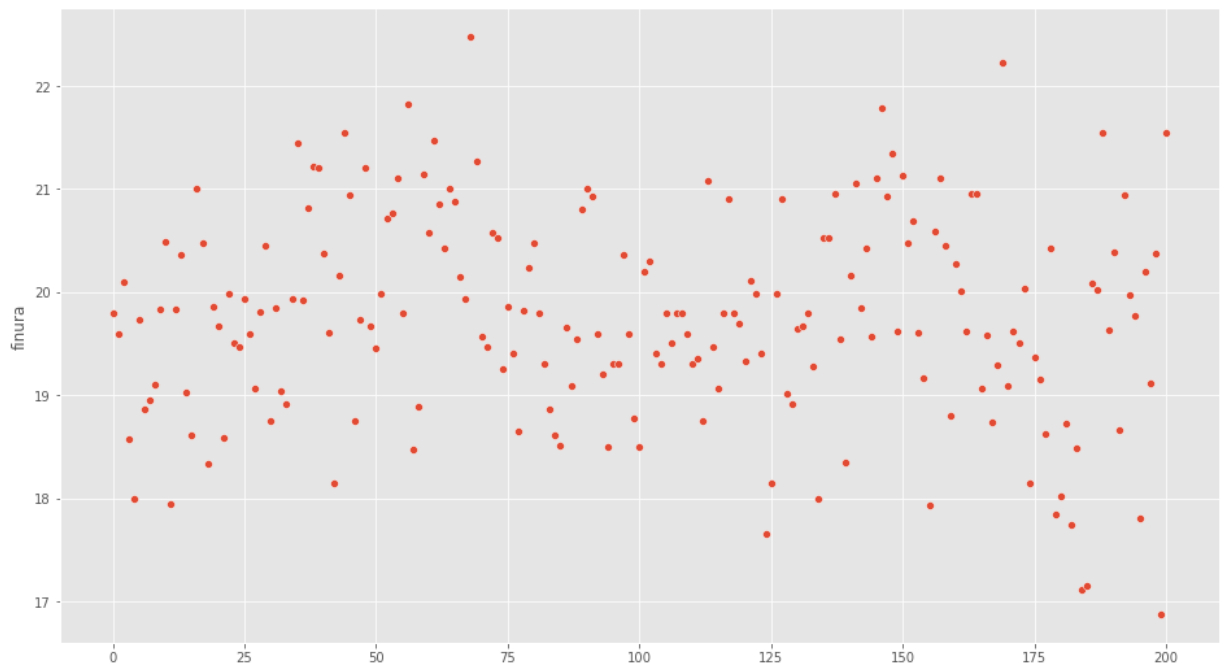
Explicación del gráfico de boxplot observado:

- Primer cuartil: el 25% de los valores son menores o iguales a 19.10 de finura.
- Segundo cuartil: el 50% de los valores son menores o iguales a 19.77 de finura.
- Tercer cuartil: el 75% de los valores son menores o iguales a 20.45 de finura.
- Mediana: a mediana de la finura de la lana es 19.77 y divide la caja en dos partes casi iguales. Sin embargo, debido a que la parte inferior de la caja es levemente más larga que la superior, se observa una asimetría negativa o sesgo hacia la izquierda.
- Bigotes: el límite inferior es de 17.12 y el límite superior es de 22.22.
- Valores atípicos: hay un valor que supera el límite inferior y otro valor que supera el límite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el total de ovinos se distribuyen en un plano cartesiano.

```
In [10]: sns.scatterplot(data=datos['finura'])
```

```
Out[10]: <AxesSubplot:ylabel='finura'>
```



```
In [11]: sns.lineplot(data=datos['finura'])
```

```
Out[11]: <AxesSubplot:ylabel='finura'>
```



En el gráfico de dispersión, se puede apreciar que no existe una clara relación entre las variables representadas en el plano cartesiano. La disposición de los puntos no sigue un patrón definido, como una línea recta o agrupaciones relevantes. Por otro lado, el gráfico de línea muestra cómo la variación en la finura de la lana cambia de una observación a otra, pero no se observa una tendencia clara. Es decir, no se puede afirmar que haya un aumento o disminución constante en la finura de la lana en las diferentes observaciones.

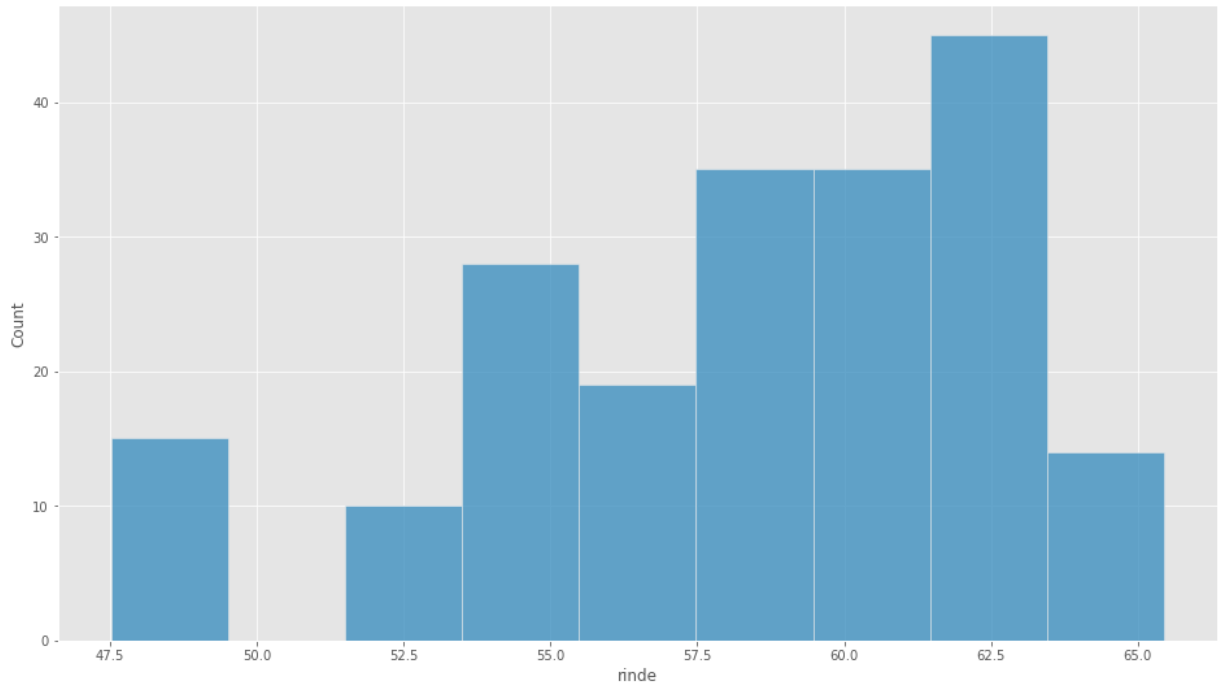
Es importante tener en cuenta que aunque no se detecte una tendencia lineal en los datos, podría existir una relación no lineal o compleja entre las variables que no es evidente en estos gráficos.

Rinde

El histograma presentado muestra que la variable `rinde` presenta una distribución asimétrica.

```
In [12]: sns.histplot(data=datos['rinde'])
```

```
Out[12]: <AxesSubplot:xlabel='rinde', ylabel='Count'>
```



Podemos hacer varias observaciones:

- La distribución tiene un sesgo hacia la izquierda.
- El valor máximo de la variable es 65.5 y el mínimo es 47.5.
- La mayoría de los valores se encuentran en el rango de 53 a 63 en el eje x, donde se concentran las barras más altas del histograma.
- En el extremo izquierdo del histograma hay una única barra correspondiente al intervalo 47.5-49.5 en el eje x, que tiene una frecuencia moderada de observaciones.

En el siguiente boxplot, vemos más claramente la asimetría para el `rinde` que nos indicaba el histograma.

```
In [13]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['rinde'], orient="h")

# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Rinde')

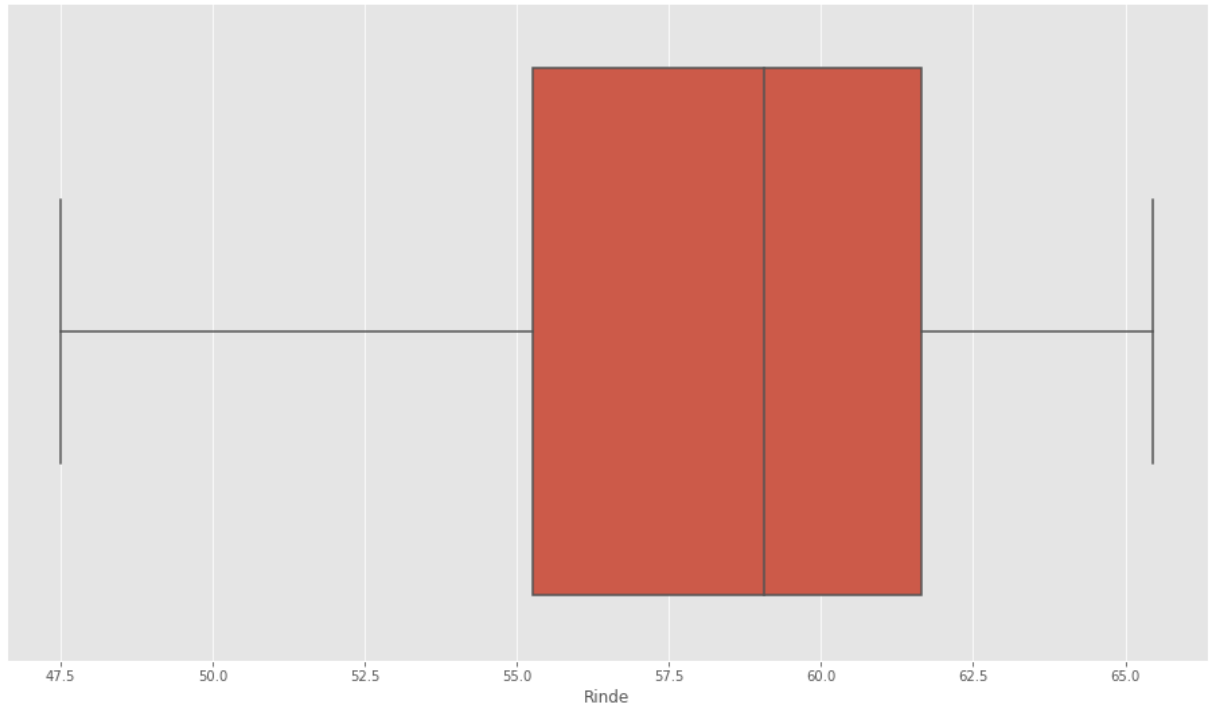
# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
```

```
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 47.51

Valor del bigote superior: 65.45



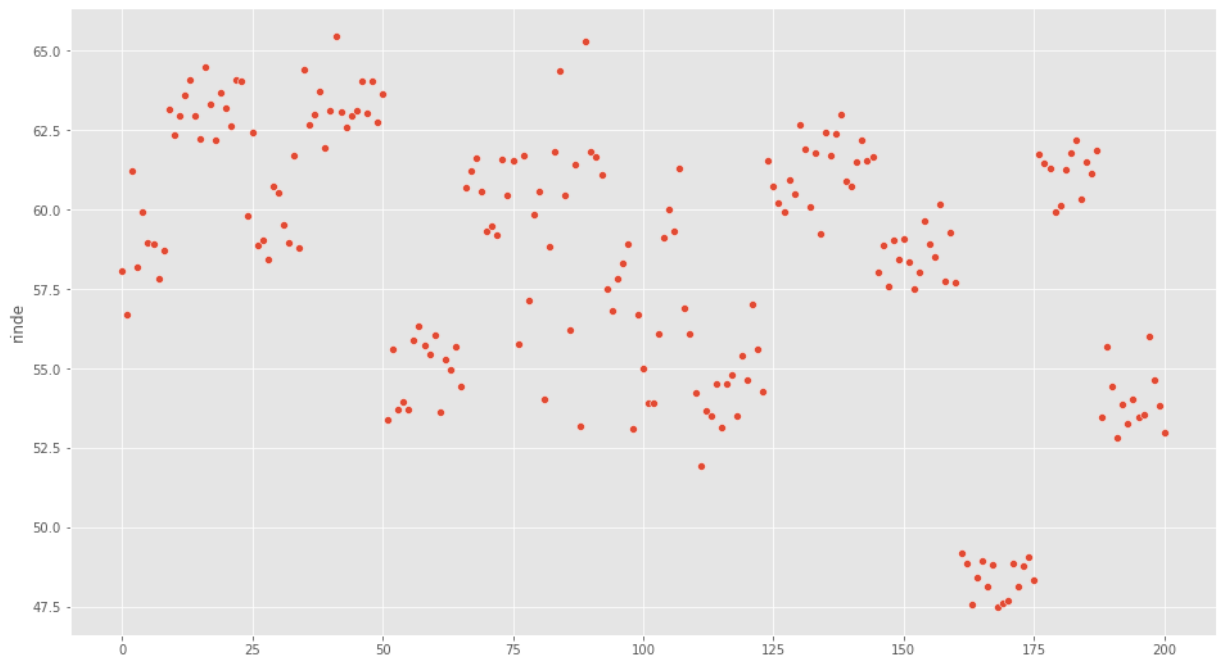
Explicación gráfico boxplot observado:

- Primer cuartil: el 25% de los valores son menores o igual a 55.26 de rinde.
- Segundo Cuartil: el 50% de los valores son menores o igual a 59.06 de rinde.
- Tercer cuartil: el 75% de los valores son menores o igual a 61.66 de rinde.
- Mediana: la mediana es 59.06 de rinde y corta la caja en dos lados desiguales, muestra una asimetría negativa o segada a la izquierda dado que la parte más larga de la caja es la parte inferior a la mediana, lo que indica que los valores más bajos son más frecuentes que los valores más altos.
- Bigotes: el limite inferior es de 47.51 y el limite superior es de 65.45.
- Valores atípicos: no hay valores que superen los limites.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el `rinde` se distribuyen en un plano cartesiano.

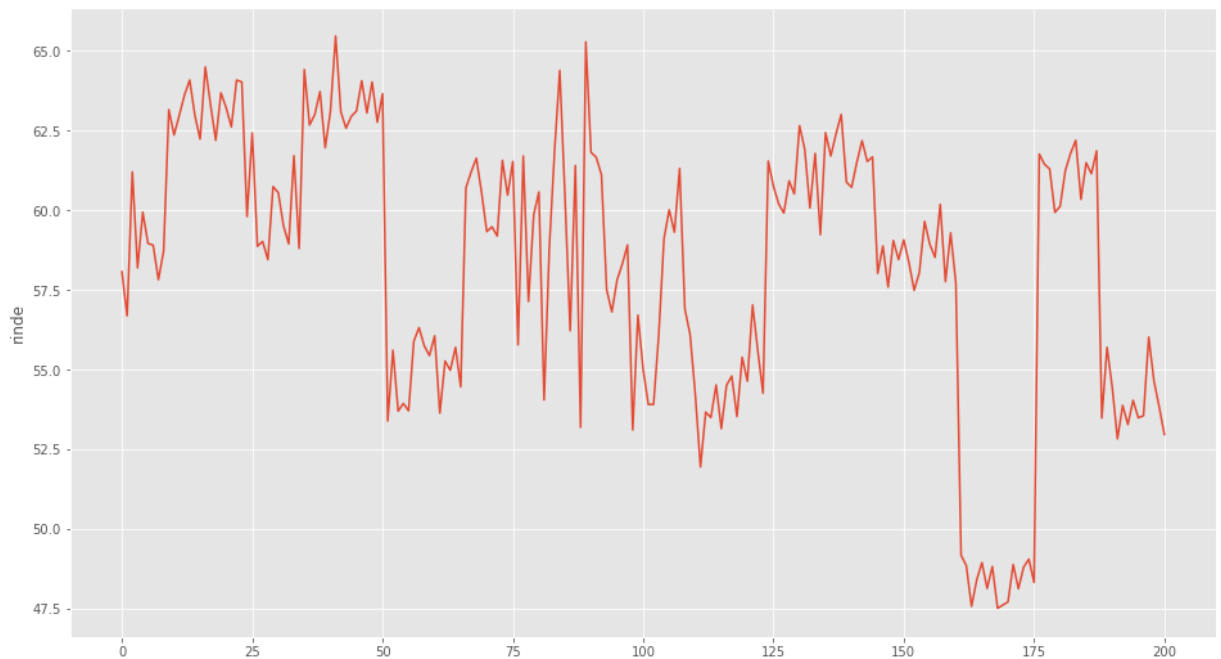
```
In [14]: sns.scatterplot(data=datos['rinde'])
```

```
Out[14]: <AxesSubplot:ylabel='rinde'>
```



```
In [15]: sns.lineplot(data=datos['rinde'])
```

```
Out[15]: <AxesSubplot:ylabel='rinde'>
```



Al examinar el diagrama de dispersión, se observa que no existe una relación evidente entre las variables representadas en el plano cartesiano. La ubicación de los puntos no sigue una pauta definida, como una línea recta, aunque se pueden identificar once pequeños conjuntos que podrían ser interesantes. En cambio, el gráfico de línea ilustra cómo el rinde de la lana varía de una observación a otra, y en general no se distingue una tendencia definida. Por lo tanto, no se puede concluir que haya una tendencia constante al alza o a la baja en el rinde en las observaciones.

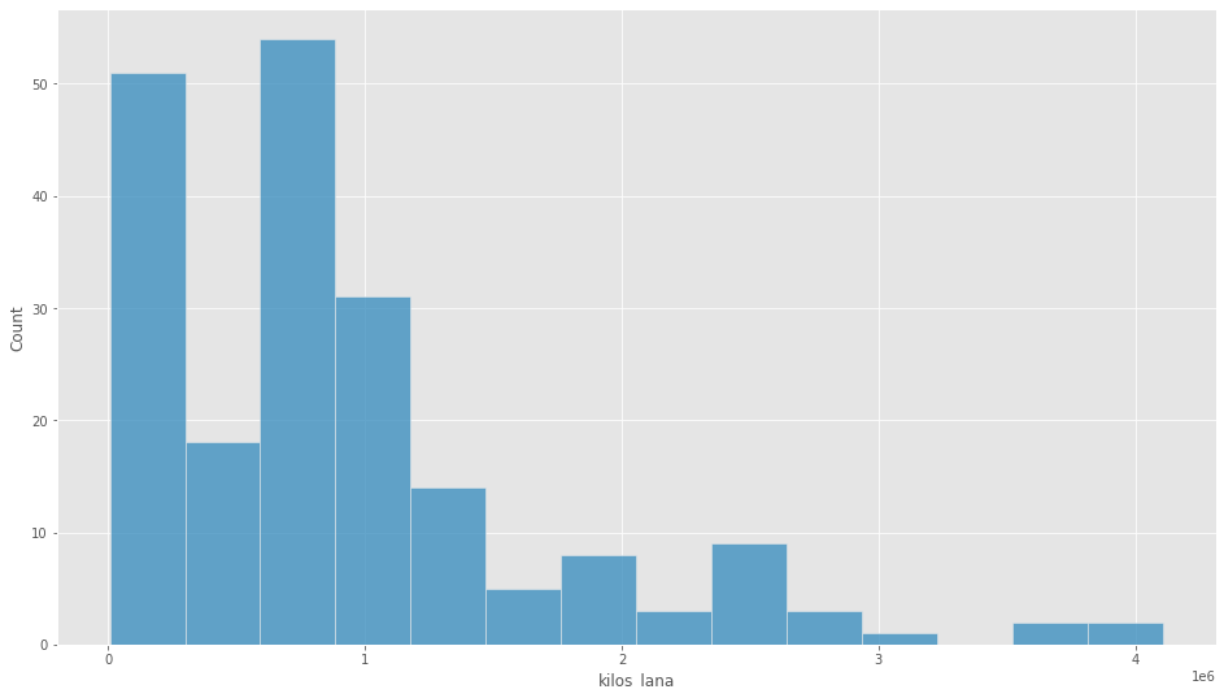
Es importante tener en cuenta que aunque no se detecte una tendencia lineal en los datos, podría existir una relación no lineal o compleja entre las variables que no es evidente en estos gráficos.

Kilogramos de lana

En el siguiente histograma vemos que los `kilos_lana` tiene una distribución asimétrica.

```
In [16]: sns.histplot(data=datos['kilos_lana'])
```

```
Out[16]: <AxesSubplot:xlabel='kilos_lana', ylabel='Count'>
```



En el histograma que se observa hay un total de 13 barras. De esas barras, 8 van desde el valor cero hasta menos de diez, y están todas situadas en el lado derecho de la distribución. Las cinco barras restantes representan valores más altos, y se ubican en el lado izquierdo del gráfico. En este caso, la distribución está sesgada positivamente, lo que significa que hay una cola larga que se extiende hacia la derecha de la distribución, y que se deben considerar los valores atípicos en ese extremo. Esta asimetría se produce porque hay más valores en el lado izquierdo de la distribución que en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el **total de kilogramos de lana** que nos indicaba el histograma.

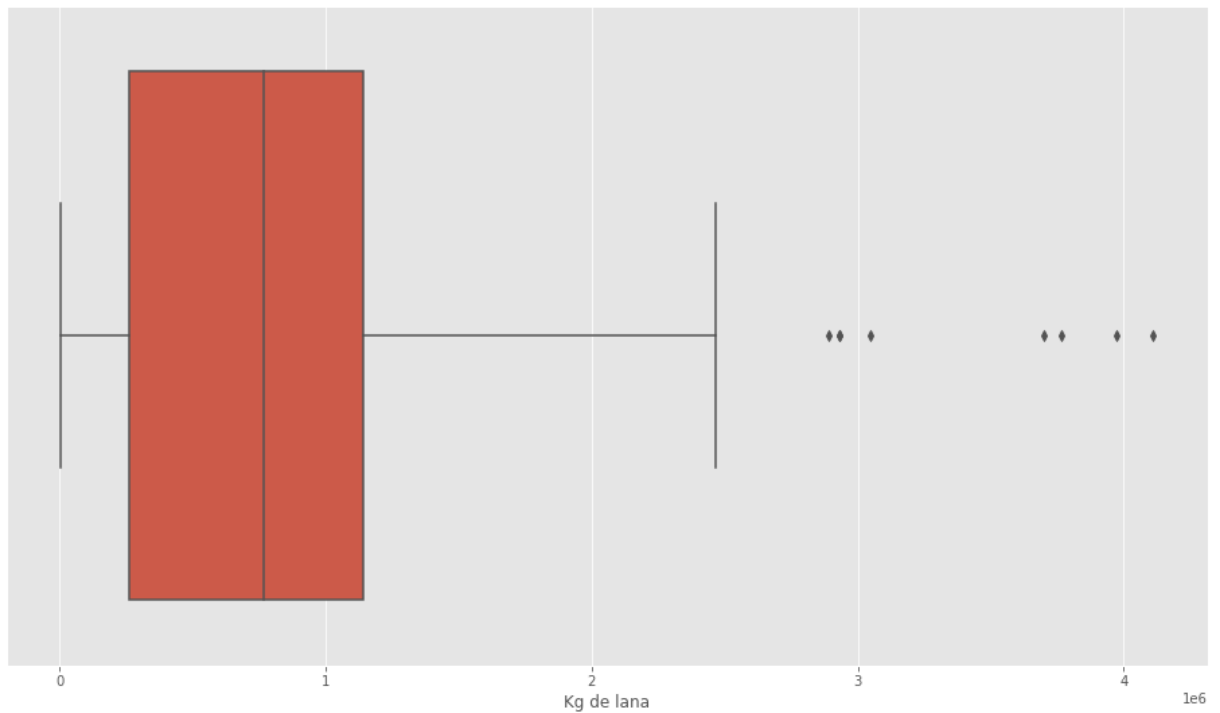
```
In [17]: ax = sns.boxplot(data=datos['kilos_lana'], orient="h")

# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Kg de lana')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)
```


Valor del bigote inferior: 5245.0
Valor del bigote superior: 2465280.0

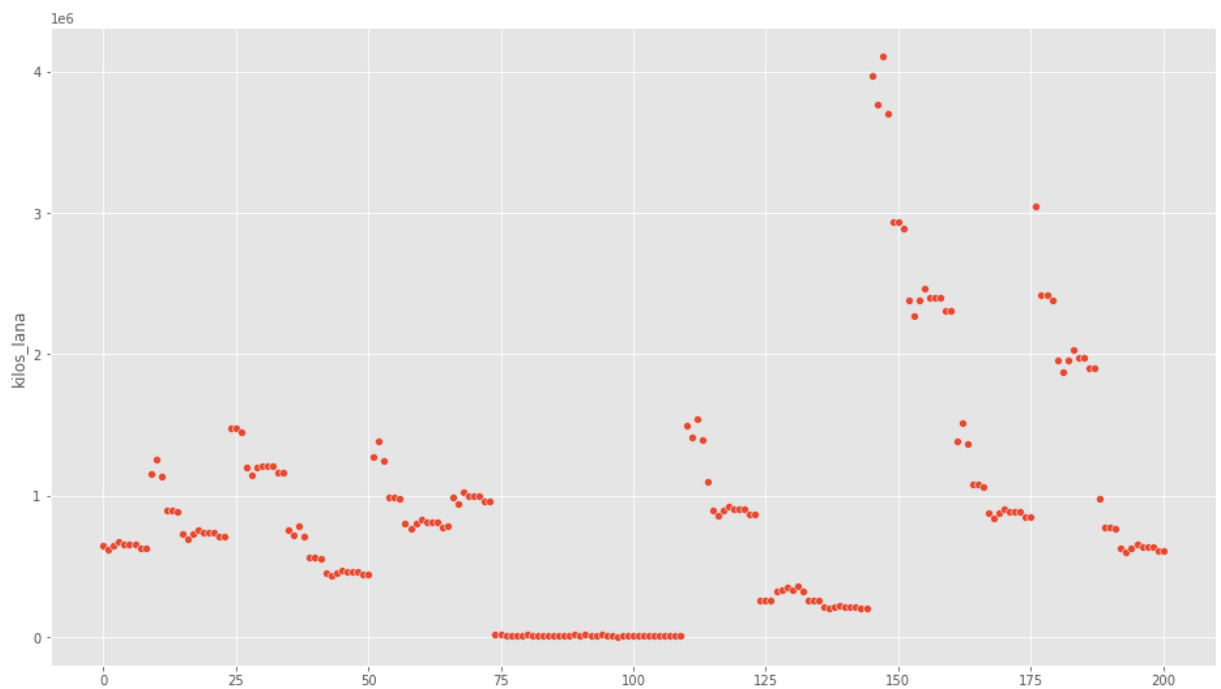


Se puede observar en el gráfico de boxplot que los datos están expresados en notación científica con una escala de millones, ya que los datos originales se transformaron de toneladas a kilogramos. El primer cuartil, que representa el 25% de los datos, tiene un valor de 260224.0 kg de lana o menos, mientras que el segundo cuartil, que representa el 50% de los datos, tiene un valor de 768912.0 kg de lana o menos. El tercer cuartil, que representa el 75% de los datos, tiene un valor de 1143408 kg de lana o menos. La mediana, tiene un valor de 768912.0 kg de lana y corta la caja en dos lados desiguales. Los bigotes, que representan el rango de los datos, tienen un límite inferior de 5245 y un límite superior de 2465280. Además, se identifican 7 valores atípicos que superan el límite superior, lo que indica la presencia de valores extremos en la distribución de los datos.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de lana** se distribuyen en un plano cartesiano.

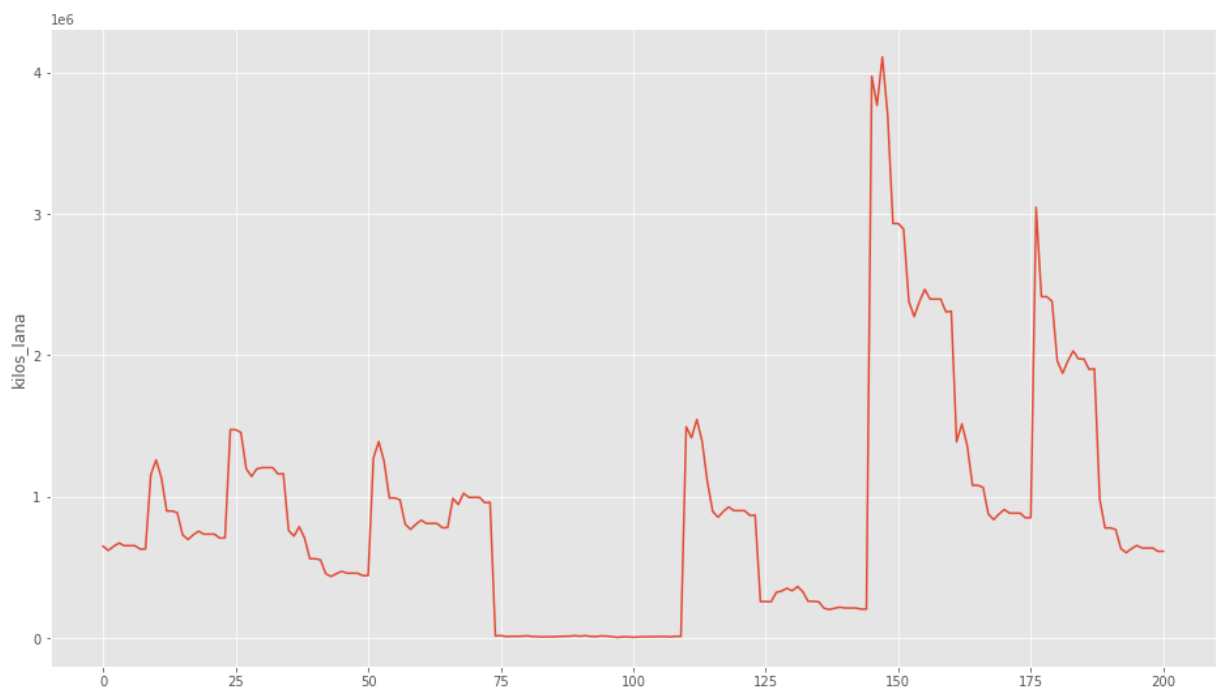
```
In [18]: sns.scatterplot(data=datos['kilos_lana'])
```

```
Out[18]: <AxesSubplot:ylabel='kilos_lana'>
```



```
In [19]: sns.lineplot(data=datos['kilos_lana'])
```

```
Out[19]: <AxesSubplot:ylabel='kilos_lana'>
```



Al analizar los datos representados en el gráfico de dispersión, se observa que los datos parecen tener una distribución dispersa sin un patrón o tendencia clara entre las variables. Sin embargo, se pueden identificar tres grupos grandes que podrían ser relevantes.

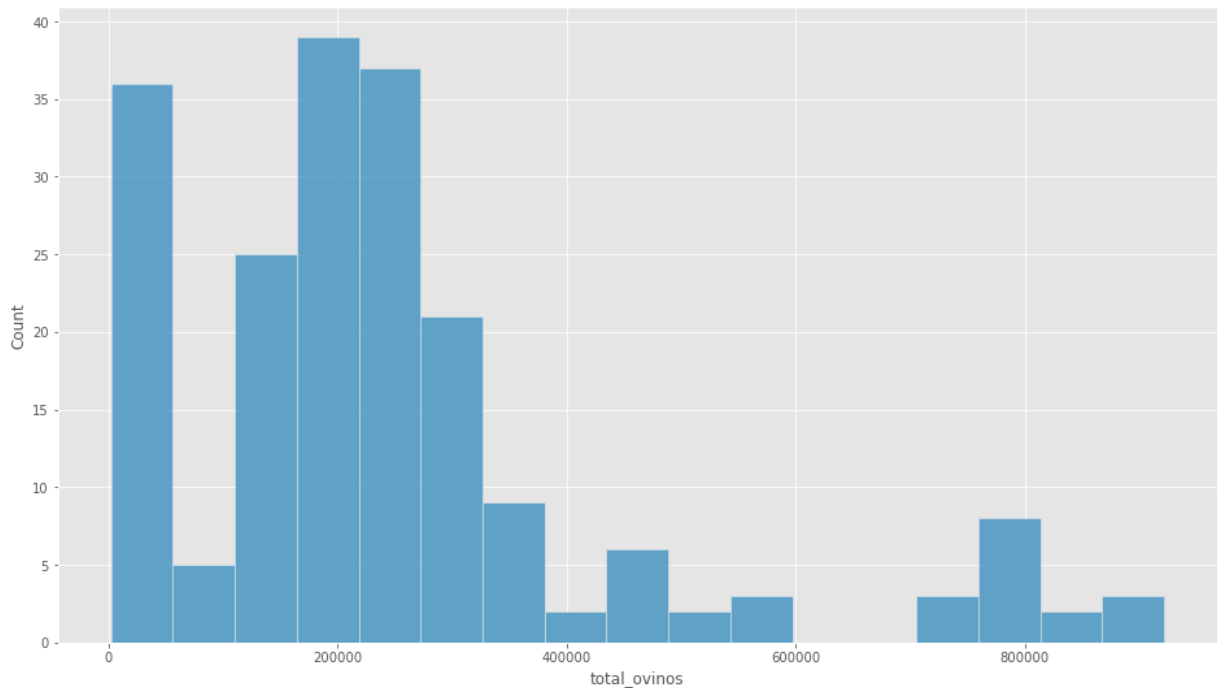
Al observar el gráfico de línea de los kilogramos de lana, se puede notar que hay una variabilidad en la producción, representada por los picos en los datos. Esto podría indicar una producción fluctuante o con una gran variabilidad. También se observa que cada vez que baja y sube la cantidad de lana los picos tienen una forma similar, sin embargo, no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Total de ovinos

En el siguiente histograma vemos que el `total_ovinos` tiene una distribución asimétrica.

```
In [20]: sns.histplot(data=datos['total_ovinos'])
```

```
Out[20]: <AxesSubplot:xlabel='total_ovinos', ylabel='Count'>
```



El gráfico representa un histograma con una distribución asimétrica derecha, lo que se puede ver debido a la presencia de barras más altas en el lado izquierdo y barras más bajas en el lado derecho. Esta distribución también se conoce como una distribución positivamente sesgada. El sesgo positivo sugiere que la cola de la distribución se extiende hacia la derecha, lo que indica la presencia de valores extremos o atípicos en el lado derecho de la distribución que aumentan el promedio y empujan la distribución hacia la derecha.

En el siguiente boxplot, vemos más claramente los datos extremos para el `total de ovinos` que nos indicaba el histograma.

```
In [21]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['total_ovinos'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 960000, 50000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

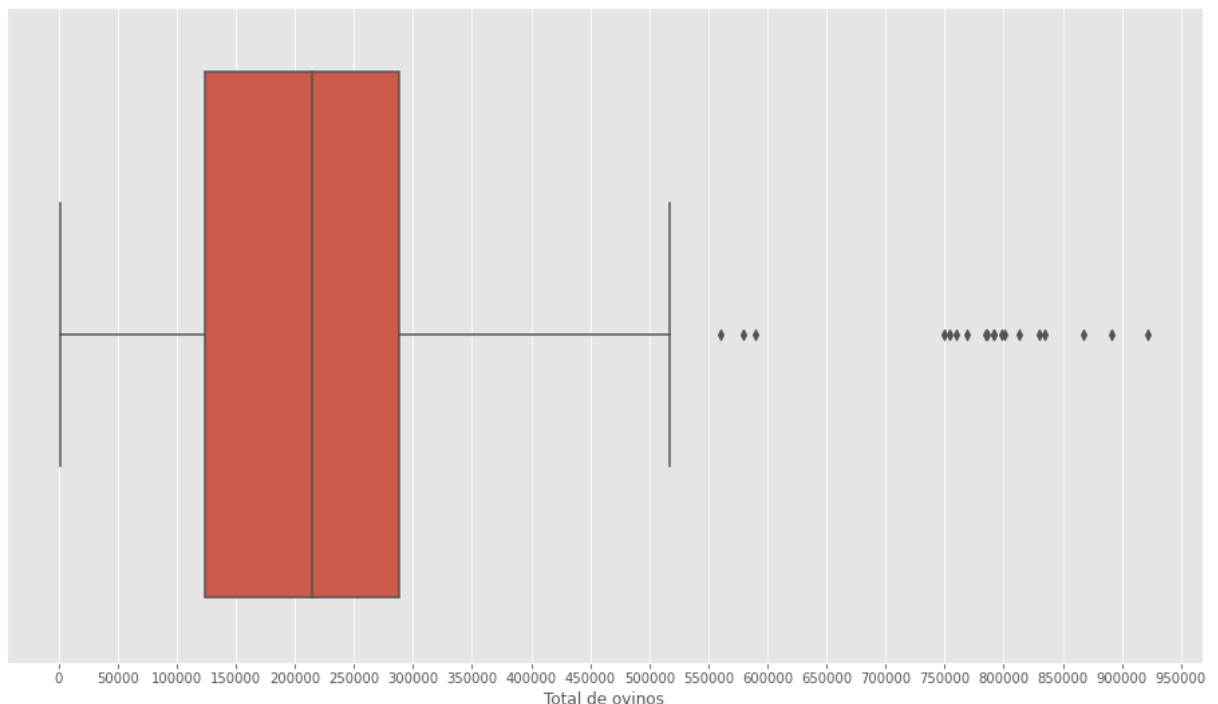
# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Total de ovinos')
```

```
# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 1771.0
Valor del bigote superior: 516771.0



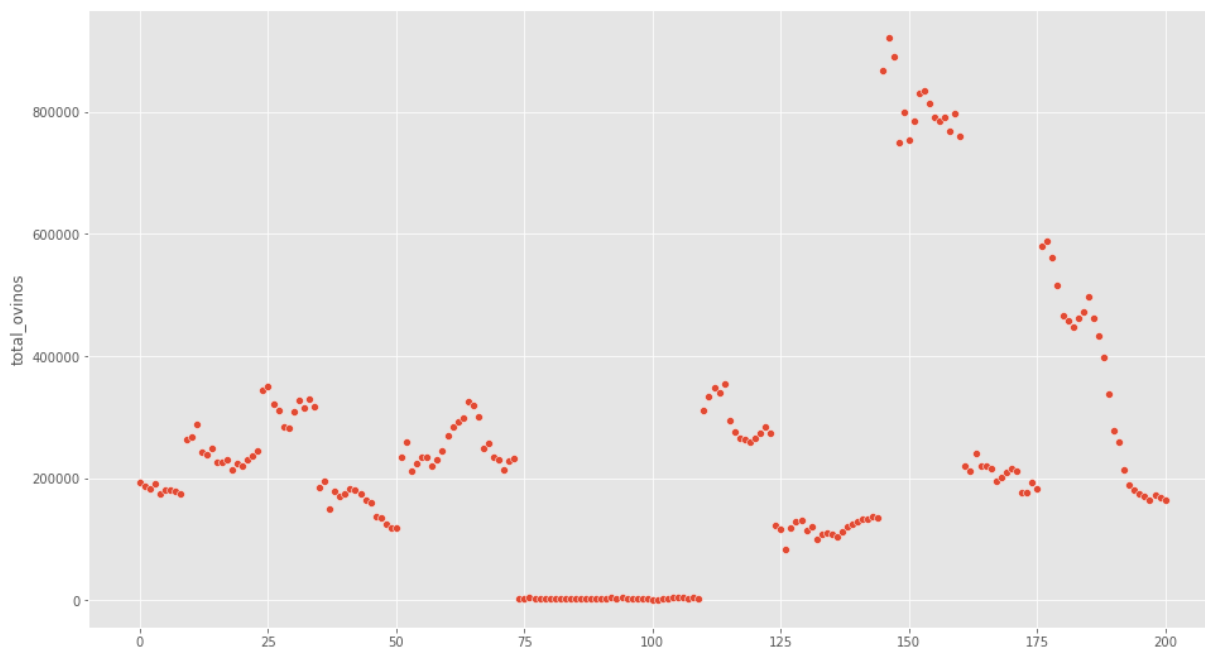
El gráfico boxplot nos proporciona una representación visual de la distribución de los datos. Los valores del primer cuartil, el segundo cuartil (o mediana) y el tercer cuartil nos indican la posición relativa de los datos en la distribución. En este caso, el primer cuartil muestra que el 25% de los valores son menores o iguales a 17144 ovinos, mientras que el tercer cuartil nos indica que el 75% de los valores son menores o iguales a 41337 ovinos.

La mediana, que es 27014 ovinos, divide la caja en dos partes desiguales. Los bigotes del gráfico indican los valores mínimo y máximo dentro de un rango aceptable, que se extiende desde el límite inferior de 1771 hasta el límite superior de 516771. Además, el gráfico muestra que hay 17 valores atípicos que superan el límite superior, lo que indica que hay algunas observaciones que están muy por encima de la mayoría de los datos y que podrían ser considerados como excepciones o datos extremos.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de ovinos** se distribuyen en un plano cartesiano.

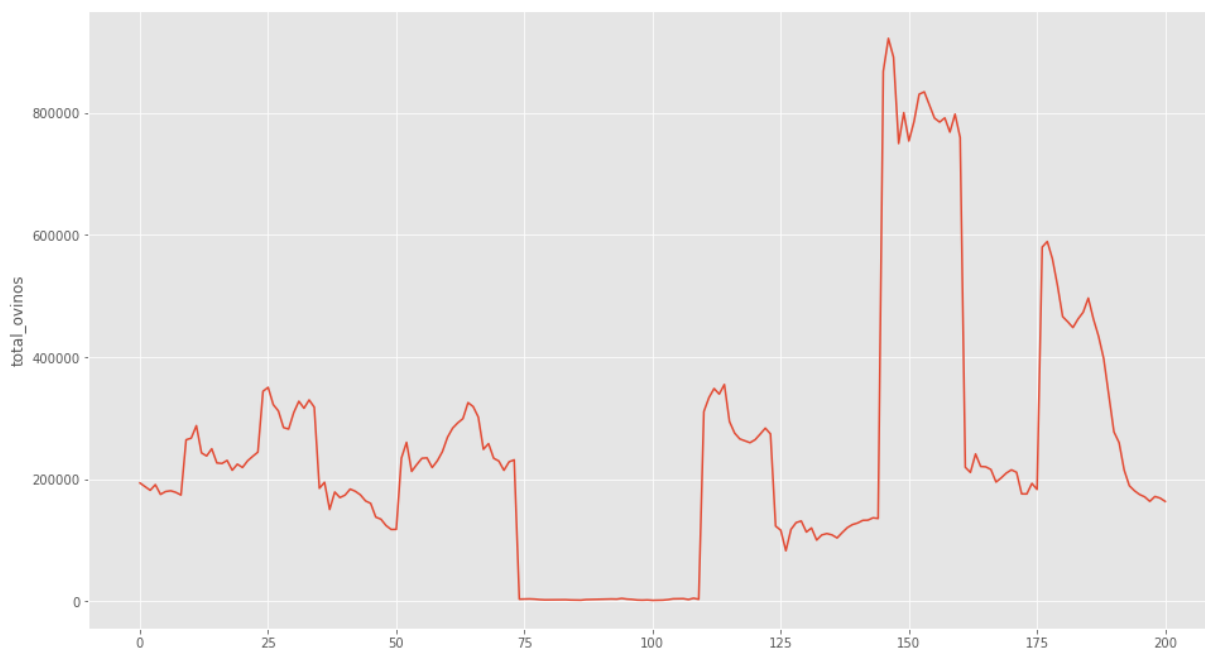
In [22]: `sns.scatterplot(data=datos['total_ovinos'])`

```
Out[22]: <AxesSubplot:ylabel='total_ovinos'>
```



```
In [23]: sns.lineplot(data=datos['total_ovinos'])
```

```
Out[23]: <AxesSubplot:ylabel='total_ovinos'>
```



Al examinar el diagrama de dispersión, no se puede distinguir ninguna tendencia o patrón definido entre las variables representadas, aunque se pueden identificar siete grupos grandes que podrían ser relevantes, no hay una disposición clara de los puntos, como una línea recta o un grupo bien definido.

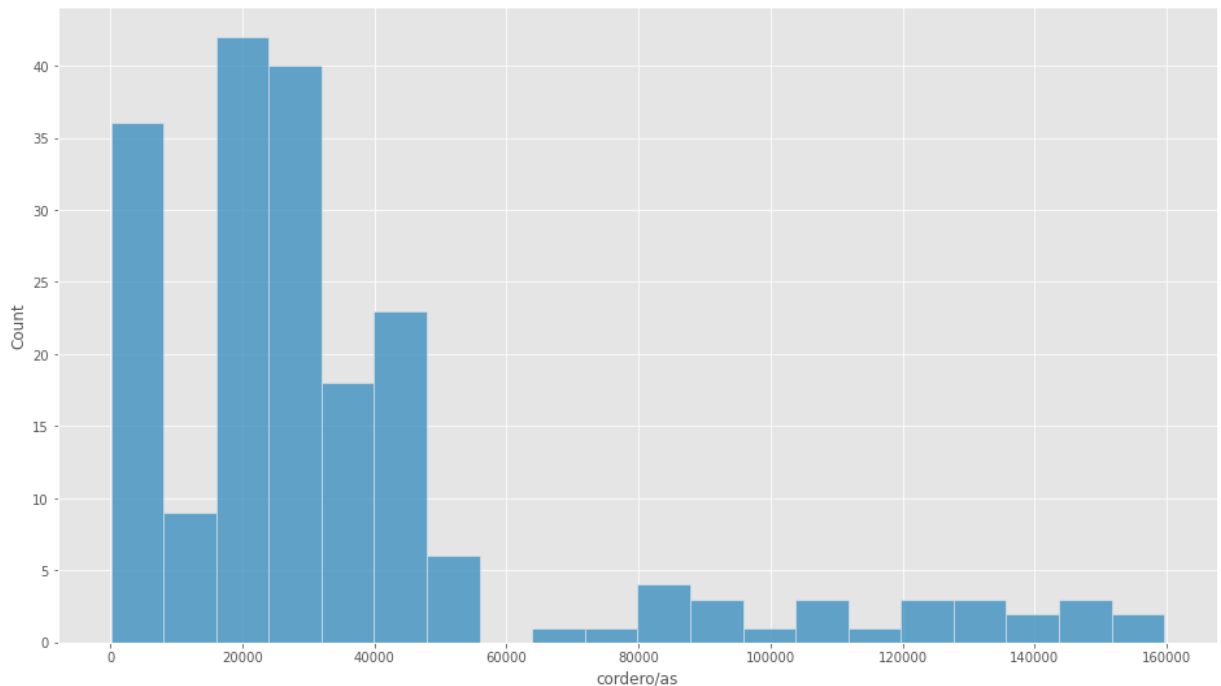
Por otro lado, al observar el gráfico de línea que representa la producción total de ovinos, se puede notar que existe una variabilidad en la producción, lo que se refleja en los picos de los datos. Esto podría indicar que la población ovina tiene fluctuaciones o una variabilidad relativamente alta. No se observa ninguna tendencia clara, por lo que no es posible afirmar que exista un aumento o disminución constante en las observaciones.

Corderos

En el siguiente histograma vemos que el `cordero/as` tiene una distribución asimétrica.

```
In [24]: sns.histplot(data=datos['cordero/as'])
```

```
Out[24]: <AxesSubplot:xlabel='cordero/as', ylabel='Count'>
```



En un histograma, una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho indica una distribución asimétrica derecha, también conocida como una distribución positivamente sesgada. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el `total de corderos` que nos indicaba el histograma.

```
In [25]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['cordero/as'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 170000, 10000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

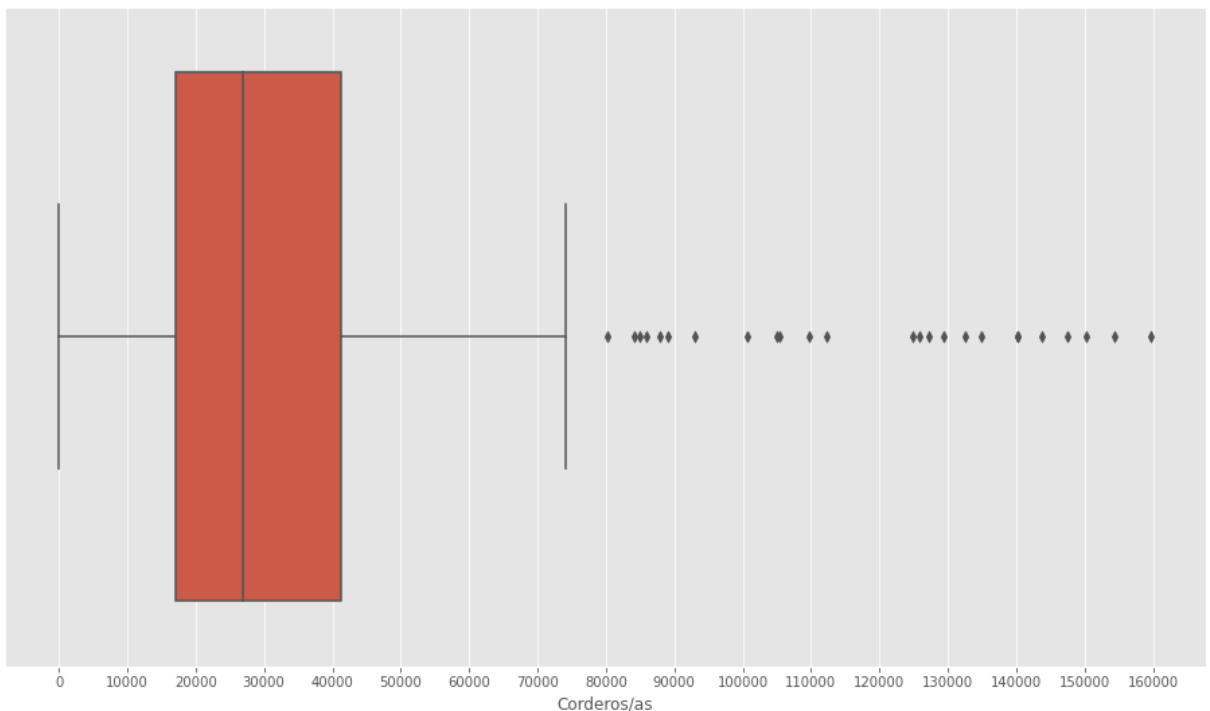
# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Corderos/as')
```

```
# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 34.0
Valor del bigote superior: 74093.0



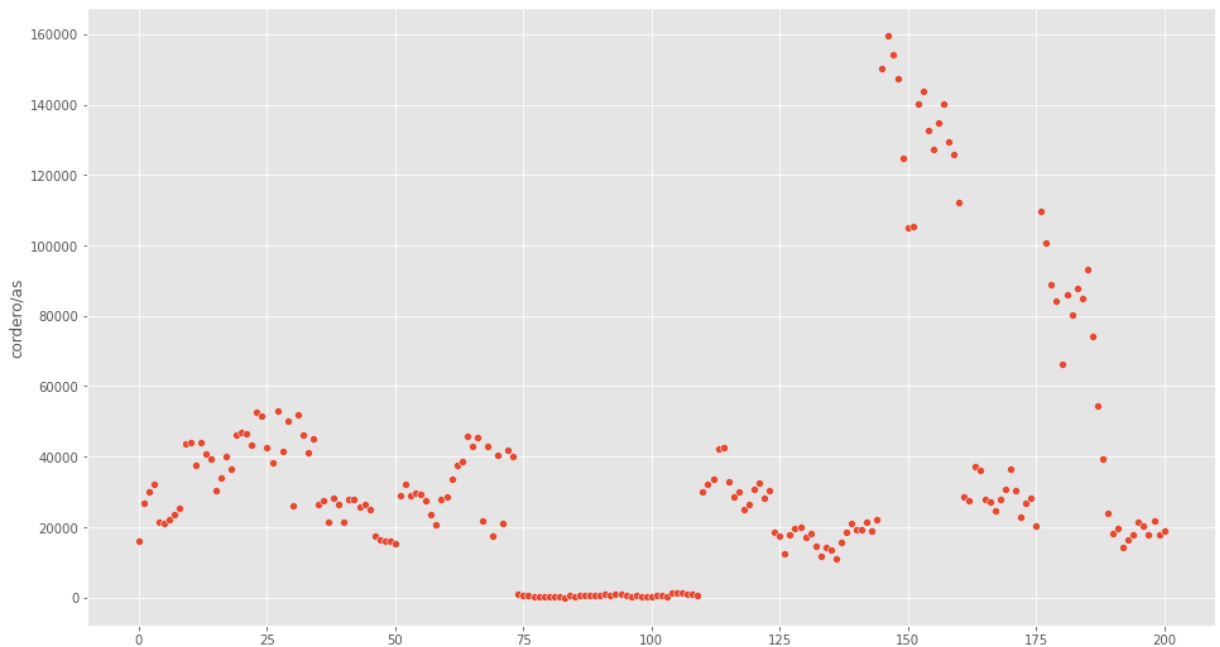
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 17144 corderos/as.
- Segundo Cuartil: el 50% de los valores son menores o igual a 27014 corderos/as.
- Tercer cuartil: el 75% de los valores son menores o igual a 41337 corderos/as.
- Mediana: la mediana es 27014 corderos/as y corta la caja en dos lados desiguales, por lo tanto muestra una asimetría positiva o segada a la derecha dado que la parte más larga de la caja es la parte superior a la mediana.
- Bigotes: el limite inferior es de 34 y el limite superior es de 74093.
- Valores atípicos: hay 24 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de corderos** se distribuyen en un plano cartesiano.

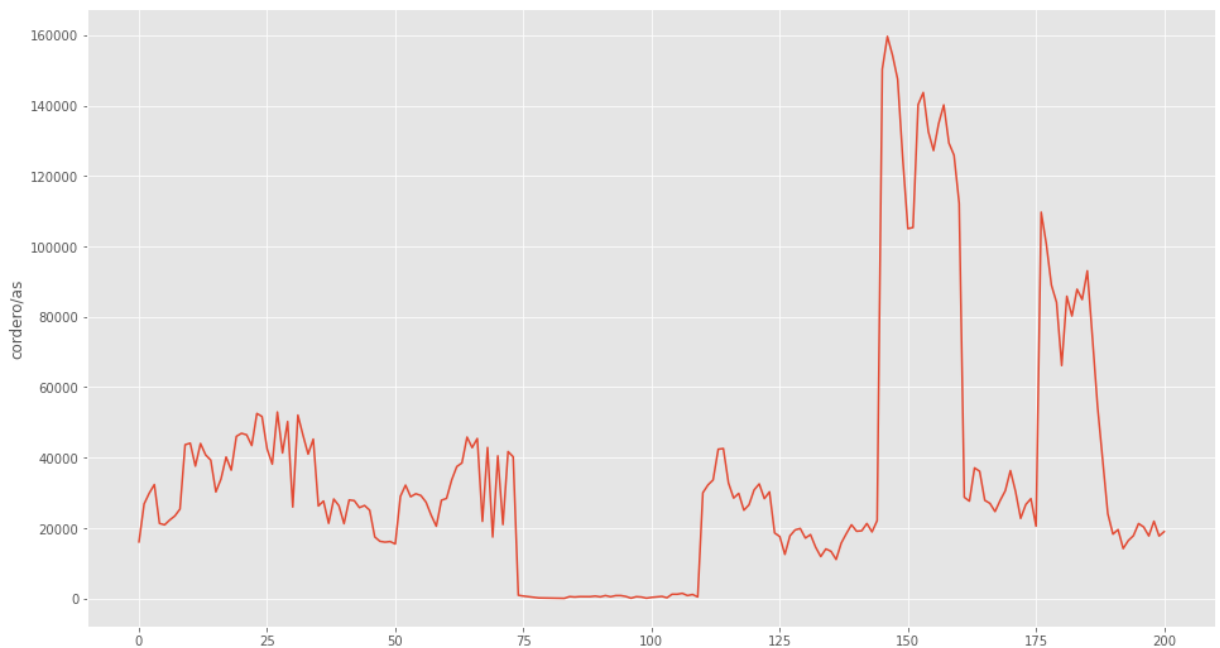
```
In [26]: sns.scatterplot(data=datos['cordero/as'])
```

```
Out[26]: <AxesSubplot:ylabel='cordero/as'>
```



In [27]: `sns.lineplot(data=datos['cordero/as'])`

Out[27]: `<AxesSubplot:ylabel='cordero/as'>`



Al analizar los datos representados en el gráfico de dispersión, no se puede identificar un patrón o tendencia clara entre las variables. Aunque se pueden distinguir seis o siete grupos chicos que podrían ser relevantes, no hay una disposición definida de los puntos, como una línea recta o agrupaciones bien definidas.

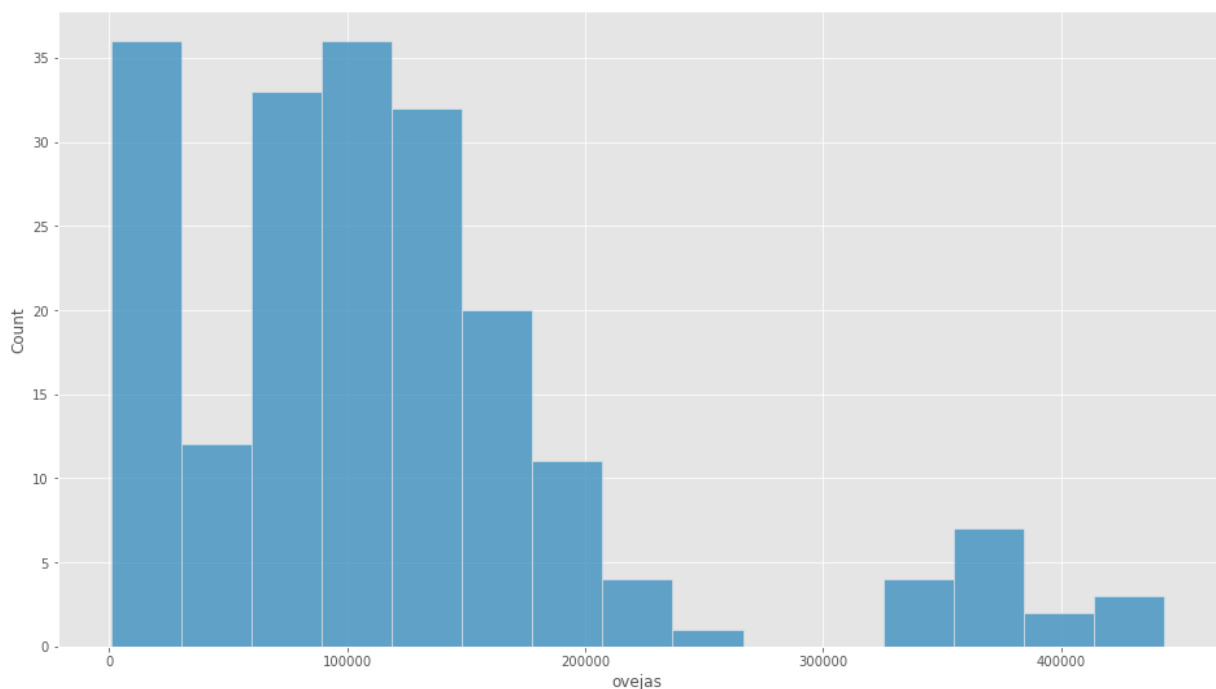
En cuanto al gráfico de línea del total de corderos/as, se puede observar una variabilidad en la producción, representada por los altos en los datos. Sin embargo, la mayoría de los datos se encuentran entre 20000 y 60000 corderos/as, lo que podría indicar una fluctuación moderada. Pero no se aprecia una tendencia clara, por lo que no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Ovejas

En el siguiente histograma vemos que la **ovejas** tienen una distribución asimétrica.


```
In [28]: sns.histplot(data=datos['ovejas'])
```

```
Out[28]: <AxesSubplot:xlabel='ovejas', ylabel='Count'>
```



El histograma muestra una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho, lo que indica una distribución asimétrica derecha. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el **total de ovejas** que nos indicaba el histograma.

```
In [29]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['ovejas'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 460000, 50000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

# Eliminar el label del eje Y
ax.set_yticks([])

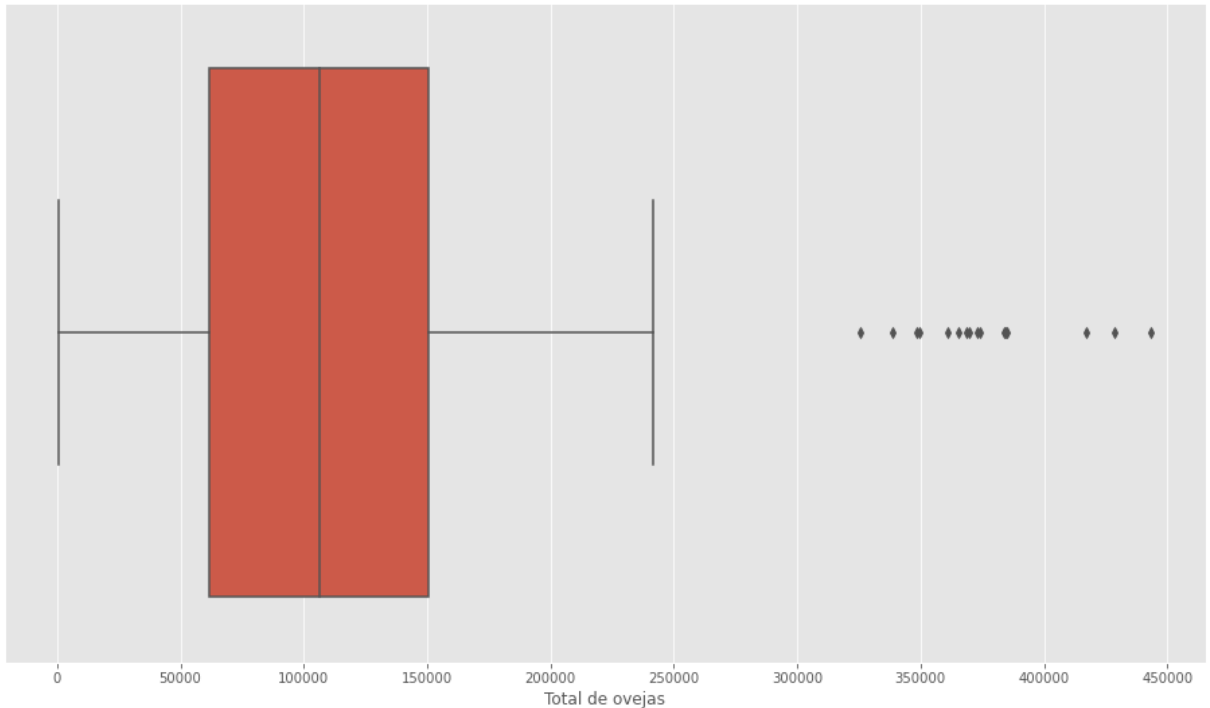
# Agregar un label al eje X
ax.set_xlabel('Total de ovejas')

# Obtener valores de los bigotes
```

```
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 768.0
Valor del bigote superior: 241352.0



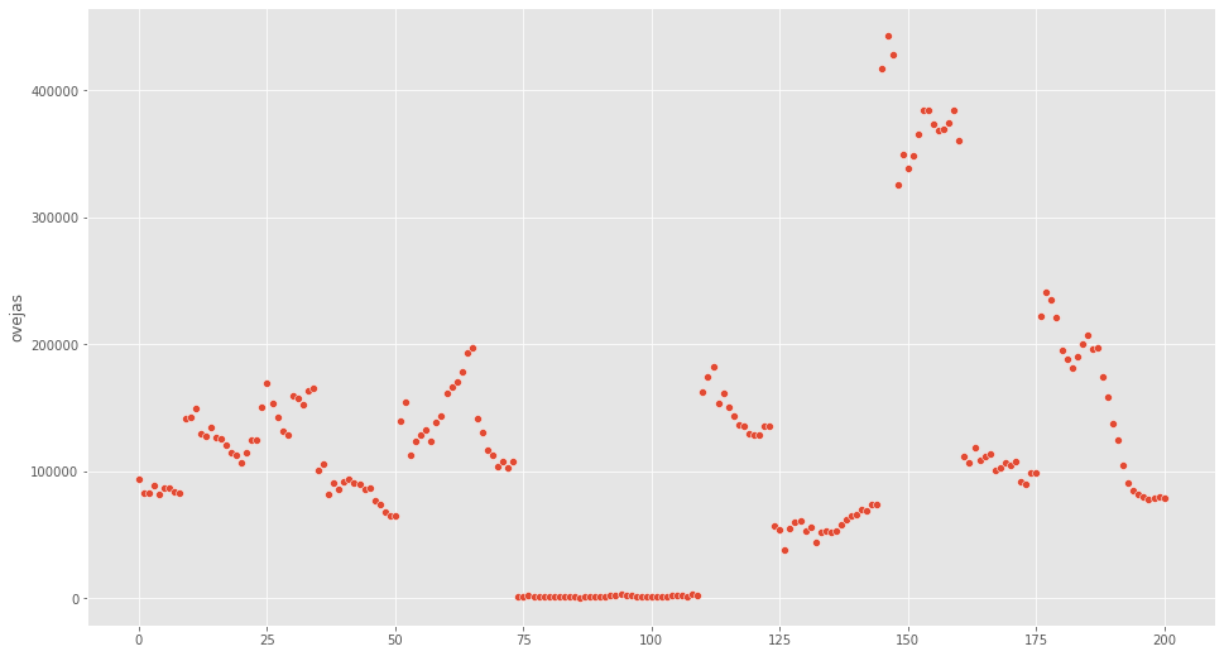
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 61699 ovejas.
- Segundo Cuartil: el 50% de los valores son menores o igual a 106471 ovejas.
- Tercer cuartil: el 75% de los valores son menores o igual a 150454 ovejas.
- Mediana: la mediana es 106471 de ovejas y corta la caja en dos lados casi iguales.
- Bigotes: el limite inferior es de 768 y el limite superior es de 241352.
- Valores atípicos: hay 15 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de ovejas** se distribuyen en un plano cartesiano.

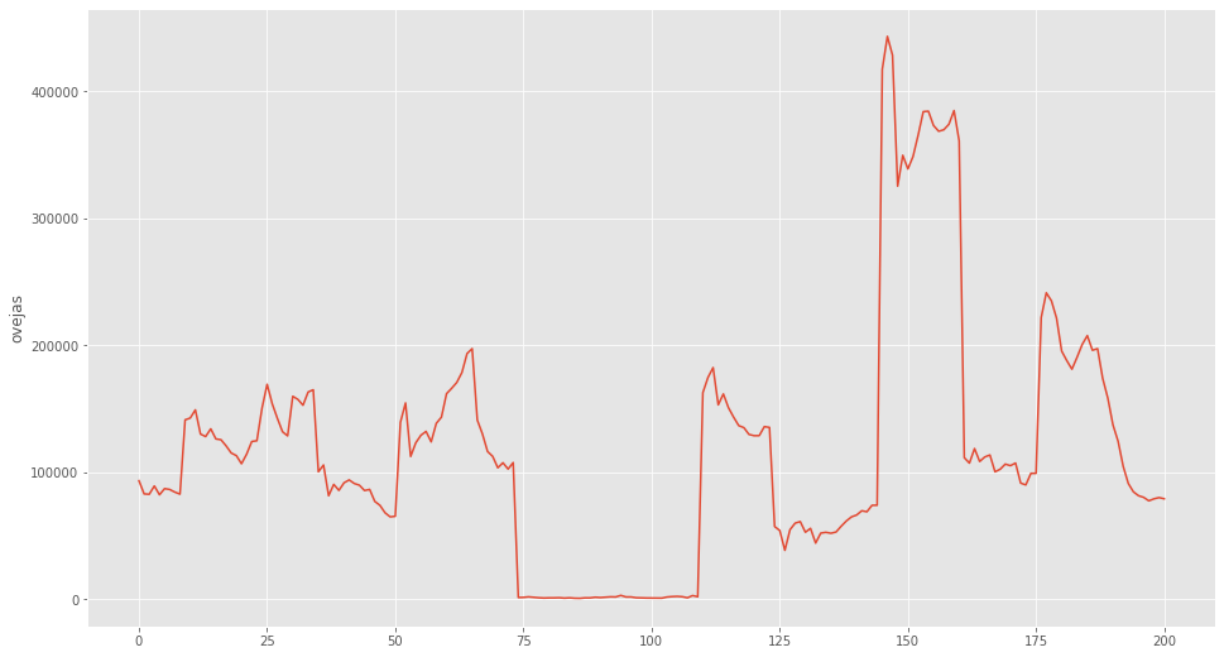
In [30]: `sns.scatterplot(data=datos['ovejas'])`

Out[30]: `<AxesSubplot:ylabel='ovejas'>`



```
In [31]: sns.lineplot(data=datos['ovejas'])
```

```
Out[31]: <AxesSubplot:ylabel='ovejas'>
```



Al analizar los datos representados en el gráfico de dispersión, no se puede identificar un patrón o tendencia clara entre las variables, aunque se pueden identificar alrededor de 4 o 5 grupos grandes en los datos.

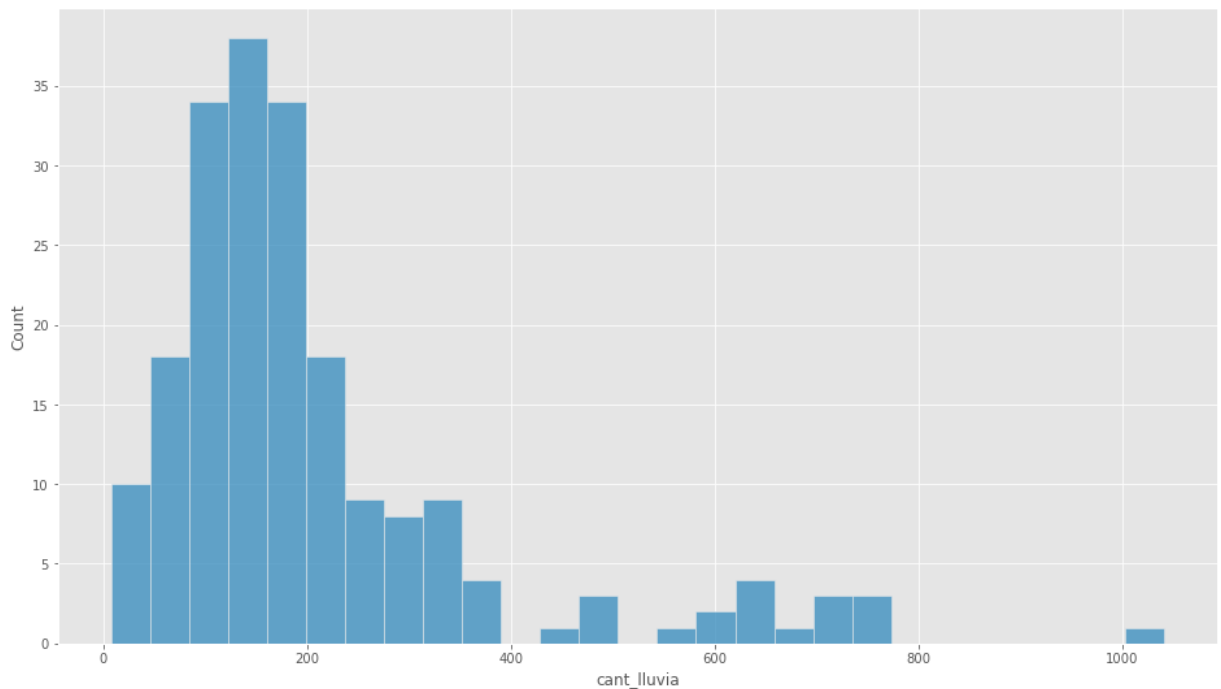
Por otro lado, al observar el gráfico de línea del total de ovejas, se puede notar que la mayoría de las observaciones se encuentran entre 50000 y 200000 ovejas. Sin embargo, no se aprecia una tendencia clara, por lo que no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Lluvia

En el siguiente histograma vemos que la `cant_lluvia` tiene una distribución asimétrica.

```
In [32]: sns.histplot(data=datos['cant_lluvia'])
```

Out[32]: <AxesSubplot:xlabel='cant_lluvia', ylabel='Count'>



Se ve una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho lo que indica una distribución asimétrica derecha. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el **total de mm lluvia** que nos indicaba el histograma.

```
In [33]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['cant_lluvia'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 1100, 100)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

# Eliminar el label del eje Y
ax.set_yticks([])

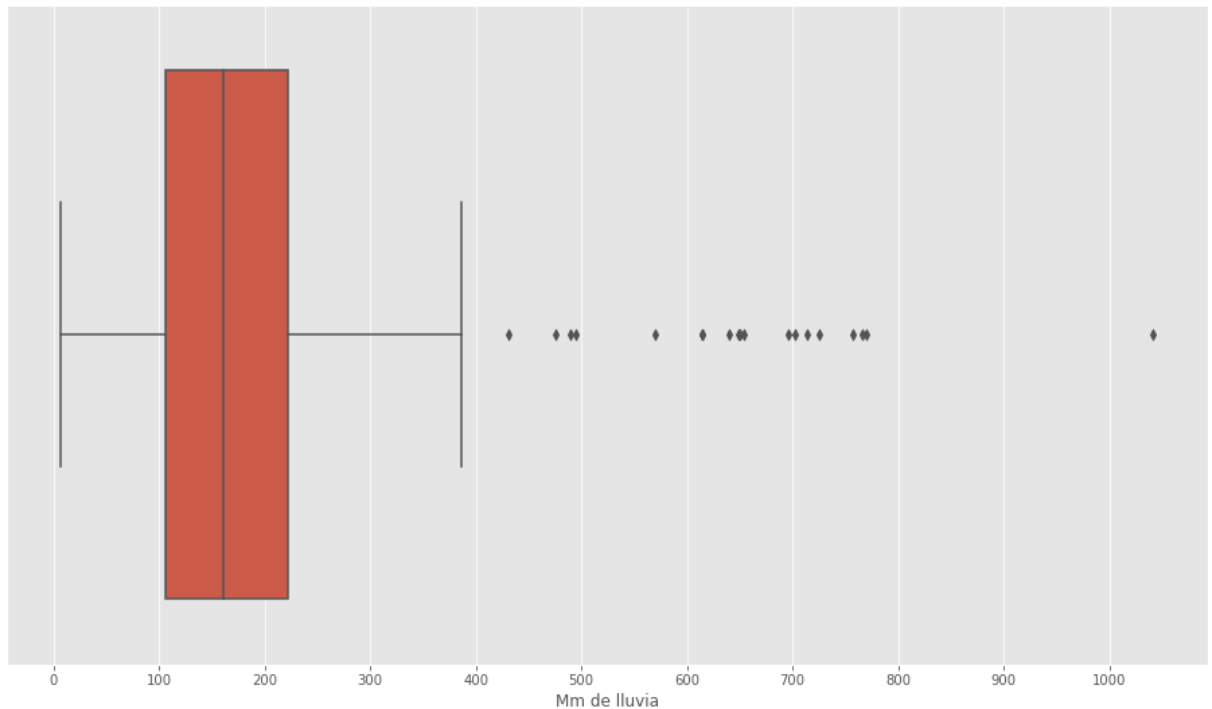
# Agregar un label al eje X
ax.set_xlabel('Mm de lluvia')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
```

```
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 7.1
 Valor del bigote superior: 386.47



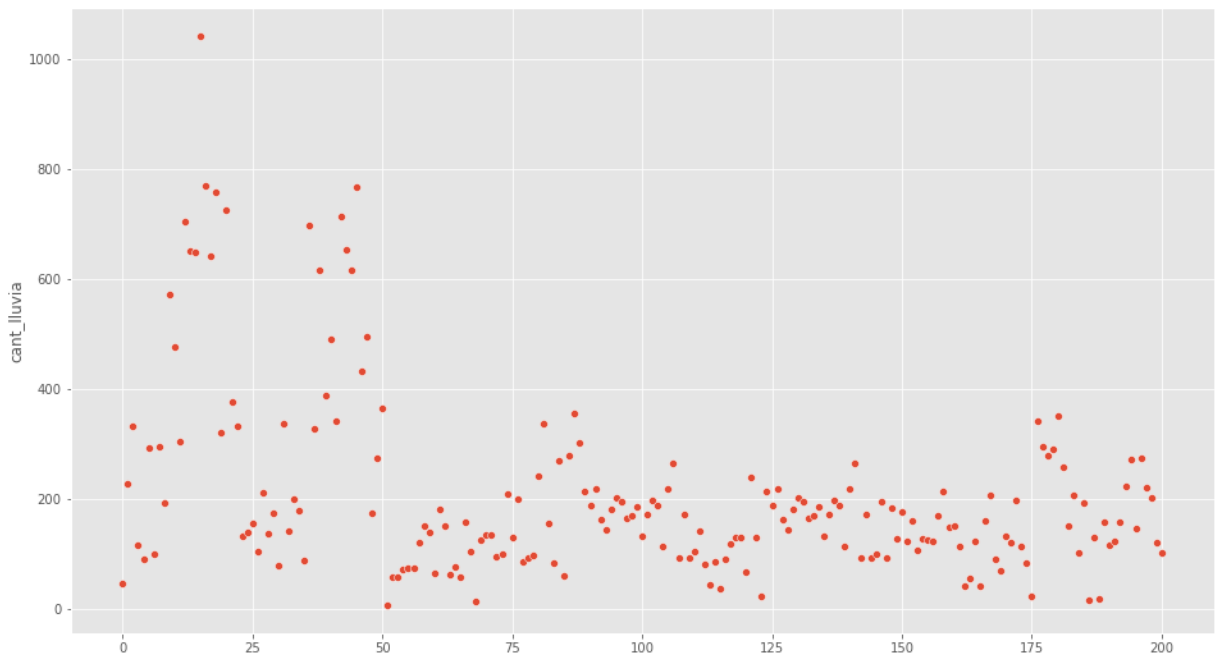
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 105.77 mm de lluvia.
- Segundo Cuartil: el 50% de los valores son menores o igual a 161.00 mm de lluvia.
- Tercer cuartil: el 75% de los valores son menores o igual a 222.05 mm de lluvia.
- Mediana: la mediana es 161.00 mm de lluvia y corta la caja en dos lados desiguales, por lo tanto muestra una asimetría positiva o segada a la derecha dado que la parte más larga de la caja es la parte superior a la mediana.
- Bigotes: el limite inferior es de 7.1 mm y el limite superior es de 386.47 mm.
- Valores atípicos: hay 17 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el total de lluvia se distribuyen en un plano cartesiano.

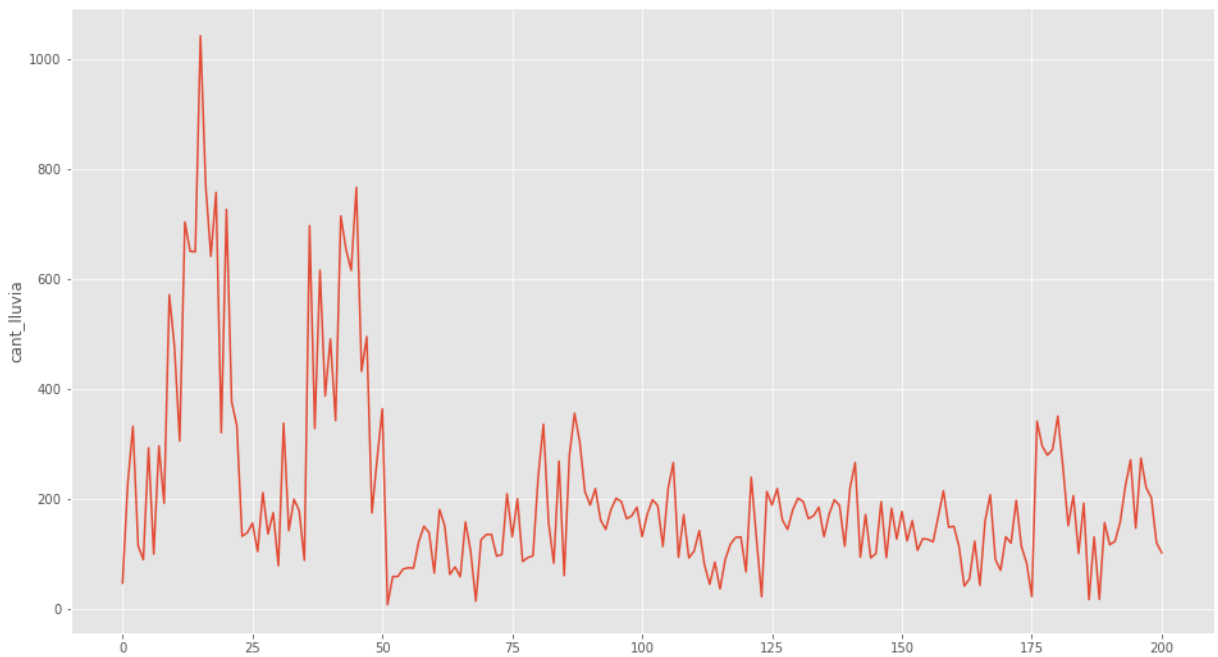
```
In [34]: sns.scatterplot(data=datos['cant_lluvia'])
```

```
Out[34]: <AxesSubplot:ylabel='cant_lluvia'>
```



```
In [35]: sns.lineplot(data=datos['cant_lluvia'])
```

```
Out[35]: <AxesSubplot:ylabel='cant_lluvia'>
```



Tras analizar los gráficos de dispersión y de línea, no se puede identificar un patrón o tendencia clara en los datos de lluvia en Chubut. En el gráfico de dispersión, no se observa ningún grupo o agrupación definida de los puntos. Por su parte, en el gráfico de línea se aprecian varios picos altos, lo que indica que existe una variabilidad significativa en los valores de lluvia. Sin embargo, se puede establecer que la precipitación en Chubut varía en un rango de 0 a 400 mm, tal como se refleja en ambos gráficos.

Como observación final de las variables independientes, se puede notar que en los gráficos de dispersión y línea, los datos de lana y existencias muestran una similitud casi idéntica en su forma. También se puede observar una caída abrupta en los valores productivos al incorporar los datos de El Mirabueno al conjunto de datos de los departamentos. Esto se debe a que en los departamentos, los valores de producción suelen superar el millón o cientos de miles, mientras que en El Mirabueno, al ser un campo, los valores son de miles, y no de millones de kilos de lana o cientos de miles de ovejas. No se realizó una estandarización

para que todos los valores estuvieran en el mismo rango, es decir no se escalaron los datos, para conocer el conjunto de datos en su estado original.

Variables combinadas

A continuación se va a mejorar la paleta de colores para que sea más fácil distinguir entre ellos. Para lograrlo, crearemos una paleta de colores personalizada que constará de 15 colores profundos ("deep") y la estableceremos como la paleta por defecto para los gráficos de Seaborn.

```
In [5]: mi_paleta = sns.color_palette("deep", 15)
sns.set_palette(mi_paleta)
```

A continuación, se crearán dos dataframes más pequeños divididos por los departamentos que pertenecen a la cordillera y los que no, con el fin de facilitar el proceso de gráficos conjuntos y posiblemente obtener información adicional a partir de la división de los datos.

```
In [6]: departamentos_cordillera =
datos[datos['departamento'].isin(['CUSHAMEN', 'RIO SENGUERR',
'TEHUELCHES', 'LANGUIÑEO', 'FUTALEUFU'])]
departamentos_meseta =
datos[~datos['departamento'].isin(['CUSHAMEN', 'RIO SENGUERR',
'TEHUELCHES', 'LANGUIÑEO', 'FUTALEUFU'])]
```

Gráficos

Diagrama de dispersión

El comando `sns.scatterplot(data=datos, x='variable_X', y='variable_Y', hue='departamento')` va a generar un gráfico de dispersión donde se muestra la relación entre dos variables: la `variable_X` en el eje X y la `variable_Y` en el eje Y. Además, se utiliza el argumento `hue` para agregar una tercera dimensión de información, que en este caso es el departamento. Cada punto en el gráfico representa una observación del dataset, donde la posición en el eje X corresponde al valor de `'variable_X'` de la observación, la posición en el eje Y corresponde al valor de `'variable_Y'` en esa observación y el color corresponde al departamento.

Diagrama de línea

El código `sns.lineplot(data=datos, x='variable_X', y='variable_Y')` genera un gráfico de línea que representa la evolución de la `'variable_X'` a lo largo de `'variable_Y'`. Cada punto en el gráfico representa la `'variable_X'` registrada para un valor específico de `'variable_Y'`, la línea que los une representa la tendencia general de la variable a lo largo de `'variable_Y'`, es decir, la evolución de `'variable_X'` a medida que cambia `'variable_Y'`, y el borde de color más claro indica el intervalo de confianza. Esto significa que el gráfico muestra un valor promedio de la variable `'variable_X'` para cada valor de `'variable_Y'`, junto con un intervalo de confianza alrededor de ese valor promedio. El intervalo de confianza indica la incertidumbre en la estimación del valor promedio y se calcula comúnmente como la desviación estándar de los datos dividido por la raíz cuadrada del número de observaciones.

Como el intervalo de confianza se calcula en función del número de observaciones que hay en los datos, en cada uno de los gráficos que vamos a ver a continuación vemos que comienzan en el año 2000 y el intervalo de confianza se muestra a partir del 2006, esto es porque hay pocas observaciones en los años anteriores a 2006, lo que hace que el intervalo de confianza no sea significativo o no pueda ser calculado con precisión.

Diagrama de barras

El comando `sns.barplot(data=datos, x='fecha', y='variable_Y')` genera un gráfico de barras que muestra la cantidad total de 'variable_Y' para cada año en el conjunto de datos. Cada año se mostraría en el eje x y la cantidad total 'variable_Y' correspondiente se mostraría en el eje y. La altura de cada barra representaría la cantidad la 'variable_Y' para una fecha determinada. La línea finita que se encuentra en la parte superior de cada barra representa el intervalo de confianza del 95% para la media de cada grupo de datos. Es decir, es una estimación de la incertidumbre en la medida de la media para cada grupo. La altura de la línea muestra el rango de valores dentro del cual se espera que la media real esté con una confianza del 95%.

Fecha con otras variables

Existencias ovinas

Con el fin de manejar adecuadamente la información de las fechas, se procedió a ordenar el conjunto de datos según este atributo y se guardó en un nuevo DataFrame. Posteriormente, se agregó una nueva columna que contiene únicamente el año, lo que permitirá trabajar de manera más eficiente con esta información en caso de ser necesario.

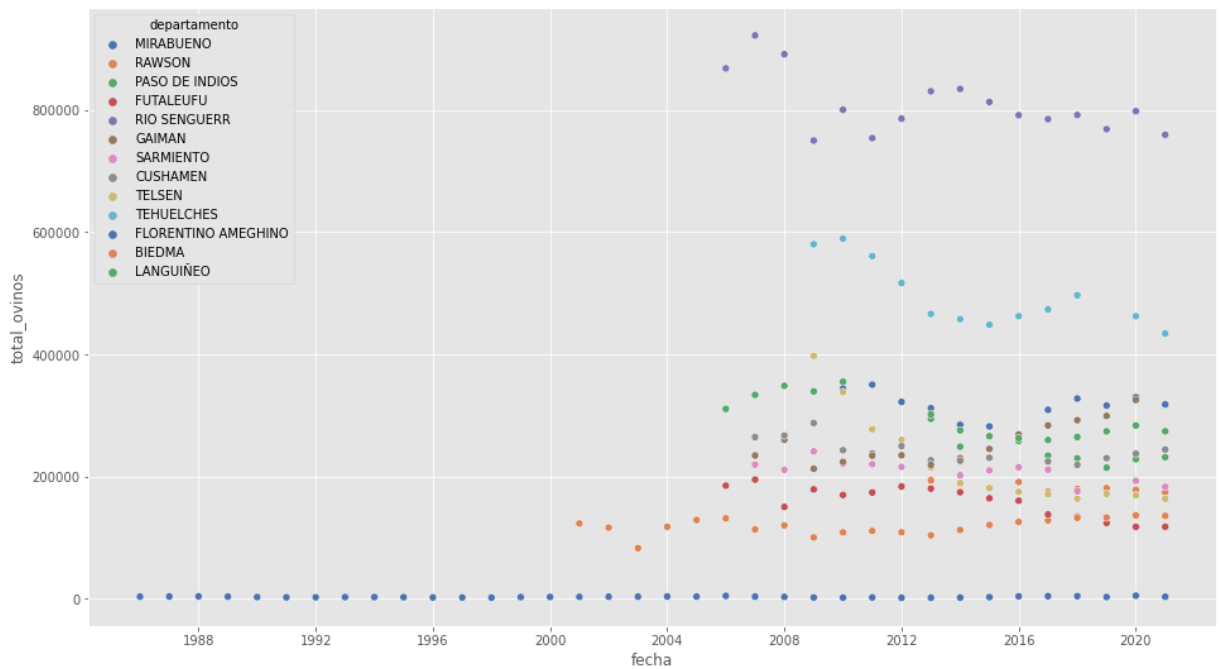
```
In [37]: datos_ordenados_fecha = datos.sort_values('fecha')
datos_ordenados_fecha['anio'] =
datos_ordenados_fecha['fecha'].dt.strftime('%Y')
```

Total de ovinos

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo `total_ovinos` para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de ovinos a lo largo del tiempo y cómo se distribuyen en los diferentes años.

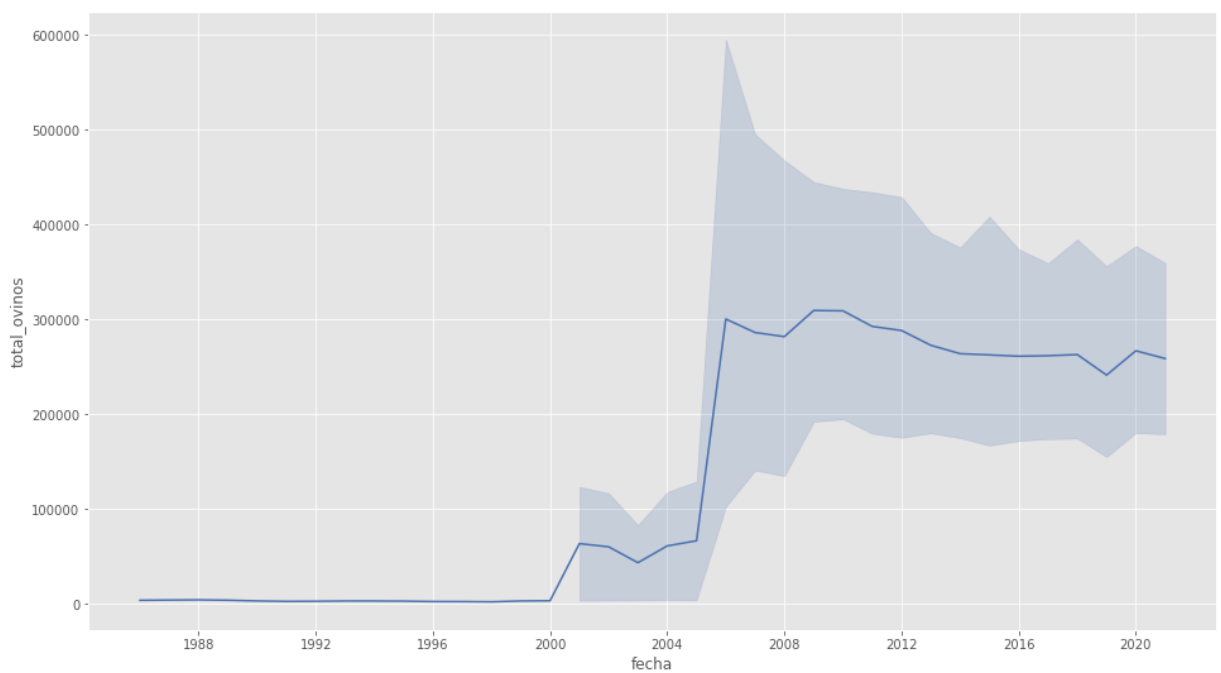
```
In [38]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',
y='total_ovinos', hue='departamento')
```

```
Out[38]: <AxesSubplot:xlabel='fecha', ylabel='total_ovinos'>
```

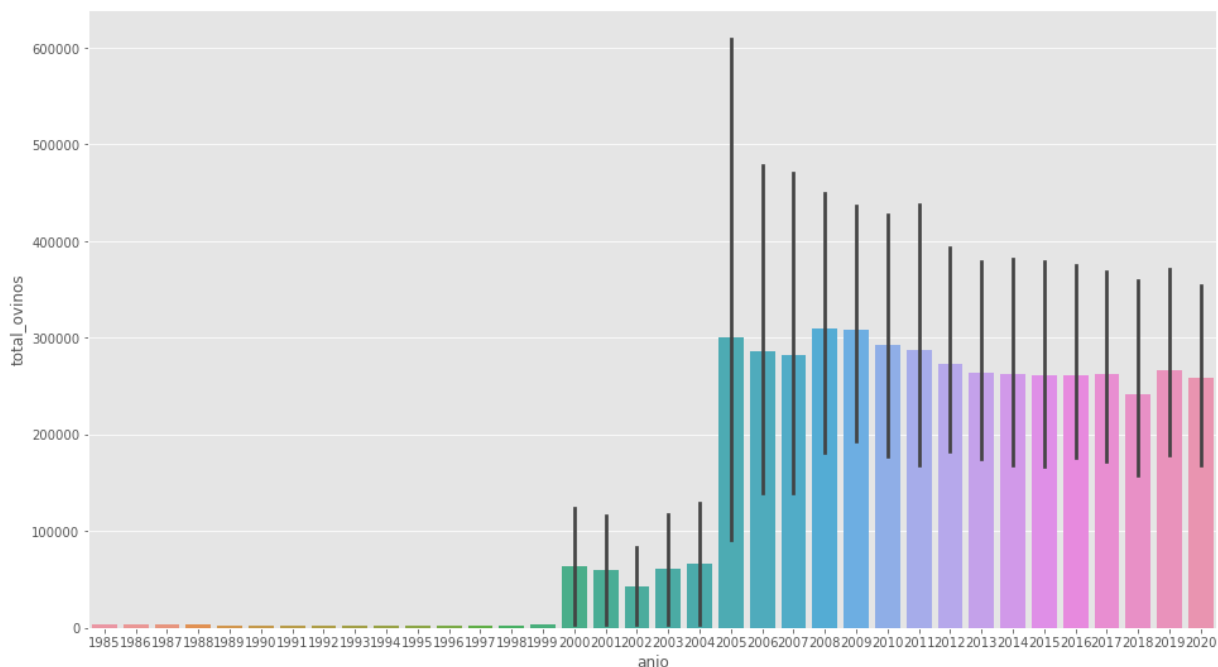
```
In [39]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',
y='total_ovinos')
```

```
Out[39]: <AxesSubplot:xlabel='fecha', ylabel='total_ovinos'>
```



```
In [40]: sns.barplot(data=datos_ordenados_fecha, x='anio',
y='total_ovinos')
```

```
Out[40]: <AxesSubplot:xlabel='anio', ylabel='total_ovinos'>
```



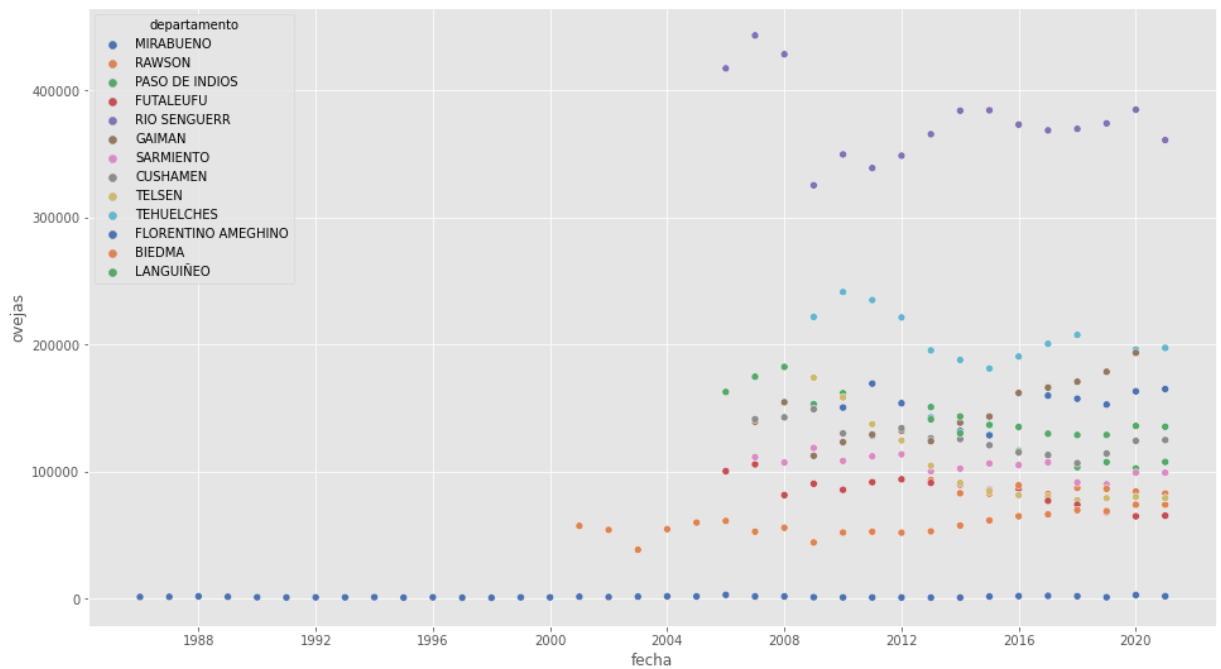
El análisis de estos tres gráficos proporciona información sobre la variación del número total de ovinos a lo largo del tiempo. En el gráfico de dispersión, cada punto representa un departamento en un plano cartesiano, donde su posición en los ejes X e Y representa la cantidad de ovinos en un año específico. Los puntos se colorean según el departamento, lo que facilita la comparación visual entre departamentos. Se observa que la mayoría de los departamentos mantienen una cantidad de ovinos relativamente constante, que no supera los 400 mil, excepto Tehuelches y Río Senger, que presentan valores más altos y variables. Cuando el número de ovinos supera los 400 mil, los puntos se dispersan mucho, lo que indica una gran variabilidad en la producción de ovinos a lo largo de los años. En cambio, cuando el número de ovinos es inferior a 400 mil, las filas de puntos son más uniformes, lo que sugiere una producción más estable a lo largo del tiempo. Este patrón también se refleja en el gráfico de línea y en el de barras, que incluyen intervalos de confianza para mostrar la variabilidad de los datos. Además, dado que El Mirabueno tiene valores muchos más pequeños da la sensación que el total de ovinos es cero, pero en realidad sabemos que no.

Ovejas

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo `ovejas` para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de ovejas a lo largo del tiempo y cómo se distribuyen en los diferentes años.

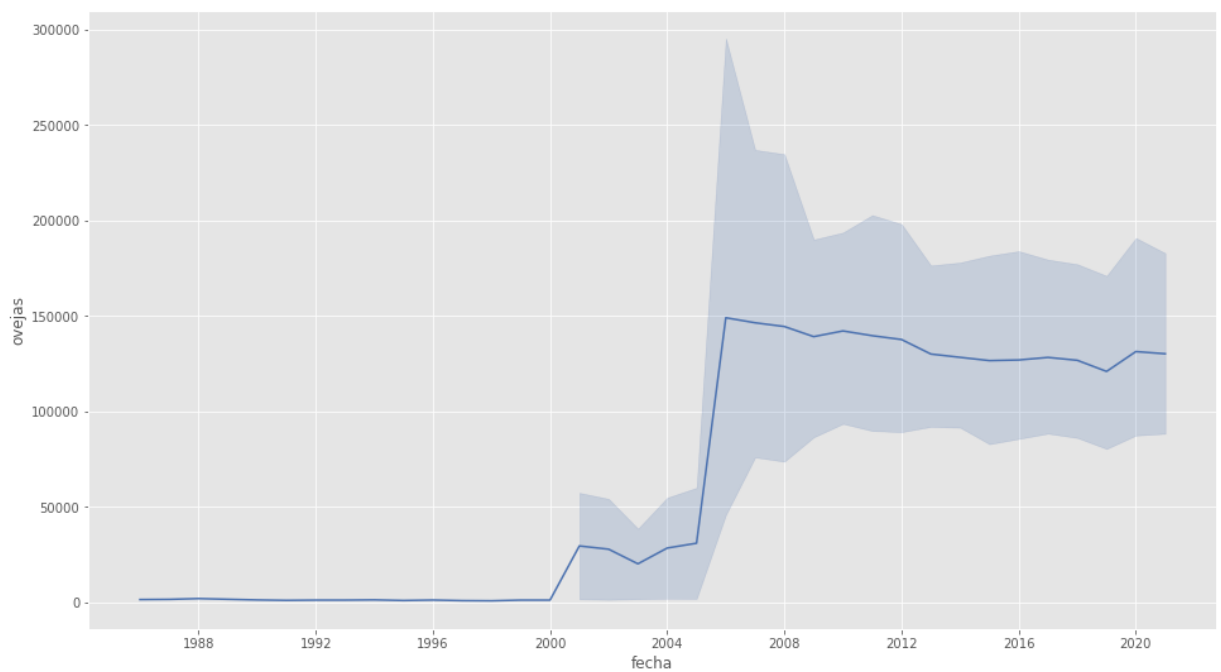
```
In [41]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='ovejas',
hue='departamento')
```

