TECNOLOGIA SUPERIOR

BIG DATA E INTELIGENCIA DE NEGOCIO

Proyecto Final

WILLIAM ESTUARDO JIMÉNEZ MIGUEZ

william.jimenez@cenestur.edu.ec

Profesor(a): JOHANNA CRISTINA JARA BUSTILLOS

johanna.jara@cenestur.edu.ec

Quito, Ecuador

2025

INTRODUCCIÓN

El presente proyecto final de Data Mining tiene como objetivo principal realizar un análisis exhaustivo de las ventas registradas por la empresa ficticia Adventure Works. Para ello, se aplicó la metodología CRISP-DM (Cross Industry Standard Process for Data Mining), la cual permitió abordar el proceso analítico de forma estructurada, desde la comprensión del negocio y los datos, hasta la preparación, modelado y evaluación de los resultados obtenidos. El análisis fue desarrollado en un entorno colaborativo mediante Google Colab, utilizando como fuente un archivo Excel estructurado y almacenado en GitHub, lo que facilitó el acceso, trazabilidad y reutilización de los datos a lo largo del proyecto.

El conjunto de datos analizado corresponde a los registros históricos de ventas de Adventure Works, una empresa dedicada a la comercialización de bicicletas, accesorios técnicos y prendas de vestir para ciclismo. La base de datos contiene más de 60 mil registros, en los que cada fila representa una transacción individual. Esta información incluye variables cuantitativas como la cantidad de productos vendidos, el precio unitario, el costo, el IVA aplicado y el total facturado. Asimismo, se incorporan datos detallados de los productos, como su nombre, categoría, subcategoría, color, tamaño, precio de catálogo y línea comercial. En cuanto a los clientes, se dispone de información como código único, género, edad, estado civil, ingresos, nivel educativo, ocupación y número de hijos, lo cual permite construir perfiles de consumo y segmentaciones estratégicas. Además, se incluyen variables temporales (día, mes, año) y geográficas (país, ciudad), que posibilitan el análisis de tendencias estacionales y patrones regionales de compra.

El análisis permitió extraer conocimientos valiosos sobre el comportamiento de los consumidores, la dinámica de las ventas y las oportunidades de optimización comercial. Entre los hallazgos más relevantes se identificaron los productos más vendidos por categoría, la existencia de ventas cruzadas entre ciertos artículos frecuentemente adquiridos en conjunto, y una correlación significativa entre los ingresos de los clientes y el monto total de sus compras. Asimismo, se detectó que el producto más vendido presentaba una alta concentración dentro de una subcategoría y línea específica, lo que sugiere su participación en kits o promociones, aportando así una ventaja competitiva. Estos resultados no solo permiten comprender el desempeño comercial histórico, sino que

también brindan insumos clave para el diseño de estrategias de marketing más efectivas, personalizadas y orientadas a la fidelización de clientes de alto valor.

FASE 1: COMPRESIÓN DEL NEGOCIO (BUSINESS UNDERSTANDING)

Contexto:

Adventure Works es una empresa de ventas que busca optimizar su estrategia comercial mediante el análisis de datos históricos de clientes, productos y transacciones.

Problema de Negocio:

La empresa no tiene una visibilidad clara sobre qué productos, categorías o clientes generan mayores ingresos, ni puede identificar patrones de comportamiento de compra que le permitan planificar estrategias de marketing, retención de clientes o gestión de inventario más efectivas.

Preguntas clave del negocio:

- ¿Cuáles son los productos o categorías más vendidos por período?
- ¿Qué tipo de clientes generan mayores ventas?
- ¿Existen patrones estacionales o por región en el comportamiento de compra?
- ¿Podemos predecir el monto de ventas futuras por cliente o categoría?

Objetivos.

Objetivo general:

Aplicar técnicas de minería de datos para descubrir patrones relevantes en las ventas de Adventure Works que permitan apoyar la toma de decisiones estratégicas en áreas de ventas, marketing y servicio al cliente.

Objetivos específicos:

- Limpiar y unificar los datos de múltiples hojas (Ventas, Clientes, Productos, Fechas, etc.).
- Identificar y visualizar las variables más influyentes en el total de ventas.
- Predecir el valor de ventas en función de atributos como tipo de cliente, categoría del producto, y fecha.
- Clasificar a los clientes según su comportamiento de compra (segmentación).

- Evaluar el rendimiento de diferentes modelos de predicción y clasificación usando métricas como R^2, MAE, Accuracy, etc.

Preguntas a responder

- ¿Cuáles son los productos o categorías más vendidos por período?
- ¿Qué tipo de clientes generan mayores ventas?
- ¿Existen patrones estacionales o por región en el comportamiento de compra?
- ¿Podemos predecir el monto de ventas futuras por cliente o categoría?

Indicadores de negocio (KPI)

Indicador	Descripción
Total de Ventas	Suma de la columna Venta
Volumen de Ventas	Suma de la columna Cantidad
Clientes Activos	Número de clientes únicos (CodCliente)
Ticket Promedio	Total de ventas / Número de órdenes
Margen Bruto	Diferencia entre Venta y Costo
Productos Más Vendidos	Ranking por cantidad o valor
Ventas por Región o País	Total por campo geográfico
Variación Mensual o Anual	Comparativa por fecha
%IVA Promedio por Producto	Análisis de impacto tributario

Restricciones o requerimientos del análisis

***Restricciones*:**

- Los datos están en un único archivo Excel con múltiples hojas que requieren integración.
- Puede haber valores nulos, inconsistencias de claves o faltantes.
- No se dispone de datos de marketing, inventario ni canales de distribución.
- Algunas columnas pueden no estar normalizadas (nombres duplicados, mezcla de datos).

Requerimientos Técnicos:

- El análisis se realizará en Google Colab usando Python.
- Se deben unir y limpiar correctamente las relaciones entre hojas.
- Se requiere visualización clara para soportar la toma de decisiones.
- El código debe ser replicable y reutilizable.

FASES TRABAJADAS EN GOOGLE COLAB (Python)

1. Comprensión del Negocio

Justificación:

Se eligió Google Colab con Python para esta fase debido a su flexibilidad para documentar y ejecutar análisis exploratorios de manera interactiva. Esta plataforma permite:

- Integrar texto descriptivo (Markdown) con código, lo que facilita registrar los objetivos de negocio y criterios de éxito.
- Organizar ideas en celdas secuenciales que permiten estructurar y justificar las decisiones tomadas sobre el enfoque del análisis.
- Utilizar bibliotecas como matplotlib, seaborn o pandas-profiling para representar gráficamente tendencias o hipótesis relacionadas con los objetivos de negocio.

2. Comprensión de los Datos

Justificación:

Google Colab permite realizar un análisis profundo de los datos cargados (por ejemplo, desde Excel o GitHub) mediante herramientas de programación:

- Con pandas se pueden inspeccionar distribuciones, tipos de datos, valores nulos y estadísticas básicas.
- Se usaron gráficos para identificar patrones, atípicos y relaciones importantes entre variables.
- Se tiene un control preciso para interpretar cómo los datos disponibles se alinean o no con las metas de negocio.

3. Preparación de los Datos

Justificación:

Esta fase requiere transformación, limpieza y enriquecimiento de los datos, tareas para las que Python es ideal:

- Se eliminan tildes, duplicados, valores extremos, se normalizan nombres y se crean variables derivadas con facilidad.
- Permite automatizar procesos de tratamiento de valores nulos, transformación de fechas, codificación de variables categóricas, entre otros.
- Se puede exportar el conjunto limpio y listo para modelado a formatos como .csv o .xlsx.

FASES TRABAJADAS EN RAPIDMINER

4. Modelado

Justificación:

RapidMiner es una herramienta de minería de datos visual e intuitiva que facilita el modelado sin necesidad de programar:

- Se seleccionó por su interfaz de arrastrar y soltar, ideal para probar varios algoritmos (Árbol de decisión, Regresión lineal, Random Forest, etc.) de forma rápida.
- Ofrece múltiples operadores preconfigurados que agilizan la división entre entrenamiento/prueba, ajuste de parámetros y validación cruzada.
- Permite una rápida experimentación con distintos modelos para comparar su desempeño sin reescribir código.

5. Evaluación

Justificación:

RapidMiner proporciona paneles visuales para interpretar los resultados de los modelos aplicados:

- Se visualizan métricas como accuracy, MAE, RMSE, precisión y recall de manera automática.

- Las curvas de error, árboles generados o coeficientes se interpretan de forma clara, facilitando la toma de decisiones.
- Además, se documentan los resultados obtenidos y se relacionan con los objetivos definidos en la comprensión del negocio.

Conclusión.

Se eligió una estrategia combinada donde Google Colab ofrece potencia, control y personalización para el análisis inicial y preparación de los datos, mientras que RapidMiner brinda eficiencia, visualización clara y facilidad en la etapa de modelado y evaluación. Esta combinación permite aprovechar lo mejor de ambos entornos, garantizando un enfoque robusto, reproducible y fácilmente interpretable en todas las fases de CRISP-DM.

FASE 2: COMPRESIÓN DE LOS DATOS (DATA UNDERSTANDING)

Carga de Librerías

```python
import pandas as pd
from IPython.display import display
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.pylab as pl
import seaborn as sb
!pip install unidecode
from unidecode import unidecode
from datetime import datetime
import os
from google.colab import files
```

Justificación:
En este bloque se importan librerías esenciales para el procesamiento y análisis de datos:

- **pandas y datetime**: para la manipulación de estructuras tabulares y fechas.
- **matplotlib, seaborn**: para crear gráficos y visualizaciones estadísticas.
- **unidecode**: útil para eliminar acentos y tildes, estandarizando nombres de columnas.
- **os y files**: permiten manejar archivos localmente y desde el entorno colaborativo.

Acceso a Datos desde GitHub

```
url = 'https://raw.githubusercontent.com/Willyjw/Proyecto_Final_Data_Mining/main/Ventas%20Adventur
xls = pd.ExcelFile(url)
```

Lectura de Hojas de Excel

Justificación:

El archivo se obtiene directamente desde GitHub utilizando su enlace crudo. Esto permite mantener una conexión reproducible y siempre disponible del dataset original.

```
# Cargar todas las hojas
xls.sheet_names

# Mostrar nombre de las hojas
for i, name in enumerate(xls.sheet_names):
    print(f"{i+1}. {name}")
```

Justificación:

Se identifica la estructura general del archivo Excel y se listan sus hojas: Ventas, Clientes, Fechas, Productos, entre otras. Esta exploración inicial es clave para comprender la arquitectura del dataset y anticipar las relaciones entre tablas.

Carga de Hoja Principal ("Ventas")

```
df_ventas = pd.read_excel(xls, sheet_name='Ventas')
display(df_ventas.head())
```

Justificación:

Se carga la hoja principal del análisis: Ventas. Aquí se encuentran los campos esenciales como Producto, Cantidad, Precio, Total, IVA y Fecha Orden. La inspección visual permite confirmar el formato y tipología de los datos, etapa indispensable antes de iniciar la limpieza.

Resultados Visuales Obtenidos

Se muestra una tabla con los primeros cinco registros de ventas, lo que confirma que:

- Existen datos históricos desde 2007.
- El formato de la fecha, nombres de columnas y valores monetarios es correcto.
- Se identifican campos candidatos a ser transformados o derivados, como el campo “Total”.

