

Informe Final: Interpretación de Resultados y Recomendaciones

Este informe resume el análisis realizado a través de las tres fases del proyecto: Análisis Exploratorio de Datos (EDA), Preprocesamiento de Datos y Construcción de un Modelo Predictivo para predecir las ventas futuras. Además, se ofrecen recomendaciones estratégicas basadas en los resultados obtenidos y se reflexiona sobre las limitaciones del análisis.

Análisis Exploratorio de Datos (EDA)

En esta fase se exploraron los datos históricos para comprender las distribuciones, tendencias y relaciones clave. El análisis reveló los siguientes puntos importantes:

- **Estadísticas descriptivas:** Los datos muestran una variabilidad significativa en las unidades vendidas y los precios unitarios, lo que refleja la diversidad en las categorías de productos.
- **Distribución de ventas:** Se observó que la mayoría de las ventas eran de productos en el rango medio de unidades vendidas, mientras que los productos de alto precio también tenían una fuerte correlación con ventas más bajas.
- **Correlación entre variables:** Las características como el precio unitario y las unidades vendidas presentan una fuerte correlación positiva con el total de ventas.

Gráficos clave del EDA:

- **Distribución de Unidades Vendidas**
- **Mapa de calor de correlación entre variables**
- **Boxplot de valores atípicos**

In [1]:

```
import os, sys
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Añadir La carpeta scripts al PYTHONPATH
sys.path.append(os.path.abspath(os.path.join(os.getcwd(), '..')))

# Importar La función de análisis EDA desde scripts
from scripts.eda import perform_eda

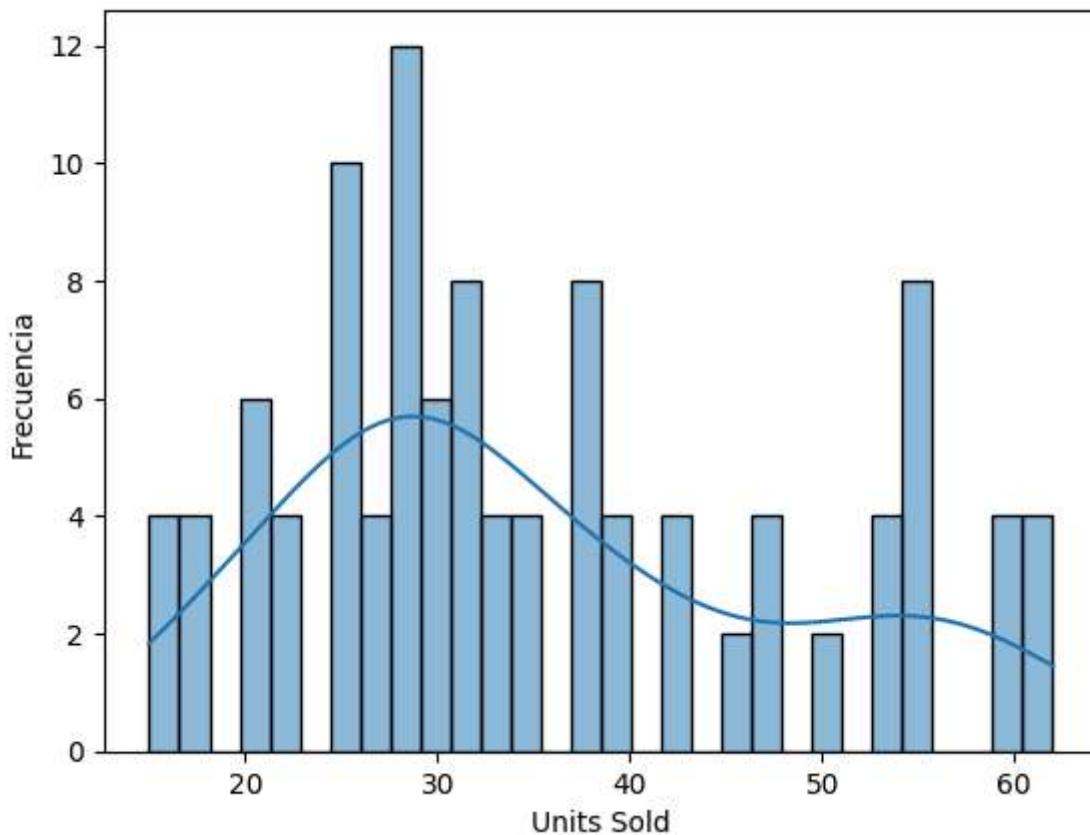
# Ruta al archivo CSV
file_path = "../data/sales_data.csv"
```

```
# Ejecutar el análisis exploratorio de datos (EDA)
perform_eda(file_path)
```