```
Out[41]: <AxesSubplot:xlabel='fecha', ylabel='ovejas'>
```



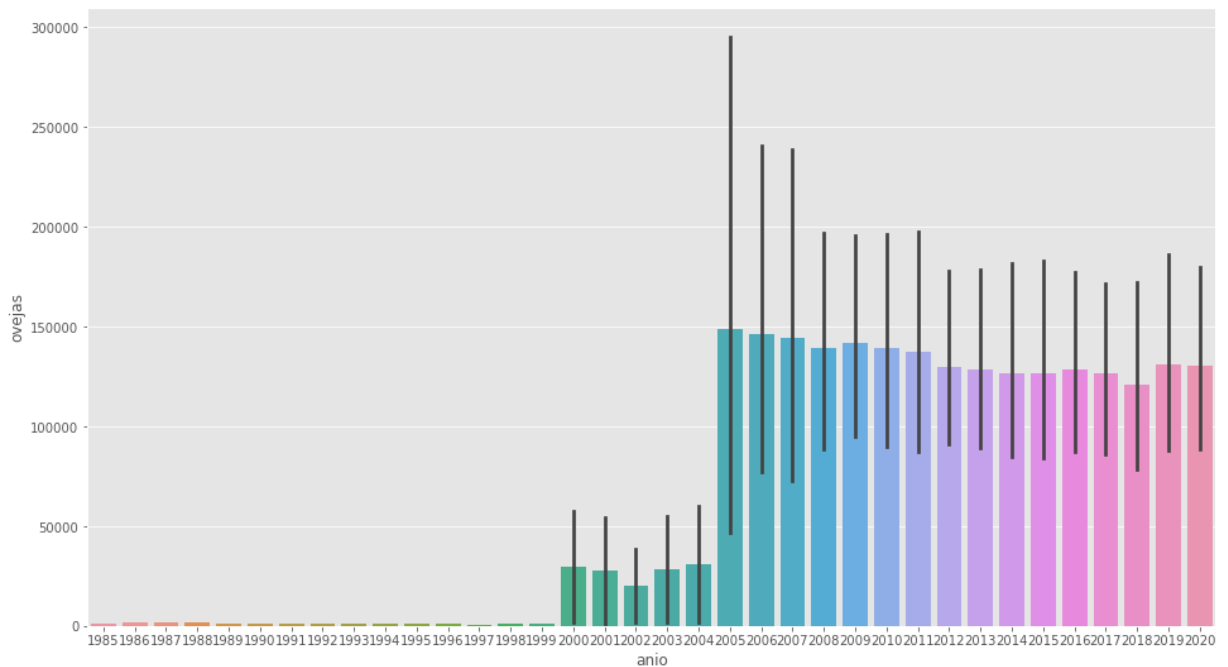
In [42]: `sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='ovejas')`

Out[42]: `<AxesSubplot:xlabel='fecha', ylabel='ovejas'>`



In [43]: `sns.barplot(data=datos_ordenados_fecha, x='anio', y='ovejas')`

Out[43]: `<AxesSubplot:xlabel='anio', ylabel='ovejas'>`



Los tres gráficos proporcionan datos acerca de la evolución del total de ovejas en diferentes departamentos a lo largo del tiempo. El gráfico de dispersión muestra cada departamento como un punto en un plano cartesiano, donde la posición de los puntos en los ejes X e Y representa la cantidad de ovejas en un año específico. El color de cada punto indica el departamento al que pertenece, lo que nos permite comparar visualmente la cantidad de ovejas entre departamentos.

Al analizar el gráfico de dispersión, se puede apreciar que el departamento de Gaiman experimentó una disminución en la cantidad de ovejas en el año 2009, pero a partir de 2010 logró aumentar progresivamente la cantidad de ovejas hasta superar los 190 mil en el año 2020. Además, se puede observar que los departamentos de Tehuelches y Río Senger presentan valores que superan los 190 mil, y muestran una mayor variabilidad en la producción de ovejas en comparación con el resto de los departamentos que están por debajo de este valor.

Cuando la cantidad de ovejas supera los 190 mil, se observa una dispersión de los puntos en el gráfico de dispersión, lo que sugiere una mayor variabilidad en la producción de ovejas a lo largo de los años. Por otro lado, cuando la cantidad de ovejas es inferior a 190 mil, se observa una mayor uniformidad en las filas de puntos, lo que indica una producción más estable a lo largo del tiempo. Este patrón se puede confirmar en los gráficos de línea y de barras, los cuales incluyen intervalos de confianza para mostrar la variabilidad de los datos.

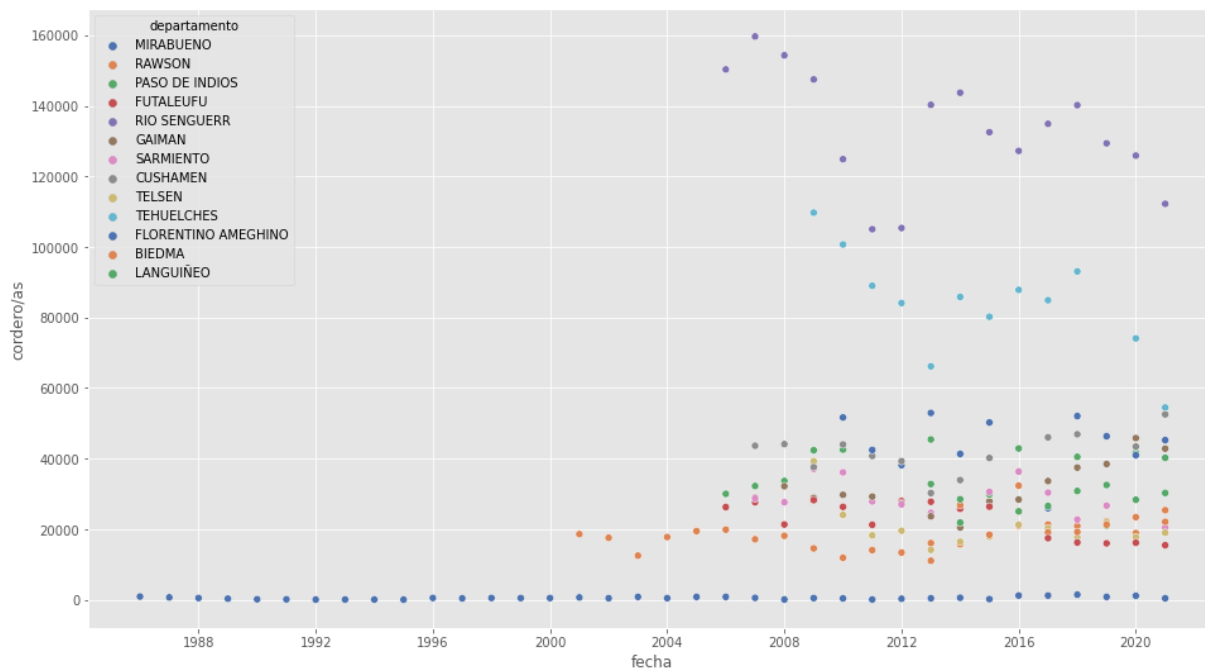
Además, dado que El Mirabueno tiene valores muchos más pequeños da la sensación que el total de ovejas es cero, pero en realidad sabemos que no.

Corderos

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo `corderos` para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de corderos a lo largo del tiempo y cómo se distribuyen en los diferentes años.

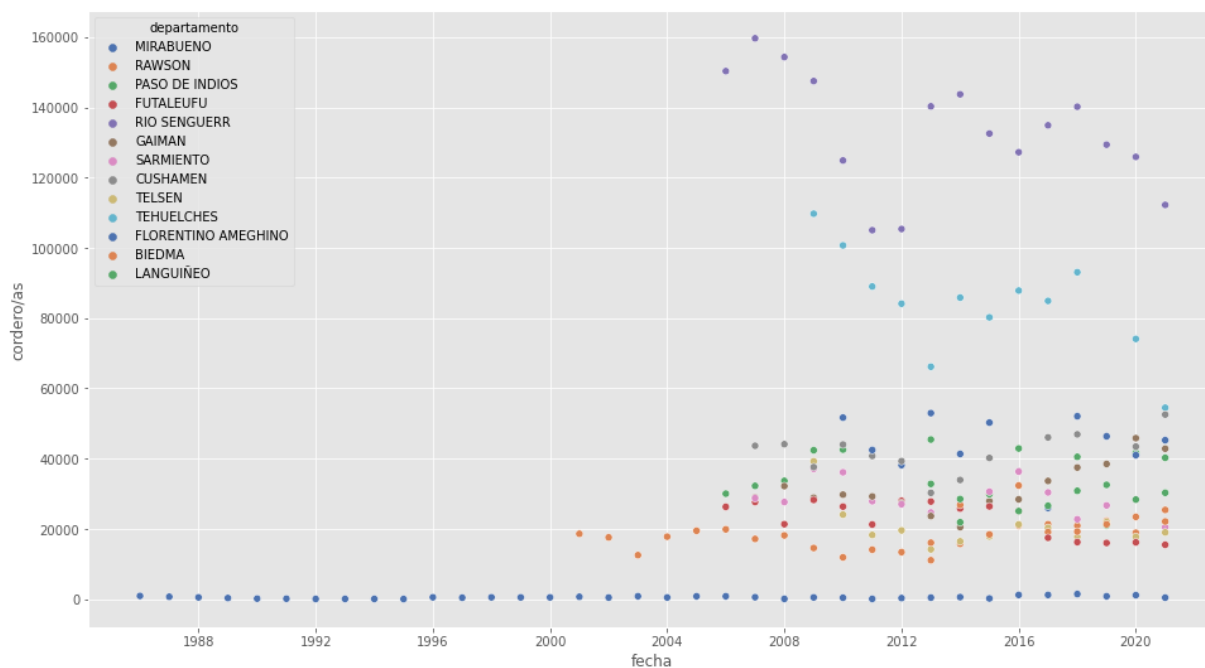
```
In [44]: # Crear el scatterplot con la nueva paleta de colores
sns.scatterplot(data=datos_ordenados_fecha, x='fecha',
y='cordero/as', hue='departamento')
```

Out[44]: <AxesSubplot: xlabel='fecha', ylabel='cordero/as'>



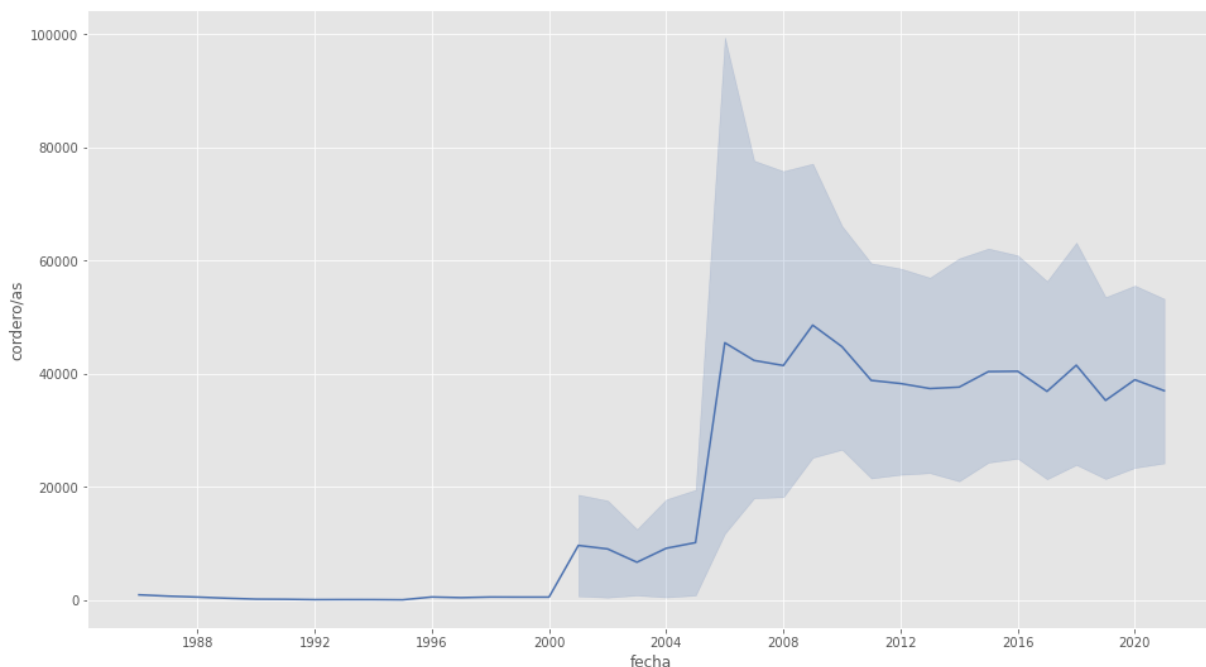
```
In [45]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',  
                        y='cordero/as', hue='departamento')
```

Out[45]: <AxesSubplot: xlabel='fecha', ylabel='cordero/as'>



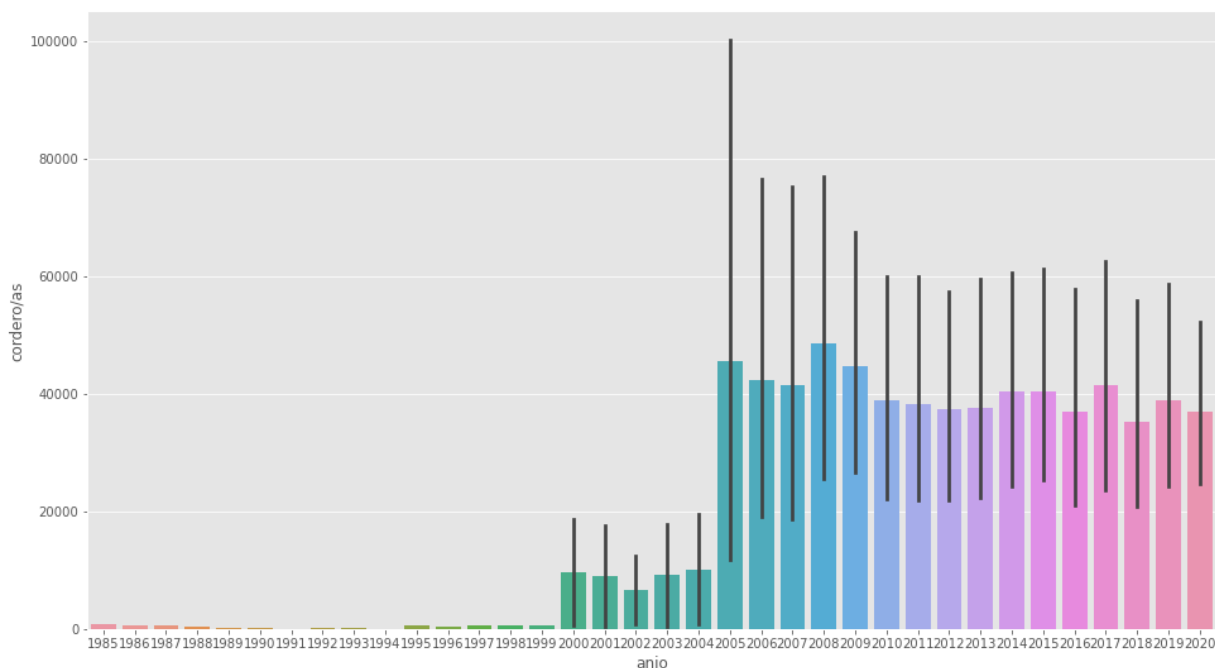
```
In [46]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',  
                      y='cordero/as')
```

Out[46]: <AxesSubplot: xlabel='fecha', ylabel='cordero/as'>



In [47]: `sns.barplot(data=datos_ordenados_fecha, x='anio', y='cordero/as')`

Out[47]: `<AxesSubplot:xlabel='anio', ylabel='cordero/as'>`



Se pueden obtener datos acerca de la variación en la producción de corderos a lo largo del tiempo mediante estos tres gráficos. El gráfico de dispersión muestra cada departamento representado por un punto en un plano cartesiano, donde la posición en los ejes X e Y indica la cantidad de corderos en un año determinado. Los puntos se codifican por colores según el departamento al que pertenecen, lo que permite comparar la cantidad de corderos entre departamentos. Se puede observar que los departamentos de Tehuelches y Río Senger presentan valores que superan los 60 mil, y los datos varían ampliamente. Además, los departamentos de Florentino Ameghino y Cushamen presentan un rango de corderos entre 44 mil y 60 mil, con una variabilidad moderada en comparación con otros departamentos que tienen valores inferiores. Cuando el total de corderos supera los 60 mil, los puntos están muy dispersos, lo que sugiere una mayor variabilidad en la producción a lo largo de los años. En cambio, cuando los totales de corderos son inferiores a 44 mil, las filas de puntos son más uniformes, lo que indica una producción más estable

en el transcurso de los años. Este patrón se refleja también en los gráficos de línea y barra, los cuales incluyen intervalos de confianza para mostrar la variabilidad de los datos. Además, dado que El Mirabueno tiene valores muchos más pequeños da la sensación que el total de ovejas es cero, pero en realidad sabemos que no.

Producción de lana

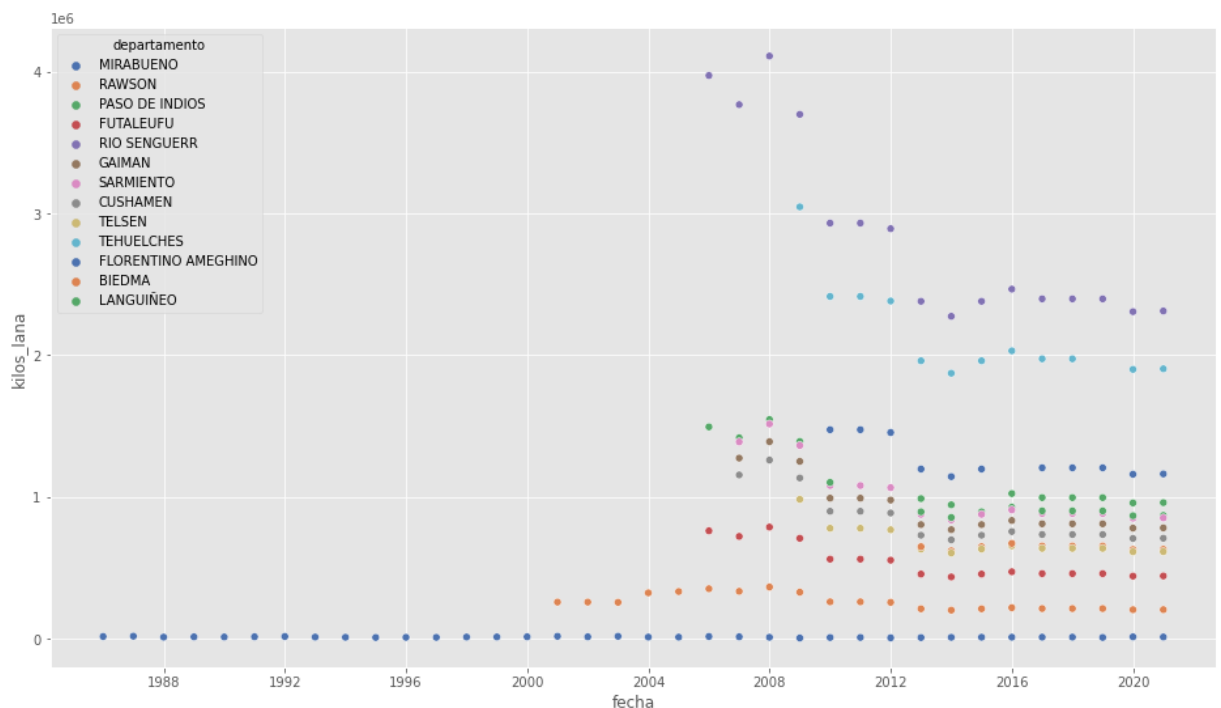
El objetivo primordial en la cría de ovejas para obtener lana es producir lanas de calidad cruzada fina y mediana, con un diámetro de entre 25 y 32 micras y un rendimiento de lavado superior al 75%. En cuanto a la situación en la provincia de Chubut, se puede observar que la producción de lana es en su mayoría de buena calidad, con un diámetro promedio en el rango deseado y un rendimiento de lavado adecuado, según lo indican los gráficos presentados a continuación.

Kilos de lana

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo `kilos de lana` para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

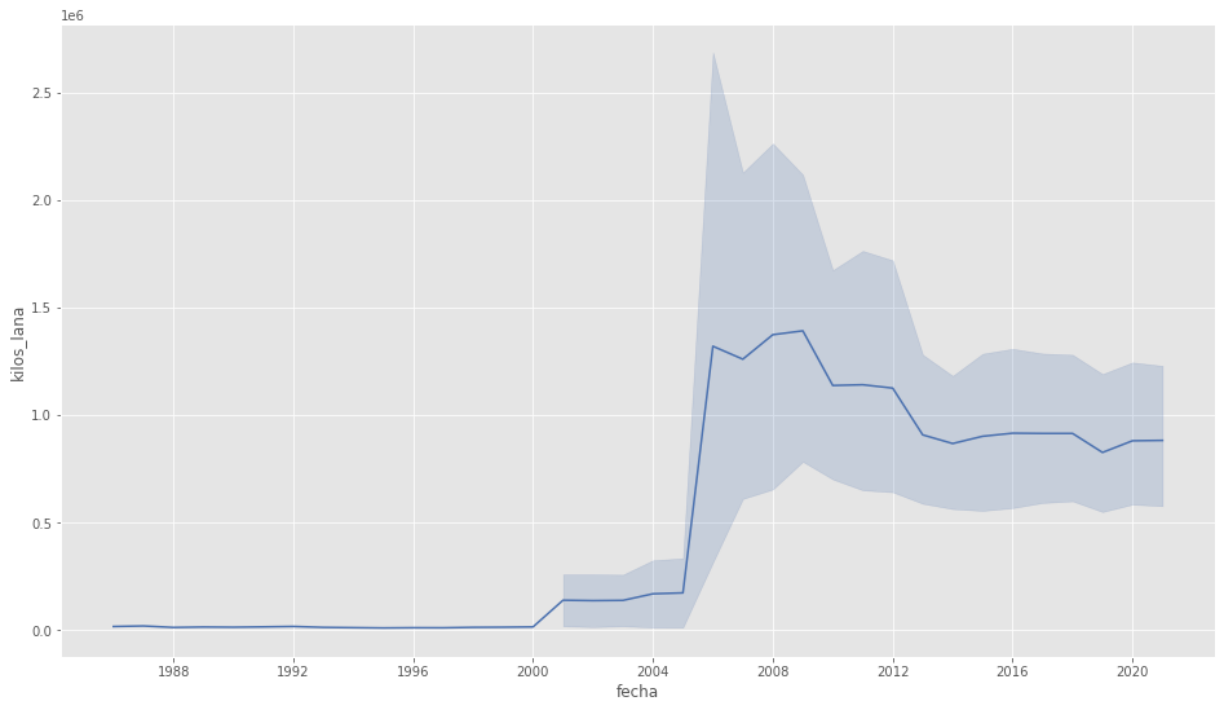
```
In [48]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',  
                        y='kilos_lana', hue='departamento')
```

```
Out[48]: <AxesSubplot:xlabel='fecha', ylabel='kilos_lana'>
```



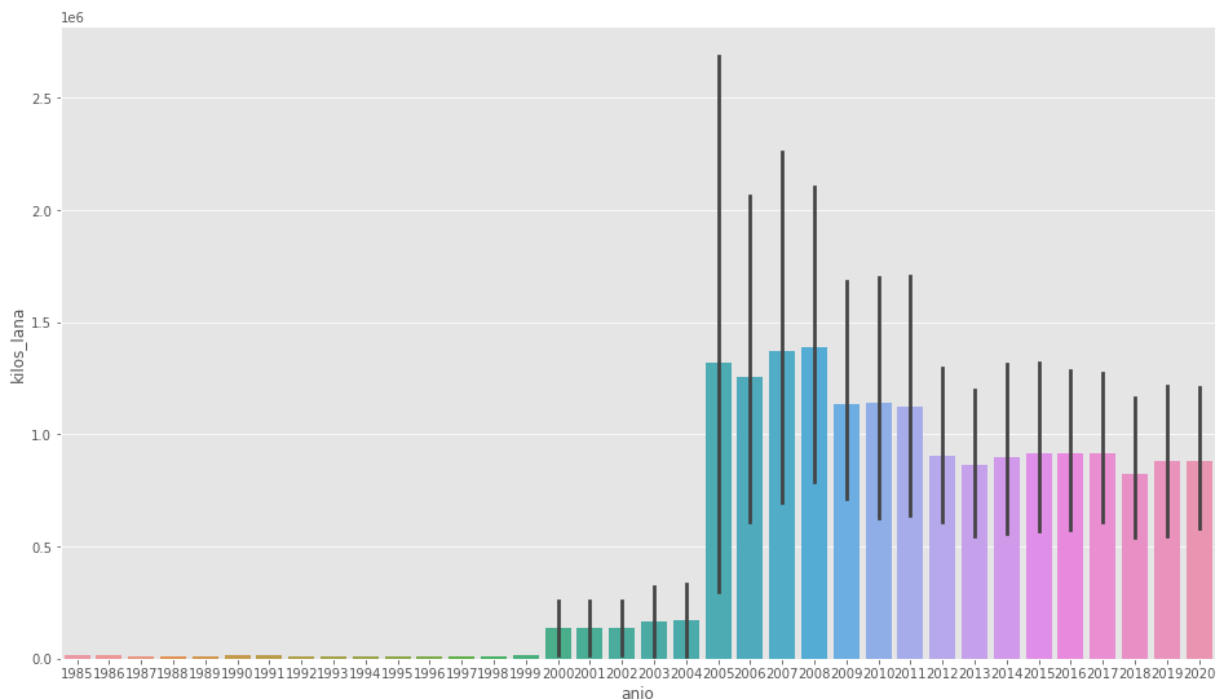
```
In [49]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',  
                      y='kilos_lana')
```

```
Out[49]: <AxesSubplot:xlabel='fecha', ylabel='kilos_lana'>
```



```
In [50]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='kilos_lana')
```

```
Out[50]: <AxesSubplot:xlabel='anio', ylabel='kilos_lana'>
```



Para obtener información sobre la variación en la producción de lana a lo largo del tiempo, se pueden examinar estos tres gráficos. El scatterplot muestra cada departamento representado por un punto en un plano cartesiano, con su ubicación en los ejes X e Y indicando la cantidad de lana en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que permite comparar visualmente la cantidad de lana entre departamentos. Se puede observar que los departamentos de Tehuelches y Río Senger presentan valores que superan la tonelada y media de lana, y son más variables en comparación con otros departamentos que presentan valores inferiores.

Entonces cuando el total de kilos de lana supera los 1500000, los puntos están muy dispersos, lo que sugiere una mayor variabilidad en la producción a lo largo de los años. En cambio, cuando la cantidad de lana es inferior a 1.5 toneladas, las filas de puntos son más uniformes, lo que indica una producción más

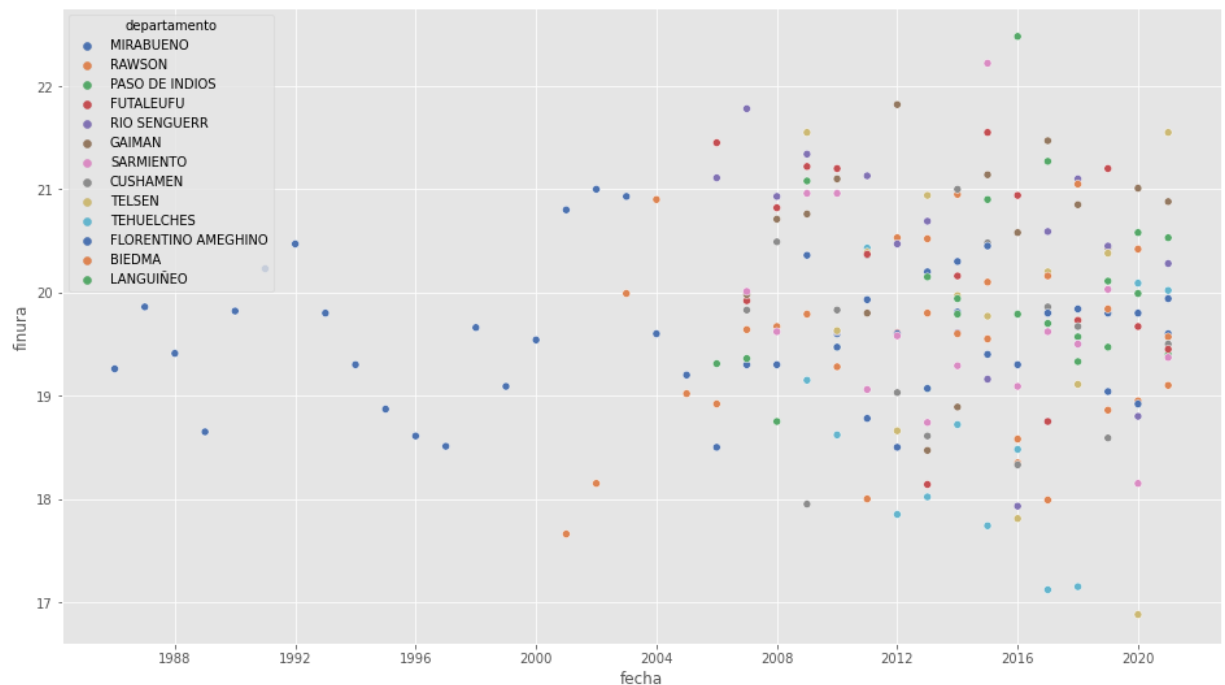
estable en el transcurso de los años. Este patrón también se refleja en los gráficos de línea y barra, los cuales incluyen intervalos de confianza para mostrar la variabilidad de los datos. Además, dado que El Mirabueno tiene valores muchos más pequeños da la sensación que el total de ovejas es cero, pero en realidad sabemos que no.

Finura

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **finura de la lana** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la finura de la lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

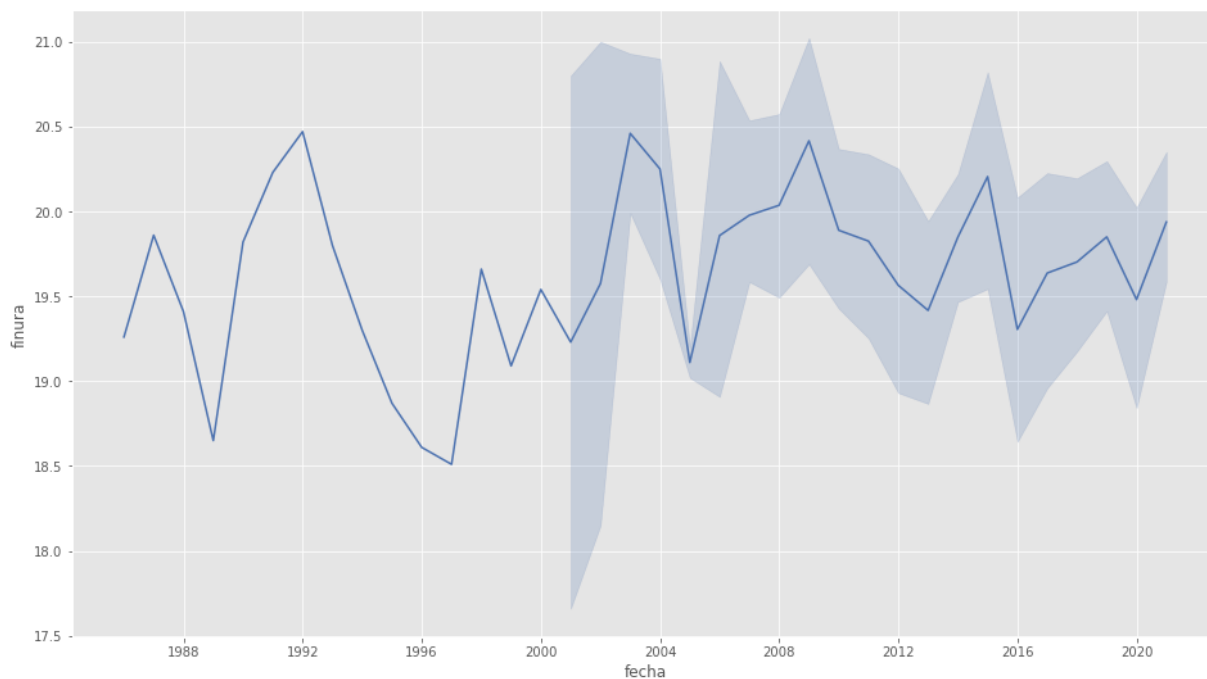
```
In [51]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='finura',  
hue='departamento')
```

```
Out[51]: <AxesSubplot:xlabel='fecha', ylabel='finura'>
```



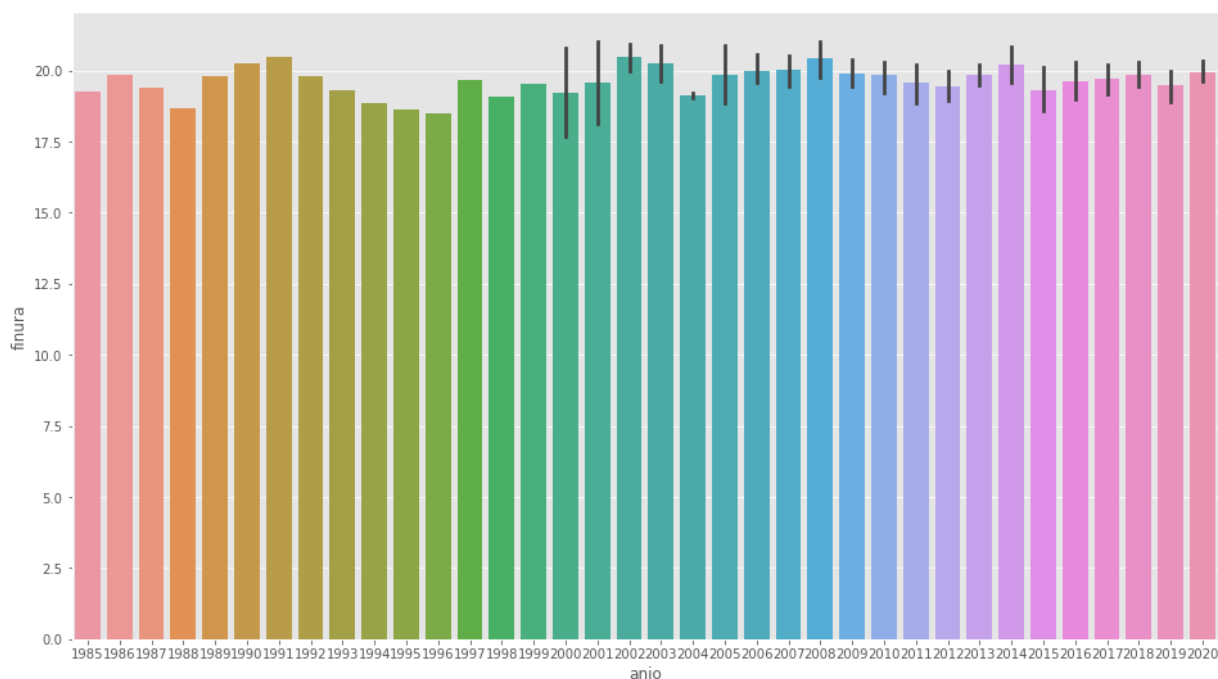
```
In [52]: sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='finura')
```

```
Out[52]: <AxesSubplot:xlabel='fecha', ylabel='finura'>
```



```
In [53]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='finura')
```

```
Out[53]: <AxesSubplot:xlabel='anio', ylabel='finura'>
```



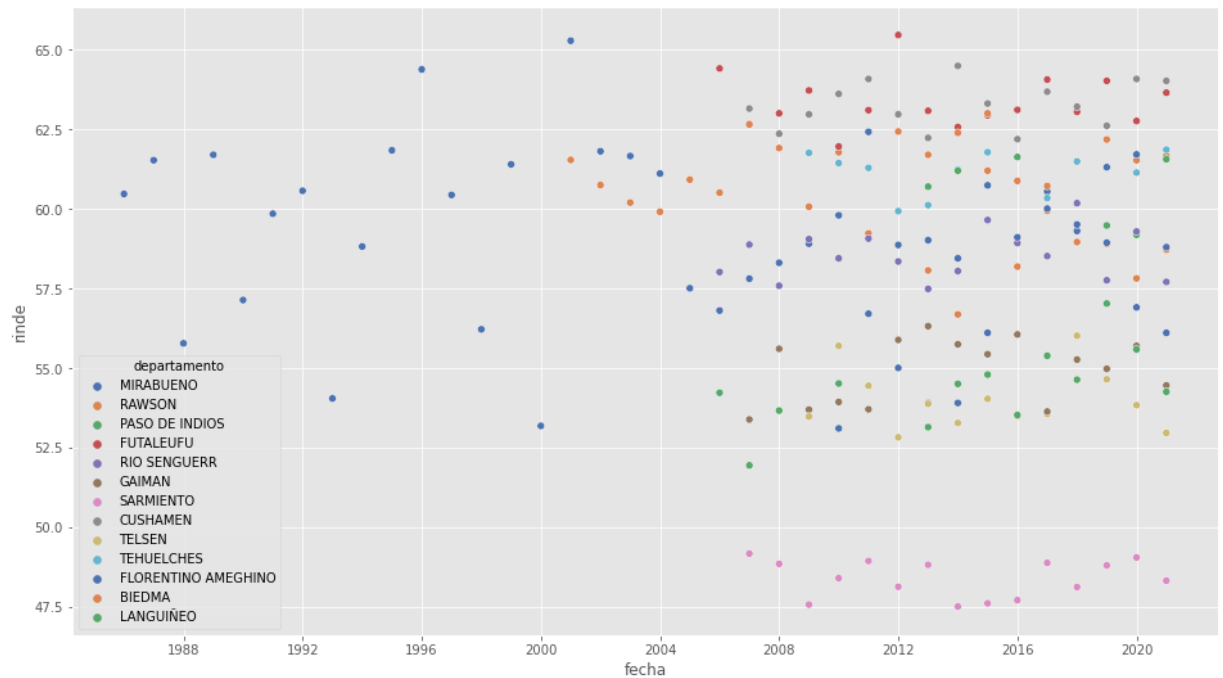
Para obtener información acerca de la variación en la finura de la lana a lo largo del tiempo, se pueden examinar estos tres gráficos. El scatterplot representa cada departamento mediante un punto en un plano cartesiano, y la posición del punto en los ejes X e Y indica la finura en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que permite una fácil comparación visual de la finura de la lana entre departamentos. Sin embargo, los datos están muy dispersos y no se puede observar ningún patrón general. A pesar de esto, es posible notar que en la provincia de Chubut, la finura de la lana se mantiene constante entre 18.5 y 20.5 en los gráficos de línea y de barras. Esto sugiere que Chubut mantiene un buen control sobre la finura de la lana, lo que es una señal positiva para la industria lanera en la región.

Rinde

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **rinde de la lana** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en el rinde de la lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

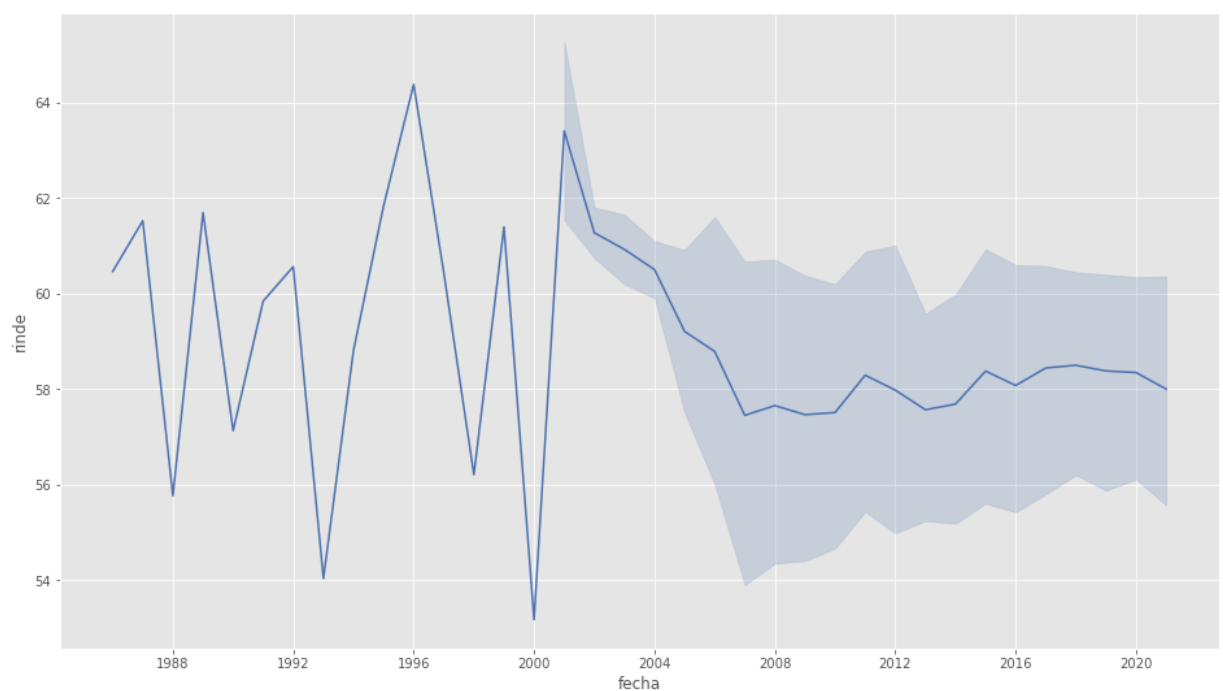
```
In [54]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='rinde',  
hue='departamento')
```

```
Out[54]: <AxesSubplot:xlabel='fecha', ylabel='rinde'>
```



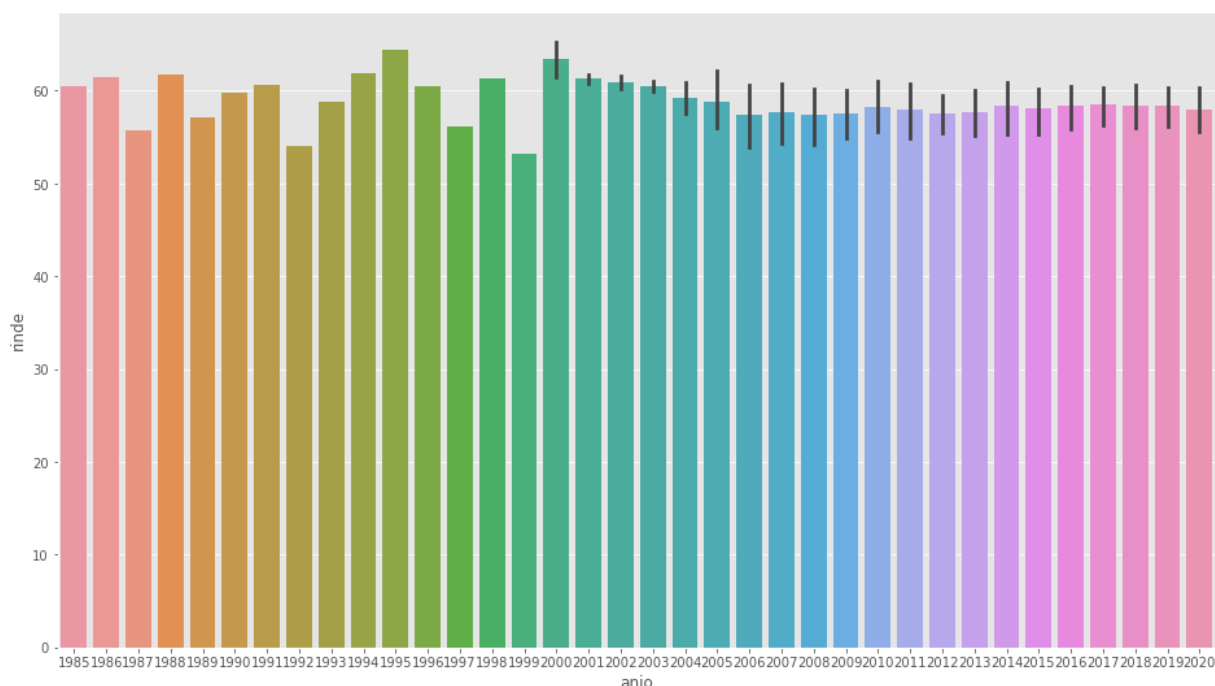
```
In [55]: sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='rinde')
```

```
Out[55]: <AxesSubplot:xlabel='fecha', ylabel='rinde'>
```



```
In [56]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='rinde')
```

```
Out[56]: <AxesSubplot:xlabel='anio', ylabel='rinde'>
```



Para obtener información sobre la variación en el rendimiento de lavado de la lana a lo largo del tiempo, se pueden analizar estos tres gráficos. El scatterplot representa cada departamento mediante un punto en un plano cartesiano, con la posición del punto en los ejes X e Y indicando el rendimiento en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que permite una fácil comparación visual del rendimiento de lavado de la lana entre departamentos.

Se puede observar que los departamentos de Cushamen y Futaleufú tienen los valores más altos de rendimiento de la lana, superiores a 62.5 para la mayoría de los años del dataset, lo que indica que la lana producida en esos departamentos tiene un alto valor en el mercado debido a su mayor rendimiento. Por otro lado, el departamento de Sarmiento presenta los valores más bajos de rendimiento de la lana, lo que indica que la lana producida en este departamento tiene un menor valor debido a su menor rendimiento.

Se observa que en 2001, solo El Mirabueno superó el valor de 65 de rinde, mientras que en 2012, solo Futaleufú logró este hito.

Sin embargo, en los gráficos de línea y barra se puede observar que el rendimiento de la lana en Chubut se mantiene bastante estable entre 54 y 64, mientras que el ideal es 75. Esto sugiere que la provincia de Chubut tiene un buen rendimiento de lavado de la lana, lo que es una buena señal para la industria de la lana en la región. Un alto rendimiento de lavado significa que la lana puede ser lavada y procesada con mayor eficiencia, lo que a su vez se traduce en una mayor calidad del producto final y mayores beneficios económicos para los productores de lana en la región.

Lluvia con otras variables

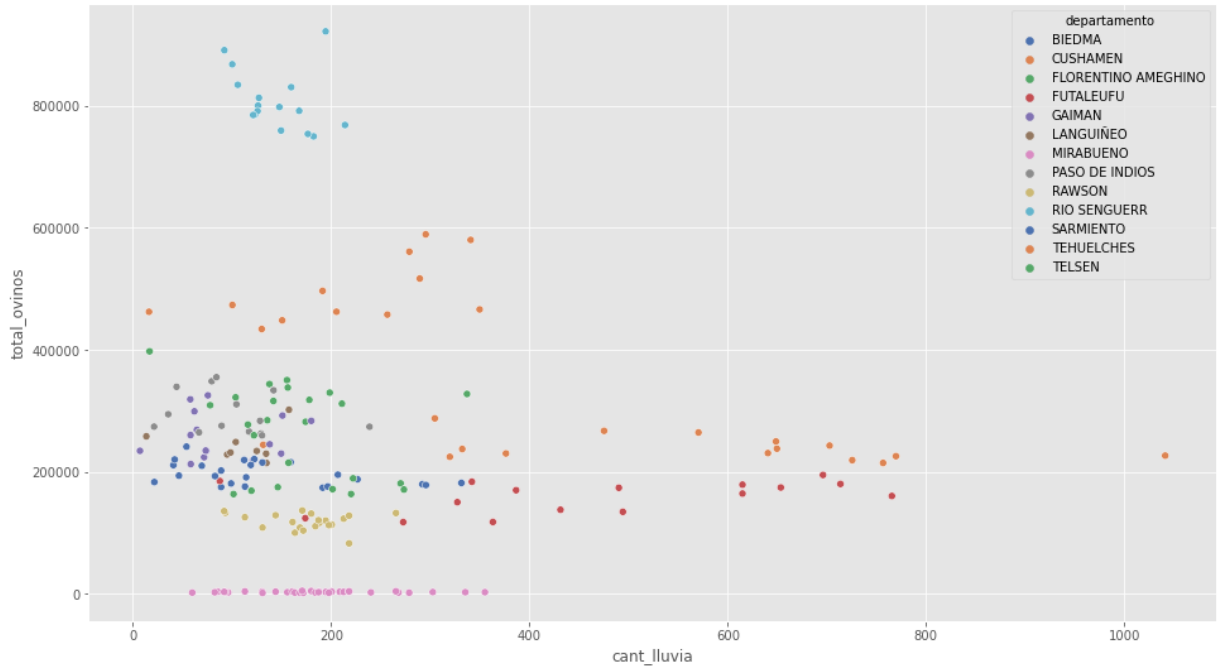
Para entender la influencia de la lluvia en la producción de ganado ovino en la región, es importante examinar la relación que existe entre la cantidad de lluvia acumulada en un año y las variables relacionadas con las existencias de ovejas y la producción de lana. De esta manera, podremos obtener una idea más clara de cómo la cantidad de lluvia puede afectar a la producción ganadera en la región.

Total de ovinos

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **total_ovinos** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de ovinos y la cantidad de lluvia recibida en cada año.

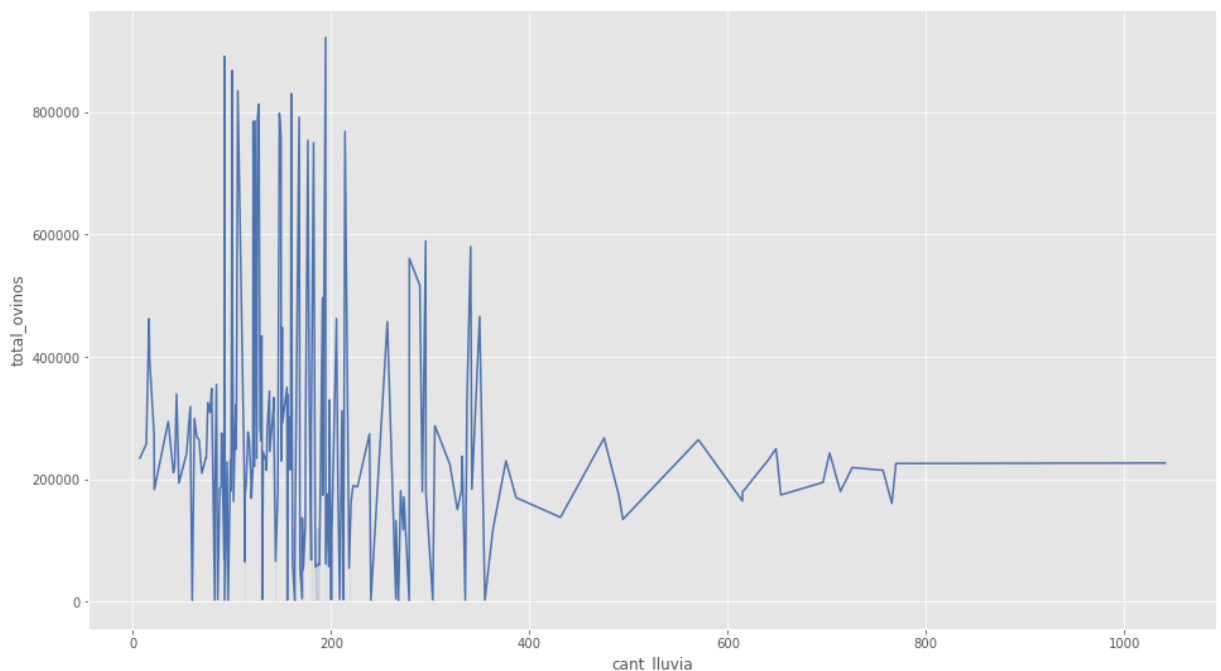
```
In [57]: sns.scatterplot(data=datos, x='cant_lluvia', y='total_ovinos',  
hue='departamento')
```

```
Out[57]: <AxesSubplot:xlabel='cant_lluvia', ylabel='total_ovinos'>
```



```
In [58]: sns.lineplot(data=datos, x='cant_lluvia', y='total_ovinos')
```

```
Out[58]: <AxesSubplot:xlabel='cant_lluvia', ylabel='total_ovinos'>
```



En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de ovinos para cada departamento. Sin embargo, si nos enfocamos en un solo

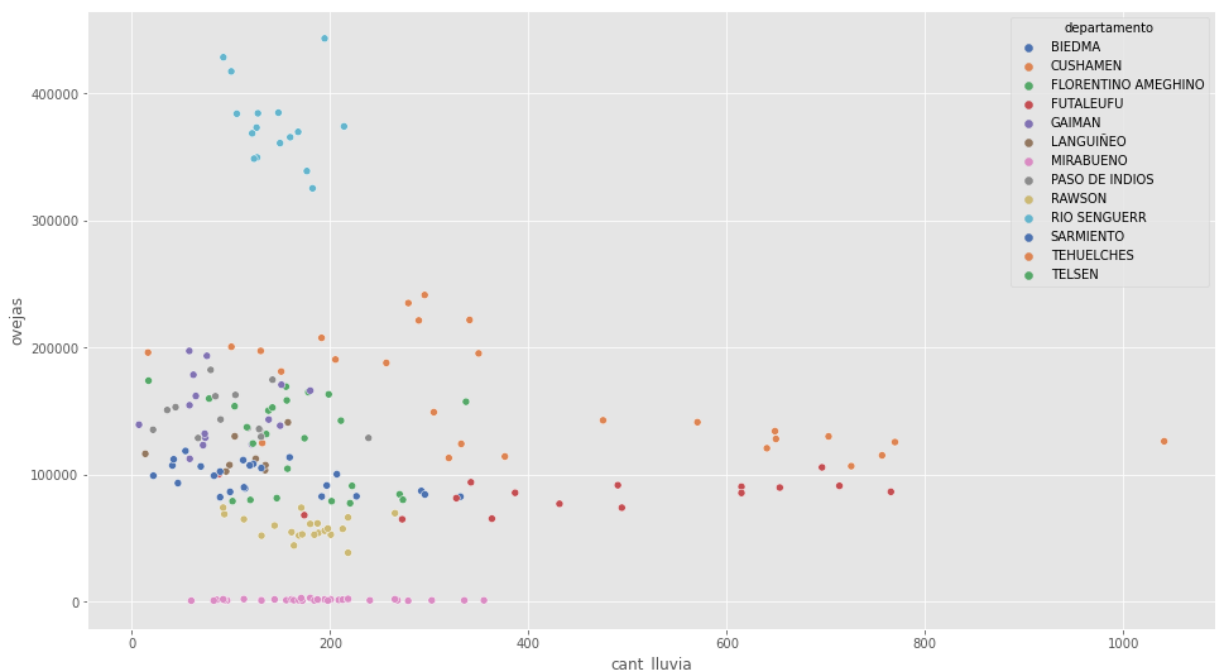
departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de ovinos, ya que al aumentar la lluvia, la cantidad de ovinos también se incrementa levemente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia excepcionalmente altos, pero no necesariamente tienen la mayor cantidad de ovinos. También se puede observar esta leve tendencia en el gráfico de líneas, aunque presenta muchos picos y valles que no tienen en cuenta el departamento. Además, dado que El Mirabueno tiene valores muchos más pequeños da la sensación que el total de ovejas es cero, pero en realidad sabemos que no.

Ovejas

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `ovejas` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de ovejas y la cantidad de lluvia recibida en cada año.

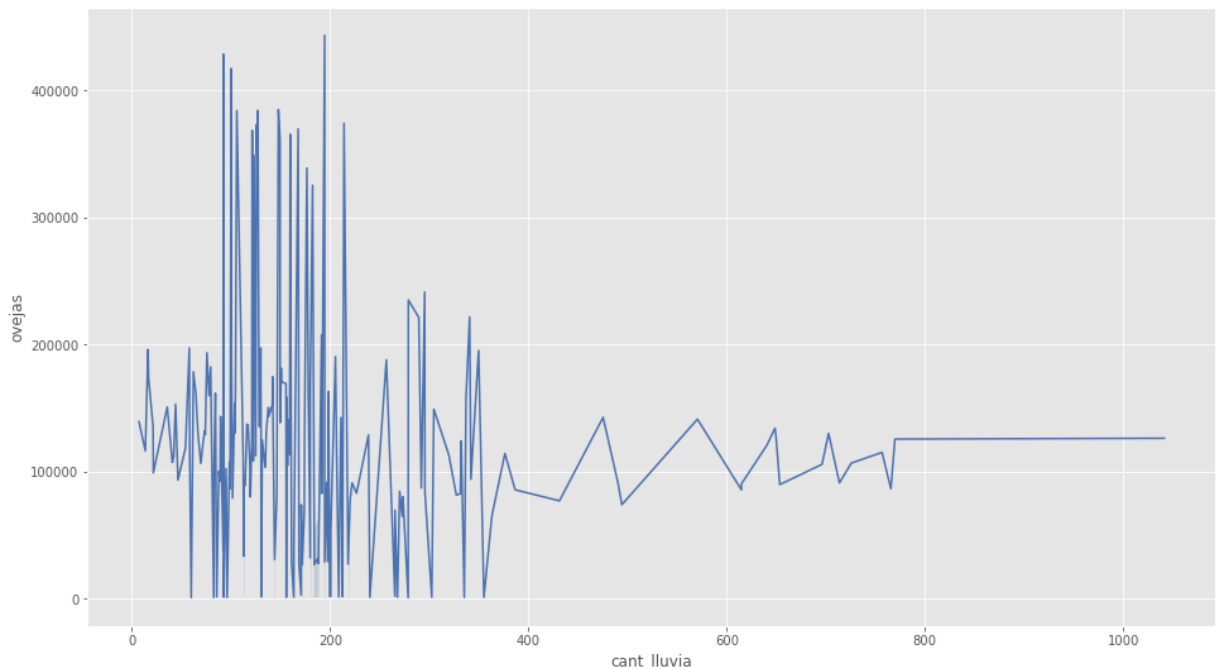
```
In [59]: sns.scatterplot(data=datos, x='cant_lluvia', y='ovejas',  
hue='departamento')
```

```
Out[59]: <AxesSubplot:xlabel='cant_lluvia', ylabel='ovejas'>
```



```
In [60]: sns.lineplot(data=datos, x='cant_lluvia', y='ovejas')
```

```
Out[60]: <AxesSubplot:xlabel='cant_lluvia', ylabel='ovejas'>
```



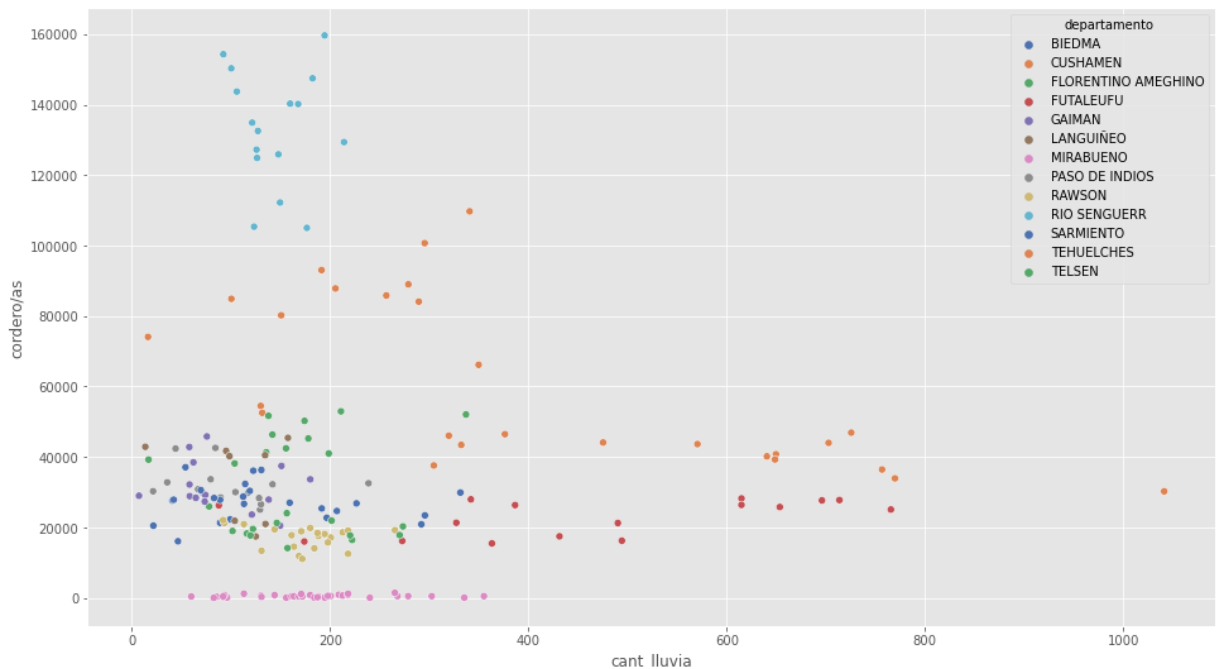
En los gráficos de dispersión y líneas que hemos analizado, no se puede apreciar una relación clara entre la cantidad de lluvia acumulada y la cantidad de ovejas en cada departamento. Sin embargo, si nos enfocamos en un departamento en particular, podemos observar una relación lineal leve entre la cantidad de lluvia y el número total de ovejas, lo que sugiere que cuando aumenta la cantidad de lluvia, también lo hace ligeramente el número de ovejas. Sin embargo, debemos tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que tienen datos atípicos de lluvia muy altos debido a su ubicación en la cordillera, pero no necesariamente tienen la mayor cantidad de ovejas. Además, el gráfico de líneas también muestra esta tendencia leve, pero con muchos picos y valles que dificultan su interpretación. También, dado que El Mirabueno tiene valores muchos más pequeños da la sensación que el total de ovejas es cero, pero en realidad sabemos que no.

Corderos

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **corderos** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de corderos y la cantidad de lluvia recibida en cada año.

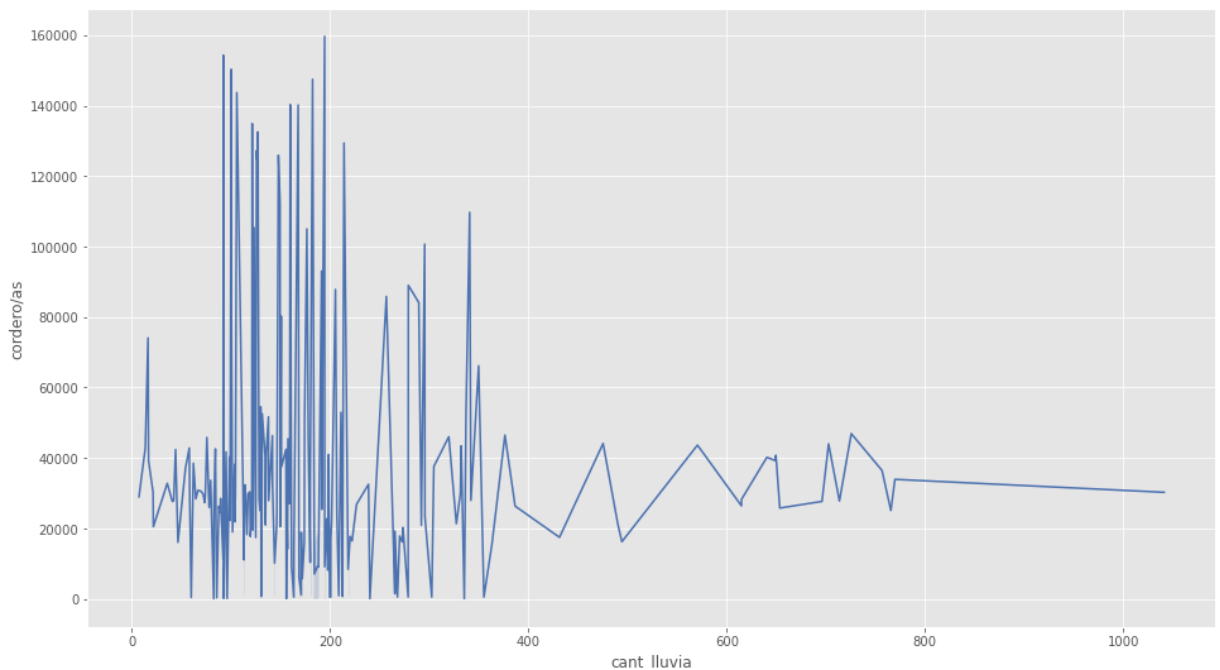
```
In [61]: sns.scatterplot(data=datos, x='cant_lluvia', y='cordero/as',
hue='departamento')
```

```
Out[61]: <AxesSubplot:xlabel='cant_lluvia', ylabel='cordero/as'>
```



```
In [62]: sns.lineplot(data=datos, x='cant_lluvia', y='cordero/as')
```

```
Out[62]: <AxesSubplot:xlabel='cant_lluvia', ylabel='cordero/as'>
```



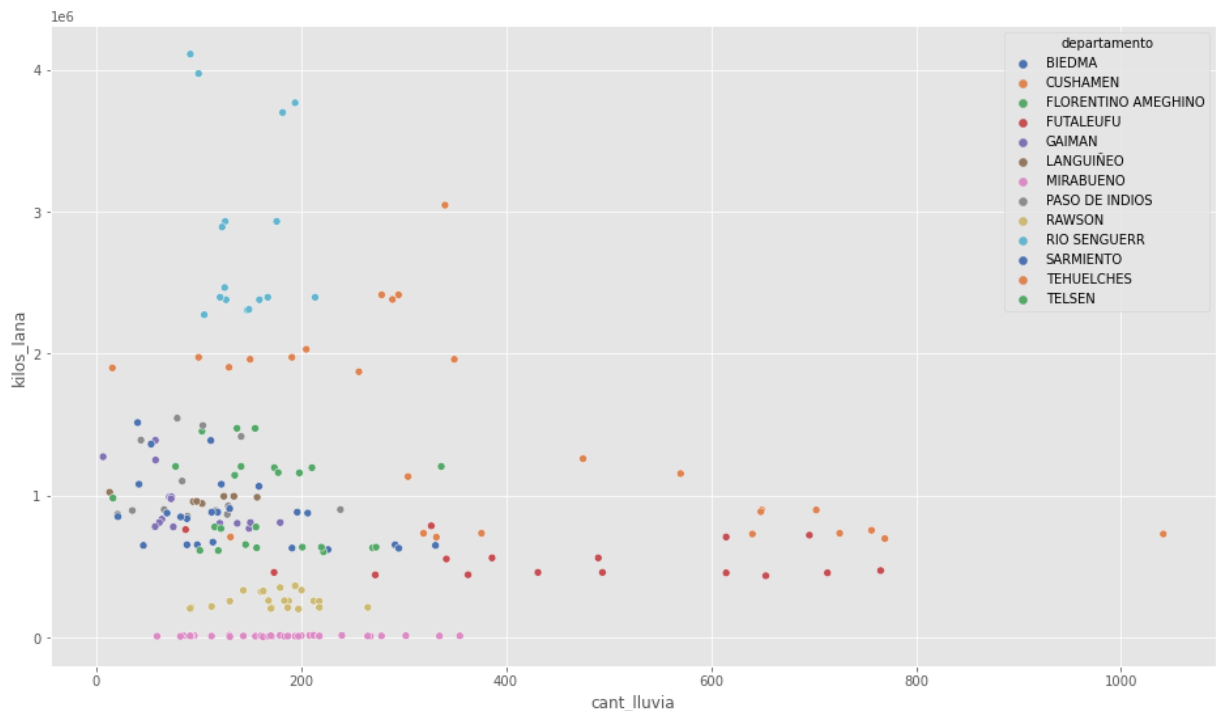
En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de corderos para cada departamento. Y sus formas son muy similares a los anteriores graficos del total de ovinos y ovejas, por lo tanto, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de corderos, ya que al aumentar la lluvia, la cantidad de corderos también se incrementa ligeramente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia, pero no necesariamente tienen la mayor cantidad de corderos. También se puede observar esta leve tendencia en el gráfico de líneas, aunque al presentar tantos picos y valles, hace difícil su interpretación. Además, dado que El Mirabueno tiene valores muchos más pequeños da la sensación que el total de ovejas es cero, pero en realidad sabemos que no.

Kilo de lana

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **kilos de lana** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de lana y la cantidad de lluvia recibida en cada año.

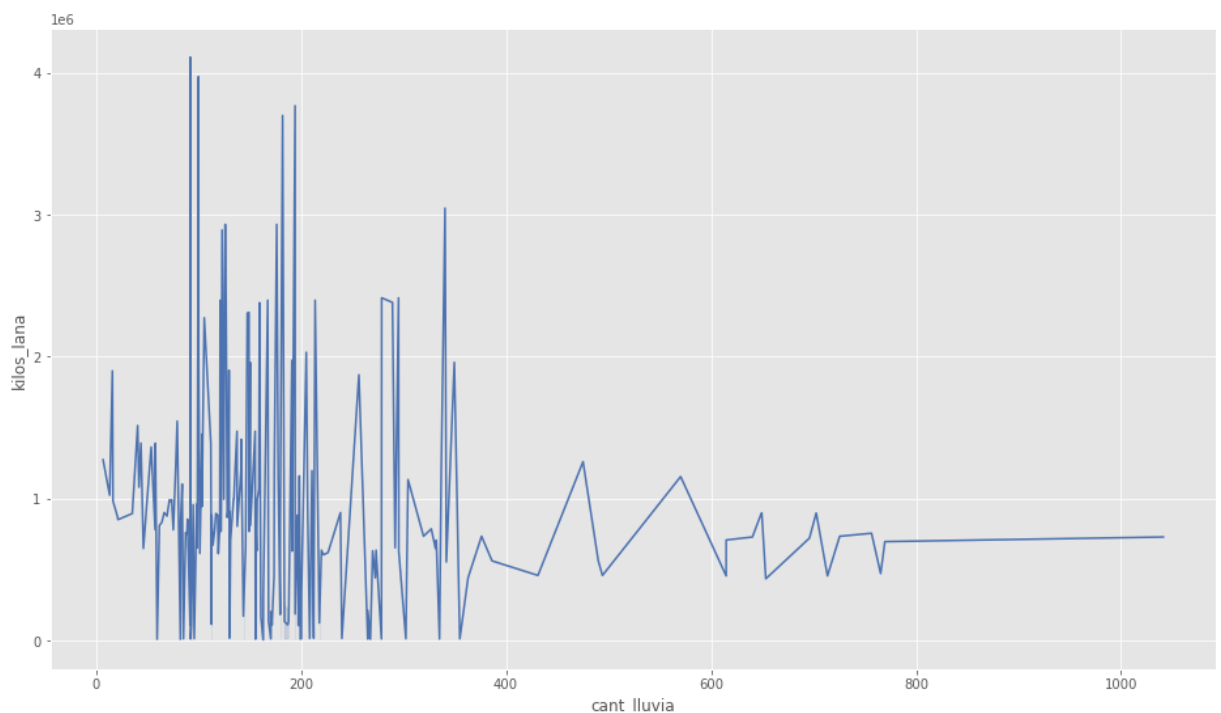
```
In [63]: sns.scatterplot(data=datos, x='cant_lluvia', y='kilos_lana',  
hue='departamento')
```

```
Out[63]: <AxesSubplot:xlabel='cant_lluvia', ylabel='kilos_lana'>
```



```
In [64]: sns.lineplot(data=datos, x='cant_lluvia', y='kilos_lana')
```

```
Out[64]: <AxesSubplot:xlabel='cant_lluvia', ylabel='kilos_lana'>
```



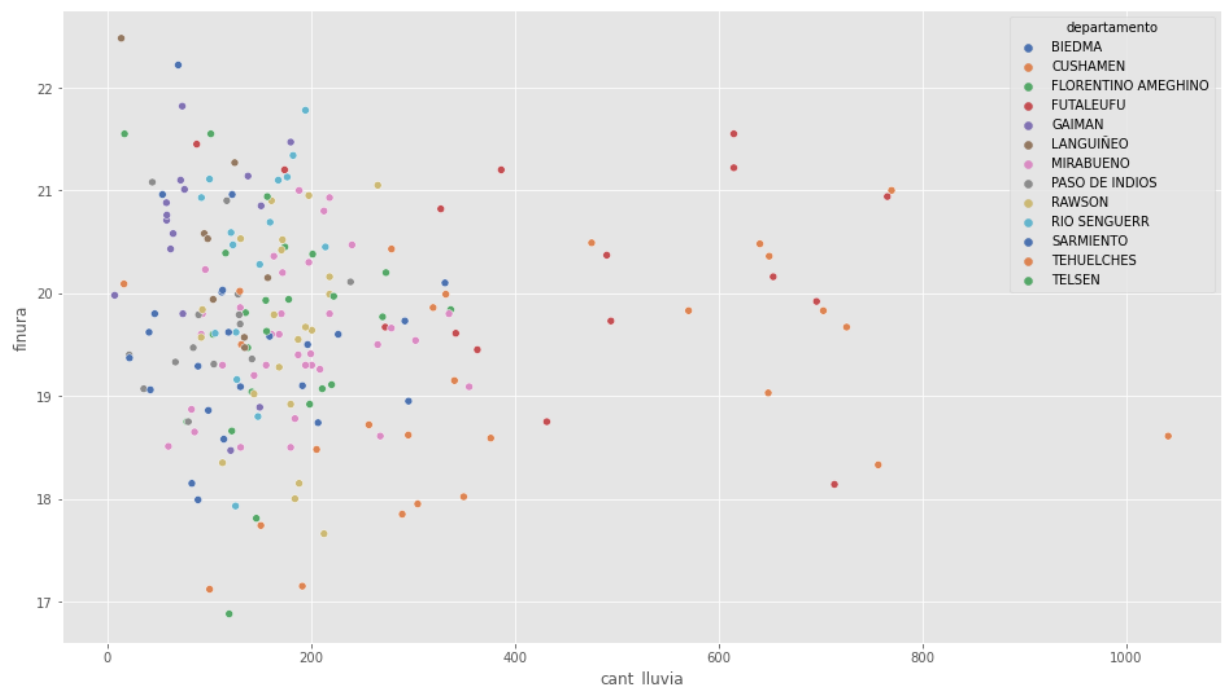
En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de lana para cada departamento. Y sus formas son muy similares a los anteriores 3 gráficos de existencias, por lo tanto, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de lana, ya que al aumentar la lluvia, la cantidad de lana también se incrementa ligeramente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia excepcionalmente altos, pero no necesariamente tienen la mayor cantidad de lana. También se puede observar esta leve tendencia en el gráfico de líneas, aunque presenta muchos picos y valles que no tienen en cuenta el departamento. Además, dado que El Mirabueno tiene valores muchos más pequeños da la sensación que el total de ovejas es cero, pero en realidad sabemos que no.

Finura

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `finura` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en el valor de la finura y la cantidad de lluvia recibida en cada año.

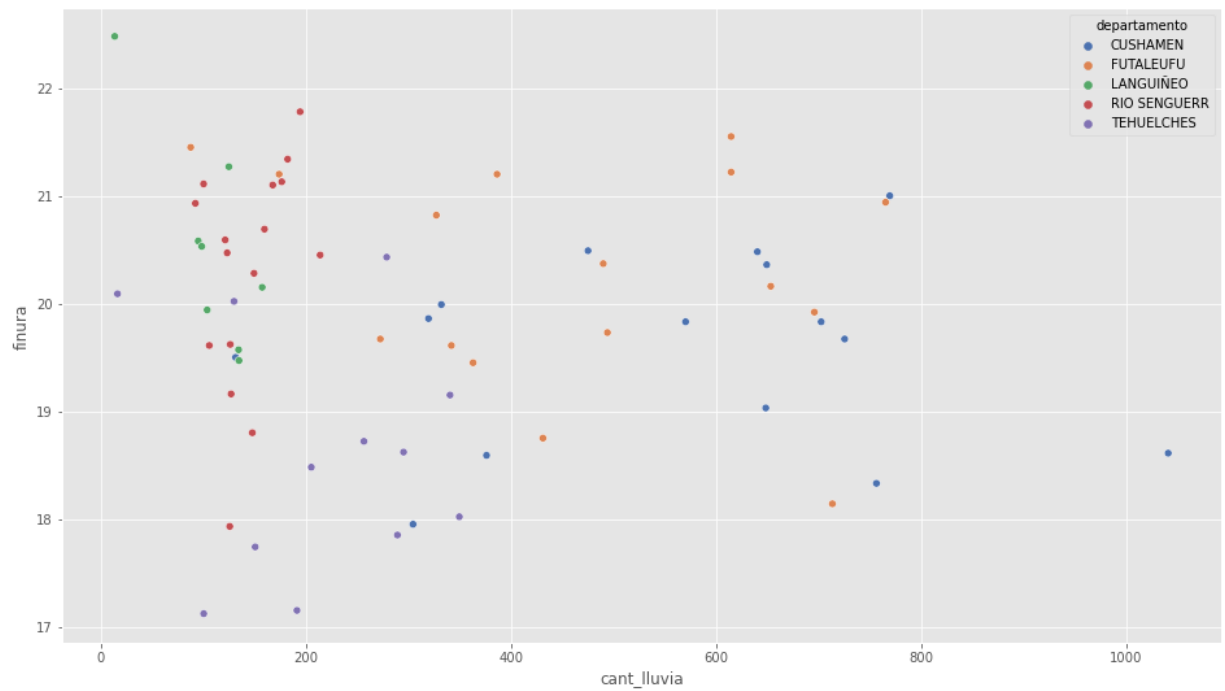
```
In [65]: sns.scatterplot(data=datos, x='cant_lluvia', y='finura',  
hue='departamento')
```

```
Out[65]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



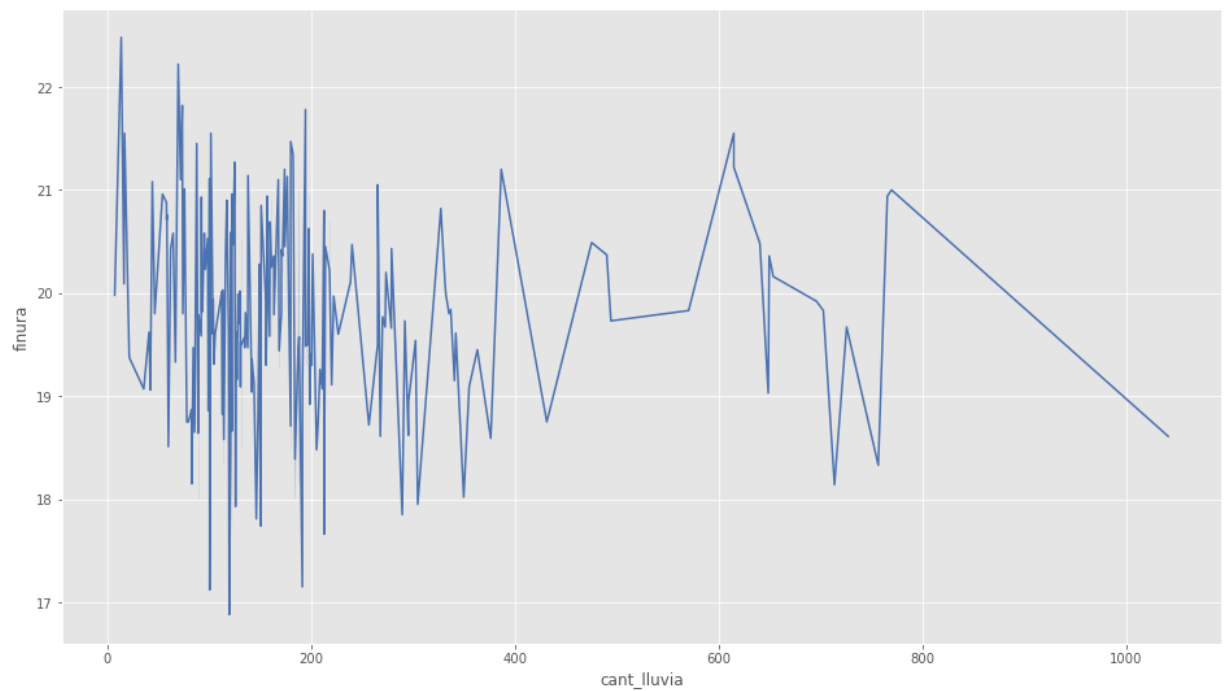
```
In [7]: sns.scatterplot(data=departamentos_cordillera, x='cant_lluvia',  
y='finura', hue='departamento')
```

```
Out[7]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



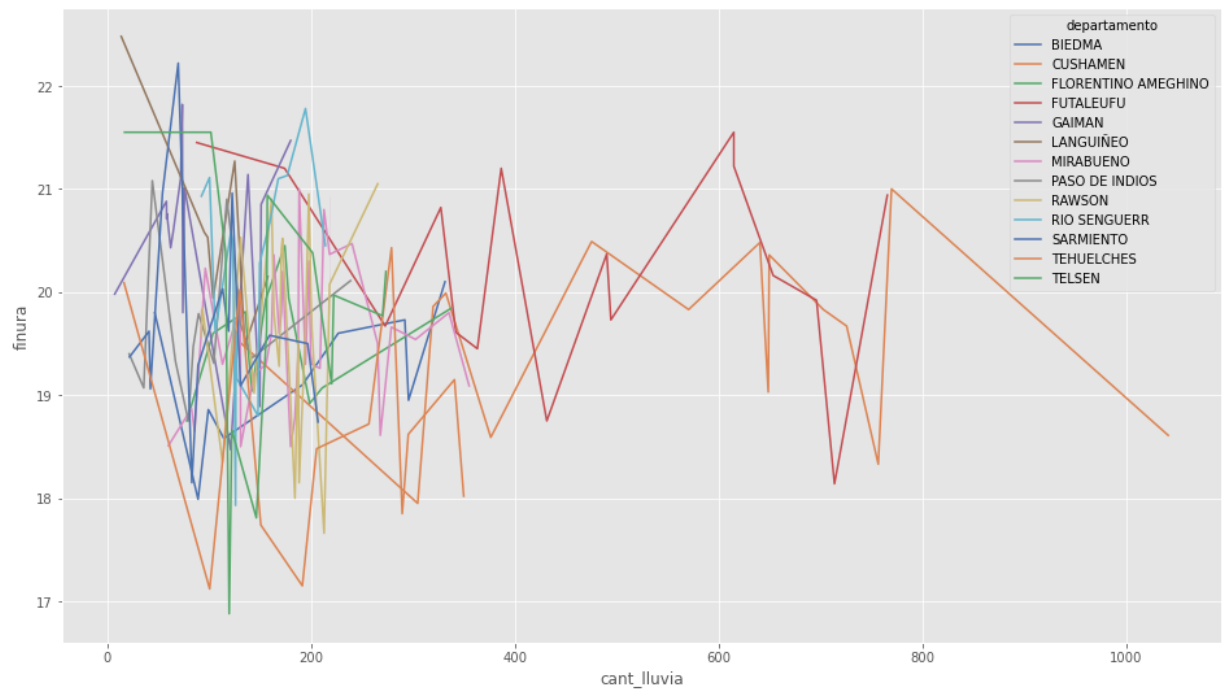
In [66]: `sns.lineplot(data=datos, x='cant_lluvia', y='finura')`

Out[66]: `<AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>`



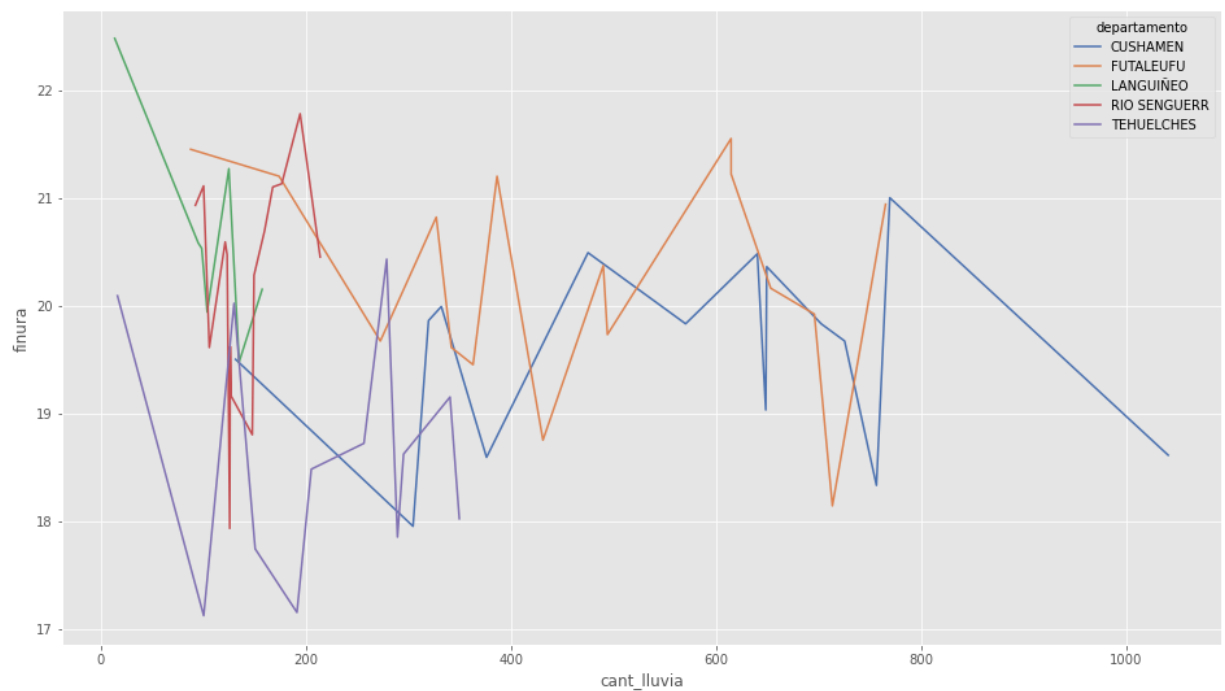
In [8]: `sns.lineplot(data=datos, x='cant_lluvia', y='finura',
hue='departamento')`

Out[8]: `<AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>`



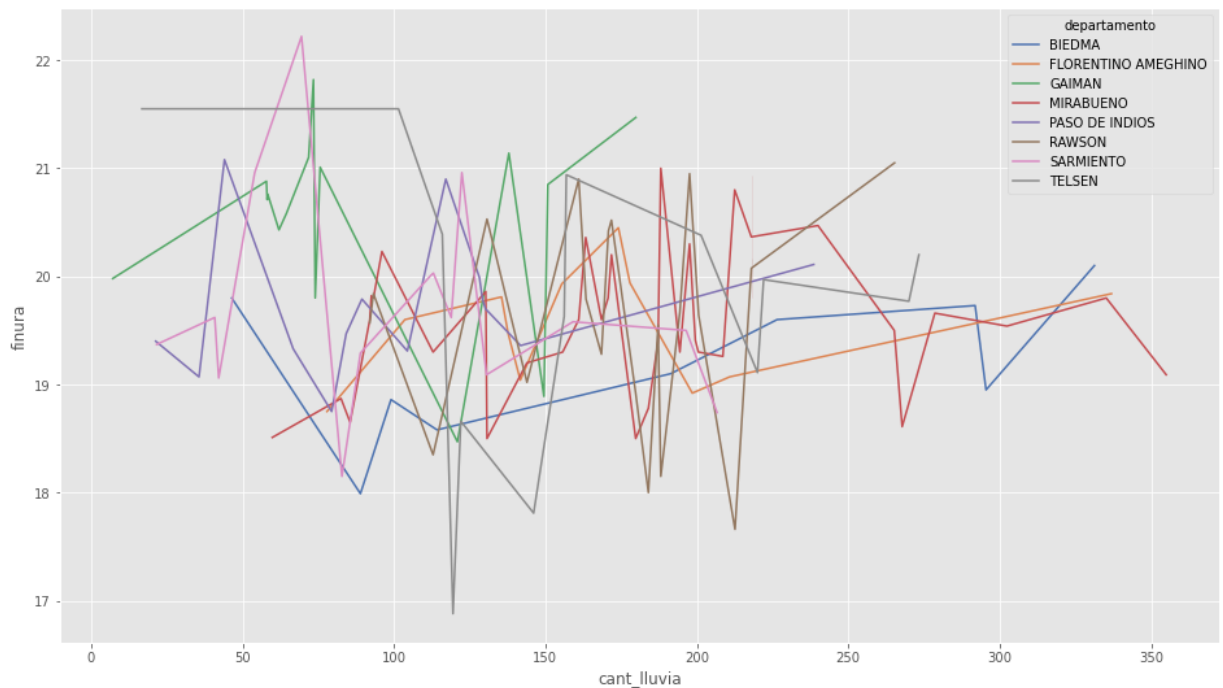
```
In [9]: sns.lineplot(data=departamentos_cordillera, x='cant_lluvia',
y='finura', hue='departamento')
```

```
Out[9]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



```
In [10]: sns.lineplot(data=departamentos_meseta, x='cant_lluvia',
y='finura', hue='departamento')
```

```
Out[10]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



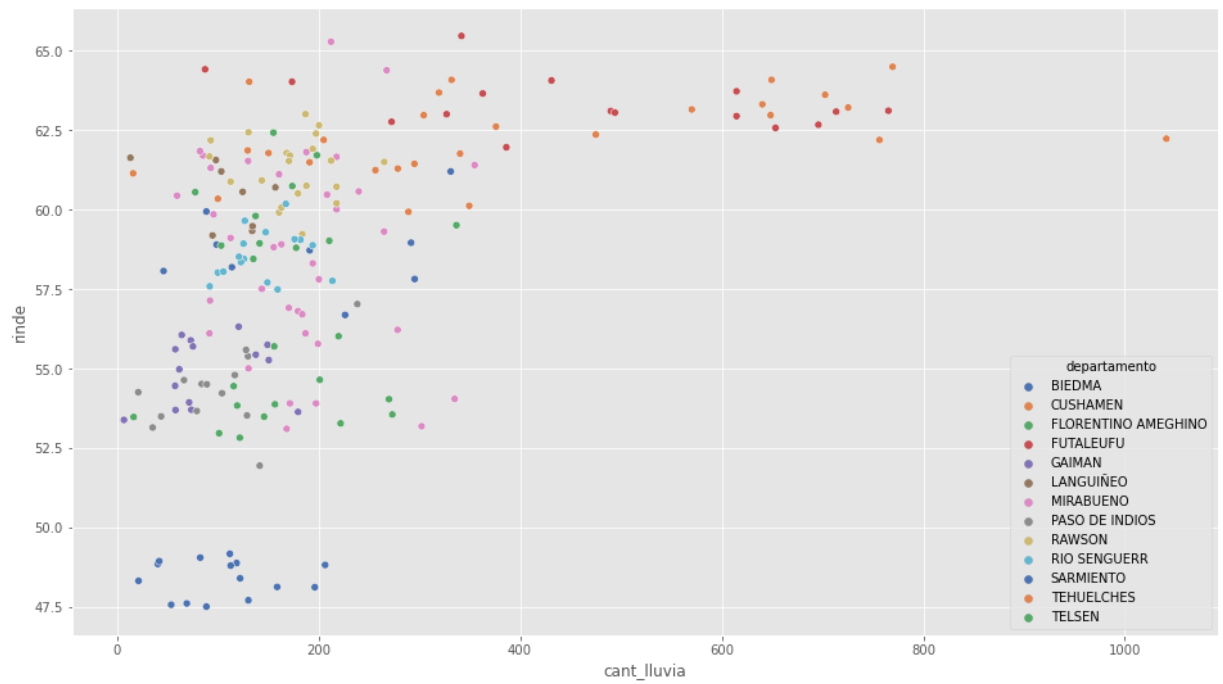
No se puede apreciar una relación clara entre la finura de la lana y la cantidad de lluvia acumulada en el año, y se observa que en algunos casos, a medida que aumenta la cantidad de lluvia, la lana se vuelve más gruesa, que no necesariamente es lo que buscan los productores ganaderos. En el gráfico de líneas no se observa ningún patrón discernible y en el gráfico de dispersión, la mayoría de los datos se concentran en un rango estrecho entre 0 y 400 mm de lluvia y una finura de lana entre 18 y 21 micrones, lo que sugiere que hay poca variabilidad en los datos. Sin embargo, existen valores atípicos en los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen una mayor cantidad de lluvia en promedio.

Rinde

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `rinde` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en el valor del rinde y la cantidad de lluvia recibida en cada año.

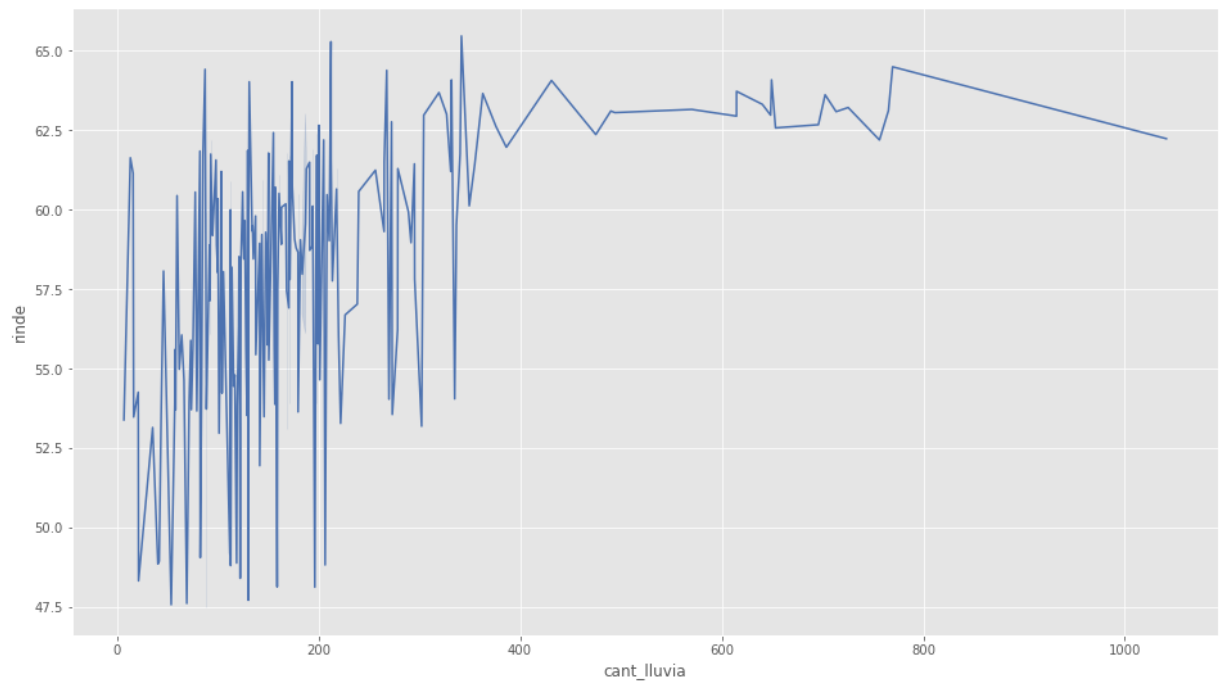
```
In [67]: sns.scatterplot(data=datos, x='cant_lluvia', y='rinde',
hue='departamento')
```

```
Out[67]: <AxesSubplot:xlabel='cant_lluvia', ylabel='rinde'>
```



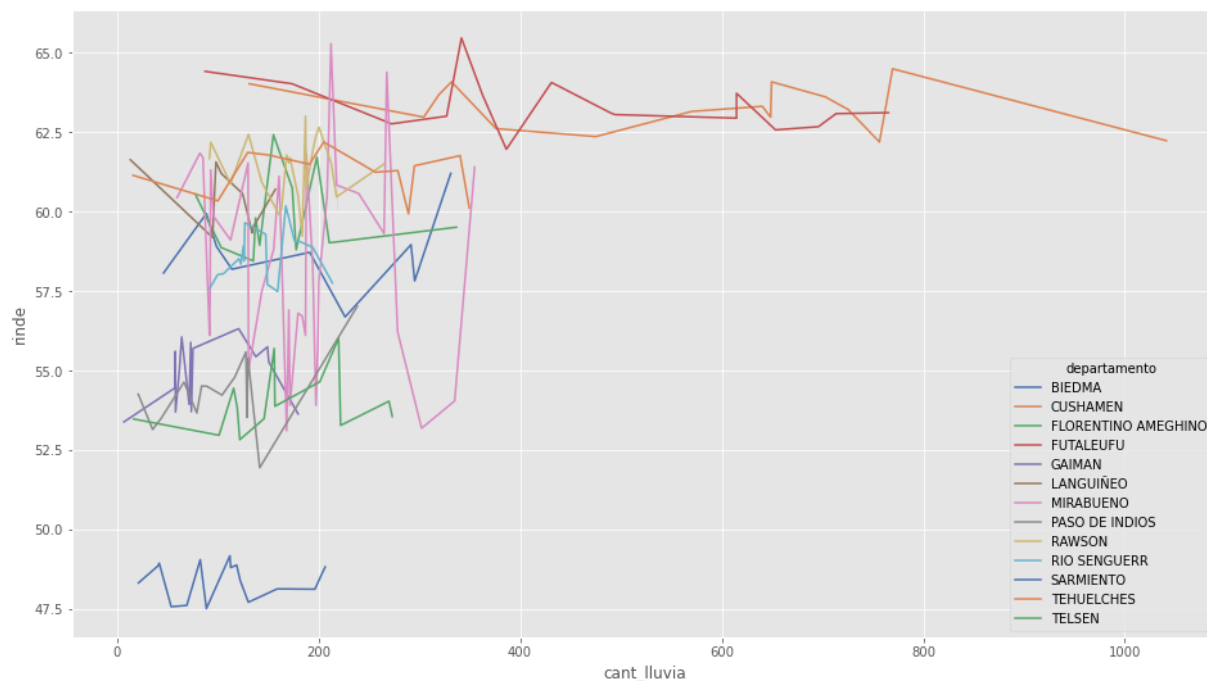
```
In [68]: sns.lineplot(data=datos, x='cant_lluvia', y='rinde')
```

```
Out[68]: <AxesSubplot:xlabel='cant_lluvia', ylabel='rinde'>
```



```
In [11]: sns.lineplot(data=datos, x='cant_lluvia', y='rinde',  
hue="departamento")
```

```
Out[11]: <AxesSubplot:xlabel='cant_lluvia', ylabel='rinde'>
```



En cuanto al rendimiento, tampoco se observa una relación clara con la cantidad de lluvia, aunque sí se puede ver una relación un poco más directa que en la finura de la lana. En este caso, cuando aumenta la cantidad de lluvia, también aumenta el rendimiento, lo cual es algo que los productores de ganadería ovina buscan. Esto podría significar que la lana está menos sucia y, por lo tanto, rinde más al lavarla. Sin embargo, los datos se mantienen constantes en la mayoría de los departamentos, independientemente de la cantidad de lluvia. En Cushamen y Futaleufú, que son los departamentos con la mayor cantidad de lluvia, se obtiene el mayor rendimiento, alcanzando un máximo de 64.

2.6. Análisis Gráfico - Dataset 3

A continuación, realizaremos un análisis visual del conjunto de datos con el objetivo de identificar patrones o relaciones entre las variables. Para ello, haremos uso de las librerías Pandas, Seaborn y Matplotlib.

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [2]: %matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

```
In [3]: datos =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_actua
```

Para que los graficos funcionen mejor es necesario que el atributo fecha sea de tipo date.

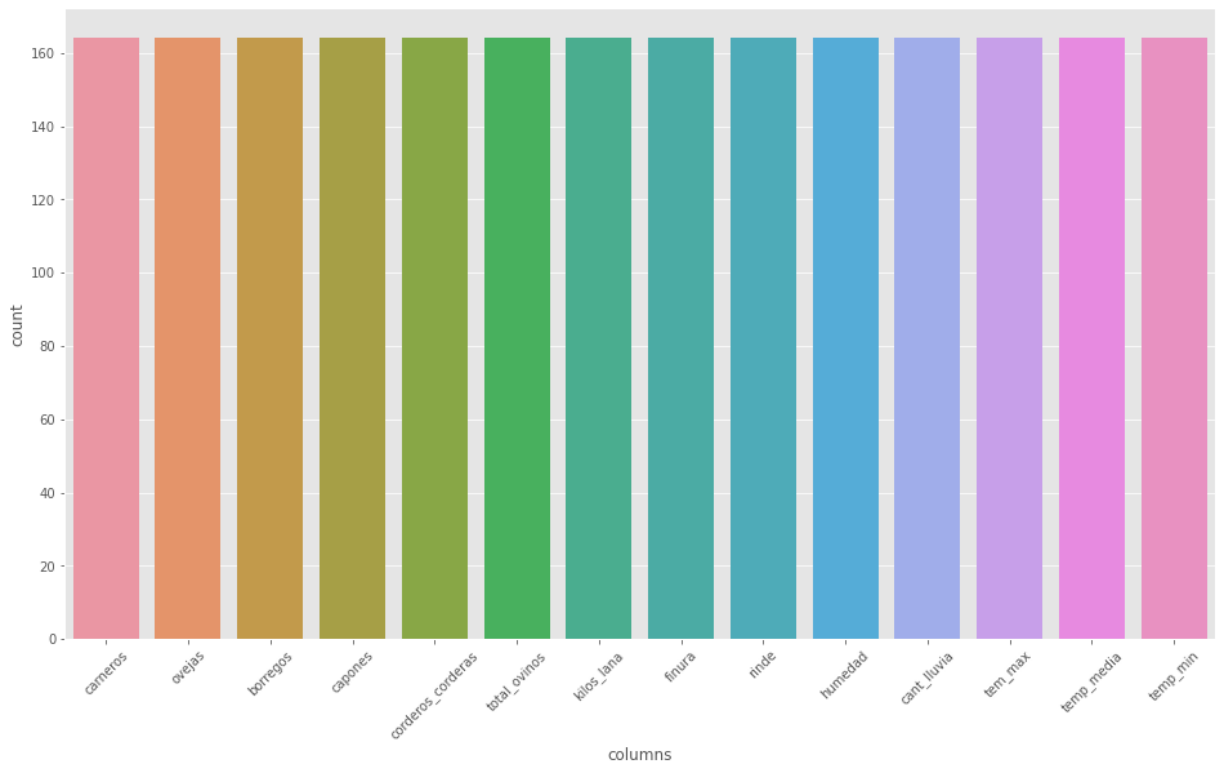
```
In [4]: datos['fecha'] = pd.to_datetime(datos['fecha'])
```

En primer lugar, realizaremos un análisis conjunto de las variables para tener una visión general del dataset. A continuación, procederemos a analizar las variables de forma individual para obtener una comprensión más detallada de cada una de ellas. Por último, relacionaremos los atributos del conjunto de datos para identificar patrones y posibles relaciones entre ellos.

Variables en conjunto

En primer lugar, realizaremos un gráfico que nos permitirá conocer la cantidad de datos disponibles para cada atributo del dataset, observando que cada columna cuenta con 164 datos.

```
In [5]: sns.countplot(data=datos)
plt.xticks(rotation=45)
plt.xlabel('columns')
plt.show()
```

Luego, utilizaremos la función `pairplot` de seaborn para obtener una matriz que muestra un gráfico de dispersión para cada combinación de atributos. Dicha matriz es bastante amplia debido a la cantidad de variables presentes en el dataset, pero podemos identificar algunas gráficas que parecen ajustarse a una regresión lineal en principio, principalmente aquellas relacionadas con las existencias de ovinos, como carneros, ovejas, borregos, capones, corderos, coderas y el total de ovinos.

```
In [6]: sns.pairplot(datos, hue="departamento")
```

```
Out[6]: <seaborn.axisgrid.PairGrid at 0x7fab73167c40>
```



Asimismo, se pueden apreciar variables relacionadas con la temperatura, como la temperatura máxima, media y mínima, que también muestran una tendencia lineal. Sin embargo, al combinar datos de producción con información climática, la mayoría de las columnas no parecen ajustarse a un modelo lineal en principio.

Para mejorar la visualización de las relaciones entre variables, vamos a crear una matriz de correlación y graficarla en un mapa de calor de seaborn. De esta forma, podremos identificar de manera más clara cuáles son las variables que tienen una mayor o menor correlación entre sí.

Una matriz de correlación es una tabla que muestra las correlaciones entre todas las variables de un conjunto de datos. Las correlaciones miden el grado en que dos variables están relacionadas entre sí, y pueden oscilar entre -1 y 1.

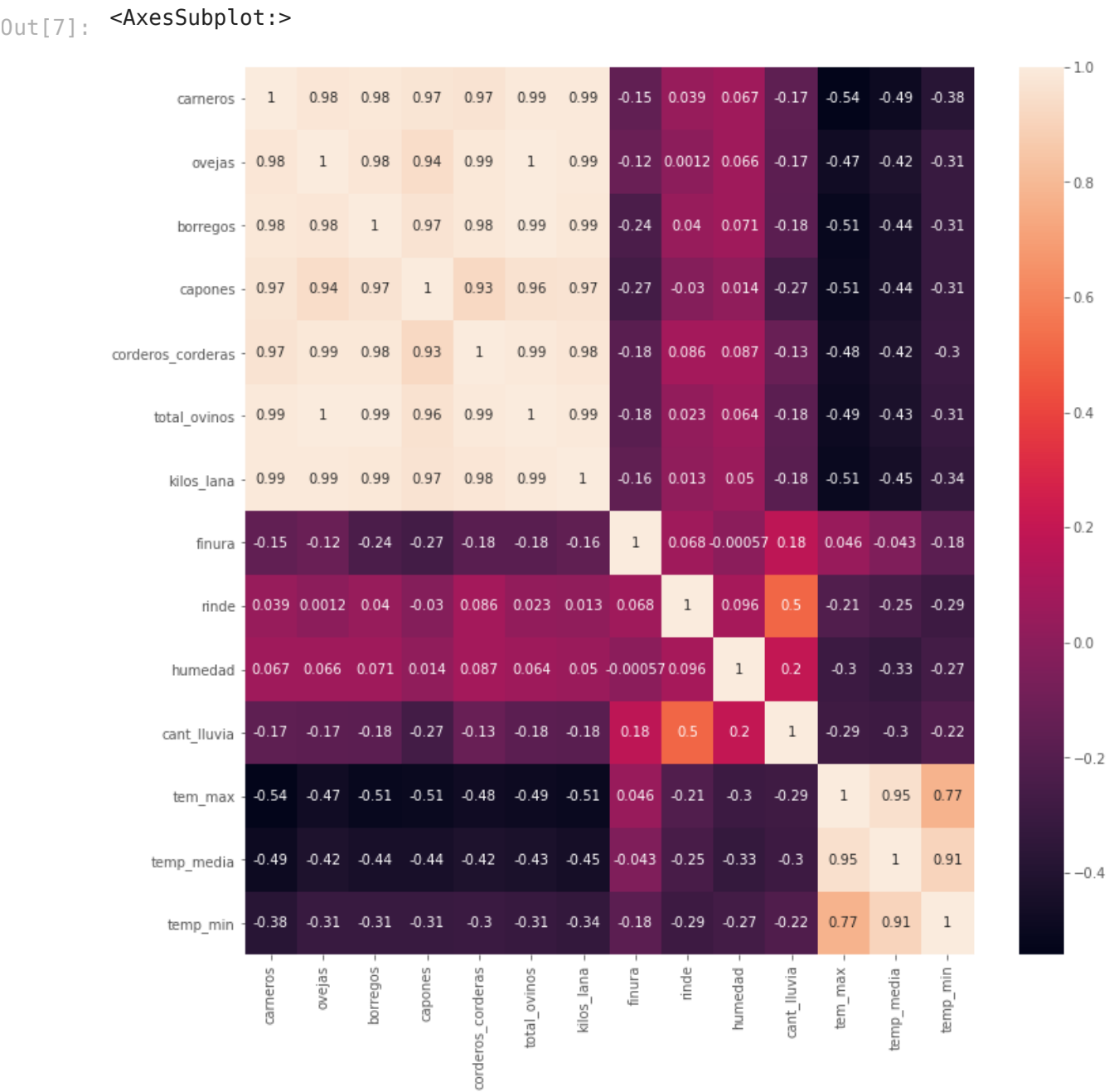
Un valor de correlación de 1 significa que dos variables están perfectamente correlacionadas positivamente, es decir, que cuando una aumenta, la otra también lo hace en la misma medida. Por otro lado, un valor de -1 indica una correlación perfectamente negativa, lo que significa que cuando una variable aumenta, la otra disminuye en la misma medida. Finalmente, un valor de 0 indica que no hay correlación entre dos variables.

La matriz de correlación se representa como una tabla cuadrada, donde cada variable se encuentra tanto en la fila como en la columna de la matriz. En la diagonal principal, los valores son siempre 1, ya que representa la correlación de una variable consigo misma. Los valores fuera de la diagonal principal

indican la correlación entre dos variables diferentes, y su posición en la matriz es simétrica, es decir, el valor de la fila i y la columna j es el mismo que el valor de la fila j y la columna i.

En la matriz de correlación, representada por un gráfico de calor, se puede observar la intensidad de la correlación entre las diferentes variables del dataset. De acuerdo con la documentación de seaborn para el heatmap, las correlaciones negativas se muestran en violeta oscuro, las correlaciones positivas en naranja claro y las correlaciones cercanas a cero en rosado.