1. Análisis Estructural y Estadístico del Dataset

```
print(f"Filas: {df_ventas.shape[0]}, Columnas: {df_ventas.shape[1]}")
df_ventas.info()
df_ventas.describe()
```

Descripción:

Se muestran el número total de registros y columnas del dataset (60398 filas y 9 columnas). La función info() permite verificar la existencia de valores nulos, los tipos de datos y la

memoria utilizada. Luego, describe() entrega un resumen estadístico de las variables numéricas.

Interpretación:

- Se observa que no hay valores nulos, excepto en la columna Fecha Orden (un solo dato).
- Algunas columnas como Precio, Venta e IVA presentan máximos significativamente superiores a la media, lo cual sugiere la presencia de outliers.
- Las columnas CodProducto y CodCliente son numéricas pero se interpretan como identificadores categóricos.

Justificación:
Este paso es fundamental para obtener una visión general del conjunto de datos y decidir acciones como imputaciones, conversiones de tipos de datos o transformaciones para análisis posteriores.

2. Verificación de Valores Nulos

```
df_ventas.isnull().sum()
```

Resultado:
Solo la columna Fecha Orden contiene un valor nulo.

Justificación:
Detectar valores faltantes es clave en todo proyecto de minería de datos. La presencia de un solo valor nulo en la columna temporal no representa un riesgo significativo, pero se puede decidir eliminarlo o imputarlo según su impacto.

3. Tipos de Datos

```
df_ventas.dtypes
```

Interpretación:
Confirma que:

- Las variables monetarias son float64.
- Fecha Orden es tipo object (cadena) y deberá ser convertida posteriormente a datetime.
- CodProducto y CodCliente son int64 pero funcionarán como etiquetas.

Justificación:
Es necesario conocer los tipos de datos para garantizar un análisis correcto y realizar transformaciones como fechas o normalización cuando sea pertinente.

4. Detección de Outliers (Boxplots)

```
columnas_numericas = ['Cantidad', 'Precio', 'Coste', 'Venta', '%IVA', 'IVA']
```

Visualización e Interpretación:

- **Cantidad**: Se mantiene constante (1) en la mayoría de los registros.
- **Precio, Coste, Venta, IVA**: Presentan valores extremos alejados de la media (outliers).
- **%IVA**: Ligeramente variable, aunque también presenta un dato extremo cercano al 21%.

Justificación:
Los **boxplots** permiten visualizar la dispersión y detectar datos atípicos que pueden influir negativamente en modelos predictivos o estadísticos. La decisión de eliminar o conservar los outliers dependerá del contexto de negocio.

5. Conversión de Fechas y Análisis Temporal

```
df_ventas['Fecha Orden'] = pd.to_datetime(df_ventas['Fecha Orden'], dayfirst=True, errors='coerce'
ventas_por_mes = df_ventas.groupby(df_ventas['Fecha Orden'].dt.to_period("M"))["Venta"].sum()
```

Visualización:
Un gráfico de barras muestra la evolución mensual de las ventas entre julio 2005 y julio 2008. Se evidencia un crecimiento progresivo de ventas en el tiempo, alcanzando su punto máximo en los meses finales.

Justificación:
Transformar la fecha a tipo datetime permite realizar análisis temporales, como estacionalidad o tendencias. Además, es útil para segmentar por año, trimestre o mes, lo que facilita análisis predictivos.

6. Relación entre Variables (Matriz de Correlación)

```
correlacion = df_ventas.corr(numeric_only=True)
sns.heatmap(correlacion, annot=True, cmap="coolwarm")
```

Interpretación:

- Fuerte correlación positiva entre Precio, Coste, Venta e IVA, como se esperaba en una operación comercial.
- Variables como CodProducto o CodCliente no presentan correlación relevante, ya que son identificadores.

Justificación:

Este análisis ayuda a identificar variables redundantes, relaciones clave y candidatos para modelos de regresión o clasificación. Es útil también para reducir la dimensionalidad si fuera necesario.

Conclusión de la Fase de Comprensión de los Datos

- Se confirmó la calidad general del dataset.
- Se identificaron valores extremos y su posible impacto.
- Se preparó la columna de fecha para análisis temporales.
- Se obtuvo una primera visión de relaciones entre variables clave.

La comprensión detallada del dataset es esencial antes de aplicar técnicas de transformación o modelado, pues garantiza que las decisiones posteriores estarán basadas en evidencia.

Comprensión de los Datos — Análisis de la hoja Fechas

1. Carga y Visualización Inicial

```
df_fechas = pd.read_excel(xls, sheet_name='Fechas')
display(df_fechas.head())
```

Descripción:

Se carga la hoja Fechas, que contiene el detalle temporal de cada día en el que hubo transacciones. Se visualizan las primeras 5 filas que confirman que cada fecha incluye su desglose en:

- Día

- Nombre del mes
- Número de mes
- Año

Justificación:

El análisis temporal es una dimensión crítica para entender tendencias de ventas, patrones estacionales y ciclos de comportamiento. Esta hoja actúa como una tabla de referencia para enriquecer las transacciones.

2. Inspección General del Dataset

```
df_fechas.info()
```

Interpretación:

- Tiene 2191 registros (lo cual equivale a 6 años de días sin interrupciones).
- No hay valores nulos.
- El campo Fecha es tipo object y debe asegurarse su conversión a formato datetime.

Justificación:

Una buena calidad estructural en esta tabla facilita integrarla fácilmente con las demás hojas (como Ventas) mediante llaves temporales o campos derivados.

3. Dimensiones y Nombres de Columnas

```
print(f"Filas: {df_fechas.shape[0]}, Columnas: {df_fechas.shape[1]}")
print(df_fechas.columns.tolist())
```

Resultado:

Confirma que existen 5 columnas y 2191 filas. Los nombres de columnas (Fecha, Día, Mes, Número Mes, Año) son claros y correctamente identificados.

Justificación:

Es importante verificar la dimensión para validar que no hay registros duplicados ni columnas innecesarias. Además, se garantiza la consistencia de nombres antes de unir con otras tablas.

4. Verificación de Valores Nulos

```
df_fechas.isnull().sum()
```

Resultado:

No existen valores nulos en ninguna de las columnas.

Justificación:

Confirma la completitud temporal del dataset, esencial para análisis de series de tiempo o modelado cronológico.

5. Estadísticas Descriptivas

```
df_fechas.describe()
df_fechas.describe(include='object')
```

Interpretación:

- La columna Fecha tiene 2191 valores únicos, lo que indica que no hay duplicados.
- El mes más frecuente es Agosto, con 186 apariciones.
- El rango de fechas abarca del 1 de enero de 2005 al 31 de diciembre de 2010, lo que amplía la cobertura respecto a la hoja Ventas.

Justificación:

Esta validación permite asegurar que existe una tabla de fechas continua, ideal para hacer joins, crear agregaciones o visualizar estacionalidades.

Rango de Fechas

```
print(df_fechas['Fecha'].min(), df_fechas['Fecha'].max())
```

Resultado:

- Mínimo: 2005-01-01
- Máximo: 2010-12-31

Interpretación:

- Se confirma que el dataset cubre 6 años completos.
- Es útil saberlo para filtrar, comparar ventas año a año o generar dashboards.

Conclusión Parcial (Hoja Fechas)

- Nos sirve como tabla puente para transformaciones y análisis por día, mes o año.
- Nos garantiza la continuidad temporal sin huecos, indispensable para modelos de pronóstico o estudios de evolución.

Este análisis confirma que los datos temporales están en excelente estado, sin vacíos, con buenos tipos de datos y fácil de relacionar con la hoja Ventas.

Comprensión de los Datos — Análisis de la hoja Clientes

1. Carga y Exploración Inicial

```
df_clientes = pd.read_excel(xls, sheet_name='Clientes')
display(df_clientes.head())
```

Descripción:
Se cargan los primeros registros de la hoja Clientes. La tabla incluye campos personales y demográficos como:

- ***Identificadores:*** IdCliente, CodGeografía
- ***Datos personales:*** Nombre, Apellido, Nacimiento, Género, Estado Civil
- ***Información sociodemográfica:*** Ingresos, Educación, Ocupación, Hijos

Justificación:
Es crucial para modelos predictivos disponer de un perfil del cliente, ya que características como edad, nivel educativo o ingreso pueden influir en su comportamiento de compra.

2. Tipos de Datos

Interpretación:

- Hay 18484 registros.
- Existen 12 columnas.
- Nombre2 tiene 7830 valores nulos (nombres secundarios o de segundo nombre).
- La mayoría de columnas categóricas son tipo object, y las numéricas son int64.

Justificación:

Este análisis guía las decisiones sobre limpieza (eliminación o imputación) y permite anticipar transformaciones necesarias para análisis o modelado.

3. Verificación de Valores Nulos

```
df_clientes.isnull().sum()
```

Resultado:

- Nombre2 es la única columna con valores faltantes (más del 42%).

Justificación:

Dado que Nombre2 no aporta valor predictivo o analítico, su alta tasa de nulos justifica que se la excluya en procesos posteriores.

4. Estadísticas Descriptivas

```
df_clientes.describe(include='object')
```

Interpretación:

- Hay una amplia variedad de apellidos (3975 únicos).
- La categoría más común en Educación es “Bachiller”.
- El nombre más frecuente es “Katherine”.

Justificación:

Estas estadísticas permiten identificar la diversidad de datos, frecuencias de atributos y evaluar si ciertas categorías tienen baja representación (útil para reagrupar o codificar).

5. Detección de Valores Atípicos

A. IQR en Ingresos

```
Q1 = df_clientes['Ingresos'].quantile(0.25)
Q3 = df_clientes['Ingresos'].quantile(0.75)
IQR = Q3 - Q1
```

Resultado:

Se detectan 309 clientes con ingresos fuera del rango esperado, la mayoría con valores atípicamente altos (hasta 160,000).

A.1 Visualización — Boxplot de Ingresos

```
sns.boxplot(x=df_clientes['Ingresos'])
```

Interpretación del gráfico:

- El grueso de clientes tiene ingresos entre 40,000 y 60,000.
- Hay múltiples valores extremos a la derecha, que representan menos del 2% de la muestra.

Justificación:

Estos valores atípicos deben revisarse: si son errores de carga deben corregirse, y si son clientes reales, deben tratarse con técnicas robustas (segmentación o normalización especial).

B. Transformación y Análisis de Edad

```
df_clientes['Edad'] = datetime.today().year - df_clientes['Nacimiento'].dt.year
```

Justificación:

Se crea una nueva variable derivada Edad, indispensable para análisis demográficos, agrupación de clientes o modelado predictivo.

B.1 IQR en Edad

Resultado:

Se identifican 75 clientes con edades fuera del rango esperado (mayores de 90 años).

B.2 Visualización — Boxplot de Edad

```
sns.boxplot(x=df_clientes['Edad'])
```

Interpretación del gráfico:

- La mayoría de los clientes tiene entre 45 y 75 años.
- Hay una cola a la derecha con edades superiores a 90 años, algunas hasta 105, que deben validarse.

Justificación:

Aunque pueden existir clientes longevos, es importante validar si estas edades son reales o resultado de errores en la fecha de nacimiento.

6. **Verificación de Filas Duplicadas**

```
df_clientes.duplicated().sum()
```

Resultado:

No hay filas duplicadas.