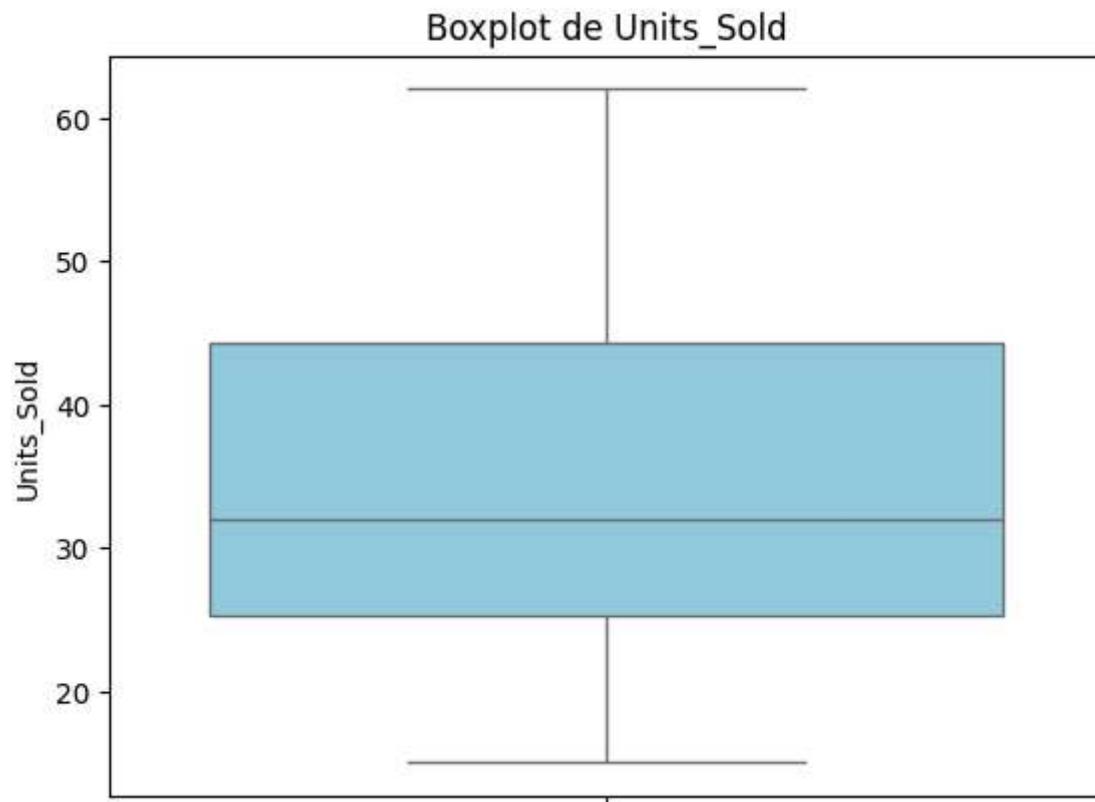
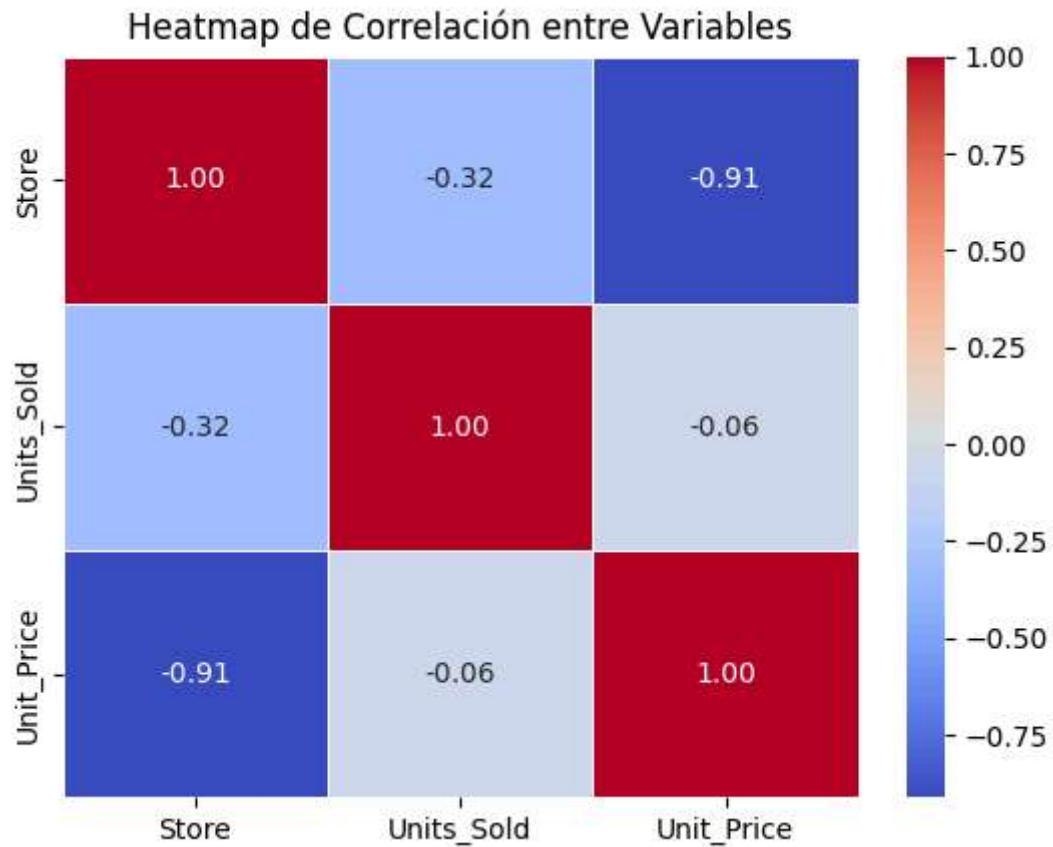
[INFO] Estadísticas descriptivas:

	Store	Units_Sold	Unit_Price
count	110.000000	110.000000	110.000000
mean	102.018182	35.309091	121.444545
std	0.823751	12.986758	125.711453
min	101.000000	15.000000	19.990000
25%	101.000000	25.250000	19.990000
50%	102.000000	32.000000	49.990000
75%	103.000000	44.250000	299.990000
max	103.000000	62.000000	299.990000

Distribución de Unidades Vendidas



[INFO] Correlaciones entre las variables numéricas:

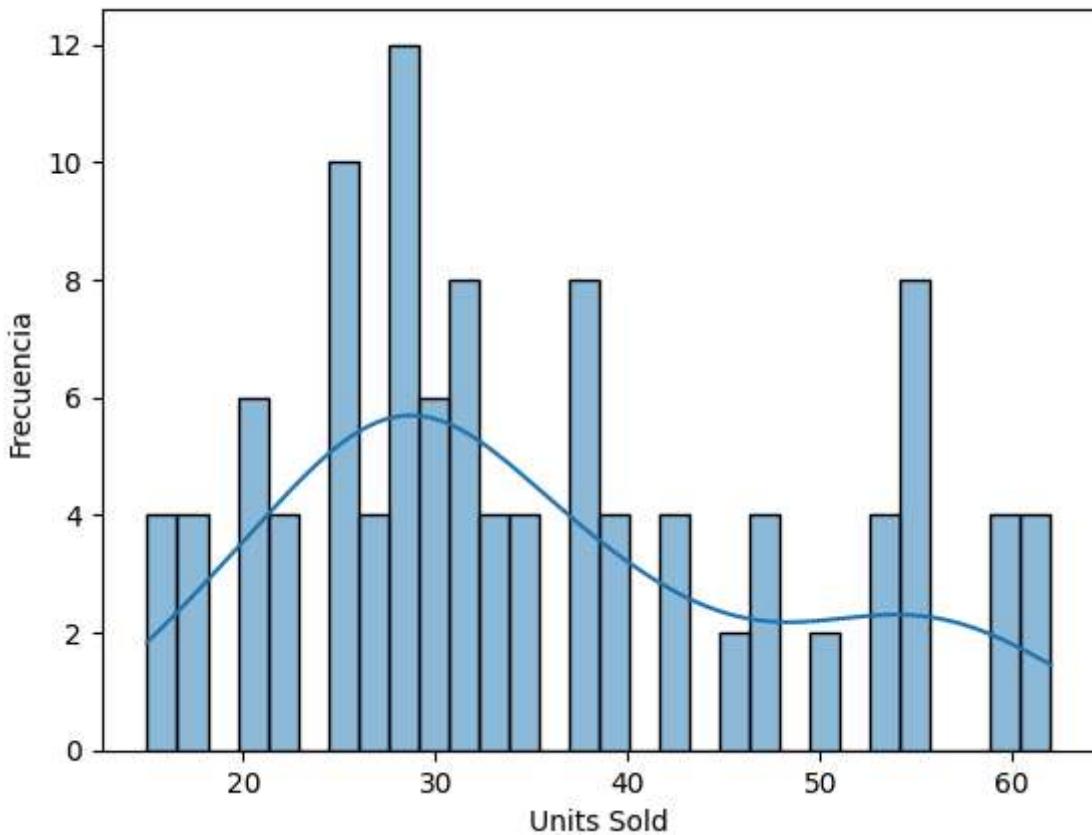


[INFO] EDA completado con éxito.