```
In [7]: correlation_mat = datos.corr()  
fig, ax = plt.subplots(figsize = (12, 12))  
sns.heatmap(correlation_mat, annot = True, ax = ax)
```



En este caso, se puede observar que los atributos relacionados con las existencias ovinas tienen una correlación muy alta entre ellos, con valores que oscilan entre 0.93 y 0.99. Además, los atributos relacionados con la temperatura también presentan una alta correlación entre ellos, con valores que van desde 0.77 a 0.95. Estos atributos son los que previamente se habían identificado como los que seguían una forma lineal en el gráfico de dispersión.

Otra correlación interesante a destacar es la que se presenta entre la cantidad de kilos de lana producidos y las existencias ovinas, ya que se observa una alta correlación con valores que oscilan entre 0.97 y 0.99.

Teniendo en cuenta la bibliografía y la experiencia de algunos productores, se sabe que la lluvia es un factor determinante en la producción ganadera ovina de la región, debido a que se trata de una zona con bajos niveles de precipitaciones. Es por esto que resulta interesante analizar la correlación existente entre la lluvia acumulada en el año y las existencias ovinas, así como también con la producción de lana. Al observar la matriz de correlación, los valores obtenidos para estas relaciones están más cerca del color rosa que del violeta o naranja, lo que indica que no existe una correlación significativa entre estas variables. En detalle, se aprecia una correlación negativa débil entre la cantidad de lluvia y las existencias ovinas, así como también con la producción de lana. Los valores obtenidos oscilan entre -0.17 y -0.18 para carneros, ovejas y borregos, mientras que para los corderos la relación es de -0.13 y para los capones de -0.27. Los kilos de lana también muestran una correlación negativa débil de -0.18 con la cantidad de lluvia, lo que sugiere que a medida que aumenta la cantidad de lluvia en el año, la producción de lana y existencias tiende a disminuir ligeramente. Sin embargo, en el caso de la finura y el rinde de la lana, se observa una correlación positiva con la cantidad de lluvia. La correlación entre la finura de la lana y la cantidad de lluvia es positiva débil, con un valor de 0.18, mientras que la correlación entre el rinde de la lana y la cantidad de lluvia es positiva moderada, con un valor de 0.5. Esto significa que a medida que aumenta la cantidad de lluvia, el rinde y la finura de la lana también tienden a aumentar. Es importante tener en cuenta que la correlación no implica causalidad y que existen otros factores que pueden influir en esta relación.

Variables independientes

A continuación lo que se va a hacer es explicar el funcionamiento de los gráficos que se van a usar para el análisis y después vamos a graficar para cada una de las variables que se van a utilizar en los entrenamientos.

Gráficos

Histogramas

Para saber si los datos están sesgados, es necesario analizar la distribución de los datos. Una distribución sesgada indica que los datos se concentran en un lado de la distribución, lo que puede afectar la interpretación de los resultados.

Se pueden visualizar las distribuciones utilizando histogramas o gráficos de densidad para identificar visualmente si hay algún tipo de sesgo en los datos.

El comando `sns.histplot(data=datos['variable_X'])` te muestra un histograma de la columna 'variable_X' del dataframe 'datos'. En el eje `x` se encuentra el rango de valores de la variable y en el eje `y` se encuentra la frecuencia de ocurrencia de cada valor. Este tipo de gráfico es útil para visualizar la distribución de los datos y para detectar posibles valores atípicos o sesgos en la distribución.

Diagrama de caja

También, para detectar valores atípicos, se puede utilizar la visualización de un diagrama de caja (boxplot), donde se pueden identificar los valores que se encuentran por encima o por debajo de los

límites del diagrama. En un boxplot, los valores atípicos se representan como puntos más allá de los límites de los bigotes del diagrama.

El comando `sns.boxplot(data=datos['variable_X'])` muestra un diagrama de caja (boxplot) que resume la distribución de los valores en la columna 'variable_X' del DataFrame 'datos'. La línea horizontal dentro de la caja representa la mediana, mientras que la caja en sí representa el rango intercuartil (IQR), es decir, el rango de valores que se encuentran entre el primer y tercer cuartil de la distribución. Los puntos fuera de la caja, llamados valores atípicos, representan valores que están más allá del IQR. Este gráfico puede ser útil para identificar valores extremos o fuera de lo común en la distribución de los datos.

Diagrama de dispersión

Con `sns.scatterplot(data=datos['variable_X'])` se mostrará un gráfico de dispersión con el eje *X* representando los índices de las filas del dataframe y el eje *Y* representando los valores de la columna 'variable_X'. Cada punto en el gráfico representa un valor de 'variable_X' para una fila específica del dataframe. El gráfico puede ser útil para visualizar la distribución y la relación entre los valores de la columna 'variable_X'.

Diagrama de línea

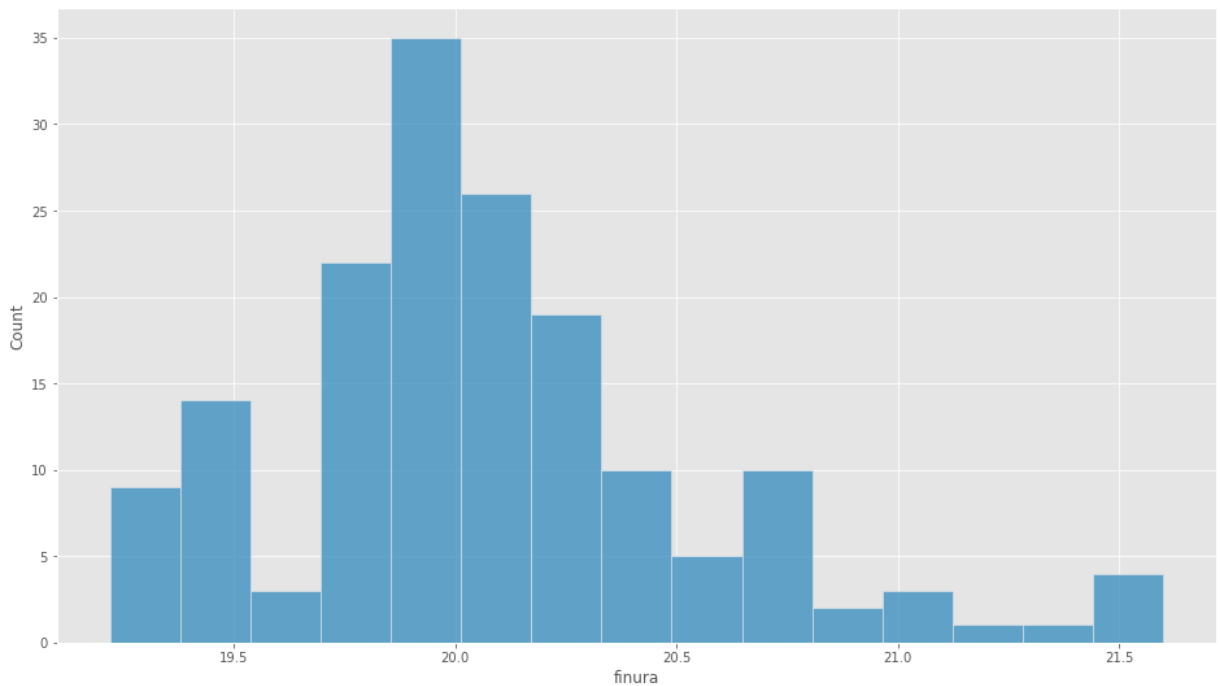
El código `sns.lineplot(data=datos['variable_X'])` genera un gráfico de línea de la columna 'variable_X' del DataFrame 'datos'. El eje *x* del gráfico representa el índice de la serie (es decir, el número de fila), y el eje *y* representa el valor de la columna 'variable_X' en cada fila. Por defecto, Seaborn dibuja una línea que conecta los puntos en orden, lo que ayuda a visualizar la tendencia a lo largo del tiempo (o cualquier otra dimensión representada por el índice).

Finura

El histograma presentado muestra que la variable `finura` presenta una distribución asimétrica.

```
In [8]: sns.histplot(data=datos['finura'])
```

```
Out[8]: <AxesSubplot:xlabel='finura', ylabel='Count'>
```



Podemos observar en el histograma una asimetría hacia la derecha, lo que se conoce como una distribución positivamente sesgada. Esta asimetría se caracteriza por la presencia de barras más altas en el lado izquierdo y barras más bajas en el lado derecho, lo que sugiere la presencia de valores extremos o atípicos que aumentan el promedio y desplazan la distribución hacia la derecha. En otras palabras, la mayor concentración de valores se encuentra en el lado izquierdo, mientras que el lado derecho muestra una menor cantidad de valores.

En el siguiente boxplot, vemos más claramente los datos extremos para la `finura` que nos indicaba el histograma.

```
In [9]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['finura'], orient="h")

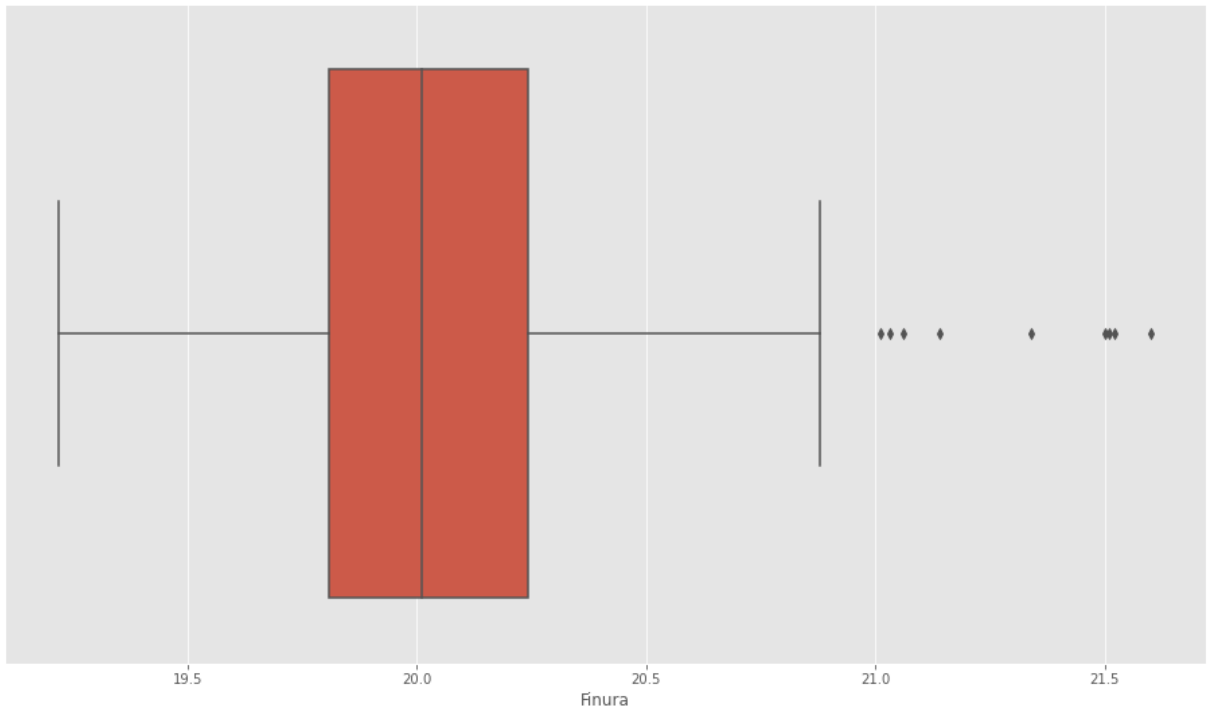
# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Finura')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=="-"]
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 19.22
Valor del bigote superior: 20.88



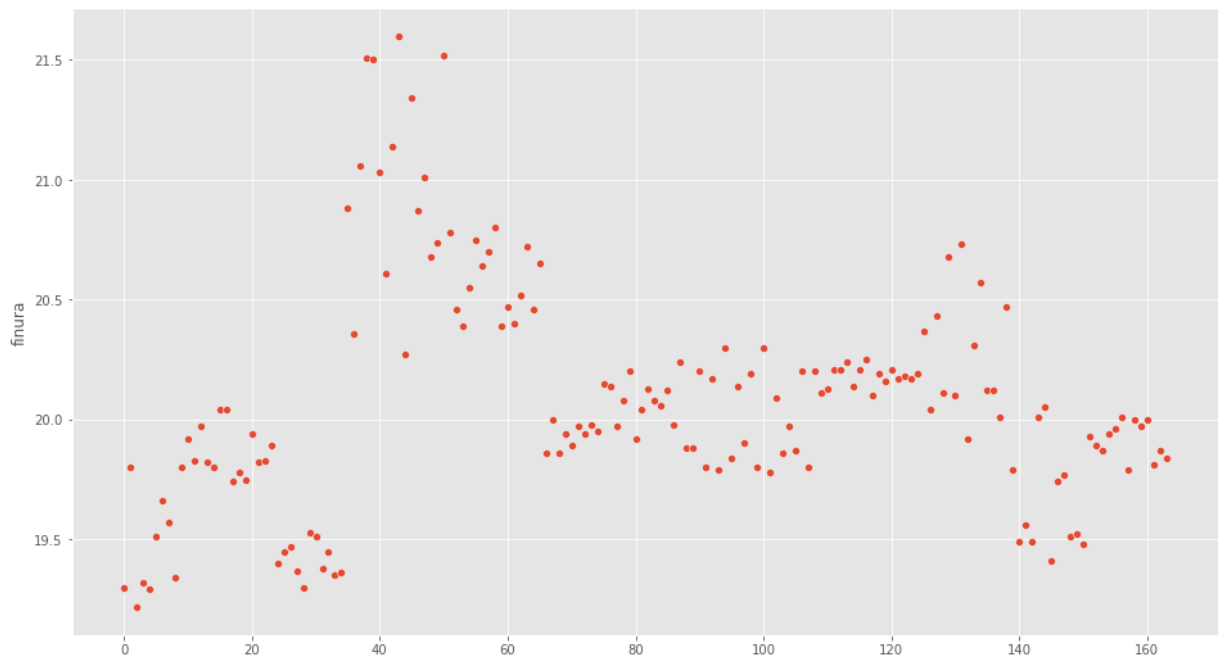
Explicación del gráfico de boxplot observado:

- Primer cuartil: el 25% de los valores son menores o iguales a 19.81 de finura.
- Segundo cuartil: el 50% de los valores son menores o iguales a 20.01 de finura.
- Tercer cuartil: el 75% de los valores son menores o iguales a 20.24 de finura.
- Mediana: la mediana es 20.01 de finura y corta la caja en dos lados desiguales, ya que la parte superior de la caja es levemente más larga que la inferior, lo que indica una asimetría positiva o sesgo a la derecha.
- Bigotes: el límite inferior es de 19.22 y el límite superior es de 20.88.
- Valores atípicos: hay 9 valores que superan el límite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el total de ovinos se distribuyen en un plano cartesiano.

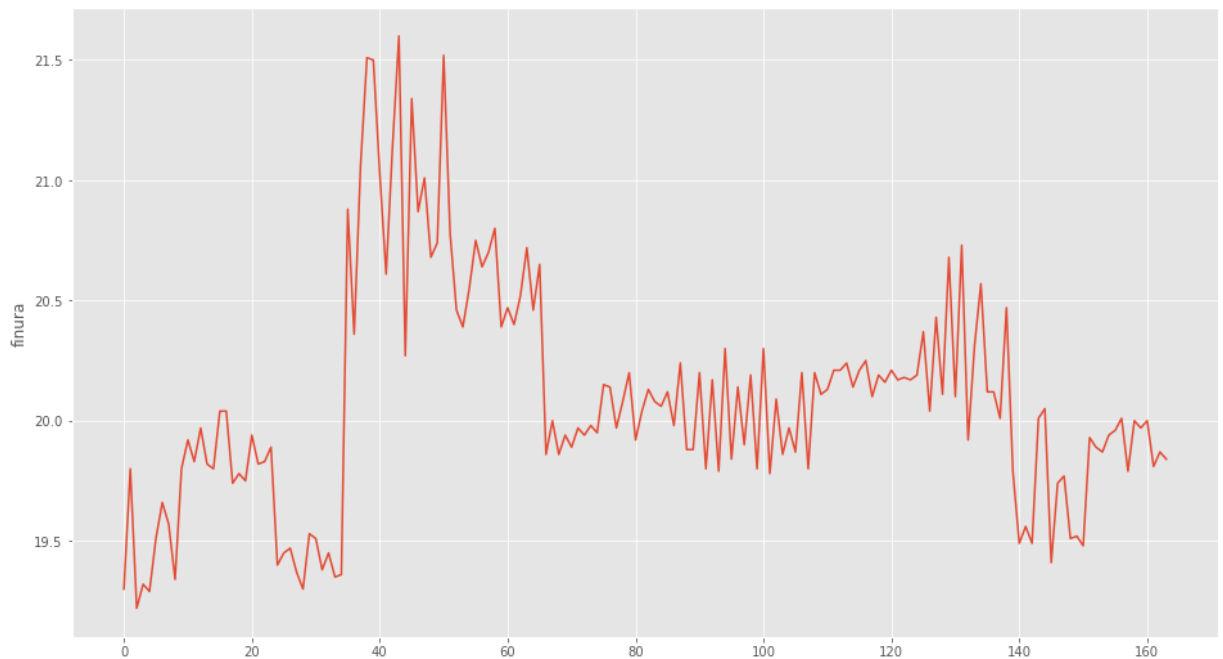
```
In [10]: sns.scatterplot(data=datos['finura'])
```

```
Out[10]: <AxesSubplot:ylabel='finura'>
```



```
In [11]: sns.lineplot(data=datos['finura'])
```

```
Out[11]: <AxesSubplot:ylabel='finura'>
```



En el gráfico de dispersión, se puede apreciar que no existe una clara relación entre las variables representadas en el plano cartesiano. La disposición de los puntos no sigue un patrón definido, como una línea recta, aunque se pueden identificar cuatro agrupaciones grandes que podrían ser relevantes. Por otro lado, el gráfico de línea muestra cómo la variación en la finura de la lana cambia de una observación a otra, pero no se observa una tendencia clara. Es decir, no se puede afirmar que haya un aumento o disminución constante en la finura de la lana en las diferentes observaciones.

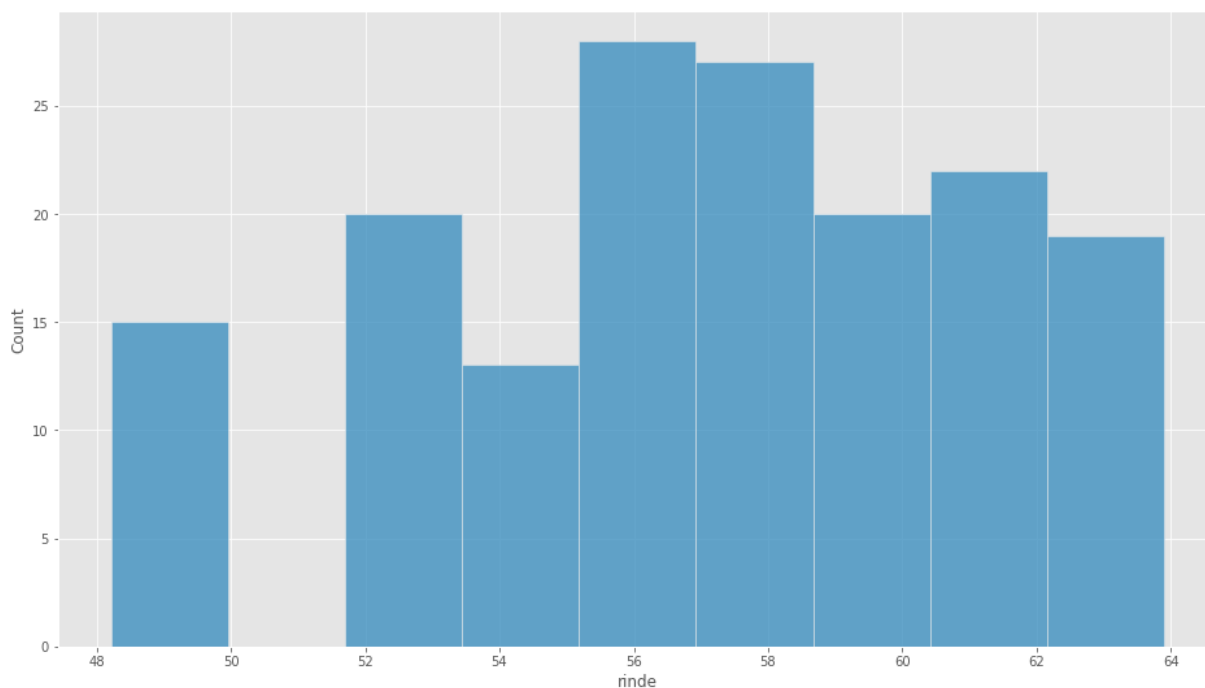
Es importante tener en cuenta que aunque no se detecte una tendencia lineal en los datos, podría existir una relación no lineal o compleja entre las variables que no es evidente en estos gráficos.

Rinde

El histograma presentado muestra que la variable `rinde` presenta una distribución asimétrica.


```
In [12]: sns.histplot(data=datos['rinde'])
```

```
Out[12]: <AxesSubplot:xlabel='rinde', ylabel='Count'>
```



Podemos hacer varias observaciones:

- La distribución tiene un pequeño sesgo hacia la izquierda.
- El valor máximo de la variable es 64 y el mínimo es 48.
- La mayoría de los valores se encuentran en el rango de 55 a 60 en el eje x, donde se concentran las barras más altas del histograma.
- En el extremo izquierdo del histograma hay una única barra correspondiente al intervalo 48-50 en el eje x, que tiene la menor frecuencia de observaciones.

En el siguiente boxplot, vemos más claramente la asimetría para el `rinde` que nos indicaba el histograma.

```
In [13]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['rinde'], orient="h")

# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Rinde')

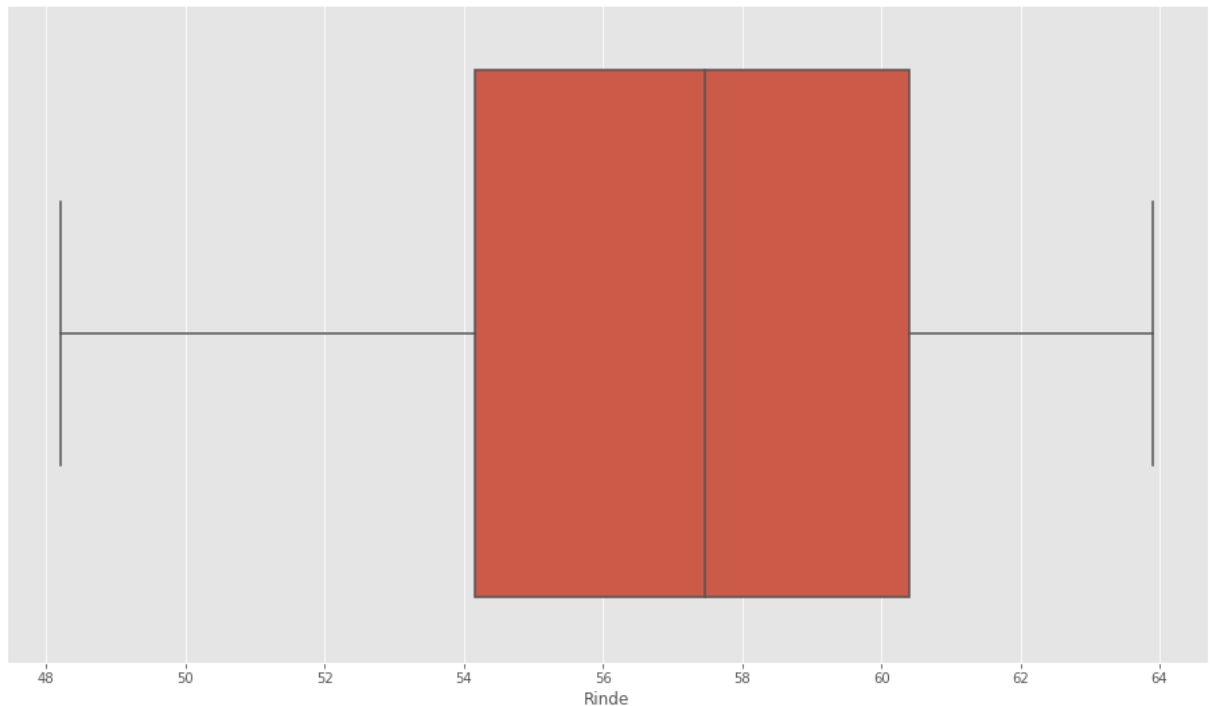
# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
```

```
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 48.21

Valor del bigote superior: 63.9



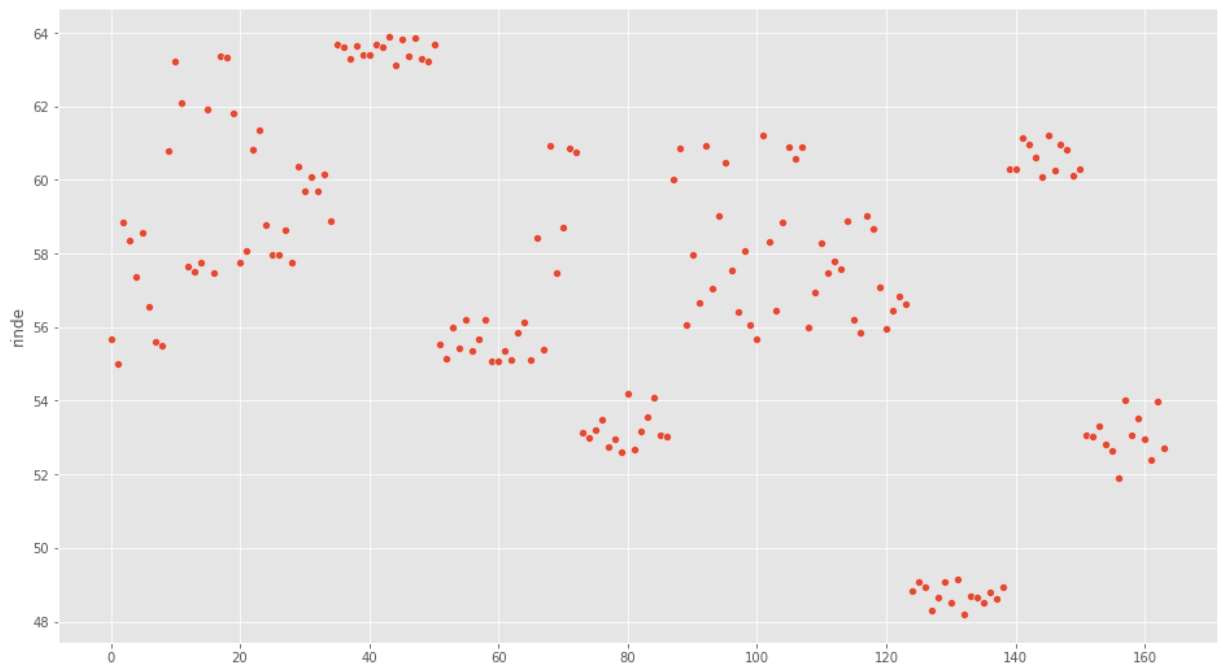
Explicación gráfico boxplot observado:

- Primer cuartil: el 25% de los valores son menores o igual a 54.16 de rinde.
- Segundo Cuartil: el 50% de los valores son menores o igual a 57.46 de rinde.
- Tercer cuartil: el 75% de los valores son menores o igual a 60.39 de rinde.
- Mediana: la mediana es 57.46 de rinde y corta la caja en dos lados desiguales, muestra una leve asimetría negativa o segada a la izquierda dado que la parte más larga de la caja es la parte inferior a la mediana, lo que indica que los valores más bajos son más frecuentes que los valores más altos.
- Bigotes: el limite inferior es de 48.21 y el limite superior es de 63.9.
- Valores atípicos: no hay valores que superen los limites.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el `rinde` se distribuyen en un plano cartesiano.

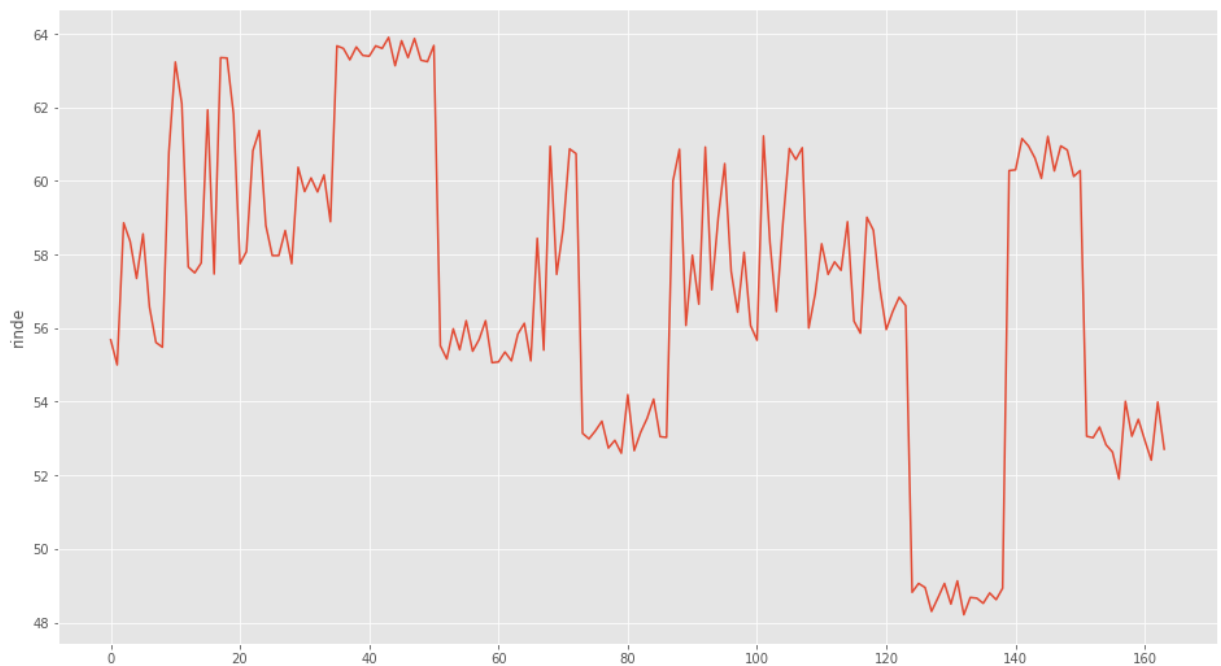
```
In [14]: sns.scatterplot(data=datos['rinde'])
```

```
Out[14]: <AxesSubplot:ylabel='rinde'>
```



```
In [15]: sns.lineplot(data=datos['rinde'])
```

```
Out[15]: <AxesSubplot:ylabel='rinde'>
```



Al examinar el diagrama de dispersión, se observa que no existe una relación evidente entre las variables representadas en el plano cartesiano. La ubicación de los puntos no sigue una pauta definida, como una línea recta, aunque se pueden identificar ocho grandes conjuntos que podrían ser importantes. En cambio, el gráfico de línea ilustra cómo el rinde de la lana varía de una observación a otra, pero no se distingue una tendencia definida. Por lo tanto, no se puede concluir que haya una tendencia constante al alza o a la baja en el rinde en las observaciones.

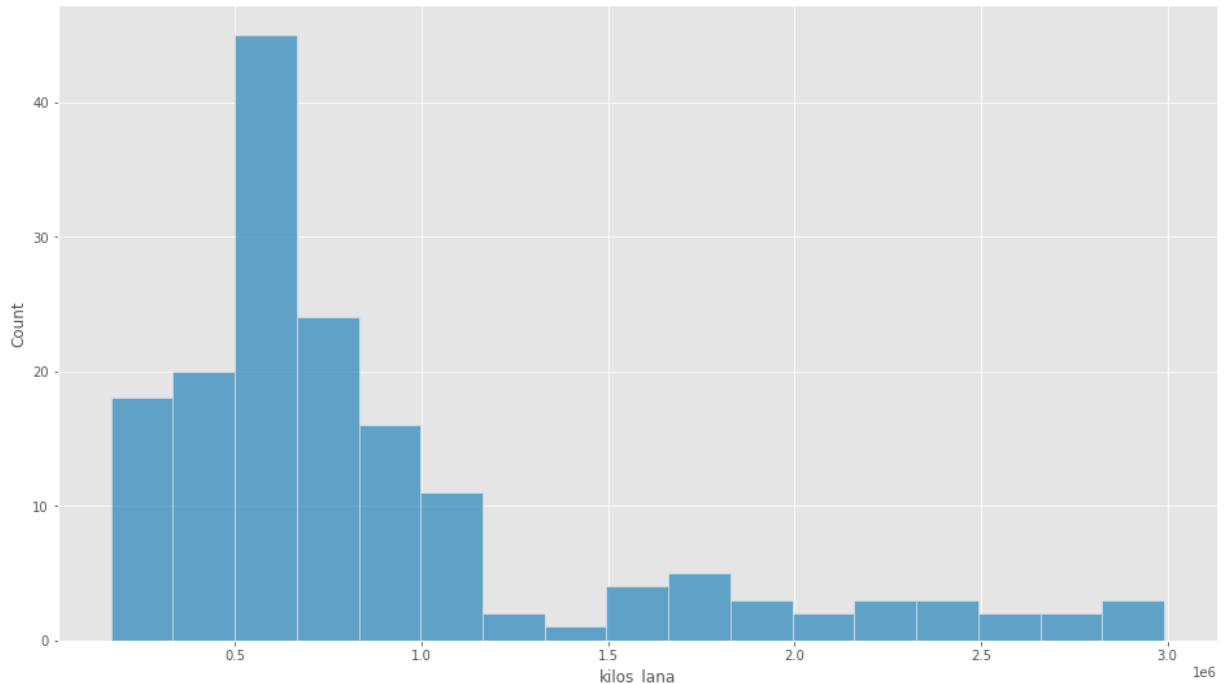
Es importante tener en cuenta que aunque no se detecte una tendencia lineal en los datos, podría existir una relación no lineal o compleja entre las variables que no es evidente en estos gráficos.

Kilogramos de lana

En el siguiente histograma vemos que los `kilos_lana` tiene una distribución asimétrica.

```
In [16]: sns.histplot(data=datos['kilos_lana'])
```

```
Out[16]: <AxesSubplot:xlabel='kilos_lana', ylabel='Count'>
```



En el histograma que se observa hay un total de 17 barras. De esas barras, 11 van desde el valor cero hasta cinco, y están todas situadas en el lado derecho de la distribución. Las seis barras restantes representan valores más altos, y se ubican en el lado izquierdo del gráfico. En este caso, la distribución está sesgada positivamente, lo que significa que hay una cola larga que se extiende hacia la derecha de la distribución, y que se deben considerar los valores atípicos en ese extremo. Esta asimetría se produce porque hay más valores en el lado izquierdo de la distribución que en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el `total de kilogramos de lana` que nos indicaba el histograma.

```
In [17]: ax = sns.boxplot(data=datos['kilos_lana'], orient="h")

# Eliminar el label del eje Y
ax.set_yticks([])

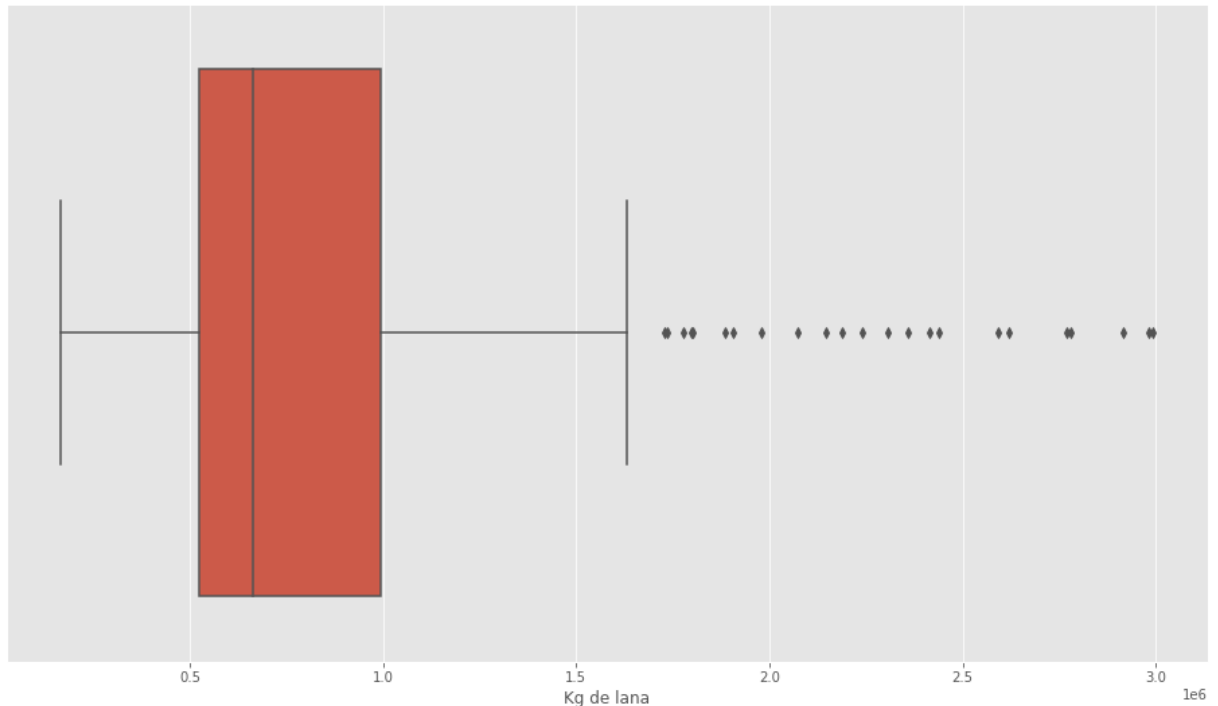
# Agregar un label al eje X
ax.set_xlabel('Kg de lana')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
```

```
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)
```

Valor del bigote inferior: 166901.0

Valor del bigote superior: 1630646.0

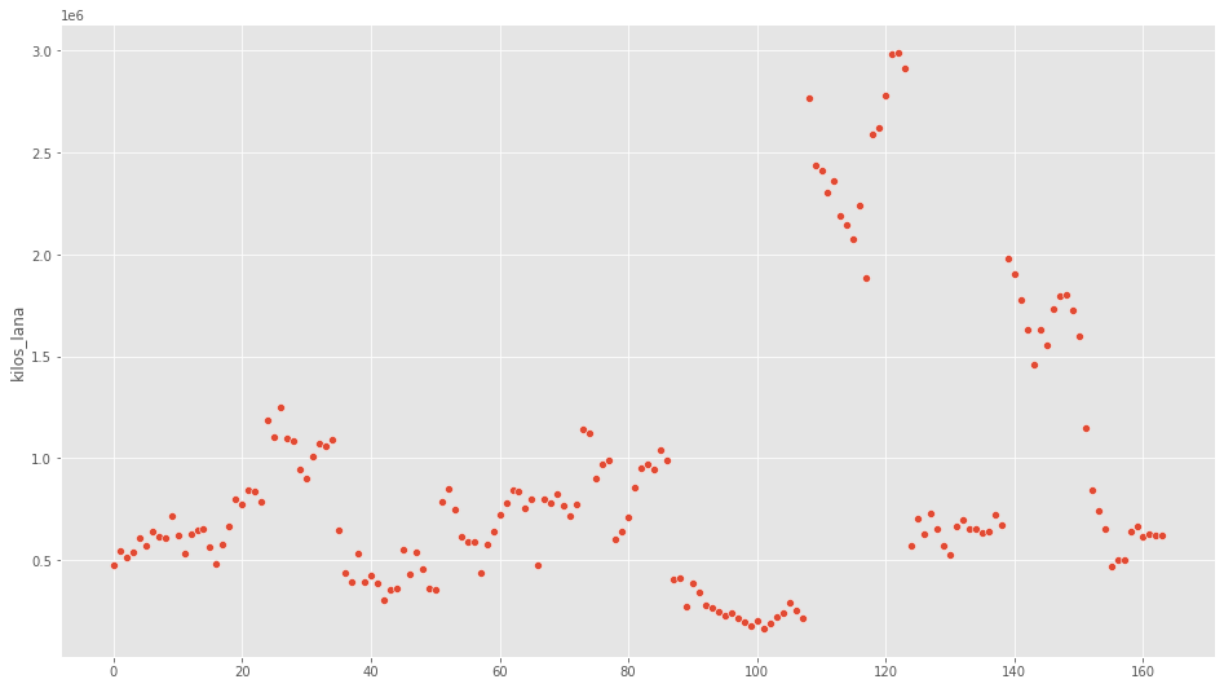


Se puede observar en el gráfico de boxplot que los datos están expresados en notación científica con una escala de millones, ya que los datos originales se transformaron de toneladas a kilogramos. El primer cuartil, que representa el 25% de los datos, tiene un valor de 523910,2 kg de lana o menos, mientras que el segundo cuartil, que representa el 50% de los datos, tiene un valor de 663384,1 kg de lana o menos. El tercer cuartil, que representa el 75% de los datos, tiene un valor de 994783,3 kg de lana o menos. La mediana, tiene un valor de 663384.05 kg de lana y muestra una asimetría positiva o sesgada hacia la derecha, ya que la parte superior de la caja es más larga que la inferior, lo que indica que los datos se concentran en la parte superior de la distribución. Los bigotes, que representan el rango de los datos, tienen un límite inferior de 166901 y un límite superior de 1630646. Además, se identifican 22 valores atípicos que superan el límite superior, lo que indica la presencia de varios valores extremos en la distribución de los datos.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el `total de lana` se distribuyen en un plano cartesiano.

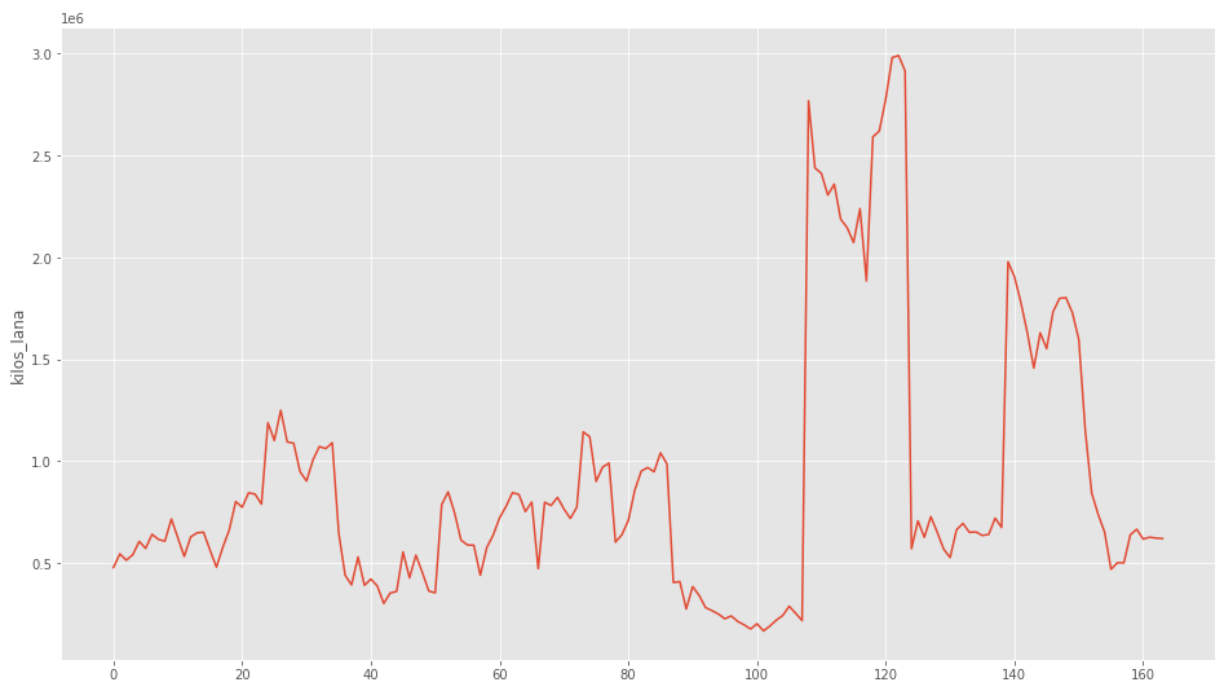
```
In [18]: sns.scatterplot(data=datos['kilos_lana'])
```

```
Out[18]: <AxesSubplot:ylabel='kilos_lana'>
```



```
In [19]: sns.lineplot(data=datos['kilos_lana'])
```

```
Out[19]: <AxesSubplot:ylabel='kilos_lana'>
```



Al analizar los datos representados en el gráfico de dispersión, no se puede identificar un patrón o tendencia clara entre las variables, aunque se pueden identificar seis grupos grandes que podrían ser relevantes.

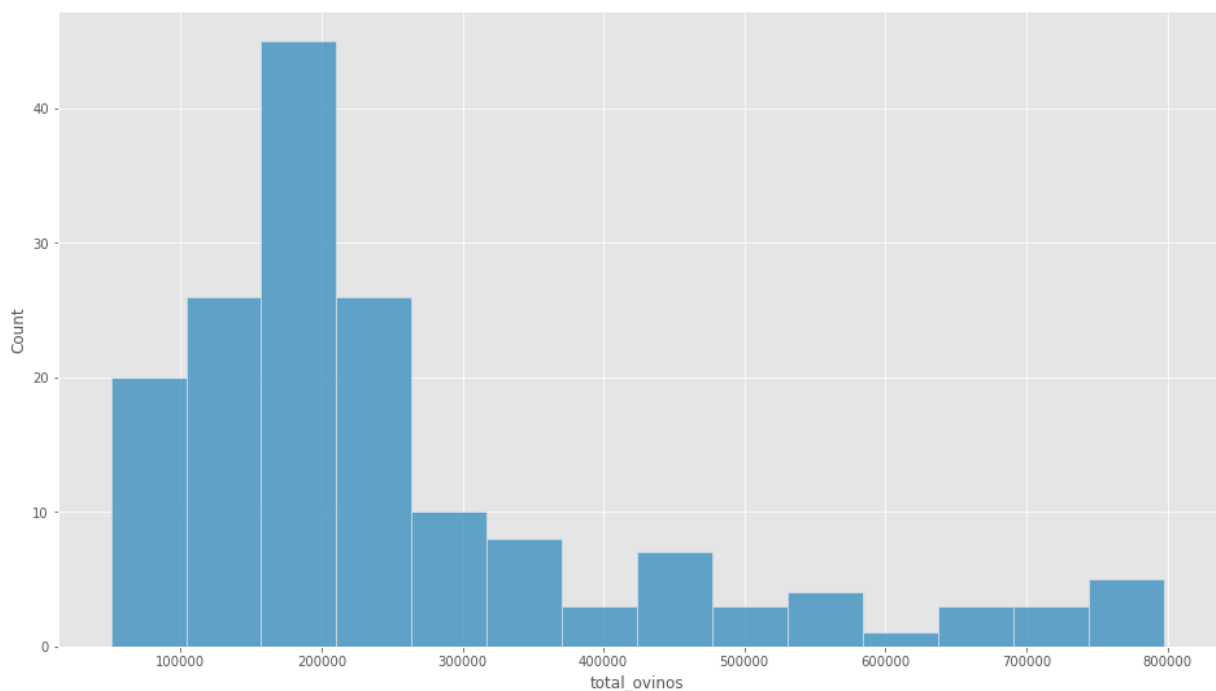
Al observar el gráfico de línea de los kilogramos de lana, se puede notar que hay una variabilidad alta en la producción, representada por los altos picos en los datos. Esto podría indicar una producción fluctuante o con una gran variabilidad. Pero como no se aprecia una tendencia clara, no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Total de ovinos

En el siguiente histograma vemos que el `total_ovinos` tiene una distribución asimétrica.

```
In [20]: sns.histplot(data=datos['total_ovinos'])
```

```
Out[20]: <AxesSubplot:xlabel='total_ovinos', ylabel='Count'>
```



El gráfico representa un histograma con una distribución asimétrica derecha, lo que se puede ver debido a la presencia de barras más altas en el lado izquierdo y barras más bajas en el lado derecho. Esta distribución también se conoce como una distribución positivamente sesgada. El sesgo positivo sugiere que la cola de la distribución se extiende hacia la derecha, lo que indica la presencia de valores extremos o atípicos en el lado derecho de la distribución que aumentan el promedio y empujan la distribución hacia la derecha.

En el siguiente boxplot, vemos más claramente los datos extremos para el **total de ovinos** que nos indicaba el histograma.

```
In [21]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['total_ovinos'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 850000, 50000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

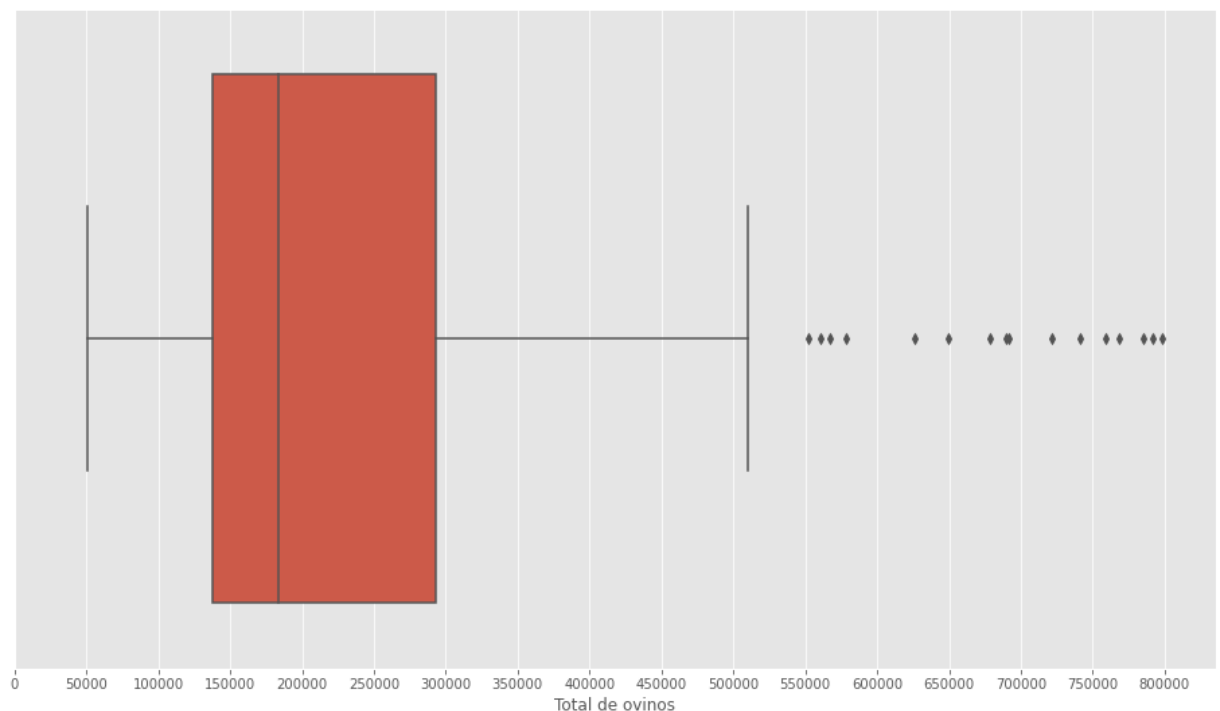
# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Total de ovinos')
```

```
# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 50513.0
Valor del bigote superior: 510093.0



El gráfico boxplot nos proporciona una representación visual de la distribución de los datos. Los valores del primer cuartil, el segundo cuartil (o mediana) y el tercer cuartil nos indican la posición relativa de los datos en la distribución. En este caso, el primer cuartil muestra que el 25% de los valores son menores o iguales a 138213 ovinos, mientras que el tercer cuartil nos indica que el 75% de los valores son menores o iguales a 293195 ovinos.

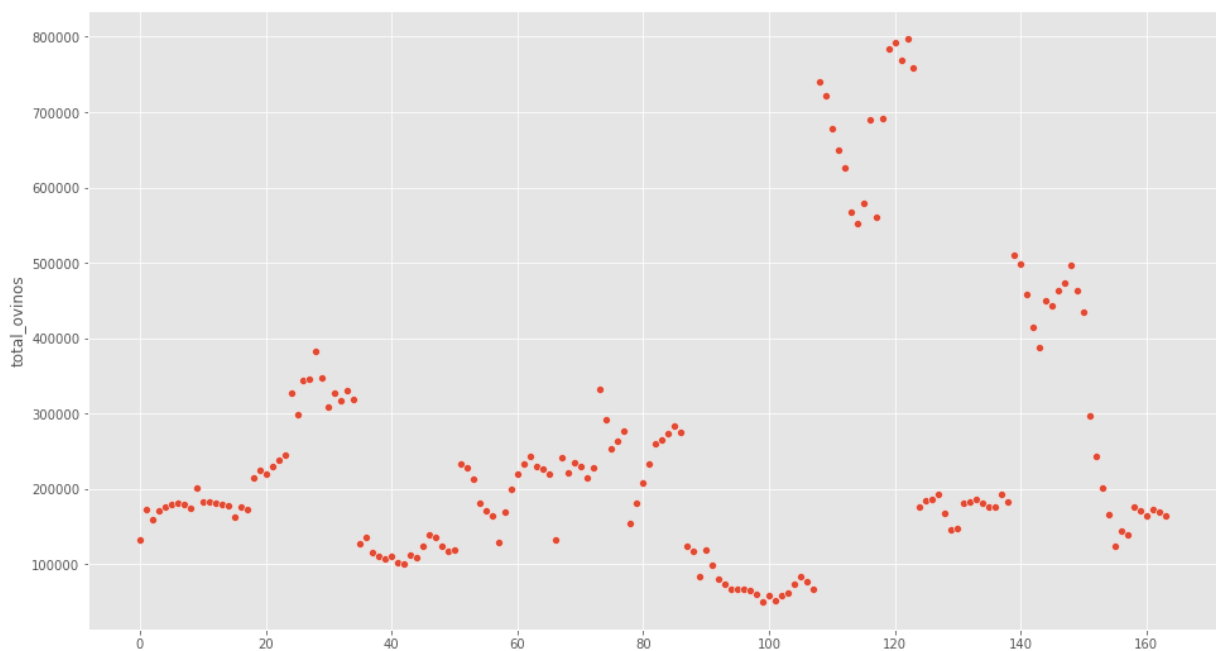
La mediana, que es 183361 ovinos, divide la caja en dos partes desiguales, mostrando una asimetría positiva o sesgada hacia la derecha, dado que la mayor parte de los datos se concentran en la parte superior de la distribución.

Los bigotes del gráfico indican los valores mínimo y máximo dentro de un rango aceptable, que se extiende desde el límite inferior de 50513 hasta el límite superior de 510093. Además, el gráfico muestra que hay 15 valores atípicos que superan el límite superior, lo que indica que hay algunas observaciones que están muy por encima de la mayoría de los datos y que podrían ser considerados como excepciones o datos extremos.

En los siguientes gráficos de dispersión y de línea, vemos como los datos para el **total de ovinos** se distribuyen en un plano cartesiano.

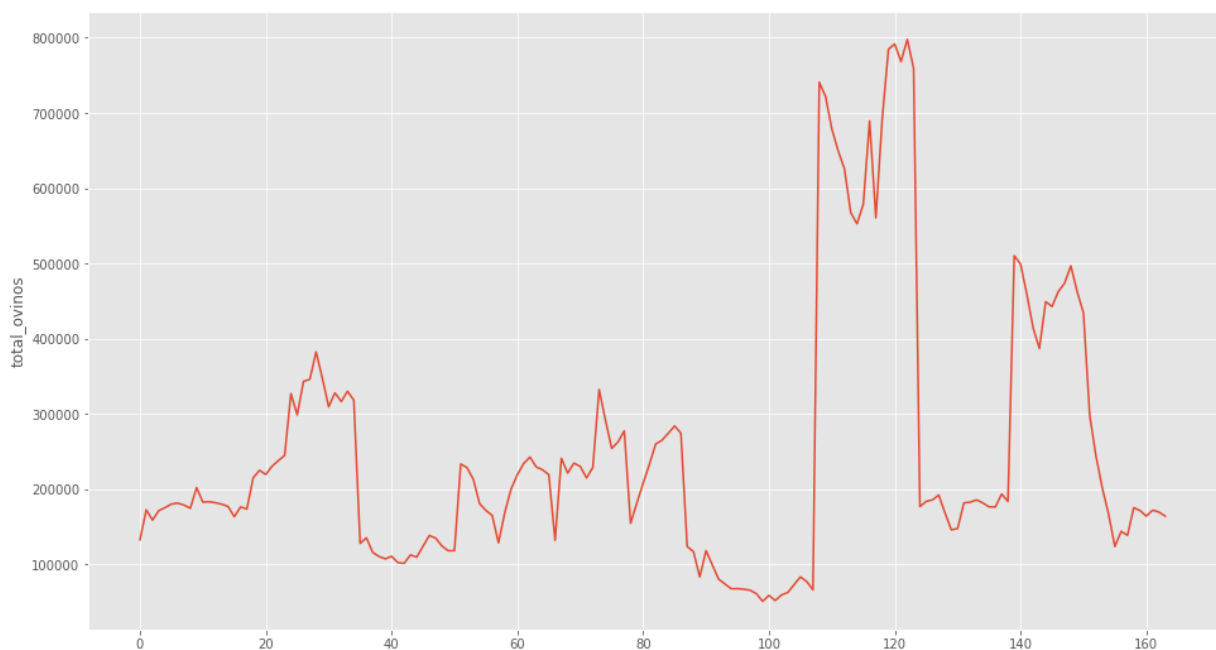

```
In [22]: sns.scatterplot(data=datos['total_ovinos'])
```

```
Out[22]: <AxesSubplot:ylabel='total_ovinos'>
```



```
In [23]: sns.lineplot(data=datos['total_ovinos'])
```

```
Out[23]: <AxesSubplot:ylabel='total_ovinos'>
```



Al examinar el diagrama de dispersión, no se puede distinguir ninguna tendencia o patrón definido entre las variables representadas, aunque se pueden identificar cuatro grupos grandes que podrían ser relevantes, no hay una disposición clara de los puntos, como una línea recta o un grupo compacto bien definido.

Por otro lado, al observar el gráfico de línea que representa la producción total de ovinos, se puede notar que existe una gran variabilidad en la producción, lo que se refleja en los picos elevados de los datos. Esto podría indicar que la población ovina tiene fluctuaciones o una variabilidad alta. No se observa ninguna

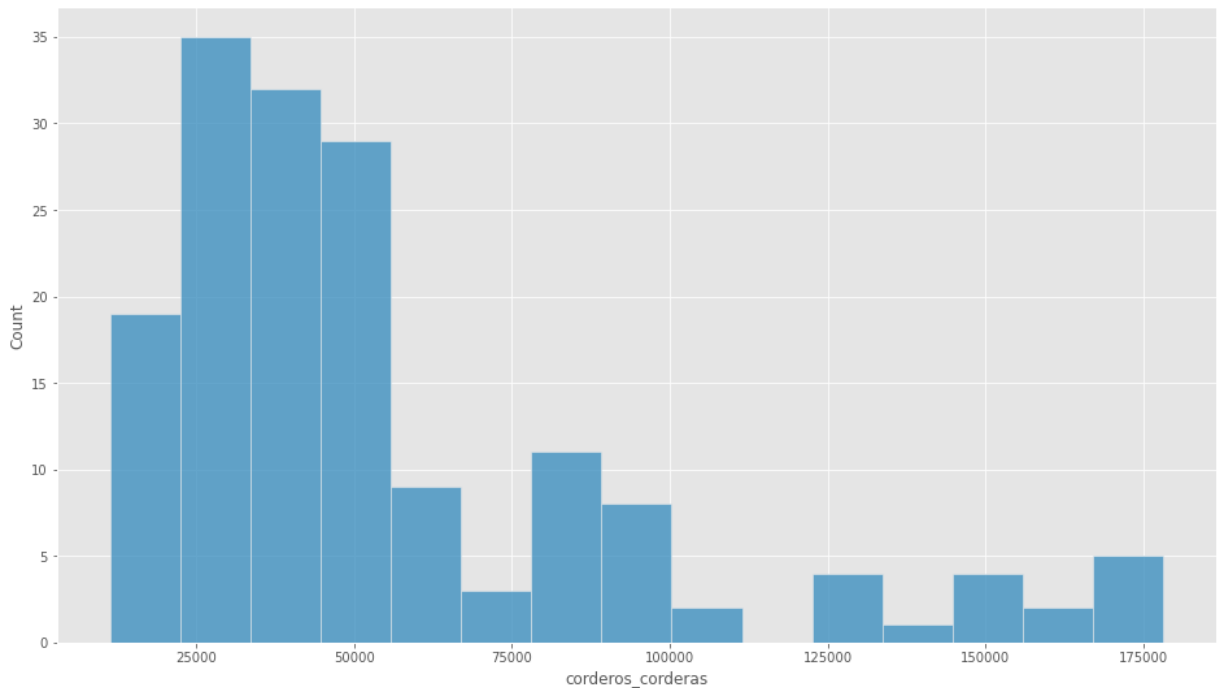
tendencia clara, por lo que no es posible afirmar que exista un aumento o disminución constante en las observaciones.

Corderos

En el siguiente histograma vemos que el `corderos_corderas` tiene una distribución asimétrica.

```
In [24]: sns.histplot(data=datos['corderos_corderas'])
```

```
Out[24]: <AxesSubplot:xlabel='corderos_corderas', ylabel='Count'>
```



En un histograma, una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho indica una distribución asimétrica derecha, también conocida como una distribución positivamente sesgada. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el `total de corderos` que nos indicaba el histograma.

```
In [25]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['corderos_corderas'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 185000, 10000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

# Eliminar el label del eje Y
```

```

ax.set_yticks([])

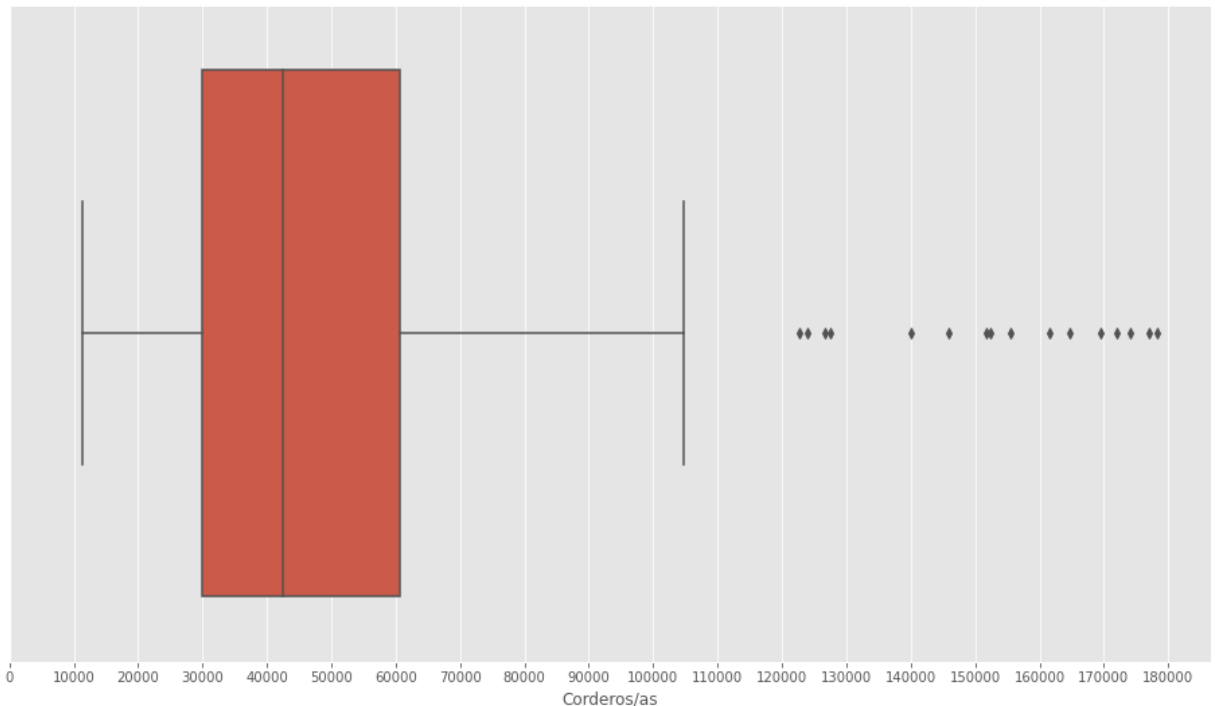
# Agregar un label al eje X
ax.set_xlabel('Corderos/as')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()

```

Valor del bigote inferior: 11300.0
 Valor del bigote superior: 104767.0



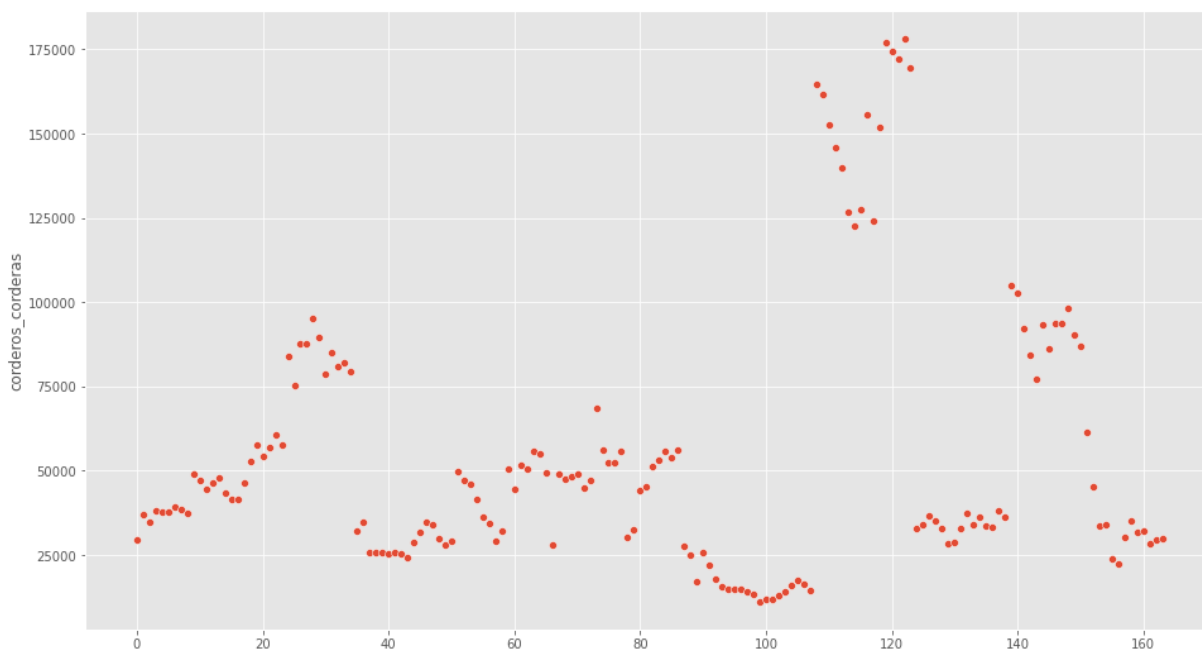
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 30004 corderos/as.
- Segundo Cuartil: el 50% de los valores son menores o igual a 42456.5 corderos/as.
- Tercer cuartil: el 75% de los valores son menores o igual a 60730 corderos/as.
- Mediana: la mediana es 42456.5 corderos/as y corta la caja en dos lados desiguales, por lo tanto muestra una asimetría positiva o segada a la derecha dado que la parte más larga de la caja es la parte superior a la mediana.
- Bigotes: el limite inferior es de 11300 y el limite superior es de 104767.
- Valores atípicos: hay 16 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de corderos** se distribuyen en un plano cartesiano.

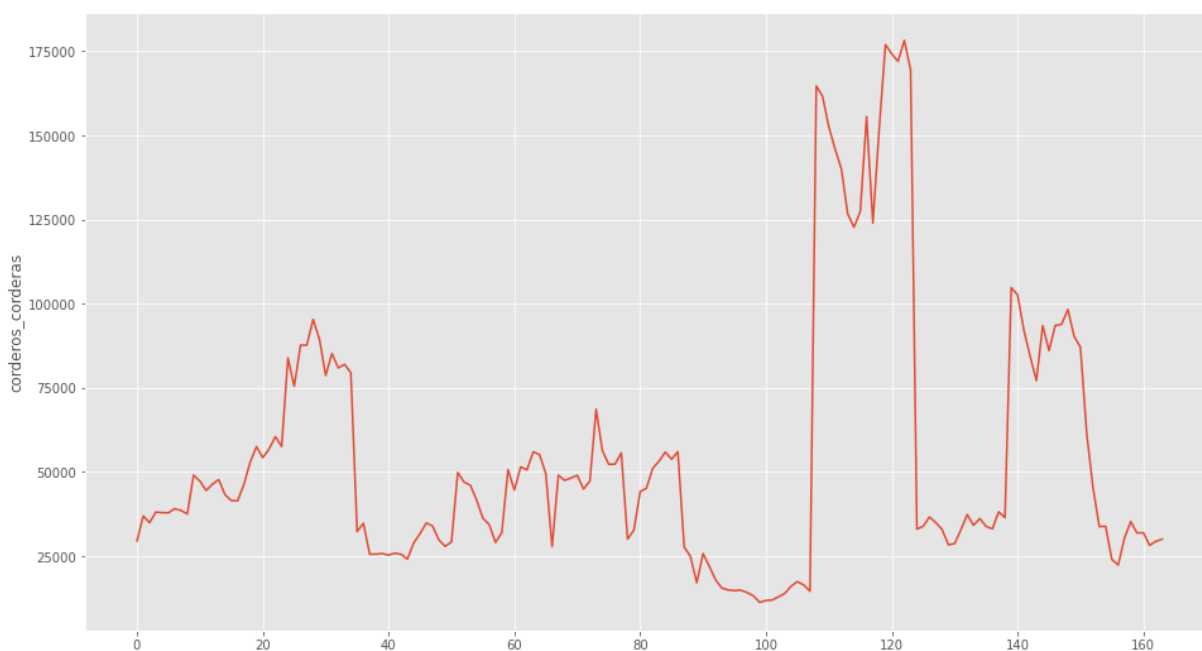
```
In [26]: sns.scatterplot(data=datos['corderos_corderas'])
```

```
Out[26]: <AxesSubplot:ylabel='corderos_corderas'>
```



```
In [27]: sns.lineplot(data=datos['corderos_corderas'])
```

```
Out[27]: <AxesSubplot:ylabel='corderos_corderas'>
```



Al analizar los datos representados en el gráfico de dispersión, no se puede identificar un patrón o tendencia clara entre las variables. Aunque se pueden distinguir cuatro grupos grandes que podrían ser relevantes, no hay una disposición definida de los puntos, como una línea recta o agrupaciones bien definidas.

En cuanto al gráfico de línea del total de corderos/as, se puede observar una alta variabilidad en la producción, representada por los altos picos en los datos. Sin embargo, la mayoría de los datos se encuentran entre 25000 y 100000 corderos/as, lo que podría indicar una fluctuación moderada. Pero no se

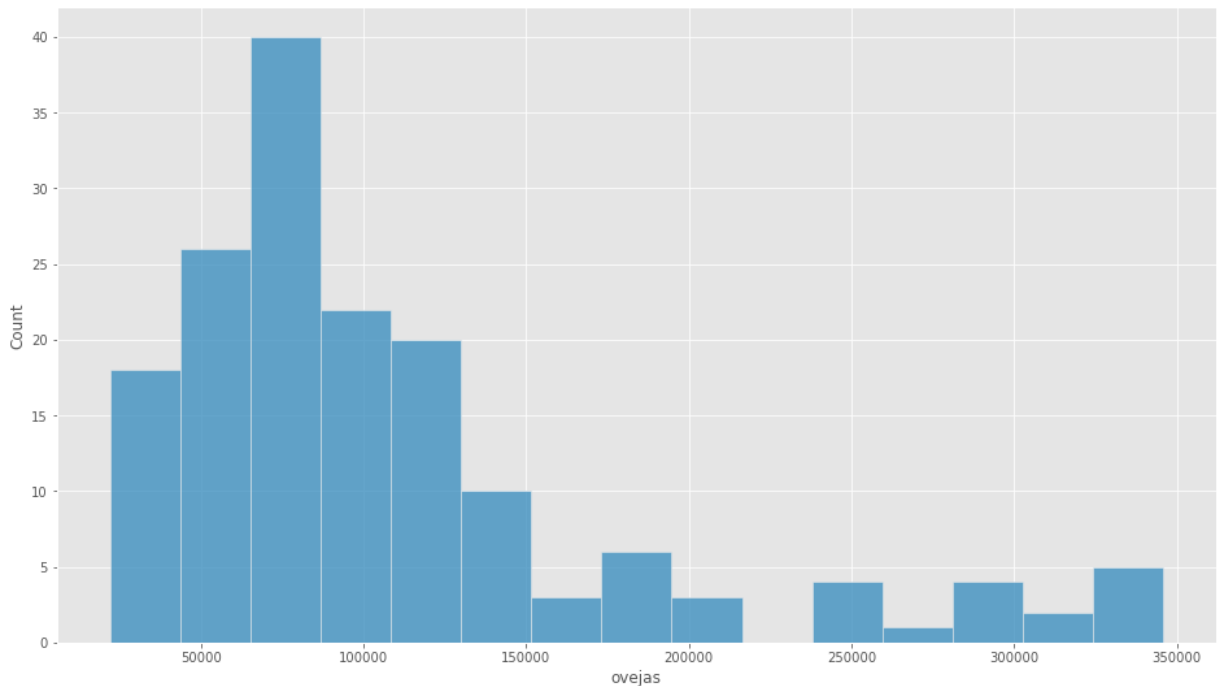
aprecia una tendencia clara, por lo que no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Ovejas

En el siguiente histograma vemos que la `ovejas` tienen una distribución asimétrica.

```
In [28]: sns.histplot(data=datos['ovejas'])
```

```
Out[28]: <AxesSubplot:xlabel='ovejas', ylabel='Count'>
```



El histograma muestra una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho, lo que indica una distribución asimétrica derecha. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el `total de ovejas` que nos indicaba el histograma.

```
In [29]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['ovejas'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 350000, 50000)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

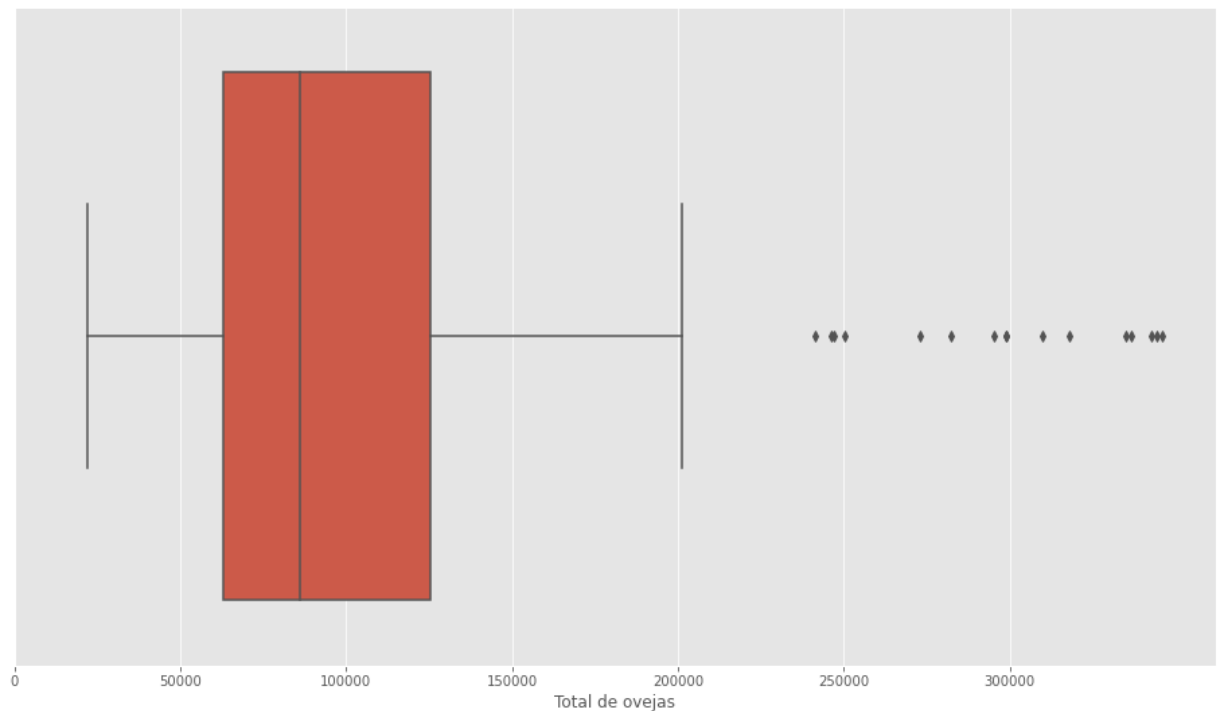
# Eliminar el label del eje Y
ax.set_yticks([])
```

```
# Agregar un label al eje X
ax.set_xlabel('Total de ovejas')

# Obtener valores de los bigotes
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 21913.0
Valor del bigote superior: 201450.0



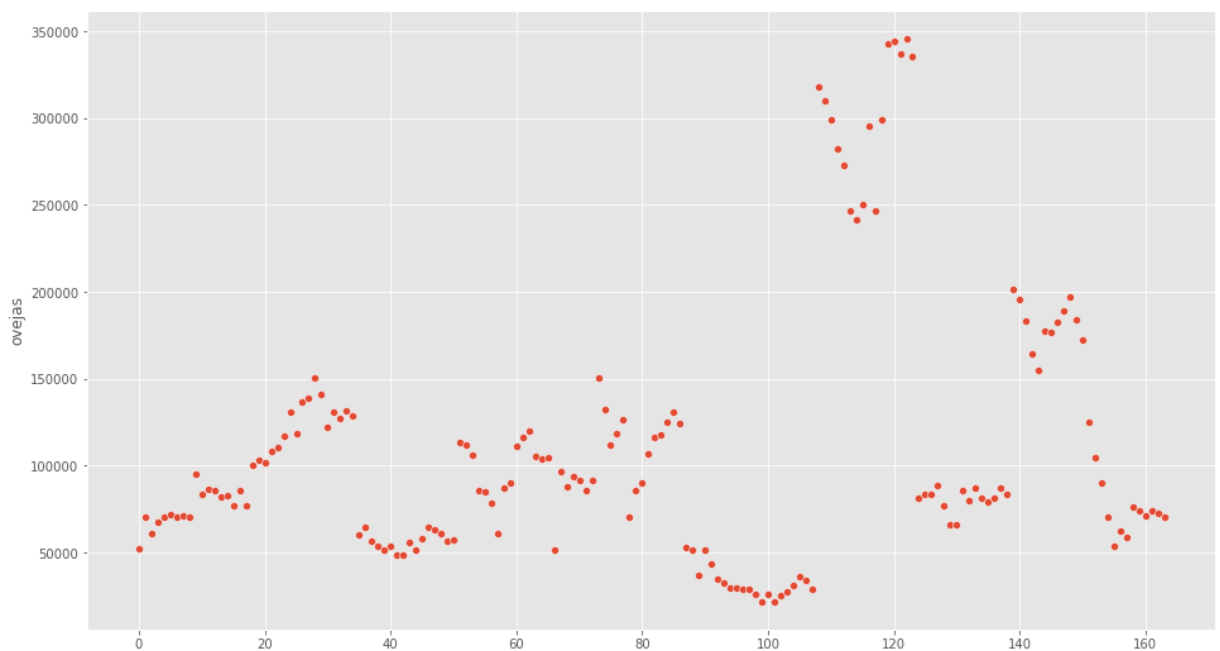
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 63116.50 ovejas.
- Segundo Cuartil: el 50% de los valores son menores o igual a 85986 ovejas.
- Tercer cuartil: el 75% de los valores son menores o igual a 125565.25 ovejas.
- Mediana: la mediana es 85986 de ovejas y corta la caja en dos lados desiguales, por lo tanto muestra una asimetría positiva o segada a la derecha dado que la parte más larga de la caja es la parte superior a la mediana.
- Bigotes: el limite inferior es de 21913 y el limite superior es de 201450.
- Valores atípicos: hay 15 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el **total de ovejas** se distribuyen en un plano cartesiano.

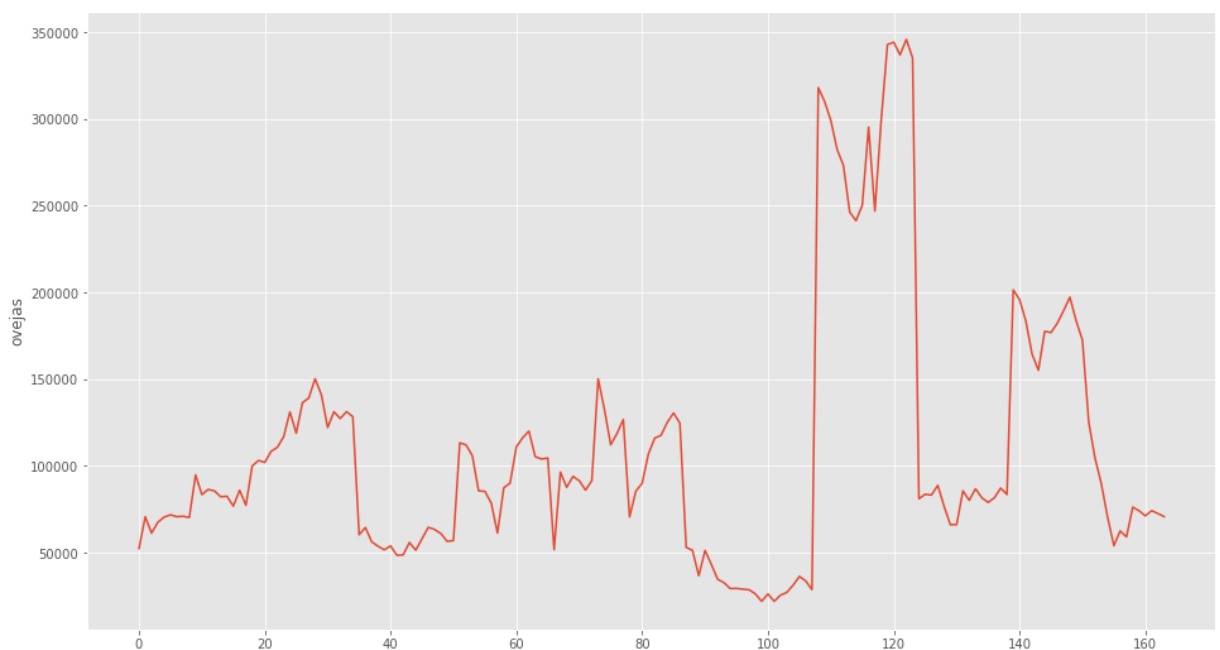
```
In [30]: sns.scatterplot(data=datos['ovejas'])
```

```
Out[30]: <AxesSubplot:ylabel='ovejas'>
```



```
In [31]: sns.lineplot(data=datos['ovejas'])
```

```
Out[31]: <AxesSubplot:ylabel='ovejas'>
```



Al analizar los datos representados en el gráfico de dispersión, no se puede identificar un patrón o tendencia clara entre las variables, aunque se pueden identificar alrededor de 7 u 8 grupos en los datos.

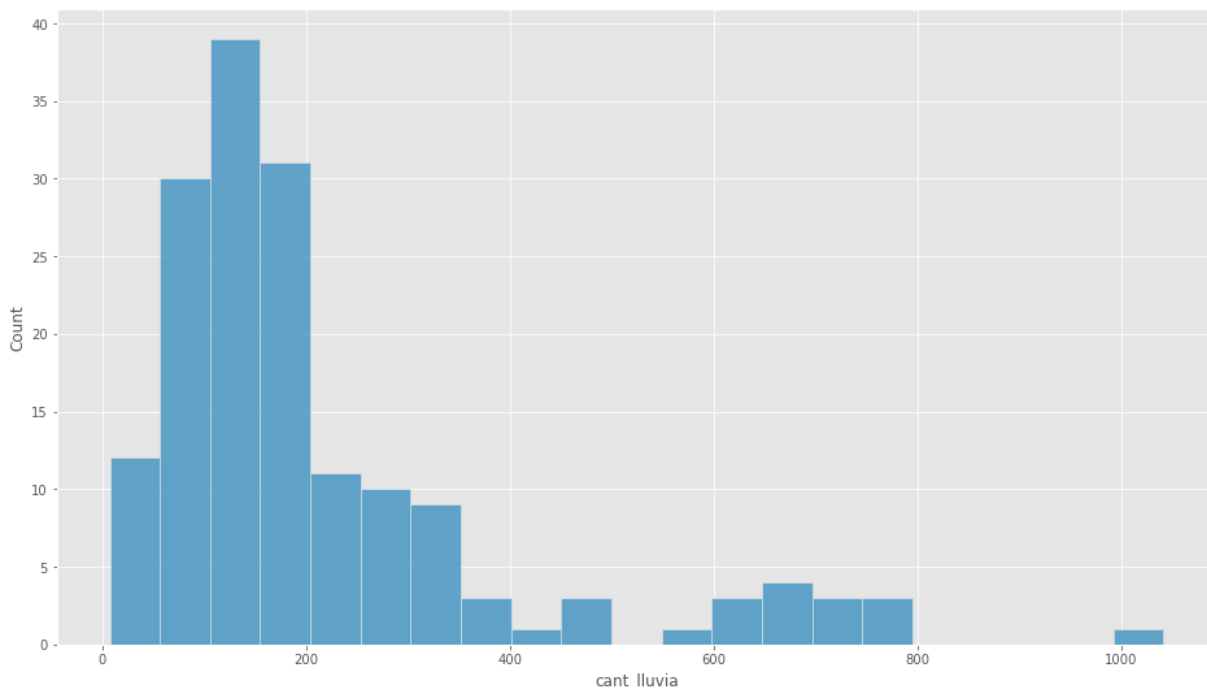
Por otro lado, al observar el gráfico de línea del total de ovejas, se puede notar que la mayoría de las observaciones se encuentran entre 50000 y 200000 ovejas. Sin embargo, no se aprecia una tendencia clara, por lo que no se puede afirmar que haya un aumento o disminución constante en las observaciones.

Lluvia

En el siguiente histograma vemos que la `cant_lluvia` tiene una distribución asimétrica.

```
In [32]: sns.histplot(data=datos['cant_lluvia'])
```

```
Out[32]: <AxesSubplot:xlabel='cant_lluvia', ylabel='Count'>
```



Se ve una distribución con barras más altas del lado izquierdo y barras más bajas del lado derecho lo que indica una distribución asimétrica derecha. Esto significa que la cola de la distribución se extiende hacia la derecha, lo que indica que hay valores extremos o atípicos en el lado derecho de la distribución que están aumentando el promedio y empujando la distribución hacia la derecha. En otras palabras, hay más valores en el lado izquierdo de la distribución y menos valores en el lado derecho.

En el siguiente boxplot, vemos más claramente los datos extremos para el **total de mm lluvia** que nos indicaba el histograma.

```
In [33]: # Crear el gráfico de caja horizontal
ax = sns.boxplot(data=datos['cant_lluvia'], orient="h")

# Definir los valores y etiquetas del eje X
valores_x = range(0, 1100, 100)
etiquetas_x = [str(valor) for valor in valores_x]
ax.set_xticks(valores_x)
ax.set_xticklabels(etiquetas_x)

# Eliminar el label del eje Y
ax.set_yticks([])

# Agregar un label al eje X
ax.set_xlabel('Mm de lluvia')

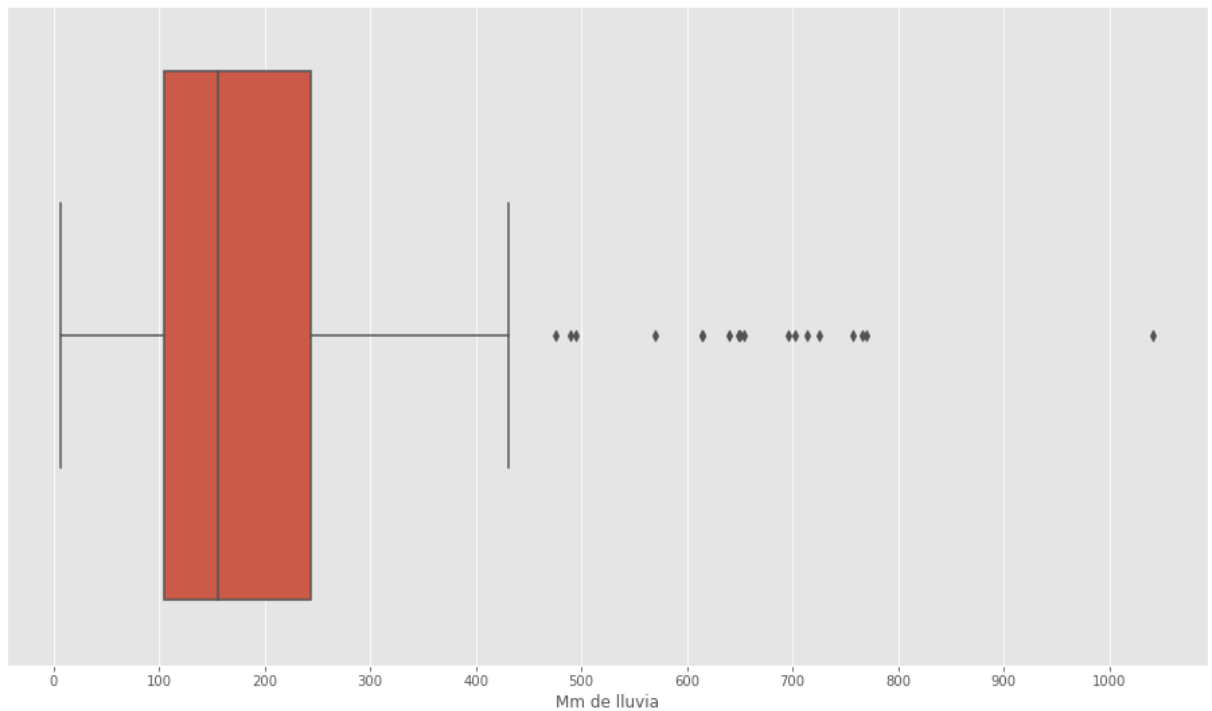
# Obtener valores de los bigotes
```



```
bigotes = [line.get_xdata() for line in ax.lines if
line.get_linestyle()=='-']
bigote_inferior = bigotes[0][1]
bigote_superior = bigotes[1][1]
print("Valor del bigote inferior:", bigote_inferior)
print("Valor del bigote superior:", bigote_superior)

plt.show()
```

Valor del bigote inferior: 7.1
Valor del bigote superior: 431.23



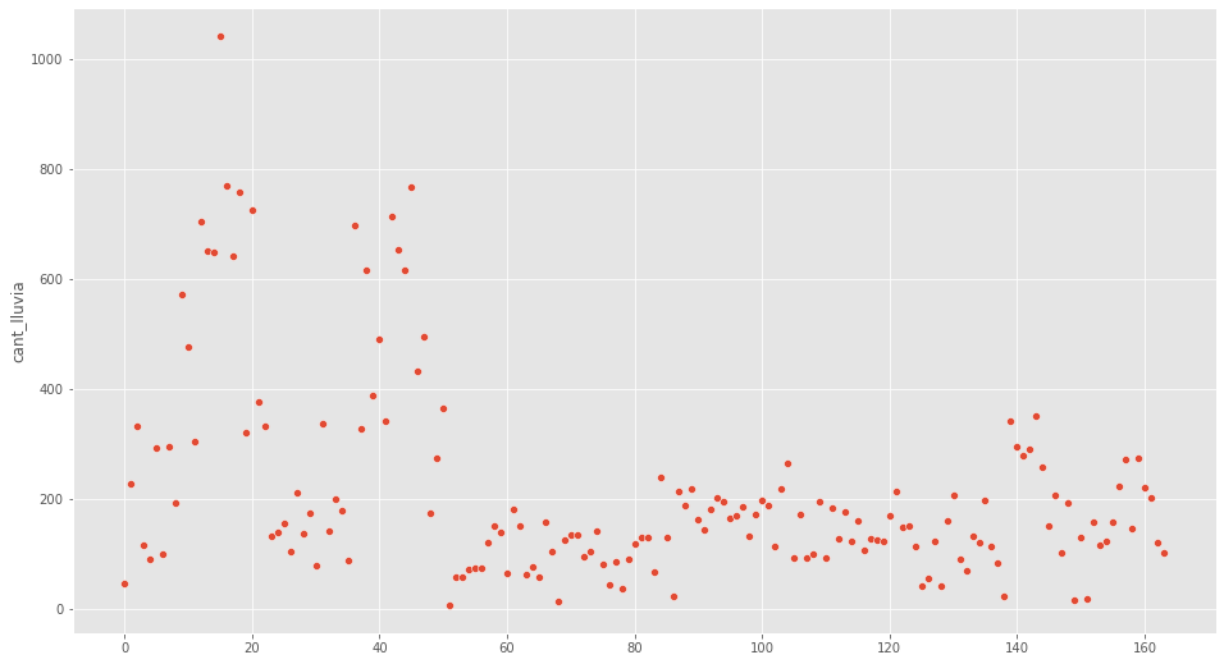
Explicación:

- Primer cuartil: el 25% de los valores son menores o igual a 104.22 mm de lluvia.
- Segundo Cuartil: el 50% de los valores son menores o igual a 155.80 mm de lluvia.
- Tercer cuartil: el 75% de los valores son menores o igual a 243.10 mm de lluvia.
- Mediana: la mediana es 155.80 mm de lluvia y corta la caja en dos lados desiguales, por lo tanto muestra una asimetría positiva o segada a la derecha dado que la parte más larga de la caja es la parte superior a la mediana.
- Bigotes: el limite inferior es de 7.1 mm y el limite superior es de 431.23 mm.
- Valores atípicos: hay 16 valores que superan el limite superior.

En los siguientes graficos de dispersión y de línea, vemos como los datos para el total de lluvia se distribuyen en un plano cartesiano.

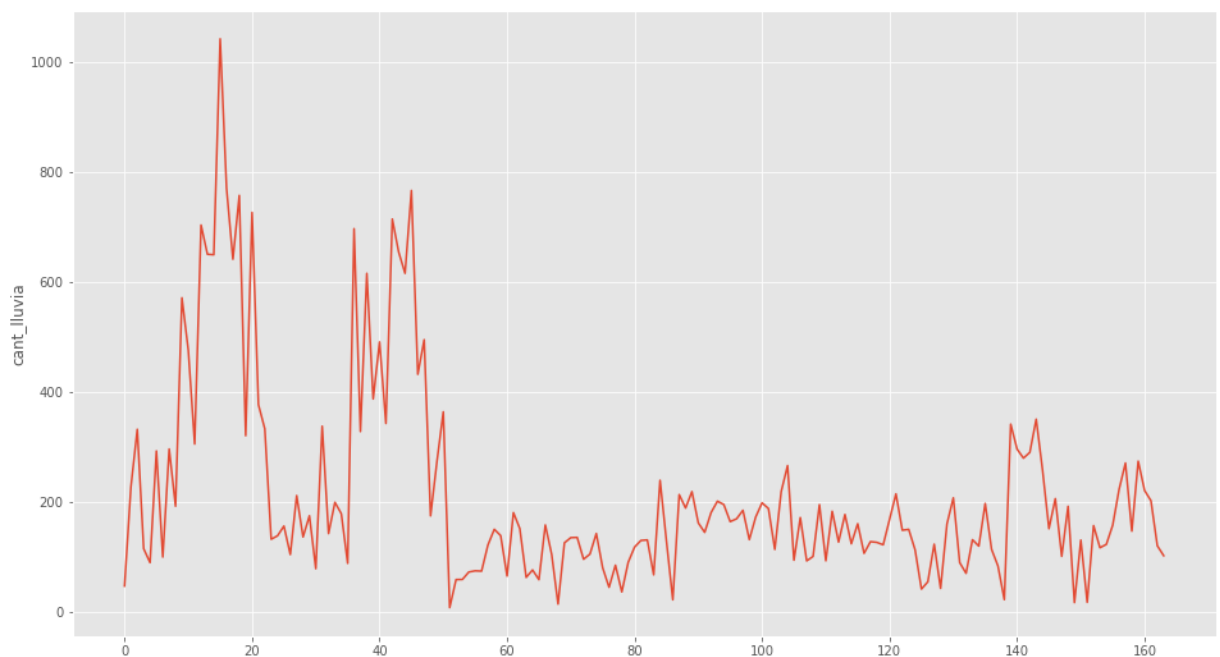
In [34]: `sns.scatterplot(data=datos['cant_lluvia'])`

Out[34]: `<AxesSubplot:ylabel='cant_lluvia'>`



```
In [35]: sns.lineplot(data=datos['cant_lluvia'])
```

```
Out[35]: <AxesSubplot:ylabel='cant_lluvia'>
```



Tras analizar los gráficos de dispersión y de línea, no se puede identificar un patrón o tendencia clara en los datos de lluvia en Chubut. En el gráfico de dispersión, no se observa ningún grupo o agrupación definida de los puntos. Por su parte, en el gráfico de línea se aprecian varios picos altos, lo que indica que existe una variabilidad significativa en los valores de lluvia. Sin embargo, se puede establecer que la precipitación en Chubut varía en un rango de 0 a 600 mm, tal como se refleja en ambos gráficos.

Como observación final de las variables independientes vemos que en los gráficos de dispersión y de línea, los datos de lana y de existencias son muy parecidos, casi que tienen la misma forma.

Variables combinadas

A continuación se va a mejorar la paleta de colores para que sea más fácil distinguir entre ellos. Para lograrlo, crearemos una paleta de colores personalizada que constará de 15 colores profundos ("deep") y

la estableceremos como la paleta por defecto para los gráficos de Seaborn.

```
In [5]: mi_paleta = sns.color_palette("deep", 15)
sns.set_palette(mi_paleta)
```

A continuación, se crearán dos dataframes más pequeños divididos por los departamentos que pertenecen a la cordillera y los que no, con el fin de facilitar el proceso de gráficos conjuntos y posiblemente obtener información adicional a partir de la división de los datos.