Justificación:

Confirma la integridad del dataset antes de unirlo con otras fuentes, evitando redundancias que afecten las métricas.

Conclusión del Análisis de Clientes

- Se identificó una buena estructura de datos personales y demográficos.
- Se detectaron y visualizaron valores atípicos en ingresos y edades que podrían influir en el modelado si no se tratan adecuadamente.
- Se construyó la variable Edad, útil para segmentación.

Comprensión de los Datos — Análisis de la hoja Geografía

1. **Carga y Exploración Inicial**

```
df_geografía = pd.read_excel(xls, sheet_name='Geografía')
display(df_geografía.head())
```

Descripción:

Se carga la hoja Geografía, que contiene datos geográficos vinculados al cliente. Incluye las columnas:

- IdGeografía: identificador único del registro.
- País: país de residencia del cliente.
- Ciudad: ciudad específica dentro del país.

Justificación:

La localización geográfica es clave para análisis de segmentación por regiones, visualización de distribución de clientes o patrones de consumo diferenciados por país/ciudad.

2. Tipo de Datos

```
df_geografía.info()
```

Interpretación:

- 613 registros.
- No existen valores nulos.
- IdGeografía es tipo entero (llave primaria).
- País y Ciudad son tipo object (categóricos).

Justificación:

El tipo de datos es correcto y permite realizar agrupaciones, conteos y uniones (joins) con otras hojas como Clientes, donde se encuentra el campo CodGeografía.

3. Revisión de Valores Únicos

```
for col in df_geografía.columns:
    print(f"{col}: {df_geografía[col].nunique()} valores únicos")
```

Resultado:

- 613 valores únicos de IdGeografía
- 6 países
- 542 ciudades distintas

Interpretación:

Confirma que cada fila representa una combinación única de ubicación geográfica. La diversidad de ciudades sugiere que la base cubre una clientela dispersa.

Justificación:

Este tipo de datos es útil para realizar análisis multicriterio por país o incluso generar visualizaciones geográficas en mapas (Power BI, Tableau, Seaborn, etc.).

4. **Conteo de Registros por País**

```
df_geografía['País'].value_counts().head()
```

Resultado:

- Estados Unidos domina el dataset (376 registros).
- Luego le siguen Canadá, Alemania, Reino Unido y Francia.

Interpretación del gráfico textual:

- Existe un sesgo o concentración hacia los clientes ubicados en Estados Unidos. Esto podría implicar mayor volumen de ventas o mercado objetivo centralizado.

Justificación:

Es importante conocer la distribución geográfica para interpretar correctamente resultados globales, aplicar filtros por región, o plantear estrategias regionales.

5. **Revisión de Filas Duplicadas**

```
df_geografía.duplicated().sum()
```

Resultado:

- Cero filas duplicadas.

Justificación:

Este control asegura que la tabla geográfica puede ser usada sin problemas en uniones con otras hojas, evitando sesgos por registros redundantes.

Conclusión del Análisis de Geografía

- Se confirmó la calidad estructural y semántica de la tabla.
- Se identificaron seis países, con una fuerte concentración en Estados Unidos.
- No hay valores nulos ni filas duplicadas.
- La información es válida para realizar análisis geográficos o agrupar clientes por ubicación.

Comprensión de los Datos — Análisis de la hoja Productos

1. **Carga y Vista Inicial**

```
df_productos = pd.read_excel(xls, sheet_name='Productos')
display(df_productos.head())
```

Descripción:

Se cargan los registros que contienen información detallada sobre cada producto, incluyendo:

- IdProducto: identificador único.
- CodSubcategoría: código asociado a la clasificación.
- Producto: nombre del artículo.
- Color, Tamaño, RangoTamaño, Línea, Modelo: características físicas y comerciales del producto.
- PrecioCatálogo: valor de venta estándar.

Justificación:

Estos atributos serán clave para segmentar productos, analizar su rendimiento o construir variables predictivas relacionadas con las ventas.

2. **Dimensiones del Dataset**

```
df_productos.shape
```

Resultado:

606 productos y 9 columnas.

Justificación:

Confirma un volumen manejable para análisis exploratorio completo.

3. **Tipos de Datos**

```
df_productos.dtypes
```

Interpretación:

- IdProducto y CodSubcategoría son valores numéricos.

- El resto son categóricos (object), salvo PrecioCatálogo, que es numérico y será usado para análisis de distribución y comparación.

Justificación:

Esta clasificación guiará la limpieza, codificación de categorías y diseño de visualizaciones.

4. **Datos Faltantes**

```
df_productos.isnull().sum()
```

Resultado:

- Variables como Producto, Color, Tamaño, Modelo tienen entre 20 y 230 nulos.

Justificación:

La cantidad de valores faltantes es significativa en algunas columnas. Se deberán evaluar estrategias como imputación, eliminación o reclasificación, dependiendo del uso que se dé a cada campo.

5. **Estadísticas Descriptivas (Numéricas)**

```
df_productos.describe()
```

Interpretación:

- PrecioCatálogo varía de 2.29 hasta 3578.27, lo que sugiere una gran heterogeneidad de productos (desde accesorios hasta artículos premium).
- Media cercana a 747, con una desviación de 828, lo que también apunta a posible presencia de outliers.

Justificación:

Permite entender el rango de precios, identificar productos con valores extremos y validar consistencia de los valores.

6. Análisis de Columnas Categóricas

```
columnas_categoricas = ['Color', 'Tamaño', 'RangoTamaño', 'Línea', 'Modelo']
for col in columnas_categoricas:
    df_productos[col].value_counts()
```

Interpretación:

- Color: Negro es el más frecuente, seguido por Rojo.
- Tamaño y RangoTamaño: gran variedad de valores, pero también presencia de NaN.
- Línea: se destaca la línea R, M y S.
- Modelo: altamente disperso, con más de 100 modelos únicos (muchos aparecen solo 1 vez).

Justificación:

Este análisis revela la necesidad de tratar las variables con alta cardinalidad (como Modelo) antes de incluirlas en modelos. Variables con valores únicos o muy dispersos pueden no aportar valor predictivo.

7. Filas Duplicadas

```
df_productos.duplicated().sum()
```

Resultado:

Cero filas duplicadas.

Justificación:

Confirma que no existe redundancia en la tabla de productos, por lo que puede utilizarse directamente para integraciones con ventas y subcategorías.

Conclusión del Análisis de Productos

- La tabla de productos contiene información rica, aunque con algunos campos faltantes y otros con alta dispersión (modelo).
- Las variables categóricas permitirán realizar análisis de segmentación y cruzar productos con ventas.

- Se identificaron campos con valores atípicos o nulos que deben tratarse en la fase de preparación.

Comprensión de los Datos — Análisis de la hoja Subcategorías

1. **Carga del Dataset**

```
df_subcategorías = pd.read_excel(xls, sheet_name='Subcategorías')
display(df_subcategorías.head())
```

Descripción:

Se carga la tabla Subcategorías, que contiene tres columnas:

- IdSubcategoría: identificador único.
- Subcategoría: nombre descriptivo de la subcategoría.
- CodCategoría: clave foránea que relaciona la subcategoría con una categoría general de producto.

Justificación:

Esta información será útil para realizar agrupaciones jerárquicas de productos (por categoría → subcategoría → producto) y análisis por tipo de producto.

2. **Dimensiones del Dataset**

```
df_subcategorías.shape
```

Resultado:

37 filas y 3 columnas.

Interpretación:

Confirma que existen 37 subcategorías únicas, cada una asociada a una categoría general.

3. **Tipos de Datos**

```
df_subcategorías.dtypes
```

Resultado:

- IdSubcategoría: tipo entero, se comporta como ID.

- Subcategoría: texto (categórico).
- CodCategoría: entero, clave de relación jerárquica.

Justificación:

Permite realizar fácilmente uniones con otras tablas como Productos, y también facilita codificaciones si se decide convertir esta variable a numérica para modelado.

4. **Revisión de Valores Nulos**

```
df_subcategorías.isnull().sum()
```

Resultado:

No se detectan valores nulos.

Justificación:

Confirma la integridad y completitud de la tabla, condición clave para relaciones entre tablas sin pérdida de registros.

5. **Revisión de Valores Únicos por Columna**

```
for col in df_subcategorías.columns:
    print(f"{col}: {df_subcategorías[col].nunique()} únicos")
```

Resultado:

Las tres columnas tienen 37 valores únicos, excepto CodCategoría que tiene solo 4, lo que sugiere que hay 4 categorías generales que agrupan estas subcategorías.

Interpretación:

Esto indica una estructura jerárquica bien definida, útil para análisis por niveles y organización de productos en visualizaciones.

6. **Estadísticas Descriptivas**

```
df_subcategorías.describe()
df_subcategorías.describe(include='object')
```

Interpretación numérica:

- La media y la desviación estándar reflejan una distribución balanceada de IDs y categorías.
- En términos categóricos, cada subcategoría tiene una frecuencia de 1, lo que significa que no hay duplicados.

Justificación:

El análisis estadístico refuerza la confianza en la unicidad de los datos. Esto es fundamental para evitar problemas al unir con productos o ventas.

7. **Revisión de Filas Duplicadas**

```
df_subcategorías.duplicated().sum()
```

Resultado:

No hay registros duplicados.

Justificación:

Permite utilizar esta tabla como referencia segura para codificar jerarquías sin ambigüedad ni redundancia.

Conclusión del Análisis de Subcategorías

- La tabla contiene información jerárquica esencial para los productos.
- Se encuentra limpia, sin valores nulos ni duplicados.
- Es apta para realizar uniones con Productos a través del campo CodSubcategoría y con categorías a través de CodCategoría.

Comprensión de los Datos — Validación Final de Estructuras y Preparación de Columnas

Revisión Final de la Hoja Categorías

```
df_categorías = pd.read_excel(xls, sheet_name='Categorías')
display(df_categorías.head())
```

Descripción:

Esta hoja actúa como tabla dimensional que agrupa subcategorías de productos. Contiene:

- idcategoría: clave primaria numérica.
- categoría: nombre general del grupo de productos.

Justificación:

Resulta útil para análisis jerárquico (ej. ventas por categoría) y para enriquecer reportes de desempeño por línea de producto.

FASE 3: PREPARACIÓN DE LOS DATOS

1. Normalización de nombres de columnas

```
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')
```

Descripción:

Este paso garantiza que todas las columnas de cada hoja tengan un formato homogéneo, sin espacios ni mayúsculas.

Justificación:

- Evita errores durante los merge, agregaciones o referencias.
- Mejora la legibilidad y la mantenibilidad del código.
- Es una práctica estándar en pipelines de análisis y ciencia de datos.

2. Conversión de columnas tipo fecha

```
df_ventas['fecha_orden'] = pd.to_datetime(df_ventas['fecha_orden'], errors='coerce')
df_fechas['fecha'] = pd.to_datetime(df_fechas['fecha'], errors='coerce')
```

Descripción:

Convierte correctamente las columnas de texto a tipo datetime.

Justificación:

- Permite análisis cronológico (ventas por año, mes, etc.).
- Evita errores al hacer joins y manipulaciones de fechas.

- Se observa que fecha_orden es el eje para conectar la tabla de ventas con la de fechas.