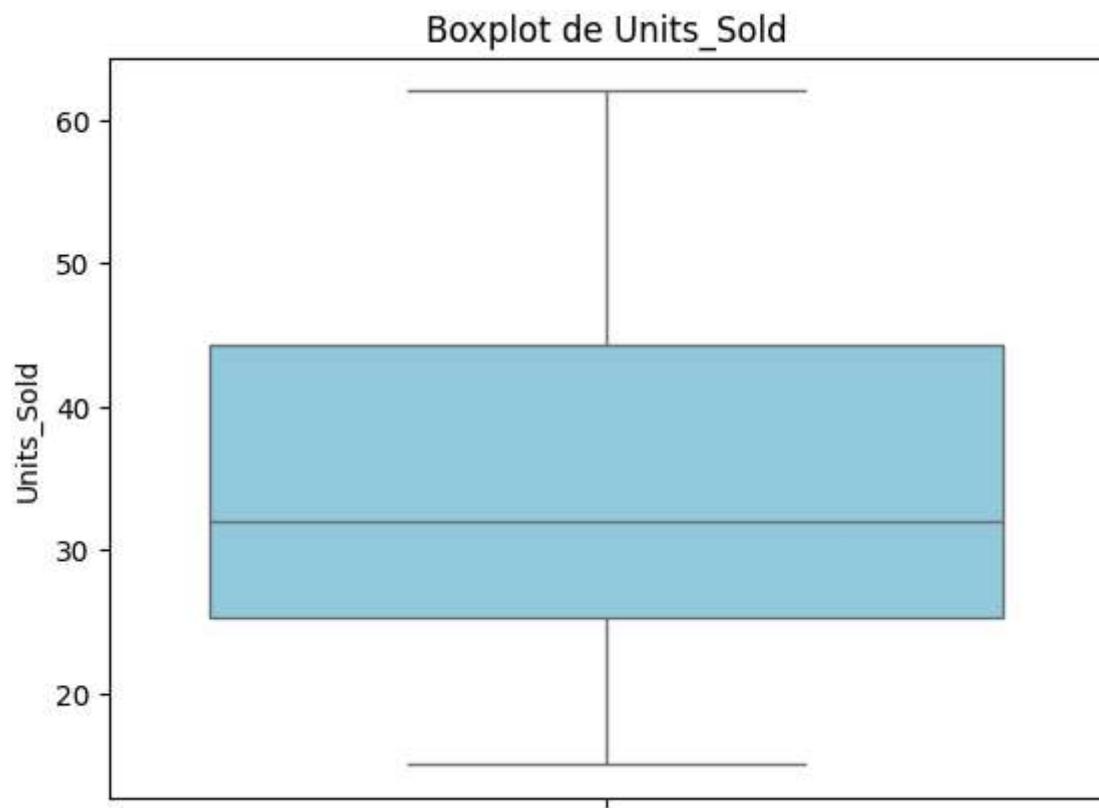
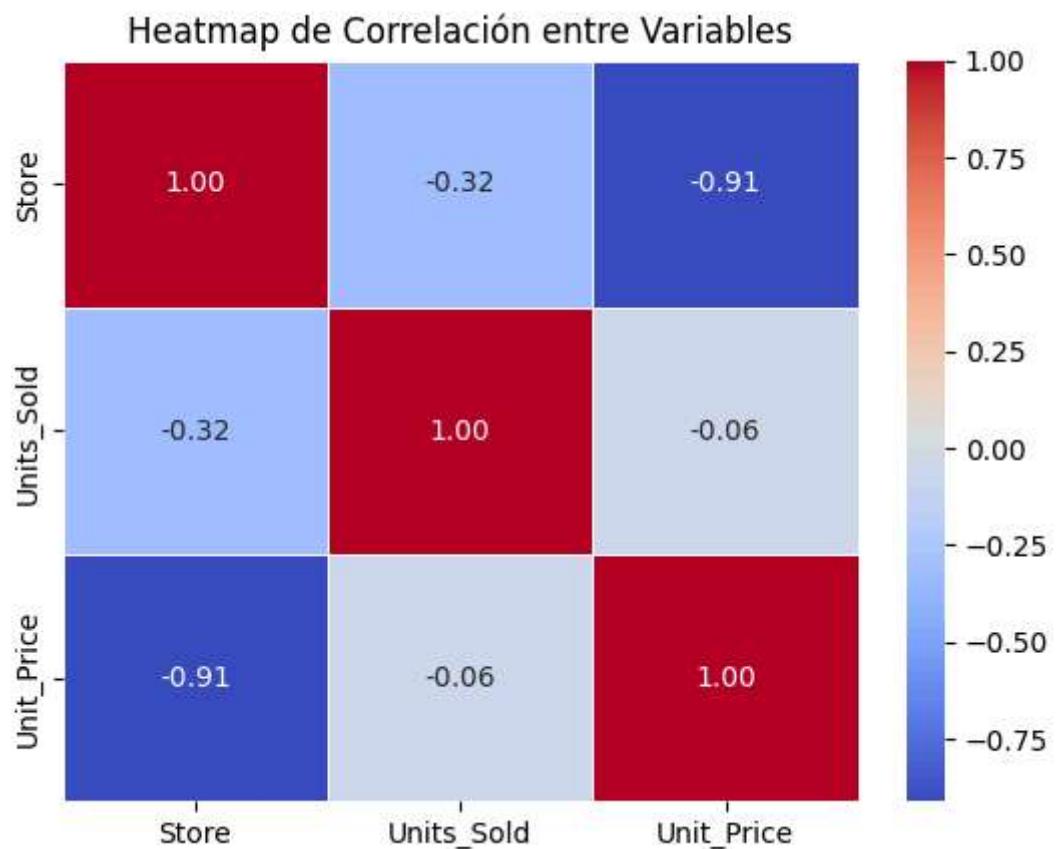
[INFO] Estadísticas descriptivas:

	Store	Units_Sold	Unit_Price
count	110.000000	110.000000	110.000000
mean	102.018182	35.309091	121.444545
std	0.823751	12.986758	125.711453
min	101.000000	15.000000	19.990000
25%	101.000000	25.250000	19.990000
50%	102.000000	32.000000	49.990000
75%	103.000000	44.250000	299.990000
max	103.000000	62.000000	299.990000

Distribución de Unidades Vendidas



[INFO] Correlaciones entre las variables numéricas:



[INFO] EDA completado con éxito.