```
In [6]: departamentos_cordillera =
datos[datos['departamento'].isin(['CUSHAMEN', 'RIO SENGUERR',
'TEHUELCHES', 'LANGUIÑEO', 'FUTALEUFU'])]
departamentos_meseta =
datos[~datos['departamento'].isin(['CUSHAMEN', 'RIO SENGUERR',
'TEHUELCHES', 'LANGUIÑEO', 'FUTALEUFU'])]
```

Graficos

Diagrama de dispersión

El comando `sns.scatterplot(data=datos, x='variable_X', y='variable_Y', hue='departamento')` va a generar un gráfico de dispersión donde se muestra la relación entre dos variables: la `variable_X` en el eje X y la `variable_Y` en el eje Y. Además, se utiliza el argumento `hue` para agregar una tercera dimensión de información, que en este caso es el departamento. Cada punto en el gráfico representa una observación del dataset, donde la posición en el eje X corresponde al valor de `'variable_X'` de la observación, la posición en el eje Y corresponde al valor de `'variable_Y'` en esa observación y el color corresponde al departamento.

Diagrama de línea

El código `sns.lineplot(data=datos, x='variable_X', y='variable_Y')` genera un gráfico de línea que representa la evolución de la `'variable_X'` a lo largo de `'variable_Y'`. Cada punto en el gráfico representa la `'variable_X'` registrada para un valor específico de `'variable_Y'`, la línea que los une representa la tendencia general de la variable a lo largo de `'variable_Y'`, es decir, la evolución de `'variable_X'` a medida que cambia `'variable_Y'`, y el borde de color más claro indica el intervalo de confianza. Esto significa que el gráfico muestra un valor promedio de la variable `'variable_X'` para cada valor de `'variable_Y'`, junto con un intervalo de confianza alrededor de ese valor promedio. El intervalo de confianza indica la incertidumbre en la estimación del valor promedio y se calcula comúnmente como la desviación estándar de los datos dividido por la raíz cuadrada del número de observaciones.

Como el intervalo de confianza se calcula en función del número de observaciones que hay en los datos, en cada uno de los gráficos que vamos a ver a continuación vemos que comienzan en el año 2000 y el intervalo de confianza se muestra a partir del 2006, esto es porque hay pocas observaciones en los años anteriores a 2006, lo que hace que el intervalo de confianza no sea significativo o no pueda ser calculado con precisión.

Diagrama de barras

El comando `sns.barplot(data=datos, x='fecha', y='variable_Y')` genera un gráfico de barras que muestra la cantidad total de 'variable_Y' para cada año en el conjunto de datos. Cada año se mostraría en el eje x y la cantidad total 'variable_Y' correspondiente se mostraría en el eje y. La altura de cada barra representaría la cantidad la 'variable_Y' para una fecha determinada. La línea finita que se encuentra en la parte superior de cada barra representa el intervalo de confianza del 95% para la media de cada grupo de datos. Es decir, es una estimación de la incertidumbre en la medida de la media para cada grupo. La altura de la línea muestra el rango de valores dentro del cual se espera que la media real esté con una confianza del 95%.

Fecha con otras variables

Existencias ovinas

Con el fin de manejar adecuadamente la información de las fechas, se procedió a ordenar el conjunto de datos según este atributo y se guardó en un nuevo DataFrame. Posteriormente, se agregó una nueva columna que contiene únicamente el año, lo que permitirá trabajar de manera más eficiente con esta información en caso de ser necesario.

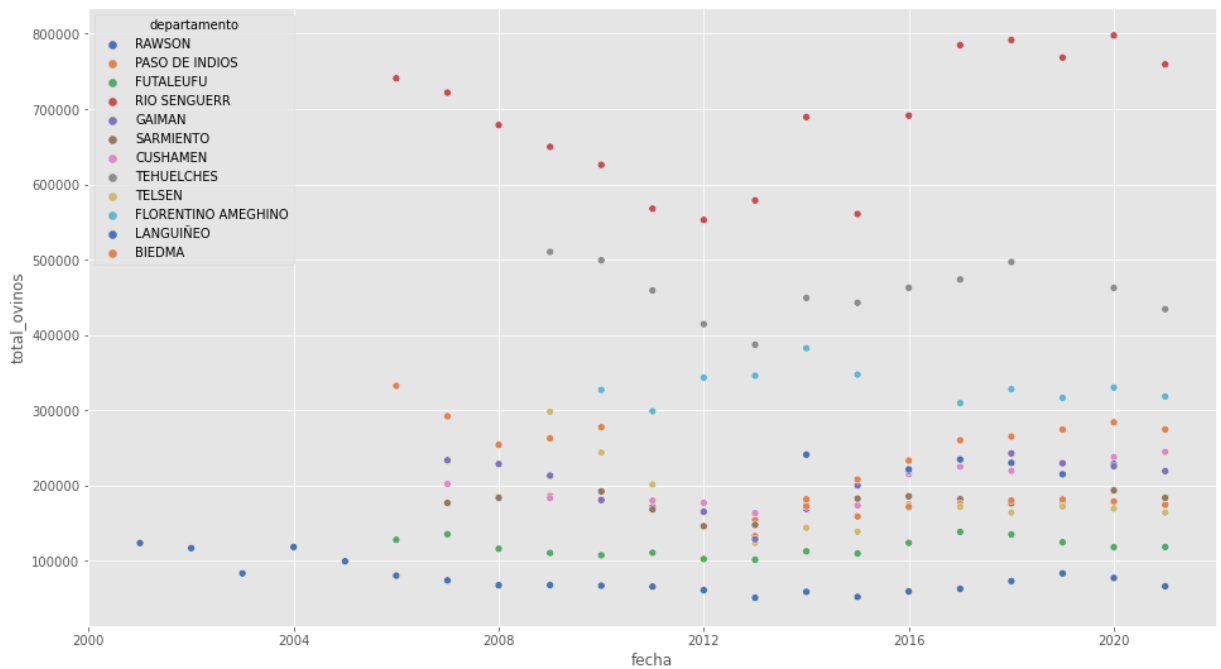
```
In [37]: datos_ordenados_fecha = datos.sort_values('fecha')
datos_ordenados_fecha['anio'] =
datos_ordenados_fecha['fecha'].dt.strftime('%Y')
```

Total de ovinos

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo `total_ovinos` para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de ovinos a lo largo del tiempo y cómo se distribuyen en los diferentes años.

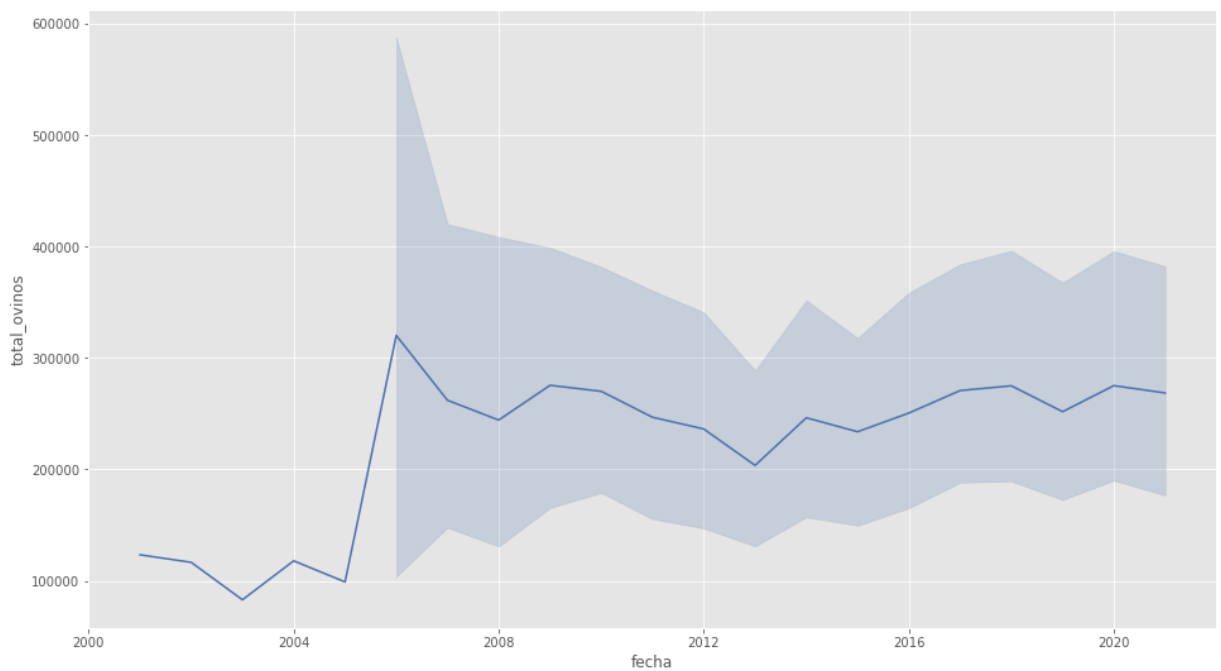
```
In [38]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',
y='total_ovinos', hue='departamento')
```

```
Out[38]: <AxesSubplot:xlabel='fecha', ylabel='total_ovinos'>
```



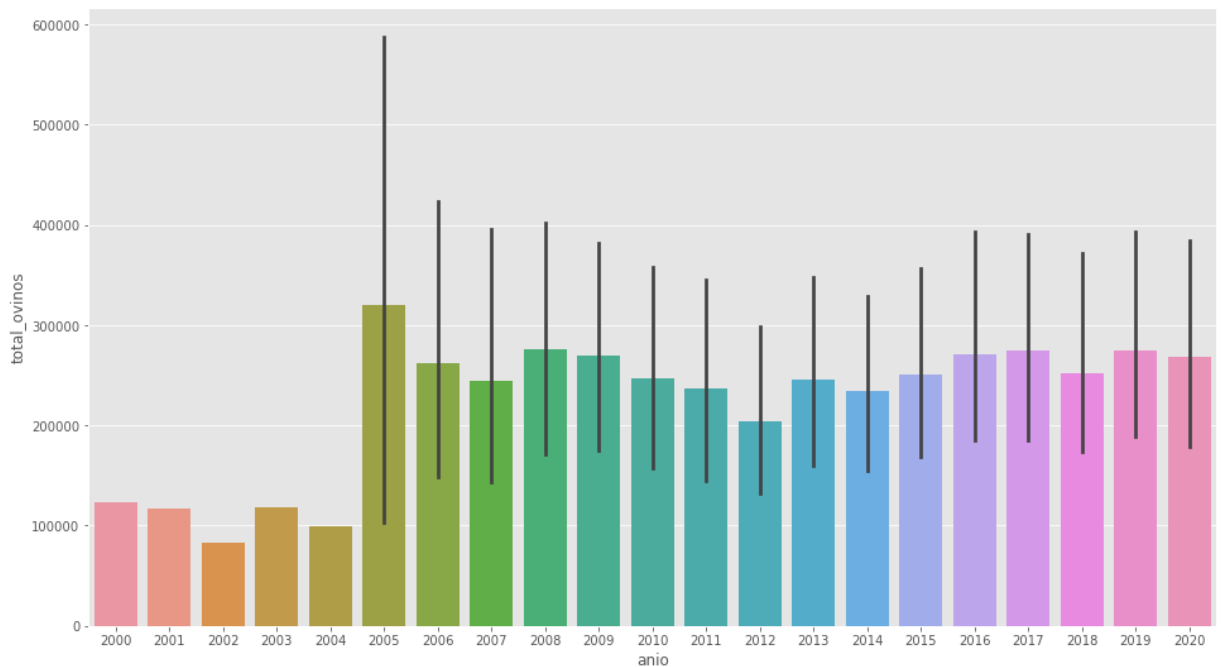
```
In [39]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',
y='total_ovinos')
```

```
Out[39]: <AxesSubplot:xlabel='fecha', ylabel='total_ovinos'>
```



```
In [40]: sns.barplot(data=datos_ordenados_fecha, x='anio',
y='total_ovinos')
```

```
Out[40]: <AxesSubplot:xlabel='anio', ylabel='total_ovinos'>
```



Estos tres gráficos nos brindan información sobre la variación del total de ovinos a lo largo del tiempo. El gráfico de dispersión muestra cada departamento como un punto en el plano cartesiano, con su posición en el eje X e Y representando la cantidad de ovinos en un año determinado. Cada punto está coloreado según el departamento al que pertenece, lo que nos permite comparar visualmente la cantidad de ovinos entre departamentos. Podemos notar que los departamentos de Rawson y Futaleufú mantienen una cantidad de ovinos relativamente constante, que no supera los 150 mil, mientras que los departamentos de Tehuelches y Río Senger presentan valores más altos y variables.

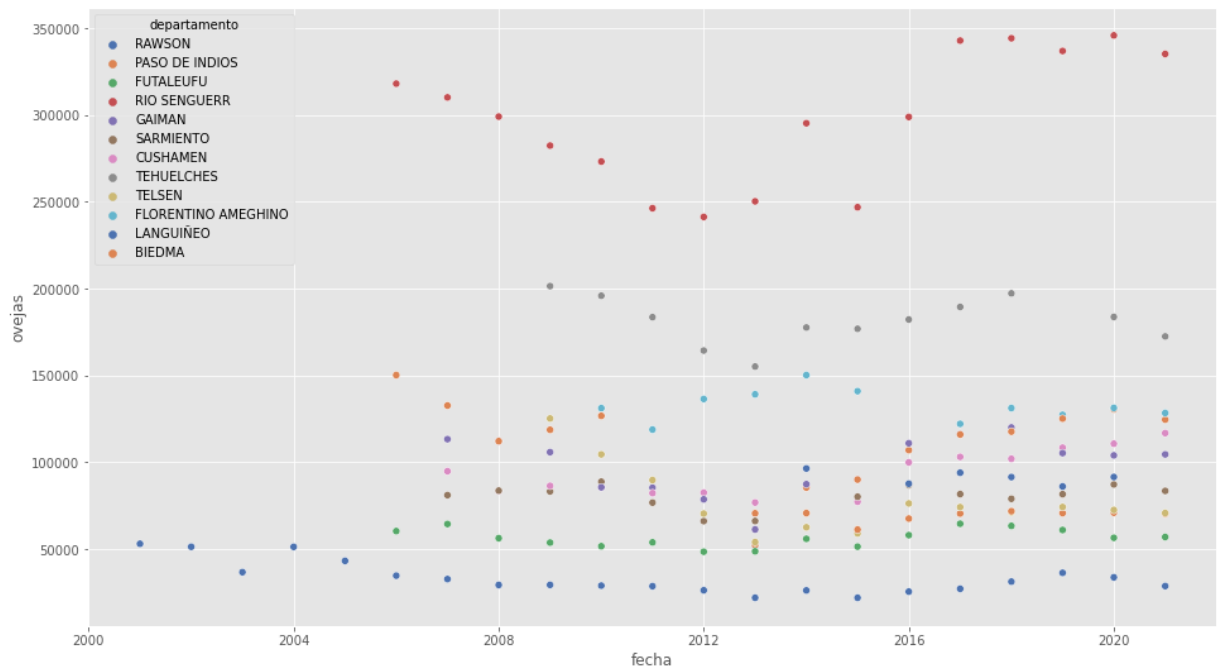
Entonces cuando los totales de ovinos superan los 400 mil, los puntos están muy dispersos, lo que puede indicar una gran variabilidad en la producción de ovinos a lo largo de los años. Por otro lado, cuando los totales de ovinos son inferiores a 400 mil, las filas de puntos son más uniformes, lo que puede indicar una producción más estable a lo largo de los años. Este patrón también se puede observar en el gráfico de línea y en el de barras, que incluyen intervalos de confianza para mostrar la variabilidad de los datos.

Ovejas

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **ovejas** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de ovejas a lo largo del tiempo y cómo se distribuyen en los diferentes años.

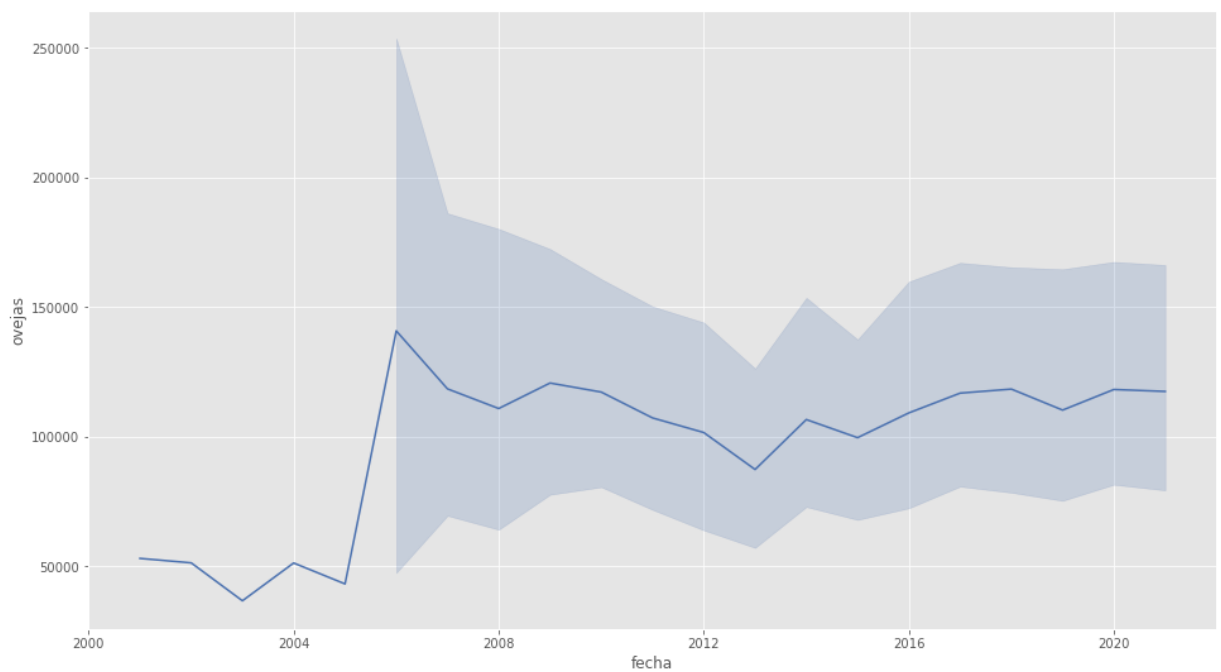
```
In [41]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='ovejas',
hue='departamento')
```

```
Out[41]: <AxesSubplot:xlabel='fecha', ylabel='ovejas'>
```



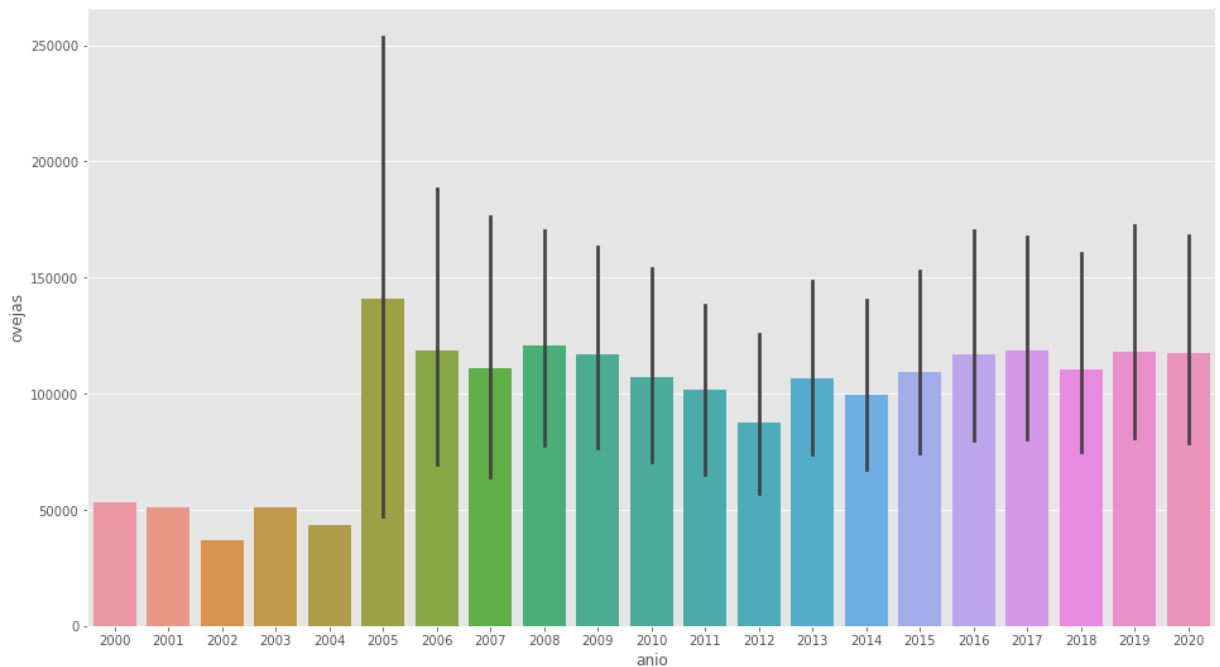
In [42]: `sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='ovejas')`

Out[42]: `<AxesSubplot:xlabel='fecha', ylabel='ovejas'>`



In [43]: `sns.barplot(data=datos_ordenados_fecha, x='anio', y='ovejas')`

Out[43]: `<AxesSubplot:xlabel='anio', ylabel='ovejas'>`



Estos tres gráficos nos brindan información sobre la variación del total de ovejas a lo largo del tiempo. El gráfico de dispersión muestra cada departamento como un punto en el plano cartesiano, con su posición en el eje X e Y representando la cantidad de ovejas en un año determinado. Cada punto está coloreado según el departamento al que pertenece, lo que nos permite comparar visualmente la cantidad de ovejas entre departamentos. Podemos notar que los departamentos de Tehuelches y Río Senger presentan valores que superan los 150 mil y por ende son más variables, en comparación con el resto de departamentos que están por debajo de ese valor.

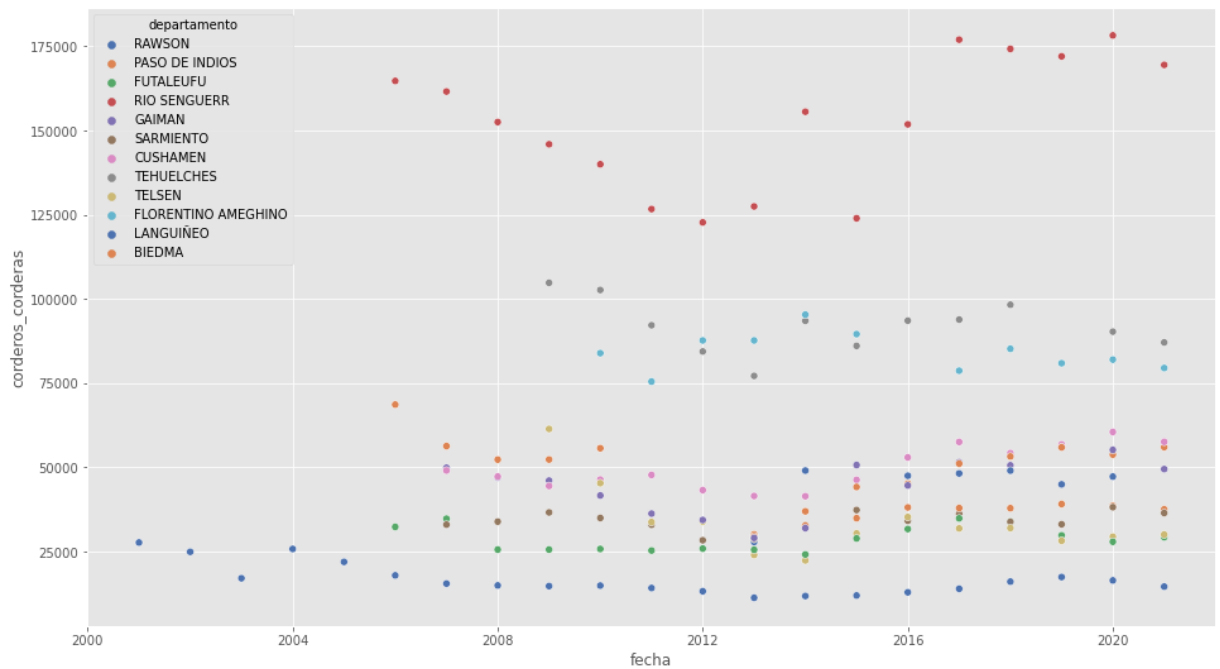
Entonces cuando la cantidad de ovejas superan los 150 mil los puntos están muy dispersos, lo que puede indicar una gran variabilidad en la producción de ovejas a lo largo de los años; y por otro lado, cuando los totales de ovejas son inferiores a 150 mil, las filas de puntos son más uniformes, lo que puede indicar una producción más estable a lo largo de los años. Este patrón también se puede observar en el gráfico de línea y en el de barras, que incluyen intervalos de confianza para mostrar la variabilidad de los datos.

Corderos

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **corderos** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de corderos a lo largo del tiempo y cómo se distribuyen en los diferentes años.

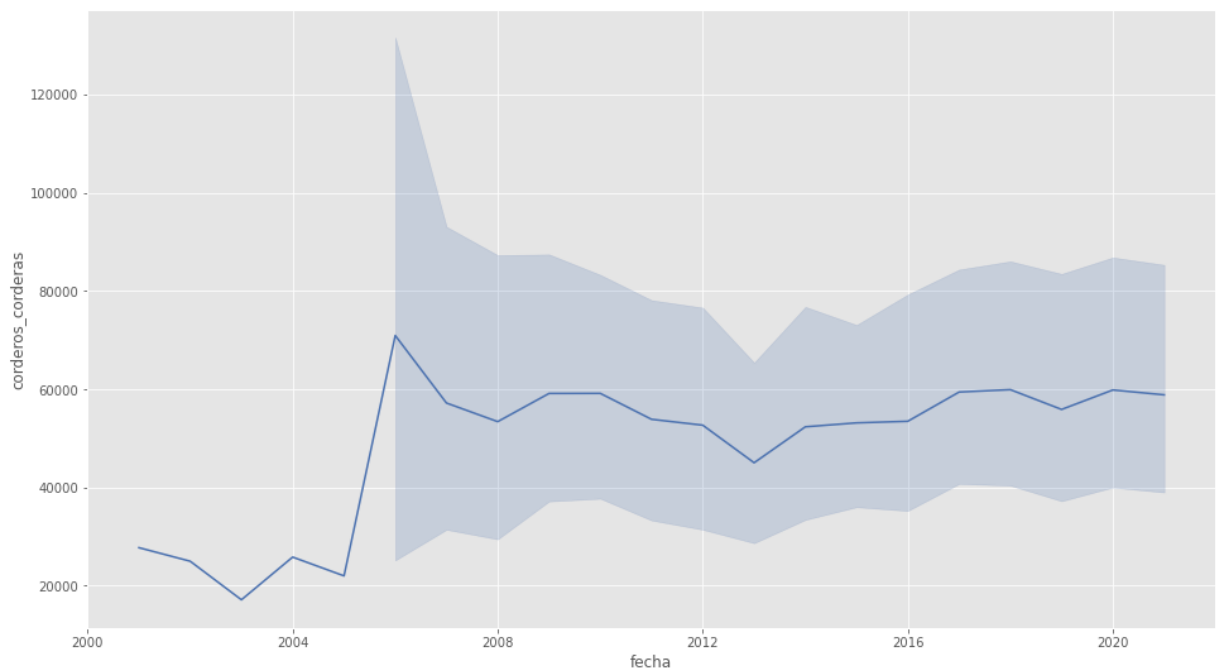
```
In [44]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',
                        y='corderos_corderas', hue='departamento')
```

```
Out[44]: <AxesSubplot:xlabel='fecha', ylabel='corderos_corderas'>
```

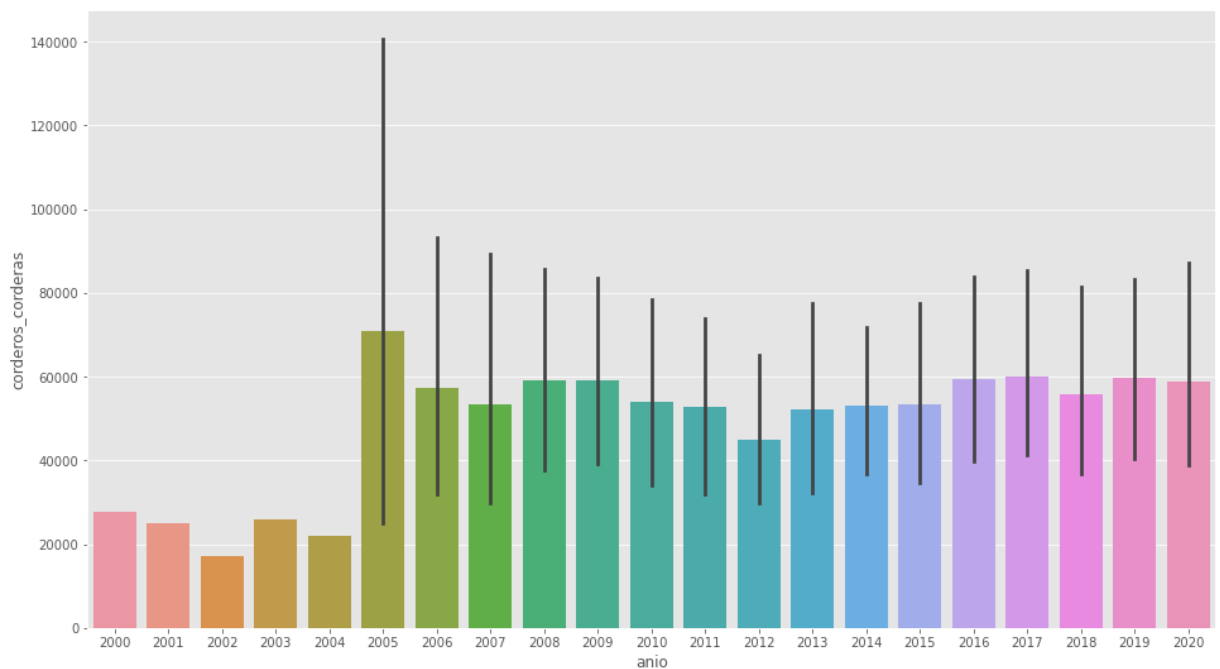
```
In [45]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',
y='corderos_corderas')
```

```
Out[45]: <AxesSubplot:xlabel='fecha', ylabel='corderos_corderas'>
```



```
In [46]: sns.barplot(data=datos_ordenados_fecha, x='anio',
y='corderos_corderas')
```

```
Out[46]: <AxesSubplot:xlabel='anio', ylabel='corderos_corderas'>
```



Se puede obtener información acerca de la variación en la producción de corderos a lo largo del tiempo a través de estos tres gráficos. El scatterplot muestra cada departamento representado por un punto en un plano cartesiano, con su ubicación en los ejes X e Y indicando la cantidad de corderos en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que facilita la comparación de la cantidad de corderos entre departamentos. Se puede observar que los departamentos de Tehuelches, Río Senger y Florentino Ameghino presentan valores que superan los 75 mil, y los datos varían más en comparación con otros departamentos que presentan valores inferiores.

Entonces cuando el total de corderos supera los 75 mil, los puntos están muy dispersos, lo que sugiere una mayor variabilidad en la producción a lo largo de los años. En cambio, cuando los totales de corderos son inferiores a 75 mil, las filas de puntos son más uniformes, lo que indica una producción más estable en el transcurso de los años. Este patrón se refleja también en los gráficos de línea y barra, los cuales incluyen intervalos de confianza para mostrar la variabilidad de los datos.

Producción de lana

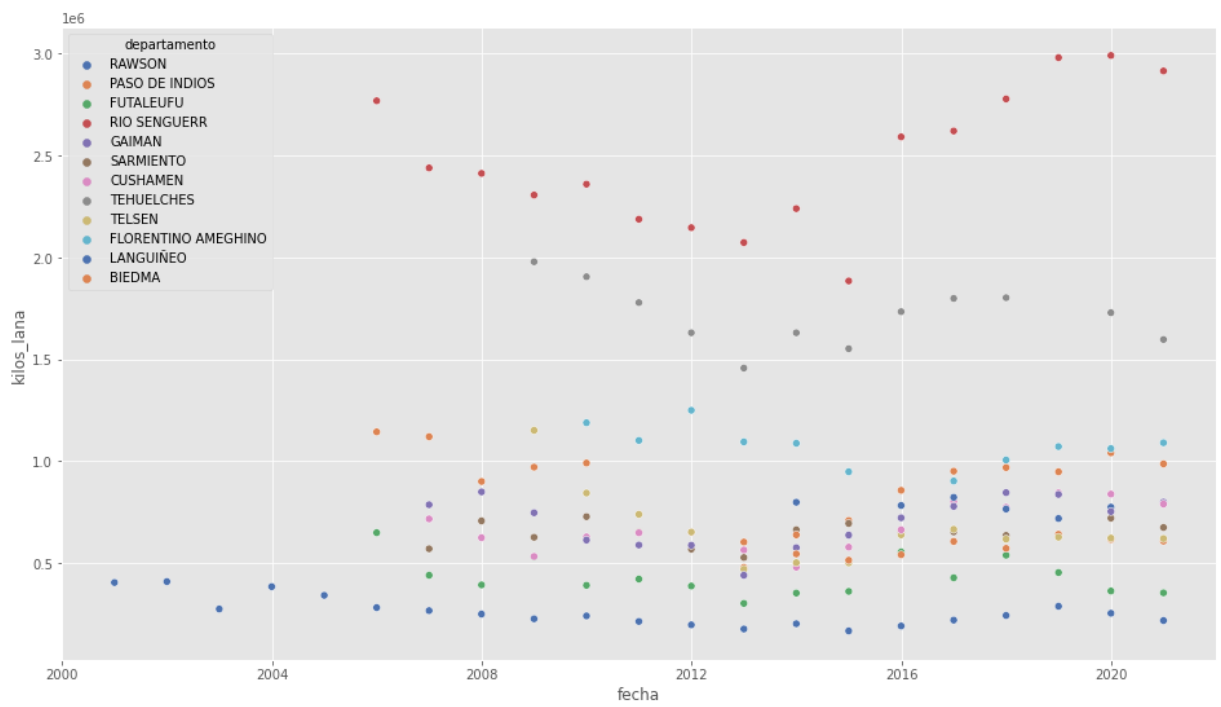
El objetivo primordial en la cría de ovejas para obtener lana es producir lanas de calidad cruzada fina y mediana, con un diámetro de entre 25 y 32 micras y un rendimiento de lavado superior al 75%. En cuanto a la situación en la provincia de Chubut, se puede observar que la producción de lana es en su mayoría de buena calidad, con un diámetro promedio en el rango deseado y un rendimiento de lavado adecuado, según lo indican los gráficos presentados a continuación.

Kilos de lana

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **kilos de lana** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la cantidad de lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

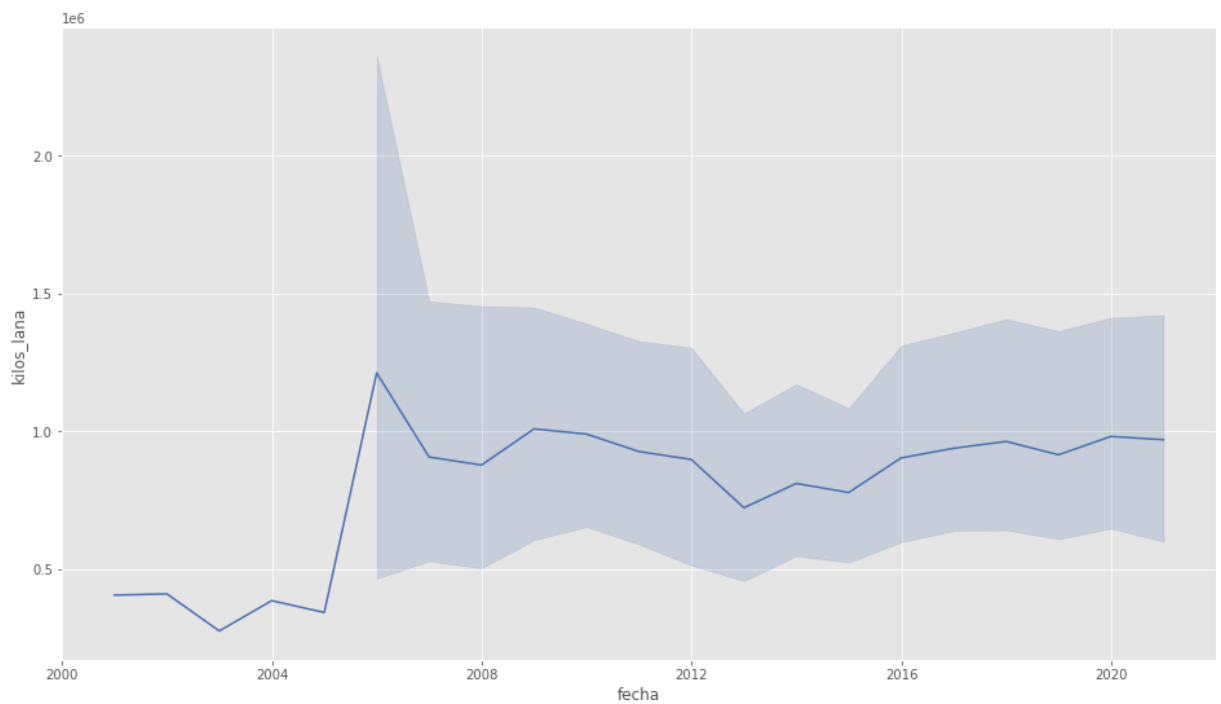
```
In [47]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha',
                        y='kilos_lana', hue='departamento')
```

```
Out[47]: <AxesSubplot:xlabel='fecha', ylabel='kilos_lana'>
```



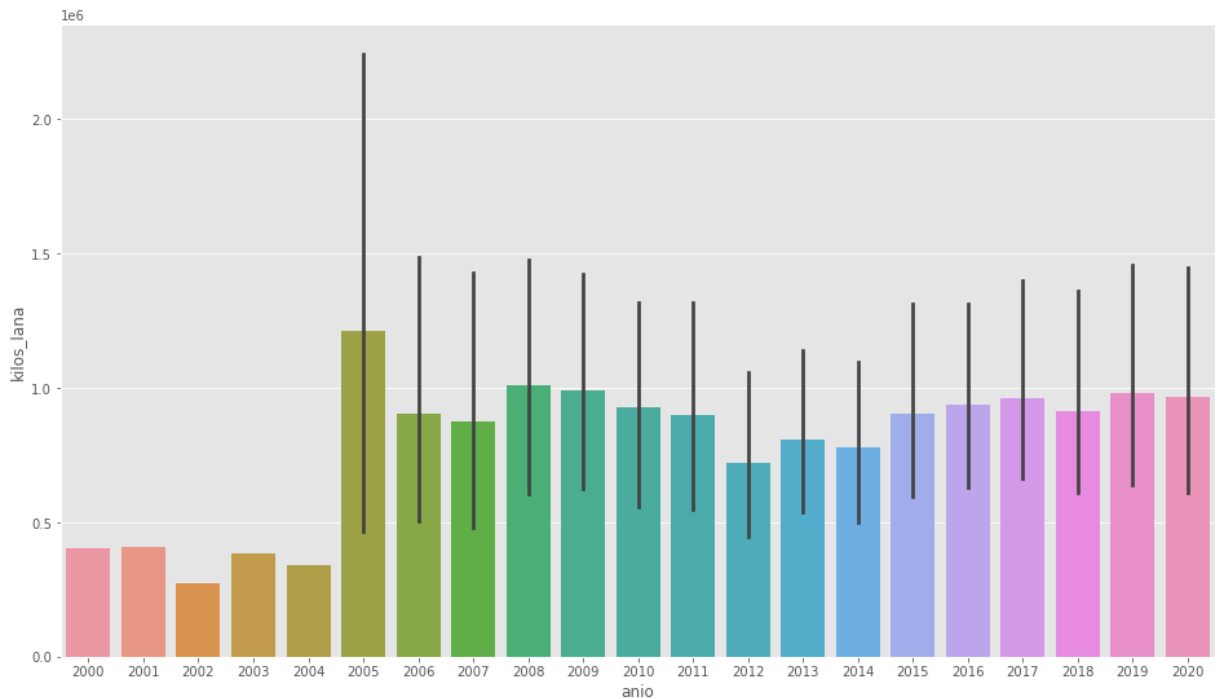
```
In [48]: sns.lineplot(data=datos_ordenados_fecha, x='fecha',
y='kilos_lana')
```

```
Out[48]: <AxesSubplot:xlabel='fecha', ylabel='kilos_lana'>
```



```
In [49]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='kilos_lana')
```

```
Out[49]: <AxesSubplot:xlabel='anio', ylabel='kilos_lana'>
```



Para obtener información sobre la variación en la producción de lana a lo largo del tiempo, se pueden examinar estos tres gráficos. El scatterplot muestra cada departamento representado por un punto en un plano cartesiano, con su ubicación en los ejes X e Y indicando la cantidad de lana en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que permite comparar visualmente la cantidad de lana entre departamentos. Se puede observar que los departamentos de Tehuelches y Río Senger presentan valores que superan la tonelada y media de lana, y son más variables en comparación con otros departamentos que presentan valores inferiores.

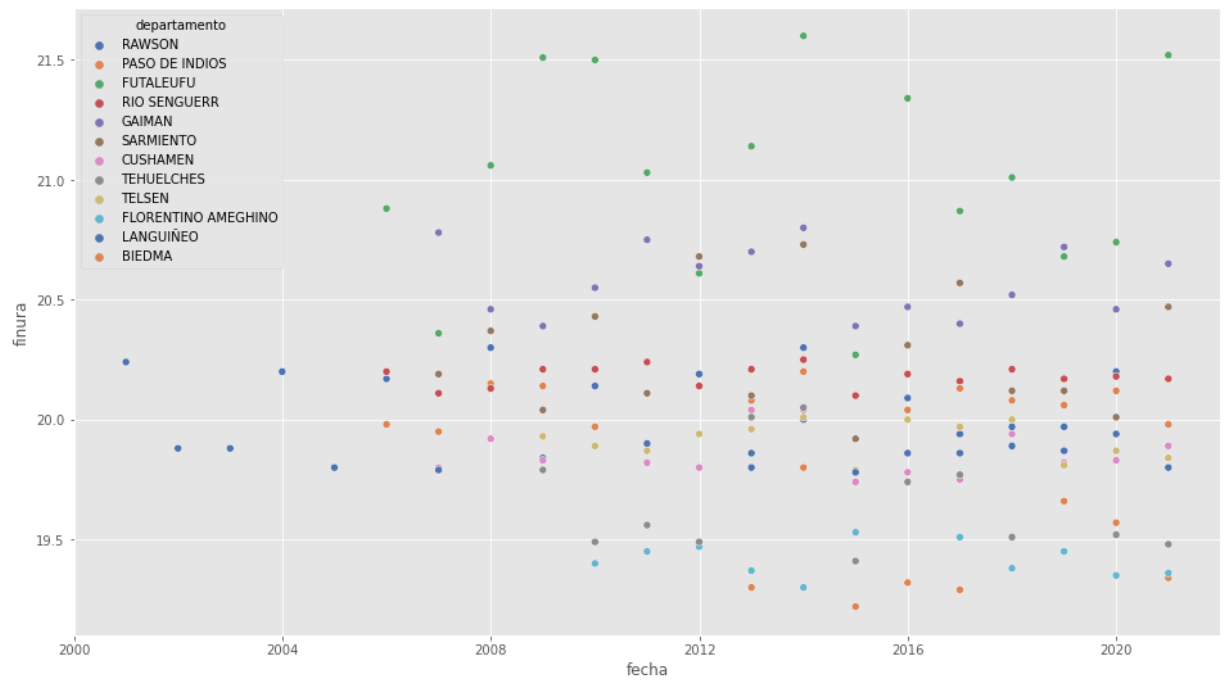
Entonces cuando el total de kilos de lana supera los 1500000, los puntos están muy dispersos, lo que sugiere una mayor variabilidad en la producción a lo largo de los años. En cambio, cuando la cantidad de lana es inferior a 1.5 toneladas, las filas de puntos son más uniformes, lo que indica una producción más estable en el transcurso de los años. Este patrón también se refleja en los gráficos de línea y barra, los cuales incluyen intervalos de confianza para mostrar la variabilidad de los datos. Estos gráficos también muestran una leve tendencia general de aumento en la producción de lana a partir del año 2012 en la mayoría de los departamentos, con algunas excepciones.

Finura

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **finura de la lana** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en la finura de la lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

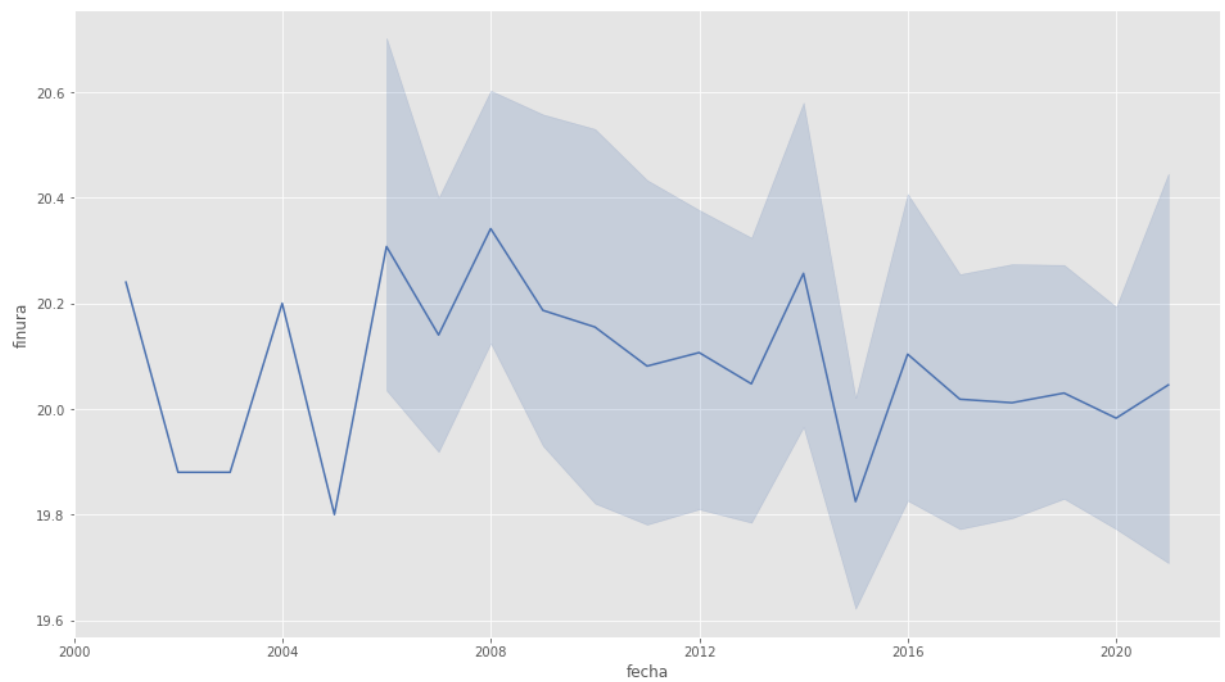
```
In [50]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='finura',
                        hue='departamento')

Out[50]: <AxesSubplot:xlabel='fecha', ylabel='finura'>
```



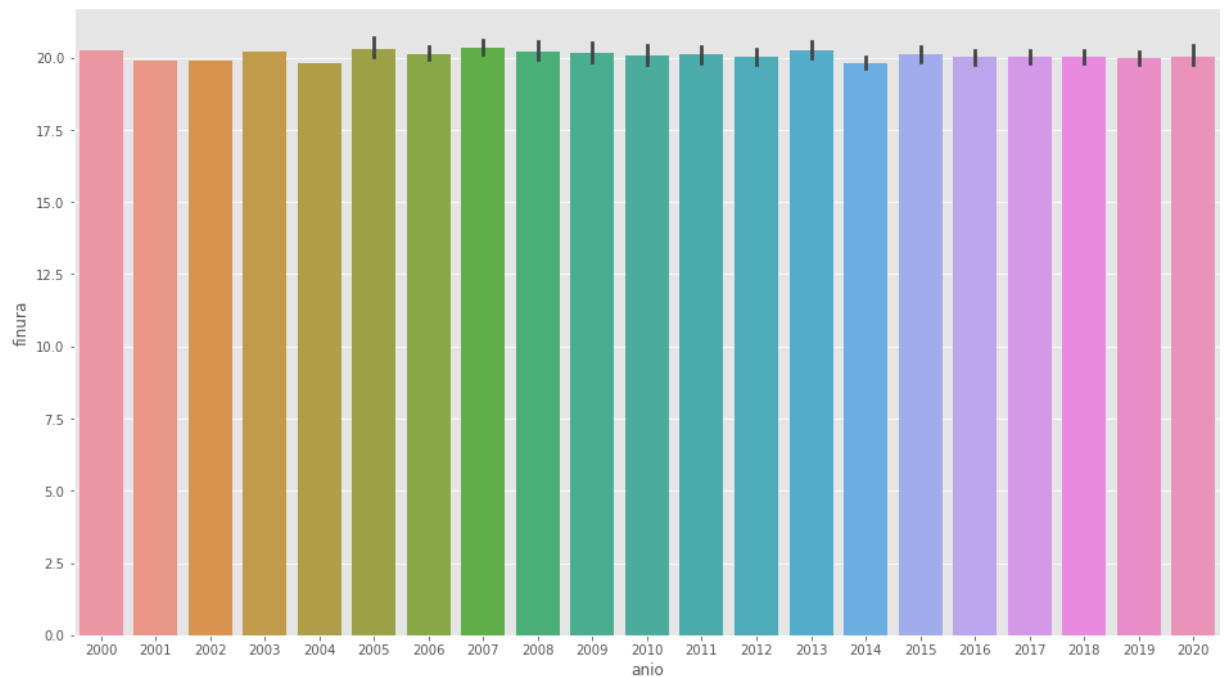
```
In [51]: sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='finura')
```

```
Out[51]: <AxesSubplot: xlabel='fecha', ylabel='finura'>
```



```
In [52]: sns.barplot(data=datos_ordenados_fecha, x='anio', y='finura')
```

```
Out[52]: <AxesSubplot: xlabel='anio', ylabel='finura'>
```



Para obtener información sobre la variación en la finura de la lana a lo largo del tiempo, es posible analizar estos tres gráficos. El scatterplot representa cada departamento mediante un punto en un plano cartesiano, y la posición del punto en los ejes X e Y indica la finura en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que facilita la comparación visual de la finura de la lana entre departamentos. Se puede observar que los departamentos de Tehuelches, Florentino Ameghino y Biedma presentan valores más bajos, lo que indica lanas más finas y altamente valoradas en el mercado. En contraste, Futalefu y Gaiman tienen la lana más gruesa en Chubut.

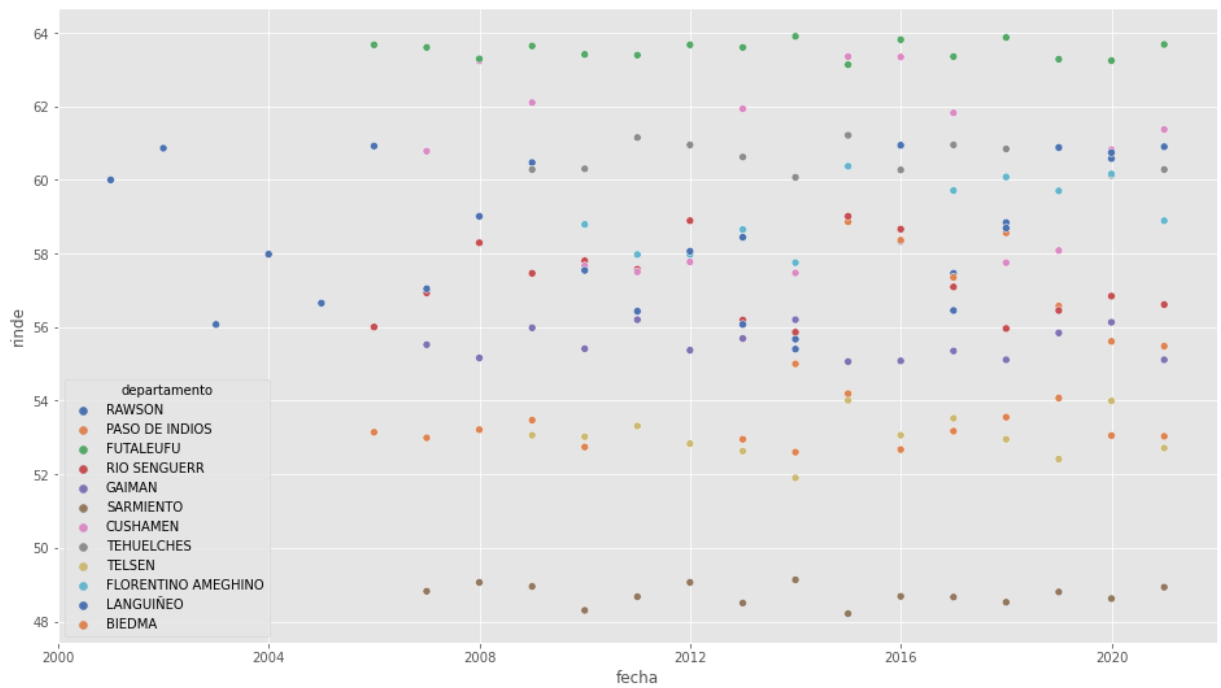
Sin embargo, también se puede observar que la finura de la lana en Chubut no varía significativamente, ya que se mantiene entre 19.6 y 20.6 en los gráficos de línea y barra. Esto sugiere que la provincia de Chubut mantiene un buen manejo de la finura de la lana, lo que es una buena señal para la industria de la lana en la región.

Rinde

A continuación, se mostrarán tres gráficos (scatterplot, lineplot, barplot) que permitirán visualizar la distribución y evolución del atributo **rinde de la lana** para cada año en el conjunto de datos. De esta forma, se podrán observar posibles patrones o tendencias en el rinde de la lana a lo largo del tiempo y cómo se distribuyen en los diferentes años.

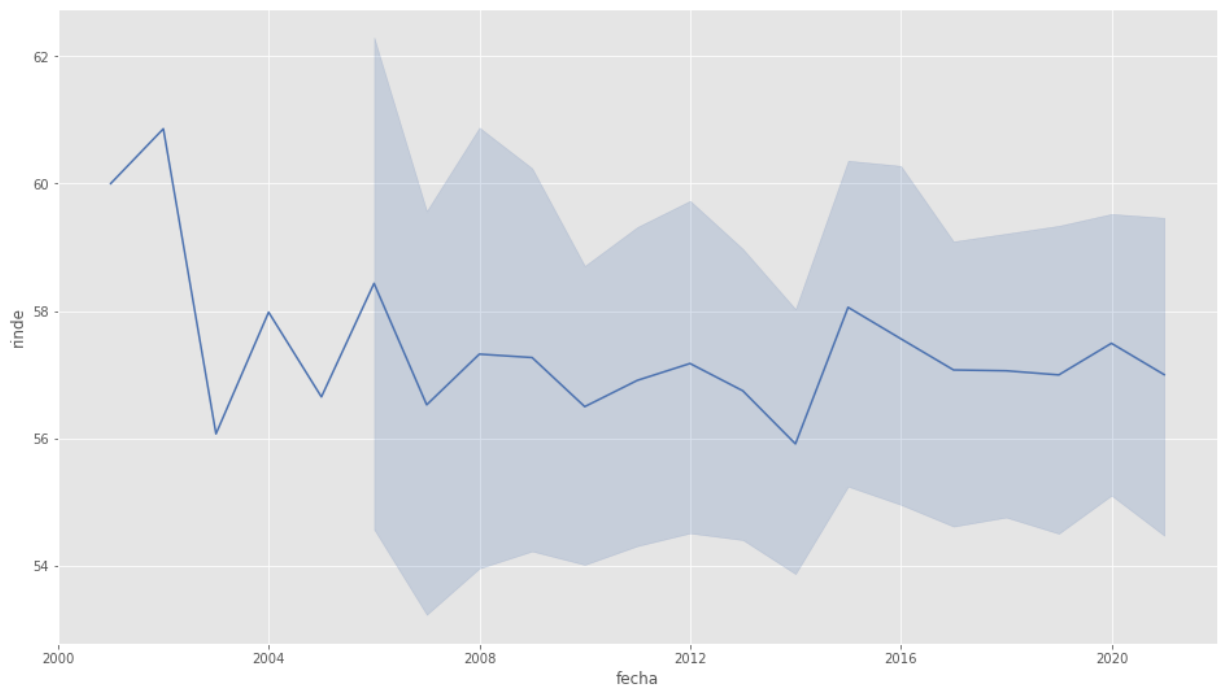
```
In [53]: sns.scatterplot(data=datos_ordenados_fecha, x='fecha', y='rinde',
hue='departamento')
```

```
Out[53]: <AxesSubplot:xlabel='fecha', ylabel='rinde'>
```



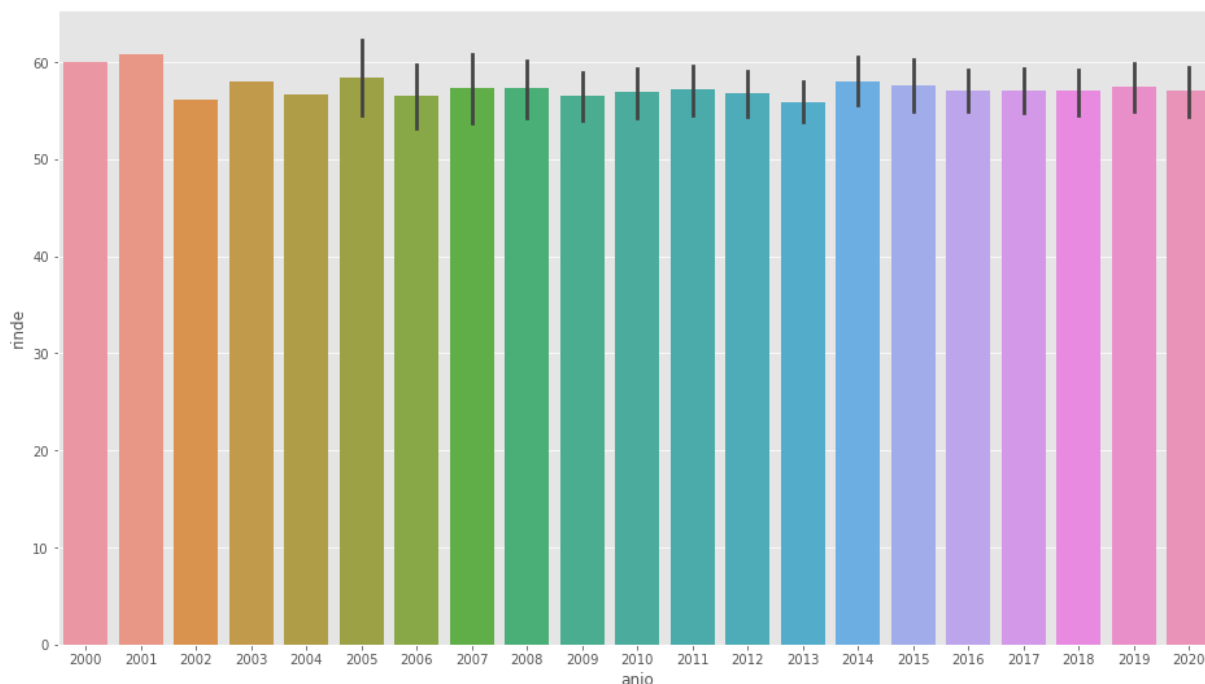
In [54]: `sns.lineplot(data=datos_ordenados_fecha, x='fecha', y='rinde')`

Out[54]: `<AxesSubplot:xlabel='fecha', ylabel='rinde'>`



In [55]: `sns.barplot(data=datos_ordenados_fecha, x='anio', y='rinde')`

Out[55]: `<AxesSubplot:xlabel='anio', ylabel='rinde'>`



Para obtener información sobre la variación en el rendimiento de lavado de la lana a lo largo del tiempo, se pueden examinar estos tres gráficos. El scatterplot muestra cada departamento representado por un punto en un plano cartesiano, con su ubicación en los ejes X e Y indicando el rinde en un año determinado. Los puntos están codificados por colores según el departamento al que pertenecen, lo que permite comparar visualmente el rendimiento de lavado de la lana entre departamentos. Se puede observar que los departamentos de Cushamen y Futaleufú presentan los valores más altos de rinde de la lana, lo que indica que la lana producida en esos departamentos tiene un alto valor en el mercado debido a su mayor rendimiento.

Por otro lado, se puede notar que el departamento de Sarmiento presenta los valores más bajos de rinde de la lana, lo que indica que la lana producida en este departamento tiene un menor valor debido a su menor rendimiento. Sin embargo, el rinde de la lana en Chubut se mantiene bastante estable entre 52 y 62 en los gráficos de línea y barra. Esto sugiere que la provincia de Chubut tiene un buen rendimiento de lavado de la lana, lo que es una buena señal para la industria de la lana en la región. Un alto rendimiento de lavado significa que la lana puede ser lavada y procesada con mayor eficiencia, lo que a su vez se traduce en una mayor calidad del producto final y mayores beneficios económicos para los productores de lana en la región.

Lluvia con otras variables

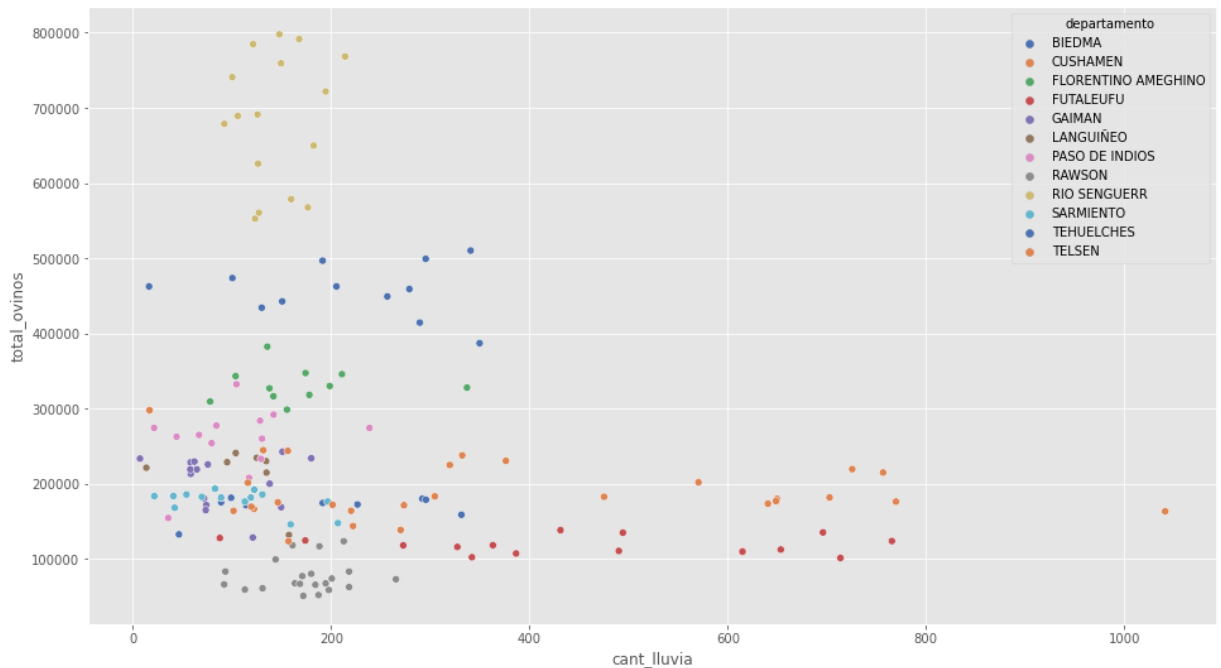
Para entender la influencia de la lluvia en la producción de ganado ovino en la región, es importante examinar la relación que existe entre la cantidad de lluvia acumulada en un año y las variables relacionadas con las existencias de ovejas y la producción de lana. De esta manera, podremos obtener una idea más clara de cómo la cantidad de lluvia puede afectar a la producción ganadera en la región.

Total de ovinos

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `total_ovinos` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de ovinos y la cantidad de lluvia recibida en cada año.

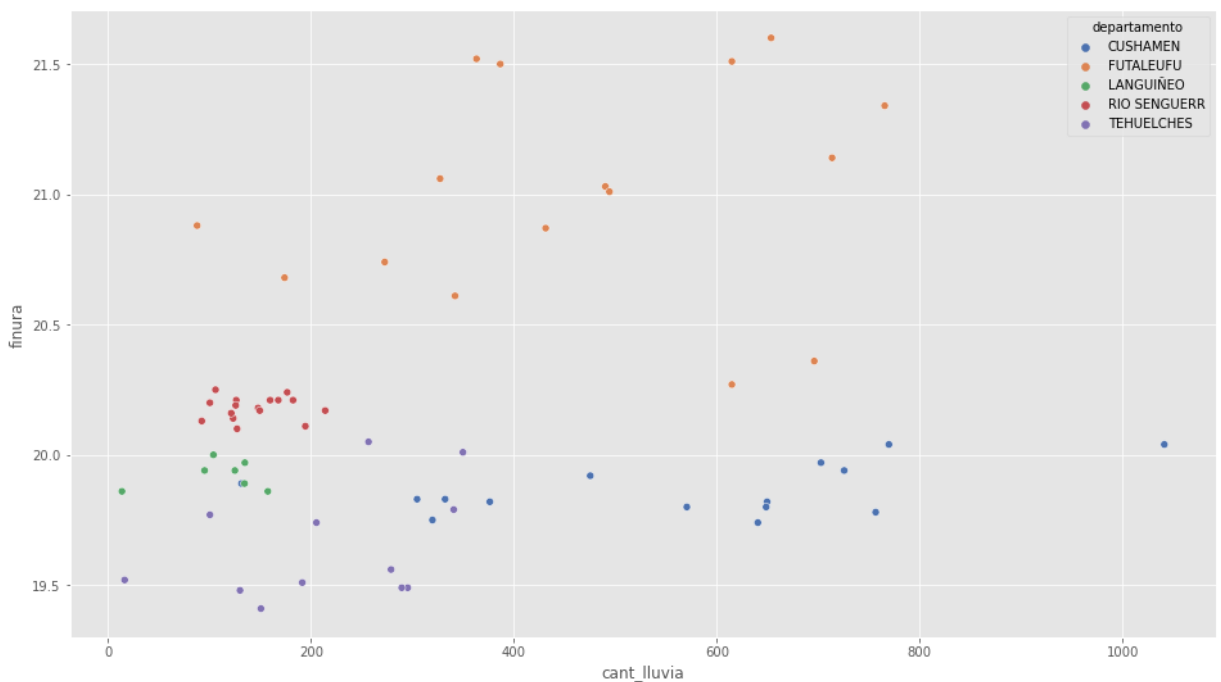

```
In [56]: sns.scatterplot(data=datos, x='cant_lluvia', y='total_ovinos',  
hue='departamento')
```

```
Out[56]: <AxesSubplot:xlabel='cant_lluvia', ylabel='total_ovinos'>
```



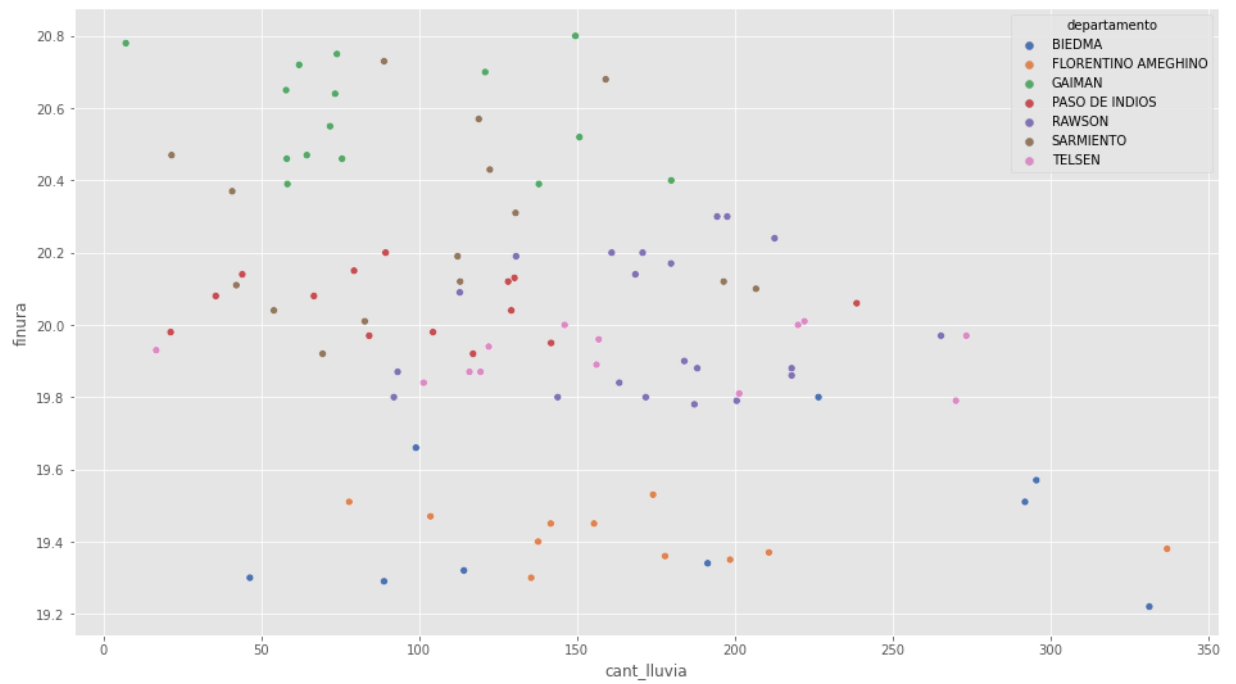
```
In [7]: sns.scatterplot(data=departamentos_cordillera, x='cant_lluvia',  
y='finura', hue='departamento')
```

```
Out[7]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



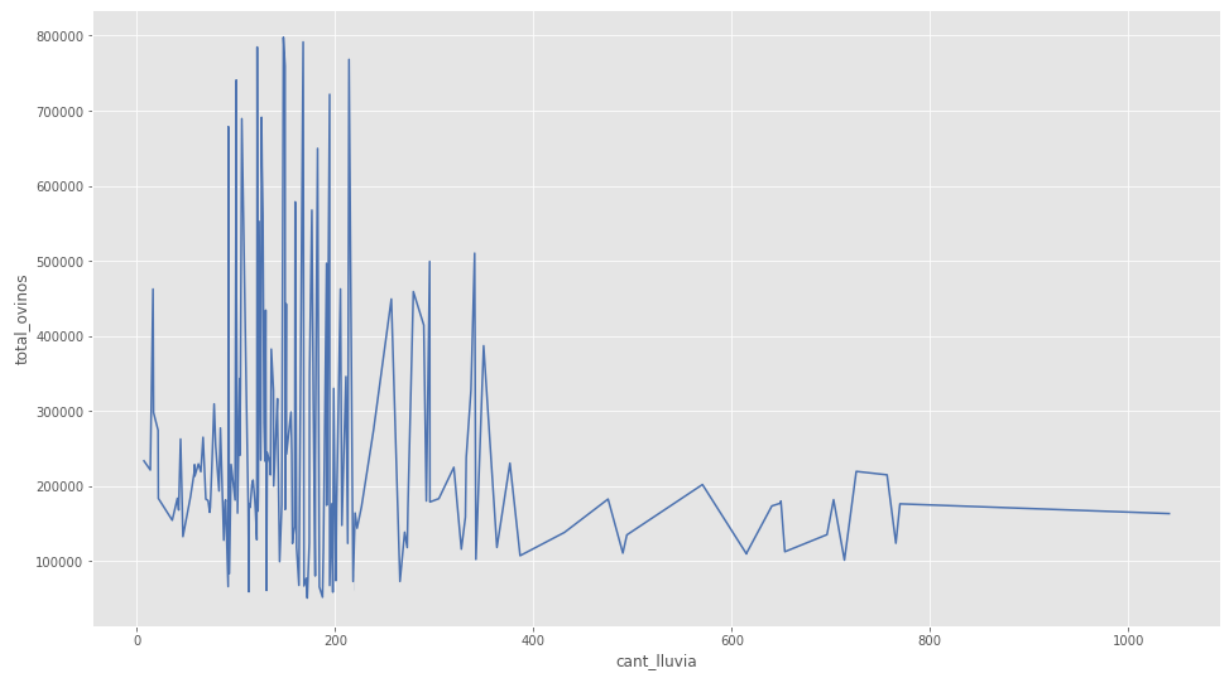
```
In [8]: sns.scatterplot(data=departamentos_meseta, x='cant_lluvia',  
y='finura', hue='departamento')
```

```
Out[8]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



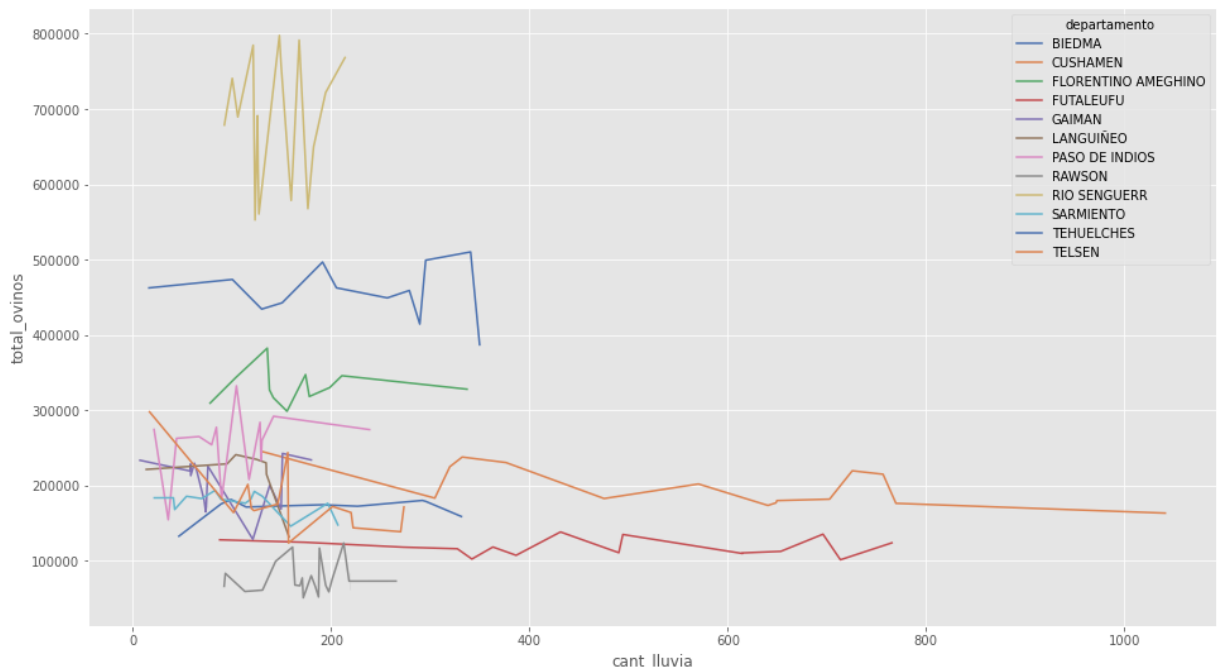
```
In [57]: sns.lineplot(data=datos, x='cant_lluvia', y='total_ovinos')
```

```
Out[57]: <AxesSubplot:xlabel='cant_lluvia', ylabel='total_ovinos'>
```



```
In [9]: sns.lineplot(data=datos, x='cant_lluvia', y='total_ovinos',  
hue='departamento')
```

```
Out[9]: <AxesSubplot:xlabel='cant_lluvia', ylabel='total_ovinos'>
```



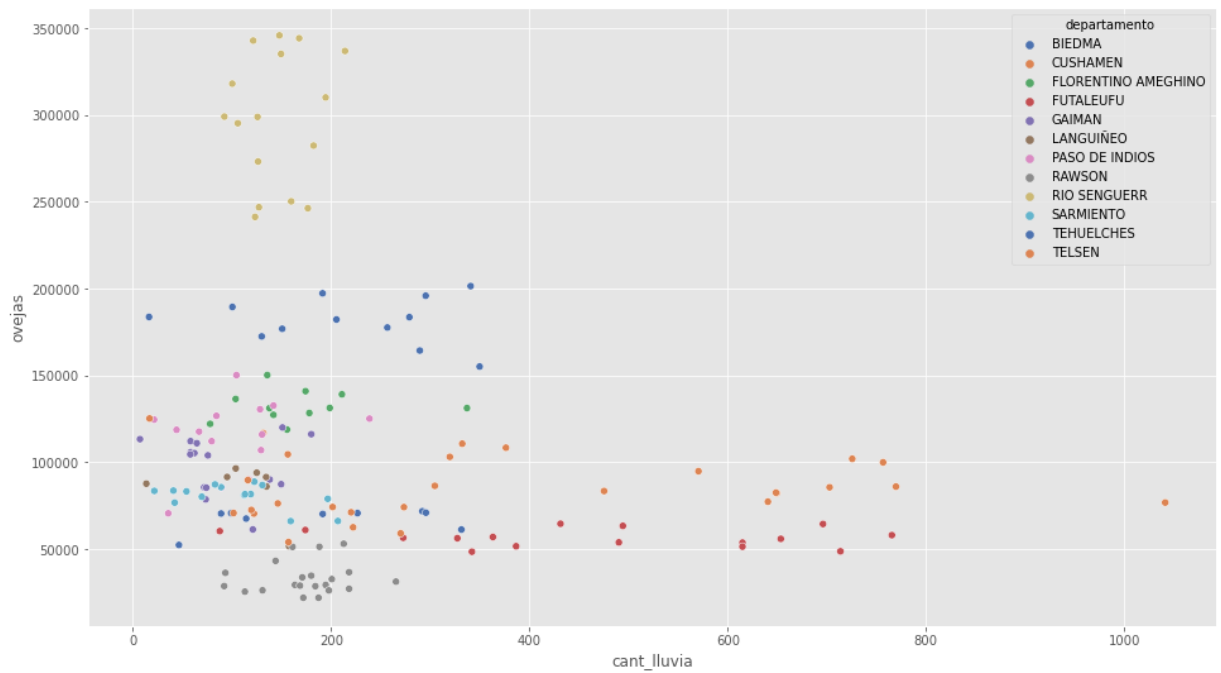
En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de ovinos para cada departamento. Sin embargo, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de ovinos, ya que al aumentar la lluvia, la cantidad de ovinos también se incrementa levemente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia excepcionalmente altos, pero no necesariamente tienen la mayor cantidad de ovinos. También se puede observar esta leve tendencia en el gráfico de líneas, aunque presenta muchos picos y valles que dificultan su interpretación.

Ovejas

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `ovejas` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de ovejas y la cantidad de lluvia recibida en cada año.

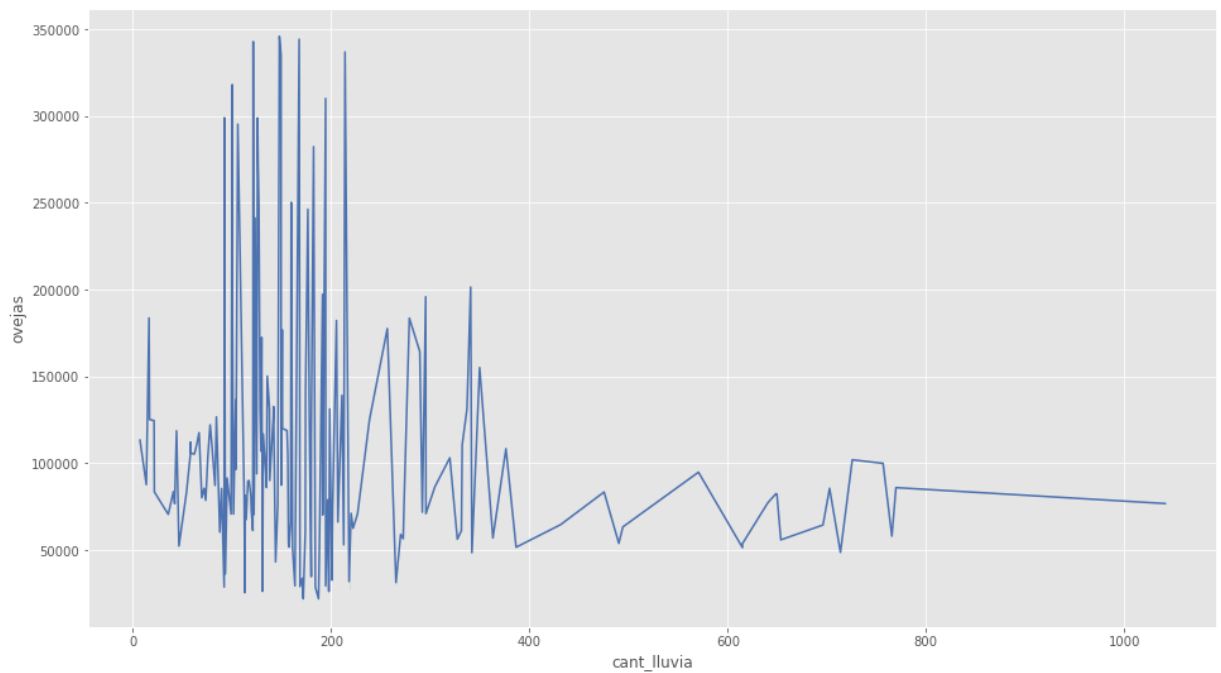
```
In [58]: sns.scatterplot(data=datos, x='cant_lluvia', y='ovejas',
hue='departamento')
```

```
Out[58]: <AxesSubplot:xlabel='cant_lluvia', ylabel='ovejas'>
```



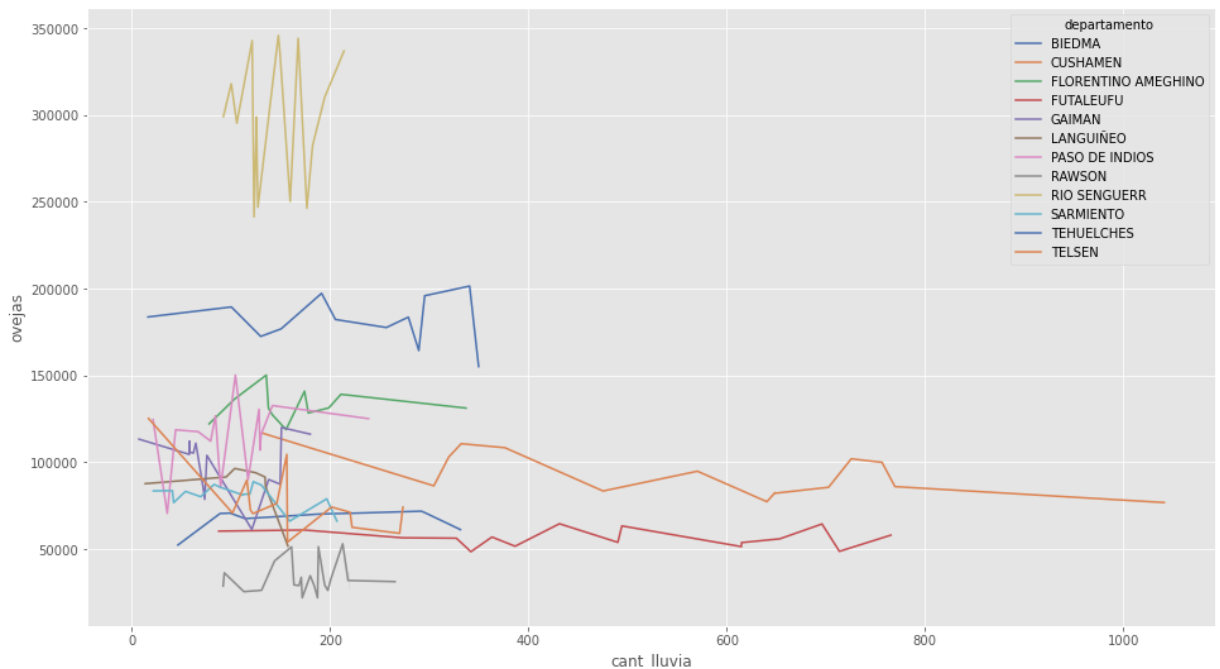
```
In [59]: sns.lineplot(data=datos, x='cant_lluvia', y='ovejas')
```

```
Out[59]: <AxesSubplot:xlabel='cant_lluvia', ylabel='ovejas'>
```



```
In [10]: sns.lineplot(data=datos, x='cant_lluvia', y='ovejas',  
hue='departamento')
```

```
Out[10]: <AxesSubplot:xlabel='cant_lluvia', ylabel='ovejas'>
```



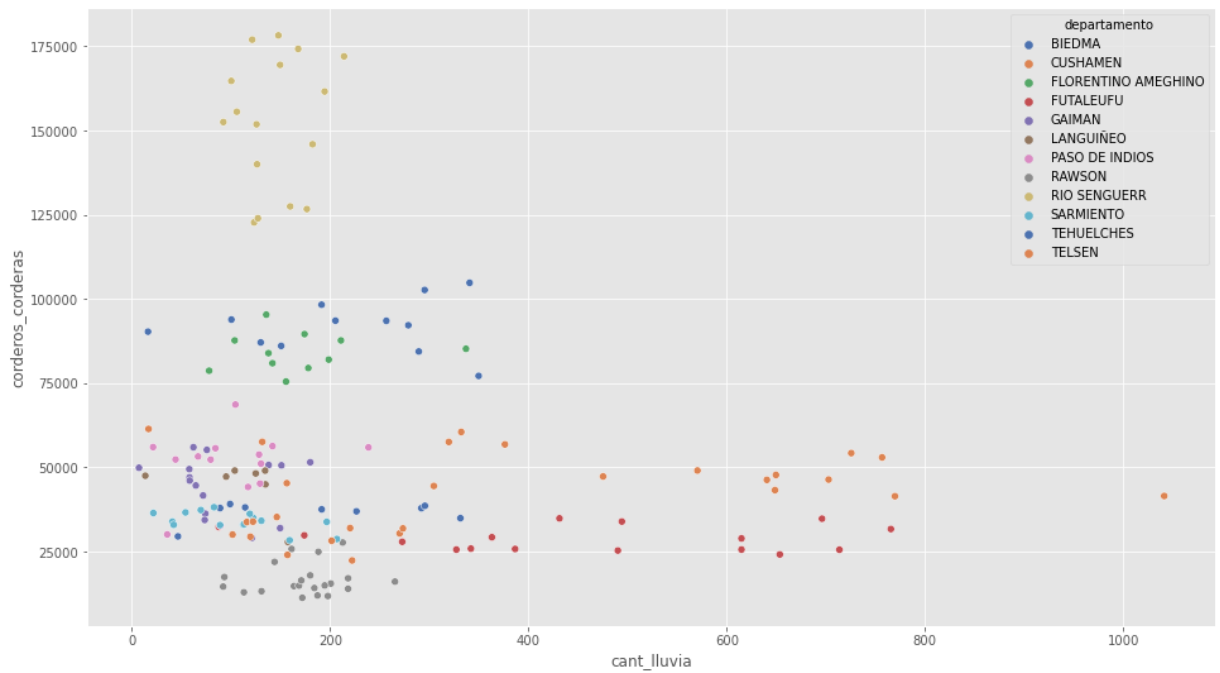
En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de ovejas para cada departamento. Y sus formas son muy similares al anterior con el total de ovinos, por lo tanto, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de ovejas, ya que al aumentar la lluvia, la cantidad de ovejas también se incrementa ligeramente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia muy altos, pero no necesariamente tienen la mayor cantidad de ovejas. También se puede observar esta leve tendencia en el gráfico de líneas, aunque presenta muchos picos y valles que hacen difícil ver esta leve tendencia.

Corderos

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **corderos** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de corderos y la cantidad de lluvia recibida en cada año.

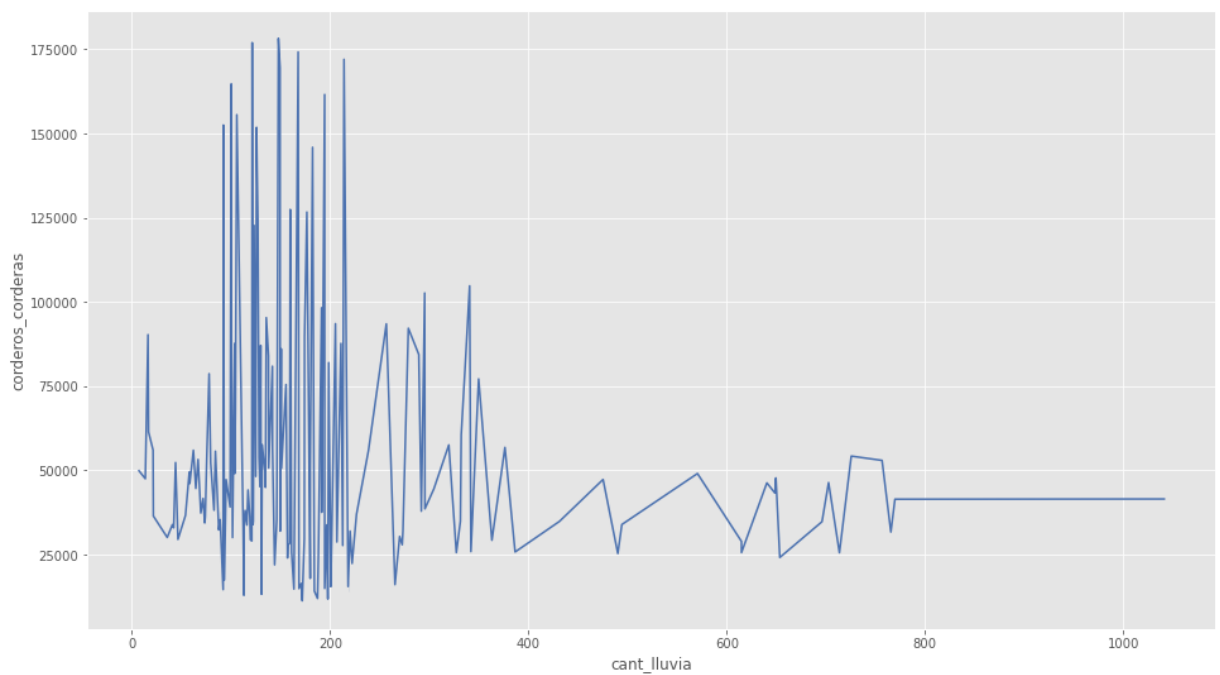
```
In [60]: sns.scatterplot(data=datos, x='cant_lluvia',
                        y='corderos_corderas', hue='departamento')
```

```
Out[60]: <AxesSubplot:xlabel='cant_lluvia', ylabel='corderos_corderas'>
```



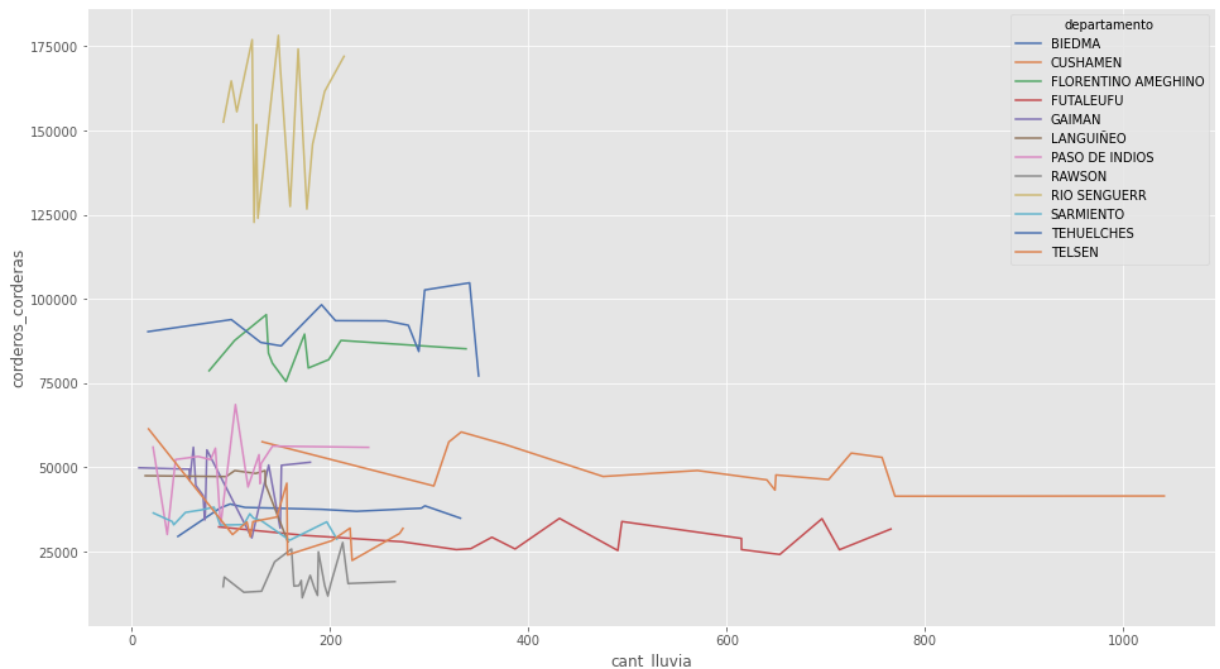
In [61]: `sns.lineplot(data=datos, x='cant_lluvia', y='corderos_corderas')`

Out[61]: `<AxesSubplot:xlabel='cant_lluvia', ylabel='corderos_corderas'>`



In [11]: `sns.lineplot(data=datos, x='cant_lluvia', y='corderos_corderas', hue='departamento')`

Out[11]: `<AxesSubplot:xlabel='cant_lluvia', ylabel='corderos_corderas'>`



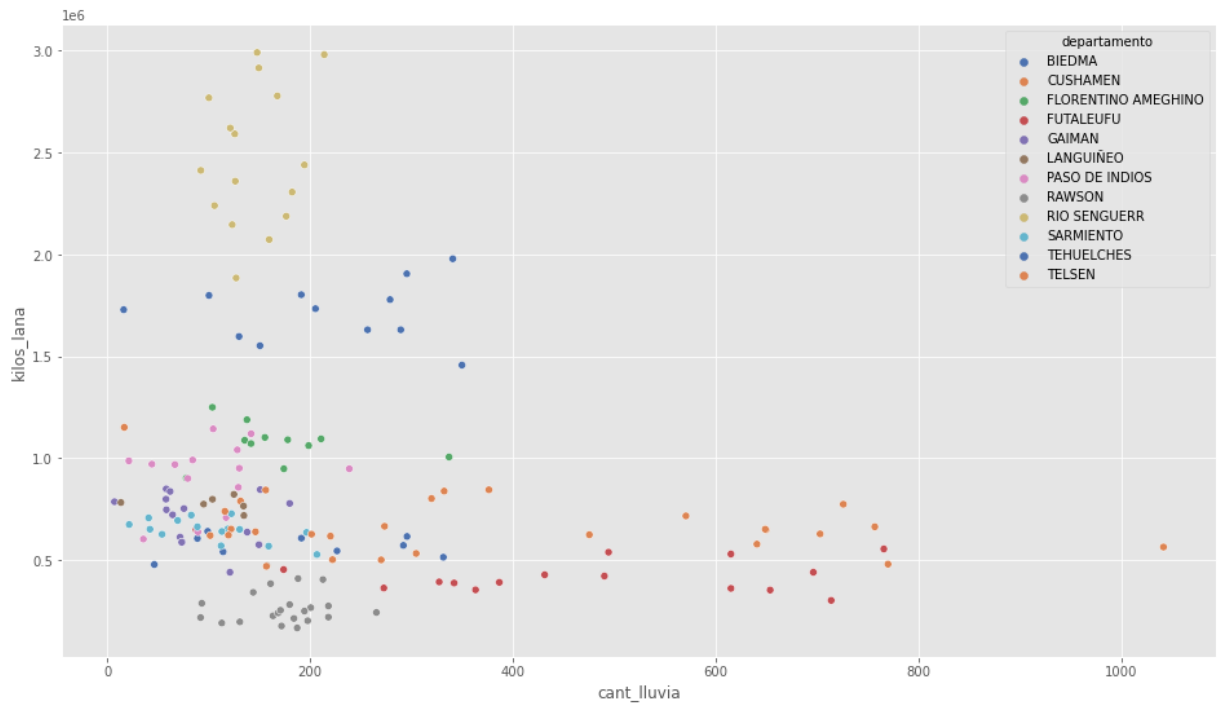
En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de corderos para cada departamento. Y sus formas son muy similares a los anteriores graficos del total de ovinos y ovejas, por lo tanto, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar también, una leve relación lineal entre la cantidad de lluvia y el total de corderos, ya que al aumentar la lluvia, la cantidad de corderos también se incrementa ligeramente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia altos, pero no necesariamente tienen la mayor cantidad de corderos. También se puede observar esta leve tendencia en el gráfico de líneas, aunque presenta muchos picos y valles que hacen difícil su interpretación.

Kilo de lana

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **kilos de lana** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en la cantidad de lana y la cantidad de lluvia recibida en cada año.

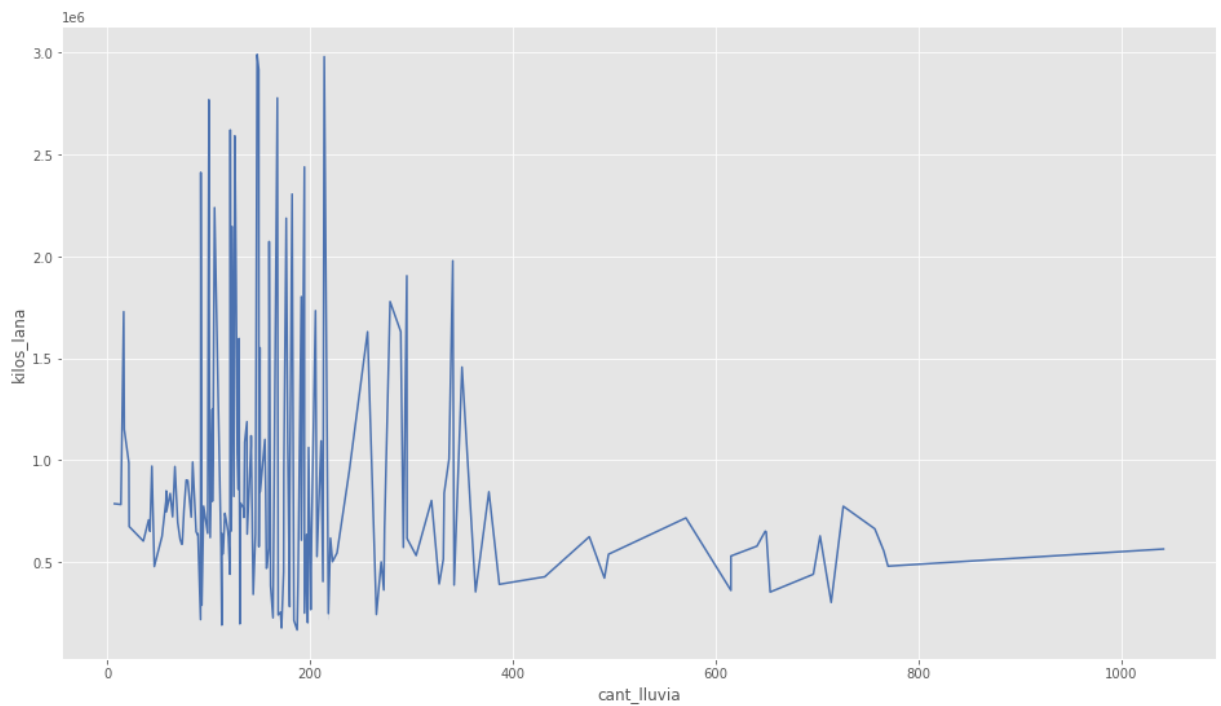
```
In [62]: sns.scatterplot(data=datos, x='cant_lluvia', y='kilos_lana',
hue='departamento')
```

```
Out[62]: <AxesSubplot:xlabel='cant_lluvia', ylabel='kilos_lana'>
```



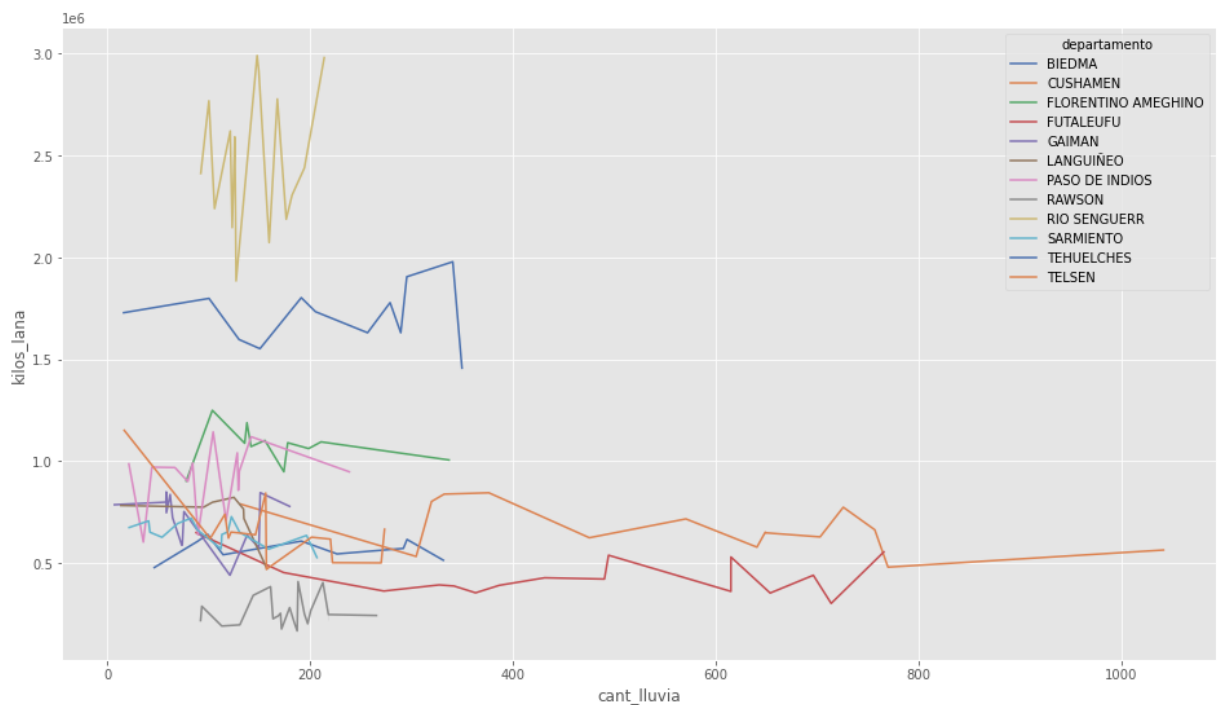
In [63]: `sns.lineplot(data=datos, x='cant_lluvia', y='kilos_lana')`

Out[63]: `<AxesSubplot:xlabel='cant_lluvia', ylabel='kilos_lana'>`



In [12]: `sns.lineplot(data=datos, x='cant_lluvia', y='kilos_lana', hue='departamento')`

Out[12]: `<AxesSubplot:xlabel='cant_lluvia', ylabel='kilos_lana'>`



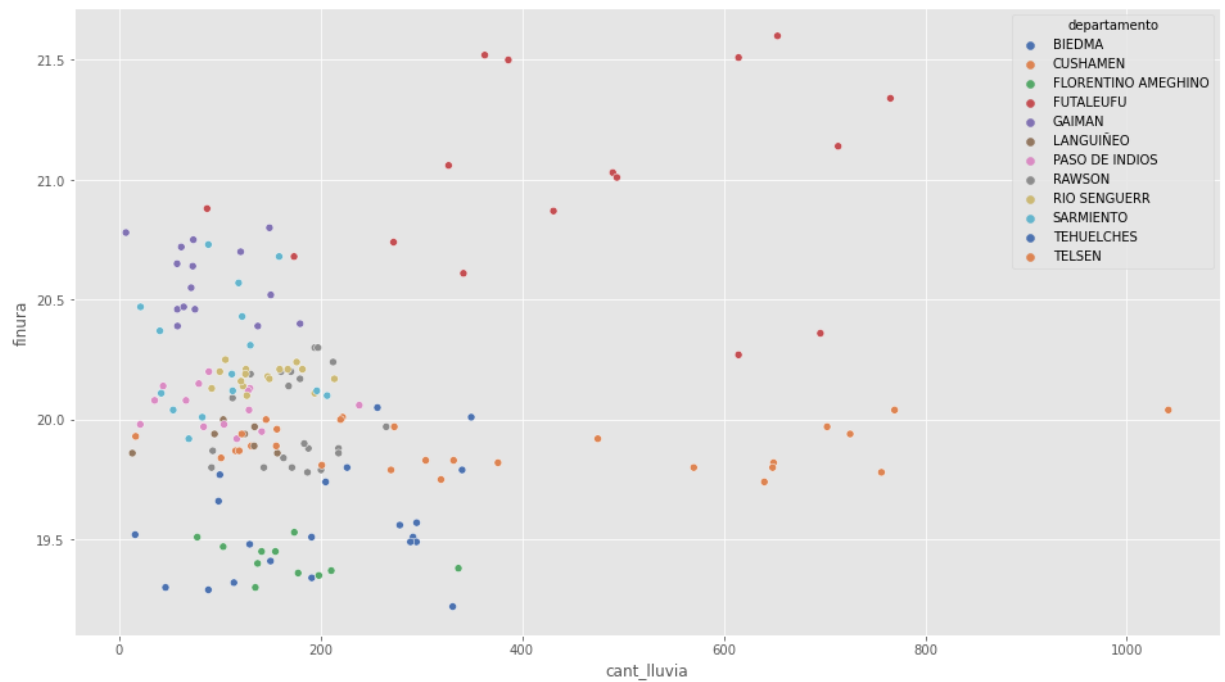
En los gráficos de dispersión y líneas no se puede apreciar una relación directa entre la cantidad de lluvia acumulada y la cantidad de lana para cada departamento. Y sus formas son muy similares a los anteriores 3 gráficos de existencias, por lo tanto, si nos enfocamos en un solo departamento y analizamos sus datos, podemos observar una leve relación lineal entre la cantidad de lluvia y el total de lana, ya que al aumentar la lluvia, la cantidad de lana también se incrementa ligeramente. Es importante tener en cuenta que existen excepciones a esta tendencia, como los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen datos atípicos de lluvia excepcionalmente altos, pero no necesariamente tienen la mayor producción de lana. También se puede observar esta leve tendencia en el gráfico de líneas, aunque presenta muchos picos y valles que no tienen en cuenta el departamento y por ende se dificulta su interpretación.

Finura

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo **finura** y la **lluvia** para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en el valor de la finura y la cantidad de lluvia recibida en cada año.

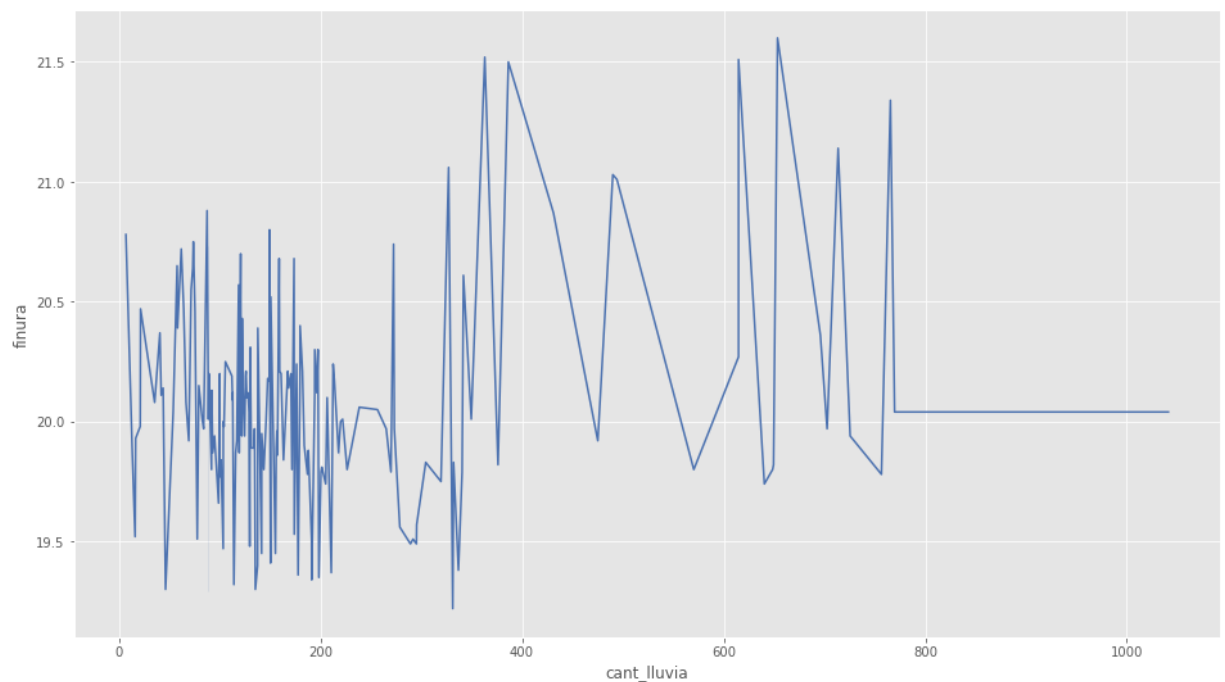
```
In [64]: sns.scatterplot(data=datos, x='cant_lluvia', y='finura',
hue='departamento')
```

```
Out[64]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



```
In [65]: sns.lineplot(data=datos, x='cant_lluvia', y='finura')
```

```
Out[65]: <AxesSubplot:xlabel='cant_lluvia', ylabel='finura'>
```



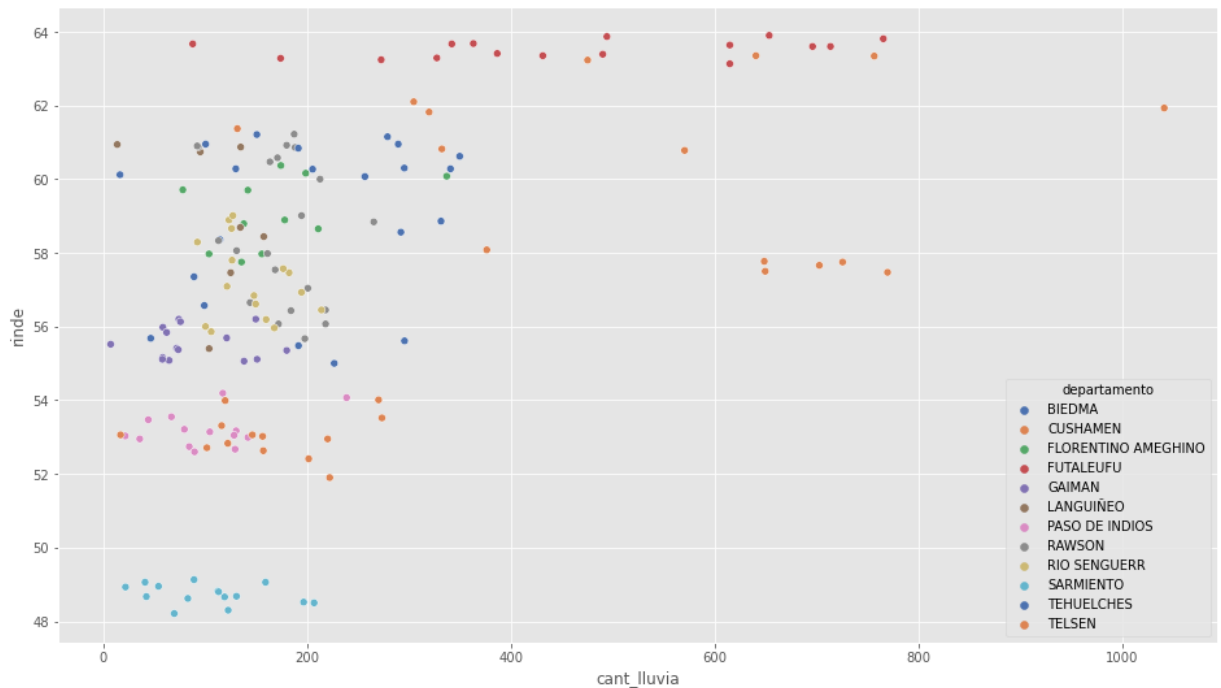
No se puede apreciar una relación clara entre la finura de la lana y la cantidad de lluvia acumulada en el año, y se observa que a medida que aumenta la cantidad de lluvia, la lana se vuelve más gruesa, que no necesariamente es lo que buscan los productores ganaderos. En el gráfico de líneas no se observa ningún patrón discernible y en el gráfico de dispersión, la mayoría de los datos se concentran en un rango estrecho entre 0 y 400 mm de lluvia y una finura de lana entre 19.2 y 20.8 micrones, lo que sugiere que hay poca variabilidad en los datos. Sin embargo, existen valores atípicos en los departamentos de Cushamen y Futaleufú, que se encuentran en la cordillera y tienen una mayor cantidad de lluvia en promedio.