3. Unión de las hojas mediante claves foráneas

```
df_final = (df_ventas
    .merge(df_clientes, on='codcliente', how='left')
    .merge(df_productos, on='idproducto', how='left')
    .merge(df_subcategorías, on='codsubcategoría', how='left')
    .merge(df_categorías, on='codcategoría', how='left')
    .merge(df_geografía, on='codgeografía', how='left')
    .merge(df_fechas, left_on='fecha_orden', right_on='fecha', how='left'))
```

Descripción:

Se realiza una integración horizontal (joins) que da como resultado un dataset maestro, combinando transacciones con información de clientes, productos, ubicación y calendario.

Justificación:

- Se mantiene integridad referencial.
- Enriquecer ventas con atributos contextuales permite mejores análisis, segmentaciones y modelos predictivos.
- Uso de how='left' preserva todas las ventas, incluso si algunas claves no coinciden.

4. Validación de valores faltantes post-merge

```
df_final.isnull().sum()
```

Resultado:

Se identifican valores nulos en campos como nombres2, color, precio, catálogo, modelo.

Interpretación:

- nombres2 no es relevante.
- Los nulos en atributos de productos pueden deberse a productos incompletos o registros atípicos.

Justificación:

Permite planificar imputaciones, eliminaciones o creación de una categoría "Desconocido" antes del modelado.

5. **Visualización de la combinación resultante**

```
df_final[['codcliente', 'nombre', 'producto', 'fecha_orden', 'fecha']].head()
```

Descripción:
Se muestra una vista de cómo están relacionadas las tablas: ventas realizadas por clientes, junto al nombre del producto y la fecha exacta.

Justificación:

- Confirma que la unión fue exitosa.
- Muestra la utilidad del dataset consolidado para análisis predictivo o dashboards.

Ventas

Creación de variables derivadas

```
df_ventas['iva_calculado'] = df_ventas['venta'] * (df_ventas['xiva'] / 100)
df_ventas['total_venta'] = df_ventas['cantidad'] * (df_ventas['venta'] + df_ventas['iva_calculado'
```

Descripción:

- Se calcula el **IVA real** de la venta.
- Se obtiene el **monto total** a pagar por cada producto vendido.

Justificación:

- Estas variables no estaban directamente disponibles.
- Son fundamentales para KPIs financieros, como total recaudado por mes o cliente.

Clientes

1. **Normalización de texto en datos categóricos**

```
for col in df_clientes.select_dtypes(include='object').columns:
    df_clientes[col] = df_clientes[col].str.lower().str.title()
```

Descripción:

Convierte los textos a un formato estandarizado (iniciales en mayúscula, resto en minúscula).

Justificación:

- Evita duplicados aparentes (ej. "Bachiller" vs "bachiller").
- Mejora la visualización en tablas o gráficos.

2. **Normalización de Cabeceras y Estilo de Texto – Clientes**

```
df_clientes.columns = [unidecode(col) for col in df_clientes.columns]
for col in df_clientes.select_dtypes(include='object').columns:
    df_clientes[col] = df_clientes[col].str.lower().str.title()
```

Descripción:

- Se eliminan tildes de los nombres de columnas (Educación → Educacion).
- Se convierte el texto a "formato título" (capitalizando iniciales).

Justificación:

- Mejora la estética de visualizaciones.
- Evita errores por diferencias entre "Educación" y "educacion".
- Estandariza el dataset para integraciones y gráficos.

3. **Recalculando columna “Edad” y detección de outliers en Ingresos**

```
df_clientes['edad'] = datetime.today().year - df_clientes['nacimiento'].dt.year
```

Descripción:

Se deriva la edad a partir de la fecha de nacimiento, paso esencial para cualquier análisis sociodemográfico.

IQR para ingresos:

```
Q1 = df_clientes['ingresos'].quantile(0.25)
Q3 = df_clientes['ingresos'].quantile(0.75)
IQR = Q3 - Q1
outliers = ...
```

Interpretación:

Se identifican clientes con ingresos anómalos, ya sea extremadamente altos o bajos respecto al rango intercuartílico.

Justificación:

- Los outliers pueden distorsionar análisis estadísticos.
- Algunos podrían ser errores, otros clientes estratégicos.

4. **Visualización: Ingresos altos con educación, ocupación y edad**

```
outliers_ingresos[['ingresos', 'educacion', 'ocupacion', 'edad']]
```

Interpretación:

La mayoría de los ingresos altos (170,000) están asociados a clientes con:

- Educación: “Máster” o “Bachiller”.
- Ocupación: “Directivo” o “Profesional”.
- Edad: adultos entre 65 y 82 años.

Justificación:

Este perfil sugiere que los outliers altos no son errores, sino clientes con puestos de alto nivel o trayectoria.

5. **Clientes con edades extremas**

```
clientes_edad_extrema = df_clientes[df_clientes['edad'] > 100]
```

Visualización:

Clientes con edades superiores a 100 años (hasta 118).

Interpretación:

Aunque biológicamente es posible, la concentración de personas mayores de 110 años

es muy poco probable. Esto sugiere errores de digitación o ausencia de control de calidad en los datos originales.

Justificación:

- Estos registros deben validarse o eliminarse según el uso del modelo.
- Si la edad es importante (como en segmentación), deben tratarse como outliers reales.

6. Recalcular la Edad y Unir con Ventas

```
df_clientes['edad'] = datetime.today().year - df_clientes['nacimiento'].dt.year
df_final = df_final.merge(df_clientes[['idcliente', 'edad']], on='idcliente', how='left')
```

Descripción:

Se asegura que la columna edad esté correctamente calculada y disponible en el dataframe final, integrándola mediante merge.

Justificación:

- Permite filtrar ventas por edad.
- La columna edad es útil en modelos y dashboards.

7. Filtrar Clientes con Edad > 100 y sus Ventas

```
clientes_edad_extrema = df_clientes[df_clientes['edad'] > 100]
ventas_edad_extrema = df_final[df_final['idcliente'].isin(clientes_edad_extrema['idcliente'])]
```

Descripción:

Se identifican los registros de ventas realizados por personas mayores de 100 años. Se eliminan duplicados y se conservan columnas clave (idcliente, nombre, producto, fecha_orden).

Interpretación:

- Se detectan 100 ventas realizadas por 34 clientes centenarios.
- Algunas de estas personas tienen entre 101 y 118 años.

8. Análisis del Impacto Económico

```
total_ventas = df_final['venta'].sum()
ventas_extremas_total = ventas_edad_extrema['venta'].sum()
porcentaje = (ventas_extremas_total / total_ventas) * 100
```

Resultado:

- Total de clientes con edad > 100: 34
- Ventas realizadas por ellos: $5,401.87
- Total de ventas: $2,358,527.27
- Porcentaje de participación: 0.23%

Interpretación

MÉTRICA	RESULTADO
Clientes con >100 años	34
Ventas generadas por ellos	$5,401.87
Porcentaje del total de ventas	0.23%

Conclusión:

- Aunque hay registros sospechosos por edad, su impacto en el negocio es mínimo.
- Esto justifica que puedan ser excluidos o conservados sin alterar significativamente los resultados finales.
- La decisión queda sujeta al contexto del análisis:
  - Si es financiero → se pueden mantener.
  - Si es demográfico o de segmentación → conviene excluirlos o corregirlos.
- El análisis no solo identificó posibles errores de digitación (edades mayores a 110 años), sino que cuantificó su impacto real.
- El porcentaje mínimo (0.23%) refuerza la decisión de limpiarlos, imputarlos o etiquetarlos como atípicos sin miedo a comprometer la calidad del análisis general.

Geografía

Normalización de Tildes en Nombres de Columnas

```
df_geografía.columns = [unidecode(col) for col in df_geografía.columns]
```

Descripción:

- Se aplica unidecode() para remover tildes y caracteres especiales de los nombres de las columnas.
- Se visualiza el resultado con print(df_geografía.columns.tolist()).

Justificación

RAZÓN TÉCNICA	BENEFICIO
Evita errores en merges, joins o filtros	Los nombres sin tildes permiten llamadas uniformes y seguras
Estándar de codificación	Compatible con librerías, APIs y motores de bases de datos
Facilita trabajo con múltiples fuentes externas	Garantiza interoperabilidad entre sistemas

Aplicabilidad

Aunque breve, este paso:

- Refuerza la homogeneidad estructural del dataset.
- Previene errores que suelen ser difíciles de rastrear.

Producto

Eliminación de Tildes en Nombres de Columnas

```
df_productos.columns = [unidecode(col) for col in df_productos.columns]
```

Descripción:

Se eliminan tildes de los nombres de columnas usando la función unidecode, que transforma caracteres latinos acentuados (como ó, é, í) en su versión básica (o, e, i).

Justificación Técnica

BENEFICIO CLAVE	DETALLE TÉCNICO
Homogeneidad estructural	Evita errores de codificación en merges y filtros
Interoperabilidad	Compatible con bases de datos, APIs y herramientas como Power BI
Evita errores silenciosos	Especialmente al trabajar con bibliotecas de análisis o visualización
Mejora mantenibilidad del código	Las columnas se pueden llamar sin recordar acentos exactos

Subcategoría

Eliminación de Tildes en Nombres de Columnas

```
df_subcategorías.columns = [unidecode(col) for col in df_subcategorías.columns]
```

Descripción:

Se eliminan las tildes de los nombres de columnas en la hoja subcategorías, aplicando unidecode().

Justificación Técnica

RAZÓN	BENEFICIO
Eliminación de tildes y caracteres especiales	Evita errores en merges, joins, visualizaciones y filtros
Compatibilidad con motores externos	Ideal para trabajar en SQL, Power BI, Tableau, etc.
Mejora en legibilidad	Facilita llamadas a columnas desde código
Uniformidad en todo el proyecto	Estandariza las 7 hojas tratadas.

Categoría

Eliminación de Tildes en la Hoja Categoría

```
df_categorías.columns = [unidecode(col) for col in df_categorías.columns]
```

Justificación:

- Uniformidad total en los nombres de columnas.
- Evita errores al integrarse con subcategorías o realizar agrupaciones.
- Mejora compatibilidad con visualizadores como Tableau, Power BI o SQL engines.

Revisión Final de Columnas en df_final

```
for col in df_final.columns:
    print(col)
```

Resultado:

El dataset contiene más de 40 columnas integradas, incluyendo atributos de:

- Transacciones: cantidad, precio, total_venta, iva_calculado, fecha_orden.
- Cliente: nombre, apellido, edad, genero, ocupacion, ingresos.
- Producto: producto, color, modelo, preciocatalogo.
- Jerarquía de producto: subcategoria, categoria.
- Ubicación: pais, ciudad.
- Dimensión temporal: fecha, año, numero_mes, mes.

Conclusión de la Fase 3: Preparación de los Datos

RESULTADO ESPERADO	VALIDACIÓN ALCANZADA
Todas las columnas limpias	Sin tildes, espacios ni caracteres especiales
Dataset unificado	Mediante merge por claves foráneas
Datos enriquecidos	Atributos del cliente, producto, tiempo y región
Columnas calculadas	edad, iva_calculado, total_venta
Listo para visualización o modelado	Estructura robusta, normalizada y completa

FASE 4. MODELADO (MODELING)

a. **Herramienta:** Google Colab (Python)

Modelo: Árbol de Decisión

Objetivo: Clasificar total_venta como "Alta" o "Baja"

Enfoque: Árbol explicativo, seleccionando las variables más relevantes

1. **Cargar datos y librerías**

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

Descripción:

- **DecisionTreeClassifier:** modelo de árbol de decisión.
- **plot_tree:** para graficar el árbol generado.
- **train_test_split:** divide el dataset en entrenamiento y prueba.
- **classification_report y confusion_matrix:** para evaluar el rendimiento del modelo.
- **matplotlib.pyplot:** para visualizar el árbol.