Rinde

En los siguientes gráficos (scatterplot y lineplot) se presentará la asociación entre el atributo `rinde` y la `lluvia` para cada año registrado en los datos. Con la ayuda de estos gráficos, será posible detectar cualquier patrón o tendencia en el valor del rinde y la cantidad de lluvia recibida en cada año.

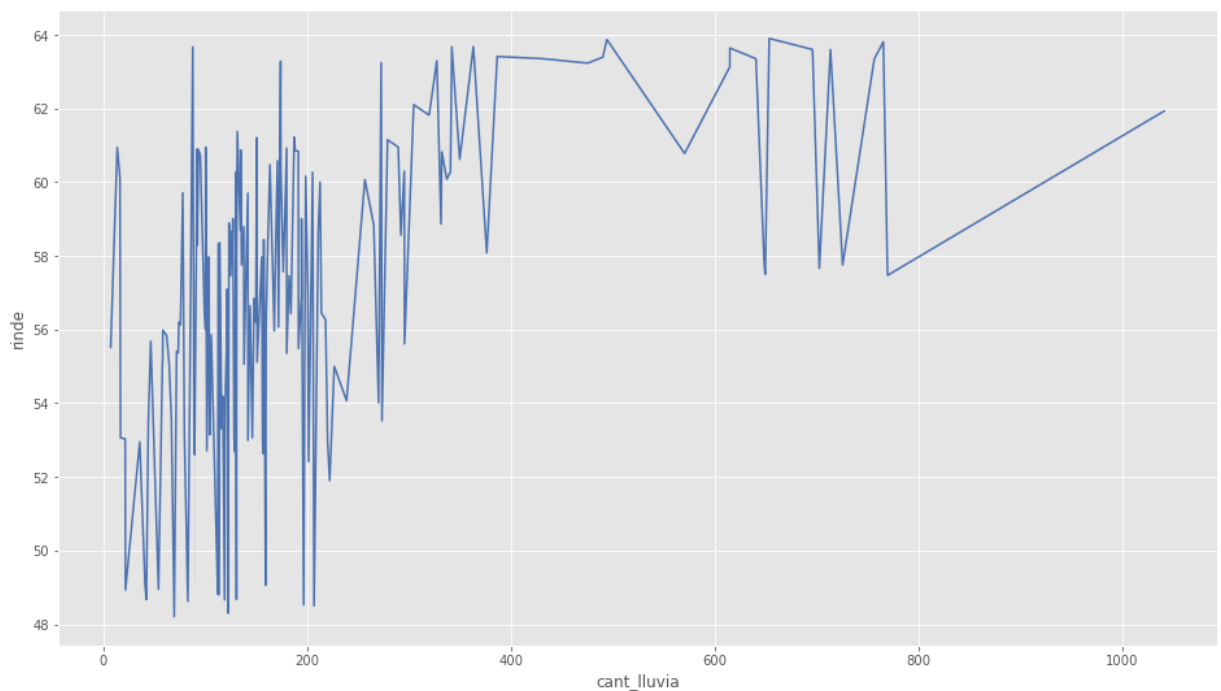
```
In [66]: sns.scatterplot(data=datos, x='cant_lluvia', y='rinde',
hue='departamento')
```

```
Out[66]: <AxesSubplot:xlabel='cant_lluvia', ylabel='rinde'>
```



```
In [67]: sns.lineplot(data=datos, x='cant_lluvia', y='rinde')
```

```
Out[67]: <AxesSubplot:xlabel='cant_lluvia', ylabel='rinde'>
```



En cuanto al rendimiento, tampoco se observa una relación clara con la cantidad de lluvia, aunque sí se puede ver una relación un poco más directa que en la finura de la lana. En este caso, cuando aumenta la

cantidad de lluvia, también aumenta el rendimiento, lo cual es algo que los productores de ganadería ovina buscan. Esto podría significar que la lana está menos sucia y, por lo tanto, rinde más al lavarla. Sin embargo, los datos se mantienen constantes en la mayoría de los departamentos, independientemente de la cantidad de lluvia. En Cushamen y Futaleufú, que son los departamentos con la mayor cantidad de lluvia, se obtiene el mayor rendimiento, alcanzando un máximo de 64.

3.1. Entrenamiento modelo de lana - Dataset 1

Vamos a entrenar un modelo de regresión lineal múltiple con el primer dataset obtenido.

En la siguiente línea de código, se importarán varias bibliotecas que brindan una amplia gama de herramientas para el análisis estadístico y el modelado de datos, así como para la visualización de los resultados.

- Pandas se utiliza para manejar los datos a través de estructuras más flexibles, en este caso, se trabajará con DataFrames.
- Numpy se usará para calcular la métrica RMSE y para ayudar con el gráfico 3D a través de sus funciones linspace para definir el rango de valores para dibujar el plano y meshgrid.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados del entrenamiento mediante gráficos.
- Scikit-learn (sklearn) se usa para crear el modelo lineal. Train_test_split se utilizará para separar los datos en entrenamiento y prueba, y cross_val_score y KFold se utilizarán como formas de medir que el modelo funciona bien y no tiene mucho overfitting. Además, se utilizarán las siguientes métricas: mean_squared_error, r2_score, explained_variance_score y mean_absolute_error.
- Por último, se usará save_pickle para guardar el modelo entrenado.

```
In [1]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score,
explained_variance_score, mean_absolute_error

from statsmodels.iolib.smpickle import save_pickle
```

Lo siguiente que vamos a hacer es recuperar el dataset que tiene los datos por departamento de la producción y clima de Chubut.

```
In [2]: dataset =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut.csv")
dataset
```

```
Out[2]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_la
0	2012-12-31	CHUBUT	BIEDMA	5785	93378	30168	48447	16071	193849	649130

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_la
1	2013-12-31	CHUBUT	BIEDMA	5376	82998	25341	47350	26832	187897	620176
2	2014-12-31	CHUBUT	BIEDMA	4785	82706	29379	35155	29900	181925	649130
3	2015-12-31	CHUBUT	BIEDMA	4440	89262	29663	35437	32343	191145	672480
4	2016-12-31	CHUBUT	BIEDMA	4452	82295	33662	33470	21326	175205	653800
...	...	...	...	...	...	...	...	...	...	...
160	2016-12-31	CHUBUT	TELSEN	4896	80262	26471	39243	20290	171162	637000
161	2017-12-31	CHUBUT	TELSEN	4640	77556	27296	36570	17750	163812	637000
162	2018-12-31	CHUBUT	TELSEN	4832	79117	28366	37558	21933	171806	637000
163	2019-12-31	CHUBUT	TELSEN	4781	80109	28428	38109	17731	169158	613021
164	2020-12-31	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163716	614250

165 rows × 17 columns

Predicción: Kilogramos de Lana

Variables independientes:

- Cantidad total de lluvia acumulada de cada año (mm)
- Cantidad total de ovinos de cada año (animales)

Variables dependientes:

- Kilos de lana producidos

Empezamos extrayendo dos variables del conjunto de datos 'dataset', que se utilizarán posteriormente para la creación del modelo de regresión lineal.

La variable 'X_lluvia_ovinos' se crea seleccionando las columnas 'cant_lluvia' y 'total_ovinos' del conjunto de datos 'dataset', estas columnas se utilizan como variables independientes o predictores del modelo.

La variable 'y_lana' se crea seleccionando la columna 'kilos_lana' del conjunto de datos 'dataset', esta columna se utiliza como variable dependiente o respuesta del modelo.

In [3]:

```
X_lluvia_ovinos = dataset[['cant_lluvia', 'total_ovinos']]
y_lana = dataset[['kilos_lana']]
```

Lo siguiente es dividir los datos en dos conjuntos: uno para entrenamiento y otro para prueba.

- X_train y y_train corresponden a los datos de entrenamiento que se utilizarán para ajustar el modelo.

- `X_test` y `y_test` corresponden a los datos de prueba que se utilizarán para evaluar el rendimiento del modelo.

La función `train_test_split` toma como entrada los datos originales, `X_lluvia_ovinos` y `y_lana`, y divide aleatoriamente los datos en conjuntos de entrenamiento y prueba. El parámetro `test_size` indica el porcentaje de datos que se utilizarán para prueba (en este caso, el 20% de los datos) y `random_state` es una semilla que se utiliza para asegurar que los conjuntos de entrenamiento y prueba sean siempre los mismos en cada ejecución del código.

```
In [4]: X_train, X_test, y_train, y_test = train_test_split(
                                                X_lluvia_ovinos,
                                                y_lana,
                                                test_size=0.20,
                                                random_state=0
                                              )
```

Luego creamos un objeto `KFold` que se utiliza para realizar una validación cruzada en el conjunto de datos. La validación cruzada es una técnica que se utiliza para evaluar el rendimiento del modelo y estimar su capacidad para generalizar a nuevos datos. El objeto `KFold` divide el conjunto de datos en `k` partes iguales (en este caso, `k=5`), y utiliza cada una de ellas como conjunto de prueba y el resto como conjunto de entrenamiento. De esta manera, se obtiene una estimación del rendimiento del modelo en distintas combinaciones de entrenamiento y prueba.

Entonces para entrenar el modelo de manera más efectiva, utilizaremos el método `K-Folds` con 5 divisiones. Este enfoque consiste en iterar 5 veces, donde en cada iteración se separan 33 muestras para utilizarlas como conjunto de prueba, mientras que las 132 restantes se usan para entrenar el modelo. Luego se mide el rendimiento del modelo utilizando el conjunto de prueba apartado en cada iteración. Esto nos permite realizar 5 entrenamientos independientes y obtener 5 resultados de rendimiento. Finalmente, el rendimiento del modelo se calculará como el promedio de los 5 resultados obtenidos. De esta manera, evitamos sobrestimar el rendimiento del modelo al utilizar el mismo conjunto de prueba en todas las iteraciones.

```
In [5]: kf = KFold(n_splits=5)
```

Luego de llevar a cabo estas 5 iteraciones, se espera obtener 5 puntuaciones (score) que sean cercanas entre sí, lo que servirá como **una señal de que el modelo está rindiendo bien y que tiene una buena capacidad de generalización**. Una vez completada esta fase de validación, el siguiente paso es crear el modelo y proceder a su entrenamiento con los datos.

```
In [6]: model_lana = linear_model.LinearRegression()
        model_lana.fit(X_train, y_train)
```

```
Out[6]: ▼ LinearRegression
        LinearRegression()
```

Se procede a comparar los resultados del modelo inicial de entrenamiento con los obtenidos en los 5 procesos de validación cruzada, a través del promedio de estos últimos. Es posible notar que existe una **coincidencia o similitud** entre los resultados. La métrica del modelo es de *0.89*, mientras que las métricas de la validación cruzada son [0.92, 0.90, 0.82, 0.81, 0.84], cuya media es de *0.86*.

```
In [7]: score = model_lana.score(X_train,y_train)
'Métrica del modelo en Train {:.2}'.format(score)
```

```
Out[7]: 'Métrica del modelo en Train 0.89'
```

```
In [8]: scores = cross_val_score(model_lana, X_train, y_train, cv=kf)
scores_rounded = [round(score, 2) for score in scores]
'Métricas cross validation: {}'.format(scores_rounded)
```

```
Out[8]: 'Métricas cross validation: [0.92, 0.9, 0.82, 0.81, 0.84]'
```

```
In [9]: 'Media de cross validation {:.2}'.format(scores.mean())
```

```
Out[9]: 'Media de cross validation 0.86'
```

Finalmente, realizamos una predicción utilizando el conjunto de prueba y verificamos que también obtenemos un **buen puntaje de precisión o score**.

```
In [10]: pred_model_lana = model_lana.predict(X_test)
```

```
In [11]: score_pred = r2_score(y_test, pred_model_lana)
'Metrica del modelo en Test {:.2%}'.format(score_pred)
```

```
Out[11]: 'Metrica del modelo en Test 70.71%'
```

Entonces guardamos el modelo como candidato para utilizarlo en la aplicación web.

```
In [12]: save_pickle(model_lana,'modelos/modelo_prediccion_lana_1')
```

Métricas

Score

Score: El "score" es un método que se incluye en todos los algoritmos y proporciona el resultado de un criterio de evaluación por defecto para el problema al que están dirigidos. En el caso de una regresión lineal, el método score devuelve el coeficiente de determinación, también conocido como **R²**, que se define como el porcentaje de la varianza de la variable dependiente que puede ser explicada por la variable independiente. Esta métrica nos ayuda a evaluar la calidad de nuestro modelo y a compararlo con otros modelos. En general, cuanto mayor sea el valor de **R²**, mejor ajustado estará el modelo a los datos.

```
In [13]: 'Score del modelo en entrenamiento:
{:.2%}'.format(model_lana.score(X_train, y_train))
```



```
Out[13]: 'Score del modelo en entrenamiento: 88.94%'
```

Interpretación: Un R^2 de 0.8894 significa que el modelo explica el 88.94% de la variabilidad de la producción de lana en función de la cantidad de lluvia y el total de ovinos. Es una métrica muy aceptable.

```
In [14]: 'Score del modelo en test: {:.2%}'.format(model_lana.score(X_test,
y_test))
```

```
Out[14]: 'Score del modelo en test: 70.71%'
```

Interpretación: el modelo tiene una precisión del 70.71% al predecir los valores de la variable dependiente (en este caso, los kilos de lana) utilizando los valores de las variables independientes (cantidad de lluvia y número total de ovinos) en el conjunto de datos de prueba. Es decir, el modelo puede explicar aproximadamente el 70.71% de la varianza en la variable dependiente con los datos que no ha visto antes. Es una métrica muy aceptable.

Varianza explicada

Varianza explicada: es una medida del porcentaje de la varianza de la variable de salida que es explicado por el modelo. Esta métrica evalúa la variación o dispersión de los puntos de datos, y la mejor puntuación posible es 1.0, lo que indica que el modelo explica perfectamente toda la variabilidad de los datos. Sin embargo, una puntuación muy alta en esta métrica también puede indicar un riesgo de overfitting, es decir, que el modelo se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a nuevos datos. En sklearn, se puede utilizar el método **explained_variance_score** para calcular esta métrica.

```
In [15]: 'Varianza Explicada:
{:.2%}'.format(explained_variance_score(y_test, pred_model_lana))
```

```
Out[15]: 'Varianza Explicada:70.73%'
```

Interpretación: En este caso, la varianza explicada del modelo de regresión lineal es del 70.73%, lo que indica que el modelo es capaz de explicar el 70.73% de la variabilidad observada en la variable de salida (lana), en base a las variables de entrada proporcionadas (lluvia, total de ovinos). Una varianza explicada del 70.73% indica que el modelo es razonablemente bueno para predecir la variable de salida en función de las variables de entrada sin llegar a indicar un sobreajuste. Es una métrica muy aceptable.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.

```
In [16]: 'MAE: {:.2f} kgs'.format(mean_absolute_error(y_test,
pred_model_lana))
```

```
Out[16]: 'MAE: 210901.05 kgs'
```

Interpretación: El valor obtenido de 210901.05 kgs indica que, en promedio, el modelo se equivoca en esa cantidad al predecir los kilos de lana producidos. Es decir, en promedio, las predicciones del modelo

tienen un error absoluto medio de 210901.05 kgs en relación al valor real de la producción de lana. Una forma de interpretar esta métrica es pensar en el valor de la producción de lana que se está analizando y evaluar si el error medio absoluto es aceptable en relación a ese valor, dado que estamos hablando de toneladas en muchos casos el valor no es tan malo. Es una métrica relativamente buena.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica MSE, la cual es la media de las diferencias entre el valor objetivo y el predicho al cuadrado, y al elevar al cuadrado magnifica los errores grandes. La ventaja de RMSE es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [18]: 'RMSE: {:.2f} kgs'.format(mean_squared_error(y_test,
pred_model_lana, squared = False))
```

```
Out[18]: 'RMSE: 279409.14 kgs'
```

Interpretación: En este caso, la raíz cuadrada del MSE de 279409.14 kgs de lana indica que, en promedio, la diferencia entre los valores predichos y los valores observados de kilos de lana es de aproximadamente 279,409.14 kgs de lana. Es una métrica relativamente buena.

R cuadrado

R cuadrado (R^2): es una métrica utilizada para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo.

Es importante destacar que el resultado del coeficiente de determinación, es un valor que oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto, menos fiable será. Esto se debe a que R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

En resumen, el valor de R^2 nos proporciona una medida de la calidad del ajuste de nuestro modelo a los datos de entrenamiento, siendo un valor cercano a 1 deseable para garantizar una buena capacidad predictiva.

El coeficiente de determinación se puede calcular utilizando la función `r2_score` de la biblioteca sklearn.

```
In [19]: 'R2:{:.2%}'.format(r2_score(y_test, pred_model_lana))
```

```
Out[19]: 'R2:70.71%'
```

Interpretación: En este caso, el valor de R cuadrado es 70.71%, lo que significa que aproximadamente el 70.71% de la variación en los datos de kilos de lana producidos se puede explicar por las variables independientes utilizadas en el modelo de regresión lineal. Entonces el modelo de regresión lineal explica en un 71% la variable de salida, en este caso los kilos de lana. Es una métrica muy buena.

El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas. Es decir, si al modelo se añaden cinco variables explicativas que guardan poca relación con lo que se intenta predecir, el R cuadrado aumentará. Es por ello que muchos expertos econométricos, estadísticos y matemáticos se oponen al uso del R cuadrado como medida representativa de la bondad del ajuste real.

R cuadrado ajustado

La fórmula del R cuadrado ajustado es como sigue: $R_{ajustado}^2 = 1 - \frac{(n-1)}{n-k-1} * (1 - R^2)$

Donde:

- R^2 : coeficiente de determinación
- n : número de observaciones
- k : número de predictores (variables explicativas) en el modelo.

Se va a crear una función que aplique esta fórmula a los datos.

```
In [20]: def r2_ajustado(n,k,r2):  
         return round(1-((n-1)/(n-k-1))*(1-r2),4)  
  
n = len(y_test)  
k = 2  
r2 = r2_score(y_test, pred_model_lana)  
  
'R2 ajustado:{:.2%}'.format(r2_ajustado(n,k,r2))
```

```
Out[20]: 'R2 ajustado:68.76%'
```

Interpretación: un R2 ajustado de 68.76% indica que el modelo explica aproximadamente el 68.76% de la variabilidad en la variable objetivo, teniendo en cuenta las variables explicativas incluidas en el modelo. Es decir, la bondad de ajuste es del 69% y por lo tanto el modelo explica en un 69% los kilos de lana. Es una métrica aceptable.

Gráficos predicción de lana

A continuación, se presentarán dos gráficos para visualizar el rendimiento del modelo de regresión lineal múltiple en la predicción de la producción de lana. En primer lugar, se utilizará un gráfico de dispersión en 3D para representar la relación entre las variables de entrada y la variable de salida. En segundo lugar, se empleará un gráfico de líneas para comparar los valores reales de la producción de lana con los valores predichos por el modelo. Esto permitirá una mejor comprensión del desempeño del modelo y de la precisión de las predicciones.

Gráfico regresión lineal múltiple

Recordemos que las variables, `X_lluvia_ovinos` y `y_lana`, son subconjuntos del dataset original que contienen las columnas "cant_lluvia", "total_ovinos" y "kilos_lana", respectivamente.

Lo siguiente es transformar el dataframe `X_lluvia_ovinos` en un arreglo numpy de dos dimensiones (2D) para poder ser utilizado en el modelo de regresión lineal.

La función `values` de un dataframe de pandas devuelve un arreglo numpy que representa los datos del dataframe, y `reshape` se utiliza para darle una nueva forma o dimensión al arreglo, en este caso convirtiéndolo en una matriz de dos columnas (una para cada variable predictora).

El parámetro `-1` en `reshape` indica a NumPy que calcule el número necesario de filas en función de los datos y de la cantidad de columnas especificadas, de tal manera que los datos originales del dataframe se mantengan.

```
In [21]: X = X_lluvia_ovinos.values.reshape(-1,2)
```

Luego le asignamos a la variable `Y` la columna `'kilos_lana'` del dataframe original dataset. Esto se hace para tener una variable separada que contenga solo la variable de salida (en este caso `'kilos_lana'`) y poder usarla en las operaciones posteriores.

```
In [22]: Y = y_lana
```

La siguiente celda contiene líneas de código que crean una malla de puntos en 2D para hacer una visualización en 3D de la superficie de regresión del modelo.

- `x_pred = np.linspace(7.1, 1041.16, 30)` crea un array de 30 valores igualmente espaciados que van desde 7.1 hasta 1041.16, que representan los posibles valores de la variable `cant_lluvia`.
- `y_pred = np.linspace(82956, 921789, 30)` crea un array de 30 valores igualmente espaciados que van desde 82956 hasta 921789, que representan los posibles valores de la variable `total_ovinos`.
- `xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)` crea una malla de puntos en 2D a partir de los dos arrays anteriores, donde `xx_pred` representa los valores de `cant_lluvia` y `yy_pred` los valores de `total_ovinos`.
- `model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T` crea un array `model_viz` que contiene una lista de todas las posibles combinaciones de valores para `cant_lluvia` y `total_ovinos`, que será usado para hacer la predicción y graficar la superficie de regresión.

```
In [23]: x_pred = np.linspace(7.1, 1041.16, 30) # range of cant_lluvia
y_pred = np.linspace(82956, 921789, 30) # range of total_ovinos
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T
```

La siguiente celda prepara los datos necesarios para visualizar el modelo de regresión lineal múltiple en un gráfico en 3D.

- `x = X[:, 0]` extrae todos los valores de la primera columna de `X` y los asigna a la variable `x`, que representará los valores de la variable independiente `cant_lluvia`.

- `y = X[:, 1]` extrae todos los valores de la segunda columna de `X` y los asigna a la variable `y`, que representará los valores de la variable independiente `total_ovinos`.
- `z = Y` asigna todos los valores de la variable dependiente `kilos_lana` a la variable `z`, que representará la variable que se quiere predecir en función de las variables independientes `x` e `y`.

In [24]:

```
x = X[:, 0]
y = X[:, 1]
z = Y
```

El código `predicted = model_lana.predict(model_viz)` utiliza el modelo de regresión lineal previamente entrenado (`model_lana`) para hacer una predicción de los valores de salida (`kilos_lana`) correspondientes a todas las combinaciones posibles de valores de entrada para `cant_lluvia` y `total_ovinos` que se encuentran en la malla de puntos en 2D (`model_viz`).

Esto se logra utilizando el método `predict` del objeto del modelo de regresión lineal, que toma como entrada una matriz `X` de características y devuelve una matriz de valores de salida predichos correspondientes a cada entrada. En este caso, la matriz `X` es la malla de puntos `model_viz` que contiene todas las posibles combinaciones de valores de `cant_lluvia` y `total_ovinos`. Los valores predichos se almacenan en la variable `predicted`.

In [25]:

```
predicted = model_lana.predict(model_viz)
```

El código `r2 = model_lana.score(X, Y)` calcula el coeficiente de determinación R^2 del modelo de regresión lineal, que como vimos antes, es una medida estadística que indica la proporción de la varianza de la variable dependiente (`kilos_lana`) que es explicada por el modelo en función de las variables independientes (`cant_lluvia` y `total_ovinos`).

Este método toma como entrada una matriz de características `X` (en este caso, la matriz `X` que se creó anteriormente a partir de las variables `cant_lluvia` y `total_ovinos`) y una matriz de valores de salida `Y` (en este caso, la matriz `Y` que contiene los valores reales de `kilos_lana`). Luego, compara los valores de salida predichos por el modelo para cada muestra de `X` con los valores reales correspondientes de `Y`, y devuelve un valor entre menos infinito y 1 que indica la calidad del ajuste del modelo.

Un valor de R^2 más cercano a 1 indica que el modelo explica una gran proporción de la varianza de la variable dependiente, lo que indica un mejor ajuste del modelo a los datos. En este caso, el valor de R^2 obtenido se almacena en la variable `r2`.

In [26]:

```
r2 = model_lana.score(X, Y)
```

El siguiente código crea una figura con tres subplots en 3D utilizando la biblioteca Matplotlib. Cada subplot contiene una visualización de la superficie de regresión lineal para diferentes ángulos de vista.

En cada subplot, se hace una visualización de los datos de entrada y de los valores predichos por el modelo. Los datos de entrada se representan como puntos negros en el fondo, mientras que los valores predichos se representan como puntos azules en la superficie.

También se establecen las etiquetas de los ejes y los parámetros del locator para la ubicación de las etiquetas de los ejes.

Finalmente, se establecen las posiciones de vista de los subplots y se agrega un título a la figura que muestra el valor del coeficiente de determinación (R^2) obtenido del modelo.

La llamada a la función `fig.tight_layout()` se encarga de ajustar automáticamente los espacios entre los subplots para que no haya superposiciones.

In [27]:

```
plt.style.use('default')
fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

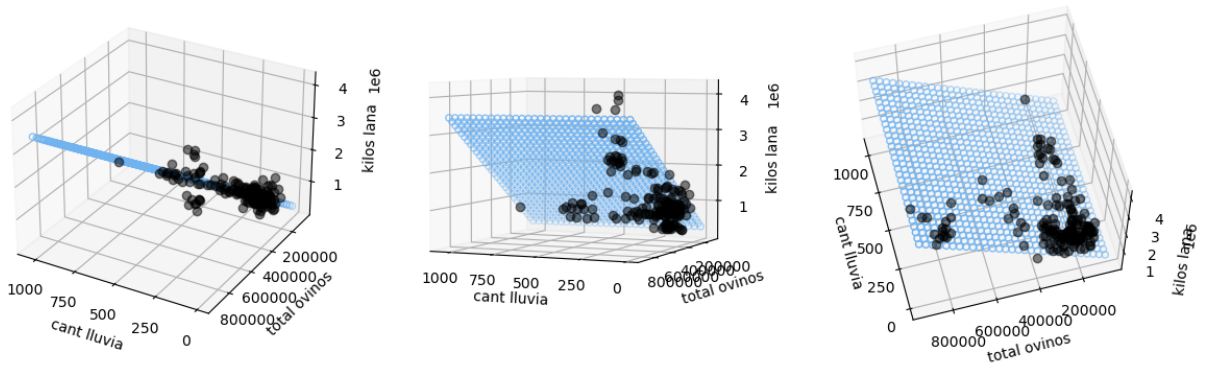
for ax in axes:
    flattenX = x.flatten()
    flattenY = y.flatten()
    flattenZ = z.values.flatten()
    ax.plot(
        flattenX, flattenY, flattenZ, color='k',
        zorder=15, linestyle='none', marker='o', alpha=0.5
    )
    ax.scatter(
        xx_pred.flatten(), yy_pred.flatten(), predicted,
        facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0'
    )
    ax.set_xlabel('cant lluvia', fontsize=10)
    ax.set_ylabel('total ovinos', fontsize=10)
    ax.set_zlabel('kilos lana', fontsize=10)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='y')

ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)

fig.tight_layout()
```

$$R^2 = 0.87$$



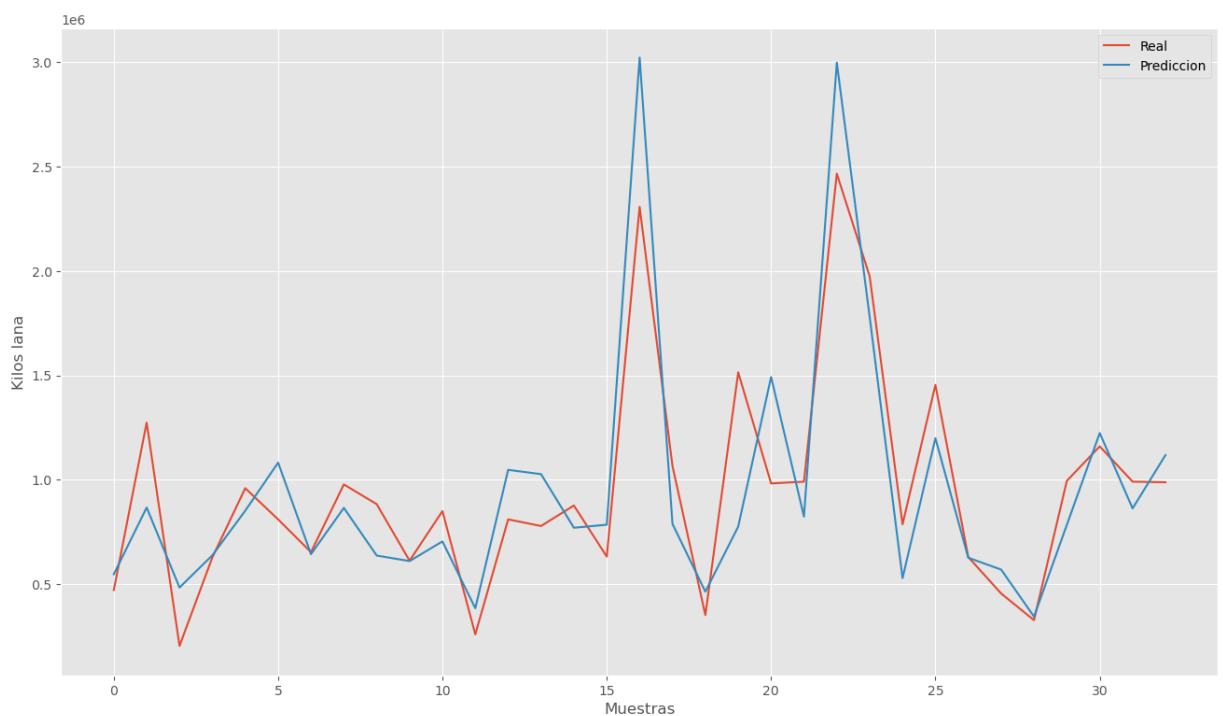
Se observa que los datos no se ajustan perfectamente al plano, lo que va acorde con la información extraída de las métricas.

Gráfico predicción vs real

A continuación, se va a generar un gráfico 2D para comparar los datos reales con la predicción realizada por el modelo.

In [29]:

```
plt.plot(range(len(y_test)),y_test,label="Real")
plt.plot(range(len(pred_model_lana)),pred_model_lana,label="Predicción")
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('Muestras')
plt.ylabel('Kilos lana')
plt.legend()
plt.show()
```



Podemos observar que la línea roja que representa los valores reales y la línea azul que representa la predicción del modelo tienen una tendencia similar, sin embargo, existen algunas desviaciones. Esto

sugiere que el modelo no está sobreajustando los datos de entrenamiento y es más probable que tenga un mejor desempeño al predecir valores para nuevos datos.

3.2. Entrenamiento modelo de lana - Dataset 2

Vamos a entrenar un modelo de regresión lineal múltiple con el segundo dataset obtenido.

En la siguiente línea de código, se importarán varias bibliotecas que brindan una amplia gama de herramientas para el análisis estadístico y el modelado de datos, así como para la visualización de los resultados.

- Pandas se utiliza para manejar los datos a través de estructuras más flexibles, en este caso, se trabajará con DataFrames.
- Numpy se usará para calcular la métrica RMSE y para ayudar con el gráfico 3D a través de sus funciones linspace para definir el rango de valores para dibujar el plano y meshgrid.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados del entrenamiento mediante gráficos.
- Scikit-learn (sklearn) se usa para crear el modelo lineal. Train_test_split se utilizará para separar los datos en entrenamiento y prueba, y cross_val_score y KFold se utilizarán como formas de medir que el modelo funciona bien y no tiene mucho overfitting. Además, se utilizarán las siguientes métricas: mean_squared_error, r2_score, explained_variance_score y mean_absolute_error.
- Por último, se usará save_pickle para guardar el modelo entrenado.

```
In [1]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score,
explained_variance_score, mean_absolute_error

from statsmodels.iolib.smpickle import save_pickle
```

Lo siguiente que vamos a hacer es recuperar el dataset que tiene los datos por departamento de la producción y clima de Chubut, más los datos del Mirabueno.

```
In [2]: dataset =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_mirabueno.csv")
dataset
```

```
Out[2]:
```

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lana
0	2012-12-31	1294000	BIEDMA	5785	93378	30168	48447	16071	193849	649130

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lana
1	2013-12-31	1294000	BIEDMA	5376	82998	25341	47350	26832	187897	620176
2	2014-12-31	1294000	BIEDMA	4785	82706	29379	35155	29900	181925	649130
3	2015-12-31	1294000	BIEDMA	4440	89262	29663	35437	32343	191145	672480
4	2016-12-31	1294000	BIEDMA	4452	82295	33662	33470	21326	175205	653800
...	...	...	...	...	...	...	...	...	...	...
196	2016-12-31	1989300	TELSEN	4896	80262	26471	39243	20290	171162	637000
197	2017-12-31	1989300	TELSEN	4640	77556	27296	36570	17750	163812	637000
198	2018-12-31	1989300	TELSEN	4832	79117	28366	37558	21933	171806	637000
199	2019-12-31	1989300	TELSEN	4781	80109	28428	38109	17731	169158	613021
200	2020-12-31	1989300	TELSEN	2870	79157	25804	36901	18984	163716	614250

201 rows × 11 columns

Predicción: Kilogramos de Lana

Variables independientes:

- Cantidad total de lluvia acumulada de cada año (mm)
- Cantidad total de ovinos de cada año (animales)

Variables dependientes:

- Kilos de lana producidos

Empezamos extrayendo dos variables del conjunto de datos 'dataset', que se utilizarán posteriormente para la creación del modelo de regresión lineal.

La variable 'X_lluvia_ovinos' se crea seleccionando las columnas 'cant_lluvia' y 'total_ovinos' del conjunto de datos 'dataset', estas columnas se utilizan como variables independientes o predictores del modelo.

La variable 'y_lana' se crea seleccionando la columna 'kilos_lana' del conjunto de datos 'dataset', esta columna se utiliza como variable dependiente o respuesta del modelo.

```
In [3]: X_lluvia_ovinos = dataset[['cant_lluvia', 'total_ovinos']]
        y_lana = dataset[['kilos_lana']]
```

Lo siguiente es dividir los datos en dos conjuntos: uno para entrenamiento y otro para prueba.

- X_train y y_train corresponden a los datos de entrenamiento que se utilizarán para ajustar el modelo.

- `X_test` y `y_test` corresponden a los datos de prueba que se utilizarán para evaluar el rendimiento del modelo.

La función `train_test_split` toma como entrada los datos originales, `X_lluvia_ovinos` y `y_lana`, y divide aleatoriamente los datos en conjuntos de entrenamiento y prueba. El parámetro `test_size` indica el porcentaje de datos que se utilizarán para prueba (en este caso, el 20% de los datos) y `random_state` es una semilla que se utiliza para asegurar que los conjuntos de entrenamiento y prueba sean siempre los mismos en cada ejecución del código.

```
In [4]: X_train, X_test, y_train, y_test = train_test_split(
                                             X_lluvia_ovinos,
                                             y_lana,
                                             test_size=0.20,
                                             random_state=0
                                             )
```

Luego creamos un objeto `KFold` que se utiliza para realizar una validación cruzada en el conjunto de datos. La validación cruzada es una técnica que se utiliza para evaluar el rendimiento del modelo y estimar su capacidad para generalizar a nuevos datos. El objeto `KFold` divide el conjunto de datos en `k` partes iguales (en este caso, `k=5`), y utiliza cada una de ellas como conjunto de prueba y el resto como conjunto de entrenamiento. De esta manera, se obtiene una estimación del rendimiento del modelo en distintas combinaciones de entrenamiento y prueba.

Entonces para entrenar el modelo de manera más efectiva, utilizaremos el método **K-Folds** con 5 divisiones. Este enfoque consiste en iterar 5 veces, donde en cada iteración se separan 40 muestras para utilizarlas como conjunto de prueba, mientras que las 161 restantes se usan para entrenar el modelo. Luego se mide el rendimiento del modelo utilizando el conjunto de prueba apartado en cada iteración. Esto nos permite realizar 5 entrenamientos independientes y obtener 5 resultados de rendimiento. Finalmente, el rendimiento del modelo se calculará como el promedio de los 5 resultados obtenidos. De esta manera, evitamos sobrestimar el rendimiento del modelo al utilizar el mismo conjunto de prueba en todas las iteraciones.

```
In [5]: kf = KFold(n_splits=5)
```

Luego de llevar a cabo estas 5 iteraciones, se espera obtener 5 puntuaciones (score) que sean cercanas entre sí, lo que servirá como **una señal de que el modelo está rindiendo bien y que tiene una buena capacidad de generalización**. Una vez completada esta fase de validación, el siguiente paso es crear el modelo y proceder a su entrenamiento con los datos.

```
In [6]: model_lana = linear_model.LinearRegression()
        model_lana.fit(X_train, y_train)
```

```
Out[6]: ▼ LinearRegression
        LinearRegression()
```

Se procede a comparar los resultados del modelo inicial de entrenamiento con los obtenidos en los 5 procesos de validación cruzada, a través del promedio de estos últimos. Es posible notar que existe una **coincidencia o similitud** entre los resultados. La métrica del modelo es de *0.9*, mientras que las métricas de la validación cruzada son [0.89, 0.92, 0.87, 0.85, 0.89], cuya media es de *0.88*.

```
In [7]: score = model_lana.score(X_train,y_train)
'Métrica del modelo en Train {:.2}'.format(score)
```

```
Out[7]: 'Métrica del modelo en Train 0.9'
```

```
In [8]: scores = cross_val_score(model_lana, X_train, y_train, cv=kf)
scores_rounded = [round(score, 2) for score in scores]
'Métricas cross validation: {}'.format(scores_rounded)
```

```
Out[8]: 'Métricas cross validation: [0.89, 0.92, 0.87, 0.85, 0.89]'
```

```
In [9]: 'Media de cross validation {:.2}'.format(scores.mean())
```

```
Out[9]: 'Media de cross validation 0.88'
```

Finalmente, realizamos una predicción utilizando el conjunto de prueba y verificamos que también obtenemos un **buen puntaje de precisión o score**.

```
In [10]: pred_model_lana = model_lana.predict(X_test)
```

```
In [11]: score_pred = r2_score(y_test, pred_model_lana)
'Metrica del modelo en Test {:.2%}'.format(score_pred)
```

```
Out[11]: 'Metrica del modelo en Test 92.34%'
```

Entonces guardamos el modelo como candidato para utilizarlo en la aplicación web.

```
In [12]: save_pickle(model_lana,'modelos/modelo_prediccion_lana_2')
```

Métricas

Score

Score: El "score" es un método que se incluye en todos los algoritmos y proporciona el resultado de un criterio de evaluación por defecto para el problema al que están dirigidos. En el caso de una regresión lineal, el método score devuelve el coeficiente de determinación, también conocido como **R²**, que se define como el porcentaje de la varianza de la variable dependiente que puede ser explicada por la variable independiente. Esta métrica nos ayuda a evaluar la calidad de nuestro modelo y a compararlo con otros modelos. En general, cuanto mayor sea el valor de **R²**, mejor ajustado estará el modelo a los datos.

```
In [13]: 'Score del modelo en entrenamiento:
{:.2%}'.format(model_lana.score(X_train, y_train))
```

```
Out[13]: 'Score del modelo en entrenamiento: 89.58%'
```

Interpretación: Un R^2 de 0.8958 significa que el modelo explica el 88.94% de la variabilidad de la producción de lana en función de la cantidad de lluvia y el total de ovinos. Es una métrica muy buena.

```
In [14]: 'Score del modelo en test: {:.2%}'.format(model_lana.score(X_test,
y_test))
```

```
Out[14]: 'Score del modelo en test: 92.34%'
```

Interpretación: el modelo tiene una precisión del 92.34% al predecir los valores de la variable dependiente (en este caso, los kilos de lana) utilizando los valores de las variables independientes (cantidad de lluvia y número total de ovinos) en el conjunto de datos de prueba. Es decir, el modelo puede explicar aproximadamente el 92.34% de la varianza en la variable dependiente con los datos que no ha visto antes. Es una métrica excelente.

Varianza explicada

Varianza explicada: es una medida del porcentaje de la varianza de la variable de salida que es explicado por el modelo. Esta métrica evalúa la variación o dispersión de los puntos de datos, y la mejor puntuación posible es 1.0, lo que indica que el modelo explica perfectamente toda la variabilidad de los datos. Sin embargo, una puntuación muy alta en esta métrica también puede indicar un riesgo de overfitting, es decir, que el modelo se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a nuevos datos. En sklearn, se puede utilizar el método **explained_variance_score** para calcular esta métrica.

```
In [15]: 'Varianza Explicada:
{:.2%}'.format(explained_variance_score(y_test, pred_model_lana))
```

```
Out[15]: 'Varianza Explicada:92.36%'
```

Interpretación: En este caso, la varianza explicada del modelo de regresión lineal es del 92.36%, lo que indica que el modelo es capaz de explicar el 92.36% de la variabilidad observada en la variable de salida (lana), en base a las variables de entrada proporcionadas (lluvia, total de ovinos). Una varianza explicada del 92.36% indica que el modelo es bueno para predecir la variable de salida en función de las variables de entrada sin llegar a indicar un sobreajuste. Es una métrica excelente.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.

```
In [16]: 'MAE: {:.2f} kgs'.format(mean_absolute_error(y_test,
pred_model_lana))
```

```
Out[16]: 'MAE: 195870.43 kgs'
```

Interpretación: El valor obtenido de 195870.43 kgs indica que, en promedio, el modelo se equivoca en esa cantidad al predecir los kilos de lana producidos. Es decir, en promedio, las predicciones del modelo

tienen un error absoluto medio de 195870.43 kgs en relación al valor real de la producción de lana. Una forma de interpretar esta métrica es pensar en el valor de la producción de lana que se está analizando y evaluar si el error medio absoluto es aceptable en relación a ese valor, dado que estamos hablando de toneladas en muchos casos el valor no es tan malo. Es una métrica relativamente buena.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica MSE, la cual es la media de las diferencias entre el valor objetivo y el predicho al cuadrado, y al elevar al cuadrado magnifica los errores grandes. La ventaja de RMSE es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [18]: 'RMSE: {:.2f} kgs'.format(mean_squared_error(y_test,
pred_model_lana, squared = False))
```

```
Out[18]: 'RMSE: 280099.96 kgs'
```

Interpretación: En este caso, la raíz cuadrada del MSE de 280099.96 kgs de lana indica que, en promedio, la diferencia entre los valores predichos y los valores observados de kilos de lana es de aproximadamente 280099.96 kgs de lana. Es una métrica relativamente buena.

R cuadrado

R cuadrado (R^2): es una métrica utilizada para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo.

Es importante destacar que el resultado del coeficiente de determinación, es un valor que oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto, menos fiable será. Esto se debe a que R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

En resumen, el valor de R^2 nos proporciona una medida de la calidad del ajuste de nuestro modelo a los datos de entrenamiento, siendo un valor cercano a 1 deseable para garantizar una buena capacidad predictiva.

El coeficiente de determinación se puede calcular utilizando la función `r2_score` de la biblioteca sklearn.

```
In [19]: 'R2:{:.2%}'.format(r2_score(y_test, pred_model_lana))
```

```
Out[19]: 'R2:92.34%'
```

Interpretación: En este caso, el valor de R cuadrado es 92.34%, lo que significa que aproximadamente el 92.34% de la variación en los datos de kilos de lana producidos se puede explicar por las variables independientes utilizadas en el modelo de regresión lineal. Entonces el modelo de regresión lineal explica en un 92% la variable de salida, en este caso los kilos de lana. Es una métrica excelente.

El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas. Es decir, si al modelo se añaden cinco variables explicativas que guardan poca relación con lo que se intenta predecir, el R cuadrado aumentará. Es por ello que muchos expertos econométricos, estadísticos y matemáticos se oponen al uso del R cuadrado como medida representativa de la bondad del ajuste real.

R cuadrado ajustado

La fórmula del R cuadrado ajustado es como sigue: $R_{ajustado}^2 = 1 - \frac{(n-1)}{n-k-1} * (1 - R^2)$

Donde:

- R^2 : coeficiente de determinación
- n : número de observaciones
- k : número de predictores (variables explicativas) en el modelo.

Se va a crear una función que aplique esta fórmula a los datos.

In [20]:

```
def r2_ajustado(n,k,r2):  
    return round(1-((n-1)/(n-k-1))*(1-r2),4)  
  
n = len(y_test)  
k = 2  
r2 = r2_score(y_test, pred_model_lana)  
  
'R2 ajustado:{:.2%}'.format(r2_ajustado(n,k,r2))
```

Out[20]: 'R2 ajustado:91.94%'

Interpretación: un R2 ajustado de 91.94% indica que el modelo explica aproximadamente el 91.94% de la variabilidad en la variable objetivo, teniendo en cuenta las variables explicativas incluidas en el modelo. Es decir, la bondad de ajuste es del 92% y por lo tanto el modelo explica en un 92% los kilos de lana. Es una métrica excelente.

Gráficos predicción de lana

A continuación, se presentarán dos gráficos para visualizar el rendimiento del modelo de regresión lineal múltiple en la predicción de la producción de lana. En primer lugar, se utilizará un gráfico de dispersión en 3D para representar la relación entre las variables de entrada y la variable de salida. En segundo lugar, se empleará un gráfico de líneas para comparar los valores reales de la producción de lana con los valores predichos por el modelo. Esto permitirá una mejor comprensión del desempeño del modelo y de la precisión de las predicciones.

Gráfico regresión lineal múltiple

Recordemos que las variables, `X_lluvia_ovinos` y `y_lana`, son subconjuntos del dataset original que contienen las columnas "cant_lluvia", "total_ovinos" y "kilos_lana", respectivamente.

Lo siguiente es transformar el dataframe `X_lluvia_ovinos` en un arreglo numpy de dos dimensiones (2D) para poder ser utilizado en el modelo de regresión lineal.

La función `values` de un dataframe de pandas devuelve un arreglo numpy que representa los datos del dataframe, y `reshape` se utiliza para darle una nueva forma o dimensión al arreglo, en este caso convirtiéndolo en una matriz de dos columnas (una para cada variable predictora).

El parámetro `-1` en `reshape` indica a NumPy que calcule el número necesario de filas en función de los datos y de la cantidad de columnas especificadas, de tal manera que los datos originales del dataframe se mantengan.

```
In [21]: X = X_lluvia_ovinos.values.reshape(-1,2)
```

Luego le asignamos a la variable `Y` la columna `'kilos_lana'` del dataframe original dataset. Esto se hace para tener una variable separada que contenga solo la variable de salida (en este caso `'kilos_lana'`) y poder usarla en las operaciones posteriores.

```
In [22]: Y = y_lana
```

La siguiente celda contiene líneas de código que crean una malla de puntos en 2D para hacer una visualización en 3D de la superficie de regresión del modelo.

- `x_pred = np.linspace(7.1, 1041.16, 30)` crea un array de 30 valores igualmente espaciados que van desde 7.1 hasta 1041.16, que representan los posibles valores de la variable `cant_lluvia`.
- `y_pred = np.linspace(1771, 921789, 30)` crea un array de 30 valores igualmente espaciados que van desde 1771 hasta 921789, que representan los posibles valores de la variable `total_ovinos`.
- `xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)` crea una malla de puntos en 2D a partir de los dos arrays anteriores, donde `xx_pred` representa los valores de `cant_lluvia` y `yy_pred` los valores de `total_ovinos`.
- `model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T` crea un array `model_viz` que contiene una lista de todas las posibles combinaciones de valores para `cant_lluvia` y `total_ovinos`, que será usado para hacer la predicción y graficar la superficie de regresión.

```
In [30]: x_pred = np.linspace(7.1, 1041.16, 30) # range of cant_lluvia
y_pred = np.linspace(1771, 921789, 30) # range of total_ovinos
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T
```

La siguiente celda prepara los datos necesarios para visualizar el modelo de regresión lineal múltiple en un gráfico en 3D.

- `x = X[:, 0]` extrae todos los valores de la primera columna de `X` y los asigna a la variable `x`, que representará los valores de la variable independiente `cant_lluvia`.

- `y = X[:, 1]` extrae todos los valores de la segunda columna de `X` y los asigna a la variable `y`, que representará los valores de la variable independiente `total_ovinos`.
- `z = Y` asigna todos los valores de la variable dependiente `kilos_lana` a la variable `z`, que representará la variable que se quiere predecir en función de las variables independientes `x` e `y`.

In [31]:

```
x = X[:, 0]
y = X[:, 1]
z = Y
```

El código `predicted = model_lana.predict(model_viz)` utiliza el modelo de regresión lineal previamente entrenado (`model_lana`) para hacer una predicción de los valores de salida (`kilos_lana`) correspondientes a todas las combinaciones posibles de valores de entrada para `cant_lluvia` y `total_ovinos` que se encuentran en la malla de puntos en 2D (`model_viz`).

Esto se logra utilizando el método `predict` del objeto del modelo de regresión lineal, que toma como entrada una matriz `X` de características y devuelve una matriz de valores de salida predichos correspondientes a cada entrada. En este caso, la matriz `X` es la malla de puntos `model_viz` que contiene todas las posibles combinaciones de valores de `cant_lluvia` y `total_ovinos`. Los valores predichos se almacenan en la variable `predicted`.

In [32]:

```
predicted = model_lana.predict(model_viz)
```

El código `r2 = model_lana.score(X, Y)` calcula el coeficiente de determinación R^2 del modelo de regresión lineal, que como vimos antes, es una medida estadística que indica la proporción de la varianza de la variable dependiente (`kilos_lana`) que es explicada por el modelo en función de las variables independientes (`cant_lluvia` y `total_ovinos`).

Este método toma como entrada una matriz de características `X` (en este caso, la matriz `X` que se creó anteriormente a partir de las variables `cant_lluvia` y `total_ovinos`) y una matriz de valores de salida `Y` (en este caso, la matriz `Y` que contiene los valores reales de `kilos_lana`). Luego, compara los valores de salida predichos por el modelo para cada muestra de `X` con los valores reales correspondientes de `Y`, y devuelve un valor entre menos infinito y 1 que indica la calidad del ajuste del modelo.

Un valor de R^2 más cercano a 1 indica que el modelo explica una gran proporción de la varianza de la variable dependiente, lo que indica un mejor ajuste del modelo a los datos. En este caso, el valor de R^2 obtenido se almacena en la variable `r2`.

In [33]:

```
r2 = model_lana.score(X, Y)
```

El siguiente código crea una figura con tres subplots en 3D utilizando la biblioteca Matplotlib. Cada subplot contiene una visualización de la superficie de regresión lineal para diferentes ángulos de vista.

En cada subplot, se hace una visualización de los datos de entrada y de los valores predichos por el modelo. Los datos de entrada se representan como puntos negros en el fondo, mientras que los valores predichos se representan como puntos azules en la superficie.

También se establecen las etiquetas de los ejes y los parámetros del locator para la ubicación de las etiquetas de los ejes.

Finalmente, se establecen las posiciones de vista de los subplots y se agrega un título a la figura que muestra el valor del coeficiente de determinación (R^2) obtenido del modelo.

La llamada a la función `fig.tight_layout()` se encarga de ajustar automáticamente los espacios entre los subplots para que no haya superposiciones.

In [34]:

```
plt.style.use('default')
fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

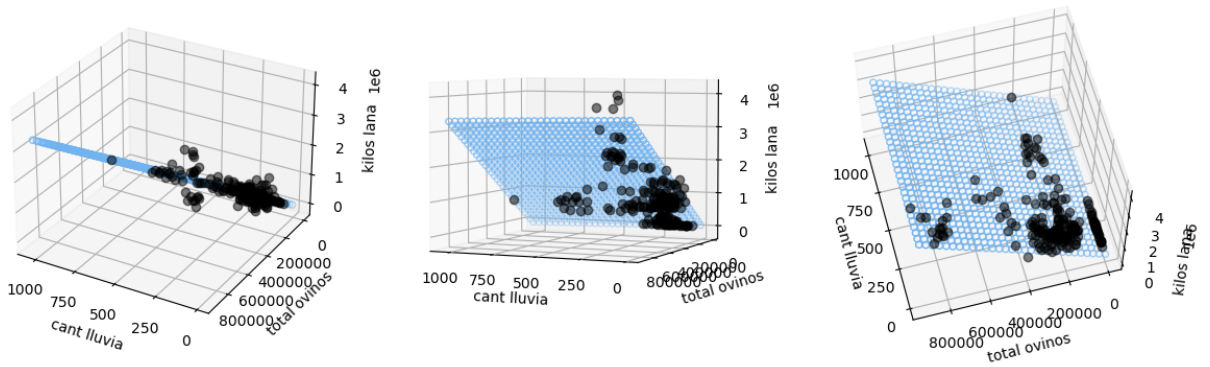
for ax in axes:
    flattenX = x.flatten()
    flattenY = y.flatten()
    flattenZ = z.values.flatten()
    ax.plot(
        flattenX, flattenY, flattenZ, color='k',
        zorder=15, linestyle='none', marker='o', alpha=0.5
    )
    ax.scatter(
        xx_pred.flatten(), yy_pred.flatten(), predicted,
        facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0'
    )
    ax.set_xlabel('cant lluvia', fontsize=10)
    ax.set_ylabel('total ovinos', fontsize=10)
    ax.set_zlabel('kilos lana', fontsize=10)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)

fig.tight_layout()
```

$$R^2 = 0.91$$

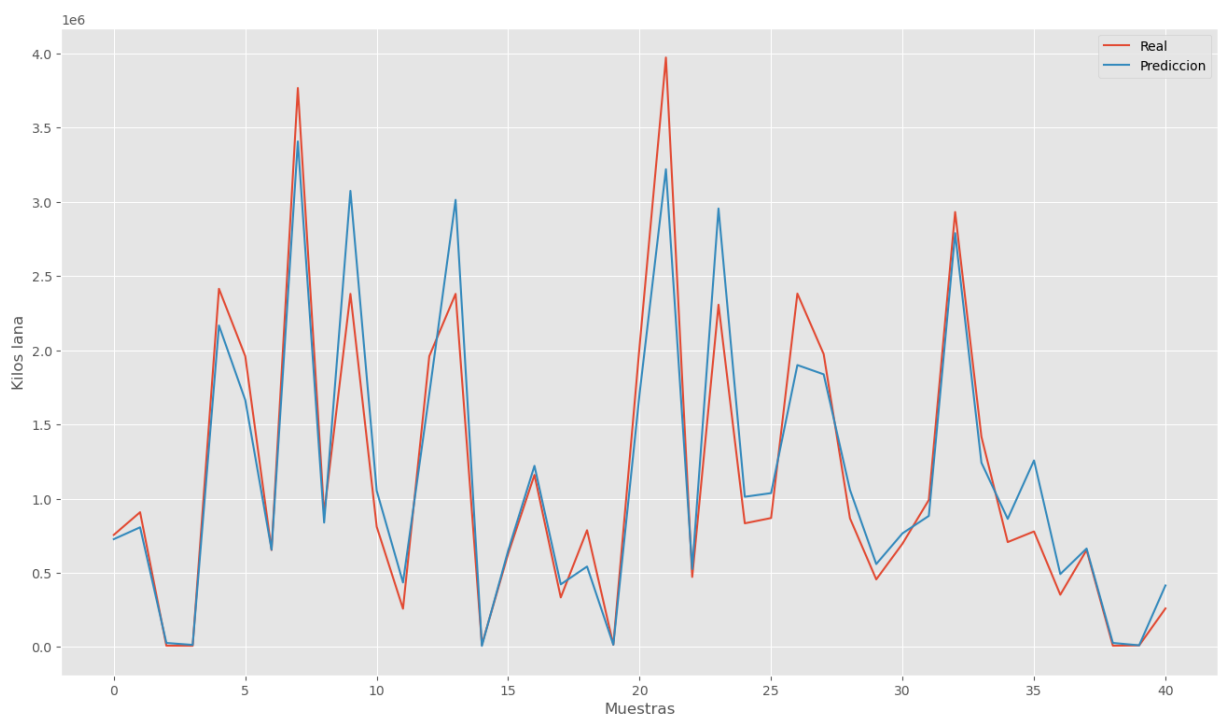


Se observa que los datos no se ajustan perfectamente al plano, lo que va acorde con la información extraída de las métricas.

Gráfico predicción vs real

A continuación, se va a generar un gráfico 2D para comparar los datos reales con la predicción realizada por el modelo.

```
In [29]: plt.plot(range(len(y_test)),y_test,label="Real")
plt.plot(range(len(pred_model_lana)),pred_model_lana,label="Predicción")
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('Muestras')
plt.ylabel('Kilos lana')
plt.legend()
plt.show()
```



Podemos observar que la línea roja que representa los valores reales y la línea azul que representa la predicción del modelo tienen una tendencia similar, pero conservan unas pequeñas desviaciones. Esto

sugiere que el modelo no está sobreajustando los datos de entrenamiento y es más probable que tenga un mejor desempeño al predecir valores para nuevos datos.

3.3. Entrenamiento modelo de lana - Dataset 2 escalado

Vamos a entrenar un modelo de regresión lineal múltiple con el segundo dataset obtenido, pero escalando los datos, dado que como estamos trabajando con un campo y con departamentos los valores son muy distintos.

En la siguiente línea de código, se importarán varias bibliotecas que brindan una amplia gama de herramientas para el análisis estadístico y el modelado de datos, así como para la visualización de los resultados.

- Pandas se utiliza para manejar los datos a través de estructuras más flexibles, en este caso, se trabajará con DataFrames.
- Numpy se usará para calcular la métrica RMSE y para ayudar con el gráfico 3D a través de sus funciones linspace para definir el rango de valores para dibujar el plano y meshgrid.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados del entrenamiento mediante gráficos.
- Scikit-learn (sklearn) se usa para crear el modelo lineal. Train_test_split se utilizará para separar los datos en entrenamiento y prueba, y cross_val_score y KFold se utilizarán como formas de medir que el modelo funciona bien y no tiene mucho overfitting. También se utilizarán otras dos clases que son comunes en el procesamiento de datos, una de ellas es el ColumnTransformer, que es una clase que permite aplicar transformaciones específicas a columnas particulares de un conjunto de datos y es útil cuando se trabaja con datos heterogéneos que requieren diferentes preprocesamientos para diferentes características, la otra es StandardScaler que es una clase que se utiliza para estandarizar los datos, es decir, escalar los datos para que tengan una media de cero y una desviación estándar de uno. Además, se utilizarán las siguientes métricas: mean_squared_error, r2_score, explained_variance_score y mean_absolute_error.
- Por último, se usará save_pickle para guardar el modelo entrenado.

```
In [1]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import linear_model
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, r2_score,
explained_variance_score, mean_absolute_error
```

```
from statsmodels.iolib.smpickle import save_pickle
```

Lo siguiente que vamos a hacer es recuperar el dataset que tiene los datos por departamento de la producción y clima de Chubut, más los datos del Mirabueno.

```
In [2]: dataset =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_mirab
dataset
```

```
Out[2]:
```

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lan
0	2012-12-31	1294000	BIEDMA	5785	93378	30168	48447	16071	193849	649130
1	2013-12-31	1294000	BIEDMA	5376	82998	25341	47350	26832	187897	620176
2	2014-12-31	1294000	BIEDMA	4785	82706	29379	35155	29900	181925	649130
3	2015-12-31	1294000	BIEDMA	4440	89262	29663	35437	32343	191145	672480
4	2016-12-31	1294000	BIEDMA	4452	82295	33662	33470	21326	175205	653800
...	...	...	...	...	...	...	...	...	...	...
196	2016-12-31	1989300	TELSEN	4896	80262	26471	39243	20290	171162	637000
197	2017-12-31	1989300	TELSEN	4640	77556	27296	36570	17750	163812	637000
198	2018-12-31	1989300	TELSEN	4832	79117	28366	37558	21933	171806	637000
199	2019-12-31	1989300	TELSEN	4781	80109	28428	38109	17731	169158	613021
200	2020-12-31	1989300	TELSEN	2870	79157	25804	36901	18984	163716	614250

201 rows × 11 columns

Predicción: Kilogramos de Lana

Variables independientes:

- Cantidad total de lluvia acumulada de cada año (mm)
- Cantidad total de ovinos de cada año (animales)

Variables dependientes:

- Kilos de lana producidos

Empezamos extrayendo dos variables del conjunto de datos 'dataset', que se utilizarán posteriormente para la creación del modelo de regresión lineal.

La variable 'X_lluvia_ovinos' se crea definiendo una transformación llamada "numeric" que estandariza (con `StandardScaler()`) las columnas "cant_lluvia" y "total_ovinos" del conjunto de datos dataset. Después se aplica la transformación mediante el método `fit_transform()` del objeto pipeline a dataset y se almacena el resultado en la variable `X_lluvia_ovinos`.

La variable 'y_lana' se crea normalizando la variable objetivo 'kilos_lana' utilizando la clase `StandardScaler()` de Scikit-learn. Primero se crea una instancia del objeto `StandardScaler()`. Luego, se seleccionan los valores de la variable objetivo de la base de datos en la línea `l_values = dataset[['kilos_lana']]`. Posteriormente, se aplica el método `fit()` de la instancia del objeto `StandardScaler()` a los valores seleccionados en la línea anterior para ajustar los parámetros de la normalización. Finalmente, se aplica la normalización a los valores seleccionados utilizando el método `transform()` de la instancia del objeto `StandardScaler()`, y se almacena el resultado en `y_lana`.

```
In [3]: num_attrs = ['cant_lluvia', 'total_ovinos']
        pipeline = ColumnTransformer([("numeric", StandardScaler(),
        num_attrs)])
        X_lluvia_ovinos = pipeline.fit_transform(dataset)

        standard_scaler = StandardScaler()
        l_values = dataset[['kilos_lana']]
        scaled_values = standard_scaler.fit(l_values)
        y_lana = standard_scaler.transform(l_values)
```

Lo siguiente es dividir los datos en dos conjuntos: uno para entrenamiento y otro para prueba.

- `X_train` y `y_train` corresponden a los datos de entrenamiento que se utilizarán para ajustar el modelo.
- `X_test` y `y_test` corresponden a los datos de prueba que se utilizarán para evaluar el rendimiento del modelo.

La función `train_test_split` toma como entrada los datos originales, `X_lluvia_ovinos` y `y_lana`, y divide aleatoriamente los datos en conjuntos de entrenamiento y prueba. El parámetro `test_size` indica el porcentaje de datos que se utilizarán para prueba (en este caso, el 20% de los datos) y `random_state` es una semilla que se utiliza para asegurar que los conjuntos de entrenamiento y prueba sean siempre los mismos en cada ejecución del código.

```
In [4]: X_train, X_test, y_train, y_test = train_test_split(
        X_lluvia_ovinos,
        y_lana,
        test_size=0.20,
        random_state=0
        )
```

Luego creamos un objeto `KFold` que se utiliza para realizar una validación cruzada en el conjunto de datos. La validación cruzada es una técnica que se utiliza para evaluar el rendimiento del modelo y

estimar su capacidad para generalizar a nuevos datos. El objeto `KFold` divide el conjunto de datos en k partes iguales (en este caso, $k=5$), y utiliza cada una de ellas como conjunto de prueba y el resto como conjunto de entrenamiento. De esta manera, se obtiene una estimación del rendimiento del modelo en distintas combinaciones de entrenamiento y prueba.

Entonces para entrenar el modelo de manera más efectiva, utilizaremos el método `K-Folds` con 5 divisiones. Este enfoque consiste en iterar 5 veces, donde en cada iteración se separan 40 muestras para utilizarlas como conjunto de prueba, mientras que las 161 restantes se usan para entrenar el modelo. Luego se mide el rendimiento del modelo utilizando el conjunto de prueba apartado en cada iteración. Esto nos permite realizar 5 entrenamientos independientes y obtener 5 resultados de rendimiento. Finalmente, el rendimiento del modelo se calculará como el promedio de los 5 resultados obtenidos. De esta manera, evitamos sobrestimar el rendimiento del modelo al utilizar el mismo conjunto de prueba en todas las iteraciones.

```
In [5]: kf = KFold(n_splits=5)
```

Luego de llevar a cabo estas 5 iteraciones, se espera obtener 5 puntuaciones (score) que sean cercanas entre sí, lo que servirá como **una señal de que el modelo está rindiendo bien y que tiene una buena capacidad de generalización**. Una vez completada esta fase de validación, el siguiente paso es crear el modelo y proceder a su entrenamiento con los datos.

```
In [6]: model_lana = linear_model.LinearRegression()  
model_lana.fit(X_train, y_train)
```

```
Out[6]: ▼ LinearRegression  
LinearRegression()
```

Se procede a comparar los resultados del modelo inicial de entrenamiento con los obtenidos en los 5 procesos de validación cruzada, a través del promedio de estos últimos. Es posible notar que existe una **coincidencia o similitud** entre los resultados. La métrica del modelo es de 0.9 , mientras que las métricas de la validación cruzada son $[0.89, 0.92, 0.87, 0.85, 0.89]$, cuya media es de 0.88 .

```
In [7]: score = model_lana.score(X_train, y_train)  
'Métrica del modelo en Train {:.2}'.format(score)
```

```
Out[7]: 'Métrica del modelo en Train 0.9'
```

```
In [8]: scores = cross_val_score(model_lana, X_train, y_train, cv=kf)  
scores_rounded = [round(score, 2) for score in scores]  
'Métricas cross validation: {}'.format(scores_rounded)
```

```
Out[8]: 'Métricas cross validation: [0.89, 0.92, 0.87, 0.85, 0.89]'
```

```
In [9]: 'Media de cross validation {:.2}'.format(scores.mean())
```

```
Out[9]: 'Media de cross validation 0.88'
```


Finalmente, realizamos una predicción utilizando el conjunto de prueba y verificamos que también obtenemos un **buen puntaje de precisión o score**.

```
In [10]: pred_model_lana = model_lana.predict(X_test)
```

```
In [11]: score_pred = r2_score(y_test, pred_model_lana)
'Metrica del modelo en Test {:.2%}'.format(score_pred)
```

```
Out[11]: 'Metrica del modelo en Test 92.34%'
```

Entonces guardamos el modelo como candidato para utilizarlo en la aplicación web.

```
In [12]: save_pickle(model_lana, 'modelos/modelo_prediccion_lana_2_e')
```

Métricas

Score

Score: El "score" es un método que se incluye en todos los algoritmos y proporciona el resultado de un criterio de evaluación por defecto para el problema al que están dirigidos. En el caso de una regresión lineal, el método score devuelve el coeficiente de determinación, también conocido como R^2 , que se define como el porcentaje de la varianza de la variable dependiente que puede ser explicada por la variable independiente. Esta métrica nos ayuda a evaluar la calidad de nuestro modelo y a compararlo con otros modelos. En general, cuanto mayor sea el valor de R^2 , mejor ajustado estará el modelo a los datos.

```
In [13]: 'Score del modelo en entrenamiento:
{:.2%}'.format(model_lana.score(X_train, y_train))
```

```
Out[13]: 'Score del modelo en entrenamiento: 89.58%'
```

Interpretación: Un R^2 de 0.8894 significa que el modelo explica el 89.58% de la variabilidad de la producción de lana en función de la cantidad de lluvia y el total de ovinos. Es una métrica muy buena.

```
In [14]: 'Score del modelo en test: {:.2%}'.format(model_lana.score(X_test,
y_test))
```

```
Out[14]: 'Score del modelo en test: 92.34%'
```

Interpretación: el modelo tiene una precisión del 92.34% al predecir los valores de la variable dependiente (en este caso, los kilos de lana) utilizando los valores de las variables independientes (cantidad de lluvia y número total de ovinos) en el conjunto de datos de prueba. Es decir, el modelo puede explicar aproximadamente el 92.34% de la varianza en la variable dependiente con los datos que no ha visto antes. Es una métrica excelente.

Varianza explicada

Varianza explicada: es una medida del porcentaje de la varianza de la variable de salida que es explicado por el modelo. Esta métrica evalúa la variación o dispersión de los puntos de datos, y la mejor puntuación posible es 1.0, lo que indica que el modelo explica perfectamente toda la variabilidad de los datos. Sin embargo, una puntuación muy alta en esta métrica también puede indicar un riesgo de overfitting, es

decir, que el modelo se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a nuevos datos. En sklearn, se puede utilizar el método **explained_variance_score** para calcular esta métrica.

```
In [15]: 'Varianza Explicada:
{: .2%}'.format(explained_variance_score(y_test, pred_model_lana))
```

```
Out[15]: 'Varianza Explicada:92.36%'
```

Interpretación: En este caso, la varianza explicada del modelo de regresión lineal es del 92.36%, lo que indica que el modelo es capaz de explicar el 92.36% de la variabilidad observada en la variable de salida (lana), en base a las variables de entrada proporcionadas (lluvia, total de ovinos). Una varianza explicada del 92.36% indica que el modelo es muy bueno para predecir la variable de salida en función de las variables de entrada sin llegar a indicar un sobreajuste. Es una métrica excelente.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.

```
In [16]: 'MAE: {:.2f}'.format(mean_absolute_error(y_test, pred_model_lana))
```

```
Out[16]: 'MAE: 0.24'
```

Interpretación: El valor obtenido de 0.24 kgs indica que, en promedio, el modelo se equivoca en esa cantidad al predecir los kilos de lana producidos. Es decir, en promedio, las predicciones del modelo tienen un error absoluto medio de 0.24 kgs en relación al valor real de la producción de lana. Recordemos que los valores están estandarizados acá, lo que dificulta su interpretación.

Error cuadrático medio

Error cuadrático medio (MSE): Es la media de las diferencias entre el valor objetivo y el predicho al cuadrado. Al elevar al cuadrado los errores, magnifica los errores grandes, por lo que hay que utilizarla con cuidado cuando tenemos valores anómalos en nuestro conjunto de datos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_squared_error**.

```
In [17]: 'MSE: {:.2f} kgs'.format(mean_squared_error(y_test,
pred_model_lana))
```

```
Out[17]: 'MSE: 0.12 kgs'
```

Interpretación: En este caso, el MSE de 0.12 kgs de lana indica que, en promedio, la diferencia entre los valores predichos y los valores observados de kilos de lana es de aproximadamente ese valor. Recordemos que los valores están estandarizados acá, lo que dificulta su interpretación.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica MSE, la cual es la media de las diferencias entre el valor objetivo y el predicho al cuadrado, y al elevar al cuadrado magnifica los errores grandes. La ventaja de RMSE es que presenta el error en las mismas

unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [18]: 'RMSE: {:.2f} kgs'.format(mean_squared_error(y_test,
pred_model_lana, squared = False))
```

```
Out[18]: 'RMSE: 0.34 kgs'
```

Interpretación: En este caso, la raíz cuadrada del MSE de 0.34 kgs de lana indica que, en promedio, la diferencia entre los valores predichos y los valores observados de kilos de lana es de aproximadamente 0.34 kgs de lana. Recordemos que los valores están estandarizados acá, lo que dificulta su interpretación.

R cuadrado

R cuadrado (R^2): es una métrica utilizada para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo.

Es importante destacar que el resultado del coeficiente de determinación, es un valor que oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto, menos fiable será. Esto se debe a que R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

En resumen, el valor de R^2 nos proporciona una medida de la calidad del ajuste de nuestro modelo a los datos de entrenamiento, siendo un valor cercano a 1 deseable para garantizar una buena capacidad predictiva.

El coeficiente de determinación se puede calcular utilizando la función `r2_score` de la biblioteca sklearn.

```
In [19]: 'R2:{:.2%}'.format(r2_score(y_test, pred_model_lana))
```

```
Out[19]: 'R2:92.34%'
```

Interpretación: En este caso, el valor de R cuadrado es 92.34%, lo que significa que aproximadamente el 92.34% de la variación en los datos de kilos de lana producidos se puede explicar por las variables independientes utilizadas en el modelo de regresión lineal. Entonces el modelo de regresión lineal explica en un 92% la variable de salida, en este caso los kilos de lana. Es una métrica excelente.

El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas. Es decir, si al modelo se añaden cinco variables explicativas que guardan poca relación con lo que se intenta predecir, el R cuadrado aumentará. Es por ello que muchos expertos econométricos, estadísticos y matemáticos se oponen al uso del R cuadrado como medida representativa de la bondad del ajuste real.

R cuadrado ajustado

La fórmula del R cuadrado ajustado es como sigue: $R_{ajustado}^2 = 1 - \frac{(n-1)}{n-k-1} * (1 - R^2)$

Donde:

- R^2 : coeficiente de determinación
- n : número de observaciones
- k : número de predictores (variables explicativas) en el modelo.

Se va a crear una función que aplique esta fórmula a los datos.

In [20]:

```
def r2_ajustado(n,k,r2):  
    return round(1-((n-1)/(n-k-1))*(1-r2),4)  
  
n = len(y_test)  
k = 2  
r2 = r2_score(y_test, pred_model_lana)  
  
'R2 ajustado:{:.2%}'.format(r2_ajustado(n,k,r2))
```

Out[20]: 'R2 ajustado:91.94%'

Interpretación: un R2 ajustado de 91.94% indica que el modelo explica aproximadamente el 91.94% de la variabilidad en la variable objetivo, teniendo en cuenta las variables explicativas incluidas en el modelo. Es decir, la bondad de ajuste es del 92% y por lo tanto el modelo explica en un 92% los kilos de lana. Es una métrica excelente.

Gráficos predicción de lana

A continuación, se presentarán dos gráficos para visualizar el rendimiento del modelo de regresión lineal múltiple en la predicción de la producción de lana. En primer lugar, se utilizará un gráfico de dispersión en 3D para representar la relación entre las variables de entrada y la variable de salida. En segundo lugar, se empleará un gráfico de líneas para comparar los valores reales de la producción de lana con los valores predichos por el modelo. Esto permitirá una mejor comprensión del desempeño del modelo y de la precisión de las predicciones.

Gráfico regresión lineal múltiple

Recordemos que las variables, `X_lluvia_ovinos` y `y_lana`, son subconjuntos del dataset original que contienen las columnas "cant_lluvia", "total_ovinos" y "kilos_lana", respectivamente.

Lo siguiente es transformar el dataframe `X_lluvia_ovinos` en un arreglo numpy de dos dimensiones (2D) para poder ser utilizado en el modelo de regresión lineal.

La función `values` de un dataframe de pandas devuelve un arreglo numpy que representa los datos del dataframe, y `reshape` se utiliza para darle una nueva forma o dimensión al arreglo, en este caso convirtiéndolo en una matriz de dos columnas (una para cada variable predictora).

El parámetro `-1` en `reshape` indica a NumPy que calcule el número necesario de filas en función de los datos y de la cantidad de columnas especificadas, de tal manera que los datos originales del dataframe se mantengan.

In [21]:

```
X = X_lluvia_ovinos.reshape(-1,2)
```

Luego le asignamos a la variable `Y` la columna `'kilos_lana'` del dataframe original dataset. Esto se hace para tener una variable separada que contenga solo la variable de salida (en este caso `'kilos_lana'`) y poder usarla en las operaciones posteriores.

```
In [22]: Y = y_lana
```

La siguiente celda contiene líneas de código que crean una malla de puntos en 2D para hacer una visualización en 3D de la superficie de regresión del modelo.

- `x_pred = np.linspace(-0.50, 5.45, 30)` crea un array de 30 valores igualmente espaciados que van desde -0.50 hasta 5.45, que representan los posibles valores de la variable `cant_lluvia` estandarizados.
- `y_pred = np.linspace(-1.10, 3.30, 30)` crea un array de 30 valores igualmente espaciados que van desde -1.10 hasta 3.30, que representan los posibles valores de la variable `total_ovinos` estandarizados.
- `xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)` crea una malla de puntos en 2D a partir de los dos arrays anteriores, donde `xx_pred` representa los valores de `cant_lluvia` y `yy_pred` los valores de `total_ovinos`.
- `model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T` crea un array `model_viz` que contiene una lista de todas las posibles combinaciones de valores para `cant_lluvia` y `total_ovinos`, que será usado para hacer la predicción y graficar la superficie de regresión.

```
In [23]: x_pred = np.linspace(-0.50, 5.45, 30) # range of cant_lluvia
y_pred = np.linspace(-1.10, 3.30, 30) # range of total_ovinos
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T
```

La siguiente celda prepara los datos necesarios para visualizar el modelo de regresión lineal múltiple en un gráfico en 3D.

- `x = X[:, 0]` extrae todos los valores de la primera columna de `X` y los asigna a la variable `x`, que representará los valores de la variable independiente `cant_lluvia`.
- `y = X[:, 1]` extrae todos los valores de la segunda columna de `X` y los asigna a la variable `y`, que representará los valores de la variable independiente `total_ovinos`.
- `z = Y` asigna todos los valores de la variable dependiente `kilos_lana` a la variable `z`, que representará la variable que se quiere predecir en función de las variables independientes `x` e `y`.

```
In [24]: x = X[:, 0]
y = X[:, 1]
z = Y
```

El código `predicted = model_lana.predict(model_viz)` utiliza el modelo de regresión lineal previamente entrenado (`model_lana`) para hacer una predicción de los valores de salida (`kilos_lana`) correspondientes a todas las combinaciones posibles de valores de entrada para `cant_lluvia` y `total_ovinos` que se encuentran en la malla de puntos en 2D (`model_viz`).

Esto se logra utilizando el método `predict` del objeto del modelo de regresión lineal, que toma como entrada una matriz `X` de características y devuelve una matriz de valores de salida predichos correspondientes a cada entrada. En este caso, la matriz `X` es la malla de puntos `model_viz` que contiene todas las posibles combinaciones de valores de `cant_lluvia` y `total_ovinos`. Los valores predichos se almacenan en la variable `predicted`.

```
In [25]: predicted = model_lana.predict(model_viz)
```

El código `r2 = model_lana.score(X, Y)` calcula el coeficiente de determinación R^2 del modelo de regresión lineal, que como vimos antes, es una medida estadística que indica la proporción de la varianza de la variable dependiente (`kilos_lana`) que es explicada por el modelo en función de las variables independientes (`cant_lluvia` y `total_ovinos`).

Este método toma como entrada una matriz de características `X` (en este caso, la matriz `X` que se creó anteriormente a partir de las variables `cant_lluvia` y `total_ovinos`) y una matriz de valores de salida `Y` (en este caso, la matriz `Y` que contiene los valores reales de `kilos_lana`). Luego, compara los valores de salida predichos por el modelo para cada muestra de `X` con los valores reales correspondientes de `Y`, y devuelve un valor entre menos infinito y 1 que indica la calidad del ajuste del modelo.

Un valor de R^2 más cercano a 1 indica que el modelo explica una gran proporción de la varianza de la variable dependiente, lo que indica un mejor ajuste del modelo a los datos. En este caso, el valor de R^2 obtenido se almacena en la variable `r2`.

```
In [26]: r2 = model_lana.score(X, Y)
```

El siguiente código crea una figura con tres subplots en 3D utilizando la biblioteca Matplotlib. Cada subplot contiene una visualización de la superficie de regresión lineal para diferentes ángulos de vista.

En cada subplot, se hace una visualización de los datos de entrada y de los valores predichos por el modelo. Los datos de entrada se representan como puntos negros en el fondo, mientras que los valores predichos se representan como puntos azules en la superficie.

También se establecen las etiquetas de los ejes y los parámetros del locator para la ubicación de las etiquetas de los ejes.

Finalmente, se establecen las posiciones de vista de los subplots y se agrega un título a la figura que muestra el valor del coeficiente de determinación (R^2) obtenido del modelo.

La llamada a la función `fig.tight_layout()` se encarga de ajustar automáticamente los espacios entre los subplots para que no haya superposiciones.

```
In [27]: plt.style.use('default')
fig = plt.figure(figsize=(12, 4))
```

```

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

for ax in axes:
    flattenX = x.flatten()
    flattenY = y.flatten()
    flattenZ = z.flatten()
    ax.plot(
        flattenX, flattenY, flattenZ, color='k',
        zorder=15, linestyle='none', marker='o', alpha=0.5
    )
    ax.scatter(
        xx_pred.flatten(), yy_pred.flatten(), predicted,
        facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0'
    )
    ax.set_xlabel('cant lluvia', fontsize=10)
    ax.set_ylabel('total ovinos', fontsize=10)
    ax.set_zlabel('kilos lana', fontsize=10)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='y')

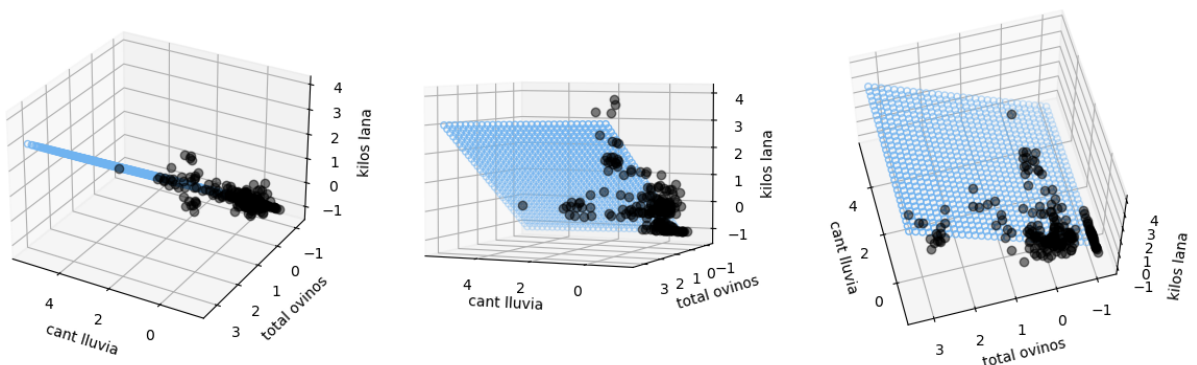
ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)

fig.tight_layout()

```

$$R^2 = 0.91$$

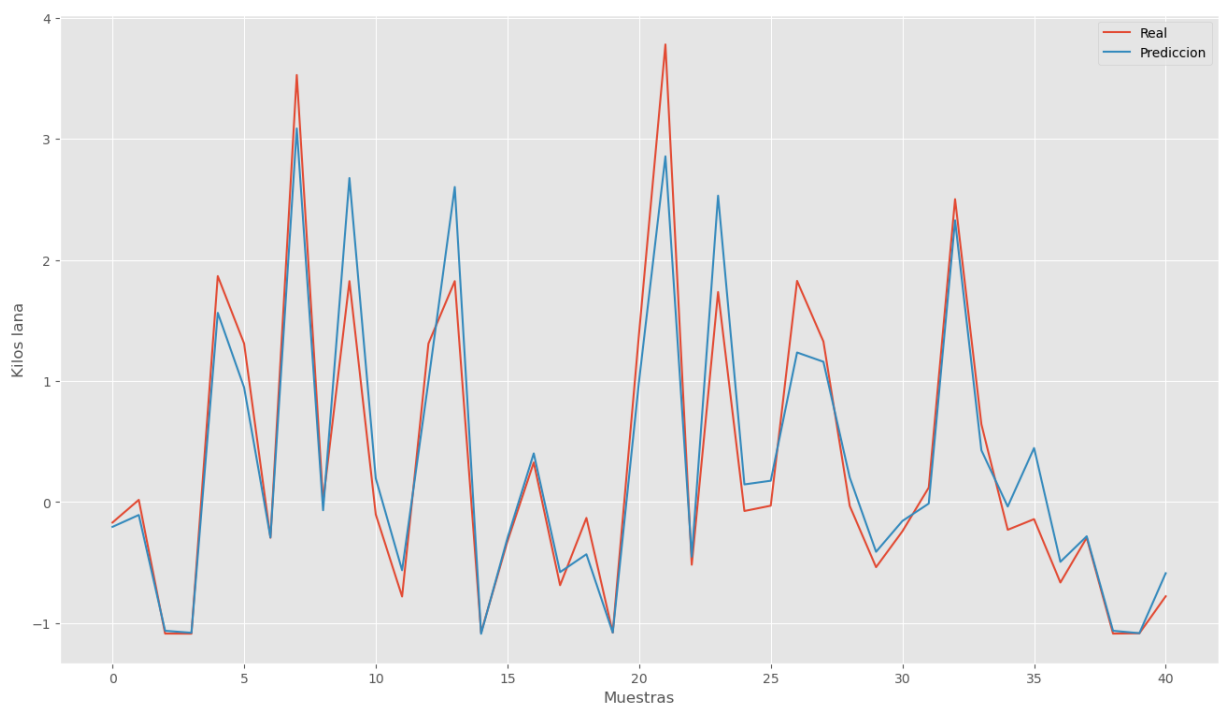


Se observa que los datos no se ajustan perfectamente al plano, lo que va acorde con la información extraída de las métricas.

Gráfico predicción vs real

A continuación, se va a generar un gráfico 2D para comparar los datos reales con la predicción realizada por el modelo.

```
In [29]: plt.plot(range(len(y_test)),y_test,label="Real")
plt.plot(range(len(pred_model_lana)),pred_model_lana,label="Predicci
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('Muestras')
plt.ylabel('Kilos lana')
plt.legend()
plt.show()
```



Podemos observar que la línea roja que representa los valores reales y la línea azul que representa la predicción del modelo tienen una tendencia similar, pero existen algunas desviaciones pequeñas. Esto sugiere que el modelo no está sobreajustando los datos de entrenamiento y es más probable que tenga un mejor desempeño al predecir valores para nuevos datos.

3.4. Entrenamiento modelo de lana - Dataset 3

Vamos a entrenar un modelo de regresión lineal múltiple con el tercer dataset obtenido.

En la siguiente línea de código, se importarán varias bibliotecas que brindan una amplia gama de herramientas para el análisis estadístico y el modelado de datos, así como para la visualización de los resultados.

- Pandas se utiliza para manejar los datos a través de estructuras más flexibles, en este caso, se trabajará con DataFrames.
- Numpy se usará para calcular la métrica RMSE y para ayudar con el gráfico 3D a través de sus funciones linspace para definir el rango de valores para dibujar el plano y meshgrid.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados del entrenamiento mediante gráficos.
- Scikit-learn (sklearn) se usa para crear el modelo lineal. Train_test_split se utilizará para separar los datos en entrenamiento y prueba, y cross_val_score y KFold se utilizarán como formas de medir que el modelo funciona bien y no tiene mucho overfitting. Además, se utilizarán las siguientes métricas: mean_squared_error, r2_score, explained_variance_score y mean_absolute_error.
- Por último, se usará save_pickle para guardar el modelo entrenado.

```
In [1]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score,
explained_variance_score, mean_absolute_error

from statsmodels.iolib.smpickle import save_pickle
```

Lo siguiente que vamos a hacer es recuperar el dataset que tiene los datos por departamento de la producción y clima de Chubut, actualizado con los nuevos datos obtenidos de las encuestas.

```
In [2]: dataset =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_actua
dataset
```

```
Out[2]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2012-12-31	CHUBUT	BIEDMA	2740	52320	22319	25489	29517	132385

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
1	2013-12-31	CHUBUT	BIEDMA	3925	70677	27227	33389	36961	172179
2	2014-12-31	CHUBUT	BIEDMA	3223	61195	29491	29688	34926	158523
3	2015-12-31	CHUBUT	BIEDMA	3885	67528	28319	33296	38117	171145
4	2016-12-31	CHUBUT	BIEDMA	3743	70451	30050	33038	37923	175205
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	TELSEN	4711	74125	28715	31728	31883	171162
160	2017-12-31	CHUBUT	TELSEN	4496	71149	27574	28606	31987	163812
161	2018-12-31	CHUBUT	TELSEN	4634	74179	28793	35979	28221	171806
162	2019-12-31	CHUBUT	TELSEN	4305	72515	27648	35273	29417	169158
163	2020-12-31	CHUBUT	TELSEN	4416	70736	27084	31418	30062	163716

164 rows × 10 columns

Predicción: Kilogramos de Lana

Variables independientes:

- Cantidad total de lluvia acumulada de cada año (mm)
- Cantidad total de ovinos de cada año (animales)

Variables dependientes:

- Kilos de lana producidos

Empezamos extrayendo dos variables del conjunto de datos 'dataset', que se utilizarán posteriormente para la creación del modelo de regresión lineal.

La variable 'X_lluvia_ovinos' se crea seleccionando las columnas 'cant_lluvia' y 'total_ovinos' del conjunto de datos 'dataset', estas columnas se utilizan como variables independientes o predictores del modelo.

La variable 'y_lana' se crea seleccionando la columna 'kilos_lana' del conjunto de datos 'dataset', esta columna se utiliza como variable dependiente o respuesta del modelo.

```
In [3]: X_lluvia_ovinos = dataset[['cant_lluvia', 'total_ovinos']]
        y_lana = dataset[['kilos_lana']]
```

Lo siguiente es dividir los datos en dos conjuntos: uno para entrenamiento y otro para prueba.

- X_train y y_train corresponden a los datos de entrenamiento que se utilizarán para ajustar el modelo.

- `X_test` y `y_test` corresponden a los datos de prueba que se utilizarán para evaluar el rendimiento del modelo.

La función `train_test_split` toma como entrada los datos originales, `X_lluvia_ovinos` y `y_lana`, y divide aleatoriamente los datos en conjuntos de entrenamiento y prueba. El parámetro `test_size` indica el porcentaje de datos que se utilizarán para prueba (en este caso, el 20% de los datos) y `random_state` es una semilla que se utiliza para asegurar que los conjuntos de entrenamiento y prueba sean siempre los mismos en cada ejecución del código.

```
In [4]: X_train, X_test, y_train, y_test = train_test_split(
                                                X_lluvia_ovinos,
                                                y_lana,
                                                test_size=0.20,
                                                random_state=0
                                              )
```

Luego creamos un objeto `KFold` que se utiliza para realizar una validación cruzada en el conjunto de datos. La validación cruzada es una técnica que se utiliza para evaluar el rendimiento del modelo y estimar su capacidad para generalizar a nuevos datos. El objeto `KFold` divide el conjunto de datos en `k` partes iguales (en este caso, `k=5`), y utiliza cada una de ellas como conjunto de prueba y el resto como conjunto de entrenamiento. De esta manera, se obtiene una estimación del rendimiento del modelo en distintas combinaciones de entrenamiento y prueba.

Entonces para entrenar el modelo de manera más efectiva, utilizaremos el método **K-Folds** con 5 divisiones. Este enfoque consiste en iterar 5 veces, donde en cada iteración se separan 33 muestras para utilizarlas como conjunto de prueba, mientras que las 131 restantes se usan para entrenar el modelo. Luego se mide el rendimiento del modelo utilizando el conjunto de prueba apartado en cada iteración. Esto nos permite realizar 5 entrenamientos independientes y obtener 5 resultados de rendimiento. Finalmente, el rendimiento del modelo se calculará como el promedio de los 5 resultados obtenidos. De esta manera, evitamos sobrestimar el rendimiento del modelo al utilizar el mismo conjunto de prueba en todas las iteraciones.

```
In [5]: kf = KFold(n_splits=5)
```

Luego de llevar a cabo estas 5 iteraciones, se espera obtener 5 puntuaciones (score) que sean cercanas entre sí, lo que servirá como **una señal de que el modelo está rindiendo bien y que tiene una buena capacidad de generalización**. Una vez completada esta fase de validación, el siguiente paso es crear el modelo y proceder a su entrenamiento con los datos.

```
In [6]: model_lana = linear_model.LinearRegression()
        model_lana.fit(X_train, y_train)
```

```
Out[6]: ▼ LinearRegression
        LinearRegression()
```

Se procede a comparar los resultados del modelo inicial de entrenamiento con los obtenidos en los 5 procesos de validación cruzada, a través del promedio de estos últimos. Es posible notar que existe una **coincidencia o similitud** entre los resultados. La métrica del modelo es de *0.98*, mientras que las métricas de la validación cruzada son [0.99, 0.99, 0.98, 0.96, 0.97], cuya media es de *0.98*.

```
In [7]: score = model_lana.score(X_train,y_train)
'Métrica del modelo en Train {:.2}'.format(score)
```

```
Out[7]: 'Métrica del modelo en Train 0.98'
```

```
In [8]: scores = cross_val_score(model_lana, X_train, y_train, cv=kf)
scores_rounded = [round(score, 2) for score in scores]
'Métricas cross validation: {}'.format(scores_rounded)
```

```
Out[8]: 'Métricas cross validation: [0.99, 0.99, 0.98, 0.96, 0.97]'
```

```
In [9]: 'Media de cross validation {:.2}'.format(scores.mean())
```

```
Out[9]: 'Media de cross validation 0.98'
```

Finalmente, realizamos una predicción utilizando el conjunto de prueba y verificamos que también obtenemos un **buen puntaje de precisión o score**.

```
In [10]: pred_model_lana = model_lana.predict(X_test)
```

```
In [11]: score_pred = r2_score(y_test, pred_model_lana)
'Metrica del modelo en Test {:.2%}'.format(score_pred)
```

```
Out[11]: 'Metrica del modelo en Test 98.95%'
```

Entonces guardamos el modelo como candidato para utilizarlo en la aplicación web.

```
In [12]: save_pickle(model_lana,'modelos/modelo_prediccion_lana_3')
```

Métricas

Score

Score: El "score" es un método que se incluye en todos los algoritmos y proporciona el resultado de un criterio de evaluación por defecto para el problema al que están dirigidos. En el caso de una regresión lineal, el método score devuelve el coeficiente de determinación, también conocido como **R²**, que se define como el porcentaje de la varianza de la variable dependiente que puede ser explicada por la variable independiente. Esta métrica nos ayuda a evaluar la calidad de nuestro modelo y a compararlo con otros modelos. En general, cuanto mayor sea el valor de **R²**, mejor ajustado estará el modelo a los datos.

```
In [13]: 'Score del modelo en entrenamiento:
{:.2%}'.format(model_lana.score(X_train, y_train))
```

```
Out[13]: 'Score del modelo en entrenamiento: 98.48%'
```

Interpretación: Un R^2 de 0.9848 significa que el modelo explica el 98.48% de la variabilidad de la producción de lana en función de la cantidad de lluvia y el total de ovinos. Es una métrica demasiado buena.

```
In [14]: 'Score del modelo en test: {:.2%}'.format(model_lana.score(X_test,
y_test))
```

```
Out[14]: 'Score del modelo en test: 98.95%'
```

Interpretación: el modelo tiene una precisión del 98.95% al predecir los valores de la variable dependiente (en este caso, los kilos de lana) utilizando los valores de las variables independientes (cantidad de lluvia y número total de ovinos) en el conjunto de datos de prueba. Es decir, el modelo puede explicar aproximadamente el 98.95% de la varianza en la variable dependiente con los datos que no ha visto antes. Es una métrica demasiado buena.

Varianza explicada

Varianza explicada: es una medida del porcentaje de la varianza de la variable de salida que es explicado por el modelo. Esta métrica evalúa la variación o dispersión de los puntos de datos, y la mejor puntuación posible es 1.0, lo que indica que el modelo explica perfectamente toda la variabilidad de los datos. Sin embargo, una puntuación muy alta en esta métrica también puede indicar un riesgo de overfitting, es decir, que el modelo se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a nuevos datos. En sklearn, se puede utilizar el método **explained_variance_score** para calcular esta métrica.

```
In [15]: 'Varianza Explicada:
{:.2%}'.format(explained_variance_score(y_test, pred_model_lana))
```

```
Out[15]: 'Varianza Explicada:99.01%'
```

Interpretación: En este caso, la varianza explicada del modelo de regresión lineal es del 99.01%, lo que indica que el modelo es capaz de explicar el 99.01% de la variabilidad observada en la variable de salida (lana), en base a las variables de entrada proporcionadas (lluvia, total de ovinos). Una varianza explicada del 99.01% indica que el modelo puede tener un sobreajuste. Es una métrica demasiado alta.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.

```
In [16]: 'MAE: {:.2f} kgs'.format(mean_absolute_error(y_test,
pred_model_lana))
```

```
Out[16]: 'MAE: 45551.29 kgs'
```

Interpretación: El valor obtenido de 45551.29 kgs indica que, en promedio, el modelo se equivoca en esa cantidad al predecir los kilos de lana producidos. Es decir, en promedio, las predicciones del modelo

tienen un error absoluto medio de 45551.29 kgs en relación al valor real de la producción de lana. Una forma de interpretar esta métrica es pensar en el valor de la producción de lana que se está analizando y evaluar si el error medio absoluto es aceptable en relación a ese valor, dado que estamos hablando de toneladas en muchos casos el valor es muy bueno. Es una métrica excelente.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica MSE, la cual es la media de las diferencias entre el valor objetivo y el predicho al cuadrado, y al elevar al cuadrado magnifica los errores grandes. La ventaja de RMSE es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [18]: 'RMSE: {:.2f} kgs'.format(mean_squared_error(y_test,
pred_model_lana, squared = False))
```

```
Out[18]: 'RMSE: 67750.91 kgs'
```

Interpretación: En este caso, la raíz cuadrada del MSE de 67750.91 kgs de lana indica que, en promedio, la diferencia entre los valores predichos y los valores observados de kilos de lana es de aproximadamente 67750.91 kgs de lana. Que es un valor muy bueno.

R cuadrado

R cuadrado (R^2): es una métrica utilizada para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo.

Es importante destacar que el resultado del coeficiente de determinación, es un valor que oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto, menos fiable será. Esto se debe a que R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

En resumen, el valor de R^2 nos proporciona una medida de la calidad del ajuste de nuestro modelo a los datos de entrenamiento, siendo un valor cercano a 1 deseable para garantizar una buena capacidad predictiva.

El coeficiente de determinación se puede calcular utilizando la función `r2_score` de la biblioteca sklearn.

```
In [19]: 'R2:{:.2%}'.format(r2_score(y_test, pred_model_lana))
```

```
Out[19]: 'R2:98.95%'
```

Interpretación: En este caso, el valor de R cuadrado es 98.95%, lo que significa que aproximadamente el 98.95% de la variación en los datos de kilos de lana producidos se puede explicar por las variables independientes utilizadas en el modelo de regresión lineal. Entonces el modelo de regresión lineal explica en un 99% la variable de salida, en este caso los kilos de lana. Es una métrica demasiado buena.

El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas. Es decir, si al modelo se añaden cinco variables explicativas que guardan poca relación con lo que se intenta predecir, el R cuadrado aumentará. Es por ello que muchos expertos econométricos, estadísticos y matemáticos se oponen al uso del R cuadrado como medida representativa de la bondad del ajuste real.

R cuadrado ajustado

La fórmula del R cuadrado ajustado es como sigue: $R_{ajustado}^2 = 1 - \frac{(n-1)}{n-k-1} * (1 - R^2)$

Donde:

- R^2 : coeficiente de determinación
- n : número de observaciones
- k : número de predictores (variables explicativas) en el modelo.

Se va a crear una función que aplique esta fórmula a los datos.los datos.

In [20]:

```
def r2_ajustado(n,k,r2):  
    return 1-((n-1)/(n-k-1))*(1-r2)  
  
n = len(y_test)  
k = 2  
r2 = r2_score(y_test, pred_model_lana)  
  
'R2 ajustado:{:.2%}'.format(r2_ajustado(n,k,r2))
```

Out[20]: 'R2 ajustado:98.88%'

Interpretación: un R2 ajustado de 98.88% indica que el modelo explica aproximadamente el 98.88% de la variabilidad en la variable objetivo, teniendo en cuenta las variables explicativas incluidas en el modelo. Es decir, la bondad de ajuste es del 99% y por lo tanto el modelo explica en un 99% los kilos de lana. Es una métrica demasiado buena.

Gráficos predicción de lana

A continuación, se presentarán dos gráficos para visualizar el rendimiento del modelo de regresión lineal múltiple en la predicción de la producción de lana. En primer lugar, se utilizará un gráfico de dispersión en 3D para representar la relación entre las variables de entrada y la variable de salida. En segundo lugar, se empleará un gráfico de líneas para comparar los valores reales de la producción de lana con los valores predichos por el modelo. Esto permitirá una mejor comprensión del desempeño del modelo y de la precisión de las predicciones.

Gráfico regresión lineal múltiple

Recordemos que las variables, `X_lluvia_ovinos` y `y_lana`, son subconjuntos del dataset original que contienen las columnas "cant_lluvia", "total_ovinos" y "kilos_lana", respectivamente.

Lo siguiente es transformar el dataframe `X_lluvia_ovinos` en un arreglo numpy de dos dimensiones (2D) para poder ser utilizado en el modelo de regresión lineal.

La función `values` de un dataframe de pandas devuelve un arreglo numpy que representa los datos del dataframe, y `reshape` se utiliza para darle una nueva forma o dimensión al arreglo, en este caso convirtiéndolo en una matriz de dos columnas (una para cada variable predictora).

El parámetro `-1` en `reshape` indica a NumPy que calcule el número necesario de filas en función de los datos y de la cantidad de columnas especificadas, de tal manera que los datos originales del dataframe se mantengan.

```
In [21]: X = X_lluvia_ovinos.values.reshape(-1,2)
```

Luego le asignamos a la variable `Y` la columna `'kilos_lana'` del dataframe original dataset. Esto se hace para tener una variable separada que contenga solo la variable de salida (en este caso `'kilos_lana'`) y poder usarla en las operaciones posteriores.

```
In [22]: Y = y_lana
```

La siguiente celda contiene líneas de código que crean una malla de puntos en 2D para hacer una visualización en 3D de la superficie de regresión del modelo.

- `x_pred = np.linspace(7.1, 1041.16, 30)` crea un array de 30 valores igualmente espaciados que van desde 7.1 hasta 1041.16, que representan los posibles valores de la variable `cant_lluvia`.
- `y_pred = np.linspace(50513, 797790, 30)` crea un array de 30 valores igualmente espaciados que van desde 50513 hasta 797790, que representan los posibles valores de la variable `total_ovinos`.
- `xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)` crea una malla de puntos en 2D a partir de los dos arrays anteriores, donde `xx_pred` representa los valores de `cant_lluvia` y `yy_pred` los valores de `total_ovinos`.
- `model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T` crea un array `model_viz` que contiene una lista de todas las posibles combinaciones de valores para `cant_lluvia` y `total_ovinos`, que será usado para hacer la predicción y graficar la superficie de regresión.

```
In [30]: x_pred = np.linspace(7.1, 1041.16, 30) # range of cant_lluvia
y_pred = np.linspace(50513, 797790, 30) # range of total_ovinos
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T
```

La siguiente celda prepara los datos necesarios para visualizar el modelo de regresión lineal múltiple en un gráfico en 3D.

- `x = X[:, 0]` extrae todos los valores de la primera columna de `X` y los asigna a la variable `x`, que representará los valores de la variable independiente `cant_lluvia`.

- `y = X[:, 1]` extrae todos los valores de la segunda columna de `X` y los asigna a la variable `y`, que representará los valores de la variable independiente `total_ovinos`.
- `z = Y` asigna todos los valores de la variable dependiente `kilos_lana` a la variable `z`, que representará la variable que se quiere predecir en función de las variables independientes `x` e `y`.

In [31]:

```
x = X[:, 0]
y = X[:, 1]
z = Y
```

El código `predicted = model_lana.predict(model_viz)` utiliza el modelo de regresión lineal previamente entrenado (`model_lana`) para hacer una predicción de los valores de salida (`kilos_lana`) correspondientes a todas las combinaciones posibles de valores de entrada para `cant_lluvia` y `total_ovinos` que se encuentran en la malla de puntos en 2D (`model_viz`).

Esto se logra utilizando el método `predict` del objeto del modelo de regresión lineal, que toma como entrada una matriz `X` de características y devuelve una matriz de valores de salida predichos correspondientes a cada entrada. En este caso, la matriz `X` es la malla de puntos `model_viz` que contiene todas las posibles combinaciones de valores de `cant_lluvia` y `total_ovinos`. Los valores predichos se almacenan en la variable `predicted`.

In [32]:

```
predicted = model_lana.predict(model_viz)
```

El código `r2 = model_lana.score(X, Y)` calcula el coeficiente de determinación R^2 del modelo de regresión lineal, que como vimos antes, es una medida estadística que indica la proporción de la varianza de la variable dependiente (`kilos_lana`) que es explicada por el modelo en función de las variables independientes (`cant_lluvia` y `total_ovinos`).

Este método toma como entrada una matriz de características `X` (en este caso, la matriz `X` que se creó anteriormente a partir de las variables `cant_lluvia` y `total_ovinos`) y una matriz de valores de salida `Y` (en este caso, la matriz `Y` que contiene los valores reales de `kilos_lana`). Luego, compara los valores de salida predichos por el modelo para cada muestra de `X` con los valores reales correspondientes de `Y`, y devuelve un valor entre menos infinito y 1 que indica la calidad del ajuste del modelo.

Un valor de R^2 más cercano a 1 indica que el modelo explica una gran proporción de la varianza de la variable dependiente, lo que indica un mejor ajuste del modelo a los datos. En este caso, el valor de R^2 obtenido se almacena en la variable `r2`.

In [33]:

```
r2 = model_lana.score(X, Y)
```

El siguiente código crea una figura con tres subplots en 3D utilizando la biblioteca Matplotlib. Cada subplot contiene una visualización de la superficie de regresión lineal para diferentes ángulos de vista.

En cada subplot, se hace una visualización de los datos de entrada y de los valores predichos por el modelo. Los datos de entrada se representan como puntos negros en el fondo, mientras que los valores predichos se representan como puntos azules en la superficie.

También se establecen las etiquetas de los ejes y los parámetros del locator para la ubicación de las etiquetas de los ejes.

Finalmente, se establecen las posiciones de vista de los subplots y se agrega un título a la figura que muestra el valor del coeficiente de determinación (R^2) obtenido del modelo.

La llamada a la función `fig.tight_layout()` se encarga de ajustar automáticamente los espacios entre los subplots para que no haya superposiciones.

```
In [34]: plt.style.use('default')
fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

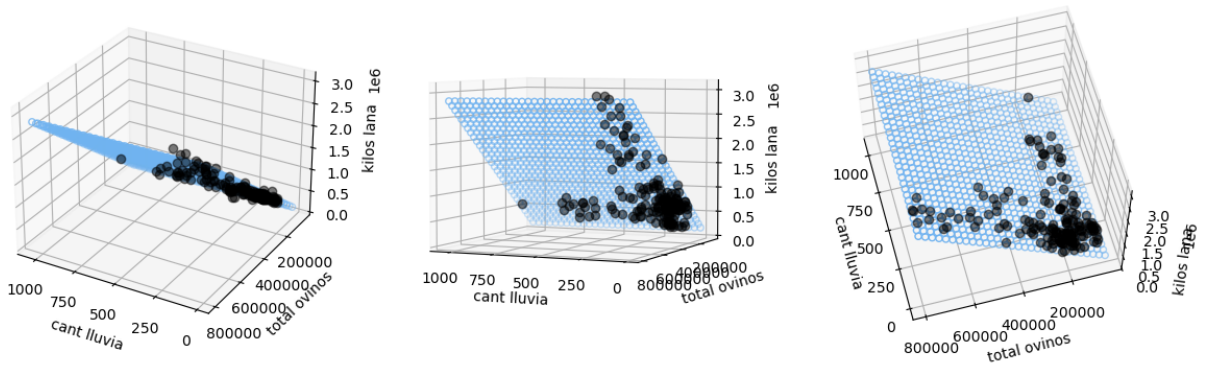
for ax in axes:
    flattenX = x.flatten()
    flattenY = y.flatten()
    flattenZ = z.values.flatten()
    ax.plot(
        flattenX, flattenY, flattenZ, color='k',
        zorder=15, linestyle='none', marker='o', alpha=0.5
    )
    ax.scatter(
        xx_pred.flatten(), yy_pred.flatten(), predicted,
        facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0'
    )
    ax.set_xlabel('cant lluvia', fontsize=10)
    ax.set_ylabel('total ovinos', fontsize=10)
    ax.set_zlabel('kilos lana', fontsize=10)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)

fig.tight_layout()
```

$$R^2 = 0.99$$

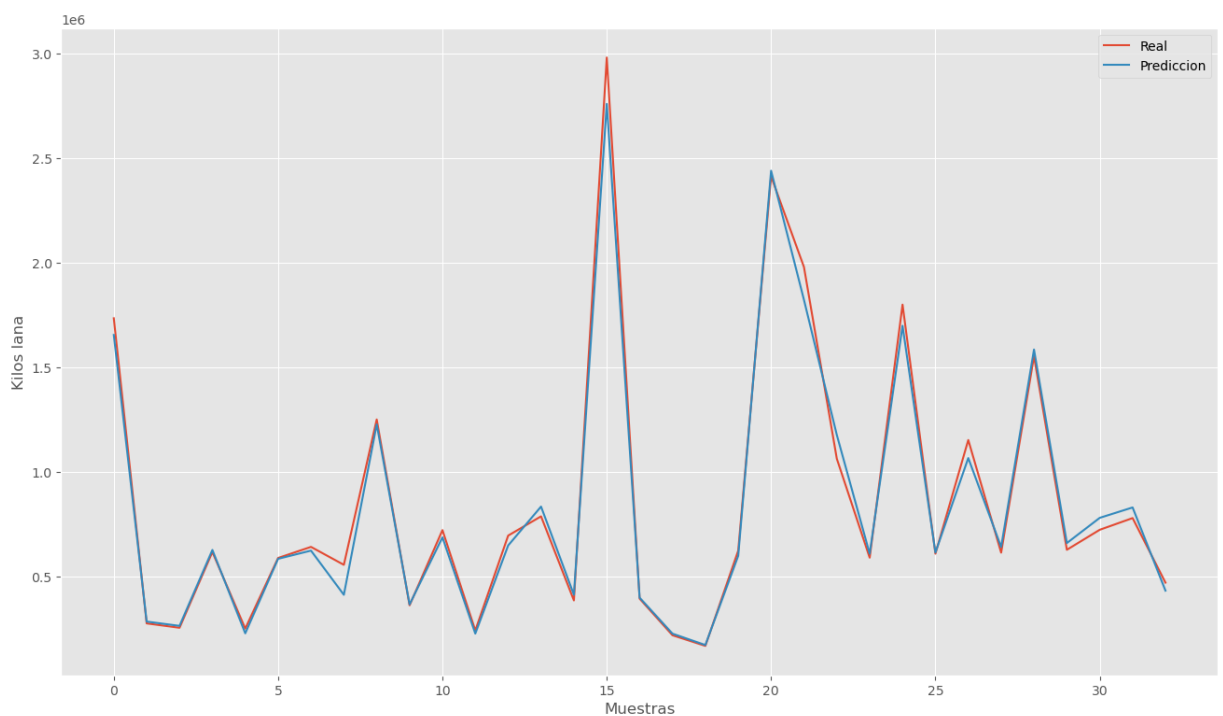


Se observa que los datos se ajustan perfectamente al plano, lo que va acorde con la información extraída de las métricas, que parecen estar indicando overfit.

Gráfico predicción vs real

A continuación, se va a generar un gráfico 2D para comparar los datos reales con la predicción realizada por el modelo.

```
In [29]: plt.plot(range(len(y_test)),y_test,label="Real")
plt.plot(range(len(pred_model_lana)),pred_model_lana,label="Predicción")
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('Muestras')
plt.ylabel('Kilos lana')
plt.legend()
plt.show()
```



Podemos observar que la línea roja que representa los valores reales y la línea azul que representa la predicción del modelo tienen una tendencia igual, casi sin desviaciones. Esto sugiere que el modelo está

sobreajustando los datos de entrenamiento y es más probable que tenga un peor desempeño al predecir valores para nuevos datos.

3.5. Comparar y probar modelos de lana - Dataset 1

Al llegar acá contamos con 4 modelos para comparar, sin embargo se eliminó como candidato al modelo entrenado con los datos escalados, dado que no aportaba nada en comparación con su igual sin escalar. Por lo tanto la comparación se va a realizar sobre los 3 modelos restantes.

Recordemos que:

- El modelo 1 se entrenó con el primer dataset a nivel departamental de Chubut, que es el vamos a estar probando en este Jupyter.
- El modelo 2 se entrenó con el dataset a nivel departamental de Chubut más los datos del Mirabueno, tener en cuenta que contiene al dataset anterior.
- El modelo 3 se entrenó con el dataset actualizado a nivel departamental de Chubut.

Para hacer la comparación se van a tomar las siguientes reglas:

- Si el score es menor a 0, es decir valores negativos, se le va a restar 1.
- Si el score es menor a 0.60, se le va a restar 0.5.
- Si el score es mayor a 0.60, se le va a sumar 1.
- Se le va a sumar un punto a aquel modelo que funcione mejor.
- Se le va a restar 0.25 a aquel modelo que puede conocer los datos debido a que fue entrenado por ese dataset, recordemos que en este punto cada modelo solo conoce el 80% de su dataset de entrenamiento.

Dado que el modelo 2 se entrenó con un dataset parecido al de este Jupyter, se lo va a penalizar con 0.2.

En la siguiente celda de código, se importarán varias librerías, entre ellas:

- Pandas para manejar los datos a través de DataFrames.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados de las predicciones mediante gráficos.
- De Scikit-learn se utilizarán las métricas **r2_score**.
- Por último, se usará `load_pickle` para recuperar el modelo entrenado guardado previamente.

```
In [1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from statsmodels.iolib.smpickle import load_pickle
```

Primero cargaremos los tres modelos para luego probarlos con datos.

```
In [2]: modelo_lana_1 = load_pickle('modelos/modelo_prediccion_lana_1')
modelo_lana_2 = load_pickle('modelos/modelo_prediccion_lana_2')
modelo_lana_3 = load_pickle('modelos/modelo_prediccion_lana_3')
```

Pruebas con dos tuplas

La idea acá es simular que un productor cuenta con dos años de datos respecto a la lluvia y al total de ovinos. Entonces le pasamos ambos datos al modelo, y con ellos hace la predicción, dándonos los kilos de lana, dato que también conocemos y nos sirve para comparar el valor real con el valor predicho.

Prueba 1

Se tomaron dos observaciones del año 2013 y 2014 correspondientes a Biedma.

total_ovinos	cant_lluvia	kilos_lana
187897	226.5	620176.0
181925	331.3	649130.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [3]: resultado_prediccion_1 = modelo_lana_1.predict([[226.5,187897],  
[331.3,181925]])
```

```
In [4]: resultado_prediccion_2 = modelo_lana_2.predict([[226.5,187897],  
[331.3,181925]])
```

```
In [5]: resultado_prediccion_3 = modelo_lana_3.predict([[226.5,187897],  
[331.3,181925]])
```

```
In [6]: dato_real =[[620176.0],[649130.0]]
```

```
In [7]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[7]: 'Score o R2 de la predicción 1: -7.01'
```

```
In [8]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[8]: 'Score o R2 de la predicción 2: -12.08'
```

```
In [9]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

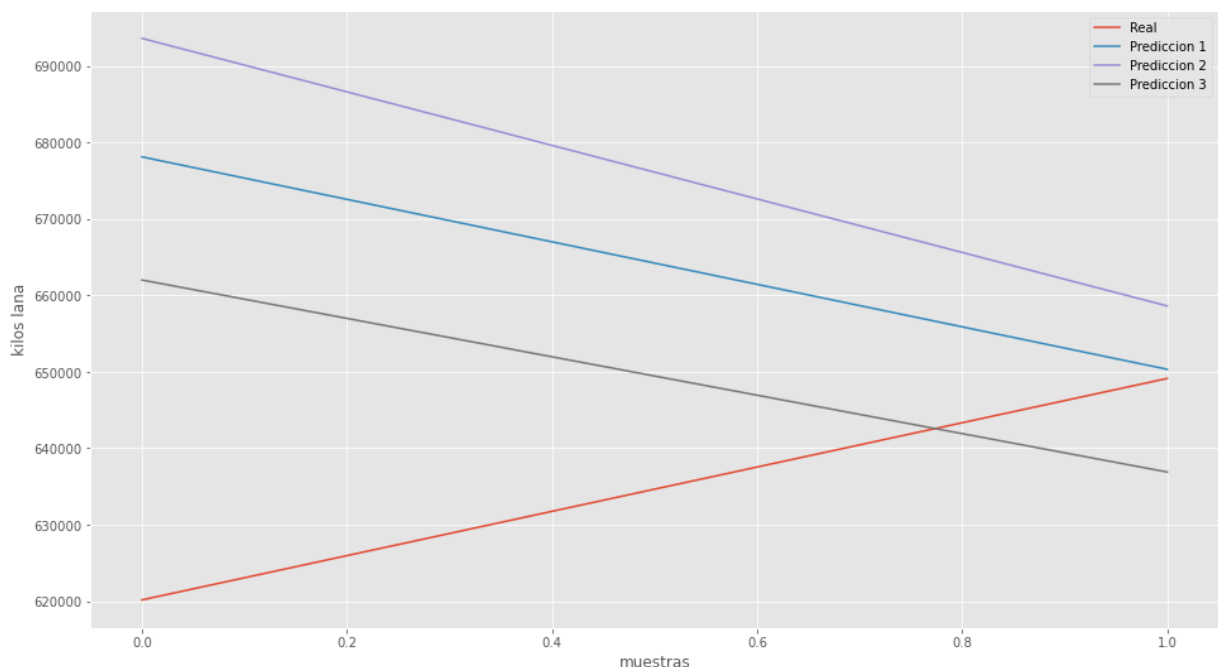
```
Out[9]: 'Score o R2 de la predicción 3: -3.53'
```

Se observa que ningún modelo logrará predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

In [10]:

```
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)),dato_real,label="Real")
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,label="1")
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,label="2")
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,label="3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelos no logran predecir estos datos dado que las tres líneas de predicción no se ajustan a la real. Se observa que en la predicción uno, el primer valor lo predice muy mal pero el segundo casi perfecto, la predicción dos tiene un comportamiento similar pero para el segundo esta un poco más alejado y la predicción tres no da un valor tan alto como primer valor y el segundo que da es un poco bajo.

Entonces el "mejor modelo" resulta ser el 3, seguido por el 1 y el que peor predice es el 2.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-1.25	-1.25
2	0	-1.20	-1.20
3	1	-1	0.0

Prueba 2

Se tomaron dos observaciones correspondientes a Biedma del 2014 y a Florentino Ameghino del 2013.

total_ovinos	cant_lluvia	kilos_lana
181925	331.3	649130.0
284639	135.5	1143408.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [11]: resultado_prediccion_1 = modelo_lana_1.predict([[331.3,181925],
[135.5,284639]])
```

```
In [12]: resultado_prediccion_2 = modelo_lana_2.predict([[331.3,181925],
[135.5,284639]])
```

```
In [13]: resultado_prediccion_3 = modelo_lana_3.predict([[331.3,181925],
[135.5,284639]])
```

```
In [14]: dato_real = [[649130.0],[1143408.0]]
```

```
In [15]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_1))
```

```
Out[15]: 'Score o R2 de la predicción 1: 0.93'
```

```
In [16]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_2))
```

```
Out[16]: 'Score o R2 de la predicción 2: 0.94'
```

```
In [17]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_3))
```

```
Out[17]: 'Score o R2 de la predicción 3: 0.86'
```

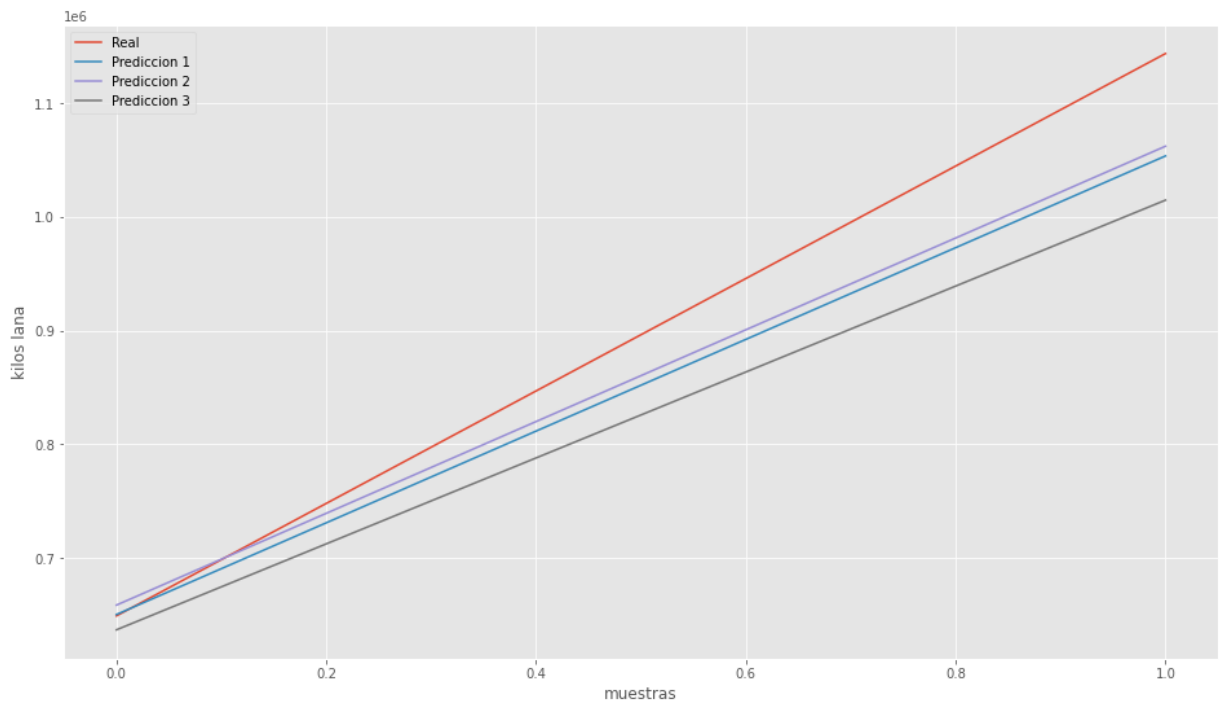
Se observa que todos los modelos logran predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [18]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)),dato_real,label="Real")
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,
1")
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,
2")
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,
```



```
3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelos logran predecir estos datos.

Entonces el mejor modelo resulta ser el 2, seguido por el 1 y el que peor predice es el 3.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	1	-0.25	-0.5
2	2	-0.20	0.6
3	1	0	1.0

Prueba 3

Se tomaron dos obsevaciones correspondientes a Biedma del 2012 y a Florentino Ameghino de 2009.

total_ovinos	cant_lluvia	kilos_lana
193849	46.40	649130.0
343991	137.7	1474032.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [19]: resultado_prediccion_1 = modelo_lana_1.predict([[46.40,193849],
[137.7,343991]])
```

```
In [20]: resultado_prediccion_2 = modelo_lana_2.predict([[46.40,193849],
[137.7,343991]])
```

```
In [21]: resultado_prediccion_3 = modelo_lana_3.predict([[46.40,193849],  
[137.7,343991]])
```

```
In [22]: dato_real =[[649130.0],[1474032.0]]
```

```
In [23]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[23]: 'Score o R2 de la predicción 1: 0.88'
```

```
In [24]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[24]: 'Score o R2 de la predicción 2: 0.87'
```

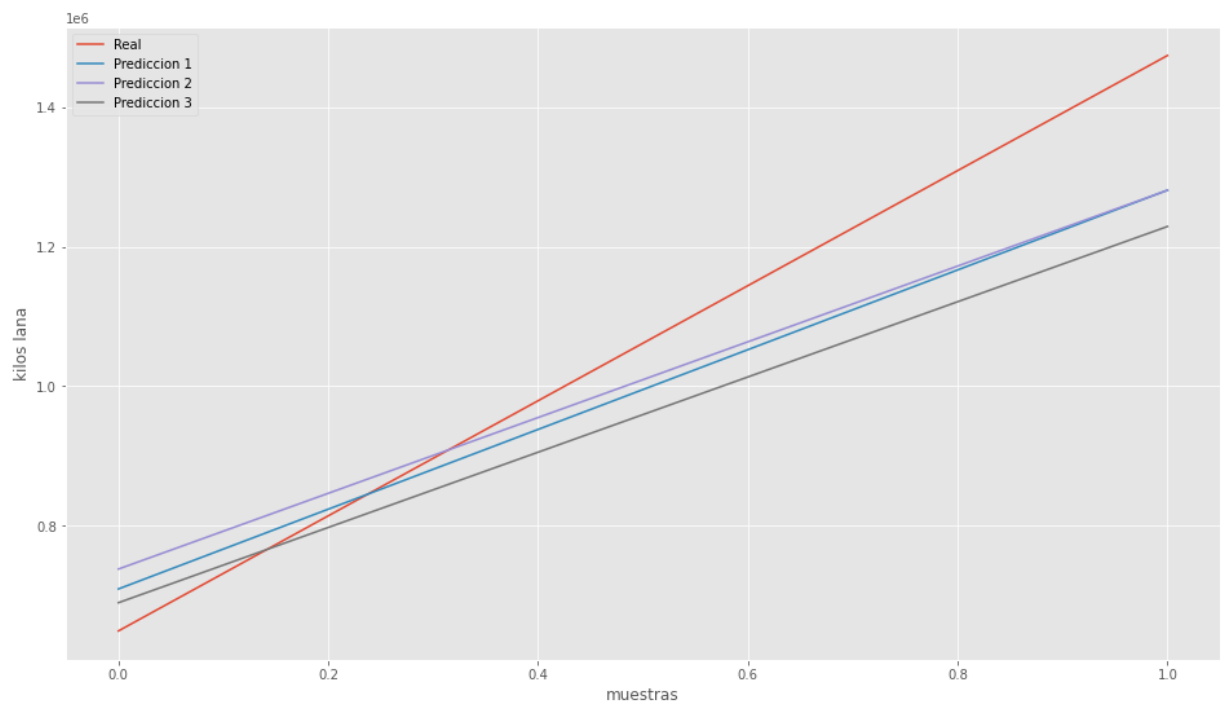
```
In [25]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

```
Out[25]: 'Score o R2 de la predicción 3: 0.82'
```

Se observa que todos los modelos logran predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [26]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.plot(range(len(dato_real)),dato_real,label="Real")  
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,  
1")  
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,  
2")  
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,  
3")  
plt.xlabel('muestras')  
plt.ylabel('kilos lana')  
plt.legend()  
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelo logran predecir estos datos.

El mejor modelo resulta ser el 1, pero como fue entrenado con estos datos se lo va a penalizar, seguido por el 2 y por último el 3 .

Modelo	Puntaje positivo	Puntaje negativo	Total
1	2	-0.25	1.25
2	1	-0.20	1.40
3	1	0	2.00

Prueba 4

Se tomaron dos obsevaciones correspondientes a Rawson del 2014 y a Río Senguer de 2013.

total_ovinos	cant_lluvia	kilos_lana
120837	187.2	211280.0
834289	105.77	2273536.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [27]: resultado_prediccion_1 = modelo_lana_1.predict([[187.2,120837],
[105.77,834289]])
```

```
In [28]: resultado_prediccion_2 = modelo_lana_2.predict([[187.2,120837],
[105.77,834289]])
```

```
In [29]: resultado_prediccion_3 = modelo_lana_3.predict([[187.2,120837],
[105.77,834289]])
```

```
In [30]: dato_real = [[211280.0],[2273536.0]]
```

```
In [31]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_1))
```

```
Out[31]: 'Score o R2 de la predicción 1: 0.61'
```

```
In [32]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_2))
```

```
Out[32]: 'Score o R2 de la predicción 2: 0.66'
```

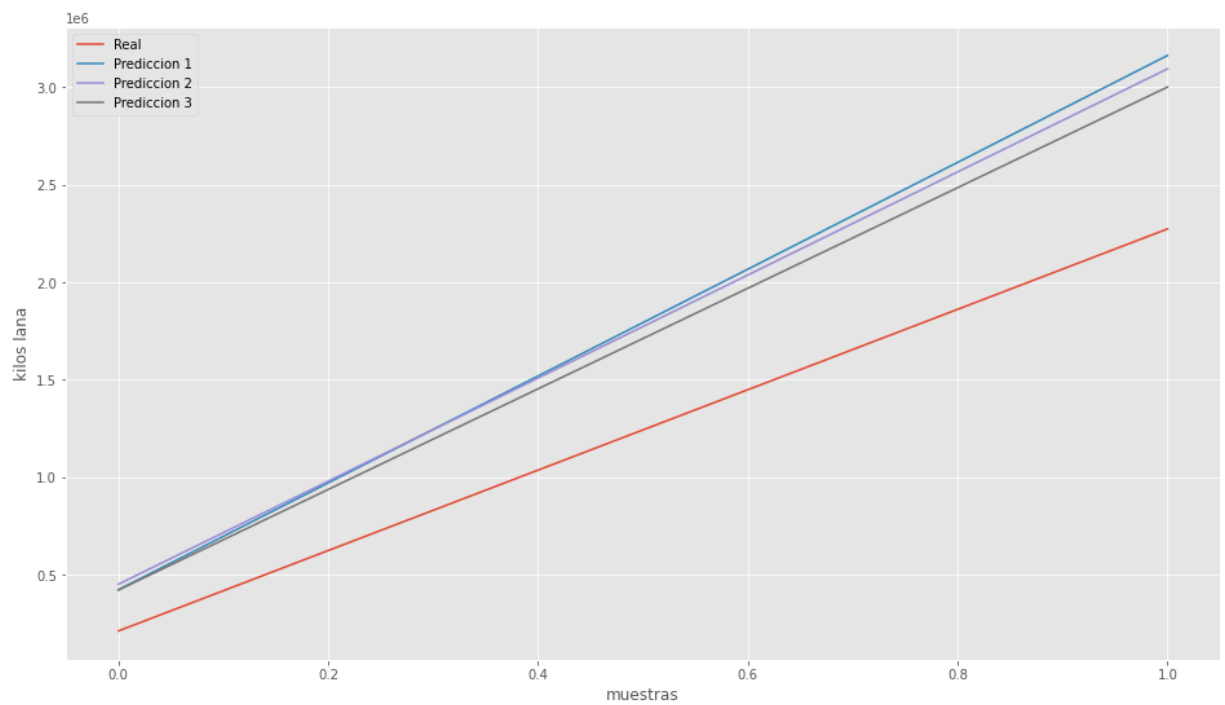
```
In [33]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_3))
```

```
Out[33]: 'Score o R2 de la predicción 3: 0.73'
```

Se observa que todos los modelos logran predecir correctamente estos datos según el R2, pero a todos les costo un poco.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [34]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)), dato_real, label="Real")
plt.plot(range(len(resultado_prediccion_1)), resultado_prediccion_1, label="Predicción 1")
plt.plot(range(len(resultado_prediccion_2)), resultado_prediccion_2, label="Predicción 2")
plt.plot(range(len(resultado_prediccion_3)), resultado_prediccion_3, label="Predicción 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelo logran predecir estos datos pero están un poco alejados.

El mejor modelo resulta ser el 3, seguido por el 2 y el 1.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	1	-0.25	2.0
2	1	-0.20	2.2
3	2	0	4.0

Prueba 5

Se tomaron dos obsevaciones correspondientes a Gaiman del 2014 y a Futalefú de 2014.

total_ovinos	cant_lluvia	kilos_lana
245238	137.93	804810.0
164541	614.79	455920.0

```
In [35]: resultado_prediccion_1 = modelo_lana_1.predict([[137.93,245238],
[614.79,164541]])
```

```
In [36]: resultado_prediccion_2 = modelo_lana_2.predict([[137.93,245238],
[614.79,164541]])
```

```
In [37]: resultado_prediccion_3 = modelo_lana_3.predict([[137.93,245238],
[614.79,164541]])
```

```
In [38]: dato_real =[[804810.0],[455920.0]]
```

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [39]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_1))
```

```
Out[39]: 'Score o R2 de la predicción 1: 0.63'
```

```
In [40]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_2))
```

```
Out[40]: 'Score o R2 de la predicción 2: 0.62'
```

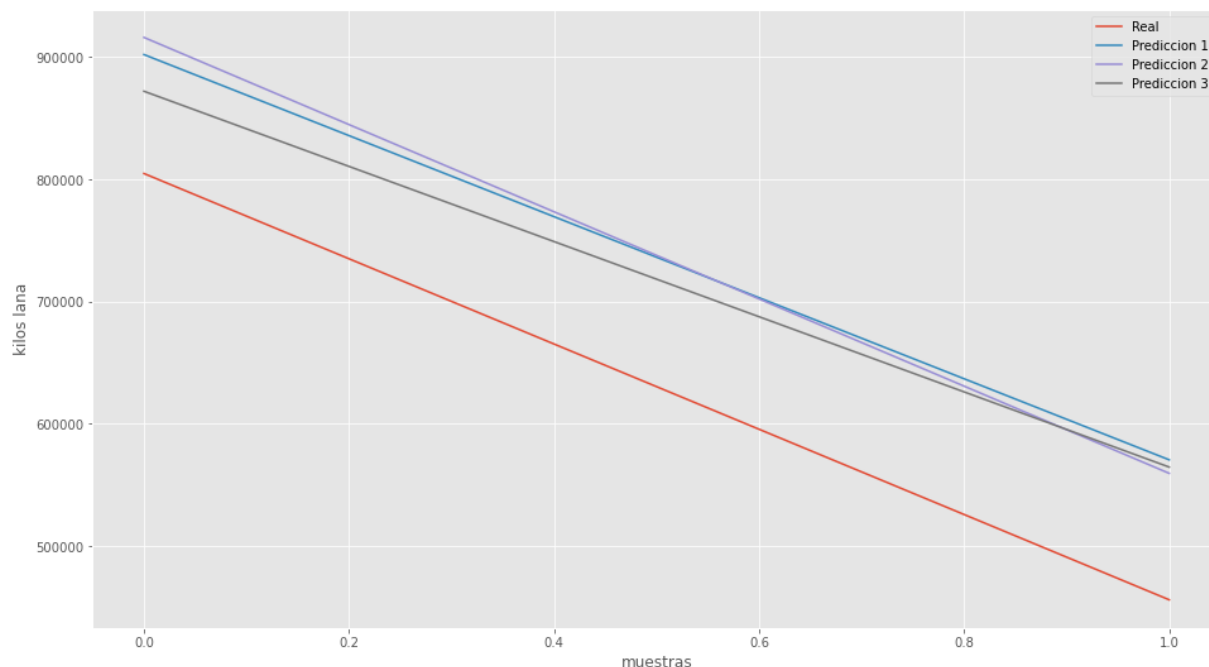
```
In [41]: 'Score o R2 de la predicción: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_3))
```

```
Out[41]: 'Score o R2 de la predicción: 0.73'
```

Se observa que todos los modelos logran predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [42]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)), dato_real, label="Real")
plt.plot(range(len(resultado_prediccion_1)), resultado_prediccion_1, label="1")
plt.plot(range(len(resultado_prediccion_2)), resultado_prediccion_2, label="2")
plt.plot(range(len(resultado_prediccion_3)), resultado_prediccion_3, label="3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelo logran predecir estos datos, pero estan un poco lejos. El mejor modelo resulta ser el 3, seguido por el 1 y por último el 2.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	1	-0.25	2.75
2	1	-0.20	3.0
3	2	0	6.0

Pruebas con una tupla

La idea acá es simular que un productor cuenta con los datos de la lluvia y el total de ovinos, del año actual respectivo. Entonces le pasamos ambos datos al modelo, y con ellos hace la predicción para ese año, dándonos los kilos de lana, dato que también conocemos y nos sirve para comparar el valor real con el valor predicho.

Prueba 1

Se tomo una observación del dataset correspondiente a Cushamen del 2014.

total_ovinos	cant_lluvia	kilos_lana
230802	640.4	729750.0

In [43]: `dato_real = 729750.0`

In [44]: `resultado_prediccion_1 = modelo_lana_1.predict([[640.4,230802]])`
`round(resultado_prediccion_1[0][0] - dato_real,2)`

Out[44]: 93698.36

In [45]: `resultado_prediccion_2 = modelo_lana_2.predict([[640.4,230802]])`
`round(resultado_prediccion_2[0][0] - dato_real,2)`

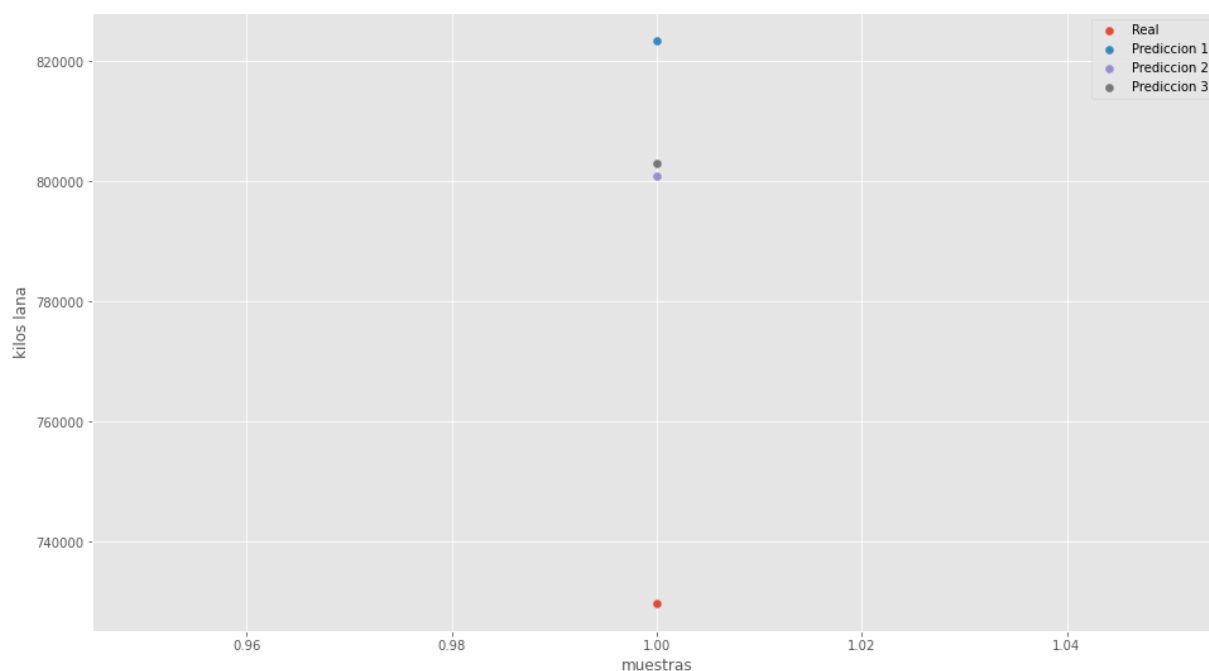
Out[45]: 71097.97

```
In [46]: resultado_prediccion_3 = modelo_lana_3.predict([[640.4,230802]])
round(resultado_prediccion_3[0][0] - dato_real,2)
```

Out[46]: 73307.61

En estos tres resultados, vemos que los modelos predijeron kilogramos de lana demás, es decir el modelo 1 se paso por 93698.36 kgs, el modelo 2 se paso por 71097.97 kgs y el modelo 3 se paso por 73307.61 kgs.

```
In [47]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.scatter(y=dato_real, x=1, label="Real")
plt.scatter(y=resultado_prediccion_1,x=1, label="Prediccion 1")
plt.scatter(y=resultado_prediccion_2,x=1, label="Prediccion 2")
plt.scatter(y=resultado_prediccion_3,x=1, label="Prediccion 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



El mejor modelo resulta ser el 2, seguido por el 3 y por último el 1. Cómo acá no se puede calcular el R2, por estar trabajando con un solo dato, solo se va a tener en cuenta cuál es el modelo que esta mas cerca del dato real.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-0.25	2.5
2	0	-0.20	2.8
3	1	0	7.0

Prueba 2

Se tomo una observación del dataset correspondiente a Sarmiento del 2015.

total_ovinos	cant_lluvia	kilos_lana
215378	130.55	908640.0

```
In [48]: dato_real = 908640.0
```

```
In [49]: resultado_prediccion_1 = modelo_lana_1.predict([[130.55,215378]])  
round(resultado_prediccion_1[0][0] - dato_real,2)
```

```
Out[49]: -120668.33
```

```
In [50]: resultado_prediccion_2 = modelo_lana_2.predict([[130.55,215378]])  
round(resultado_prediccion_2[0][0] - dato_real,2)
```

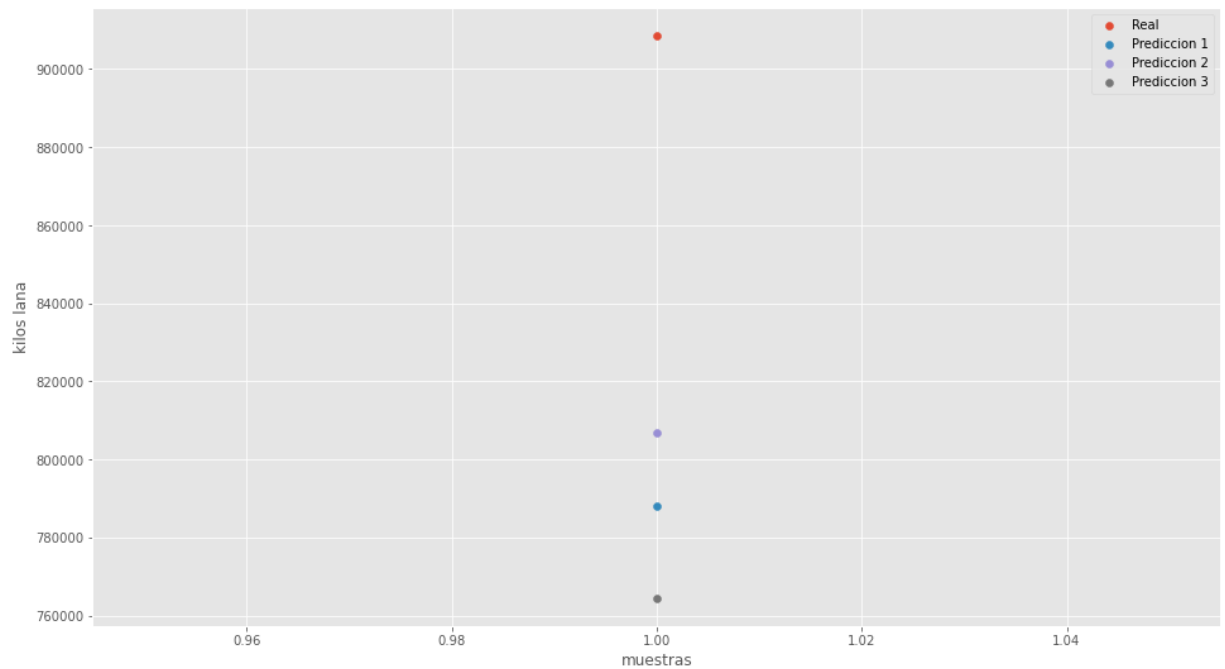
```
Out[50]: -101743.24
```

```
In [51]: resultado_prediccion_3 = modelo_lana_3.predict([[130.55,215378]])  
round(resultado_prediccion_3[0][0] - dato_real,2)
```

```
Out[51]: -144122.57
```

En estos tres resultados, vemos que los modelos predijeron kilogramos de lana de menos, es decir al modelo 1 le faltó 120668.33 kgs, al modelo 2 le faltó 101743.24 kgs y al modelo 3 le faltó 144122.57 kgs.

```
In [52]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.scatter(y=dato_real, x=1, label="Real")  
plt.scatter(y=resultado_prediccion_1, x=1, label="Prediccion 1")  
plt.scatter(y=resultado_prediccion_2, x=1, label="Prediccion 2")  
plt.scatter(y=resultado_prediccion_3, x=1, label="Prediccion 3")  
plt.xlabel('muestras')  
plt.ylabel('kilos lana')  
plt.legend()  
plt.show()
```



El mejor modelo resulta ser el 2, seguido por el 1 y por último el 3. Cómo acá no se puede calcular el R2, por estar trabajando con un solo dato, solo se va a tener en cuenta cuál es el modelo que esta mas cerca del dato real.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-0.25	2.25
2	1	-0.20	3.60
3	0	0	7.00

Resultados

El mejor **modelo es el 3** con un puntaje final de 7.0, seguido del modelo 2 con 3.6 y por ultimo el modelo 1 con 2.5 puntos. Acá hay que tener en cuenta que el modelo 1 fue entrenado con el dataset que estamos trabajando y fue el que peor puntaje obtuvo. Sin embargo el modelo 3 logro predecir estos datos que no conocía.

3.6. Comparar y probar modelos de lana - Dataset 2

Al llegar acá contamos con 4 modelos para comparar, sin embargo se eliminó como candidato al modelo entrenado con los datos escalados, dado que no aportaba nada en comparación con su igual sin escalar. Por lo tanto la comparación se va a realizar sobre los 3 modelos restantes.

Recordemos que:

- El modelo 1 se entrenó con el primer dataset a nivel departamental de Chubut, tener en cuenta que está incluido en el dataset siguiente.
- El modelo 2 se entrenó con el dataset a nivel departamental de Chubut más los datos del Mirabueno, que es el vamos a estar probando en este Jupyter .
- El modelo 3 se entrenó con el dataset actualizado a nivel departamental de Chubut.

Para hacer la comparación se van a tomar las siguientes reglas:

- Si el score es menor a 0, es decir valores negativos, se le va a restar 1.
- Si el score es menor a 0.60, se le va a restar 0.5.
- Si el score es mayor a 0.60, se le va a sumar 1.
- Se le va a sumar un punto a aquel modelo que funcione mejor.
- Se le va a restar 0.25 a aquel modelo que puede conocer los datos debido a que fue entrenado por ese dataset, recordemos que en este punto cada modelo solo conoce el 80% de su dataset de entrenamiento.

Dado que el modelo 1 se entrenó con un dataset parecido al de este Jupyter, se lo va a penalizar con 0.2.

En la siguiente celda de código, se importarán varias librerías, entre ellas:

- Pandas para manejar los datos a través de DataFrames.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados de las predicciones mediante gráficos.
- De Scikit-learn se utilizarán las métricas **r2_score**.
- Por último, se usará `load_pickle` para recuperar el modelo entrenado guardado previamente.

```
In [2]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from statsmodels.iolib.smpickle import load_pickle
```

Primero cargaremos los tres modelos para luego probarlos con datos.

```
In [3]: modelo_lana_1 = load_pickle('modelos/modelo_prediccion_lana_1')
modelo_lana_2 = load_pickle('modelos/modelo_prediccion_lana_2')
modelo_lana_3 = load_pickle('modelos/modelo_prediccion_lana_3')
```

Pruebas con dos tuplas

La idea acá es simular que un productor cuenta con dos años de datos respecto a la lluvia y al total de ovinos. Entonces le pasamos ambos datos al modelo, y con ellos hace la predicción, dándonos los kilos de lana, dato que también conocemos y nos sirve para comparar el valor real con el valor predicho.

Prueba 1

Se tomaron dos observaciones del año 2013 y 2014 correspondientes a Mirabueno.

total_ovinos	cant_lluvia	kilos_lana
2096	197.6	9892.0
2908	187.2	649130.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [4]: resultado_prediccion_1 = modelo_lana_1.predict([[197.6,2096],  
[187.2,2908]])
```

```
In [5]: resultado_prediccion_2 = modelo_lana_2.predict([[197.6,2096],  
[187.2,2908]])
```

```
In [6]: resultado_prediccion_3 = modelo_lana_3.predict([[197.6,2096],  
[187.2,2908]])
```

```
In [7]: dato_real = [[9892.0],[10510.0]]
```

```
In [8]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[8]: 'Score o R2 de la predicción 1: -18190.53'
```

```
In [9]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[9]: 'Score o R2 de la predicción 2: -144.22'
```

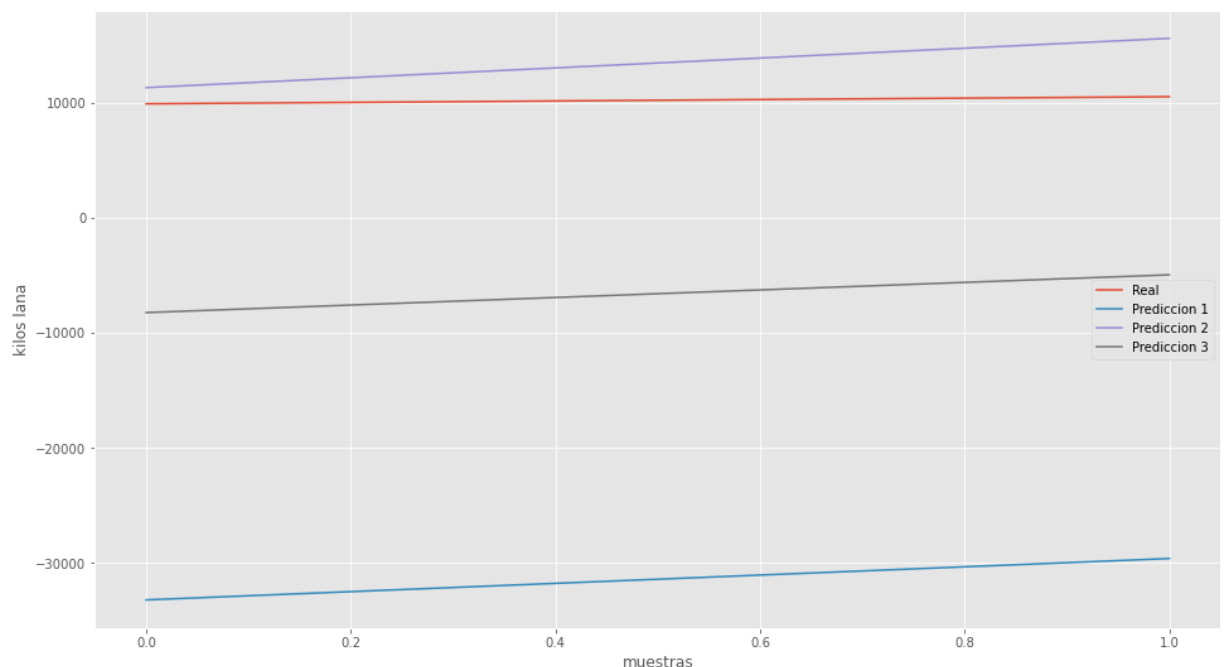
```
In [10]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

```
Out[10]: 'Score o R2 de la predicción 3: -2981.53'
```

Se observa que ningún modelo logrará predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [11]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)),dato_real,label="Real")
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,label="1")
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,label="2")
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,label="3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelos no logran predecir estos datos dado que las tres líneas de predicción no se ajustan a la real.

Entonces el "mejor modelo" resulta ser el 2, seguido por el 3 y el que peor predice es el 1.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-1.20	-1.20
2	0	-1.25	-1.25
3	0	-1	-1.00

Prueba 2

Se tomaron dos obsevaciones correspondientes a Biedma del 2014 y a Florentino Ameghino del 2013.

total_ovinos	cant_lluvia	kilos_lana
181925	331.3	649130.0

total_ovinos	cant_lluvia	kilos_lana
284639	135.5	1143408.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [12]: resultado_prediccion_1 = modelo_lana_1.predict([[331.3,181925],
[135.5,284639]])
```

```
In [13]: resultado_prediccion_2 = modelo_lana_2.predict([[331.3,181925],
[135.5,284639]])
```

```
In [14]: resultado_prediccion_3 = modelo_lana_3.predict([[331.3,181925],
[135.5,284639]])
```

```
In [15]: dato_real =[[649130.0],[1143408.0]]
```

```
In [16]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_1))
```

```
Out[16]: 'Score o R2 de la predicción 1: 0.93'
```

```
In [17]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_2))
```

```
Out[17]: 'Score o R2 de la predicción 2: 0.94'
```

```
In [18]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_3))
```

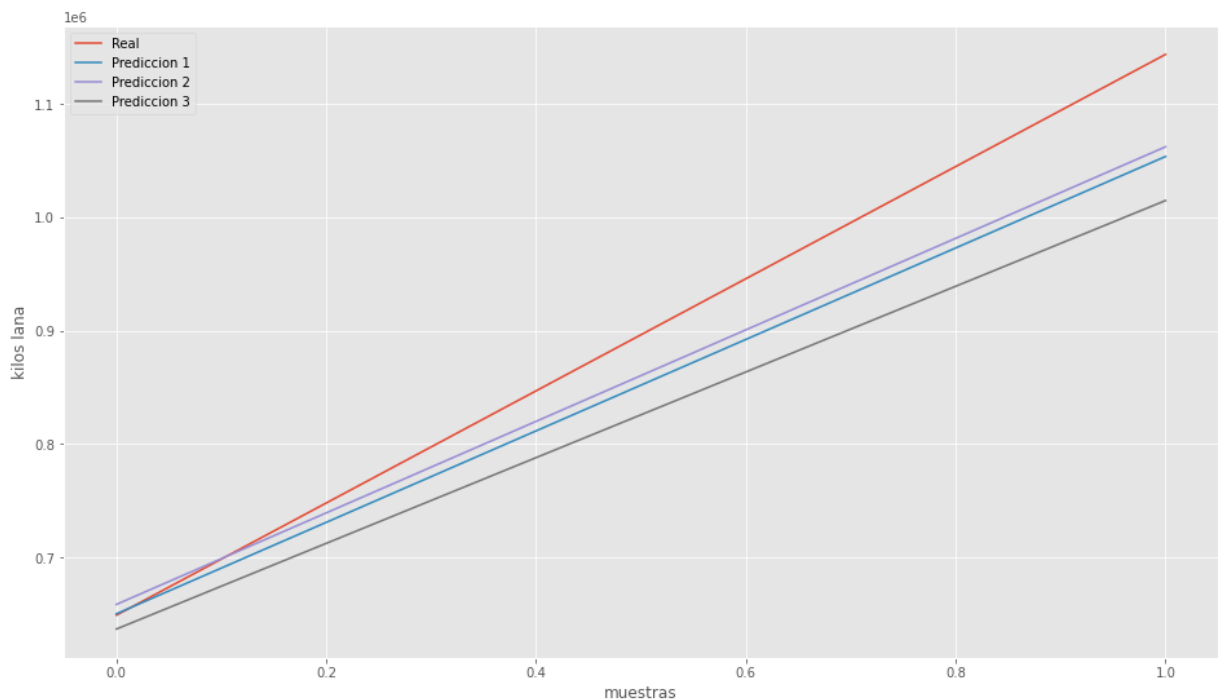
```
Out[18]: 'Score o R2 de la predicción 3: 0.86'
```

Se observa que todos los modelos logran predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [19]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)),dato_real,label="Real")
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,
label="1")
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,
label="2")
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,
label="3")
plt.xlabel('muestras')
```

```
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelos logran predecir estos datos.

Entonces el mejor modelo resulta ser el 2, seguido por el 1 y el que peor predice es el 3.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	1	-0.20	-0.4
2	2	-0.25	0.5
3	1	0	0.0

Prueba 3

Se tomaron dos obsevaciones correspondientes a Mirabueno del 2012 y 2009.

total_ovinos	cant_lluvia	kilos_lana
2051	168.5	8739.0
1828	171.8	8376.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [20]: resultado_prediccion_1 = modelo_lana_1.predict([[168.5,2051],
[171.8,1828]])
```

```
In [21]: resultado_prediccion_2 = modelo_lana_2.predict([[168.5,2051],
[171.8,1828]])
```

```
In [22]: resultado_prediccion_3 = modelo_lana_3.predict([[168.5,2051],  
[171.8,1828]])
```

```
In [23]: dato_real =[[8739.0],[8376.0]]
```

```
In [24]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[24]: 'Score o R2 de la predicción 1: -51356.18'
```

```
In [25]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[25]: 'Score o R2 de la predicción 2: -942.29'
```

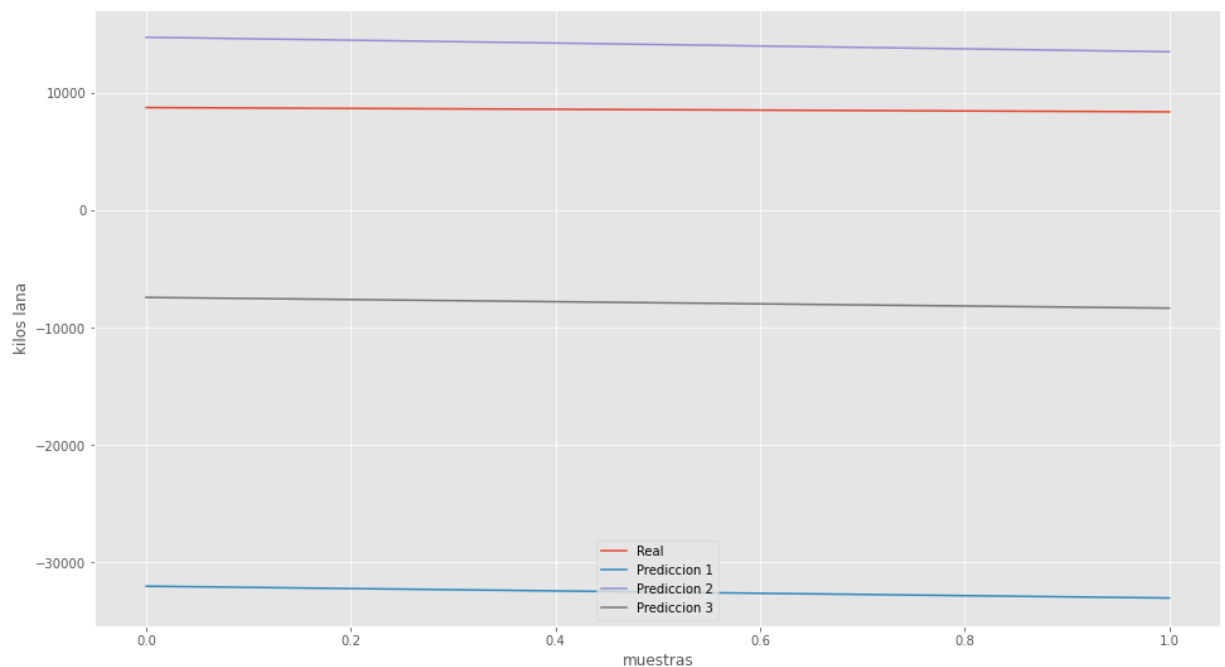
```
In [26]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

```
Out[26]: 'Score o R2 de la predicción 3: -8227.13'
```

Se observa que ningún modelo logrará predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [27]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.plot(range(len(dato_real)),dato_real,label="Real")  
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,label="1")  
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,label="2")  
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,label="3")  
plt.xlabel('muestras')  
plt.ylabel('kilos lana')  
plt.legend()  
plt.show()
```

Los gráficos nos muestran lo que nos indicaban las métricas, los modelos no logran predecir estos datos dado que las tres líneas de predicción no se ajustan a la real.

Entonces el "mejor modelo" resulta ser el 2, seguido por el 3 y el que peor predice es el 1.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-1.20	-1.60
2	1	-1.25	0.25
3	0	-1.00	-1.00

Pruebas con una tupla

La idea acá es simular que un productor cuenta con los datos de la lluvia y el total de ovinos, del año actual respectivo. Entonces le pasamos ambos datos al modelo, y con ellos hace la predicción para ese año, dándonos los kilos de lana, dato que también conocemos y nos sirve para comparar el valor real con el valor predicho.

Prueba 1

Se tomo una observación del dataset correspondiente a Mirabueno del 2016.

total_ovinos	cant_lluvia	kilos_lana
4409	218.0	10990.0

```
In [28]: dato_real = 10990.0
```

```
In [29]: resultado_prediccion_1 = modelo_lana_1.predict([[218.0,4409]])
round(resultado_prediccion_1[0][0] - dato_real,2)
```

```
Out[29]: -36313.51
```

```
In [30]: resultado_prediccion_2 = modelo_lana_2.predict([[218.0,4409]])  
round(resultado_prediccion_2[0][0] - dato_real,2)
```

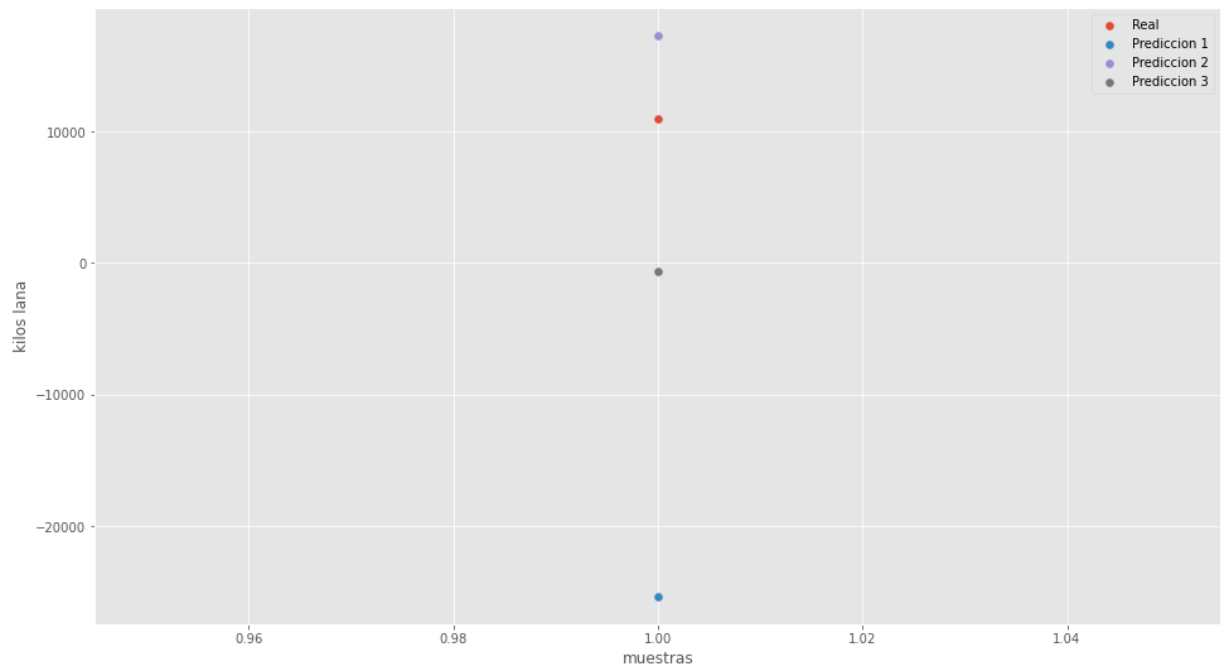
Out[30]: 6329.71

```
In [31]: resultado_prediccion_3 = modelo_lana_3.predict([[218.0,4409]])  
round(resultado_prediccion_3[0][0] - dato_real,2)
```

Out[31]: -11584.87

En estos tres resultados, vemos que los modelos predijeron kilogramos de lana demás y de menos, es decir el modelo 1 le faltó 36313.51 kgs, el modelo 2 se pasó por 6329.71 kgs y el modelo 3 le faltó 11584.87 kgs.

```
In [32]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.scatter(y=dato_real, x=1, label="Real")  
plt.scatter(y=resultado_prediccion_1,x=1, label="Prediccion 1")  
plt.scatter(y=resultado_prediccion_2,x=1, label="Prediccion 2")  
plt.scatter(y=resultado_prediccion_3,x=1, label="Prediccion 3")  
plt.xlabel('muestras')  
plt.ylabel('kilos lana')  
plt.legend()  
plt.show()
```



El mejor modelo resulta ser el 2, seguido por el 3 y por último el 1. Cómo acá no se puede calcular el R2, por estar trabajando con un solo dato, solo se va a tener en cuenta cuál es el modelo que está más cerca del dato real.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-0.20	-1.8
2	1	-0.25	1.0
3	0	0	-1.0

Resultados

No hay un modelo que funcione, la idea era ver si estos modelos podían predecir la producción de lana de un campo de la región sin tener muchos datos respecto a ellos, y la respuesta fue que no.

3.7. Comparar y probar modelos de lana - Dataset 3

Al llegar acá contamos con 4 modelos para comparar, sin embargo se eliminó como candidato al modelo entrenado con los datos escalados, dado que no aportaba nada en comparación con su igual sin escalar. Por lo tanto la comparación se va a realizar sobre los 3 modelos restantes.

Recordemos que:

- El modelo 1 se entrenó con el primer dataset a nivel departamental de Chubut.
- El modelo 2 se entrenó con el dataset a nivel departamental de Chubut más los datos del Mirabueno.
- El modelo 3 se entrenó con el dataset actualizado a nivel departamental de Chubut, que es el vamos a estar probando en este Jupyter.

Para hacer la comparación se van a tomar las siguientes reglas:

- Si el score es menor a 0, es decir valores negativos, se le va a restar 1.
- Si el score es menor a 0.60, se le va a restar 0.5.
- Si el score es mayor a 0.60, se le va a sumar 1.
- Se le va a sumar un punto a aquel modelo que funcione mejor.
- Se le va a restar 0.25 a aquel modelo que puede conocer los datos debido a que fue entrenado por ese dataset.

En la siguiente celda de código, se importarán varias librerías, entre ellas:

- Pandas para manejar los datos a través de DataFrames.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados de las predicciones mediante gráficos.
- De Scikit-learn se utilizarán las métricas **r2_score**.
- Por último, se usará `load_pickle` para recuperar el modelo entrenado guardado previamente.

```
In [1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from statsmodels.iolib.smpickle import load_pickle
```

Primero cargaremos los tres modelos para luego probarlos con datos.

```
In [2]: modelo_lana_1 = load_pickle('modelos/modelo_prediccion_lana_1')
modelo_lana_2 = load_pickle('modelos/modelo_prediccion_lana_2')
modelo_lana_3 = load_pickle('modelos/modelo_prediccion_lana_3')
```

Pruebas con dos tuplas

La idea acá es simular que un productor cuenta con dos años de datos respecto a la lluvia y al total de ovinos. Entonces le pasamos ambos datos al modelo, y con ellos hace la predicción, dándonos los kilos de

lana, dato que también conocemos y nos sirve para comparar el valor real con el valor predicho.

Prueba 1

Se tomaron dos observaciones del año 2013 y 2014 correspondientes a Biedma.

total_ovinos	cant_lluvia	kilos_lana
172179	226.5	544941
158523	331.3	513951

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [3]: resultado_prediccion_1 = modelo_lana_1.predict([[226.5,172179],  
[331.3,158523]])
```

```
In [4]: resultado_prediccion_2 = modelo_lana_2.predict([[226.5,172179],  
[331.3,158523]])
```

```
In [5]: resultado_prediccion_3 = modelo_lana_3.predict([[226.5,172179],  
[331.3,158523]])
```

```
In [6]: dato_real = [[544941],[513951]]
```

```
In [7]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[7]: 'Score o R2 de la predicción 1: -14.58'
```

```
In [8]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[8]: 'Score o R2 de la predicción 2: -23.18'
```

```
In [9]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

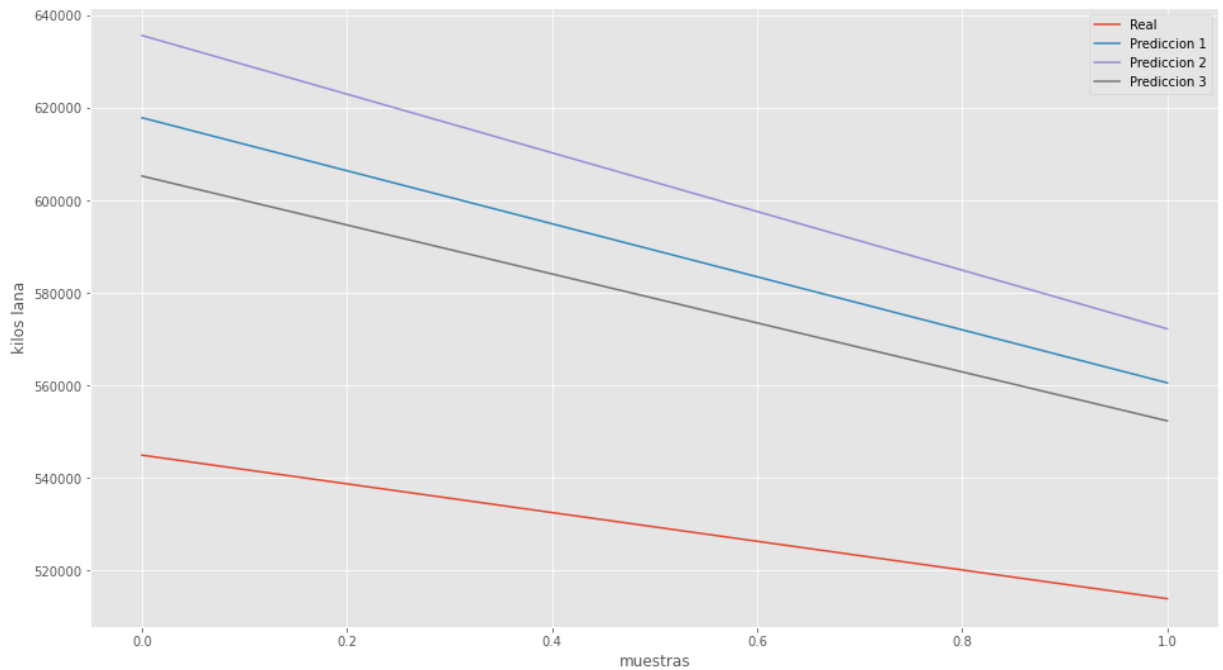
```
Out[9]: 'Score o R2 de la predicción 3: -9.64'
```

Se observa que ningún modelo logró predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [10]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.plot(range(len(dato_real)),dato_real,label="Real")  
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,
```

```
1")
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,1
2")
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,1
3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelos no logran predecir estos datos dado que las tres líneas de predicción están por encima de la real. Se observa que la predicción uno se pasa por unos 80 mil kilos, la predicción dos se pasa por unos 100 mil kilos y la predicción tres se pasa por unos 60 mil kilos.

Entonces el "mejor modelo" resulta ser el 3, pero como fue entrenado con este dataset se lo va a penalizar, seguido por el 1 y el que peor predice es el 2.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-1	-1
2	0	-1	-1
3	1	-1.25	-0.25

Prueba 2

Se tomaron dos obsevaciones correspondientes a Biedma del 2014 y a Florentino Ameghino del 2013.

total_ovinos	cant_lluvia	kilos_lana
158523	331.3	513951
382226	135.5	1088145

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [11]: resultado_prediccion_1 = modelo_lana_1.predict([[331.3,158523],  
[135.5,382226]])
```

```
In [12]: resultado_prediccion_2 = modelo_lana_2.predict([[331.3,158523],  
[135.5,382226]])
```

```
In [13]: resultado_prediccion_3 = modelo_lana_3.predict([[331.3,158523],  
[135.5,382226]])
```

```
In [14]: dato_real = [[513951.0],[1088145.0]]
```

```
In [15]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[15]: 'Score o R2 de la predicción 1: 0.29'
```

```
In [16]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[16]: 'Score o R2 de la predicción 2: 0.30'
```

```
In [17]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

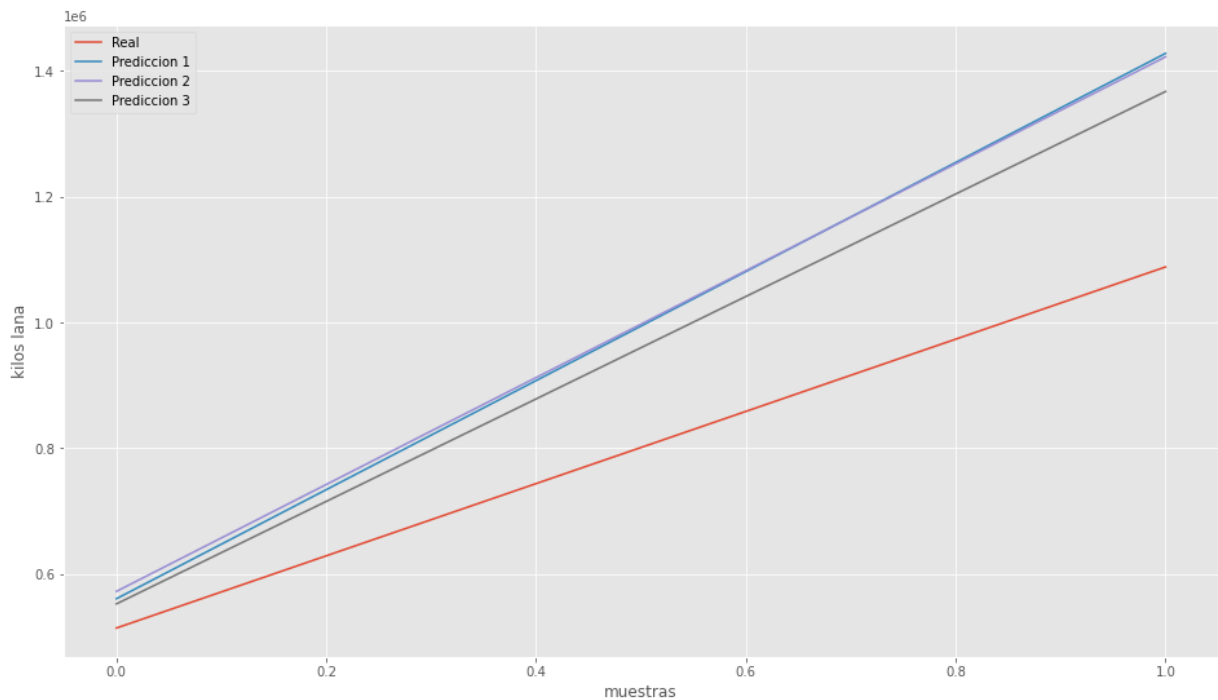
```
Out[17]: 'Score o R2 de la predicción 3: 0.52'
```

Se observa que ningún modelo logró predecir correctamente estos datos según el R2, pero al menos no son valores negativos, el que estuvo mas cerca fue el modelo 3.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [18]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.plot(range(len(dato_real)),dato_real,label="Real")  
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,label="1")  
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,label="2")  
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,label="3")  
plt.xlabel('muestras')  
plt.ylabel('kilos lana')
```

```
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelos no logran predecir estos datos, pero el que está más cerca es el modelo 3.

Entonces el "mejor modelo" resulta ser el 3, pero como fue entrenado con este dataset se lo va a penalizar, seguido por el 2 y el que peor predice es el 1.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-0.5	-1.5
2	0	-0.5	-1.5
3	1	-0.75	0.0

Prueba 3

Se tomaron dos observaciones correspondientes a Biedma del 2012 y a Florentino Ameghino de 2009.

total_ovinos	cant_lluvia	kilos_lana
132385	46.40	544941.0
326770	137.7	1188846.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [19]: resultado_prediccion_1 = modelo_lana_1.predict([[46.40,132385],
[137.7,326770]])
```

```
In [20]: resultado_prediccion_2 = modelo_lana_2.predict([[46.40,132385],
[137.7,326770]])
```



```
In [21]: resultado_prediccion_3 = modelo_lana_3.predict([[46.40,132385],  
[137.7,326770]])
```

```
In [22]: dato_real =[[544941.0],[1188846.0]]
```

```
In [23]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[23]: 'Score o R2 de la predicción 1: 0.97'
```

```
In [24]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[24]: 'Score o R2 de la predicción 2: 0.99'
```

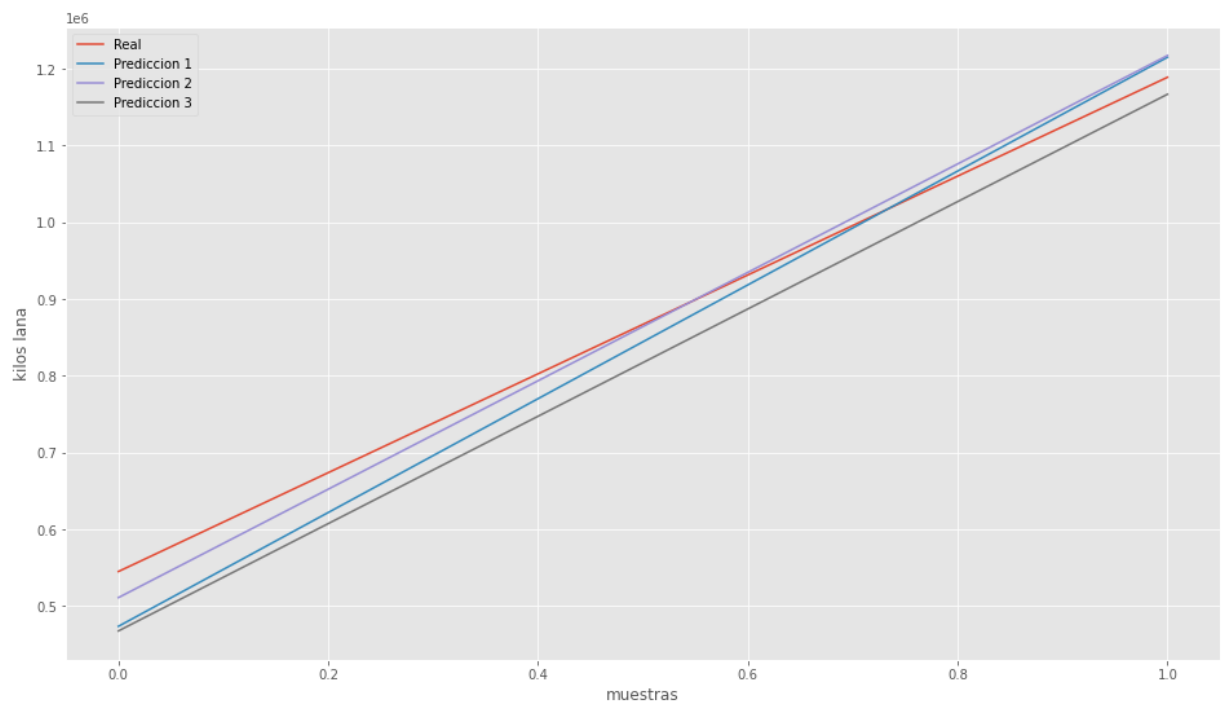
```
In [25]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

```
Out[25]: 'Score o R2 de la predicción 3: 0.97'
```

Se observa que todos los modelos logran predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [26]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.plot(range(len(dato_real)),dato_real,label="Real")  
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,  
1")  
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,  
2")  
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,  
3")  
plt.xlabel('muestras')  
plt.ylabel('kilos lana')  
plt.legend()  
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelo logran predecir estos datos.

El mejor modelo resulta ser el 2, seguido por el 3 y el 1 que comparten el mismo score.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	1	0	-0.5
2	2	0	0.5
3	1	-0.25	0.75

Prueba 4

Se tomaron dos obsevaciones correspondientes a Rawson del 2014 y a Río Senguer de 2013.

total_ovinos	cant_lluvia	kilos_lana
51630	187.2	166901.0
689360	105.77	2240084.0

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [27]: resultado_prediccion_1 = modelo_lana_1.predict([[187.2,51630],
[105.77,689360]])
```

```
In [28]: resultado_prediccion_2 = modelo_lana_2.predict([[187.2,51630],
[105.77,689360]])
```

```
In [29]: resultado_prediccion_3 = modelo_lana_3.predict([[187.2,51630],
[105.77,689360]])
```

```
In [30]: dato_real = [[166901.0],[2240084.0]]
```

```
In [31]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_1))
```

```
Out[31]: 'Score o R2 de la predicción 1: 0.94'
```

```
In [32]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_2))
```

```
Out[32]: 'Score o R2 de la predicción 2: 0.95'
```

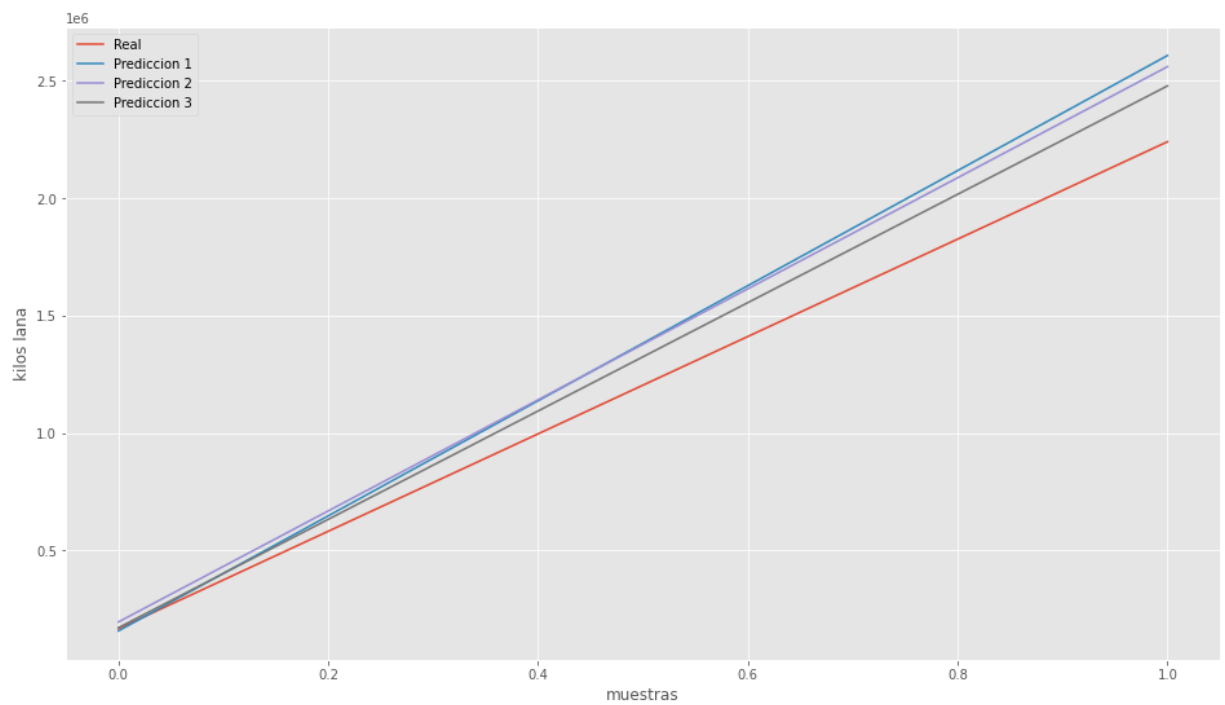
```
In [33]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_3))
```

```
Out[33]: 'Score o R2 de la predicción 3: 0.97'
```

Se observa que todos los modelos logran predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [34]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)), dato_real, label="Real")
plt.plot(range(len(resultado_prediccion_1)), resultado_prediccion_1, label="Predicción 1")
plt.plot(range(len(resultado_prediccion_2)), resultado_prediccion_2, label="Predicción 2")
plt.plot(range(len(resultado_prediccion_3)), resultado_prediccion_3, label="Predicción 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelo logran predecir estos datos.

El mejor modelo resulta ser el 3, seguido por el 2 y el 1.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	1	0	0.5
2	1	0	1.5
3	2	-0.25	2.5

Prueba 5

Se tomaron dos obsevaciones correspondientes a Gaiman del 2014 y a Futalefú de 2014.

total_ovinos	cant_lluvia	kilos_lana
199824	137.93	637321.0
109363	614.79	360959.0

```
In [35]: resultado_prediccion_1 = modelo_lana_1.predict([[137.93,199824],
[614.79,109363]])
```

```
In [36]: resultado_prediccion_2 = modelo_lana_2.predict([[137.93,199824],
[614.79,109363]])
```

```
In [37]: resultado_prediccion_3 = modelo_lana_3.predict([[137.93,199824],
[614.79,109363]])
```

```
In [38]: dato_real = [[637321.0],[360959.0]]
```

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [39]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_1))
```

```
Out[39]: 'Score o R2 de la predicción 1: 0.78'
```

```
In [40]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_2))
```

```
Out[40]: 'Score o R2 de la predicción 2: 0.68'
```

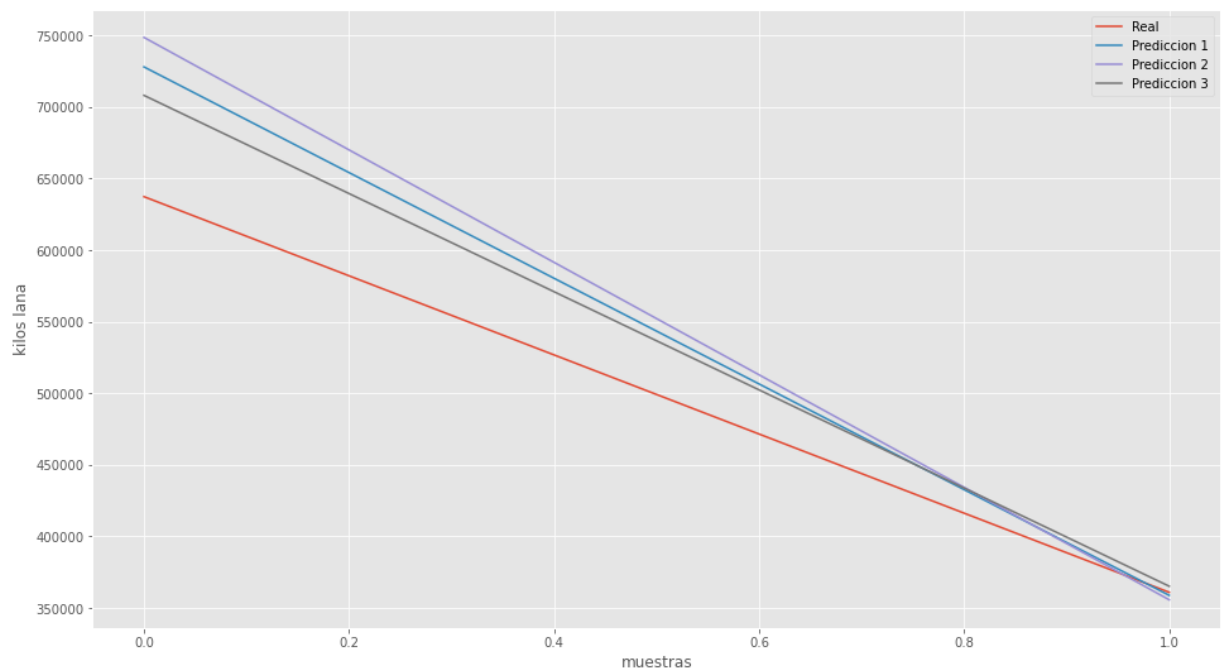
```
In [41]: 'Score o R2 de la predicción: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_3))
```

```
Out[41]: 'Score o R2 de la predicción: 0.87'
```

Se observa que todos los modelos logran predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [42]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)), dato_real, label="Real")
plt.plot(range(len(resultado_prediccion_1)), resultado_prediccion_1, label="Predicción 1")
plt.plot(range(len(resultado_prediccion_2)), resultado_prediccion_2, label="Predicción 2")
plt.plot(range(len(resultado_prediccion_3)), resultado_prediccion_3, label="Predicción 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelo logran predecir estos datos.

Pero de todas maneras vamos a empezar a realizar la comparación. El mejor modelo resulta ser el 3, seguido por el 1 y por último el 2.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	1	0	1.5
2	1	0	2.5
3	2	-0.25	4.25

Pruebas con una tupla

La idea acá es simular que un productor cuenta con los datos de la lluvia y el total de ovinos, del año actual respectivo. Entonces le pasamos ambos datos al modelo, y con ellos hace la predicción para ese año, dándonos los kilos de lana, dato que también conocemos y nos sirve para comparar el valor real con el valor predicho.

Prueba 1

Se tomo una observación del dataset correspondiente a Cushamen del 2014.

total_ovinos	cant_lluvia	kilos_lana
173238	640.4	578184.0

In [43]: `dato_real = 578184.0`

In [44]: `resultado_prediccion_1 = modelo_lana_1.predict([[640.4,173238]])
round(resultado_prediccion_1[0][0] - dato_real,2)`

Out[44]: 24461.03

```
In [45]: resultado_prediccion_2 = modelo_lana_2.predict([[640.4,173238]])  
round(resultado_prediccion_2[0][0] - dato_real,2)
```

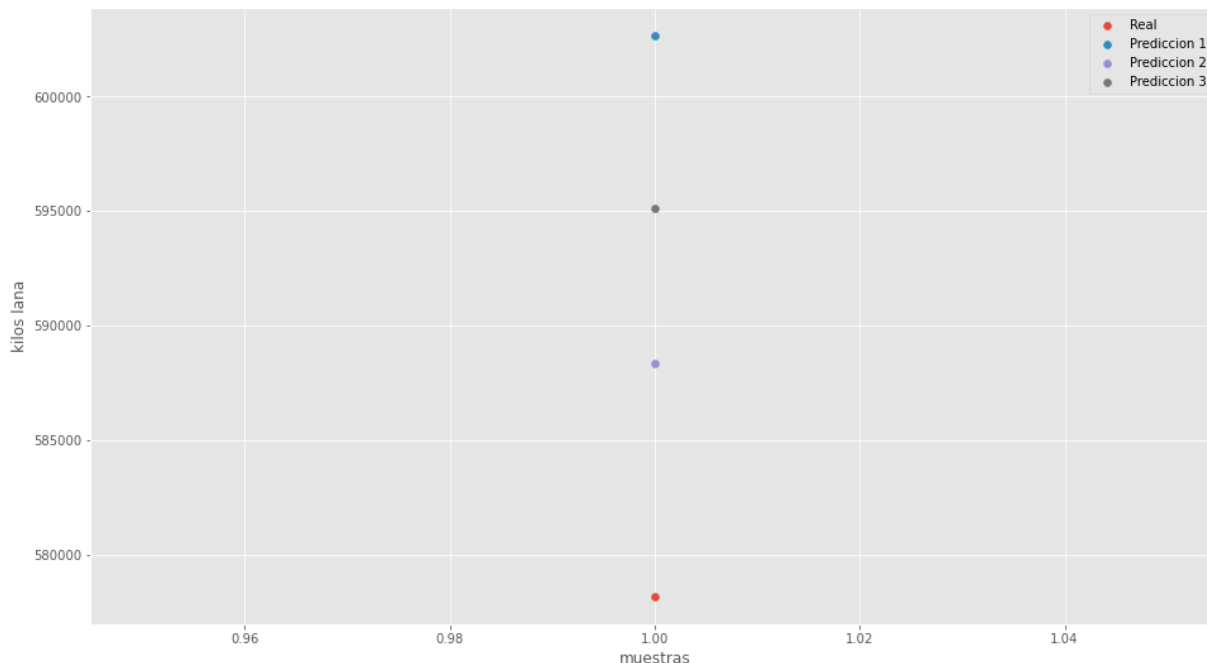
Out[45]: 10171.46

```
In [46]: resultado_prediccion_3 = modelo_lana_3.predict([[640.4,173238]])  
round(resultado_prediccion_3[0][0] - dato_real,2)
```

Out[46]: 16912.82

En estos tres resultados, vemos que los modelos predijeron kilogramos de lana demás, es decir el modelo 1 se paso por 24461.03 kgs, el modelo 2 se paso por 10171.46 kgs y el modelo 3 se paso por 16912.82 kgs.

```
In [47]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.scatter(y=dato_real, x=1, label="Real")  
plt.scatter(y=resultado_prediccion_1,x=1, label="Prediccion 1")  
plt.scatter(y=resultado_prediccion_2,x=1,label="Prediccion 2")  
plt.scatter(y=resultado_prediccion_3,x=1,label="Prediccion 3")  
plt.xlabel('muestras')  
plt.ylabel('kilos lana')  
plt.legend()  
plt.show()
```



El mejor modelo resulta ser el 2, seguido por el 3 y por último el 1. Cómo acá no se puede calcular el R2, por estar trabajando con un solo dato, solo se va a tener en cuenta cuál es el modelo que esta mas cerca del dato real.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	0	1.5
2	1	0	3.5
3	0	-0.25	4.0

Prueba 2

Se tomo una observación del dataset correspondiente a Sarmiento del 2015.

total_ovinos	cant_lluvia	kilos_lana
185378	130.55	650676.78

```
In [48]: dato_real = 650676.78
```

```
In [49]: resultado_prediccion_1 = modelo_lana_1.predict([[130.55,185378]])
round(resultado_prediccion_1[0][0] - dato_real,2)
```

```
Out[49]: 22221.24
```

```
In [50]: resultado_prediccion_2 = modelo_lana_2.predict([[130.55,185378]])
round(resultado_prediccion_2[0][0] - dato_real,2)
```

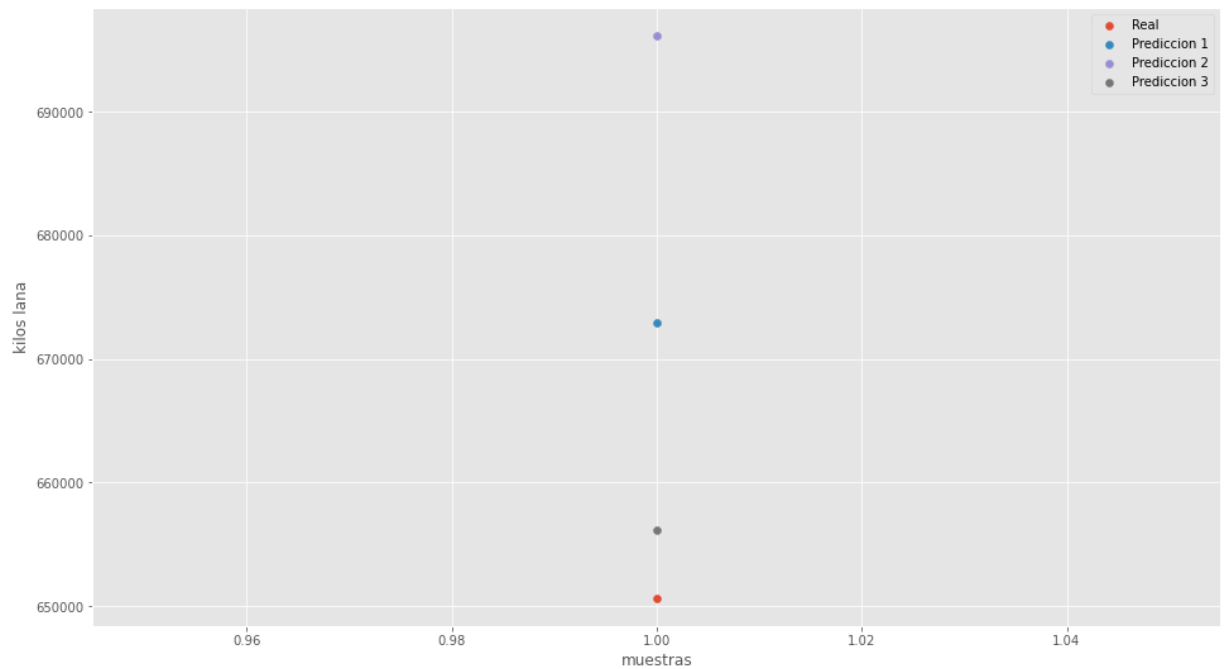
```
Out[50]: 45477.59
```

```
In [51]: resultado_prediccion_3 = modelo_lana_3.predict([[130.55,185378]])
round(resultado_prediccion_3[0][0] - dato_real,2)
```

```
Out[51]: 5460.0
```

En estos tres resultados, vemos que los modelos predijeron kilogramos de lana demás, es decir el modelo 1 se paso por 22221.24 kgs, el modelo 2 se paso por 45477.59 kgs y el modelo 3 se paso por 5460.0 kgs.

```
In [52]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.scatter(y=dato_real, x=1, label="Real")
plt.scatter(y=resultado_prediccion_1, x=1, label="Prediccion 1")
plt.scatter(y=resultado_prediccion_2, x=1, label="Prediccion 2")
plt.scatter(y=resultado_prediccion_3, x=1, label="Prediccion 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```

El mejor modelo resulta ser el 3, seguido por el 1 y por último el 2. Cómo acá no se puede calcular el R2, por estar trabajando con un solo dato, solo se va a tener en cuenta cuál es el modelo que esta mas cerca del dato real.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	0	1.5
2	0	0	3.5
3	1	-0.25	4.75

Resultados

El mejor **modelo es el 3** con un puntaje final de 4.75, seguido del modelo 2 con 3.5 y por ultimo el modelo 1 con 1.5 puntos. Sin embargo, teniendo en cuenta que el modelo 3 fue entrenado con el dataset que se esta probando, quizás el mejor *modelo es el 2*, ya que generaliza bastante bien.

3.8. Modelo de lana seleccionado

Después de realizar una comparación, el modelo que mejor funciono fue el 3, entrenado con los datos de departamentos de Chubut actualizados con los datos de las encuestas; que si bien en un principio parecía tener overfit debido a las altas métricas fue capaz de generalizar bien para datos que no conocía. Por lo tanto es el que va ser utilizado en la aplicación.

En la siguiente celda de código, se importarán varias librerías, entre ellas:

- Pandas para manejar los datos a través de DataFrames.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados de las predicciones mediante gráficos.
- De Scikit-learn se utilizarán la métricas **r2_score**.
- Por último, se usará **load_pickle** para recuperar el modelo entrenado guardado previamente.

```
In [1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from statsmodels.iolib.smpickle import load_pickle
```

Cargamos el modelo para mostrar como se vería en la aplicación.

```
In [2]: modelo_lana = load_pickle('modelos/modelo_prediccion_lana_3')
```

Se creo un modelo con datos aleatorios del dataset 1 y 2, para ver con un conjunto de datos cuál sería el resultado.

```
In [3]: data = pd.read_csv("datos/datos_reducidos_lana.csv")
data
```

```
Out[3]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_l
0	2012-12-31	CHUBUT	BIEDMA	5785	93378	30168	48447	16071	193849	64913
1	2014-12-31	CHUBUT	BIEDMA	4785	82706	29379	35155	29900	181925	64913
2	2019-12-31	CHUBUT	BIEDMA	3606	70938	32085	33311	38587	178527	61591
3	2020-12-31	CHUBUT	BIEDMA	3084	82806	29134	33727	25394	174145	63045
4	2008-12-31	CHUBUT	CUSHAMEN	4578	86402	28200	19288	44487	182955	53239
5	2013-12-31	CHUBUT	CUSHAMEN	12516	125655	31127	22615	33934	225847	69720
6	2014-12-31	CHUBUT	CUSHAMEN	4198	77263	26512	18975	46290	173238	57818
7	2015-	CHUBUT	CUSHAMEN	5764	115205	30865	26410	36446	214690	75600

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_l
	12-31									
8	2020-12-31	CHUBUT	CUSHAMEN	5385	124862	33570	28086	52545	244448	70875
9	2009-12-31	CHUBUT	FLORENTINO AMEGHINO	7970	131131	56083	47690	83896	326770	118884
10	2014-12-31	CHUBUT	FLORENTINO AMEGHINO	9623	128662	54128	39480	50249	282142	119679
11	2016-12-31	CHUBUT	FLORENTINO AMEGHINO	11974	159866	63418	47987	25964	309209	120540
12	2017-12-31	CHUBUT	FLORENTINO AMEGHINO	8064	131186	54791	48486	85190	327717	100609
13	2007-12-31	CHUBUT	FUTALEUFU	5129	81560	24740	17626	21341	150396	78720
14	2008-12-31	CHUBUT	FUTALEUFU	5639	90494	30119	24634	28237	179123	70848
15	2009-12-31	CHUBUT	FUTALEUFU	5155	85752	28633	24140	26350	170030	56153
16	2019-12-31	CHUBUT	FUTALEUFU	3775	56479	14981	14614	27926	117775	36274
17	2014-12-31	CHUBUT	GAIMAN	4936	90039	27772	26376	50701	199824	63732
18	2020-12-31	CHUBUT	GAIMAN	5295	197372	27275	46137	42834	318913	78165
19	2012-12-31	CHUBUT	LANGUIÑEO	8550	141153	52659	54014	45432	301808	98829
20	2016-12-31	CHUBUT	LANGUIÑEO	7588	112547	47178	49593	17445	234351	99540
21	2017-12-31	CHUBUT	PASO DE INDIOS	7789	128838	43440	53838	30833	264738	90160
22	2018-12-31	CHUBUT	PASO DE INDIOS	8198	128880	47244	57146	32537	274005	90160
23	2000-12-31	CHUBUT	RAWSON	2355	57320	20632	24405	18609	123321	25840
24	2004-12-31	CHUBUT	RAWSON	2769	43180	14374	16714	21962	98999	34154
25	2018-12-31	CHUBUT	RAWSON	3558	68936	20065	19143	21294	132996	21280
26	2019-12-31	CHUBUT	RAWSON	2198	33686	11357	13155	16450	76846	25359
27	2020-12-31	CHUBUT	RAWSON	2273	74096	18058	19267	22102	135796	20520
28	2009-12-31	CHUBUT	RIO SENGUERR	23858	349639	175695	126080	124871	800143	293094
29	2010-12-31	CHUBUT	RIO SENGUERR	14701	246297	97591	82269	126654	567512	218805
30	2011-12-31	CHUBUT	RIO SENGUERR	24926	348657	160536	146116	105368	785603	289156

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_l
31	2013-12-31	CHUBUT	SARMIENTO	6978	102420	31288	33644	27841	202171	83796
32	2014-12-31	CHUBUT	SARMIENTO	7348	106471	31146	34529	30598	210092	87709
33	2017-12-31	CHUBUT	SARMIENTO	5633	78931	26286	31383	33832	176065	63559
34	2008-12-31	CHUBUT	TEHUELCHES	14960	221763	107149	126501	109720	580093	304560
35	2009-12-31	CHUBUT	TEHUELCHES	17055	241352	99948	130175	100704	589234	241392
36	2015-12-31	CHUBUT	TELSEN	4477	76246	29205	29821	35267	175016	63880
37	2016-12-31	CHUBUT	TELSEN	4896	80262	26471	39243	20290	171162	63700
38	2017-12-31	CHUBUT	TELSEN	4640	77556	27296	36570	17750	163812	63700

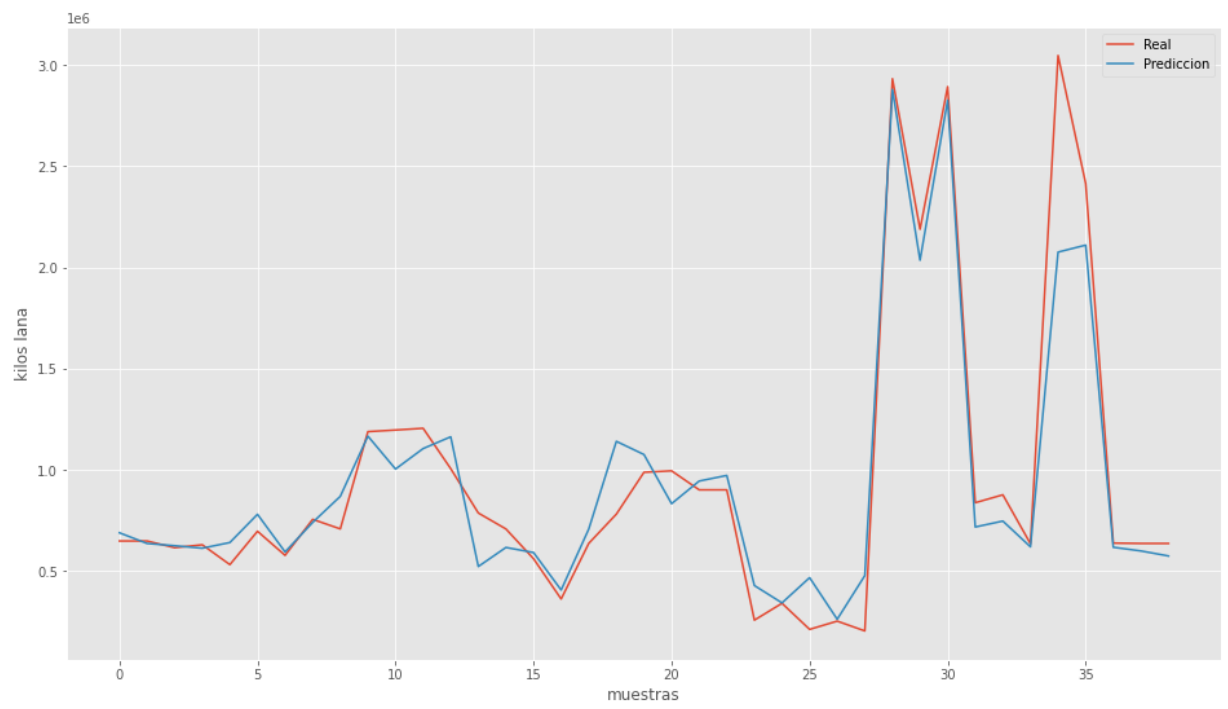
```
In [4]: X = data[['cant_lluvia', 'total_ovinos']]
        y = data[['kilos_lana']]
```

```
In [5]: resultado_prediccion = modelo_lana.predict(X)
```

```
In [8]: 'Score o R2 de la predicción: {:.2%}'.format(r2_score(y,
        resultado_prediccion))
```

```
Out[8]: 'Score o R2 de la predicción: 91.91%'
```

```
In [10]: plt.plot(range(len(y)),y,label="Real")
        plt.plot(range(len(resultado_prediccion)),resultado_prediccion,label="Predicción")
        plt.rcParams['figure.figsize'] = (16, 9)
        plt.style.use('ggplot')
        plt.xlabel('muestras')
        plt.ylabel('kilos lana')
        plt.legend()
        plt.show()
```



Vemos que el modelo predice bien con un 92% de exactitud.

3.9. Entrenamiento modelo de corderos - Dataset 1

Vamos a entrenar un modelo de regresión lineal múltiple con el primer dataset obtenido.

En la siguiente línea de código, se importarán varias bibliotecas que brindan una amplia gama de herramientas para el análisis estadístico y el modelado de datos, así como para la visualización de los resultados.

- Pandas se utiliza para manejar los datos a través de estructuras más flexibles, en este caso, se trabajará con DataFrames.
- Numpy se usará para calcular la métrica RMSE y para ayudar con el gráfico 3D a través de sus funciones linspace para definir el rango de valores para dibujar el plano y meshgrid.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados del entrenamiento mediante gráficos.
- Scikit-learn (sklearn) se usa para crear el modelo lineal. Train_test_split se utilizará para separar los datos en entrenamiento y prueba, y cross_val_score y KFold se utilizarán como formas de medir que el modelo funciona bien y no tiene mucho overfitting. Además, se utilizarán las siguientes métricas: mean_squared_error, r2_score, explained_variance_score y mean_absolute_error.
- Por último, se usará save_pickle para guardar el modelo entrenado.

```
In [2]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score,
explained_variance_score, mean_absolute_error

from statsmodels.iolib.smpickle import save_pickle
```

Lo siguiente que vamos a hacer es recuperar el dataset que tiene los datos por departamento de la producción y clima de Chubut.

```
In [3]: dataset =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut.csv")
dataset
```

```
Out[3]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_la
0	2012-12-31	CHUBUT	BIEDMA	5785	93378	30168	48447	16071	193849	649130

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_la
1	2013-12-31	CHUBUT	BIEDMA	5376	82998	25341	47350	26832	187897	620176
2	2014-12-31	CHUBUT	BIEDMA	4785	82706	29379	35155	29900	181925	649130
3	2015-12-31	CHUBUT	BIEDMA	4440	89262	29663	35437	32343	191145	672480
4	2016-12-31	CHUBUT	BIEDMA	4452	82295	33662	33470	21326	175205	653800
...	...	...	...	...	...	...	...	...	...	...
160	2016-12-31	CHUBUT	TELSEN	4896	80262	26471	39243	20290	171162	637000
161	2017-12-31	CHUBUT	TELSEN	4640	77556	27296	36570	17750	163812	637000
162	2018-12-31	CHUBUT	TELSEN	4832	79117	28366	37558	21933	171806	637000
163	2019-12-31	CHUBUT	TELSEN	4781	80109	28428	38109	17731	169158	613021
164	2020-12-31	CHUBUT	TELSEN	2870	79157	25804	36901	18984	163716	614250

165 rows × 11 columns

Predicción: Corderos

Variables independientes:

- Cantidad total de lluvia acumulada de cada año (mm)
- Cantidad de ovejas

Variables dependientes:

- Cantidad de corderos

Empezamos extrayendo dos variables del conjunto de datos 'dataset', que se utilizarán posteriormente para la creación del modelo de regresión lineal.

La variable 'X_lluvia_ovejas' se crea seleccionando las columnas 'cant_lluvia' y 'ovejas' del conjunto de datos 'dataset', estas columnas se utilizan como variables independientes o predictores del modelo.

La variable 'y_corderos' se crea seleccionando la columna 'cordero/as' del conjunto de datos 'dataset', esta columna se utiliza como variable dependiente o respuesta del modelo.

In [4]:

```
X_lluvia_ovejas = dataset[['cant_lluvia', 'ovejas']]
y_corderos = dataset[['cordero/as']]
```

Lo siguiente es dividir los datos en dos conjuntos: uno para entrenamiento y otro para prueba.

- X_train y y_train corresponden a los datos de entrenamiento que se utilizarán para ajustar el modelo.

- `X_test` y `y_test` corresponden a los datos de prueba que se utilizarán para evaluar el rendimiento del modelo.

La función `train_test_split` toma como entrada los datos originales, `X_lluvia_ovejas` y `y_corderos`, y divide aleatoriamente los datos en conjuntos de entrenamiento y prueba. El parámetro `test_size` indica el porcentaje de datos que se utilizarán para prueba (en este caso, el 20% de los datos) y `random_state` es una semilla que se utiliza para asegurar que los conjuntos de entrenamiento y prueba sean siempre los mismos en cada ejecución del código.

```
In [5]: X_train, X_test, y_train, y_test = train_test_split(
                                             X_lluvia_ovejas,
                                             y_corderos,
                                             test_size=0.20,
                                             random_state=0
                                             )
```

Luego creamos un objeto `KFold` que se utiliza para realizar una validación cruzada en el conjunto de datos. La validación cruzada es una técnica que se utiliza para evaluar el rendimiento del modelo y estimar su capacidad para generalizar a nuevos datos. El objeto `KFold` divide el conjunto de datos en `k` partes iguales (en este caso, `k=5`), y utiliza cada una de ellas como conjunto de prueba y el resto como conjunto de entrenamiento. De esta manera, se obtiene una estimación del rendimiento del modelo en distintas combinaciones de entrenamiento y prueba.