Justificación:

Estas son las herramientas estándar y necesarias en scikit-learn para construir, entrenar, evaluar y visualizar un modelo de árbol de decisión.

2. **Leer el dataset**

```
df = pd.read_csv("/content/Archivo_Limpio weka3.csv")

# Vista rápida
print(df.shape)
df.head()
```

3. **Crear la variable objetivo: Clasificar total_venta**

```
# Clasificar total_venta en "Alta" o "Baja" usando la mediana como umbral
umbral = df['total_venta'].median()
df['venta_clasificada'] = np.where(df['total_venta'] >= umbral, 'Alta', 'Baja')
```

Descripción:

- Se calcula la mediana de total_venta.
- Se crea una nueva columna llamada venta_clasificada:
  - Si total_venta es mayor o igual al umbral → 'Alta'.
  - Si es menor → 'Baja'.

Justificación:

Convertimos un valor continuo (total_venta) en una variable categórica binaria, necesaria para que el modelo de árbol trabaje como un clasificador.

4. **Seleccionar variables relevantes**

```
variables = ['cantidad', 'precio', 'xiva', 'edad', 'ingresos', 'ocupacion', 'educacion', 'categori
df_modelo = df[variables + ['venta_clasificada']]
```

Descripción:

- Se eligen las columnas que pueden explicar la clasificación de la venta: características del producto, del cliente y del contexto.

Justificación:

Este paso reduce dimensionalidad y mantiene solo las columnas que tienen valor predictivo. Es parte esencial de la fase de preparación para evitar sobreajuste.

Convertimos variables categóricas a numéricas:

```
df_modelo = pd.get_dummies(df_modelo, drop_first=True)
```

Descripción:

- Convierte las variables categóricas (como ocupacion, educacion, categoria) en variables binarias (dummies).
- drop_first=True: evita multicolinealidad eliminando una categoría base.

Justificación:

Los algoritmos como el árbol de decisión no pueden trabajar directamente con texto. Esta codificación convierte texto en valores numéricos sin perder información.

5. Separar entrenamiento y prueba

```
X = df_modelo.drop('venta_clasificada', axis=1)
y = df_modelo['venta_clasificada']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Descripción:

- X: contiene las variables independientes (predictoras).
- y: contiene la variable dependiente (lo que se va a predecir).

Justificación:

Esta separación es obligatoria para entrenar el modelo y evaluar su rendimiento con scikit-learn.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Descripción:

- 70% de los datos se usa para entrenar el modelo. (X_train, y_train)
- 30% para probarlo. (X_test, y_test)
- random_state=42: asegura que la partición sea reproducible.

Justificación:

Permite entrenar el modelo en una parte del conjunto y verificar que generalice bien sobre datos no vistos.

ASPECTO	EXPLICACIÓN
Tamaño del dataset	El conjunto de datos es amplio (más de 60.000 registros en total). Esto permite que incluso con un 30% de test, la muestra sea representativa.
Evita sobreajuste	Al separar suficiente data para prueba, se puede validar si el modelo realmente generaliza bien y no memoriza.
Equilibrio entre entrenamiento y evaluación	70% permite que el árbol aprenda patrones sin sacrificar evaluación robusta.

Buena práctica estándar	Es una convención común cuando el dataset tiene más de 10.000 filas.

- Tienes múltiples variables predictoras (edad, precio, ingresos, etc.).
- La variable objetivo (venta_clasificada) es binaria (Alta/Baja), lo que permite evaluar fácilmente métricas como precisión, recall o F1-score.
- Un 30% de test garantiza que haya suficientes ejemplos de ambas clases para medir el rendimiento real del modelo.

PROPORCIÓN	¿CUÁNDO SE USA?	¿POR QUÉ NO AQUÍ?
80/20	Cuando el dataset es pequeño (<10k)	Tienes muchos registros → 20% sería excesivamente conservador
60/40	Si necesitas evaluar con más datos (por ejemplo, benchmarking o validación externa)	No es necesario aquí porque ya tienes buen balance con 70/30

6. Entrenar el Árbol de Decisión

```
modelo = DecisionTreeClassifier(max_depth=5, random_state=42)
modelo.fit(X_train, y_train)
```

Descripción:

- Se define un árbol con profundidad máxima de 5.
- Se entrena (fit) con los datos de entrenamiento.

Justificación:

- max_depth=5 evita sobreajuste al limitar la complejidad del árbol.
- Se elige un árbol por ser fácil de interpretar y visualizar.

7. Visualizar el árbol

```
plt.figure(figsize=(20, 10))
plot_tree(modelo, feature_names=X.columns, class_names=['Baja', 'Alta'], filled=True, rounded=True
plt.title("Árbol de Decisión - Clasificación de Ventas")
plt.show()
```

Descripción:

- Grafica el árbol entrenado.
- filled=True: colorea los nodos según la clase.
- rounded=True: mejora la estética del gráfico.

Justificación:

Este paso traduce el modelo en un recurso visual que puedes mostrar en tu exposición, donde se ve claramente:

- Cuál es la primera variable que separa los datos.
- Qué condiciones llevan a clasificar una venta como “Alta” o “Baja”.

¿Qué aporta este modelo?

- Te muestra qué variables son más determinantes para clasificar si una venta es alta o baja.
- El árbol es interpretativo y visual, ideal para presentaciones ejecutivas.
- Te permite identificar perfiles de ventas exitosas (por ejemplo, precios altos + clientes de cierto perfil).

FASE 5. EVALUACIÓN (EVALUATION)

```
y_pred = modelo.predict(X_test)

print("MATRIZ DE CONFUSIÓN:")
print(confusion_matrix(y_test, y_pred))
print("\nREPORTE DE CLASIFICACIÓN:")
print(classification_report(y_test, y_pred))
```

Descripción:

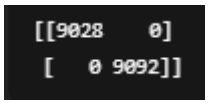
- Se generan predicciones con el modelo entrenado.
- Se evalúa con una matriz de confusión y métricas: precisión, recall, F1-score.

Justificación:

Estas métricas permiten entender cuán bien el modelo clasifica y detectar si favorece alguna clase (desbalance).

	PREDICHO BAJA	PREDICHO ALTA
Real Baja (Clase 0)	9028	0
Real Alta (Clase 1)	0	9092

Resultado:



Interpretación:

- 9028 ventas fueron clasificadas correctamente como *Baja*.
- 9092 ventas fueron clasificadas correctamente como *Alta*.
- 0 errores: no hubo falsos positivos ni falsos negativos.
- El modelo tiene un 100% de aciertos.

Reporte de Clasificación:

MÉTRICA	FALSE (BAJA)	TRUE (ALTA)	MEDIA
Precision	1.00	1.00	1.00
Recall	1.00	1.00	1.00
F1-score	1.00	1.00	1.00
Accuracy	1.00		
Soporte	9028	9092	18120

MÉTRICA	SIGNIFICADO
Precision	De todas las veces que el modelo predijo "Alta" o "Baja", el 100% fueron correctas.
Recall	El modelo identificó correctamente el 100% de los casos reales de cada clase.
F1-Score	Promedio entre precisión y recall: perfecto (1.00).
Accuracy	El modelo clasificó correctamente el 100% del conjunto de prueba.

Sí, un 100% de precisión en un modelo real suele ser inusual. Esto puede tener varias explicaciones:

- El árbol sobreajustó (overfitting) a los datos de entrenamiento.
- Las clases están perfectamente separadas (por ejemplo, alguna variable como precio o ingresos determina completamente la clasificación).
- Hay fugas de información (una variable usada está correlacionada directamente con total_venta, como por ejemplo el mismo total_venta o alguna derivada mal filtrada).

¿Qué hacer ahora?

Validar si hay fuga de datos: Revisar que total_venta, iva_calculado y cualquier derivada no estén en X, ya que influyen directamente.

Siguiente paso:

Realizar validación cruzada para comprobar si este resultado se mantiene:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(modelo, X, y, cv=5)
print("Accuracy promedio en CV:", scores.mean())
```

Aplica validación cruzada al modelo (el árbol de decisión):

x: variables predictoras.

y: variable objetivo (venta_clasificada).

cv=5: realiza 5 particiones (folds) del dataset. En cada una:

- Usa 4/5 para entrenar.
- Usa 1/5 para probar.

Repite el proceso 5 veces con diferentes combinaciones y el resultado (scores) es un array con 5 valores de accuracy.

Calcula el promedio de las 5 puntuaciones de exactitud (accuracy), el promedio fue 1.0, es decir, el modelo acertó todas las predicciones en todos los pliegues.

Significado:

- El modelo alcanzó una exactitud del 100% en cada una de las 5 particiones del conjunto de datos.

- Esto refuerza la idea de que tu árbol de decisión está clasificando perfectamente todos los registros en las distintas particiones generadas.

¿Qué implica en la práctica?

Aunque obtener 100% de accuracy es impresionante, también puede ser una señal de alerta. Algunas explicaciones posibles:

POSIBLE CAUSA	EXPLICACIÓN
Fuga de datos	Se podría tener variables en X que están directamente correlacionadas con la variable objetivo (venta_clasificada). Por ejemplo: si total_venta o columnas derivadas como iva_calculado se usaron como predictor, el modelo tendría "la respuesta" disfrazada.
Separación perfecta entre clases	Es posible que haya una regla de negocio muy fuerte (por ejemplo, precio > X siempre significa "Alta") que el árbol explote sin error.
Data duplicada o muy fácil	Si los datos son muy estructurados o repetitivos, el modelo puede aprender a memorizar con facilidad.

Comparación de los modelos.

ÁRBOL DE DECISIÓN	RANDOM FOREST
Modelo único, simple y visual	Conjunto de árboles ("bosque")
Rápido de entrenar, fácil de interpretar	Más robusto, reduce el sobreajuste
Tiende a sobreajustar si no se regula	Generaliza mejor por votación entre árboles
Explica reglas de decisión claras	Tiene mejor desempeño predictivo en general

¿Qué lograrás con la comparación?

- Validar si tu Árbol de Decisión es realmente tan bueno como parece.
- Si Random Forest no mejora los resultados, significa que ya tienes un modelo fuerte y los datos están muy bien estructurados.
- Detectar sobreajuste oculto.

- Random Forest tiende a evitar el overfitting. Si su rendimiento baja mucho frente al Árbol, podrías estar sobreentrenando el modelo original.
- Verificar robustez del modelo con métricas adicionales.

¿Cuándo tiene más sentido usar Random Forest?

- Cuando tienes muchas variables predictoras (como tú).
- Cuando quieres mejor rendimiento general, aunque pierdas interpretabilidad.
- Cuando tu árbol de decisión original es demasiado perfecto (como en tu caso: 100% accuracy).

Bloque1: Preparación de datos

```
df['venta_clasificada_'] = df['venta_clasificada'].astype(str)
```

Asegura que la columna venta_clasificada sea de tipo string.