Entonces para entrenar el modelo de manera más efectiva, utilizaremos el método `K-Folds` con 5 divisiones. Este enfoque consiste en iterar 5 veces, donde en cada iteración se separan 33 muestras para utilizarlas como conjunto de prueba, mientras que las 132 restantes se usan para entrenar el modelo. Luego se mide el rendimiento del modelo utilizando el conjunto de prueba apartado en cada iteración. Esto nos permite realizar 5 entrenamientos independientes y obtener 5 resultados de rendimiento. Finalmente, el rendimiento del modelo se calculará como el promedio de los 5 resultados obtenidos. De esta manera, evitamos sobrestimar el rendimiento del modelo al utilizar el mismo conjunto de prueba en todas las iteraciones.

```
In [6]: kf = KFold(n_splits=5)
```

Luego de llevar a cabo estas 5 iteraciones, se espera obtener 5 puntuaciones (score) que sean cercanas entre sí, lo que servirá como **una señal de que el modelo está rindiendo bien y que tiene una buena capacidad de generalización**. Una vez completada esta fase de validación, el siguiente paso es crear el modelo y proceder a su entrenamiento con los datos.

```
In [7]: model_corderos = linear_model.LinearRegression()
        model_corderos.fit(X_train, y_train)
```

```
Out[7]: ▼ LinearRegression
        LinearRegression()
```


Se procede a comparar los resultados del modelo inicial de entrenamiento con los obtenidos en los 5 procesos de validación cruzada, a través del promedio de estos últimos. Es posible notar que existe una **coincidencia o similitud** entre los resultados. La métrica del modelo es de *0.9*, mientras que las métricas de la validación cruzada son [0.92, 0.91, 0.891, 0.89, 0.84], cuya media es de *0.89*.

```
In [8]: score = model_corderos.score(X_train,y_train)
'Métrica del modelo en Train {:.2}'.format(score)
```

```
Out[8]: 'Métrica del modelo en Train 0.9'
```

```
In [9]: scores = cross_val_score(model_corderos, X_train, y_train, cv=kf)
scores_rounded = [round(score, 2) for score in scores]
'Métricas cross validation: {}'.format(scores_rounded)
```

```
Out[9]: 'Métricas cross validation: [0.92, 0.91, 0.91, 0.89, 0.84]'
```

```
In [10]: 'Media de cross validation {:.2}'.format(scores.mean())
```

```
Out[10]: 'Media de cross validation 0.89'
```

Finalmente, realizamos una predicción utilizando el conjunto de prueba y verificamos que también obtenemos un **buen puntaje de precisión o score**.

```
In [11]: pred_model_corderos = model_corderos.predict(X_test)
```

```
In [12]: score_pred = r2_score(y_test, pred_model_corderos)
'Metrica del modelo en Test {:.2%}'.format(score_pred)
```

```
Out[12]: 'Metrica del modelo en Test 87.06%'
```

Entonces guardamos el modelo como candidato para utilizarlo en la aplicación web.

```
In [13]: save_pickle(model_corderos, 'modelos/modelo_prediccion_corderos_1')
```

Métricas

Score

Score: El "score" es un método que se incluye en todos los algoritmos y proporciona el resultado de un criterio de evaluación por defecto para el problema al que están dirigidos. En el caso de una regresión lineal, el método score devuelve el coeficiente de determinación, también conocido como **R²**, que se define como el porcentaje de la varianza de la variable dependiente que puede ser explicada por la variable independiente. Esta métrica nos ayuda a evaluar la calidad de nuestro modelo y a compararlo con otros modelos. En general, cuanto mayor sea el valor de **R²**, mejor ajustado estará el modelo a los datos.

```
In [14]: 'Score del modelo en entrenamiento:
{:.2%}'.format(model_corderos.score(X_train, y_train))
```

```
Out[14]: 'Score del modelo en entrenamiento: 90.26%'
```

Interpretación: Un R^2 de 0.9026 significa que el modelo explica el 90.26% de la variabilidad de la producción de corderos en función de la cantidad de lluvia y el total de ovejas. Es una métrica muy aceptable.

```
In [15]: 'Score del modelo en test:
{:.2%}'.format(model_corderos.score(X_test, y_test))
```

```
Out[15]: 'Score del modelo en test: 87.06%'
```

Interpretación: el modelo tiene una precisión del 87.06% al predecir los valores de la variable dependiente (en este caso, los corderos) utilizando los valores de las variables independientes (cantidad de lluvia y número total de ovejas) en el conjunto de datos de prueba. Es decir, el modelo puede explicar aproximadamente el 87.06% de la varianza en la variable dependiente con los datos que no ha visto antes. Es una métrica muy aceptable.

Varianza explicada

Varianza explicada: es una medida del porcentaje de la varianza de la variable de salida que es explicado por el modelo. Esta métrica evalúa la variación o dispersión de los puntos de datos, y la mejor puntuación posible es 1.0, lo que indica que el modelo explica perfectamente toda la variabilidad de los datos. Sin embargo, una puntuación muy alta en esta métrica también puede indicar un riesgo de overfitting, es decir, que el modelo se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a nuevos datos. En sklearn, se puede utilizar el método **explained_variance_score** para calcular esta métrica.

```
In [16]: 'Varianza Explicada:
{:.2%}'.format(explained_variance_score(y_test,
pred_model_corderos))
```

```
Out[16]: 'Varianza Explicada: 87.90%'
```

Interpretación: En este caso, la varianza explicada del modelo de regresión lineal es del 87.90%, lo que indica que el modelo es capaz de explicar el 87.90% de la variabilidad observada en la variable de salida (corderos), en base a las variables de entrada proporcionadas (lluvia, total de ovejas). Una varianza explicada del 87.90% indica que el modelo es bueno para predecir la variable de salida en función de las variables de entrada sin llegar a indicar un sobreajuste. Es una métrica muy aceptable.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.

```
In [17]: 'MAE: {:.0f}'.format(mean_absolute_error(y_test,
pred_model_corderos))
```

```
Out[17]: 'MAE: 7617'
```

Interpretación: El valor obtenido de 7617 indica que, en promedio, el modelo se equivoca en esa cantidad al predecir los corderos producidos. Es decir, en promedio, las predicciones del modelo tienen un error absoluto medio de 7617 en relación al valor real de la producción de corderos. Una forma de interpretar esta métrica es pensar en el valor de la producción de corderos que se está analizando y evaluar si el error medio absoluto es aceptable en relación a ese valor, dado que estamos hablando de números altos en muchos casos el valor no es tan malo. Es una métrica relativamente buena.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica MSE, la cual es la media de las diferencias entre el valor objetivo y el predicho al cuadrado, y al elevar al cuadrado magnifica los errores grandes. La ventaja de RMSE es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [19]: 'RMSE: {:.0f}'.format(mean_squared_error(y_test,
pred_model_corderos, squared = False))
```

```
Out[19]: 'RMSE: 9521'
```

Interpretación: En este caso, la raíz cuadrada del MSE de 9521 indica que, en promedio, la diferencia entre los valores predichos y los valores observados de corderos es de aproximadamente 9521. Es una métrica relativamente buena.

R cuadrado

R cuadrado (R^2): es una métrica utilizada para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo.

Es importante destacar que el resultado del coeficiente de determinación, es un valor que oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto, menos fiable será. Esto se debe a que R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

En resumen, el valor de R^2 nos proporciona una medida de la calidad del ajuste de nuestro modelo a los datos de entrenamiento, siendo un valor cercano a 1 deseable para garantizar una buena capacidad predictiva.

El coeficiente de determinación se puede calcular utilizando la función `r2_score` de la biblioteca sklearn.

```
In [20]: 'R2: {:.2%}'.format(r2_score(y_test, pred_model_corderos))
```

```
Out[20]: 'R2: 87.06%'
```

Interpretación: En este caso, el valor de R cuadrado es 87.06%, lo que significa que aproximadamente el 87.06% de la variación en los datos de corderos producidos se puede explicar por las variables

independientes utilizadas en el modelo de regresión lineal. Entonces el modelo de regresión lineal explica en un 87% la variable de salida, en este caso los corderos. Es una métrica muy buena.

El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas. Es decir, si al modelo se añaden cinco variables explicativas que guardan poca relación con lo que se intenta predecir, el R cuadrado aumentará. Es por ello que muchos expertos econométricos, estadísticos y matemáticos se oponen al uso del R cuadrado como medida representativa de la bondad del ajuste real.

R cuadrado ajustado

La fórmula del R cuadrado ajustado es como sigue: $R^2_{ajustado} = 1 - \frac{(n-1)}{n-k-1} * (1 - R^2)$

Donde:

- R^2 : coeficiente de determinación
- n : número de observaciones
- k : número de predictores (variables explicativas) en el modelo.

Se va a crear una función que aplique esta fórmula a los datos.

```
In [21]: def r2_ajustado(n,k,r2):  
        return round(1-((n-1)/(n-k-1))*(1-r2),4)  
  
n = len(y_test)  
k = 2  
r2 = r2_score(y_test, pred_model_corderos)  
  
'R2 ajustado: {:.2%}'.format(r2_ajustado(n,k,r2))
```

```
Out[21]: 'R2 ajustado: 86.20%'
```

Interpretación: un R2 ajustado de 86.20% indica que el modelo explica aproximadamente el 86.20% de la variabilidad en la variable objetivo, teniendo en cuenta las variables explicativas incluidas en el modelo. Es decir, la bondad de ajuste es del 86% y por lo tanto el modelo explica en un 86% los corderos. Es una métrica aceptable.

Gráficos predicción de corderos

A continuación, se presentarán dos gráficos para visualizar el rendimiento del modelo de regresión lineal múltiple en la predicción de la producción de corderos. En primer lugar, se utilizará un gráfico de dispersión en 3D para representar la relación entre las variables de entrada y la variable de salida. En segundo lugar, se empleará un gráfico de líneas para comparar los valores reales de la producción de corderos con los valores predichos por el modelo. Esto permitirá una mejor comprensión del desempeño del modelo y de la precisión de las predicciones.

Gráfico regresión lineal múltiple

Recordemos que las variables, `X_lluvia_ovejas` y `y_corderos`, son subconjuntos del dataset original que contienen las columnas "cant_lluvia", "ovejas" y "cordero/as", respectivamente.

Lo siguiente es transformar el dataframe `X_lluvia_ovejas` en un arreglo numpy de dos dimensiones (2D) para poder ser utilizado en el modelo de regresión lineal.

La función `values` de un dataframe de pandas devuelve un arreglo numpy que representa los datos del dataframe, y `reshape` se utiliza para darle una nueva forma o dimensión al arreglo, en este caso convirtiéndolo en una matriz de dos columnas (una para cada variable predictora).

El parámetro `-1` en `reshape` indica a NumPy que calcule el número necesario de filas en función de los datos y de la cantidad de columnas especificadas, de tal manera que los datos originales del dataframe se mantengan.

```
In [22]: X = X_lluvia_ovejas.values.reshape(-1,2)
```

Luego le asignamos a la variable `Y` la columna `'cordero/as'` del dataframe original dataset. Esto se hace para tener una variable separada que contenga solo la variable de salida (en este caso `'cordero/as'`) y poder usarla en las operaciones posteriores.

```
In [23]: Y = y_corderos
```

La siguiente celda contiene líneas de código que crean una malla de puntos en 2D para hacer una visualización en 3D de la superficie de regresión del modelo.

- `x_pred = np.linspace(7.1, 1041.16, 30)` crea un array de 30 valores igualmente espaciados que van desde 7.1 hasta 1041.16, que representan los posibles valores de la variable `cant_lluvia`.
- `y_pred = np.linspace(38558, 443288, 30)` crea un array de 30 valores igualmente espaciados que van desde 38558 hasta 443288, que representan los posibles valores de la variable `ovejas`.
- `xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)` crea una malla de puntos en 2D a partir de los dos arrays anteriores, donde `xx_pred` representa los valores de `cant_lluvia` y `yy_pred` los valores de `ovejas`.
- `model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T` crea un array `model_viz` que contiene una lista de todas las posibles combinaciones de valores para `cant_lluvia` y `ovejas`, que será usado para hacer la predicción y graficar la superficie de regresión.

```
In [24]: x_pred = np.linspace(7.1, 1041.16, 30) # range of cant_lluvia
y_pred = np.linspace(38558, 443288, 30) # range of ovejas
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T
```

La siguiente celda prepara los datos necesarios para visualizar el modelo de regresión lineal múltiple en un gráfico en 3D.

- `x = X[:, 0]` extrae todos los valores de la primera columna de `X` y los asigna a la variable `x`, que representará los valores de la variable independiente `cant_lluvia`.
- `y = X[:, 1]` extrae todos los valores de la segunda columna de `X` y los asigna a la variable `y`, que representará los valores de la variable independiente `ovejas`.
- `z = Y` asigna todos los valores de la variable dependiente `cordero/as` a la variable `z`, que representará la variable que se quiere predecir en función de las variables independientes `x` e `y`.

In [25]:

```
x = X[:, 0]
y = X[:, 1]
z = Y
```

El código `predicted = model_corderos.predict(model_viz)` utiliza el modelo de regresión lineal previamente entrenado (`model_corderos`) para hacer una predicción de los valores de salida (`cordero/as`) correspondientes a todas las combinaciones posibles de valores de entrada para `cant_lluvia` y `ovejas` que se encuentran en la malla de puntos en 2D (`model_viz`).

Esto se logra utilizando el método `predict` del objeto del modelo de regresión lineal, que toma como entrada una matriz `X` de características y devuelve una matriz de valores de salida predichos correspondientes a cada entrada. En este caso, la matriz `X` es la malla de puntos `model_viz` que contiene todas las posibles combinaciones de valores de `cant_lluvia` y `ovejas` . Los valores predichos se almacenan en la variable `predicted`.

In [26]:

```
predicted = model_corderos.predict(model_viz)
```

El código `r2 = model_corderos.score(X, Y)` calcula el coeficiente de determinación R^2 del modelo de regresión lineal, que como vimos antes, es una medida estadística que indica la proporción de la varianza de la variable dependiente (`cordero/as`) que es explicada por el modelo en función de las variables independientes (`cant_lluvia` y `ovejas`).

Este método toma como entrada una matriz de características `X` (en este caso, la matriz `X` que se creó anteriormente a partir de las variables `cant_lluvia` y `ovejas`) y una matriz de valores de salida `Y` (en este caso, la matriz `Y` que contiene los valores reales de `cordero/as`). Luego, compara los valores de salida predichos por el modelo para cada muestra de `X` con los valores reales correspondientes de `Y`, y devuelve un valor entre menos infinito y 1 que indica la calidad del ajuste del modelo.

Un valor de R^2 más cercano a 1 indica que el modelo explica una gran proporción de la varianza de la variable dependiente, lo que indica un mejor ajuste del modelo a los datos. En este caso, el valor de R^2 obtenido se almacena en la variable `r2` .

In [27]:

```
r2 = model_corderos.score(X, Y)
```

El siguiente código crea una figura con tres subplots en 3D utilizando la biblioteca Matplotlib. Cada subplot contiene una visualización de la superficie de regresión lineal para diferentes ángulos de vista.

En cada subplot, se hace una visualización de los datos de entrada y de los valores predichos por el modelo. Los datos de entrada se representan como puntos negros en el fondo, mientras que los valores predichos se representan como puntos azules en la superficie.

También se establecen las etiquetas de los ejes y los parámetros del locator para la ubicación de las etiquetas de los ejes.

Finalmente, se establecen las posiciones de vista de los subplots y se agrega un título a la figura que muestra el valor del coeficiente de determinación (R^2) obtenido del modelo.

La llamada a la función `fig.tight_layout()` se encarga de ajustar automáticamente los espacios entre los subplots para que no haya superposiciones.

```
In [28]: plt.style.use('default')
fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

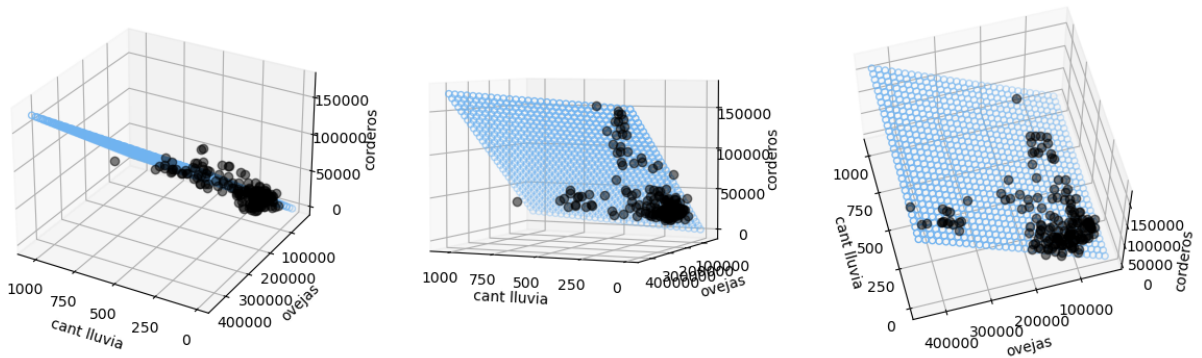
for ax in axes:
    flattenX = x.flatten()
    flattenY = y.flatten()
    flattenZ = z.values.flatten()
    ax.plot(
        flattenX, flattenY, flattenZ, color='k',
        zorder=15, linestyle='none', marker='o', alpha=0.5
    )
    ax.scatter(
        xx_pred.flatten(), yy_pred.flatten(), predicted,
        facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0'
    )
    ax.set_xlabel('cant lluvia', fontsize=10)
    ax.set_ylabel('ovejas', fontsize=10)
    ax.set_zlabel('corderos', fontsize=10)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)
```

```
fig.tight_layout()
```

$$R^2 = 0.90$$

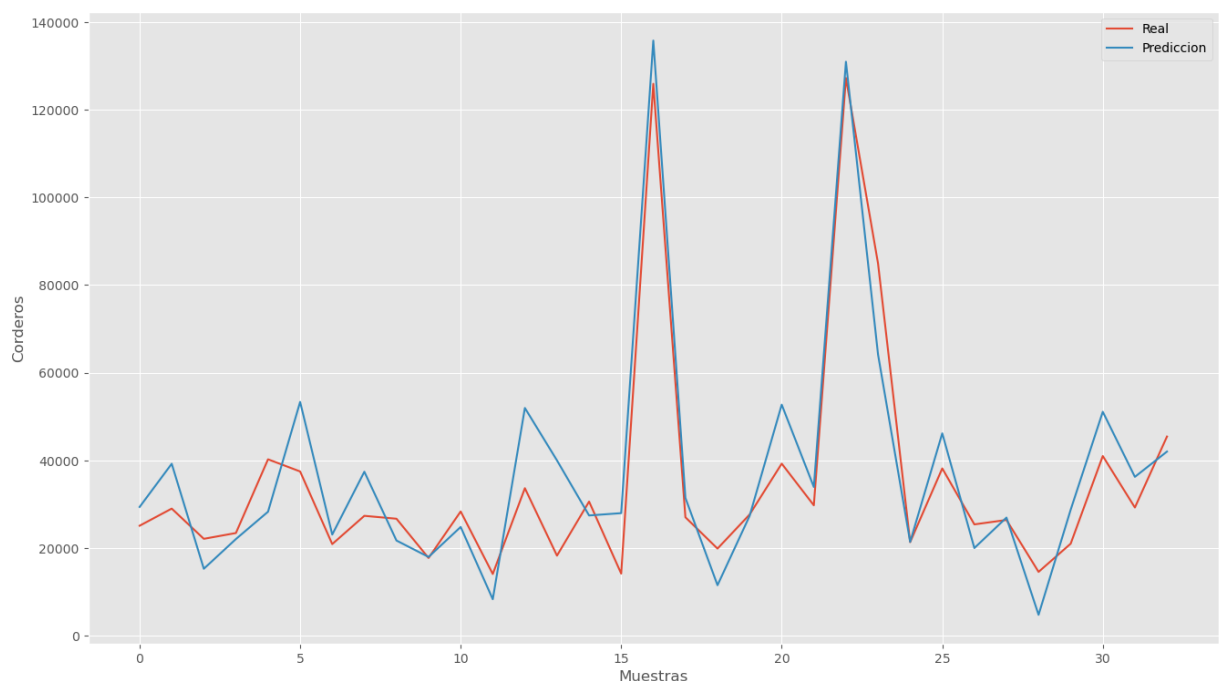


Se observa que los datos no se ajustan perfectamente al plano, lo que va acorde con la información extraída de las métricas.

Gráfico predicción vs real

A continuación, se va a generar un gráfico 2D para comparar los datos reales con la predicción realizada por el modelo.

```
In [30]: plt.plot(range(len(y_test)),y_test,label="Real")
plt.plot(range(len(pred_model_corderos)),pred_model_corderos,label="")
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('Muestras')
plt.ylabel('Corderos')
plt.legend()
plt.show()
```



Podemos observar que la línea roja que representa los valores reales y la línea azul que representa la predicción del modelo tienen una tendencia similar, sin embargo, existen algunas desviaciones. Esto sugiere que el modelo no está sobreajustando los datos de entrenamiento y es más probable que tenga un mejor desempeño al predecir valores para nuevos datos.

3.10. Entrenamiento modelo de corderos - Dataset 2

Vamos a entrenar un modelo de regresión lineal múltiple con el primer dataset obtenido.

En la siguiente línea de código, se importarán varias bibliotecas que brindan una amplia gama de herramientas para el análisis estadístico y el modelado de datos, así como para la visualización de los resultados.

- Pandas se utiliza para manejar los datos a través de estructuras más flexibles, en este caso, se trabajará con DataFrames.
- Numpy se usará para calcular la métrica RMSE y para ayudar con el gráfico 3D a través de sus funciones linspace para definir el rango de valores para dibujar el plano y meshgrid.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados del entrenamiento mediante gráficos.
- Scikit-learn (sklearn) se usa para crear el modelo lineal. Train_test_split se utilizará para separar los datos en entrenamiento y prueba, y cross_val_score y KFold se utilizarán como formas de medir que el modelo funciona bien y no tiene mucho overfitting. Además, se utilizarán las siguientes métricas: mean_squared_error, r2_score, explained_variance_score y mean_absolute_error.
- Por último, se usará save_pickle para guardar el modelo entrenado.

```
In [1]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score,
explained_variance_score, mean_absolute_error

from statsmodels.iolib.smpickle import save_pickle
```

Lo siguiente que vamos a hacer es recuperar el dataset que tiene los datos por departamento de la producción y clima de Chubut.

```
In [2]: dataset =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_mirab
dataset
```

```
Out[2]:
```

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lan
0	2012-12-31	1294000	BIEDMA	5785	93378	30168	48447	16071	193849	649130

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lan
1	2013-12-31	1294000	BIEDMA	5376	82998	25341	47350	26832	187897	620176
2	2014-12-31	1294000	BIEDMA	4785	82706	29379	35155	29900	181925	649130
3	2015-12-31	1294000	BIEDMA	4440	89262	29663	35437	32343	191145	672480
4	2016-12-31	1294000	BIEDMA	4452	82295	33662	33470	21326	175205	653800
...	...	...	...	...	...	...	...	...	...	...
196	2016-12-31	1989300	TELSEN	4896	80262	26471	39243	20290	171162	637000
197	2017-12-31	1989300	TELSEN	4640	77556	27296	36570	17750	163812	637000
198	2018-12-31	1989300	TELSEN	4832	79117	28366	37558	21933	171806	637000
199	2019-12-31	1989300	TELSEN	4781	80109	28428	38109	17731	169158	613021
200	2020-12-31	1989300	TELSEN	2870	79157	25804	36901	18984	163716	614250

201 rows × 17 columns

Predicción: Corderos

Variables independientes:

- Cantidad total de lluvia acumulada de cada año (mm)
- Cantidad de ovejas

Variables dependientes:

- Cantidad de corderos

Empezamos extrayendo dos variables del conjunto de datos 'dataset', que se utilizarán posteriormente para la creación del modelo de regresión lineal.

La variable 'X_lluvia_ovejas' se crea seleccionando las columnas 'cant_lluvia' y 'ovejas' del conjunto de datos 'dataset', estas columnas se utilizan como variables independientes o predictores del modelo.

La variable 'y_corderos' se crea seleccionando la columna 'cordero/as' del conjunto de datos 'dataset', esta columna se utiliza como variable dependiente o respuesta del modelo.

```
In [3]: X_lluvia_ovejas = dataset[['cant_lluvia', 'ovejas']]
        y_corderos = dataset[['cordero/as']]
```

Lo siguiente es dividir los datos en dos conjuntos: uno para entrenamiento y otro para prueba.

- X_train y y_train corresponden a los datos de entrenamiento que se utilizarán para ajustar el modelo.

- `X_test` y `y_test` corresponden a los datos de prueba que se utilizarán para evaluar el rendimiento del modelo.

La función `train_test_split` toma como entrada los datos originales, `X_lluvia_ovejas` y `y_corderos`, y divide aleatoriamente los datos en conjuntos de entrenamiento y prueba. El parámetro `test_size` indica el porcentaje de datos que se utilizarán para prueba (en este caso, el 20% de los datos) y `random_state` es una semilla que se utiliza para asegurar que los conjuntos de entrenamiento y prueba sean siempre los mismos en cada ejecución del código.

```
In [4]: X_train, X_test, y_train, y_test = train_test_split(
                                                X_lluvia_ovejas,
                                                y_corderos,
                                                test_size=0.20,
                                                random_state=0
                                              )
```

Luego creamos un objeto `KFold` que se utiliza para realizar una validación cruzada en el conjunto de datos. La validación cruzada es una técnica que se utiliza para evaluar el rendimiento del modelo y estimar su capacidad para generalizar a nuevos datos. El objeto `KFold` divide el conjunto de datos en `k` partes iguales (en este caso, `k=5`), y utiliza cada una de ellas como conjunto de prueba y el resto como conjunto de entrenamiento. De esta manera, se obtiene una estimación del rendimiento del modelo en distintas combinaciones de entrenamiento y prueba.

Entonces para entrenar el modelo de manera más efectiva, utilizaremos el método **K-Folds** con 5 divisiones. Este enfoque consiste en iterar 5 veces, donde en cada iteración se separan 40 muestras para utilizarlas como conjunto de prueba, mientras que las 161 restantes se usan para entrenar el modelo. Luego se mide el rendimiento del modelo utilizando el conjunto de prueba apartado en cada iteración. Esto nos permite realizar 5 entrenamientos independientes y obtener 5 resultados de rendimiento. Finalmente, el rendimiento del modelo se calculará como el promedio de los 5 resultados obtenidos. De esta manera, evitamos sobrestimar el rendimiento del modelo al utilizar el mismo conjunto de prueba en todas las iteraciones.

```
In [5]: kf = KFold(n_splits=5)
```

Luego de llevar a cabo estas 5 iteraciones, se espera obtener 5 puntuaciones (score) que sean cercanas entre sí, lo que servirá como **una señal de que el modelo está rindiendo bien y que tiene una buena capacidad de generalización**. Una vez completada esta fase de validación, el siguiente paso es crear el modelo y proceder a su entrenamiento con los datos.

```
In [6]: model_corderos = linear_model.LinearRegression()
        model_corderos.fit(X_train, y_train)
```

```
Out[6]: ▼ LinearRegression
        LinearRegression()
```

Se procede a comparar los resultados del modelo inicial de entrenamiento con los obtenidos en los 5 procesos de validación cruzada, a través del promedio de estos últimos. Es posible notar que existe una **coincidencia o similitud** entre los resultados. La métrica del modelo es de *0.9*, mientras que las métricas de la validación cruzada son [0.91, 0.89, 0.90, 0.90, 0.87], cuya media es de *0.89*.

```
In [7]: score = model_corderos.score(X_train,y_train)
'Métrica del modelo en Train {:.2}'.format(score)
```

```
Out[7]: 'Métrica del modelo en Train 0.9'
```

```
In [8]: scores = cross_val_score(model_corderos, X_train, y_train, cv=kf)
scores_rounded = [round(score, 2) for score in scores]
'Métricas cross validation: {}'.format(scores_rounded)
```

```
Out[8]: 'Métricas cross validation: [0.91, 0.89, 0.9, 0.9, 0.87]'
```

```
In [9]: 'Media de cross validation {:.2}'.format(scores.mean())
```

```
Out[9]: 'Media de cross validation 0.89'
```

Finalmente, realizamos una predicción utilizando el conjunto de prueba y verificamos que también obtenemos un **buen puntaje de precisión o score**.

```
In [10]: pred_model_corderos = model_corderos.predict(X_test)
```

```
In [11]: score_pred = r2_score(y_test, pred_model_corderos)
'Metrica del modelo en Test {:.2%}'.format(score_pred)
```

```
Out[11]: 'Metrica del modelo en Test 92.34%'
```

Entonces guardamos el modelo como candidato para utilizarlo en la aplicación web.

```
In [12]: save_pickle(model_corderos, 'modelos/modelo_prediccion_corderos_2')
```

Métricas

Score

Score: El "score" es un método que se incluye en todos los algoritmos y proporciona el resultado de un criterio de evaluación por defecto para el problema al que están dirigidos. En el caso de una regresión lineal, el método score devuelve el coeficiente de determinación, también conocido como **R²**, que se define como el porcentaje de la varianza de la variable dependiente que puede ser explicada por la variable independiente. Esta métrica nos ayuda a evaluar la calidad de nuestro modelo y a compararlo con otros modelos. En general, cuanto mayor sea el valor de **R²**, mejor ajustado estará el modelo a los datos.

```
In [13]: 'Score del modelo en entrenamiento:
{:.2%}'.format(model_corderos.score(X_train, y_train))
```

```
Out[13]: 'Score del modelo en entrenamiento: 89.87%'
```

Interpretación: Un R^2 de 0.8987 significa que el modelo explica el 89.87% de la variabilidad de la producción de corderos en función de la cantidad de lluvia y el total de ovejas. Es una métrica muy aceptable.

```
In [14]: 'Score del modelo en test:
{:.2%}'.format(model_corderos.score(X_test, y_test))
```

```
Out[14]: 'Score del modelo en test: 92.34%'
```

Interpretación: el modelo tiene una precisión del 92.34% al predecir los valores de la variable dependiente (en este caso, los corderos) utilizando los valores de las variables independientes (cantidad de lluvia y número total de ovejas) en el conjunto de datos de prueba. Es decir, el modelo puede explicar aproximadamente el 92.34% de la varianza en la variable dependiente con los datos que no ha visto antes. Es una métrica muy aceptable.

Varianza explicada

Varianza explicada: es una medida del porcentaje de la varianza de la variable de salida que es explicado por el modelo. Esta métrica evalúa la variación o dispersión de los puntos de datos, y la mejor puntuación posible es 1.0, lo que indica que el modelo explica perfectamente toda la variabilidad de los datos. Sin embargo, una puntuación muy alta en esta métrica también puede indicar un riesgo de overfitting, es decir, que el modelo se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a nuevos datos. En sklearn, se puede utilizar el método **explained_variance_score** para calcular esta métrica.

```
In [15]: 'Varianza Explicada:
{:.2%}'.format(explained_variance_score(y_test,
pred_model_corderos))
```

```
Out[15]: 'Varianza Explicada: 92.47%'
```

Interpretación: En este caso, la varianza explicada del modelo de regresión lineal es del 92.47%, lo que indica que el modelo es capaz de explicar el 92.47% de la variabilidad observada en la variable de salida (corderos), en base a las variables de entrada proporcionadas (lluvia, total de ovejas). Una varianza explicada del 92.47% indica que el modelo es bueno para predecir la variable de salida en función de las variables de entrada sin llegar a indicar un sobreajuste. Es una métrica muy aceptable.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.

```
In [16]: 'MAE: {:.0f}'.format(mean_absolute_error(y_test,
pred_model_corderos))
```

```
Out[16]: 'MAE: 9568'
```

Interpretación: El valor obtenido de 9568 indica que, en promedio, el modelo se equivoca en esa cantidad al predecir los corderos producidos. Es decir, en promedio, las predicciones del modelo tienen un error absoluto medio de 9568 en relación al valor real de la producción de corderos. Una forma de interpretar esta métrica es pensar en el valor de la producción de corderos que se está analizando y evaluar si el error medio absoluto es aceptable en relación a ese valor, dado que estamos hablando de números altos en muchos casos el valor no es tan malo. Es una métrica relativamente buena.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica MSE, la cual es la media de las diferencias entre el valor objetivo y el predicho al cuadrado, y al elevar al cuadrado magnifica los errores grandes. La ventaja de RMSE es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [18]: 'RMSE: {:.0f}'.format(mean_squared_error(y_test,
pred_model_corderos, squared = False))
```

```
Out[18]: 'RMSE: 12305'
```

Interpretación: En este caso, la raíz cuadrada del MSE de 12305 indica que, en promedio, la diferencia entre los valores predichos y los valores observados de corderos es de aproximadamente 12305. Es una métrica relativamente buena.

R cuadrado

R cuadrado (R^2): es una métrica utilizada para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo.

Es importante destacar que el resultado del coeficiente de determinación, es un valor que oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto, menos fiable será. Esto se debe a que R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

En resumen, el valor de R^2 nos proporciona una medida de la calidad del ajuste de nuestro modelo a los datos de entrenamiento, siendo un valor cercano a 1 deseable para garantizar una buena capacidad predictiva.

El coeficiente de determinación se puede calcular utilizando la función `r2_score` de la biblioteca sklearn.

```
In [19]: 'R2:{:.2%}'.format(r2_score(y_test, pred_model_corderos))
```

```
Out[19]: 'R2:92.34%'
```

Interpretación: En este caso, el valor de R cuadrado es 92.34%, lo que significa que aproximadamente el 92.34% de la variación en los datos de corderos producidos se puede explicar por las variables

independientes utilizadas en el modelo de regresión lineal. Entonces el modelo de regresión lineal explica en un 92% la variable de salida, en este caso los corderos. Es una métrica muy buena.

El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas. Es decir, si al modelo se añaden cinco variables explicativas que guardan poca relación con lo que se intenta predecir, el R cuadrado aumentará. Es por ello que muchos expertos econométricos, estadísticos y matemáticos se oponen al uso del R cuadrado como medida representativa de la bondad del ajuste real.

R cuadrado ajustado

La fórmula del R cuadrado ajustado es como sigue: $R^2_{ajustado} = 1 - \frac{(n-1)}{n-k-1} * (1 - R^2)$

Donde:

- R^2 : coeficiente de determinación
- n : número de observaciones
- k : número de predictores (variables explicativas) en el modelo.

Se va a crear una función que aplique esta fórmula a los datos.

```
In [20]: def r2_ajustado(n,k,r2):  
        return round(1-((n-1)/(n-k-1))*(1-r2),4)  
  
n = len(y_test)  
k = 2  
r2 = r2_score(y_test, pred_model_corderos)  
  
'R2 ajustado: {:.2%}'.format(r2_ajustado(n,k,r2))
```

```
Out[20]: 'R2 ajustado: 91.94%'
```

Interpretación: un R2 ajustado de 91.94% indica que el modelo explica aproximadamente el 91.94% de la variabilidad en la variable objetivo, teniendo en cuenta las variables explicativas incluidas en el modelo. Es decir, la bondad de ajuste es del 92% y por lo tanto el modelo explica en un 92% los corderos. Es una métrica aceptable.

Gráficos predicción de corderos

A continuación, se presentarán dos gráficos para visualizar el rendimiento del modelo de regresión lineal múltiple en la predicción de la producción de corderos. En primer lugar, se utilizará un gráfico de dispersión en 3D para representar la relación entre las variables de entrada y la variable de salida. En segundo lugar, se empleará un gráfico de líneas para comparar los valores reales de la producción de corderos con los valores predichos por el modelo. Esto permitirá una mejor comprensión del desempeño del modelo y de la precisión de las predicciones.

Gráfico regresión lineal múltiple

Recordemos que las variables, `X_lluvia_ovejas` y `y_corderos`, son subconjuntos del dataset original que contienen las columnas "cant_lluvia", "ovejas" y "cordero/as", respectivamente.

Lo siguiente es transformar el dataframe `X_lluvia_ovejas` en un arreglo numpy de dos dimensiones (2D) para poder ser utilizado en el modelo de regresión lineal.

La función `values` de un dataframe de pandas devuelve un arreglo numpy que representa los datos del dataframe, y `reshape` se utiliza para darle una nueva forma o dimensión al arreglo, en este caso convirtiéndolo en una matriz de dos columnas (una para cada variable predictora).

El parámetro `-1` en `reshape` indica a NumPy que calcule el número necesario de filas en función de los datos y de la cantidad de columnas especificadas, de tal manera que los datos originales del dataframe se mantengan.

```
In [21]: X = X_lluvia_ovejas.values.reshape(-1,2)
```

Luego le asignamos a la variable `Y` la columna `'cordero/as'` del dataframe original dataset. Esto se hace para tener una variable separada que contenga solo la variable de salida (en este caso `'cordero/as'`) y poder usarla en las operaciones posteriores.

```
In [22]: Y = y_corderos
```

La siguiente celda contiene líneas de código que crean una malla de puntos en 2D para hacer una visualización en 3D de la superficie de regresión del modelo.

- `x_pred = np.linspace(7.1, 1041.16, 30)` crea un array de 30 valores igualmente espaciados que van desde 7.1 hasta 1041.16, que representan los posibles valores de la variable `cant_lluvia`.
- `y_pred = np.linspace(768, 443288, 30)` crea un array de 30 valores igualmente espaciados que van desde 768 hasta 443288, que representan los posibles valores de la variable `ovejas`.
- `xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)` crea una malla de puntos en 2D a partir de los dos arrays anteriores, donde `xx_pred` representa los valores de `cant_lluvia` y `yy_pred` los valores de `ovejas`.
- `model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T` crea un array `model_viz` que contiene una lista de todas las posibles combinaciones de valores para `cant_lluvia` y `ovejas`, que será usado para hacer la predicción y graficar la superficie de regresión.

```
In [23]: x_pred = np.linspace(7.1, 1041.16, 30) # range of cant_lluvia
y_pred = np.linspace(768, 443288, 30) # range of ovejas
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T
```

La siguiente celda prepara los datos necesarios para visualizar el modelo de regresión lineal múltiple en un gráfico en 3D.

- `x = X[:, 0]` extrae todos los valores de la primera columna de `X` y los asigna a la variable `x`, que representará los valores de la variable independiente `cant_lluvia`.
- `y = X[:, 1]` extrae todos los valores de la segunda columna de `X` y los asigna a la variable `y`, que representará los valores de la variable independiente `ovejas`.
- `z = Y` asigna todos los valores de la variable dependiente `cordero/as` a la variable `z`, que representará la variable que se quiere predecir en función de las variables independientes `x` e `y`.

In [24]:

```
x = X[:, 0]
y = X[:, 1]
z = Y
```

El código `predicted = model_corderos.predict(model_viz)` utiliza el modelo de regresión lineal previamente entrenado (`model_corderos`) para hacer una predicción de los valores de salida (`cordero/as`) correspondientes a todas las combinaciones posibles de valores de entrada para `cant_lluvia` y `ovejas` que se encuentran en la malla de puntos en 2D (`model_viz`).

Esto se logra utilizando el método `predict` del objeto del modelo de regresión lineal, que toma como entrada una matriz `X` de características y devuelve una matriz de valores de salida predichos correspondientes a cada entrada. En este caso, la matriz `X` es la malla de puntos `model_viz` que contiene todas las posibles combinaciones de valores de `cant_lluvia` y `ovejas` . Los valores predichos se almacenan en la variable `predicted`.

In [25]:

```
predicted = model_corderos.predict(model_viz)
```

El código `r2 = model_corderos.score(X, Y)` calcula el coeficiente de determinación R^2 del modelo de regresión lineal, que como vimos antes, es una medida estadística que indica la proporción de la varianza de la variable dependiente (`cordero/as`) que es explicada por el modelo en función de las variables independientes (`cant_lluvia` y `ovejas`).

Este método toma como entrada una matriz de características `X` (en este caso, la matriz `X` que se creó anteriormente a partir de las variables `cant_lluvia` y `ovejas`) y una matriz de valores de salida `Y` (en este caso, la matriz `Y` que contiene los valores reales de `cordero/as`). Luego, compara los valores de salida predichos por el modelo para cada muestra de `X` con los valores reales correspondientes de `Y`, y devuelve un valor entre menos infinito y 1 que indica la calidad del ajuste del modelo.

Un valor de R^2 más cercano a 1 indica que el modelo explica una gran proporción de la varianza de la variable dependiente, lo que indica un mejor ajuste del modelo a los datos. En este caso, el valor de R^2 obtenido se almacena en la variable `r2` .

In [26]:

```
r2 = model_corderos.score(X, Y)
```

El siguiente código crea una figura con tres subplots en 3D utilizando la biblioteca Matplotlib. Cada subplot contiene una visualización de la superficie de regresión lineal para diferentes ángulos de vista.

En cada subplot, se hace una visualización de los datos de entrada y de los valores predichos por el modelo. Los datos de entrada se representan como puntos negros en el fondo, mientras que los valores predichos se representan como puntos azules en la superficie.

También se establecen las etiquetas de los ejes y los parámetros del locator para la ubicación de las etiquetas de los ejes.

Finalmente, se establecen las posiciones de vista de los subplots y se agrega un título a la figura que muestra el valor del coeficiente de determinación (R^2) obtenido del modelo.

La llamada a la función `fig.tight_layout()` se encarga de ajustar automáticamente los espacios entre los subplots para que no haya superposiciones.

```
In [27]: plt.style.use('default')
fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

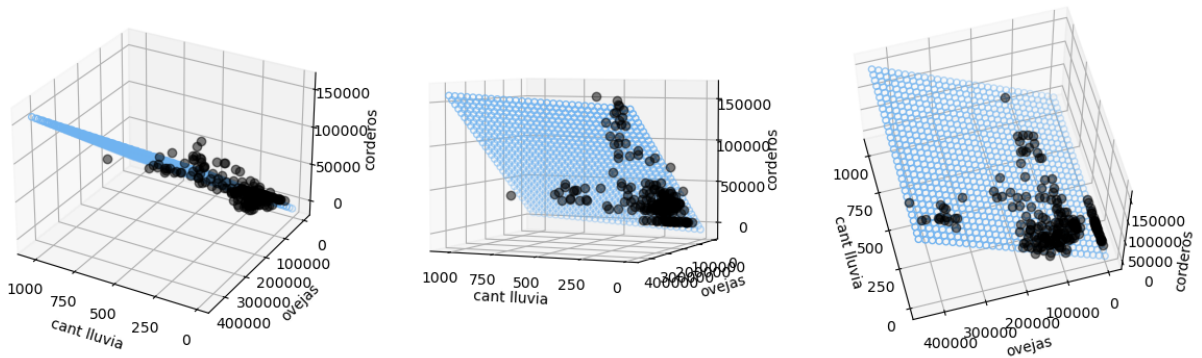
for ax in axes:
    flattenX = x.flatten()
    flattenY = y.flatten()
    flattenZ = z.values.flatten()
    ax.plot(
        flattenX, flattenY, flattenZ, color='k',
        zorder=15, linestyle='none', marker='o', alpha=0.5
    )
    ax.scatter(
        xx_pred.flatten(), yy_pred.flatten(), predicted,
        facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0'
    )
    ax.set_xlabel('cant lluvia', fontsize=10)
    ax.set_ylabel('ovejas', fontsize=10)
    ax.set_zlabel('corderos', fontsize=10)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)
```

```
fig.tight_layout()
```

$$R^2 = 0.91$$



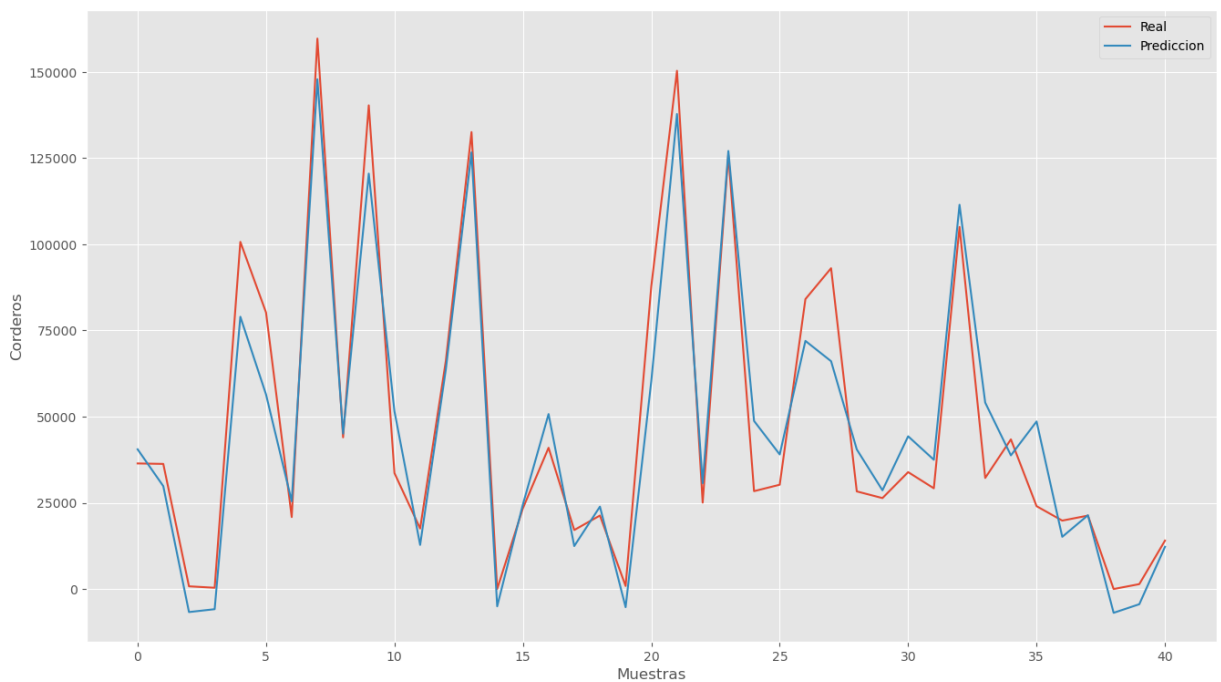
Se observa que los datos no se ajustan perfectamente al plano, lo que va acorde con la información extraída de las métricas.

Gráfico predicción vs real

A continuación, se va a generar un gráfico 2D para comparar los datos reales con la predicción realizada por el modelo.

In [29]:

```
plt.plot(range(len(y_test)),y_test,label="Real")
plt.plot(range(len(pred_model_corderos)),pred_model_corderos,label="")
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('Muestras')
plt.ylabel('Corderos')
plt.legend()
plt.show()
```



Podemos observar que la línea roja que representa los valores reales y la línea azul que representa la predicción del modelo tienen una tendencia similar, sin embargo, existen algunas desviaciones. Esto sugiere que el modelo no está sobreajustando los datos de entrenamiento y es más probable que tenga un mejor desempeño al predecir valores para nuevos datos.

3.11. Entrenamiento modelo de corderos - Dataset 3

Vamos a entrenar un modelo de regresión lineal múltiple con el primer dataset obtenido.

En la siguiente línea de código, se importarán varias bibliotecas que brindan una amplia gama de herramientas para el análisis estadístico y el modelado de datos, así como para la visualización de los resultados.

- Pandas se utiliza para manejar los datos a través de estructuras más flexibles, en este caso, se trabajará con DataFrames.
- Numpy se usará para calcular la métrica RMSE y para ayudar con el gráfico 3D a través de sus funciones linspace para definir el rango de valores para dibujar el plano y meshgrid.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados del entrenamiento mediante gráficos.
- Scikit-learn (sklearn) se usa para crear el modelo lineal. Train_test_split se utilizará para separar los datos en entrenamiento y prueba, y cross_val_score y KFold se utilizarán como formas de medir que el modelo funciona bien y no tiene mucho overfitting. Además, se utilizarán las siguientes métricas: mean_squared_error, r2_score, explained_variance_score y mean_absolute_error.
- Por último, se usará save_pickle para guardar el modelo entrenado.

```
In [1]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score,
explained_variance_score, mean_absolute_error

from statsmodels.iolib.smpickle import save_pickle
```

Lo siguiente que vamos a hacer es recuperar el dataset que tiene los datos por departamento de la producción y clima de Chubut.

```
In [2]: dataset =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_actua
dataset
```

```
Out[2]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
0	2012-12-31	CHUBUT	BIEDMA	2740	52320	22319	25489	29517	132385

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	corderos_corderas	total_ovinos
1	2013-12-31	CHUBUT	BIEDMA	3925	70677	27227	33389	36961	172179
2	2014-12-31	CHUBUT	BIEDMA	3223	61195	29491	29688	34926	158523
3	2015-12-31	CHUBUT	BIEDMA	3885	67528	28319	33296	38117	171145
4	2016-12-31	CHUBUT	BIEDMA	3743	70451	30050	33038	37923	175205
...	...	...	...	...	...	...	...	...	...
159	2016-12-31	CHUBUT	TELSEN	4711	74125	28715	31728	31883	171162
160	2017-12-31	CHUBUT	TELSEN	4496	71149	27574	28606	31987	163812
161	2018-12-31	CHUBUT	TELSEN	4634	74179	28793	35979	28221	171806
162	2019-12-31	CHUBUT	TELSEN	4305	72515	27648	35273	29417	169158
163	2020-12-31	CHUBUT	TELSEN	4416	70736	27084	31418	30062	163716

164 rows × 10 columns

Predicción: Corderos

Variables independientes:

- Cantidad total de lluvia acumulada de cada año (mm)
- Cantidad de ovejas

Variables dependientes:

- Cantidad de corderos

Empezamos extrayendo dos variables del conjunto de datos 'dataset', que se utilizarán posteriormente para la creación del modelo de regresión lineal.

La variable 'X_lluvia_ovejas' se crea seleccionando las columnas 'cant_lluvia' y 'ovejas' del conjunto de datos 'dataset', estas columnas se utilizan como variables independientes o predictores del modelo.

La variable 'y_corderos' se crea seleccionando la columna 'cordero/as' del conjunto de datos 'dataset', esta columna se utiliza como variable dependiente o respuesta del modelo.

```
In [3]: X_lluvia_ovejas = dataset[['cant_lluvia', 'ovejas']]
        y_corderos = dataset[['corderos_corderas']]
```

Lo siguiente es dividir los datos en dos conjuntos: uno para entrenamiento y otro para prueba.

- X_train y y_train corresponden a los datos de entrenamiento que se utilizarán para ajustar el modelo.

- `X_test` y `y_test` corresponden a los datos de prueba que se utilizarán para evaluar el rendimiento del modelo.

La función `train_test_split` toma como entrada los datos originales, `X_lluvia_ovejas` y `y_corderos`, y divide aleatoriamente los datos en conjuntos de entrenamiento y prueba. El parámetro `test_size` indica el porcentaje de datos que se utilizarán para prueba (en este caso, el 20% de los datos) y `random_state` es una semilla que se utiliza para asegurar que los conjuntos de entrenamiento y prueba sean siempre los mismos en cada ejecución del código.

```
In [4]: X_train, X_test, y_train, y_test = train_test_split(
                                                X_lluvia_ovejas,
                                                y_corderos,
                                                test_size=0.20,
                                                random_state=0
                                              )
```

Luego creamos un objeto `KFold` que se utiliza para realizar una validación cruzada en el conjunto de datos. La validación cruzada es una técnica que se utiliza para evaluar el rendimiento del modelo y estimar su capacidad para generalizar a nuevos datos. El objeto `KFold` divide el conjunto de datos en `k` partes iguales (en este caso, `k=5`), y utiliza cada una de ellas como conjunto de prueba y el resto como conjunto de entrenamiento. De esta manera, se obtiene una estimación del rendimiento del modelo en distintas combinaciones de entrenamiento y prueba.

Entonces para entrenar el modelo de manera más efectiva, utilizaremos el método `K-Folds` con 5 divisiones. Este enfoque consiste en iterar 5 veces, donde en cada iteración se separan 33 muestras para utilizarlas como conjunto de prueba, mientras que las 131 restantes se usan para entrenar el modelo. Luego se mide el rendimiento del modelo utilizando el conjunto de prueba apartado en cada iteración. Esto nos permite realizar 5 entrenamientos independientes y obtener 5 resultados de rendimiento. Finalmente, el rendimiento del modelo se calculará como el promedio de los 5 resultados obtenidos. De esta manera, evitamos sobrestimar el rendimiento del modelo al utilizar el mismo conjunto de prueba en todas las iteraciones.

```
In [5]: kf = KFold(n_splits=5)
```

Luego de llevar a cabo estas 5 iteraciones, se espera obtener 5 puntuaciones (score) que sean cercanas entre sí, lo que servirá como **una señal de que el modelo está rindiendo bien y que tiene una buena capacidad de generalización**. Una vez completada esta fase de validación, el siguiente paso es crear el modelo y proceder a su entrenamiento con los datos.

```
In [6]: model_corderos = linear_model.LinearRegression()
        model_corderos.fit(X_train, y_train)
```

```
Out[6]: ▼ LinearRegression
        LinearRegression()
```


Se procede a comparar los resultados del modelo inicial de entrenamiento con los obtenidos en los 5 procesos de validación cruzada, a través del promedio de estos últimos. Es posible notar que existe una **coincidencia o similitud** entre los resultados. La métrica del modelo es de *0.97*, mientras que las métricas de la validación cruzada son [0.96, 0.99, 0.98, 0.96, 0.94], cuya media es de *0.97*.

```
In [7]: score = model_corderos.score(X_train,y_train)
'Métrica del modelo en Train {:.2}'.format(score)
```

```
Out[7]: 'Métrica del modelo en Train 0.97'
```

```
In [8]: scores = cross_val_score(model_corderos, X_train, y_train, cv=kf)
scores_rounded = [round(score, 2) for score in scores]
'Métricas cross validation: {}'.format(scores_rounded)
```

```
Out[8]: 'Métricas cross validation: [0.96, 0.99, 0.98, 0.96, 0.94]'
```

```
In [9]: 'Media de cross validation {:.2}'.format(scores.mean())
```

```
Out[9]: 'Media de cross validation 0.97'
```

Finalmente, realizamos una predicción utilizando el conjunto de prueba y verificamos que también obtenemos un **buen puntaje de precisión o score**.

```
In [10]: pred_model_corderos = model_corderos.predict(X_test)
```

```
In [11]: score_pred = r2_score(y_test, pred_model_corderos)
'Metrica del modelo en Test {:.2%}'.format(score_pred)
```

```
Out[11]: 'Metrica del modelo en Test 97.87%'
```

Entonces guardamos el modelo como candidato para utilizarlo en la aplicación web.

```
In [12]: save_pickle(model_corderos, 'modelos/modelo_prediccion_corderos_3')
```

Métricas

Score

Score: El "score" es un método que se incluye en todos los algoritmos y proporciona el resultado de un criterio de evaluación por defecto para el problema al que están dirigidos. En el caso de una regresión lineal, el método score devuelve el coeficiente de determinación, también conocido como **R²**, que se define como el porcentaje de la varianza de la variable dependiente que puede ser explicada por la variable independiente. Esta métrica nos ayuda a evaluar la calidad de nuestro modelo y a compararlo con otros modelos. En general, cuanto mayor sea el valor de **R²**, mejor ajustado estará el modelo a los datos.

```
In [13]: 'Score del modelo en entrenamiento:
{:.2%}'.format(model_corderos.score(X_train, y_train))
```

```
Out[13]: 'Score del modelo en entrenamiento: 97.45%'
```

Interpretación: Un R^2 de 0.9745 significa que el modelo explica el 97.45% de la variabilidad de la producción de corderos en función de la cantidad de lluvia y el total de ovejas. Es métrica una demasiado buena.

```
In [14]: 'Score del modelo en test:
{:.2%}'.format(model_corderos.score(X_test, y_test))
```

```
Out[14]: 'Score del modelo en test: 97.87%'
```

Interpretación: el modelo tiene una precisión del 97.87% al predecir los valores de la variable dependiente (en este caso, los corderos) utilizando los valores de las variables independientes (cantidad de lluvia y número total de ovejas) en el conjunto de datos de prueba. Es decir, el modelo puede explicar aproximadamente el 97.87% de la varianza en la variable dependiente con los datos que no ha visto antes. Es métrica una demasiado buena.

Varianza explicada

Varianza explicada: es una medida del porcentaje de la varianza de la variable de salida que es explicado por el modelo. Esta métrica evalúa la variación o dispersión de los puntos de datos, y la mejor puntuación posible es 1.0, lo que indica que el modelo explica perfectamente toda la variabilidad de los datos. Sin embargo, una puntuación muy alta en esta métrica también puede indicar un riesgo de overfitting, es decir, que el modelo se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a nuevos datos. En sklearn, se puede utilizar el método **explained_variance_score** para calcular esta métrica.

```
In [15]: 'Varianza Explicada:
{:.2%}'.format(explained_variance_score(y_test,
pred_model_corderos))
```

```
Out[15]: 'Varianza Explicada: 97.87%'
```

Interpretación: En este caso, la varianza explicada del modelo de regresión lineal es del 97.87%, lo que indica que el modelo es capaz de explicar el 97.87% de la variabilidad observada en la variable de salida (corderos), en base a las variables de entrada proporcionadas (lluvia, total de ovejas). Una varianza explicada del 97.87% indica que el modelo es bueno para predecir la variable de salida en función de las variables de entrada sin llegar a indicar un sobreajuste. Es una métrica muy alta.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.

```
In [16]: 'MAE: {:.0f}'.format(mean_absolute_error(y_test,
pred_model_corderos))
```

```
Out[16]: 'MAE: 3752'
```

Interpretación: El valor obtenido de 3752 indica que, en promedio, el modelo se equivoca en esa cantidad al predecir los corderos producidos. Es decir, en promedio, las predicciones del modelo tienen un error absoluto medio de 3752 en relación al valor real de la producción de corderos. Una forma de interpretar esta métrica es pensar en el valor de la producción de corderos que se está analizando y evaluar si el error medio absoluto es aceptable en relación a ese valor, dado que estamos hablando de números altos en muchos casos el valor no es tan malo. Es una métrica relativamente buena.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica MSE, la cual es la media de las diferencias entre el valor objetivo y el predicho al cuadrado, y al elevar al cuadrado magnifica los errores grandes. La ventaja de RMSE es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [18]: 'RMSE: {:.0f}'.format(mean_squared_error(y_test,
pred_model_corderos, squared = False))
```

```
Out[18]: 'RMSE: 5575'
```

Interpretación: En este caso, la raíz cuadrada del MSE de 5575 indica que, en promedio, la diferencia entre los valores predichos y los valores observados de corderos es de aproximadamente 5575. Es una métrica relativamente buena.

R cuadrado

R cuadrado (R^2): es una métrica utilizada para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo.

Es importante destacar que el resultado del coeficiente de determinación, es un valor que oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto, menos fiable será. Esto se debe a que R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

En resumen, el valor de R^2 nos proporciona una medida de la calidad del ajuste de nuestro modelo a los datos de entrenamiento, siendo un valor cercano a 1 deseable para garantizar una buena capacidad predictiva.

El coeficiente de determinación se puede calcular utilizando la función `r2_score` de la biblioteca sklearn.

```
In [19]: 'R2: {:.2%}'.format(r2_score(y_test, pred_model_corderos))
```

```
Out[19]: 'R2: 97.87%'
```

Interpretación: En este caso, el valor de R cuadrado es 97.87%, lo que significa que aproximadamente el 97.87% de la variación en los datos de corderos producidos se puede explicar por las variables

independientes utilizadas en el modelo de regresión lineal. Entonces el modelo de regresión lineal explica en un 98% la variable de salida, en este caso los corderos. Es métrica una demasiado buena.

El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas. Es decir, si al modelo se añaden cinco variables explicativas que guardan poca relación con lo que se intenta predecir, el R cuadrado aumentará. Es por ello que muchos expertos econométricos, estadísticos y matemáticos se oponen al uso del R cuadrado como medida representativa de la bondad del ajuste real.

R cuadrado ajustado

La fórmula del R cuadrado ajustado es como sigue: $R^2_{ajustado} = 1 - \frac{(n-1)}{n-k-1} * (1 - R^2)$

Donde:

- R^2 : coeficiente de determinación
- n : número de observaciones
- k : número de predictores (variables explicativas) en el modelo.

Se va a crear una función que aplique esta fórmula a los datos.

```
In [20]: def r2_ajustado(n,k,r2):  
        return round(1-((n-1)/(n-k-1))*(1-r2),4)  
  
n = len(y_test)  
k = 2  
r2 = r2_score(y_test, pred_model_corderos)  
  
'R2 ajustado: {:.2%}'.format(r2_ajustado(n,k,r2))
```

```
Out[20]: 'R2 ajustado: 97.72%'
```

Interpretación: un R2 ajustado de 97.72% indica que el modelo explica aproximadamente el 97.72% de la variabilidad en la variable objetivo, teniendo en cuenta las variables explicativas incluidas en el modelo. Es decir, la bondad de ajuste es del 98% y por lo tanto el modelo explica en un 98% los corderos. Es métrica una demasiado buena.

Gráficos predicción de corderos

A continuación, se presentarán dos gráficos para visualizar el rendimiento del modelo de regresión lineal múltiple en la predicción de la producción de corderos. En primer lugar, se utilizará un gráfico de dispersión en 3D para representar la relación entre las variables de entrada y la variable de salida. En segundo lugar, se empleará un gráfico de líneas para comparar los valores reales de la producción de corderos con los valores predichos por el modelo. Esto permitirá una mejor comprensión del desempeño del modelo y de la precisión de las predicciones.

Gráfico regresión lineal múltiple

Recordemos que las variables, `X_lluvia_ovejas` y `y_corderos`, son subconjuntos del dataset original que contienen las columnas "cant_lluvia", "ovejas" y "cordero/as", respectivamente.

Lo siguiente es transformar el dataframe `X_lluvia_ovejas` en un arreglo numpy de dos dimensiones (2D) para poder ser utilizado en el modelo de regresión lineal.

La función `values` de un dataframe de pandas devuelve un arreglo numpy que representa los datos del dataframe, y `reshape` se utiliza para darle una nueva forma o dimensión al arreglo, en este caso convirtiéndolo en una matriz de dos columnas (una para cada variable predictora).

El parámetro `-1` en `reshape` indica a NumPy que calcule el número necesario de filas en función de los datos y de la cantidad de columnas especificadas, de tal manera que los datos originales del dataframe se mantengan.

```
In [21]: X = X_lluvia_ovejas.values.reshape(-1,2)
```

Luego le asignamos a la variable `Y` la columna `'cordero/as'` del dataframe original dataset. Esto se hace para tener una variable separada que contenga solo la variable de salida (en este caso `'cordero/as'`) y poder usarla en las operaciones posteriores.

```
In [22]: Y = y_corderos
```

La siguiente celda contiene líneas de código que crean una malla de puntos en 2D para hacer una visualización en 3D de la superficie de regresión del modelo.

- `x_pred = np.linspace(7.1, 1041.16, 30)` crea un array de 30 valores igualmente espaciados que van desde 7.1 hasta 1041.16, que representan los posibles valores de la variable `cant_lluvia`.
- `y_pred = np.linspace(21913, 345839, 30)` crea un array de 30 valores igualmente espaciados que van desde 21913 hasta 345839, que representan los posibles valores de la variable `ovejas`.
- `xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)` crea una malla de puntos en 2D a partir de los dos arrays anteriores, donde `xx_pred` representa los valores de `cant_lluvia` y `yy_pred` los valores de `ovejas`.
- `model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T` crea un array `model_viz` que contiene una lista de todas las posibles combinaciones de valores para `cant_lluvia` y `ovejas`, que será usado para hacer la predicción y graficar la superficie de regresión.

```
In [23]: x_pred = np.linspace(7.1, 1041.16, 30) # range of cant_lluvia
y_pred = np.linspace(21913, 345839, 30) # range of ovejas
xx_pred, yy_pred = np.meshgrid(x_pred, y_pred)
model_viz = np.array([xx_pred.flatten(), yy_pred.flatten()]).T
```

La siguiente celda prepara los datos necesarios para visualizar el modelo de regresión lineal múltiple en un gráfico en 3D.

- `x = X[:, 0]` extrae todos los valores de la primera columna de X y los asigna a la variable x, que representará los valores de la variable independiente cant_lluvia.
- `y = X[:, 1]` extrae todos los valores de la segunda columna de X y los asigna a la variable y, que representará los valores de la variable independiente ovejas.
- `z = Y` asigna todos los valores de la variable dependiente cordero/as a la variable z, que representará la variable que se quiere predecir en función de las variables independientes x e y.

In [24]:

```
x = X[:, 0]
y = X[:, 1]
z = Y
```

El código `predicted = model_corderos.predict(model_viz)` utiliza el modelo de regresión lineal previamente entrenado (`model_corderos`) para hacer una predicción de los valores de salida (`cordero/as`) correspondientes a todas las combinaciones posibles de valores de entrada para cant_lluvia y ovejas que se encuentran en la malla de puntos en 2D (`model_viz`).

Esto se logra utilizando el método `predict` del objeto del modelo de regresión lineal, que toma como entrada una matriz `X` de características y devuelve una matriz de valores de salida predichos correspondientes a cada entrada. En este caso, la matriz `X` es la malla de puntos `model_viz` que contiene todas las posibles combinaciones de valores de `cant_lluvia` y `ovejas` . Los valores predichos se almacenan en la variable `predicted`.

In [25]:

```
predicted = model_corderos.predict(model_viz)
```

El código `r2 = model_corderos.score(X, Y)` calcula el coeficiente de determinación R^2 del modelo de regresión lineal, que como vimos antes, es una medida estadística que indica la proporción de la varianza de la variable dependiente (cordero/as) que es explicada por el modelo en función de las variables independientes (cant_lluvia y ovejas).

Este método toma como entrada una matriz de características `X` (en este caso, la matriz `X` que se creó anteriormente a partir de las variables cant_lluvia y ovejas) y una matriz de valores de salida `Y` (en este caso, la matriz `Y` que contiene los valores reales de cordero/as). Luego, compara los valores de salida predichos por el modelo para cada muestra de `X` con los valores reales correspondientes de `Y`, y devuelve un valor entre menos infinito y 1 que indica la calidad del ajuste del modelo.

Un valor de R^2 más cercano a 1 indica que el modelo explica una gran proporción de la varianza de la variable dependiente, lo que indica un mejor ajuste del modelo a los datos. En este caso, el valor de R^2 obtenido se almacena en la variable `r2` .

In [26]:

```
r2 = model_corderos.score(X, Y)
```

El siguiente código crea una figura con tres subplots en 3D utilizando la biblioteca Matplotlib. Cada subplot contiene una visualización de la superficie de regresión lineal para diferentes ángulos de vista.

En cada subplot, se hace una visualización de los datos de entrada y de los valores predichos por el modelo. Los datos de entrada se representan como puntos negros en el fondo, mientras que los valores predichos se representan como puntos azules en la superficie.

También se establecen las etiquetas de los ejes y los parámetros del locator para la ubicación de las etiquetas de los ejes.

Finalmente, se establecen las posiciones de vista de los subplots y se agrega un título a la figura que muestra el valor del coeficiente de determinación (R^2) obtenido del modelo.

La llamada a la función `fig.tight_layout()` se encarga de ajustar automáticamente los espacios entre los subplots para que no haya superposiciones.

```
In [27]: plt.style.use('default')
fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

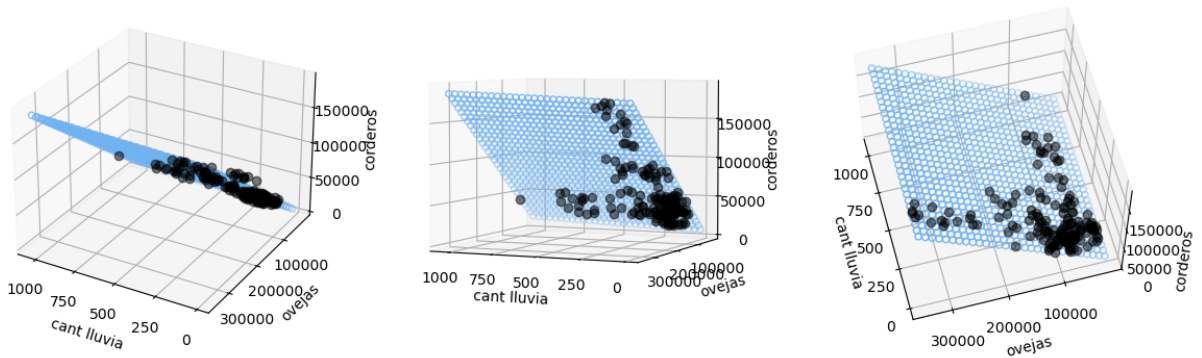
for ax in axes:
    flattenX = x.flatten()
    flattenY = y.flatten()
    flattenZ = z.values.flatten()
    ax.plot(
        flattenX, flattenY, flattenZ, color='k',
        zorder=15, linestyle='none', marker='o', alpha=0.5
    )
    ax.scatter(
        xx_pred.flatten(), yy_pred.flatten(), predicted,
        facecolor=(0,0,0,0), s=20, edgecolor='#70b3f0'
    )
    ax.set_xlabel('cant lluvia', fontsize=10)
    ax.set_ylabel('ovejas', fontsize=10)
    ax.set_zlabel('corderos', fontsize=10)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)
```

```
fig.tight_layout()
```

$$R^2 = 0.98$$



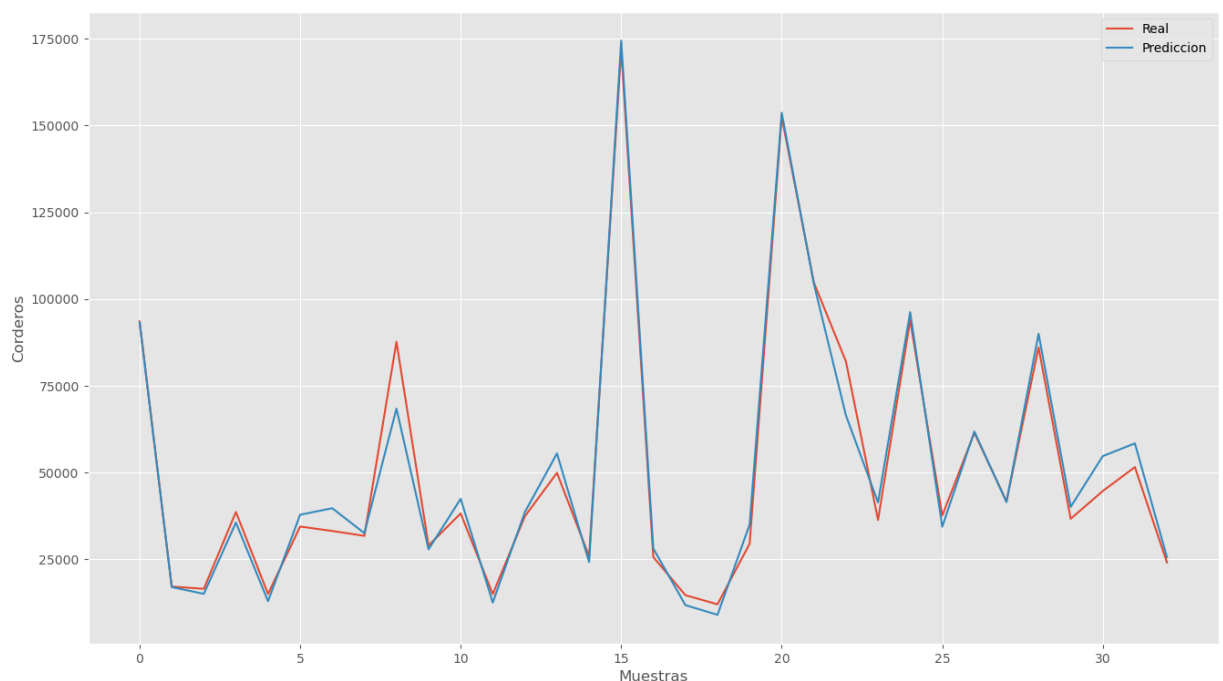
Se observa que los datos no se ajustan perfectamente al plano, lo que va acorde con la información extraída de las métricas.

Gráfico predicción vs real

A continuación, se va a generar un gráfico 2D para comparar los datos reales con la predicción realizada por el modelo.

In [29]:

```
plt.plot(range(len(y_test)),y_test,label="Real")
plt.plot(range(len(pred_model_corderos)),pred_model_corderos,label="")
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('Muestras')
plt.ylabel('Corderos')
plt.legend()
plt.show()
```



Podemos observar que la línea roja que representa los valores reales y la línea azul que representa la predicción del modelo tienen una tendencia igual, casi sin desviaciones. Esto sugiere que el modelo está sobreajustando los datos de entrenamiento y es más probable que tenga un peor desempeño al predecir valores para nuevos datos.

3.12. Comparar y probar modelos de corderos

Al llegar acá contamos con 3 modelos para comparar, por lo tanto la comparación se va a realizar sobre ellos.

Recordemos que:

- El modelo 1 se entreno con el dataset a nivel departamental de Chubut (Dataset 1).
- El modelo 2 se entreno con el dataset a nivel departamental de Chubut mas los datos del Mirabueno, tener en cuenta que contiene al dataset anterior (Dataset 2).
- El modelo 3 se entreno con el dataset actualizado a nivel departamental de Chubut (Dataset 3).

Para hacer la comparación se van a tomar las siguientes reglas:

- Si el score es menor a 0, es decir valores negativos, se le va a restar 1.
- Si el score es menor a 0.60, se le va a restar 0.5.
- Si el score es mayor o igual a 0.60, se le va a sumar 1.
- Se le va a sumar un punto a aquel modelo que funcione mejor.
- Se le va a restar 0.25 a aquel modelo que puede conocer los datos debido a que fue entrenado por ese dataset, recordemos que en este punto cada modelo solo conoce el 80% de su dataset de entrenamiento.
- Opcional: hay dos dataset que se parecen, en cuyo caso se lo va a penalizar con 0.2.

En la siguiente celda de código, se importarán varias librerías, entre ellas:

- Pandas para manejar los datos a través de DataFrames.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados de las predicciones mediante gráficos.
- De Scikit-learn se utilizarán la métricas **r2_score**.
- Por último, se usará `load_pickle` para recuperar el modelo entrenado guardado previamente.

```
In [1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from statsmodels.iolib.smpickle import load_pickle
```

Primero cargaremos los tres modelos para luego probarlos con datos.

```
In [2]: modelo_corderos_1 =
load_pickle('modelos/modelo_prediccion_corderos_1')
modelo_corderos_2 =
load_pickle('modelos/modelo_prediccion_corderos_2')
modelo_corderos_3 =
load_pickle('modelos/modelo_prediccion_corderos_3')
```

Pruebas con dos tuplas

La idea acá es simular que un productor cuenta con dos años de datos respecto a la lluvia y al total de ovejas. Entonces le pasamos ambos datos al modelo, y con ellos hace la predicción, dándonos los corderos, dato que también conocemos y nos sirve para comparar el valor real con el valor predicho.

Prueba 1

Se tomaron dos observaciones del año 2013 y 2014 correspondientes a Biedma del Dataset 1.

ovejas	cant_lluvia	corderos
82998	226.5	26832
82706	331.3	29900

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [3]: resultado_prediccion_1 = modelo_corderos_1.predict([[226.5,82998],  
[331.3,82706]])
```

```
In [4]: resultado_prediccion_2 = modelo_corderos_2.predict([[226.5,82998],  
[331.3,82706]])
```

```
In [5]: resultado_prediccion_3 = modelo_corderos_3.predict([[226.5,82998],  
[331.3,82706]])
```

```
In [6]: dato_real = [[26832],[29900]]
```

```
In [7]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[7]: 'Score o R2 de la predicción 1: -20.98'
```

```
In [8]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[8]: 'Score o R2 de la predicción 2: -8.20'
```

```
In [9]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

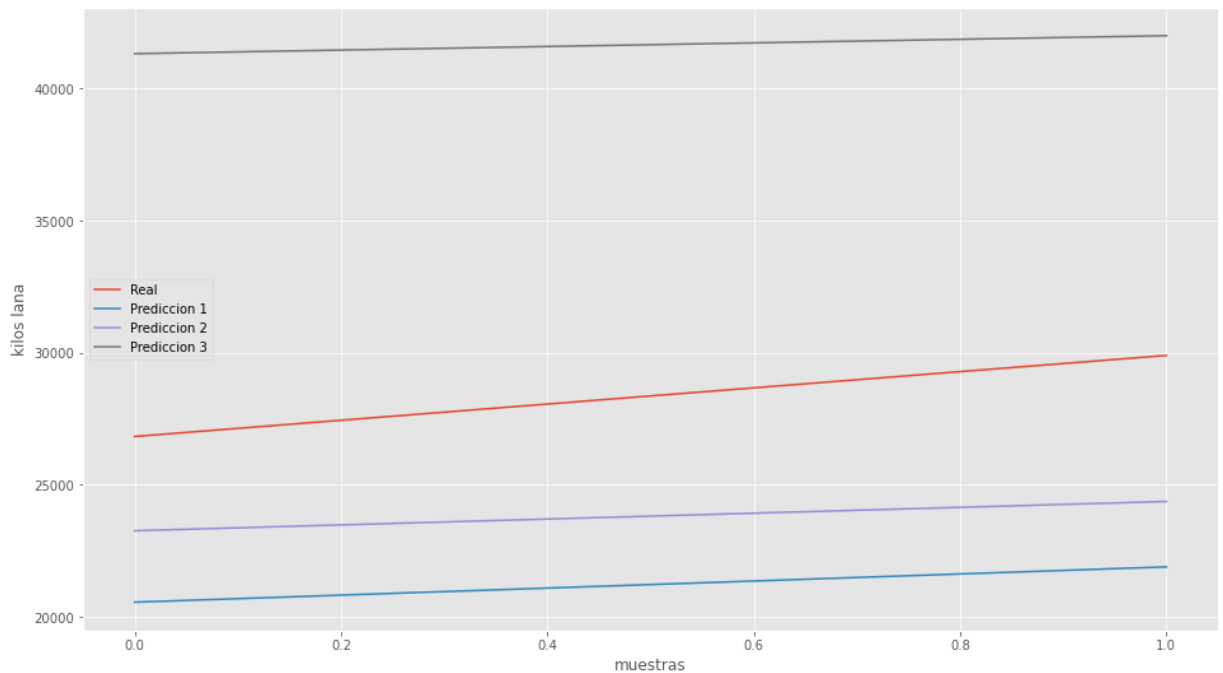
```
Out[9]: 'Score o R2 de la predicción 3: -74.77'
```

Se observa que ningún modelo logrará predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [10]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')
```

```
plt.plot(range(len(dato_real)),dato_real,label="Real")
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,label="Prediccion 1")
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,label="Prediccion 2")
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,label="Prediccion 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelos no logran predecir estos datos dado que las tres líneas de predicción no se ajustan a la real y están muy alejadas (de ahí los valores negativos).

Entonces el "mejor modelo" resulta ser el 2, seguido por el 1 y el que peor predice es el 3.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-1.25	-1.25
2	1	-1.20	-0.20
3	0	-1.00	-1.00

Prueba 2

Se tomaron dos observaciones correspondientes a Biedma del 2014 y a Florentino Ameghino del 2013 del Dataset 3.

ovejas	cant_lluvia	corderos
61195	331.3	34926
150199	135.5	95320

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [11]: resultado_prediccion_1 = modelo_corderos_1.predict([[331.3,61195],
[135.5,150199]])
```

```
In [12]: resultado_prediccion_2 = modelo_corderos_2.predict([[331.3,61195],
[135.5,150199]])
```

```
In [13]: resultado_prediccion_3 = modelo_corderos_3.predict([[331.3,61195],
[135.5,150199]])
```

```
In [14]: dato_real =[[34926],[95320]]
```

```
In [15]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_1))
```

```
Out[15]: 'Score o R2 de la predicción 1: -0.63'
```

```
In [16]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_2))
```

```
Out[16]: 'Score o R2 de la predicción 2: -0.54'
```

```
In [17]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,
resultado_prediccion_3))
```

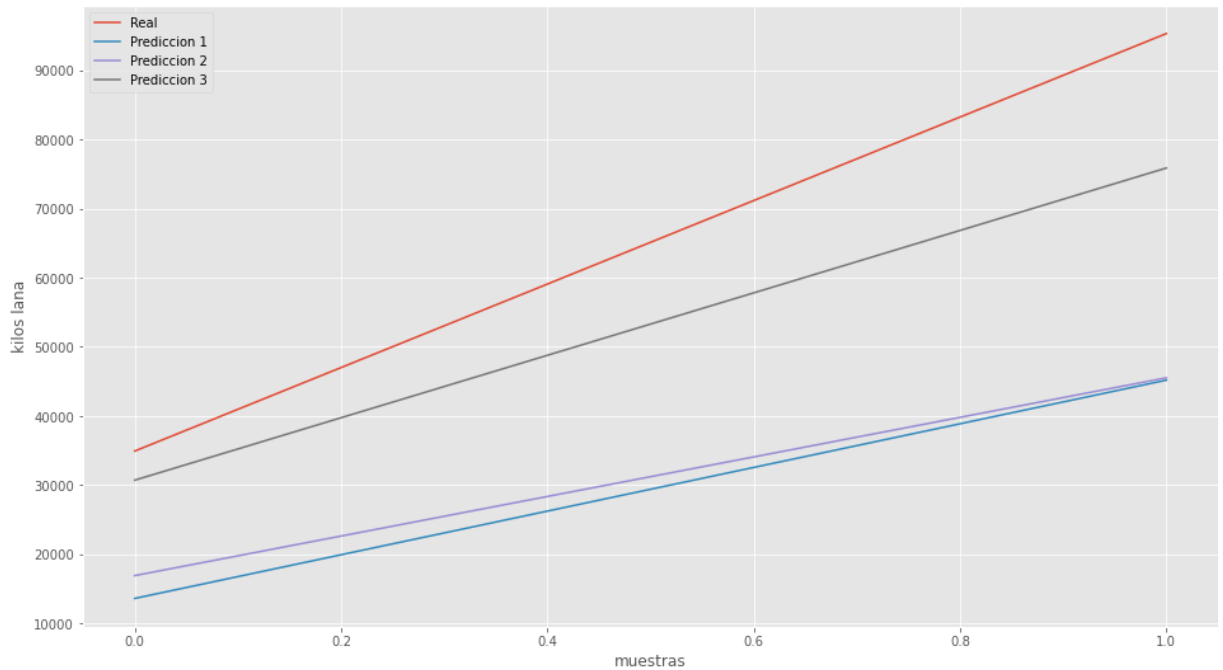
```
Out[17]: 'Score o R2 de la predicción 3: 0.78'
```

Se observa que solo un modelo logrará predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [18]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)),dato_real,label="Real")
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,
label="1")
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,
label="2")
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,
label="3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
```

```
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, un modelo está mas cerca de la línea real y los otros modelos no logran predecir estos datos.

Entonces el mejor modelo resulta ser el 3, seguido por el 2 y el que peor predice es el 1.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-1.00	-2.25
2	0	-1.00	-1.20
3	1	-0.25	-0.25

Prueba 3

Se tomaron dos obsevaciones correspondientes a Biedma del 2012 y a Florentino Ameghino de 2009 del Dataset 1.

ovejas	cant_lluvia	corderos
93378	46.40	16071
150454	137.7	51674

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [19]: resultado_prediccion_1 = modelo_corderos_1.predict([[46.40,93378],
[137.7,150454]])
```

```
In [20]: resultado_prediccion_2 = modelo_corderos_2.predict([[46.40,93378],
[137.7,150454]])
```

```
In [21]: resultado_prediccion_3 = modelo_corderos_3.predict([[46.40,93378],  
[137.7,150454]])
```

```
In [22]: dato_real =[[16071],[51674]]
```

```
In [23]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_1))
```

```
Out[23]: 'Score o R2 de la predicción 1: 0.88'
```

```
In [24]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_2))
```

```
Out[24]: 'Score o R2 de la predicción 2: 0.82'
```

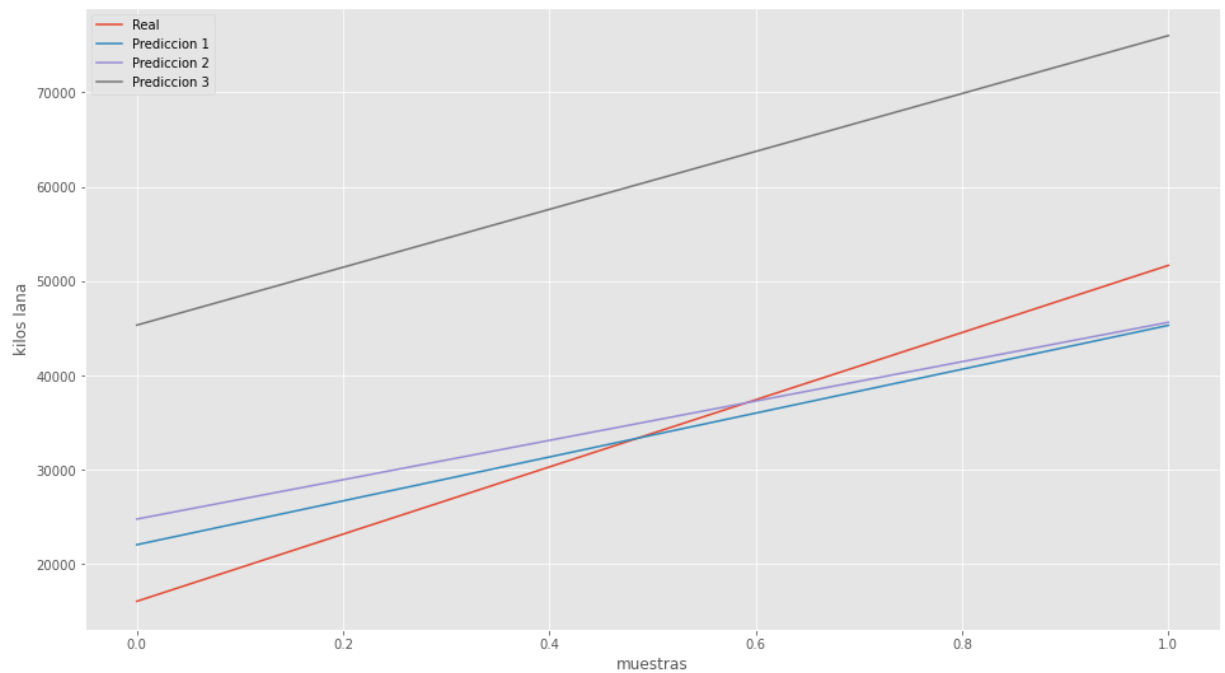
```
In [25]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,  
resultado_prediccion_3))
```

```
Out[25]: 'Score o R2 de la predicción 3: -1.29'
```

Se observa que dos modelos logran predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [26]: plt.rcParams['figure.figsize'] = (16, 9)  
plt.style.use('ggplot')  
plt.plot(range(len(dato_real)),dato_real,label="Real")  
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,label="1")  
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,label="2")  
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,label="3")  
plt.xlabel('muestras')  
plt.ylabel('kilos lana')  
plt.legend()  
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, dos modelos logran predecir estos datos, pero el modelo 3 está muy alejado.

El mejor modelo resulta ser el 1, pero como fue entrenado con estos datos se lo va a penalizar, seguido por el 2 y por último el 3 .

Modelo	Puntaje positivo	Puntaje negativo	Total
1	2	-0.25	-0.50
2	1	-0.20	-0.40
3	0	-1	-1.25

Prueba 4

Se tomaron dos obsevaciones correspondientes a Rawson del 2014 y a Río Senguer de 2013 del Dataset 1.

ovejas	cant_lluvia	corderos
61699	187.2	18436
384024	105.77	143717

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [27]: resultado_prediccion_1 = modelo_corderos_1.predict([[187.2,61699],
[105.77,384024]])
```

```
In [28]: resultado_prediccion_2 = modelo_corderos_2.predict([[187.2,61699],
[105.77,384024]])
```

```
In [29]: resultado_prediccion_3 = modelo_corderos_3.predict([[187.2,61699],
[105.77,384024]])
```



```
In [30]: dato_real = [[18436],[143717]]
```

```
In [31]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
    resultado_prediccion_1))
```

```
Out[31]: 'Score o R2 de la predicción 1: 0.98'
```

```
In [32]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
    resultado_prediccion_2))
```

```
Out[32]: 'Score o R2 de la predicción 2: 0.96'
```

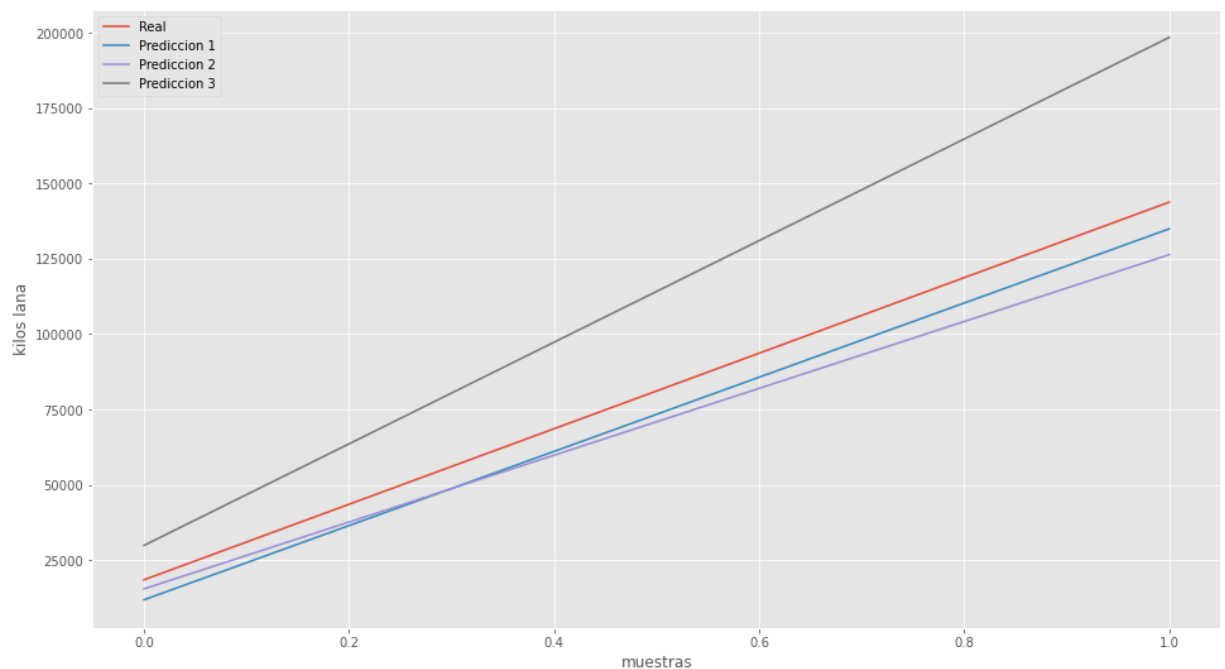
```
In [33]: 'Score o R2 de la predicción 3: {:.2f}'.format(r2_score(dato_real,
    resultado_prediccion_3))
```

```
Out[33]: 'Score o R2 de la predicción 3: 0.60'
```

Se observa que todos los modelos logran predecir correctamente estos datos según el R2, al tres le costo un poco.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [34]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)),dato_real,label="Real")
plt.plot(range(len(resultado_prediccion_1)),resultado_prediccion_1,
1")
plt.plot(range(len(resultado_prediccion_2)),resultado_prediccion_2,
2")
plt.plot(range(len(resultado_prediccion_3)),resultado_prediccion_3,
3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelo logran predecir estos datos pero están un poco alejados.

El mejor modelo resulta ser el 1, seguido por el 2 y el puntaje más bajo lo tiene el 3.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	2	-0.25	1.25
2	1	-0.20	0.40
3	1	0	-0.25

Prueba 5

Se tomaron dos obsevaciones correspondientes a Gaiman del 2014 y a Futalefú de 2014 del Dataset 3.

ovejas	cant_lluvia	corderos
90039	137.93	50701
51384	614.79	28919

```
In [35]: resultado_prediccion_1 =
          modelo_corderos_1.predict([[614.79,51384], [137.93,90039]])
```

```
In [36]: resultado_prediccion_2 =
          modelo_corderos_2.predict([[614.79,51384], [137.93,90039]])
```

```
In [37]: resultado_prediccion_3 =
          modelo_corderos_3.predict([[614.79,51384], [137.93,90039]])
```

```
In [38]: dato_real = [[28919],[50701]]
```

Vamos a realizar una predicción para cada modelo y ver cual es su score con estos datos.

```
In [39]: 'Score o R2 de la predicción 1: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_1))
```

```
Out[39]: 'Score o R2 de la predicción 1: -3.43'
```

```
In [40]: 'Score o R2 de la predicción 2: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_2))
```

```
Out[40]: 'Score o R2 de la predicción 2: -2.47'
```

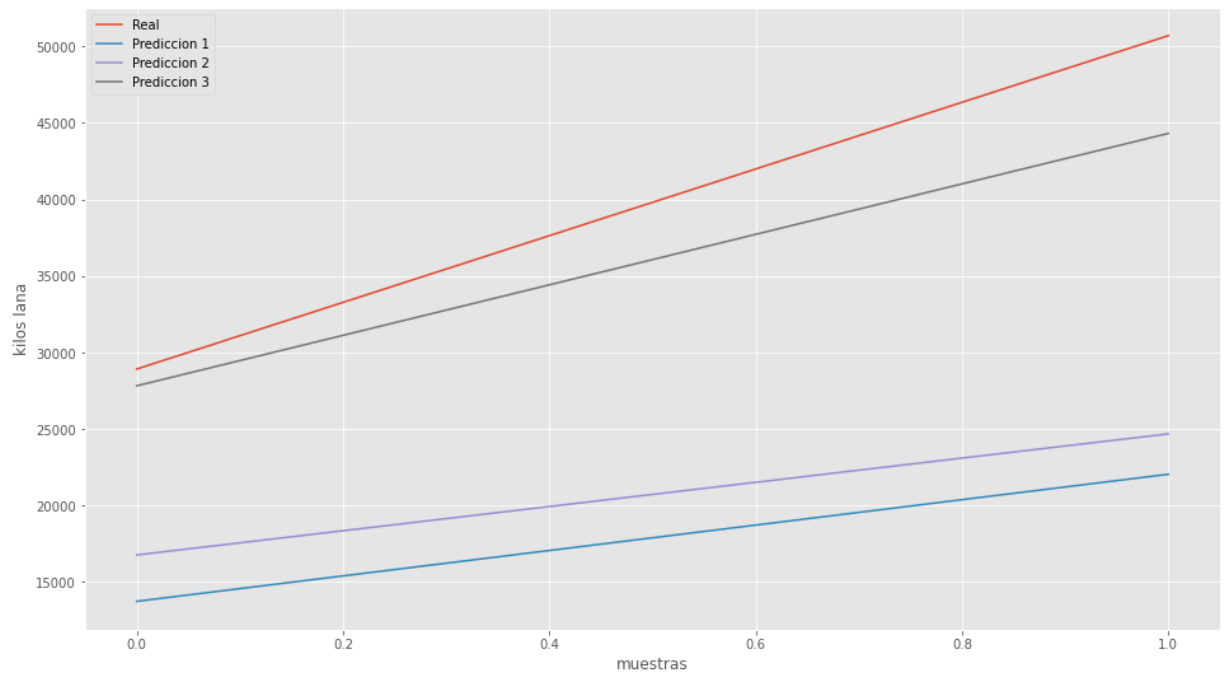
```
In [41]: 'Score o R2 de la predicción: {:.2f}'.format(r2_score(dato_real,
               resultado_prediccion_3))
```

```
Out[41]: 'Score o R2 de la predicción: 0.82'
```

Se observa que solo un modelo logra predecir correctamente estos datos según el R2.

Vamos a graficar estos resultados para ver que tan alejadas están las líneas visualmente.

```
In [42]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.plot(range(len(dato_real)), dato_real, label="Real")
plt.plot(range(len(resultado_prediccion_1)), resultado_prediccion_1, label="1")
plt.plot(range(len(resultado_prediccion_2)), resultado_prediccion_2, label="2")
plt.plot(range(len(resultado_prediccion_3)), resultado_prediccion_3, label="3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



Los gráficos nos muestran lo que nos indicaban las métricas, los modelos que no logran predecir estos datos estan más lejos de la línea real.

El mejor modelo resulta ser el 3, seguido por el 2 y por último el 1.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-1.00	0.25
2	0	-1.00	-0.60
3	2	-0.25	1.50

Pruebas con una tupla

La idea acá es simular que un productor cuenta con los datos de la lluvia y el total de ovejas, del año actual respectivo. Entonces le pasamos ambos datos al modelo, y con ellos hace la predicción para ese año, dándonos los corderos, dato que también conocemos y nos sirve para comparar el valor real con el valor predicho.

Prueba 1

Se tomo una observación del dataset correspondiente a Cushamen del 2014 del Dataset 1.

ovejas	cant_lluvia	corderos
120803	640.4	40184

```
In [43]: dato_real = 40184
```

```
In [44]: resultado_prediccion_1 =
modelo_corderos_1.predict([[640.4, 120803]])
round(resultado_prediccion_1[0][0] - dato_real)
```

```
Out[44]: 660
```

```
In [45]: resultado_prediccion_2 =
modelo_corderos_2.predict([[640.4,120803]])
round(resultado_prediccion_2[0][0] - dato_real)
```

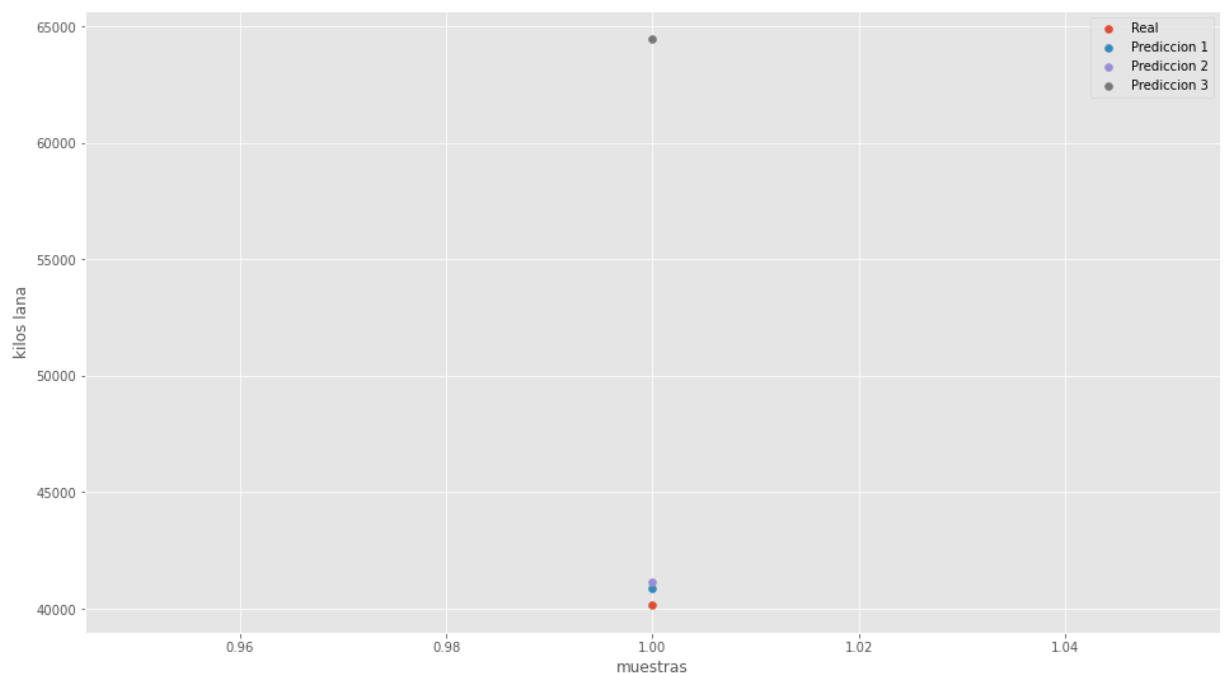
Out[45]: 967

```
In [46]: resultado_prediccion_3 =
modelo_corderos_3.predict([[640.4,120803]])
round(resultado_prediccion_3[0][0] - dato_real)
```

Out[46]: 24280

En estos tres resultados, vemos que los modelos predijeron corderos demás, es decir el modelo 1 se paso por 660 corderos, el modelo 2 se paso por 967 corderos y el modelo 3 se paso por 24280 corderos.

```
In [47]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.scatter(y=dato_real, x=1, label="Real")
plt.scatter(y=resultado_prediccion_1,x=1, label="Prediccion 1")
plt.scatter(y=resultado_prediccion_2,x=1, label="Prediccion 2")
plt.scatter(y=resultado_prediccion_3,x=1, label="Prediccion 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



El mejor modelo resulta ser el 1, seguido por el 2 y por último el 3. Cómo acá no se puede calcular el R2, por estar trabajando con un solo dato, solo se va a tener en cuenta cuál es el modelo que esta mas cerca del dato real.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	1	-0.25	1.00
2	0	-0.20	-0.80
3	0	0	1.50

Prueba 2

Se tomo una observación del dataset correspondiente a Sarmiento del 2014 del Dataset 3.

ovejas	cant_lluvia	corderos
80123	69.45	37346

```
In [48]: dato_real = 37346
```

```
In [49]: resultado_prediccion_1 =
modelo_corderos_1.predict([[69.45,80123]])
round(resultado_prediccion_1[0][0] - dato_real)
```

```
Out[49]: -20067
```

```
In [50]: resultado_prediccion_2 =
modelo_corderos_2.predict([[69.45,80123]])
round(resultado_prediccion_2[0][0] - dato_real)
```

```
Out[50]: -16891
```

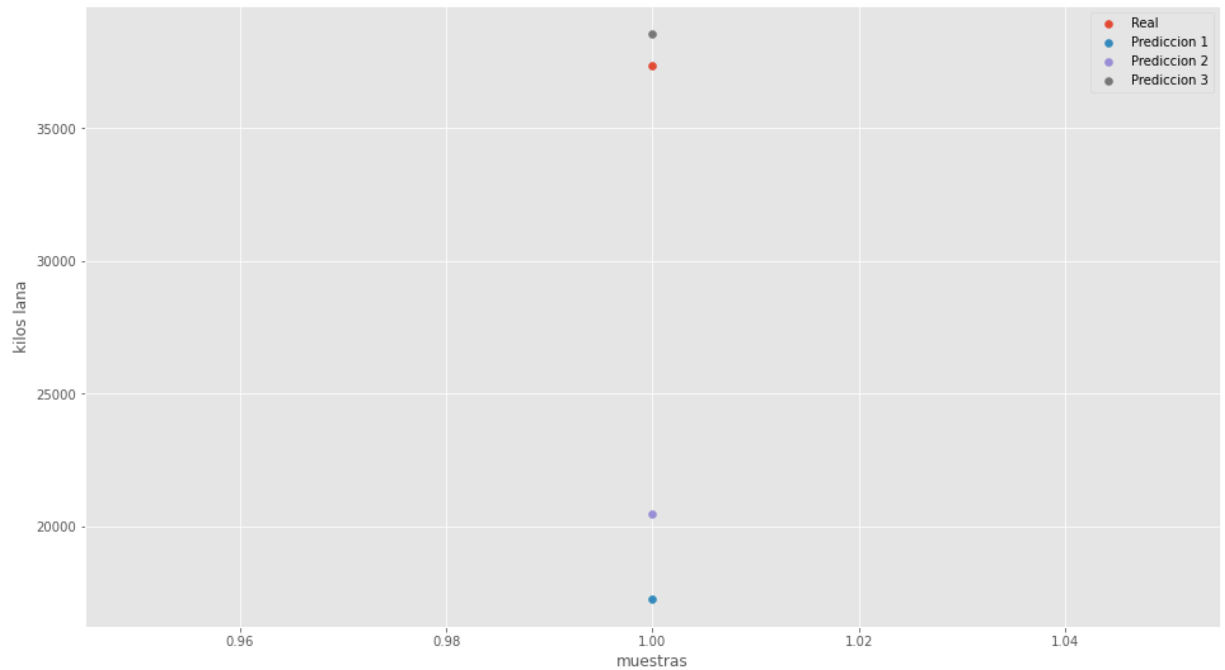
```
In [51]: resultado_prediccion_3 =
modelo_corderos_3.predict([[69.45,80123]])
round(resultado_prediccion_3[0][0] - dato_real)
```

```
Out[51]: 1219
```

En estos tres resultados, vemos que los modelos predijeron por un lado corderos de menos, como es el caso del modelo 1 que le faltó 20067 corderos y el modelo 2 que le faltó 16891 corderos, en cambio el modelo 3 se pasó por 1219 corderos.