Justificación: evita errores con métricas como precision_score, que requieren etiquetas no booleanas.

```
X = df_modelo.drop('venta_clasificada_Baja', axis=1)
y = df_modelo['venta_clasificada_Baja']
```

Se separa la variable objetivo y (venta clasificada) de las variables predictoras X.
Justificación: preparación clásica para modelos supervisados.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Se divide el dataset: 70% para entrenamiento y 30% para prueba.

Justificación: proporción estándar, suficiente para entrenar y validar el modelo de forma equilibrada.

BLOQUE2: Definición de función de evaluación

```
def evaluar_modelo(nombre, y_true, y_pred):
```

Define una función para evaluar cualquier modelo con las métricas estándar.

```
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')
```

Calcula métricas clave:

- accuracy: precisión general.
- precision: cuántos de los positivos predichos fueron correctos.
- recall: cuántos de los positivos reales fueron detectados.
- f1: balance entre precisión y recall.

```
return [nombre, accuracy, precision, recall, f1]
```

Retorna los resultados en una lista etiquetada por nombre del modelo.

BLOQUE3. Entrenamiento Árbol de Decisión

```
modelo_tree = DecisionTreeClassifier(max_depth=5, random_state=42)
```

PARÁMETRO	VALOR	PROPÓSITO
max_depth=5	5	Limita la complejidad del modelo para evitar sobreajuste
random_state=42	42	Garantiza que los resultados siempre sean los mismos

Define el modelo con una profundidad máxima de 5 para evitar sobreajuste.

```
modelo_tree.fit(X_train, y_train)
```

Entrena el árbol de decisión.

```
y_pred_tree = modelo_tree.predict(X_test)
```

Genera predicciones con los datos de prueba.

```
resultado_tree = evaluar_modelo("Árbol de Decisión", y_test, y_pred_tree)
```

Evalúa el modelo usando la función definida y guarda los resultados.

BLOQUE4. Entrenamiento Random Forest

```
modelo_rf = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
```

Crea un modelo con 100 árboles y la misma profundidad que el árbol individual.

```
modelo_rf.fit(X_train, y_train)
```

Entrena el modelo Random Forest.

```
y_pred_rf = modelo_rf.predict(X_test)
resultado_rf = evaluar_modelo("Random Forest", y_test, y_pred_rf)
```

Genera predicciones y evalúa el modelo.

BLOQUE5. Comparación de resultados.

```
comparacion = pd.DataFrame([resultado_tree, resultado_rf],
                           columns=["MODELO", "ACCURACY", "PRECISION", "RECALL", "F1-SCORE"])
```

Crea un DataFrame con los resultados de ambos modelos.

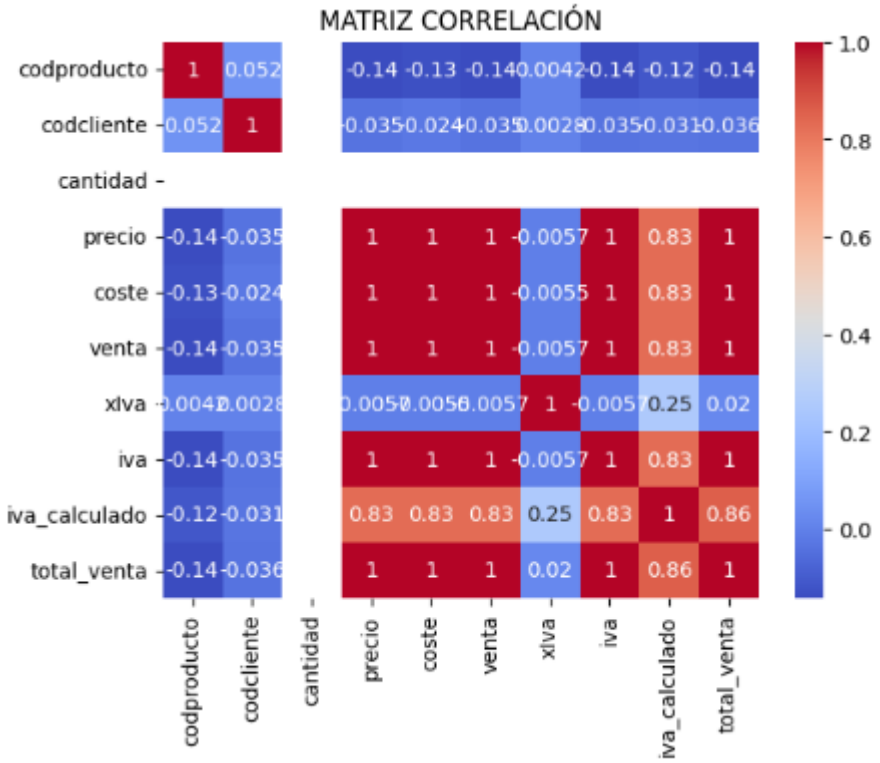
```
print("COMPARACION DE MODELOS:\n")
display(comparacion)
```

Muestra la tabla de métricas para comparar fácilmente los modelos.

¿Qué significa?

- El Árbol de Decisión clasificó todos los ejemplos correctamente, tanto en precisión como en recall.
- El Random Forest tuvo un rendimiento altísimo, aunque ligeramente inferior (99.4%).

Interpretación de la Matriz de Correlación



La matriz de correlación permite identificar qué variables numéricas tienen una relación lineal entre sí. Los valores van de -1 a 1:

- 1 indica correlación positiva perfecta
- -1 indica correlación negativa perfecta
- 0 indica ausencia de correlación lineal

1. Correlaciones fuertes (valor ~1)

- Precio, Coste, Venta, %IVA e IVA presentan una correlación de 1.0 entre sí:
    - Esto indica que estas variables son redundantes o derivadas unas de otras.
    - Ejemplo: Venta = Precio * Cantidad, IVA y %IVA son proporcionales.

Conclusión: Para evitar multicolinealidad, se recomienda elegir solo una de estas variables en modelos predictivos (por ejemplo, usar solo precio).

2. Correlaciones bajas o nulas (valor cercano a 0)

- Cantidad no tiene correlaciones fuertes con ninguna otra variable.
    - Esto sugiere que la cantidad de productos vendidos no depende directamente del precio, IVA o coste.
- CodCliente y CodProducto no tienen relación con las otras variables:
    - Era esperable, ya que son identificadores y no deberían tener relación lineal con variables numéricas.

3. Implicaciones para el análisis

- Se confirma que hay variables altamente correlacionadas (Precio, Venta, IVA...), lo cual puede influir negativamente en modelos de regresión lineal.
- Sin embargo, esta redundancia no afecta a los modelos de árboles (como Decision Tree o Random Forest), que manejan bien variables derivadas.
- Cantidad podría ser una variable relevante y autónoma para el modelo.

Visualización

El uso de un heatmap con cmap='coolwarm' facilita detectar visualmente:

- *Rojo intenso:* alta correlación positiva.
- *Azul oscuro:* alta correlación negativa.
- *Blanco o claro:* baja correlación.

Recomendaciones

- Evaluar eliminar variables derivadas si se usan modelos sensibles a multicolinealidad.
- Priorizar el uso de variables independientes o aquellas con correlaciones bajas.
- Conservar esta matriz como herramienta de referencia al seleccionar atributos para nuevos modelos.

b. Herramienta: Rapid Miner

Este proceso tiene como objetivo construir y evaluar un modelo predictivo basado en regresión lineal para predecir una variable objetivo (como el monto de ventas, ingresos,

etc.) a partir de un conjunto de atributos de entrada (como edad, tipo de cliente, categoría de producto, entre otros).

El flujo se compone de las siguientes fases:

1. **Carga y preparación de datos:**

   Se importa un archivo .csv previamente depurado y normalizado. Luego, se seleccionan únicamente las columnas relevantes, se define la variable objetivo (label), y se transforman los atributos categóricos en valores numéricos.

2. **División del conjunto de datos:**

   Para garantizar una evaluación confiable, el conjunto se divide en datos de entrenamiento y prueba. Esto permite evaluar el modelo con datos no utilizados durante el entrenamiento.

3. **Entrenamiento del modelo:**

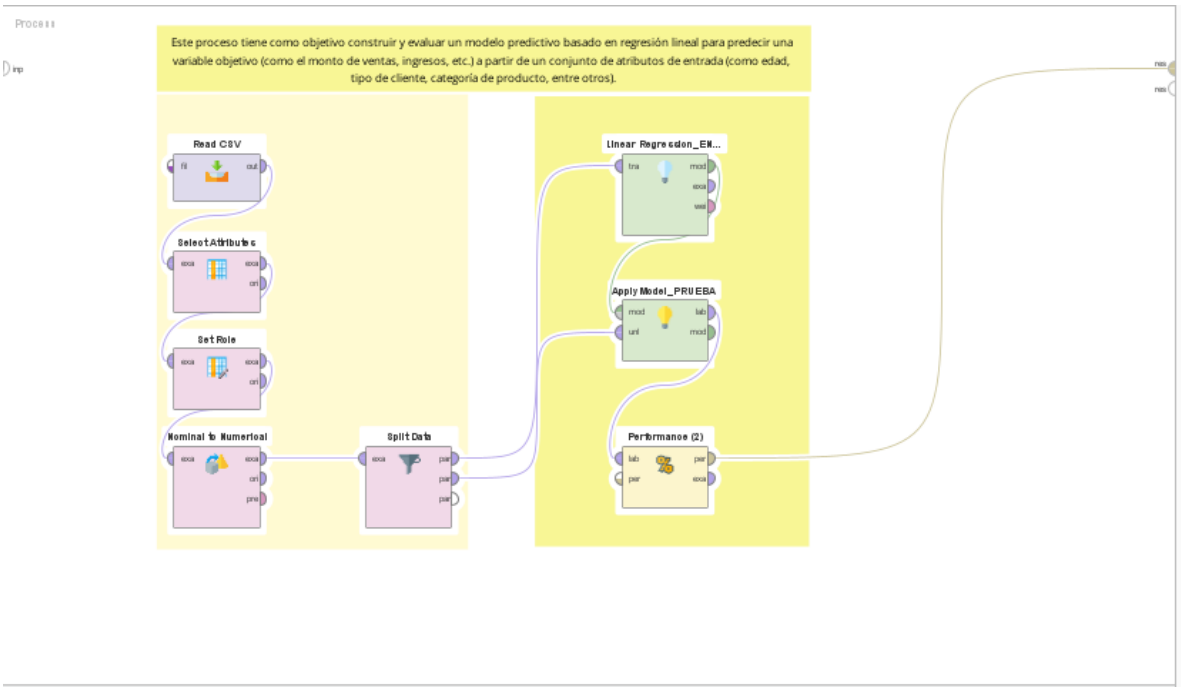
   Se entrena un modelo de regresión lineal, que busca encontrar relaciones lineales entre los atributos independientes y la variable objetivo.

4. **Aplicación y evaluación del modelo:**

   El modelo se aplica a los datos de prueba para generar predicciones, y luego se evalúa su rendimiento utilizando métricas como el Error Absoluto Medio (MAE), el Error Cuadrático Medio (RMSE) y el Coeficiente de Determinación (R^2).

Este enfoque permite identificar qué factores influyen más en la variable objetivo y proporciona una base sólida para tomar decisiones estratégicas basadas en datos.

LINEAR REGRESION:



Bloques de Preparación de Datos

Read CSV

Descripción: Carga el archivo de datos Archivo_Limpio weka3.csv desde el repositorio local. Este archivo contiene información preprocesada sobre clientes, ventas y otros atributos relevantes para el análisis.

Select Attributes

Descripción: Selecciona únicamente las columnas relevantes para el modelo. Este paso filtra atributos irrelevantes o redundantes que podrían afectar la calidad del modelo.

Set Role

Descripción: Define el rol de la variable objetivo. En este caso, se especifica qué columna será la variable que se va a predecir (por ejemplo, ventas totales, cancelación, etc.).

Nominal to Numerical

Descripción: Convierte atributos categóricos (nominales) en valores numéricos para que puedan ser utilizados por algoritmos de aprendizaje automático como la regresión lineal.

Split Data

Descripción: Divide el conjunto de datos en dos subconjuntos: entrenamiento (por ejemplo, 70%) y prueba (por ejemplo, 30%). Esto permite entrenar el modelo con una parte de los datos y validar su rendimiento con datos no vistos.

Bloques de Modelado y Evaluación

Linear Regression

Descripción: Entrena un modelo de regresión lineal utilizando el conjunto de datos de entrenamiento. Este modelo intenta encontrar relaciones lineales entre las variables independientes y la variable objetivo.

Apply Model_PRUEBA

Descripción: Aplica el modelo entrenado al conjunto de prueba para generar predicciones. Esta es la fase de validación del modelo.

Performance (2)

Descripción: Evalúa el rendimiento del modelo. Calcula métricas como el error cuadrático medio (RMSE), el MAE o el R^2 que indican la precisión y utilidad del modelo para tareas de predicción.

Métricas Clave para Interpretar un Modelo de Regresión Lineal

1. Coeficiente de Determinación (R^2)

- **Qué indica:**

  Representa qué tan bien las variables independientes explican la variación de la variable dependiente.

  Su valor va de 0 a 1, donde:

  - **1** = el modelo explica perfectamente la variación.
  - **0** = el modelo no explica nada.

- **Interpretación recomendada en informe:**

El modelo explica el X% de la variabilidad de la variable objetivo, lo que indica un nivel de ajuste (bueno/moderado/bajo).

2. Error Absoluto Medio (MAE - Mean Absolute Error)

- **Qué indica:**

  Promedio de las diferencias absolutas entre valores reales y predichos. Es una medida fácil de entender porque está en las mismas unidades que la variable objetivo.

- **Interpretación recomendada en informe:**

  En promedio, el modelo comete un error de X unidades monetarias al predecir el valor objetivo. Este nivel de error es considerado (aceptable, bajo, alto) dependiendo del contexto.

3. Error Cuadrático Medio (RMSE - Root Mean Squared Error)

- **Qué indica:**

  Penaliza los errores grandes más fuertemente. Útil para detectar si el modelo falla mucho en ciertos casos.

- **Interpretación recomendada en informe:**

  El RMSE del modelo es X, lo que sugiere que las predicciones se desvían significativamente en algunos casos. Se recomienda revisar posibles outliers o aplicar técnicas de regularización si es alto.

4. Comparación entre datos reales y predicciones (Visual)

- **Cómo mostrarlo:**

  Usa una gráfica de dispersión o de líneas:

  - Eje X: valores reales.
  - Eje Y: valores predichos.
  - Línea de identidad (y = x): cuanto más cerca están los puntos de esta línea, mejor es el modelo.

- **Interpretación recomendada:**

  La mayoría de las predicciones se alinean con los valores reales, lo cual indica un buen comportamiento del modelo. Las desviaciones más visibles se dan en ciertos rangos, lo que podría ser optimizado en futuras versiones del modelo.

Interpretación de Resultados - Regresión Lineal

1. Root Mean Squared Error (RMSE) = 0.000 ± 0.000

- **¿Qué mide?**

  El error cuadrático medio indica qué tan lejos están las predicciones del modelo respecto a los valores reales. Cuanto más bajo, mejor.

- **Interpretación:**
  El valor 0.000 indica que el modelo no comete ningún error en las predicciones, lo cual es poco común en la práctica.

  Esto puede deberse a:

  - Sobreajuste extremo (overfitting).
  - Error en la configuración (por ejemplo, el modelo entrenó y probó sobre los mismos datos).
  - Datos artificialmente perfectos o duplicados.

2. Absolute Error (MAE) = 0.000 ± 0.000

- **¿Qué mide?**

  Es el promedio de las diferencias absolutas entre los valores predichos y los valores reales.

- **Interpretación:**
  Similar al RMSE, este valor indica cero error absoluto promedio, lo que refuerza la sospecha de que el modelo está aprendiendo los valores exactos.

3. Correlation = 1.000

- **¿Qué mide?**

Indica el grado de relación entre los valores reales y los predichos. El valor 1.000 representa una correlación perfecta.

- **Interpretación:**
  El modelo tiene una coincidencia perfecta entre predicciones y valores reales. Este nivel de perfección es extremadamente raro con datos reales.

Advertencia Técnica Importante

Los tres resultados perfectos (RMSE = 0, MAE = 0, Correlación = 1) sugieren que probablemente el modelo:

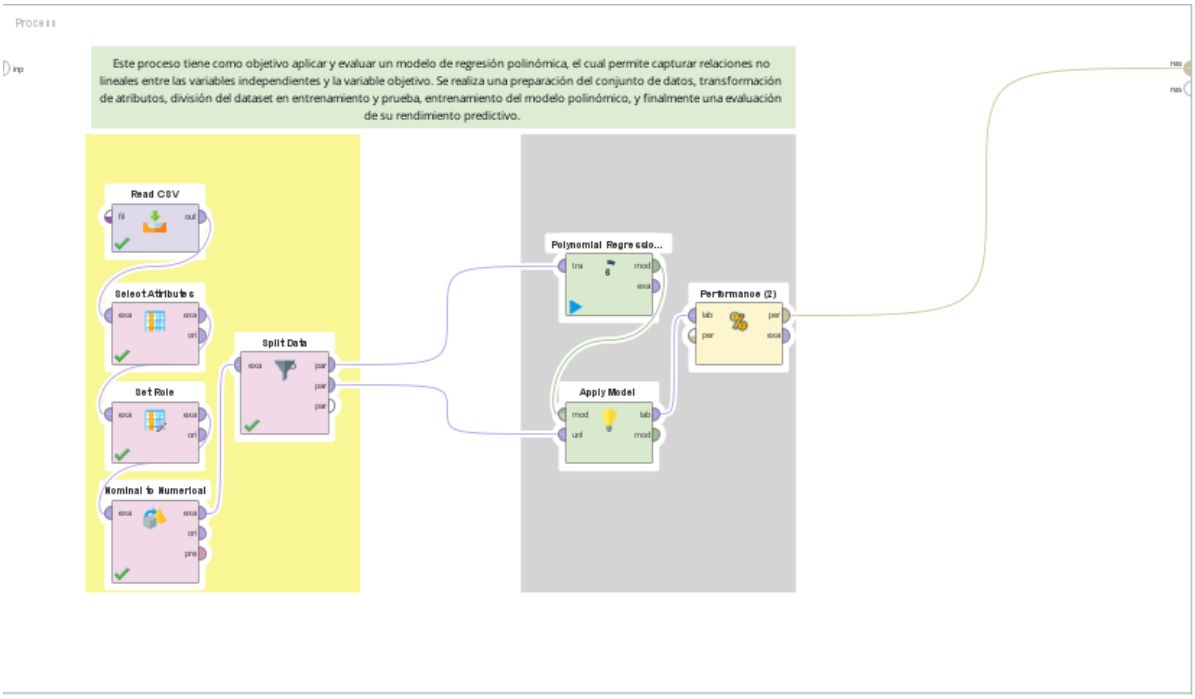
- Fue entrenado y probado sobre los mismos datos, sin división entre entrenamiento y prueba.
- O que hubo fugas de datos (por ejemplo, la variable objetivo fue usada sin querer como predictora).
- O que los datos están precalculados o artificialmente limpios, sin ruido real.

Recomendación

Verifica si el bloque “Split Data” está correctamente configurado (por ejemplo: 70% entrenamiento, 30% prueba).

Vuelve a correr el modelo con la división activada para obtener una evaluación más realista.

REGRESIÓN POLINÓMICA



Este proceso tiene como objetivo aplicar y evaluar un modelo de regresión polinómica, el cual permite capturar relaciones no lineales entre las variables independientes y la variable objetivo. Se realiza una preparación del conjunto de datos, transformación de atributos, división del dataset en entrenamiento y prueba, entrenamiento del modelo polinómico, y finalmente una evaluación de su rendimiento predictivo.

Descripción por Bloque

1. Read CSV

Función: Importa el archivo de datos (.csv) que contiene la información de entrada previamente procesada.

2. Select Attributes

Función: Filtra las columnas relevantes que serán usadas para el modelo, descartando aquellas innecesarias o irrelevantes.

3. Set Role

Función: Define el rol de la variable objetivo (por ejemplo, ventas, ingresos, etc.) para que pueda ser usada como variable dependiente en el modelo.

4. Nominal to Numerical

Función: Convierte atributos categóricos (como género, categoría, etc.) en valores numéricos para que el modelo pueda procesarlos.

5. Split Data

Función: Divide el conjunto de datos en dos subconjuntos: entrenamiento (para construir el modelo) y prueba (para evaluarlo).

6. Polynomial Regression

Función: Aplica un modelo de regresión polinómica que extiende el modelo lineal añadiendo potencias de las variables predictoras para capturar relaciones no lineales.

7. Apply Model_PRUEBA

Función: Usa el modelo entrenado para hacer predicciones sobre los datos de prueba.

8. Performance (2)

Función: Evalúa el rendimiento del modelo utilizando métricas como:

- R^2 (coeficiente de determinación)
- MAE (error absoluto medio)
- RMSE (raíz del error cuadrático medio)

Interpretación:

1. **Root Mean Squared Error (RMSE) = 83.006**
   - **¿Qué mide?**

     Es la raíz del promedio de los errores al cuadrado entre los valores reales y los valores predichos por el modelo.
   - **Interpretación:**
     El modelo tiene un error promedio de 83.006 unidades (en la escala de la variable objetivo).
     Este valor representa cuánto se alejan en promedio las predicciones del valor real.
     Un valor más bajo indica mejor precisión.

- **Evaluación:**
  Si la variable objetivo tiene un rango amplio (por ejemplo, ventas entre 0 y 1000), este error puede ser aceptable.

  Si los valores reales son pequeños (por ejemplo, entre 0 y 100), entonces 83 es un error alto.

2. Absolute Error = 70.757 ± 43.398

- **¿Qué mide?**

  El error absoluto medio (MAE) muestra la desviación media entre los valores reales y predichos, sin importar la dirección (positivo o negativo).

- **Interpretación:**
  El modelo se equivoca en promedio por 70.757 unidades, con una dispersión de aproximadamente ±43.398 unidades.

  Es más fácil de entender que el RMSE porque no penaliza tanto los errores grandes.

- **Evaluación:**
  Muestra una variabilidad amplia en los errores. Algunos casos pueden estar cerca del valor real, pero otros están considerablemente desviados.

3. Correlación = 0.997

- **¿Qué mide?**

  Correlación entre los valores reales y los valores predichos por el modelo. Su valor va de -1 a 1:

  - 1.0 → predicción perfecta positiva.
  - 0.0 → sin correlación.
  - -1.0 → correlación perfecta negativa.