```
In [52]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.scatter(y=dato_real, x=1, label="Real")
plt.scatter(y=resultado_prediccion_1, x=1, label="Prediccion 1")
plt.scatter(y=resultado_prediccion_2, x=1, label="Prediccion 2")
plt.scatter(y=resultado_prediccion_3, x=1, label="Prediccion 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
```

```
plt.legend()
plt.show()
```



El mejor modelo resulta ser el 3, seguido por el 2 y por último el 1. Cómo acá no se puede calcular el R2, por estar trabajando con un solo dato, solo se va a tener en cuenta cuál es el modelo que esta mas cerca del dato real.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	0	1.00
2	0	0	-0.80
3	1	-0.25	2.25

Prueba 3

Se tomo una observación del dataset correspondiente a Mirabueno del 2018 del Dataset 2.

ovejas	cant_lluvia	corderos
1200	93.2	820

```
In [53]: dato_real = 820
```

```
In [54]: resultado_prediccion_1 = modelo_corderos_1.predict([[93.2,1200]])
round(resultado_prediccion_1[0][0] - dato_real)
```

```
Out[54]: -13617
```

```
In [55]: resultado_prediccion_2 = modelo_corderos_2.predict([[93.2,1200]])
round(resultado_prediccion_2[0][0] - dato_real)
```

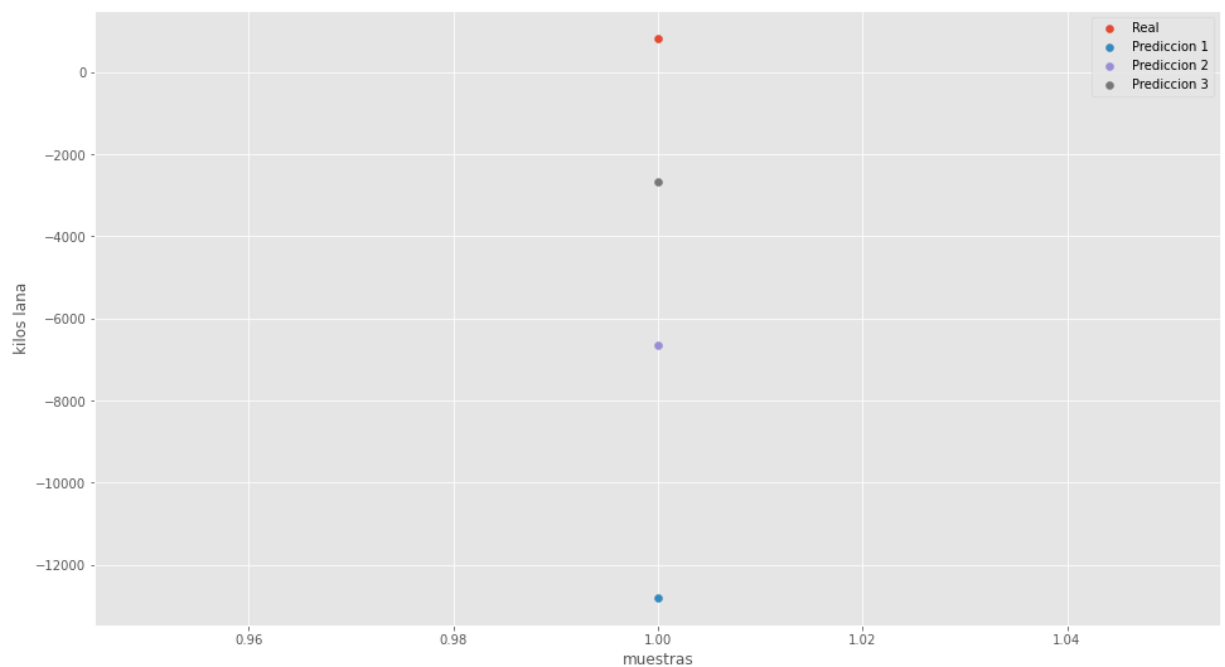
```
Out[55]: -7468
```

```
In [56]: resultado_prediccion_3 = modelo_corderos_3.predict([[93.2,1200]])
round(resultado_prediccion_3[0][0] - dato_real)
```

Out[56]: -3491

En estos tres resultados, vemos que los modelos predijeron corderos de menos, es decir al modelo 1 le faltó 13617, al modelo 2 le faltó 7468 corderos y al modelo 3 le faltó 3491 corderos.

```
In [57]: plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.scatter(y=dato_real, x=1, label="Real")
plt.scatter(y=resultado_prediccion_1, x=1, label="Prediccion 1")
plt.scatter(y=resultado_prediccion_2, x=1, label="Prediccion 2")
plt.scatter(y=resultado_prediccion_3, x=1, label="Prediccion 3")
plt.xlabel('muestras')
plt.ylabel('kilos lana')
plt.legend()
plt.show()
```



El mejor modelo resulta ser el 3, seguido por el 2 y por último el 1. Cómo acá no se puede calcular el R2, por estar trabajando con un solo dato, solo se va a tener en cuenta cuál es el modelo que esta mas cerca del dato real.

Modelo	Puntaje positivo	Puntaje negativo	Total
1	0	-0.20	0.80
2	0	-0.25	-1.05
3	1	0	3.25

Resultados

El mejor **modelo es el 3** con un puntaje final de 3.25, seguido del modelo 1 con 0.8 y por último el modelo 2 con -1.05 puntos.

3.13. Modelo de corderos seleccionado

Después de realizar una comparación, el modelo que mejor funciono fue el 3, entrenado con los datos de departamentos de Chubut actualizados con los datos de las encuestas; que si bien en un principio parecía tener overfit debido a las altas métricas fue capaz de generalizar bastante bien para datos que no conocía y fue el que obtuvo mayor puntaje. Por lo tanto es el que va ser utilizado en la aplicación.

En la siguiente celda de código, se importarán varias librerías, entre ellas:

- Pandas para manejar los datos a través de DataFrames.
- Matplotlib y Seaborn se utilizarán para visualizar los resultados de las predicciones mediante gráficos.
- De Scikit-learn se utilizarán la métricas **r2_score**.
- Por último, se usará **load_pickle** para recuperar el modelo entrenado guardado previamente.

```
In [42]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from statsmodels.iolib.smpickle import load_pickle
```

Cargamos el modelo para mostrar como se vería en la aplicación.

```
In [43]: modelo_corderos =
load_pickle('modelos/modelo_prediccion_corderos_3')
```

Se creo un modelo con datos aleatorios del dataset 1 y 2, para ver con un conjunto de datos cuál sería el resultado.

```
In [44]: data = pd.read_csv("datos/datos_reducidos_corderos.csv")
data
```

```
Out[44]:
```

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_l
0	2012-12-31	CHUBUT	BIEDMA	2740	52320	22319	25489	29517	132385	47811
1	2013-12-31	CHUBUT	BIEDMA	3925	70677	27227	33389	36961	172179	54494
2	2019-12-31	CHUBUT	BIEDMA	3606	70938	32085	33311	38587	178527	61591
3	2020-12-31	CHUBUT	BIEDMA	3084	82806	29134	33727	25394	174145	63045
4	2008-12-31	CHUBUT	CUSHAMEN	4578	86402	28200	19288	44487	182955	53239
5	2013-12-31	CHUBUT	CUSHAMEN	4751	85946	26255	17619	41451	176022	47999
6	2014-12-31	CHUBUT	CUSHAMEN	4198	77263	26512	18975	46290	173238	57818

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_l
7	2015-12-31	CHUBUT	CUSHAMEN	5377	99927	33404	23023	52959	214690	66339
8	2020-12-31	CHUBUT	CUSHAMEN	5385	124862	33570	28086	52545	244448	70875
9	2009-12-31	CHUBUT	FLORENTINO AMEGHINO	7970	131131	56083	47690	83896	326770	118884
10	2014-12-31	CHUBUT	FLORENTINO AMEGHINO	8893	140928	56282	51459	89550	347112	94822
11	2016-12-31	CHUBUT	FLORENTINO AMEGHINO	7745	122098	55202	45482	78682	309209	90289
12	2017-12-31	CHUBUT	FLORENTINO AMEGHINO	8064	131186	54791	48486	85190	327717	100609
13	2007-12-31	CHUBUT	FUTALEUFU	5129	81560	24740	17626	21341	150396	78720
14	2008-12-31	CHUBUT	FUTALEUFU	5639	90494	30119	24634	28237	179123	70848
15	2009-12-31	CHUBUT	FUTALEUFU	5155	85752	28633	24140	26350	170030	56153
16	2019-12-31	CHUBUT	FUTALEUFU	3775	56479	14981	14614	27926	117775	36274
17	2014-12-31	CHUBUT	GAIMAN	4936	90039	27772	26376	50701	199824	63732
18	2020-12-31	CHUBUT	GAIMAN	5295	197372	27275	46137	42834	318913	78165
19	2012-12-31	CHUBUT	LANGUINÑO	8550	141153	52659	54014	45432	301808	98829
20	2016-12-31	CHUBUT	LANGUINÑO	7588	112547	47178	49593	17445	234351	99540
21	2017-12-31	CHUBUT	PASO DE INDIOS	6908	117620	39207	47768	53235	264738	96894
22	2018-12-31	CHUBUT	PASO DE INDIOS	6906	125103	38540	47518	55938	274005	94805
23	2000-12-31	CHUBUT	RAWSON	2355	57320	20632	24405	18609	123321	25840
24	2004-12-31	CHUBUT	RAWSON	2769	43180	14374	16714	21962	98999	34154
25	2018-12-31	CHUBUT	RAWSON	3558	68936	20065	19143	21294	132996	21280
26	2019-12-31	CHUBUT	RAWSON	2198	33686	11357	13155	16450	76846	25359
27	2020-12-31	CHUBUT	RAWSON	2273	74096	18058	19267	22102	135796	20520
28	2009-12-31	CHUBUT	RIO SENGUERR	16103	273190	104009	92579	139943	625824	236008
29	2010-12-31	CHUBUT	RIO SENGUERR	14701	246297	97591	82269	126654	567512	218805
30	2011-12-31	CHUBUT	RIO SENGUERR	14388	241283	93912	80243	122691	552517	214650

	fecha	provincia	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_l
31	2013-12-31	CHUBUT	SARMIENTO	5978	85638	26288	30644	32877	181425	66337
32	2014-12-31	CHUBUT	SARMIENTO	5269	80123	27608	32012	37346	182358	69447
33	2017-12-31	CHUBUT	SARMIENTO	5633	78931	26286	31383	33832	176065	63559
34	2008-12-31	CHUBUT	TEHUELCHES	14960	221763	107149	126501	109720	580093	304560
35	2009-12-31	CHUBUT	TEHUELCHES	17055	241352	99948	130175	100704	589234	241392
36	2015-12-31	CHUBUT	TELSEN	4477	76246	29205	29821	35267	175016	63880
37	2016-12-31	CHUBUT	TELSEN	4896	80262	26471	39243	20290	171162	63700
38	2017-12-31	CHUBUT	TELSEN	4640	77556	27296	36570	17750	163812	63700

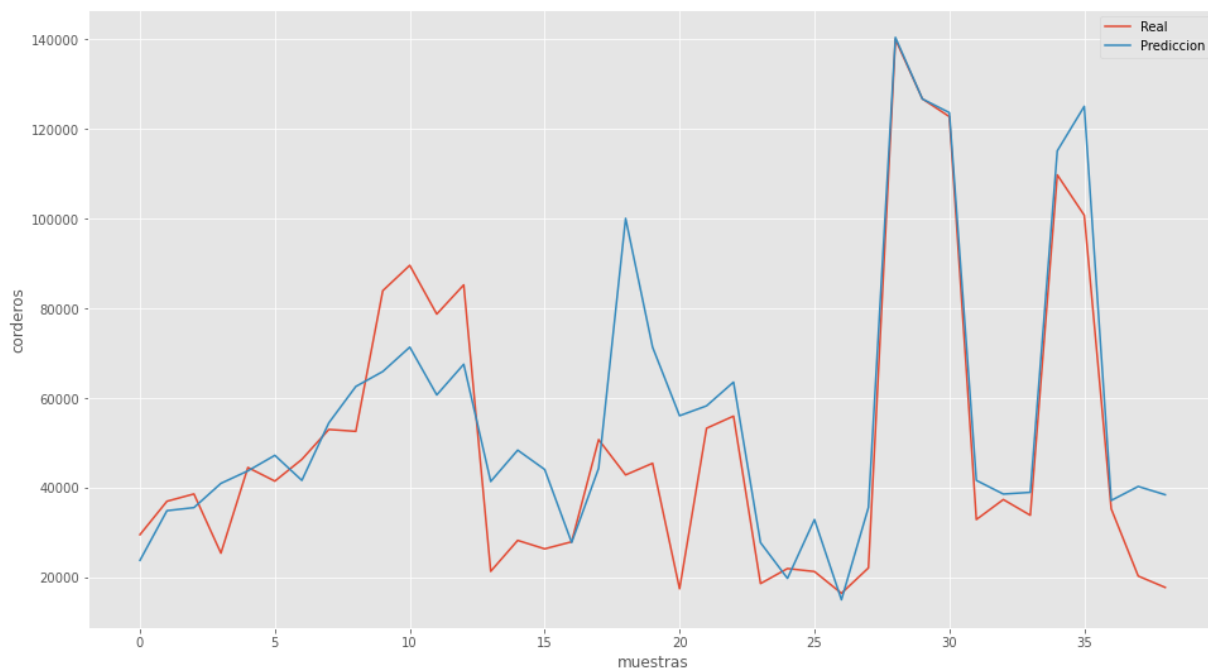
```
In [45]: X = data[['cant_lluvia', 'ovejas']]
y = data[['cordero/as']]
```

```
In [46]: resultado_prediccion = modelo_corderos.predict(X)
```

```
In [47]: 'Score o R2 de la predicción: {:.2%}'.format(r2_score(y,
resultado_prediccion))
```

```
Out[47]: 'Score o R2 de la predicción: 75.60%'
```

```
In [48]: plt.plot(range(len(y)),y,label="Real")
plt.plot(range(len(resultado_prediccion)),resultado_prediccion,label="Prediccion")
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('muestras')
plt.ylabel('corderos')
plt.legend()
plt.show()
```



Vemos que el modelo predice bien con un 92% de exactitud.

3.14. Modelo para predecir la finura

La siguiente celda de código utiliza algunas de las bibliotecas más populares para el tratamiento y análisis de datos.

- `import numpy as np` : se importa la biblioteca NumPy y se le asigna un alias `np` , esto se hace para que sea más fácil de usar y escribir en el código. NumPy es una biblioteca que proporciona soporte para cálculos matemáticos y de matrices en Python.
- `import pandas as pd` : se importa la biblioteca Pandas y se le asigna un alias `pd` . Pandas es una biblioteca que proporciona estructuras de datos flexibles para el análisis de datos y herramientas para trabajar con datos estructurados.
- `import matplotlib.pyplot as plt` : se importa la biblioteca Matplotlib , que se utiliza para crear visualizaciones en Python, y se le asigna un alias `plt` .
- `%matplotlib inline` : se habilita la visualización en línea de gráficos Matplotlib , lo que significa que los gráficos se mostrarán directamente en el cuaderno de Jupyter en el que se está trabajando.
- `from sklearn.ensemble import RandomForestRegressor` : se importa la biblioteca RandomForestRegressor que se utiliza para crear modelos de regresión aleatorios que utilizan árboles de decisión. Estos modelos se utilizan a menudo para predecir valores numéricos a partir de características del conjunto de datos.
- `from sklearn.metrics import mean_squared_error, mean_absolute_error` : se importan algunas métricas de evaluación de modelos, incluyendo `mean_squared_error` y `mean_absolute_error` . Estas métricas se utilizan para evaluar el rendimiento del modelo en términos de precisión y ajuste.
- `from skforecast.ForecasterAutoreg import ForecasterAutoreg` : se importa la biblioteca ForecasterAutoreg que se especializa en modelos de predicción de series de tiempo en Python.
- `from statsmodels.iolib.smpickle import save_pickle` : se importa `save_pickle` que se utiliza para guardar modelos de predicción en archivos para su posterior uso.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import YearLocator
%matplotlib inline

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,
mean_absolute_error

from skforecast.ForecasterAutoreg import ForecasterAutoreg
```

```
from statsmodels.iolib.smpickle import save_pickle
```

Preparación de los datos

A continuación, en la siguiente celda de código, se presenta un conjunto de datos que fue creado manualmente utilizando información del Mirabueno. El conjunto de datos abarca un período de 36 años y se enfoca en la medición de la finura de la lana y su correspondiente fecha.

```
In [2]: datos = pd.read_csv('datos/lana_mirabueno_finura.csv', sep=',')
datos
```

```
Out[2]:
```

	fecha	finura
--	-------	--------

0	1985-12-31	19.26
1	1986-12-31	19.86
2	1987-12-31	19.41
3	1988-12-31	18.65
4	1989-12-31	19.82
5	1990-12-31	20.23
6	1991-12-31	20.47
7	1992-12-31	19.80
8	1993-12-31	19.30
9	1994-12-31	18.87
10	1995-12-31	18.61
11	1996-12-31	18.51
12	1997-12-31	19.66
13	1998-12-31	19.09
14	1999-12-31	19.54
15	2000-12-31	20.80
16	2001-12-31	21.00
17	2002-12-31	20.93
18	2003-12-31	19.60
19	2004-12-31	19.20
20	2005-12-31	18.50
21	2006-12-31	19.30
22	2007-12-31	19.30
23	2008-12-31	20.36
24	2009-12-31	19.60
25	2010-12-31	18.78
26	2011-12-31	18.50
27	2012-12-31	20.20

	fecha	finura
28	2013-12-31	20.30
29	2014-12-31	19.40
30	2015-12-31	19.30
31	2016-12-31	19.80
32	2017-12-31	19.50
33	2018-12-31	19.80
34	2019-12-31	19.80
35	2020-12-31	19.60

En resumen, el siguiente código convierte la columna `fecha` en el índice, elimina la columna original, renombra la columna `finura` como `y`, convierte los datos a una frecuencia anual y finalmente ordena los datos por el índice. El resultado final es un DataFrame limpio y organizado que puede ser utilizado para análisis de series de tiempo. La función `datos.head()` muestra las primeras cinco filas del DataFrame resultante.

```
In [3]: datos = datos.set_index(pd.to_datetime(datos['fecha'],
format='%Y/%m/%d')).drop(columns=['fecha']).rename(columns=
{'finura': 'y'}).asfreq('A').sort_index()
datos.head()
```

```
Out[3]:
```

	y
fecha	
1985-12-31	19.26
1986-12-31	19.86
1987-12-31	19.41
1988-12-31	18.65
1989-12-31	19.82

Las siguientes celdas utilizan las funciones `max` y `min` de Pandas para obtener el rango de valores de la `finura`.

```
In [4]: datos['y'].max()
```

```
Out[4]: 21.0
```

```
In [5]: datos['y'].min()
```

```
Out[5]: 18.5
```

El siguiente código verifica si el índice temporal del DataFrame (`datos`) está completo, es decir, si no faltan fechas entre la fecha mínima y la fecha máxima en el índice.

Para hacerlo, se crea un rango de fechas (`pd.date_range()`) que va desde la fecha mínima hasta la fecha máxima del índice, con la misma frecuencia que tiene el índice

(datos.index.freq). Luego, se compara si todas las fechas del índice (datos.index) están presentes en este rango de fechas creado. Si esto es verdadero para todas las fechas, entonces se puede decir que el índice temporal está completo.

```
In [6]: (datos.index == pd.date_range(
        start = datos.index.min(),
        end    = datos.index.max(),
        freq   = datos.index.freq)
        ).all()
```

Out[6]: True

El siguiente código divide los datos en conjuntos de entrenamiento y prueba para su posterior uso en el modelado y la evaluación. En particular, se utiliza el parámetro `steps` para indicar la cantidad de puntos de datos que se deben reservar para el conjunto de prueba. Luego, se crea el conjunto de entrenamiento `datos_train` seleccionando todos los puntos de datos menos los últimos `steps` y el conjunto de prueba `datos_test` seleccionando los últimos `steps` puntos de datos.

```
In [7]: steps = 3
        datos_train = datos[:-steps]
        datos_test  = datos[-steps:]
```

La siguiente celda de código muestra un gráfico con los datos de entrenamiento y prueba. El eje x muestra las fechas en formato de años, con una separación de un año entre cada marca de tiempo. El eje y muestra la finura de la lana.

`YearLocator()` es una función de la librería `matplotlib.dates` que determina las posiciones de las marcas de tiempo en el eje x. En este caso, se utiliza para establecer las marcas de tiempo en años.

`plt.style.use('ggplot')` se utiliza para aplicar el estilo 'ggplot' a la gráfica.

`ax.plot()` se utiliza para trazar los datos de entrenamiento en la gráfica.

`ax.scatter()` se utiliza para trazar los datos de prueba en la gráfica.

`ax.set_xlabel()` y `ax.set_ylabel()` se utilizan para establecer las etiquetas de los ejes x e y, respectivamente.

`ax.legend()` se utiliza para mostrar la leyenda de la gráfica.

`ax.xaxis.set_major_locator()` se utiliza para establecer el separación de las marcas de tiempo en el eje x.

`plt.xticks()` se utiliza para establecer la rotación y alineación de las marcas de tiempo en el eje x.

`ax.xaxis.set_major_formatter()` se utiliza para establecer el formato de las marcas de tiempo en el eje x.

Finalmente, se imprimen las fechas mínima y máxima de los conjuntos de entrenamiento y prueba junto con su longitud, utilizando `print()`.

```
In [8]: years = YearLocator()
plt.style.use('ggplot')

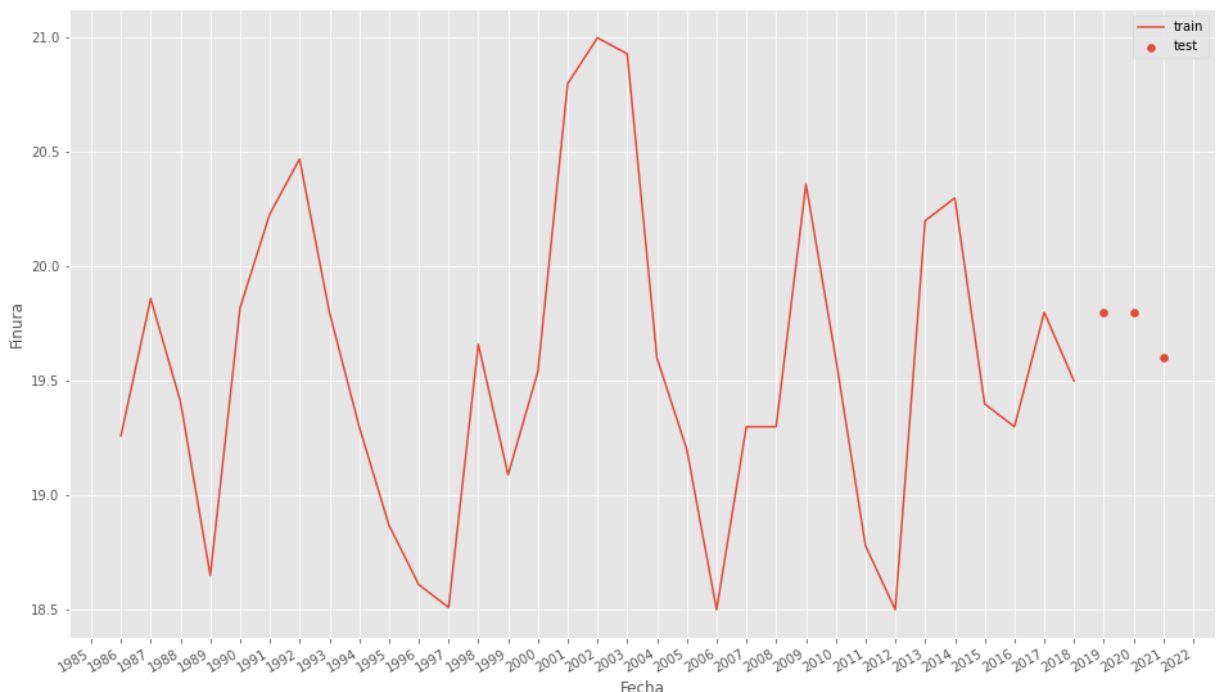
fig, ax = plt.subplots(figsize=(16, 9))
ax.plot(datos_train.index, datos_train['y'], label='train')
ax.scatter(datos_test.index, datos_test['y'], label='test')
ax.set_xlabel('Fecha')
ax.set_ylabel('Finura')
ax.legend()

ax.xaxis.set_major_locator(years)
plt.xticks(rotation=30, ha='right')
ax.xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))

print(f"Fechas train : {datos_train.index.min()} ---
{datos_train.index.max()} (n={len(datos_train)})")
print(f"Fechas test  : {datos_test.index.min()} ---
{datos_test.index.max()} (n={len(datos_test)})")
```

Fechas train : 1985-12-31 00:00:00 --- 2017-12-31 00:00:00 (n=33)

Fechas test : 2018-12-31 00:00:00 --- 2020-12-31 00:00:00 (n=3)



Entrenar al modelo

Lo siguiente que hacemos es crear y entrenar un modelo de forecasting utilizando un `RandomForestRegressor` y la clase `ForecasterAutoreg` de la librería `skforecast`. Al establecer `random_state=123` en el modelo de `RandomForestRegressor`, se fija la semilla en 123 para la generación de los números aleatorios en el modelo, lo que permite que los resultados del modelo sean reproducibles. El modelo utiliza 30 lags (retrasos) como variables predictoras y se entrena con los datos de entrenamiento `datos_train`. Finalmente, se devuelve el modelo entrenado `forecaster`.

```
In [9]: forecaster = ForecasterAutoreg(
        regressor =
        RandomForestRegressor(random_state=123),
        lags = 30
    )

    forecaster.fit(y=datos_train['y'])
    forecaster
```

```
Out[9]: =====
ForecasterAutoreg
=====
Regressor: RandomForestRegressor(random_state=123)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 2
4
25 26 27 28 29 30]
Transformer for y: None
Transformer for exog: None
Window size: 30
Included exogenous: False
Type of exogenous variable: None
Exogenous variables names: None
Training range: [Timestamp('1985-12-31 00:00:00'), Timestamp('2017-12-31 00:0
0:00:00')]
Training index type: DatetimeIndex
Training index frequency: A-DEC
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion': 'squ
ared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None,
'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'mi
n_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n
_jobs': None, 'oob_score': False, 'random_state': 123, 'verbose': 0, 'warm_st
art': False}
Creation date: 2023-04-19 08:37:37
Last fit date: 2023-04-19 08:37:37
Skforecast version: 0.5.1
Python version: 3.9.7
```

A continuación se realizan predicciones de la serie temporal utilizando el modelo entrenado `forecaster` en la celda anterior. `steps` indica el número de periodos hacia adelante que se quieren predecir. El resultado es un array con las predicciones de la serie temporal.

```
In [10]: steps = 3
predicciones = forecaster.predict(steps=steps)
predicciones
```

```
Out[10]: 2018-12-31    19.521
2019-12-31    19.529
2020-12-31    19.543
Freq: A-DEC, Name: pred, dtype: float64
```

Analizar resultados

En este caso, al utilizar un modelo de `ForecasterAutoreg` con `RandomForestRegressor`, se han elegido tres métricas para evaluar el modelo: MAE, MSE y RMSE. Se ha decidido no utilizar dos métricas que se venían utilizando previamente: R2 y Varianza explicada. Esto se debe a que el coeficiente de determinación (R2) es una medida común de la calidad de un modelo de regresión, pero puede no ser la medida más adecuada para evaluar un modelo de `ForecasterAutoreg`. En un modelo de predicción, el objetivo es predecir valores futuros y no ajustar los valores pasados. El R2 se basa en la comparación de los valores ajustados y observados en los datos de entrenamiento, lo que no es necesariamente una medida relevante para la capacidad del modelo para predecir valores futuros.

La varianza explicada, por otro lado, es una medida que indica cuánto de la varianza total de los datos puede explicarse mediante el modelo. Esta medida es útil para evaluar la capacidad del modelo para explicar la variabilidad en los datos. Sin embargo, en el caso de un modelo de predicción, esta medida puede no ser relevante, ya que lo importante es la capacidad del modelo para predecir valores futuros. En resumen, se han elegido las métricas más adecuadas para evaluar el modelo de predicción en cuestión y se han dejado de lado aquellas que no son relevantes para este propósito.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.

```
In [11]: 'MAE: {:.2f} micras'.format(mean_absolute_error(datos_test['y'],
predicciones))
```

```
Out[11]: 'MAE: 0.20 micras'
```

Interpretación: El valor obtenido de 0.20 micras indica que, en promedio, el modelo se equivoca en esa cantidad al predecir la finura de la lana. Es decir, en promedio, las predicciones del modelo tienen un error absoluto medio de 0.20 micras en relación al valor real de la producción de lana. Una forma de interpretar esta métrica es considerar el rango de la escala de la finura de la lana, que va de 18.5 a 21.0, y evaluar que el error medio absoluto es relativamente bajo en relación a dicha escala. Por lo tanto, podemos concluir que el modelo tiene un desempeño aceptable en términos de precisión en la predicción de la finura de la lana.

Error cuadrático medio

Error cuadrático medio (MSE): Es la media de las diferencias entre el valor objetivo y el predicho al cuadrado. Al elevar al cuadrado los errores, magnifica los errores grandes, por lo que hay que utilizarla con cuidado cuando tenemos valores anómalos en nuestro conjunto de datos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error`.

```
In [12]: 'MSE: {:.2f} micras'.format(mean_squared_error(datos_test['y'],
predicciones))
```

```
Out[12]: 'MSE: 0.05 micras'
```

Interpretación: El valor del MSE obtenido de 0.05 micras indica que, en promedio, el modelo tiene un error cuadrático medio de esa cantidad al predecir la finura de la lana. Es decir, la diferencia cuadrática promedio entre las predicciones del modelo y los valores reales de la producción de lana es de 0.05 micras. Dado que la escala de la finura de la lana va de 18.5 a 21.0, se podría decir que el modelo tiene un error cuadrático medio relativamente bueno en este caso.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica anterior. La ventaja de esta métrica es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [13]: 'RMSE: {:.2f} micras'.format(mean_squared_error(datos_test['y'],
predicciones, squared = False))
```

```
Out[13]: 'RMSE: 0.23 micras'
```

Interpretación: En este caso, el RMSE de 0.23 indica que, en promedio, las predicciones del modelo tienen un error de aproximadamente 0.23 micras en relación al valor real de la finura de la lana. Es una métrica relativamente aceptable para evaluar la capacidad de predicción del modelo.

El siguiente código genera un gráfico que muestra la serie de tiempo de los datos de entrenamiento (`datos_train['y']`) y luego agrega dos puntos de datos adicionales en el gráfico. Estos son los datos de prueba (`datos_test['y']`) representados por puntos rojos y las predicciones (`predicciones.values`) representadas por otro conjunto de puntos de color azul.

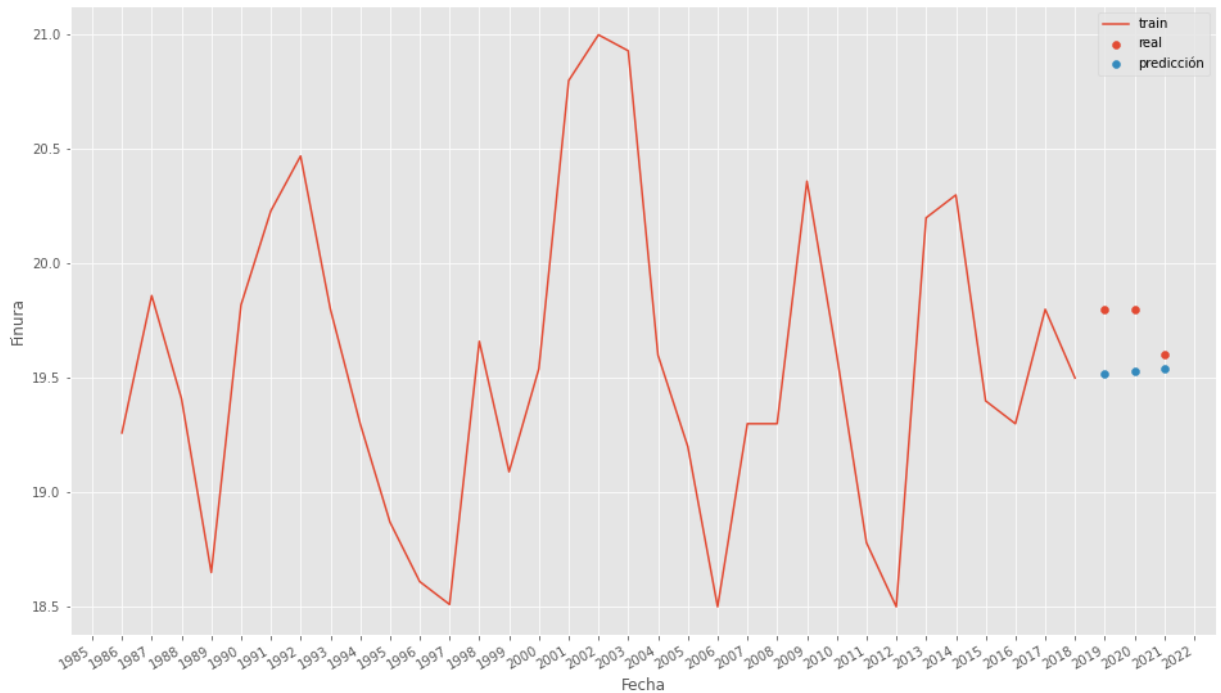
```
In [14]: years = YearLocator()
plt.style.use('ggplot')

fig, ax = plt.subplots(figsize=(16, 9))
ax.plot(datos_train.index, datos_train['y'], label='train')
```

```

ax.scatter(datos_test.index, datos_test['y'], label='real')
ax.scatter(predicciones.index, predicciones.values,
label='predicción')
ax.set_xlabel('Fecha')
ax.set_ylabel('Finura')
ax.legend()
ax.xaxis.set_major_locator(years)
plt.xticks(rotation=30, ha='right')
ax.xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))

```



El gráfico muestra que en general las predicciones de la finura fueron muy precisas, ya que dos de ellas están cercanas al valor real y la tercera se encuentra muy próxima al mismo. Como se indicó previamente a través del MSE, el modelo presenta un error cuadrático medio de 0.05 micras al realizar las predicciones.

Finalmente guardamos el modelo para ser usado en la aplicación

```

In [15]: save_pickle(forecaster, 'modelos/modelo_prediccion_lana_finura')

```

3.15. Modelo para predecir el rinde

La siguiente celda de código utiliza algunas de las bibliotecas más populares para el tratamiento y análisis de datos.

- `import numpy as np` : se importa la biblioteca `NumPy` y se le asigna un alias `np` , esto se hace para que sea más fácil de usar y escribir en el código. `NumPy` es una biblioteca que proporciona soporte para cálculos matemáticos y de matrices en Python.
- `import pandas as pd` : se importa la biblioteca `Pandas` y se le asigna un alias `pd` . `Pandas` es una biblioteca que proporciona estructuras de datos flexibles para el análisis de datos y herramientas para trabajar con datos estructurados.
- `import matplotlib.pyplot as plt` : se importa la biblioteca `Matplotlib` , que se utiliza para crear visualizaciones en Python, y se le asigna un alias `plt` .
- `%matplotlib inline` : se habilita la visualización en línea de gráficos `Matplotlib` , lo que significa que los gráficos se mostrarán directamente en el cuaderno de Jupyter en el que se está trabajando.
- `from sklearn.ensemble import RandomForestRegressor` : se importa la biblioteca `RandomForestRegressor` que se utiliza para crear modelos de regresión aleatorios que utilizan árboles de decisión. Estos modelos se utilizan a menudo para predecir valores numéricos a partir de características del conjunto de datos.
- `from sklearn.metrics import mean_squared_error, mean_absolute_error` : se importan algunas métricas de evaluación de modelos, incluyendo `mean_squared_error` y `mean_absolute_error` . Estas métricas se utilizan para evaluar el rendimiento del modelo en términos de precisión y ajuste.
- `from skforecast.ForecasterAutoreg import ForecasterAutoreg` : se importa la biblioteca `ForecasterAutoreg` que se especializa en modelos de predicción de series de tiempo en Python.
- `from statsmodels.iolib.smpickle import save_pickle` : se importa `save_pickle` que se utiliza para guardar modelos de predicción en archivos para su posterior uso.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import YearLocator
%matplotlib inline

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,
mean_absolute_error

from skforecast.ForecasterAutoreg import ForecasterAutoreg
```

```
from statsmodels.iolib.smpickle import save_pickle
```

Preparación de los datos

A continuación, en la siguiente celda de código, se presenta un conjunto de datos que fue creado manualmente utilizando información del Mirabueno. El conjunto de datos abarca un período de 36 años y se enfoca en la medición de el rinde de la lana y su correspondiente fecha.

```
In [2]: datos = pd.read_csv('datos/lana_mirabueno_rinde.csv', sep=',')
datos
```

```
Out[2]:
```

	fecha	rinde
--	-------	-------

0	1985-12-31	60.46
1	1986-12-31	61.52
2	1987-12-31	55.77
3	1988-12-31	61.69
4	1989-12-31	57.13
5	1990-12-31	59.84
6	1991-12-31	60.56
7	1992-12-31	54.04
8	1993-12-31	58.81
9	1994-12-31	61.83
10	1995-12-31	64.37
11	1996-12-31	60.43
12	1997-12-31	56.21
13	1998-12-31	61.39
14	1999-12-31	53.18
15	2000-12-31	65.27
16	2001-12-31	61.80
17	2002-12-31	61.65
18	2003-12-31	61.10
19	2004-12-31	57.50
20	2005-12-31	56.80
21	2006-12-31	57.80
22	2007-12-31	58.30
23	2008-12-31	58.90
24	2009-12-31	53.10
25	2010-12-31	56.70
26	2011-12-31	55.00
27	2012-12-31	53.90

	fecha	rinde
28	2013-12-31	53.90
29	2014-12-31	56.10
30	2015-12-31	59.10
31	2016-12-31	60.00
32	2017-12-31	59.30
33	2018-12-31	61.30
34	2019-12-31	56.90
35	2020-12-31	56.10

En resumen, el siguiente código convierte la columna `fecha` en el índice, elimina la columna original, renombra la columna `rinde` como `y`, convierte los datos a una frecuencia anual y finalmente ordena los datos por el índice. El resultado final es un DataFrame limpio y organizado que puede ser utilizado para análisis de series de tiempo. La función `datos.head()` muestra las primeras cinco filas del DataFrame resultante.

```
In [3]: datos = datos.set_index(pd.to_datetime(datos['fecha'],
format='%Y/%m/%d')).drop(columns=['fecha']).rename(columns=
{'rinde': 'y'}).asfreq('A').sort_index()
datos.head()
```

```
Out[3]:
```

	y
fecha	
1985-12-31	60.46
1986-12-31	61.52
1987-12-31	55.77
1988-12-31	61.69
1989-12-31	57.13

Las siguientes celdas utilizan las funciones `max` y `min` de Pandas para obtener el rango de valores del `rinde`.

```
In [4]: datos['y'].max()
```

```
Out[4]: 65.27
```

```
In [5]: datos['y'].min()
```

```
Out[5]: 53.1
```

El siguiente código verifica si el índice temporal del DataFrame (`datos`) está completo, es decir, si no faltan fechas entre la fecha mínima y la fecha máxima en el índice.

Para hacerlo, se crea un rango de fechas (`pd.date_range()`) que va desde la fecha mínima hasta la fecha máxima del índice, con la misma frecuencia que tiene el índice (`datos.index.freq`). Luego, se compara si todas las fechas del índice (`datos.index`) están presentes en este rango de fechas creado. Si esto es verdadero para todas las fechas, entonces se puede decir que el índice temporal está completo.

```
In [6]: (datos.index == pd.date_range(
        start = datos.index.min(),
        end    = datos.index.max(),
        freq   = datos.index.freq)
        ).all()
```

Out[6]: True

El siguiente código divide los datos en conjuntos de entrenamiento y prueba para su posterior uso en el modelado y la evaluación. En particular, se utiliza el parámetro `steps` para indicar la cantidad de puntos de datos que se deben reservar para el conjunto de prueba. Luego, se crea el conjunto de entrenamiento `datos_train` seleccionando todos los puntos de datos menos los últimos `steps` y el conjunto de prueba `datos_test` seleccionando los últimos `steps` puntos de datos.

```
In [7]: steps = 3
datos_train = datos[:-steps]
datos_test  = datos[-steps:]
```

La siguiente celda de código muestra un gráfico con los datos de entrenamiento y prueba. El eje x muestra las fechas en formato de años, con una separación de un año entre cada marca de tiempo. El eje y muestra el rinde de la lana.

`YearLocator()` es una función de la librería `matplotlib.dates` que determina las posiciones de las marcas de tiempo en el eje x. En este caso, se utiliza para establecer las marcas de tiempo en años.

`plt.style.use('ggplot')` se utiliza para aplicar el estilo 'ggplot' a la gráfica.

`ax.plot()` se utiliza para trazar los datos de entrenamiento en la gráfica.

`ax.scatter()` se utiliza para trazar los datos de prueba en la gráfica.

`ax.set_xlabel()` y `ax.set_ylabel()` se utilizan para establecer las etiquetas de los ejes x e y, respectivamente.

`ax.legend()` se utiliza para mostrar la leyenda de la gráfica.

`ax.xaxis.set_major_locator()` se utiliza para establecer el separación de las marcas de tiempo en el eje x.

`plt.xticks()` se utiliza para establecer la rotación y alineación de las marcas de tiempo en el eje x.

`ax.xaxis.set_major_formatter()` se utiliza para establecer el formato de las marcas de tiempo en el eje x.

Finalmente, se imprimen las fechas mínima y máxima de los conjuntos de entrenamiento y prueba junto con su longitud, utilizando `print()`.

```
In [8]: years = YearLocator()
plt.style.use('ggplot')

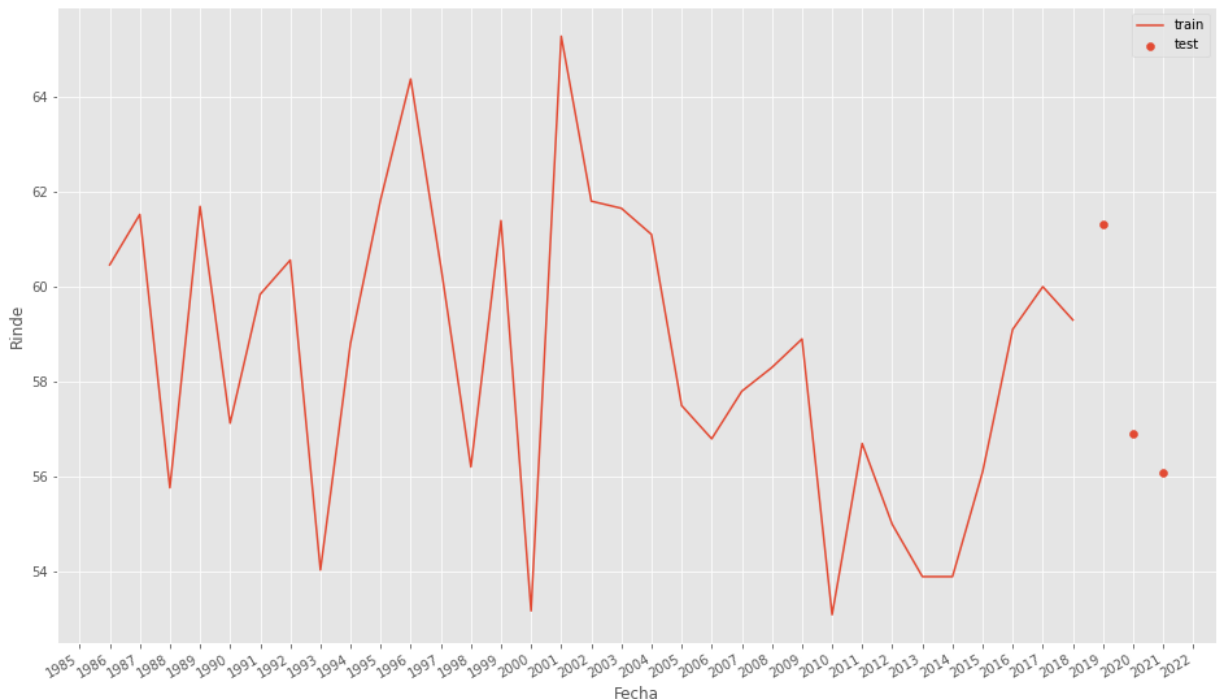
fig, ax = plt.subplots(figsize=(16, 9))
ax.plot(datos_train.index, datos_train['y'], label='train')
ax.scatter(datos_test.index, datos_test['y'], label='test')
ax.set_xlabel('Fecha')
ax.set_ylabel('Rinde')
ax.legend()

ax.xaxis.set_major_locator(years)
plt.xticks(rotation=30, ha='right')
ax.xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))

print(f"Fechas train : {datos_train.index.min()} ---
{datos_train.index.max()} (n={len(datos_train)})")
print(f"Fechas test  : {datos_test.index.min()} ---
{datos_test.index.max()} (n={len(datos_test)})")
```

Fechas train : 1985-12-31 00:00:00 --- 2017-12-31 00:00:00 (n=33)

Fechas test : 2018-12-31 00:00:00 --- 2020-12-31 00:00:00 (n=3)



Entrenar al modelo

Lo siguiente que hacemos es crear y entrenar un modelo de forecasting utilizando un `RandomForestRegressor` y la clase `ForecasterAutoreg` de la librería `skforecast`. Al establecer `random_state=123` en el modelo de `RandomForestRegressor`, se fija la semilla en 123 para la generación de los números aleatorios en el modelo, lo que permite que los resultados del modelo sean reproducibles. El modelo utiliza 24 lags (retrasos) como variables predictoras y se entrena con los datos de entrenamiento `datos_train`. Finalmente, se devuelve el modelo entrenado `forecaster`.

```
In [9]: forecaster = ForecasterAutoreg(
        regressor =
        RandomForestRegressor(random_state=123),
        lags = 24
    )

    forecaster.fit(y=datos_train['y'])
    forecaster
```

```
Out[9]: =====
ForecasterAutoreg
=====
Regressor: RandomForestRegressor(random_state=123)
Lags: [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]
Transformer for y: None
Transformer for exog: None
Window size: 24
Included exogenous: False
Type of exogenous variable: None
Exogenous variables names: None
Training range: [Timestamp('1985-12-31 00:00:00'), Timestamp('2017-12-31 00:00:00')]
Training index type: DatetimeIndex
Training index frequency: A-DEC
Regressor parameters: {'bootstrap': True, 'ccp_alpha': 0.0, 'criterion': 'squared_error', 'max_depth': None, 'max_features': 1.0, 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'random_state': 123, 'verbose': 0, 'warm_start': False}
Creation date: 2023-04-19 09:15:09
Last fit date: 2023-04-19 09:15:09
Skforecast version: 0.5.1
Python version: 3.9.7
```

A continuación se realizan predicciones de la serie temporal utilizando el modelo entrenado `forecaster` en la celda anterior. `steps` indica el número de periodos hacia adelante que se quieren predecir. El resultado es un array con las predicciones de la serie temporal.

```
In [10]: steps = 3

    predicciones = forecaster.predict(steps=steps)
```

```
predicciones
```

```
Out[10]: 2018-12-31    57.834
          2019-12-31    57.210
          2020-12-31    56.714
          Freq: A-DEC, Name: pred, dtype: float64
```

Analizar resultados

En este caso, al utilizar un modelo de `ForecasterAutoreg` con `RandomForestRegressor`, se han elegido tres métricas para evaluar el modelo: MAE, MSE y RMSE. Se ha decidido no utilizar dos métricas que se venían utilizando previamente: R2 y Varianza explicada. Esto se debe a que el coeficiente de determinación (R2) es una medida común de la calidad de un modelo de regresión, pero puede no ser la medida más adecuada para evaluar un modelo de `ForecasterAutoreg`. En un modelo de predicción, el objetivo es predecir valores futuros y no ajustar los valores pasados. El R2 se basa en la comparación de los valores ajustados y observados en los datos de entrenamiento, lo que no es necesariamente una medida relevante para la capacidad del modelo para predecir valores futuros.

La varianza explicada, por otro lado, es una medida que indica cuánto de la varianza total de los datos puede explicarse mediante el modelo. Esta medida es útil para evaluar la capacidad del modelo para explicar la variabilidad en los datos. Sin embargo, en el caso de un modelo de predicción, esta medida puede no ser relevante, ya que lo importante es la capacidad del modelo para predecir valores futuros. En resumen, se han elegido las métricas más adecuadas para evaluar el modelo de predicción en cuestión y se han dejado de lado aquellas que no son relevantes para este propósito.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de `sklearn` el método es `mean_absolute_error`.

```
In [11]: 'MAE: {:.2f} rinde'.format(mean_absolute_error(datos_test['y'],
predicciones))
```

```
Out[11]: 'MAE: 1.46 rinde'
```

Interpretación: El valor obtenido de 1.49 indica que, en promedio, las predicciones del modelo tienen un error absoluto medio de 1.49 en relación al valor real del rinde. Es decir, si tomamos cualquier predicción del modelo y la comparamos con el valor real correspondiente, en promedio la diferencia absoluta será de 1.49. En este caso, como el rango de valores del rinde va de 53.1 a 65.27, el MAE podría considerarse relativamente bueno en relación a la magnitud de los valores del rinde.

Error cuadrático medio

Error cuadrático medio (MSE): Es la media de las diferencias entre el valor objetivo y el predicho al cuadrado. Al elevar al cuadrado los errores, magnifica los errores grandes, por lo que hay que utilizarla con cuidado cuando tenemos valores anómalos en nuestro conjunto de datos. Mientras mas cercano a cero mejor. Dentro de `sklearn` el método es `mean_squared_error`.

```
In [12]: 'MSE: {:.2f} rinde'.format(mean_squared_error(datos_test['y'],
predicciones))
```

```
Out[12]: 'MSE: 4.16 rinde'
```

Interpretación: un MSE de 4.64 indica que, en promedio, el modelo se equivoca en esa cantidad al predecir el rinde de la lana. En otras palabras, la diferencia entre los valores predichos y los valores reales del rinde, al ser elevados al cuadrado y promediados, resulta en un valor de 4.64. Al ser una medida cuadrática, penaliza más los errores grandes que los errores pequeños. Por lo tanto, el valor de 4.64 indica que el modelo puede estar teniendo dificultades para predecir correctamente los valores extremos del rango de valores del rinde.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica anterior. La ventaja de esta métrica es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [13]: 'RMSE: {:.2f} rinde'.format(mean_squared_error(datos_test['y'],
predicciones, squared = False))
```

```
Out[13]: 'RMSE: 2.04 rinde'
```

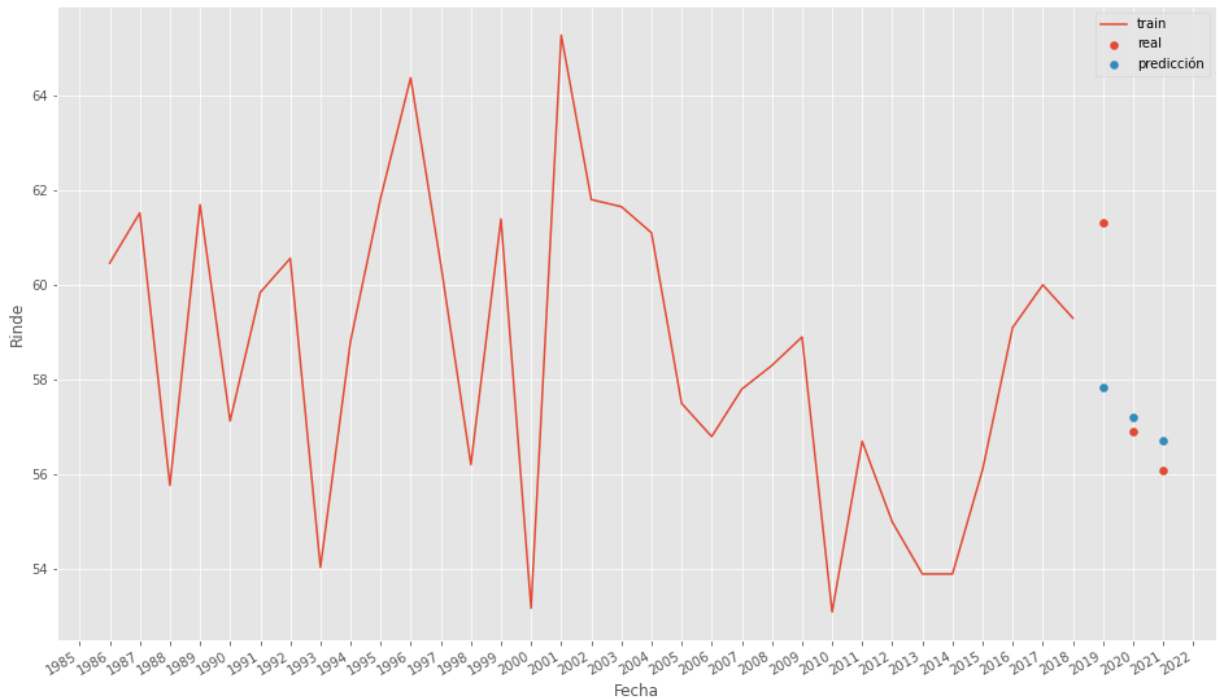
Interpretación: un valor de 2.15 indica que, en promedio, las predicciones del modelo se desvían en 2.15 unidades de la medida de la variable objetivo (en este caso, el rinde de la lana). En este caso, dado que el rango de valores del rinde va de 53.1 a 65.27, una desviación de 2.15 unidades podría considerarse aceptable.

El siguiente código genera un gráfico que muestra la serie de tiempo de los datos de entrenamiento (`datos_train['y']`) y luego agrega dos puntos de datos adicionales en el gráfico. Estos son los datos de prueba (`datos_test['y']`) representados por puntos rojos y las predicciones (`predicciones.values`) representadas por otro conjunto de puntos de color azul.

```
In [14]: years = YearLocator()
plt.style.use('ggplot')

fig, ax = plt.subplots(figsize=(16, 9))
ax.plot(datos_train.index, datos_train['y'], label='train')
ax.scatter(datos_test.index, datos_test['y'], label='real')
ax.scatter(predicciones.index, predicciones.values,
label='predicción')
ax.set_xlabel('Fecha')
ax.set_ylabel('Rinde')
ax.legend()
ax.xaxis.set_major_locator(years)
```

```
plt.xticks(rotation=30, ha='right')
ax.xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y'))
```



El gráfico muestra que en general las predicciones del rinde fueron bastante aceptables, ya que dos de ellas están muy cerca del valor real. Sin embargo, para el año 2019 el modelo tuvo dificultades para predecir el rinde. Se podría considerar que este valor es casi extremo, y como indicó el MSE, el modelo tiende a tener dificultades para predecir este tipo de valores.

Finalmente guardamos el modelo para ser usado en la aplicación

```
In [15]: save_pickle(forecaster, 'modelos/modelo_prediccion_lana_rinde')
```

3.16. Árbol de regresión Kilos Lana - Dataset 2

Vamos a utilizar un árbol de regresión para predecir la cantidad de kilos lana que se obtendrá de un rebaño de ovejas en función de diversas variables independientes.

Librerías

Primero comenzaremos importando las librerías necesarias. Para el tratamiento de los datos utilizaremos NumPy y Pandas, dos de las bibliotecas más importantes para el análisis de datos en Python. También usaremos Matplotlib para la creación de gráficos y visualizaciones. Para el modelado, implementaremos un árbol de regresión con la ayuda de Scikit-Learn, una biblioteca de aprendizaje automático de código abierto en Python. Utilizaremos DecisionTreeRegressor para construir el modelo y plot_tree para visualizarlo. Además, usaremos export_text para imprimir la regla de decisión del árbol en un formato fácil de leer. Para optimizar el modelo, utilizaremos GridSearchCV, una herramienta de Scikit-Learn que nos permitirá encontrar los mejores parámetros para el árbol de regresión. Por último, utilizaremos varias métricas de evaluación como mean_squared_error, r2_score, explained_variance_score y mean_absolute_error para evaluar el rendimiento del modelo y hacer mejoras en su precisión.

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.tree import plot_tree
from sklearn.tree import export_text
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score,
explained_variance_score, mean_absolute_error
```

Datos

Para nuestro análisis, utilizaremos el dataset 2, el cual contiene información sobre la producción y las condiciones climáticas anuales de los departamentos de Chubut, así como del campo El Mirabueno. El objetivo es ajustar un modelo de regresión que nos permita predecir la cantidad de Lana en función de las variables disponibles en el conjunto de datos.

```
In [2]: datos =
pd.read_csv("../datos_analisis/dataset_produccion_clima_chubut_mirab
datos.head(3)
```

```
Out[2]:
```

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lana
0	2012-12-31	1294000	BIEDMA	5785	93378	30168	48447	16071	193849	649130.0

	fecha	hectareas	departamento	carneros	ovejas	borregos	capones	cordero/as	total_ovinos	kilos_lana
1	2013-12-31	1294000	BIEDMA	5376	82998	25341	47350	26832	187897	620176.0
2	2014-12-31	1294000	BIEDMA	4785	82706	29379	35155	29900	181925	649130.0

Entrenamiento del modelo

La clase `DecisionTreeRegressor` del módulo `sklearn.tree` permite entrenar árboles de decisión para problemas de regresión. A continuación, se ajusta un árbol de regresión empleando como variable respuesta **kilos_lana** y como predictores **cant_lluvia**, **total_ovinos** y **temp_media**.

Como en todo estudio de regresión, no solo es importante ajustar el modelo, sino también cuantificar su capacidad para predecir nuevas observaciones. Para poder hacer la posterior evaluación, se lleva a cabo la división del conjunto de datos en dos grupos: uno para entrenamiento y otro para testeo. Se seleccionan como variables predictoras (`X_data`) las variables "cant_lluvia", "total_ovinos" y "temp_media", mientras que la variable respuesta (`y_data`) es "kilos_lana". La división en dos conjuntos se realiza mediante la función "train_test_split" del módulo "sklearn.model_selection", la cual permite seleccionar el tamaño del conjunto de testeo mediante el argumento "test_size" y fijar la semilla aleatoria con "random_state".

```
In [3]: X_data = datos[['cant_lluvia', 'total_ovinos', 'temp_media']]
y_data = datos[['kilos_lana']]
X_train, X_test, y_train, y_test = train_test_split(
                                            X_data,
                                            y_data,
                                            random_state = 123
                                            )
```

Luego, se crea el objeto "modelo" correspondiente a un árbol de regresión de profundidad máxima igual a 3 y con semilla aleatoria fija en 123. La profundidad máxima indica la máxima cantidad de divisiones que se permiten en el árbol, mientras que la semilla aleatoria asegura que el modelo sea reproducible.

```
In [4]: modelo = DecisionTreeRegressor(
            max_depth      = 3,
            random_state    = 123
        )
```

Por último, se entrena el modelo con los datos de entrenamiento mediante la función "fit" del objeto "modelo". Esta función ajusta el árbol de regresión utilizando las variables predictoras y la variable respuesta del conjunto de entrenamiento, de manera que pueda predecir nuevos valores de la variable respuesta en base a los valores de las variables predictoras.

```
In [5]: modelo.fit(X_train, y_train)
```

```
Out[5]: ▼ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=3, random_state=123)
```

Una vez entrenado el árbol, se puede representar mediante la combinación de las funciones `plot_tree()` y `export_text()`. La función `plot_tree()` dibuja la estructura del árbol y muestra el número de observaciones y valor medio de la variable respuesta en cada nodo. La función `export_text()` representa esta misma información en formato texto.

Las siguientes líneas de código imprimen la profundidad del árbol de regresión y el número de nodos terminales del mismo, respectivamente. La profundidad del árbol se refiere a la longitud del camino más largo desde la raíz del árbol hasta una hoja, es decir, el número máximo de divisiones que se realizaron para obtener las hojas. Por su parte, el número de nodos terminales se refiere a la cantidad de hojas en el árbol, es decir, la cantidad de nodos finales que se obtienen después de realizar las divisiones. Estas métricas son útiles para evaluar la complejidad del árbol y la cantidad de información que se está considerando para hacer las predicciones.

```
In [6]: 'Profundidad del árbol: {:.0f}'.format(modelo.get_depth())
```

```
Out[6]: 'Profundidad del árbol: 3'
```

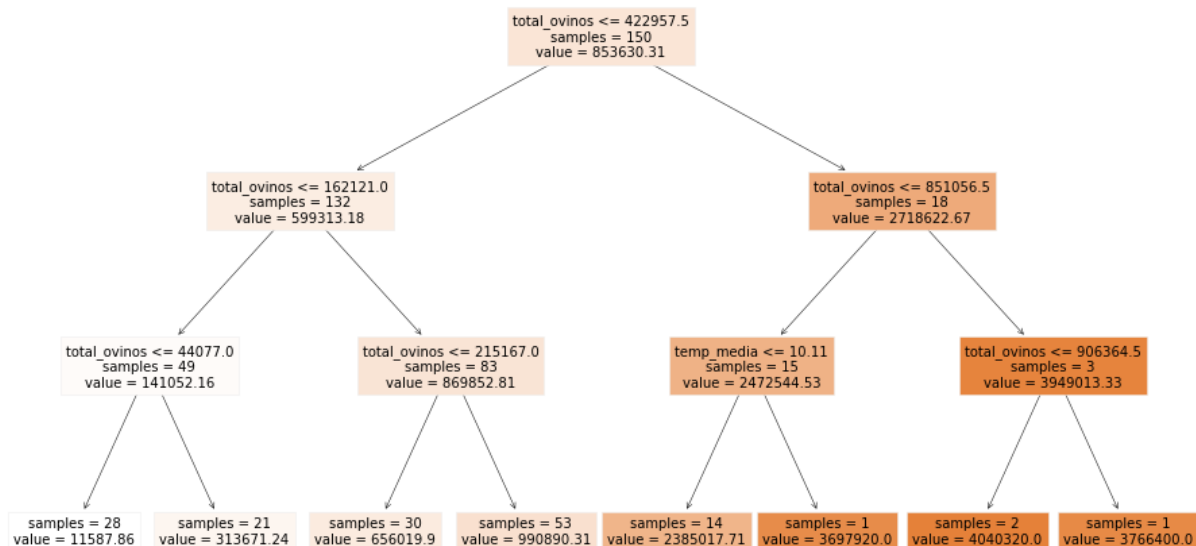
```
In [7]: 'Número de nodos terminales: {:.0f}'.format(modelo.get_n_leaves())
```

```
Out[7]: 'Número de nodos terminales: 8'
```

La siguiente celda genera una visualización del árbol de regresión que se ha creado mediante la clase `DecisionTreeRegressor`. La función `plot_tree()` se encarga de generar un diagrama del árbol con nodos y hojas etiquetados con las variables que se utilizaron para crear el modelo y los valores de división de cada nodo.

El argumento `decision_tree` especifica el modelo que se desea visualizar, mientras que `feature_names` y `class_names` son las etiquetas que se le asignan a las variables predictoras y a la variable respuesta, respectivamente. Los parámetros adicionales, como `filled`, `impurity`, `fontsize` y `precision`, permiten personalizar la apariencia de la visualización.

```
In [37]: fig, ax = plt.subplots()
plot = plot_tree(
    decision_tree = modelo,
    feature_names = X_data.columns,
    class_names   = 'kilos_lana',
    filled        = True,
    impurity      = False,
    fontsize      = 10,
    precision     = 2,
    ax            = ax
)
```



El primer nodo del árbol evalúa la condición en la variable "total_ovinos". Para las muestras que llegan a este nodo, el valor de "total_ovinos" es menor o igual a 422957.50. Se observa que 150 muestras cumplen con esta condición y que el valor promedio de la variable objetivo "kilos_lana" para estas muestras es de 853630.31.

Siguiendo la rama izquierda del árbol, el siguiente nodo indica que si el valor de "total_ovinos" es menor o igual a 162121, entonces la predicción para la variable respuesta "kilos_lana" es de 599313.18. Este nodo contiene 132 muestras que cumplen con esta condición.

El nodo siguiente a la izquierda contiene 49 muestras que también cumplen con esta condición, lo que significa que hay 49 observaciones con "total_ovinos" menor o igual a 44077. La predicción para estas 49 observaciones es de 141052.16 kilos de lana.

Este nodo finaliza en dos hojas, lo que significa que no se pueden realizar más divisiones. El número de muestras en una de las hojas es de 28, lo que indica la cantidad de registros en el conjunto de datos de entrenamiento que cumplen las condiciones para llegar a esa hoja. El valor promedio de la variable objetivo "kilos_lana" para las muestras en esta hoja es de 11587.86 kilos de lana.

Se observa que en un solo nodo se utiliza la temperatura media como condición.

El número de muestras en la otra hoja es de 21, lo que indica la cantidad de registros en el conjunto de datos de entrenamiento que cumplen las condiciones para llegar a esa hoja. El valor promedio de la variable objetivo "kilos_lana" para las muestras en esta hoja es de 313671.24 kilos de lana.

En resumen, este árbol de regresión lineal utiliza la variable "total_ovinos" para hacer predicciones sobre la producción de lana en función de las condiciones meteorológicas y la cantidad de ovinos. Se observa una reducción en la cantidad de muestras a medida que se profundiza en el árbol, lo que indica que las divisiones posteriores se basan en un número menor de observaciones. Además, los valores promedio de la variable objetivo varían en función de las condiciones de las muestras en cada hoja.

La función `export_text` permite exportar la estructura del árbol de regresión en formato de texto e imprimirlo. En este caso, muestra información acerca de la estructura del árbol de decisión, incluyendo la

regla de decisión utilizada para cada nodo. El argumento `decision_tree` especifica el modelo a exportar y `feature_names` se utiliza para nombrar las variables predictoras en el texto.

```
In [9]: print(export_text(decision_tree=modelo,
                        feature_names=list(X_data.columns)))
```

```
|--- total_ovinos <= 422957.50
|   |--- total_ovinos <= 162121.00
|   |   |--- total_ovinos <= 44077.00
|   |   |   |--- value: [11587.86]
|   |   |--- total_ovinos > 44077.00
|   |   |   |--- value: [313671.24]
|   |--- total_ovinos > 162121.00
|   |   |--- total_ovinos <= 215167.00
|   |   |   |--- value: [656019.90]
|   |   |--- total_ovinos > 215167.00
|   |   |   |--- value: [990890.31]
|--- total_ovinos > 422957.50
|   |--- total_ovinos <= 851056.50
|   |   |--- temp_media <= 10.11
|   |   |   |--- value: [2385017.71]
|   |   |--- temp_media > 10.11
|   |   |   |--- value: [3697920.00]
|   |--- total_ovinos > 851056.50
|   |   |--- total_ovinos <= 906364.50
|   |   |   |--- value: [4040320.00]
|   |   |--- total_ovinos > 906364.50
|   |   |   |--- value: [3766400.00]
```

Siguiendo la segunda rama del árbol, se observa que el valor de la lana estará determinado por la satisfacción de las siguientes condiciones:

- `total_ovinos > 422957.50`
- `total_ovinos <= 851056.50`
- `temp_media <= 10.11`

En caso de cumplirse dichas condiciones, se espera obtener un valor de 2385017.71 kilogramos de lana.

Importancia de predictores

La importancia de cada predictor en un modelo se calcula en base a la reducción total que logra en el criterio de división en todas las divisiones en las que participa. Si un predictor no se utiliza en ninguna división, no se incluye en el modelo y su importancia es 0. Esto te permite saber cuáles son los predictores más importantes en la creación del modelo y cuáles tienen poco o ningún impacto en la predicción.

El siguiente código crea un DataFrame `importancia_predictores` que contiene dos columnas: `predictor`, que contiene los nombres de las variables predictoras, y `importancia`, que contiene la importancia de cada variable en el modelo de árbol de regresión ajustado. La importancia de cada variable se calcula a partir del criterio de ganancia de información y mide cuánto contribuye cada variable en la reducción de la variabilidad de la variable respuesta (en este caso, `kilos_lana`).

Luego, el DataFrame `importancia_predictores` se ordena en orden descendente de importancia de los predictores, de manera que la variable más importante aparece en primer lugar.

```
In [10]: importancia_predictores = pd.DataFrame(  
        {'predictor': X_data.columns,  
        'importancia':  
        modelo.feature_importances_  
        })  
importancia_predictores.sort_values('importancia',  
                                     ascending=False)
```

```
Out[10]:
```

	predictor	importancia
1	total_ovinos	0.983561
2	temp_media	0.016439
0	cant_lluvia	0.000000

El predictor "total_ovinos" resultó ser el más importante en el modelo, con un valor de importancia de 0.98, lo que indica que su presencia en las divisiones del árbol tiene un gran impacto. En segundo lugar, se encuentra el predictor "temp_media", con una importancia de 0.02. Finalmente, el predictor "cant_lluvia" no aportó nada al modelo, dado que su valor de importancia fue 0, lo que sugiere que no se incluyó en ninguna división del árbol.

Podado del árbol (pruning)

Con la finalidad de reducir la varianza del modelo y así mejorar la capacidad predictiva, se somete al árbol a un proceso de pruning. Para aplicar el proceso de pruning es necesario indicar el argumento `ccp_alpha` que determina el grado de penalización por complejidad. Cuanto mayor es este valor, más agresivo el podado y menor el tamaño del árbol resultante.

La siguiente celda realiza una búsqueda por validación cruzada para encontrar el valor óptimo del parámetro `ccp_alpha` que permita realizar el pruning del árbol. Para ello, se evalúan diferentes valores de `ccp_alpha` en un grid especificado en `param_grid`.

Se define un diccionario llamado `param_grid` que contiene una única clave '`ccp_alpha`' con valores que van de 0 a 80, separados en 20 partes iguales. Este diccionario se utiliza posteriormente en una búsqueda de hiperparámetros utilizando la clase `GridSearchCV` para realizar la búsqueda por validación cruzada y encontrar el mejor valor de `ccp_alpha` en el modelo de Árbol de Decisión.

Se especifica el modelo de árbol de regresión a utilizar, que se crea con la máxima profundidad posible y sin restricciones en el número mínimo de muestras por nodo para permitir su posterior poda.

El parámetro `cv` indica el número de particiones a utilizar (`DecisionTreeRegressor`) en la validación cruzada, en este caso se utiliza un valor de 10. El parámetro `refit` se establece en `True` para refinar el modelo final con todos los datos de entrenamiento después de encontrar el valor óptimo de `ccp_alpha`. Finalmente, `return_train_score` se establece en `True` para que se devuelvan también los resultados de la evaluación en el conjunto de entrenamiento.

```
In [11]: param_grid = {'ccp_alpha': np.linspace(0, 80, 20)}

grid = GridSearchCV(
    estimator = DecisionTreeRegressor(
        max_depth      = None,
        min_samples_split = 2,
        min_samples_leaf = 1,
        random_state    = 123
    ),
    param_grid = param_grid,
    cv         = 10,
    refit      = True,
    return_train_score = True
)

grid.fit(X_train, y_train)
```

```
Out[11]:
```

► **GridSearchCV**

► **estimator: DecisionTreeRegressor**

► DecisionTreeRegressor

El código `grid.best_params` muestra el valor óptimo encontrado para el hiperparámetro `ccp_alpha` durante la búsqueda por validación cruzada realizada en la celda anterior. Esta es la combinación de hiperparámetros que resultó en el mejor desempeño del modelo según el criterio de la búsqueda.

```
In [12]: grid.best_params_
```

```
Out[12]: {'ccp_alpha': 0.0}
```

El resultado de `grid.best_params_` indica que el mejor valor de hiperparámetro encontrado en la búsqueda por validación cruzada es `ccp_alpha=0.0`. Esto significa que no se debe realizar pruning por complejidad constante, ya que el mejor resultado se obtiene con el árbol no podado. La complejidad constante pruning (CCP) es una técnica que se utiliza para evitar el sobreajuste de los árboles de decisión, reduciendo su complejidad al eliminar las ramas que aportan poco a la precisión de las predicciones. Si `ccp_alpha` fuera mayor que cero, entonces se estaría aplicando un criterio para cortar ramas innecesarias del árbol. En este caso, como el valor de `ccp_alpha` óptimo encontrado es cero, el árbol creado no requiere poda alguna.

Una vez identificado el valor óptimo de `ccp_alpha`, se reentrena el árbol indicando este valor en sus argumentos.

El siguiente código crea una instancia del modelo de árbol de decisión con el mejor valor de `ccp_alpha` encontrado a través de la búsqueda por validación cruzada. Luego se imprimen la profundidad del árbol y el número de nodos terminales del modelo final ajustado.

```
In [13]: modelo_final = grid.best_estimator_  
         'Profundidad del árbol: {:.0f}'.format(modelo_final.get_depth())
```

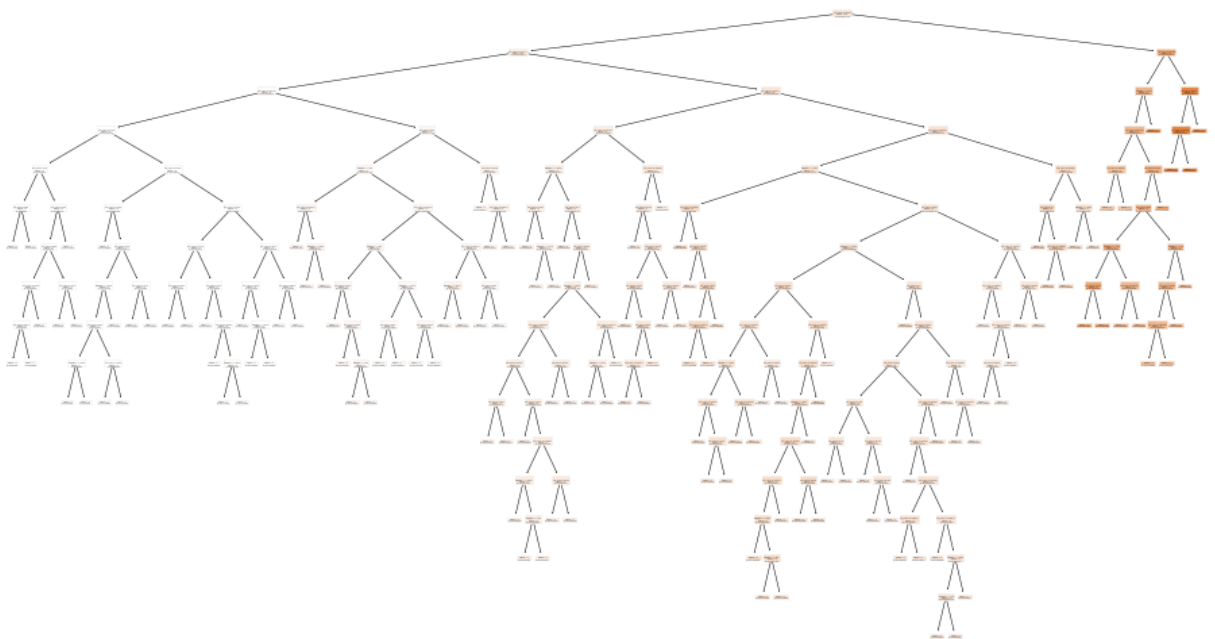
```
Out[13]: 'Profundidad del árbol: 16'
```

```
In [14]: 'Número de nodos terminales:  
         {:.0f}'.format(modelo_final.get_n_leaves())
```

```
Out[14]: 'Número de nodos terminales: 135'
```

La siguiente celda grafica la estructura del árbol final resultante del proceso de entrenamiento y validación. En particular, se utiliza la función `plot_tree` para generar un diagrama de árbol que muestra las diferentes ramas y nodos que componen el modelo final. Se especifican también algunos parámetros para el formato del gráfico, como el tamaño (`figsize`) y la opción de colorear los nodos según la variable respuesta (`filled`). Además, se especifica el nombre de las variables predictivas (`feature_names`) y el nombre de la variable respuesta (`class_names`).

```
In [38]: fig, ax = plt.subplots()  
         plot = plot_tree(  
             decision_tree = modelo_final,  
             feature_names = X_data.columns,  
             class_names   = 'kilos_lana",  
             filled        = True,  
             impurity      = False,  
             ax            = ax  
         )
```



El gráfico es difícil de interpretar debido a la gran profundidad del árbol, lo que hace que los nodos estén demasiado comprimidos y no se puedan apreciar correctamente. Para obtener una visualización más clara de la estructura del árbol de decisión, se puede exportar como texto. Esto proporcionará una lista detallada

de los nodos del árbol, incluyendo las variables y los valores de corte utilizados en cada división, así como el número de muestras y el valor de la variable objetivo en cada nodo terminal o hoja. De esta manera, se podrá entender de manera más clara cómo el árbol toma decisiones a partir de las variables predictoras y cómo se relacionan con la variable objetivo.

In [16]:

```
print(export_text(decision_tree=modelo_final,
feature_names=list(X_data.columns)))
```

```
|--- total_ovinos <= 422957.50
|   |--- total_ovinos <= 162121.00
|   |   |--- total_ovinos <= 44077.00
|   |   |   |--- total_ovinos <= 2482.00
|   |   |   |   |--- cant_lluvia <= 165.90
|   |   |   |   |   |--- cant_lluvia <= 147.00
|   |   |   |   |   |   |--- value: [6516.00]
|   |   |   |   |   |   |--- cant_lluvia > 147.00
|   |   |   |   |   |   |   |--- value: [5245.00]
|   |   |   |   |   |--- cant_lluvia > 165.90
|   |   |   |   |   |   |--- cant_lluvia <= 273.20
|   |   |   |   |   |   |   |--- cant_lluvia <= 190.80
|   |   |   |   |   |   |   |   |--- total_ovinos <= 2196.00
|   |   |   |   |   |   |   |   |   |--- cant_lluvia <= 170.15
|   |   |   |   |   |   |   |   |   |   |--- value: [8739.00]
|   |   |   |   |   |   |   |   |   |   |--- cant_lluvia > 170.15
|   |   |   |   |   |   |   |   |   |   |   |--- value: [8376.00]
|   |   |   |   |   |   |   |   |--- total_ovinos > 2196.00
|   |   |   |   |   |   |   |   |   |--- value: [9164.00]
|   |   |   |   |   |   |   |   |--- cant_lluvia > 190.80
|   |   |   |   |   |   |   |   |   |--- total_ovinos <= 2211.50
|   |   |   |   |   |   |   |   |   |   |--- value: [9892.00]
|   |   |   |   |   |   |   |   |   |   |--- total_ovinos > 2211.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [9931.00]
|   |   |   |   |   |   |   |   |--- cant_lluvia > 273.20
|   |   |   |   |   |   |   |   |   |--- value: [11763.00]
|   |   |   |--- total_ovinos > 2482.00
|   |   |   |   |--- total_ovinos <= 3293.00
|   |   |   |   |   |--- total_ovinos <= 2692.00
|   |   |   |   |   |   |--- value: [15800.00]
|   |   |   |   |   |   |--- total_ovinos > 2692.00
|   |   |   |   |   |   |   |--- cant_lluvia <= 264.80
|   |   |   |   |   |   |   |   |--- temp_media <= 14.15
|   |   |   |   |   |   |   |   |   |--- cant_lluvia <= 140.20
|   |   |   |   |   |   |   |   |   |   |--- temp_media <= 13.65
|   |   |   |   |   |   |   |   |   |   |   |--- value: [9000.00]
|   |   |   |   |   |   |   |   |   |   |   |--- temp_media > 13.65
|   |   |   |   |   |   |   |   |   |   |   |   |--- value: [9193.00]
|   |   |   |   |   |   |   |   |   |--- cant_lluvia > 140.20
|   |   |   |   |   |   |   |   |   |   |--- total_ovinos <= 3087.00
|   |   |   |   |   |   |   |   |   |   |   |--- value: [10510.00]
|   |   |   |   |   |   |   |   |   |   |--- total_ovinos > 3087.00
|   |   |   |   |   |   |   |   |   |   |   |--- value: [10336.00]
|   |   |   |   |   |   |   |   |--- temp_media > 14.15
|   |   |   |   |   |   |   |   |   |--- value: [12300.00]
|   |   |   |   |   |   |   |   |--- cant_lluvia > 264.80
```



```
| | | | | | | --- total_ovinos <= 2917.50  
| | | | | | | |--- value: [11600.00]  
| | | | | | | |--- total_ovinos > 2917.50  
| | | | | | | |--- value: [12400.00]  
| | | | | | |--- total_ovinos > 3293.00  
| | | | | | |--- total_ovinos <= 3936.50  
| | | | | | |--- cant_lluvia <= 111.15  
| | | | | | |--- total_ovinos <= 3575.00  
| | | | | | |--- value: [12136.00]  
| | | | | | |--- total_ovinos > 3575.00  
| | | | | | |--- value: [13197.00]  
| | | | | | |--- cant_lluvia > 111.15  
| | | | | | |--- cant_lluvia <= 159.20  
| | | | | | |--- value: [18167.00]  
| | | | | | |--- cant_lluvia > 159.20  
| | | | | | |--- cant_lluvia <= 198.35  
| | | | | | |--- value: [13522.00]  
| | | | | | |--- cant_lluvia > 198.35  
| | | | | | |--- temp_media <= 14.15  
| | | | | | |--- value: [17349.00]  
| | | | | | |--- temp_media > 14.15  
| | | | | | |--- value: [15600.00]  
| | | | | | |--- total_ovinos > 3936.50  
| | | | | | |--- total_ovinos <= 4722.00  
| | | | | | |--- total_ovinos <= 4039.00  
| | | | | | |--- value: [11554.00]  
| | | | | | |--- total_ovinos > 4039.00  
| | | | | | |--- temp_media <= 13.65  
| | | | | | |--- value: [10990.00]  
| | | | | | |--- temp_media > 13.65  
| | | | | | |--- value: [11200.00]  
| | | | | | |--- total_ovinos > 4722.00  
| | | | | | |--- cant_lluvia <= 175.25  
| | | | | | |--- value: [13027.00]  
| | | | | | |--- cant_lluvia > 175.25  
| | | | | | |--- value: [15753.00]  
| | | | | | |--- total_ovinos > 44077.00  
| | | | | | |--- cant_lluvia <= 245.30  
| | | | | | |--- temp_media <= 12.70  
| | | | | | |--- total_ovinos <= 119061.00  
| | | | | | |--- value: [323760.00]  
| | | | | | |--- total_ovinos > 119061.00  
| | | | | | |--- temp_media <= 12.35  
| | | | | | |--- value: [352640.00]  
| | | | | | |--- temp_media > 12.35  
| | | | | | |--- value: [364800.00]  
| | | | | | |--- temp_media > 12.70  
| | | | | | |--- total_ovinos <= 123775.50  
| | | | | | |--- temp_media <= 13.50  
| | | | | | |--- cant_lluvia <= 165.95  
| | | | | | |--- value: [328320.00]  
| | | | | | |--- cant_lluvia > 165.95  
| | | | | | |--- cant_lluvia <= 186.05  
| | | | | | |--- value: [260224.00]  
| | | | | | |--- cant_lluvia > 186.05  
| | | | | | |--- temp media <= 13.10
```

```
| | | | | | | | | | |--- value: [256880.00]
| | | | | | | | | | |--- temp_media > 13.10
| | | | | | | | | | |--- value: [258400.00]
| | | | | | | | | | |--- temp_media > 13.50
| | | | | | | | | | |--- temp_media <= 13.75
| | | | | | | | | | |--- cant_lluvia <= 192.40
| | | | | | | | | | |--- value: [211280.00]
| | | | | | | | | | |--- cant_lluvia > 192.40
| | | | | | | | | | |--- value: [201856.00]
| | | | | | | | | | |--- temp_media > 13.75
| | | | | | | | | | |--- cant_lluvia <= 171.65
| | | | | | | | | | |--- value: [256728.00]
| | | | | | | | | | |--- cant_lluvia > 171.65
| | | | | | | | | | |--- value: [258400.00]
| | | | | | | | | | |--- total_ovinos > 123775.50
| | | | | | | | | | |--- total_ovinos <= 130997.50
| | | | | | | | | | |--- temp_media <= 13.31
| | | | | | | | | | |--- value: [332880.00]
| | | | | | | | | | |--- temp_media > 13.31
| | | | | | | | | | |--- value: [459200.00]
| | | | | | | | | | |--- total_ovinos > 130997.50
| | | | | | | | | | |--- cant_lluvia <= 132.00
| | | | | | | | | | |--- value: [212800.00]
| | | | | | | | | | |--- cant_lluvia > 132.00
| | | | | | | | | | |--- value: [204789.60]
| | | | | | | | | | |--- cant_lluvia > 245.30
| | | | | | | | | | |--- cant_lluvia <= 351.92
| | | | | | | | | | |--- value: [441914.40]
| | | | | | | | | | |--- cant_lluvia > 351.92
| | | | | | | | | | |--- total_ovinos <= 149257.00
| | | | | | | | | | |--- value: [459200.00]
| | | | | | | | | | |--- total_ovinos > 149257.00
| | | | | | | | | | |--- value: [472320.00]
| | | | | | | | | | |--- total_ovinos > 162121.00
| | | | | | | | | | |--- total_ovinos <= 215167.00
| | | | | | | | | | |--- total_ovinos <= 192174.00
| | | | | | | | | | |--- temp_media <= 10.35
| | | | | | | | | | |--- cant_lluvia <= 552.43
| | | | | | | | | | |--- value: [561536.00]
| | | | | | | | | | |--- cant_lluvia > 552.43
| | | | | | | | | | |--- temp_media <= 9.68
| | | | | | | | | | |--- value: [455920.00]
| | | | | | | | | | |--- temp_media > 9.68
| | | | | | | | | | |--- value: [435584.00]
| | | | | | | | | | |--- temp_media > 10.35
| | | | | | | | | | |--- cant_lluvia <= 93.30
| | | | | | | | | | |--- value: [760960.00]
| | | | | | | | | | |--- cant_lluvia > 93.30
| | | | | | | | | | |--- cant_lluvia <= 473.05
| | | | | | | | | | |--- temp_media <= 14.65
| | | | | | | | | | |--- total_ovinos <= 184539.00
| | | | | | | | | | |--- cant_lluvia <= 132.79
| | | | | | | | | | |--- cant_lluvia <= 110.47
| | | | | | | | | | |--- value: [614250.00]
| | | | | | | | | | |--- cant_lluvia > 110.47
| | | | | | | | | | |--- value: [613021.50]
```

[illegible]

[illegible]

```
| | | | | | | | |--- value: [810600.00]
| | | | | | | | |--- cant_lluvia > 318.31
| | | | | | | | |--- total_ovinos <= 237820.50
| | | | | | | | |--- total_ovinos <= 226258.50
| | | | | | | | |--- value: [697200.00]
| | | | | | | | |--- total_ovinos > 226258.50
| | | | | | | | |--- total_ovinos <= 234170.50
| | | | | | | | |--- cant_lluvia <= 508.31
| | | | | | | | |--- value: [735000.00]
| | | | | | | | |--- cant_lluvia > 508.31
| | | | | | | | |--- value: [729750.00]
| | | | | | | | |--- total_ovinos > 234170.50
| | | | | | | | |--- value: [707332.50]
| | | | | | | | |--- total_ovinos > 237820.50
| | | | | | | | |--- cant_lluvia <= 649.06
| | | | | | | | |--- value: [886725.00]
| | | | | | | | |--- cant_lluvia > 649.06
| | | | | | | | |--- value: [898800.00]
| | | | | | | | |--- total_ovinos > 338818.00
| | | | | | | | |--- total_ovinos <= 352793.50
| | | | | | | | |--- cant_lluvia <= 61.70
| | | | | | | | |--- value: [1391040.00]
| | | | | | | | |--- cant_lluvia > 61.70
| | | | | | | | |--- total_ovinos <= 349483.50
| | | | | | | | |--- value: [1545600.00]
| | | | | | | | |--- total_ovinos > 349483.50
| | | | | | | | |--- value: [1474032.00]
| | | | | | | | |--- total_ovinos > 352793.50
| | | | | | | | |--- temp_media <= 12.65
| | | | | | | | |--- value: [1102528.00]
| | | | | | | | |--- temp_media > 12.65
| | | | | | | | |--- value: [982800.00]
| | | | | | | | |--- total_ovinos > 422957.50
| | | | | | | | |--- total_ovinos <= 851056.50
| | | | | | | | |--- temp_media <= 10.11
| | | | | | | | |--- total_ovinos <= 528713.50
| | | | | | | | |--- total_ovinos <= 469833.50
| | | | | | | | |--- value: [1959900.00]
| | | | | | | | |--- total_ovinos > 469833.50
| | | | | | | | |--- value: [1974000.00]
| | | | | | | | |--- total_ovinos > 528713.50
| | | | | | | | |--- cant_lluvia <= 317.93
| | | | | | | | |--- cant_lluvia <= 137.80
| | | | | | | | |--- temp_media <= 9.79
| | | | | | | | |--- cant_lluvia <= 124.68
| | | | | | | | |--- value: [2891568.00]
| | | | | | | | |--- cant_lluvia > 124.68
| | | | | | | | |--- value: [2930944.00]
| | | | | | | | |--- temp_media > 9.79
| | | | | | | | |--- total_ovinos <= 812739.50
| | | | | | | | |--- value: [2465280.00]
| | | | | | | | |--- total_ovinos > 812739.50
| | | | | | | | |--- value: [2273536.00]
| | | | | | | | |--- cant_lluvia > 137.80
| | | | | | | | |--- temp_media <= 9.56
| | | | | | | | |--- cant lluvia <= 246.35
```

```

| | | | | | | | | |--- total_ovinos <= 799346.50
| | | | | | | | | | |--- value: [2396800.00]
| | | | | | | | | |--- total_ovinos > 799346.50
| | | | | | | | | | |--- value: [2379680.00]
| | | | | | | | | |--- cant_lluvia > 246.35
| | | | | | | | | | |--- value: [2413920.00]
| | | | | | | | | |--- temp_media > 9.56
| | | | | | | | | | |--- value: [2311200.00]
| | | | | | | | |--- cant_lluvia > 317.93
| | | | | | | | |--- value: [3045600.00]
| | | |--- temp_media > 10.11
| | | |--- value: [3697920.00]
| |--- total_ovinos > 851056.50
| |--- cant_lluvia <= 147.28
| | |--- total_ovinos <= 879382.00
| | | |--- value: [3971840.00]
| | | |--- total_ovinos > 879382.00
| | | |--- value: [4108800.00]
| | |--- cant_lluvia > 147.28
| | |--- value: [3766400.00]

```

Si siguiendo la segunda rama del árbol, se observa que el valor de la lana estará determinado por la satisfacción de las siguientes condiciones:

- total_ovinos > 422957.50
- total_ovinos <= 851056.50
- temp_media <= 10.11
- total_ovinos > 528713.50
- cant_lluvia <= 317.93
- cant_lluvia <= 137.80
- temp_media <= 9.79
- cant_lluvia <= 124.68

En caso de cumplirse dichas condiciones, se espera obtener un valor de 2891568.00 kilogramos de lana.

```

In [17]: importancia_predictores = pd.DataFrame(
            {'predictor': X_data.columns,
             'importancia':
             modelo_final.feature_importances_}
            )

            importancia_predictores.sort_values('importancia',
            ascending=False)

```

```

Out[17]:
   predictor  importancia
1  total_ovinos    0.962906
2   temp_media    0.024964
0   cant_lluvia    0.012130

```

El predictor "total_ovinos" resultó ser el más importante en el modelo, con un valor de importancia de 0.98, lo que indica que su presencia en las divisiones del árbol tiene un gran impacto. En segundo lugar, se encuentra el predictor "temp_media", con una importancia de 0.02. Finalmente, el predictor "cant_lluvia" aportó poco al modelo, dado que su valor de importancia fue 0.01, lo que sugiere que se incluyó en pocas divisiones del árbol.

Predicción y evaluación de los modelos

Por último, se evalúa la capacidad predictiva del primer árbol y del árbol final empleando el conjunto de datos de test.

El siguiente código aplica los modelos entrenados previamente en la variable `X_train` e `y_train`, al conjunto de datos de prueba `X_test` y `y_test`. Los modelos predicen el valor de `y_test` para cada observación en `X_test` y almacenan las predicciones en las variables `predicciones` y `predicciones_final`.

```
In [18]: predicciones = modelo.predict(X = X_test)
```

```
In [19]: predicciones_final = modelo_final.predict(X = X_test)
```

Score

Score: El "score" es un método que se incluye en todos los algoritmos y proporciona el resultado de un criterio de evaluación por defecto para el problema al que están dirigidos. En el caso de una regresión, el método score devuelve el coeficiente de determinación, también conocido como R^2 , que se define como el porcentaje de la varianza de la variable dependiente que puede ser explicada por la variable independiente. Esta métrica nos ayuda a evaluar la calidad de nuestro modelo y a compararlo con otros modelos. En general, cuanto mayor sea el valor de R^2 , mejor ajustado estará el modelo a los datos.

```
In [20]: 'Score del modelo en entrenamiento:
{:.2%}'.format(modelo.score(X_train, y_train))
```

```
Out[20]: 'Score del modelo en entrenamiento: 95.47%'
```

```
In [21]: 'Score del modelo en entrenamiento:
{:.2%}'.format(modelo_final.score(X_train, y_train))
```

```
Out[21]: 'Score del modelo en entrenamiento: 100.00%'
```

Interpretación: Un R^2 de 0.9547 y 1.00, respectivamente, indican que el modelo inicial y el modelo final son capaces de explicar el 95.47% y el 100% de la variabilidad de la producción de lana en función de la cantidad de lluvia, la temperatura media y el total de ovinos. Un R^2 de 0.95 es considerado como una métrica muy aceptable, mientras que un valor de 1.00 en el modelo final puede sugerir un sobreajuste del modelo, ya que puede estar demasiado ajustado a los datos de entrenamiento y no generalizar bien en nuevos datos.

```
In [22]: 'Score del modelo en test: {:.2%}'.format(modelo.score(X_test,
y_test))
```

```
Out[22]: 'Score del modelo en test: 80.06%'
```

```
In [23]: 'Score del modelo en test:
{:.2%}'.format(modelo_final.score(X_test, y_test))
```

```
Out[23]: 'Score del modelo en test: 80.01%'
```

Interpretación: El modelo inicial y el modelo final tienen una precisión del 80.06% y 80.01%, respectivamente, al predecir los valores de la variable dependiente (kilos de lana) utilizando las variables independientes (cantidad de lluvia, temperatura media y número total de ovinos) en el conjunto de datos de prueba. Esto significa que ambos modelos pueden explicar alrededor del 80% de la varianza en la variable dependiente con los datos que no han sido vistos antes. Estas son métricas aceptables y sugieren que los modelos tienen un buen rendimiento en la predicción de la variable dependiente.

Varianza explicada

Varianza explicada: es una medida del porcentaje de la varianza de la variable de salida que es explicado por el modelo. Esta métrica evalúa la variación o dispersión de los puntos de datos, y la mejor puntuación posible es 1.0, lo que indica que el modelo explica perfectamente toda la variabilidad de los datos. Sin embargo, una puntuación muy alta en esta métrica también puede indicar un riesgo de overfitting, es decir, que el modelo se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a nuevos datos. En sklearn, se puede utilizar el método **explained_variance_score** para calcular esta métrica.

```
In [24]: 'Varianza Explicada:
{:.2%}'.format(explained_variance_score(y_test, predicciones))
```

```
Out[24]: 'Varianza Explicada: 80.21%'
```

```
In [25]: 'Varianza Explicada:
{:.2%}'.format(explained_variance_score(y_test,
predicciones_final))
```

```
Out[25]: 'Varianza Explicada: 80.40%'
```

Interpretación: En este caso, se evaluó la varianza explicada por el modelo inicial y el modelo final. Se encontró que el modelo inicial es capaz de explicar el 80.21% de la variabilidad observada en la variable de salida (kilos de lana), mientras que el modelo final puede explicar el 80.40%. Esto significa que ambos modelos son capaces de predecir la variable de salida en función de las variables de entrada (lluvia, total de ovejas, temperatura media) con una buena precisión sin indicar un sobreajuste. En general, una varianza explicada del 80% indica que el modelo es aceptable y puede ser utilizado para hacer predicciones.

Error medio absoluto

Error medio absoluto (MAE): Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Mientras mas cercano a cero mejor. Dentro de sklearn el método es **mean_absolute_error**.


```
In [26]: 'MAE: {:.2f}'.format(mean_absolute_error(y_test, predicciones))
```

```
Out[26]: 'MAE: 230492.53'
```

```
In [27]: 'MAE: {:.2f}'.format(mean_absolute_error(y_test,
predicciones_final))
```

```
Out[27]: 'MAE: 214588.80'
```

Interpretación: Los valores obtenidos de 230492.53 y 214588.80 para el error absoluto medio en el modelo 1 y 2, respectivamente, indican que, en promedio, las predicciones del modelo tienen un error de esa cantidad en relación al valor real de la producción de kilos de lana. Si bien el valor absoluto de los errores puede parecer alto, es importante tener en cuenta el contexto y el valor de la producción de lana que se está analizando para evaluar si el error medio es aceptable. En general, se considera que un error absoluto medio inferior al 10% del valor medio de la variable dependiente es aceptable. Dado que la media de producción de lana para este dataset es de 894301.4 kilos, el error del primer modelo representa el 25.79% y el del segundo modelo representa el 24%. Por lo tanto, no son métricas aceptables según esta regla general.

Raíz cuadrada del MSE

Raíz cuadrada de la media del error al cuadrado (RMSE): Es igual a la raíz cuadrada de la métrica MSE, la cual es la media de las diferencias entre el valor objetivo y el predicho al cuadrado, y al elevar al cuadrado magnifica los errores grandes. La ventaja de RMSE es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Mientras mas cercano a cero mejor. Dentro de sklearn el método es `mean_squared_error` con `squared = False`.

```
In [28]: 'RMSE: {:.2f}'.format(mean_squared_error(y_test, predicciones,
squared = False))
```

```
Out[28]: 'RMSE: 340993.64'
```

```
In [29]: 'RMSE: {:.2f}'.format(mean_squared_error(y_test,
predicciones_final, squared = False))
```

```
Out[29]: 'RMSE: 341427.08'
```

Interpretación: La raíz cuadrada del MSE es una medida de la diferencia promedio entre los valores predichos y los valores observados de kilos de lana en el modelo 1 y 2. Los valores de 340993.64 y 341427.08 indican que, en promedio, la diferencia entre las predicciones y los valores reales de kilos de lana es de alrededor de 340993.64 y 341427.08, respectivamente. No son métricas buenas.

R cuadrado

R cuadrado (R^2): es una métrica utilizada para evaluar la capacidad de un modelo para ajustarse a los datos de entrenamiento. A diferencia de otras métricas, R^2 compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo.

Es importante destacar que el resultado del coeficiente de determinación, es un valor que oscila entre menos infinito y 1, siendo 1 el valor máximo que puede tomar. Cuanto más cercano a 1 sea el valor de esta métrica, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de menos infinito, menos ajustado estará el modelo y, por tanto, menos fiable será. Esto se debe a que R^2 refleja la proporción de la varianza total de la variable objetivo que es explicada por nuestro modelo.

En resumen, el valor de R^2 nos proporciona una medida de la calidad del ajuste de nuestro modelo a los datos de entrenamiento, siendo un valor cercano a 1 deseable para garantizar una buena capacidad predictiva.

El coeficiente de determinación se puede calcular utilizando la función `r2_score` de la biblioteca `sklearn`.

```
In [30]: 'R2:{:.2%}'.format(r2_score(y_test, predicciones))
```

```
Out[30]: 'R2:80.06%'
```

```
In [31]: 'R2:{:.2%}'.format(r2_score(y_test, predicciones_final))
```

```
Out[31]: 'R2:80.01%'
```

Interpretación: En este caso, el valor de R cuadrado para los modelos inicial y final es del 80%, lo que indica que aproximadamente el 80% de la variabilidad observada en los datos de kilos de lana producidos se puede explicar por las variables independientes utilizadas en el modelo de regresión. Esto sugiere que el modelo es capaz de predecir la variable de salida (kilos de lana) con una precisión aceptable en función de las variables de entrada proporcionadas (lluvia, total de ovinos, temperatura media). Una R cuadrado del 80% se considera generalmente una métrica buena o sólida para la mayoría de las aplicaciones de regresión.

El problema del coeficiente de determinación, y razón por el cual surge el coeficiente de determinación ajustado, radica en que no penaliza la inclusión de variables explicativas no significativas. Es decir, si al modelo se añaden cinco variables explicativas que guardan poca relación con lo que se intenta predecir, el R cuadrado aumentará. Es por ello que muchos expertos econométricos, estadísticos y matemáticos se oponen al uso del R cuadrado como medida representativa de la bondad del ajuste real.

R cuadrado ajustado

La fórmula del R cuadrado ajustado es como sigue: $R^2_{ajustado} = 1 - \frac{(n-1)}{n-k-1} * (1 - R^2)$

Donde:

- R^2 : coeficiente de determinación
- n : número de observaciones
- k : número de predictores (variables explicativas) en el modelo.

Se va a crear una función que aplique esta fórmula a los datos.

```
In [32]: def r2_ajustado(n,k,r2):  
         return 1-((n-1)/(n-k-1))*(1-r2)
```

```
In [33]: n = len(y_test)
k = 3
r2 = r2_score(y_test, predicciones)

'R2 ajustado: {:.2%}'.format(r2_ajustado(n,k,r2))
```

```
Out[33]: 'R2 ajustado: 78.79%'
```

```
In [34]: n = len(y_test)
k = 3
r2 = r2_score(y_test, predicciones_final)

'R2 ajustado: {:.2%}'.format(r2_ajustado(n,k,r2))
```

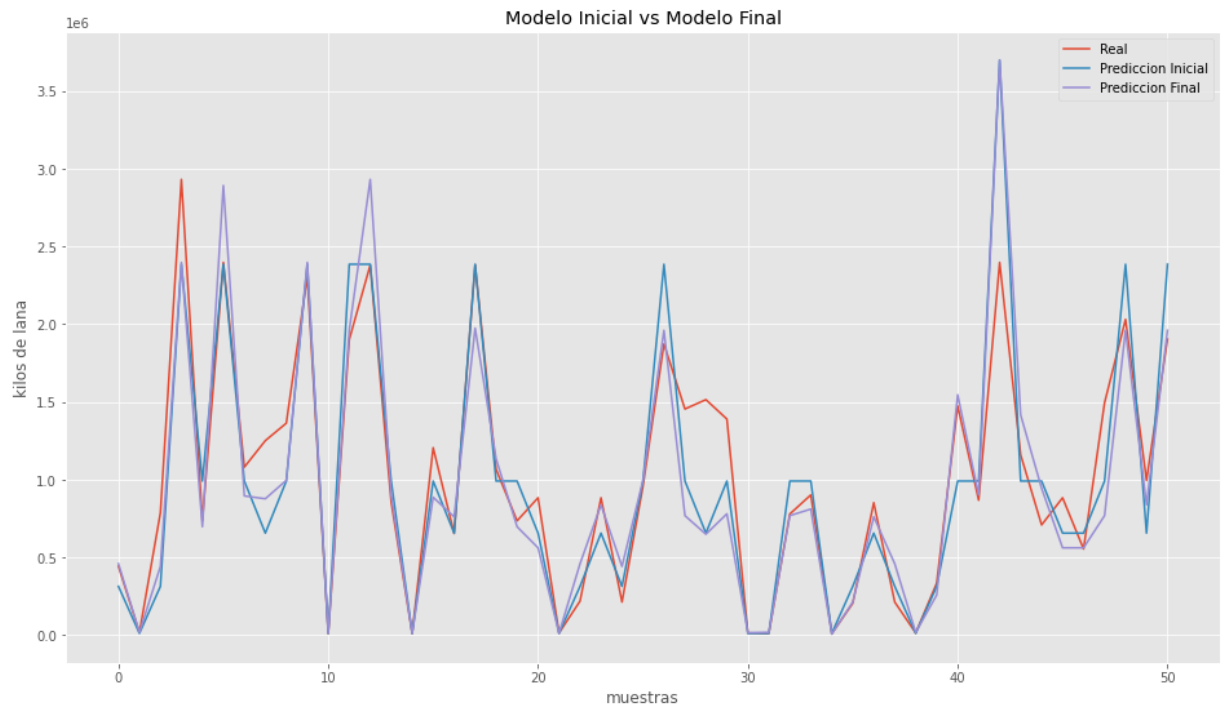
```
Out[34]: 'R2 ajustado: 78.73%'
```

Interpretación: El valor de R2 ajustado de 78.79% para el modelo inicial y 78.73% para el modelo final indica que, teniendo en cuenta las variables explicativas incluidas en el modelo, ambos modelos explican aproximadamente el 78% de la variabilidad en la variable objetivo, es decir, los kilos de lana producidos. Por lo tanto, la bondad de ajuste es del 78%, lo cual es una métrica aceptable.

1.8.2 Grafiquemos los resultados

A continuación, se presentará un gráfico que ayudará a visualizar el rendimiento de los modelos en la predicción de la producción de lana. Para ello, se empleará un gráfico de líneas que permitirá comparar los valores reales de la producción de kilos de lana con los valores predichos por ambos modelos. De esta manera, se podrá tener una mejor comprensión del desempeño de los modelos y de la precisión de las predicciones.

```
In [36]: plt.plot(range(len(y_test)),y_test,label="Real")
plt.plot(range(len(predicciones)),predicciones,label="Prediccion
Inicial")
plt.plot(range(len(predicciones_final)),predicciones_final,label="Pr
Final")
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
plt.xlabel('muestras')
plt.ylabel('kilos de lana')
plt.title("Modelo Inicial vs Modelo Final")
plt.legend()
plt.show()
```



Al observar el gráfico, podemos notar que la línea roja que representa los valores reales y la línea azul que representa la predicción del modelo inicial tienen una tendencia similar, aunque hay algunas desviaciones. Esto también se aplica al modelo final representado por la línea lila. Esta similitud sugiere que los modelos no están sobreajustando los datos de entrenamiento y es más probable que tengan un mejor desempeño al predecir valores para nuevos datos.