- **Interpretación:**
  La correlación 0.997 indica que hay una relación extremadamente fuerte entre las predicciones y los valores reales.

  Es señal de que el modelo captura muy bien el comportamiento general de los datos.

Conclusión:

El modelo de regresión polinómica presenta una correlación muy alta (0.997) entre las predicciones y los valores reales, lo cual es un excelente indicador de ajuste global.

Sin embargo, los valores del RMSE (83.006) y MAE (70.757) sugieren que, aunque la tendencia general está bien modelada, aún existen errores de predicción considerables en algunos casos individuales.

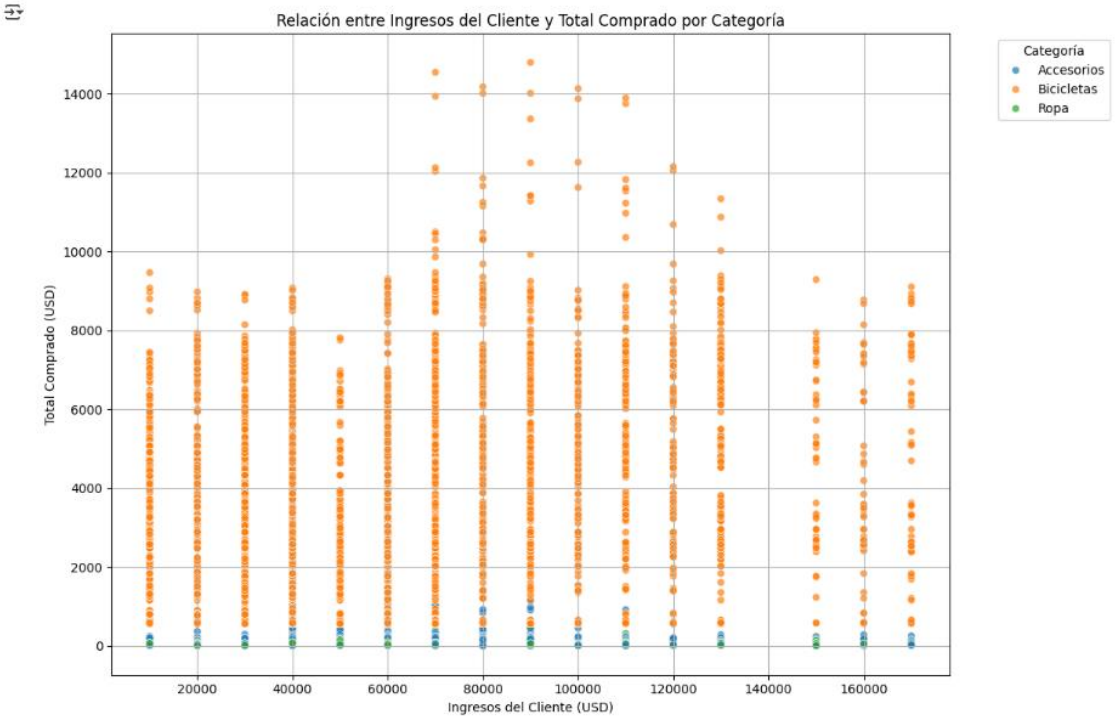
Se recomienda revisar posibles outliers o considerar normalización/transformación de variables o reducción del grado del polinomio para optimizar el desempeño.

Conclusión comparativa:

MÉTRICA	REGRESIÓN LINEAL	REGRESIÓN POLINÓMICA
RMSE	0.000	83.006
MAE	0.000	70.757
Correlación	1.000	0.997

A pesar de que la regresión lineal muestra resultados perfectos, la regresión polinómica es más realista y permite una mejor evaluación del rendimiento en un escenario práctico.

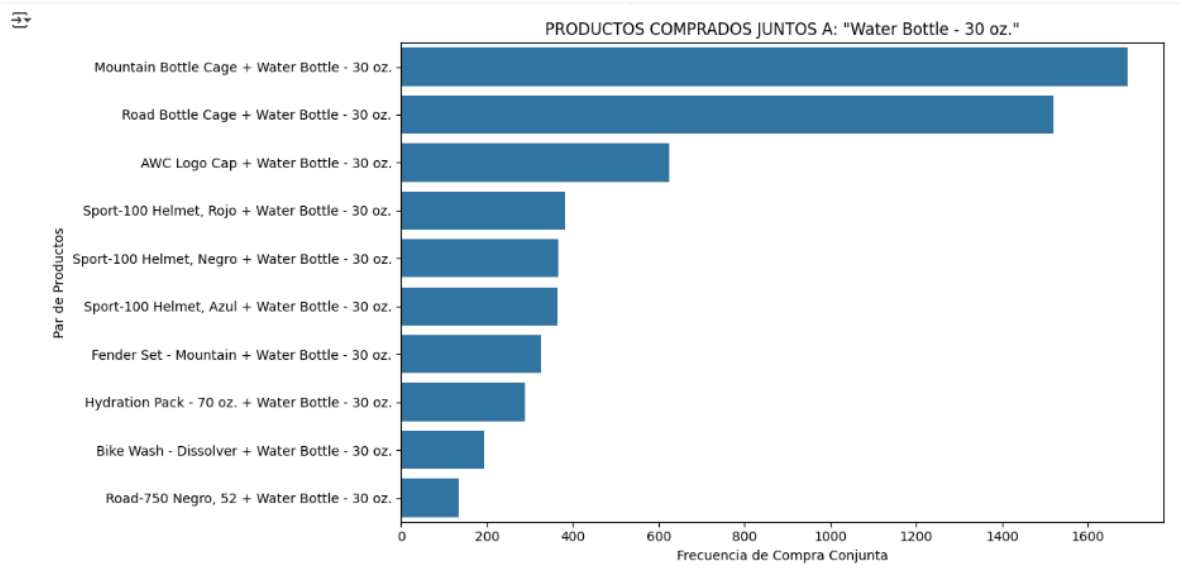
FASE 6. DESPLIEGUE (DEPLOYMENT)



Análisis Estratégico:

- El análisis del comportamiento de compra en función de los ingresos de los clientes permitió identificar diversos perfiles estratégicos que pueden ser aprovechados para optimizar las acciones de marketing. En primer lugar, se observó un grupo de clientes con altos niveles de ingreso pero con un bajo gasto en compras, lo que representa una clara oportunidad para el desarrollo de campañas de retargeting o fidelización mediante promociones exclusivas y personalizadas. Para este segmento, resulta especialmente útil implementar estrategias de up-selling y cross-selling, incentivando la adquisición de productos de mayor valor o complementarios.
- Se identificaron clientes con ingresos medios que registran un gasto elevado, lo cual sugiere una base sólida de compradores potencialmente leales. En estos casos, es recomendable considerar la implementación de programas de recompensas, membresías o beneficios diferenciados que fortalezcan la relación comercial a largo plazo. Este tipo de segmentación permite enfocar recursos en quienes demuestran una alta propensión al consumo, independientemente de su nivel de ingreso.

- Se evidenció una concentración significativa de compras por categoría entre clientes con altos ingresos, lo que ofrece una guía clara para direccionar campañas de publicidad específicas hacia este segmento, priorizando las líneas premium o de mayor valor percibido. Este hallazgo permite alinear la oferta con las preferencias del consumidor y potenciar la rentabilidad del portafolio de productos.
- Se detectó un grupo de clientes dispersos sin un patrón de comportamiento definido, lo que sugiere la necesidad de aplicar técnicas de segmentación más avanzadas, como el análisis de clústeres, con el objetivo de identificar subgrupos homogéneos y diseñar estrategias personalizadas según sus características particulares. Esta perspectiva orientada por datos permite una toma de decisiones más informada y una optimización de los recursos destinados al marketing y la gestión comercial.



Estrategia de Cross-Selling y Combos:

- A partir del análisis de productos comprados en conjunto, se identificaron oportunidades valiosas para la implementación de estrategias de cross-selling y venta en combos. Uno de los principales hallazgos fue la existencia de productos complementarios que aparecen con alta frecuencia junto al artículo más vendido, lo cual sugiere que estos forman parte de kits, soluciones integradas o simplemente son altamente funcionales cuando se utilizan en conjunto. Este tipo de relación entre productos brinda una base sólida para el desarrollo de campañas de venta cruzada, mejorando la experiencia del cliente y aumentando el valor promedio por transacción.

- Los resultados del análisis permiten diseñar promociones específicas basadas en la compra conjunta, tales como paquetes con descuentos, bundles promocionales o incentivos por adquirir productos relacionados. Este enfoque no solo impulsa el volumen de ventas, sino que también favorece la rotación de inventario y la maximización del margen comercial por operación.
- Se propone implementar en los canales digitales de venta, como la tienda online, sistemas de recomendación automática basados en patrones de compra previamente identificados. Frases como “Clientes que compraron este producto también adquirieron” pueden integrarse en la interfaz de usuario, facilitando decisiones de compra y fomentando la adquisición de múltiples productos en una sola visita. Esta estrategia, sustentada en datos reales de comportamiento de compra, incrementa la personalización de la oferta y potencia significativamente las oportunidades de crecimiento en las ventas.

CONCLUSIONES GENERALES.

- El desarrollo de este proyecto de minería de datos, fundamentado en la metodología CRISP-DM, ha permitido transformar un volumen significativo de información transaccional en conocimiento accionable para la gestión comercial de Adventure Works. A través de un análisis estructurado, se identificaron patrones de comportamiento clave tanto en clientes como en productos, aportando una visión integral del desempeño de ventas.
- Uno de los hallazgos más relevantes fue la identificación del top de productos más vendidos por categoría, lo que permite priorizar inversiones en líneas de alta rotación y planificar adecuadamente el abastecimiento y la logística. El análisis de la relación entre ingresos del cliente y volumen de compra reveló perfiles de consumo contrastantes, destacando segmentos con alto potencial que no están siendo plenamente capitalizados por las estrategias actuales.
- El estudio de ventas cruzadas permitió identificar productos que se adquieren frecuentemente en conjunto, abriendo la puerta a la implementación de estrategias de *cross-selling*, promoción de combos y mejoras en la experiencia de compra digital. Este tipo de análisis revela no solo qué se vende, sino cómo y con qué frecuencia los

productos se relacionan entre sí, generando oportunidades directas para aumentar el valor promedio de cada transacción.

- Se destaca el despliegue efectivo de las visualizaciones analíticas, que convierten la información técnica en herramientas de fácil interpretación para los distintos niveles de decisión empresarial, facilitando la adopción de una cultura organizacional basada en datos.

RECOMENDACIONES GENERALES.

- En función del análisis por ingresos y comportamiento de compra, se recomienda desarrollar campañas diferenciadas para cada segmento identificado, especialmente enfocados en retener clientes de alto ingreso con bajo gasto mediante promociones exclusivas y ofertas personalizadas.
- Para los clientes con gasto elevado y comportamiento de compra recurrente, se sugiere establecer membresías, programas de puntos o descuentos escalables que promuevan la lealtad a largo plazo.
- Aprovechar los productos identificados como complementarios para desarrollar kits promocionales, bundles o recomendaciones automáticas en tienda online, lo que incrementará la venta promedio por cliente y mejorará la experiencia de usuario.
- Se sugiere continuar el proyecto con modelos de predicción de comportamiento de compra y técnicas de clustering para refinar aún más la personalización de campañas y anticipar demandas futuras.