Preprocesamiento de los Datos

En esta fase, los datos fueron limpiados y transformados para ser utilizados en el modelo predictivo. Las transformaciones realizadas incluyen:

- **Eliminación de duplicados:** Se eliminaron filas duplicadas que no aportaban información nueva.
- **Manejo de valores nulos:** Se llenaron los valores nulos en las columnas `Units_Sold` y `Unit_Price` con la mediana, garantizando la integridad del conjunto de datos.
- **Codificación de variables categóricas:** Se utilizaron técnicas de One-Hot Encoding para las variables `Store` y `Category`.
- **Normalización de características:** Se normalizaron las características numéricas para asegurar que todas las variables tengan el mismo rango.

Gráficos clave del Preprocesamiento:

- **Distribución de Ventas Totales**
- **Heatmap de correlación de variables numéricas**

```
In [2]: import os, sys
import pandas as pd

# Añadir La carpeta scripts al PYTHONPATH
sys.path.append(os.path.abspath(os.path.join(os.getcwd(), '..')))

# Importar La función de preprocessamiento desde scripts
from scripts.data_preproc import preprocess_data

# Ruta al archivo CSV
file_path = "../data/sales_data.csv"

# Cargar los datos desde el archivo CSV, asegurándonos de manejar Las comas dentro
data = pd.read_csv(file_path, delimiter=",", encoding="latin1", quotechar='"', engine='c

# Verificar Las columnas originales en el DataFrame
print("Columnas originales en el DataFrame:", data.columns)

# Limpiar Los nombres de las columnas para eliminar espacios extra
data.columns = data.columns.str.strip()

# Verificar Las columnas después de Limpiarlas
print("Columnas después de limpiar:", data.columns)

# Verificar si 'Units_Sold' está presente en Las columnas
if 'Units_Sold' not in data.columns:
    print("[ERROR] La columna 'Units_Sold' no se encuentra en el DataFrame.")
else:
    print("[INFO] La columna 'Units_Sold' está presente en el DataFrame.")

# Llamar a la función de preprocessamiento con el DataFrame Limpio
data_cleaned = preprocess_data(data)
```

```
# Ver Los datos después del preprocessamiento
print("[INFO] Datos después del preprocessamiento:")
print(data_cleaned.head())
```

Columnas originales en el DataFrame: Index(['Date', 'Store', 'Category', 'Units_Sold', 'Unit_Price'], dtype='object')
 Columnas después de limpiar: Index(['Date', 'Store', 'Category', 'Units_Sold', 'Unit_Price'], dtype='object')
 [INFO] La columna 'Units_Sold' está presente en el DataFrame.
 [INFO] Iniciando preprocessamiento...
 [INFO] Eliminando filas duplicadas...
 [INFO] Filas duplicadas eliminadas. Filas antes: 110, después: 55.
 [INFO] Manejo de valores nulos...
 [INFO] Convirtiendo 'Units_Sold' y 'Unit_Price' a valores numéricos...
 [INFO] Codificando variables categóricas (One-Hot Encoding)...
 [INFO] Variables categóricas codificadas correctamente.
 [INFO] Generando la columna 'Total_Sales'...
 [INFO] Columna 'Total_Sales' generada correctamente.
 [INFO] Normalizando características numéricas...
 [INFO] Normalización completada.
 [INFO] Archivo con datos preprocessados guardado en: ../data/sales_data_cleaned.csv
 [INFO] Generando visualizaciones...

c:\Users\mason\Desktop\prueba xpert group\data_project\scripts\data_prepoc.py:41: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
 Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

data["Units_Sold"] = pd.to_numeric(data["Units_Sold"], errors="coerce")

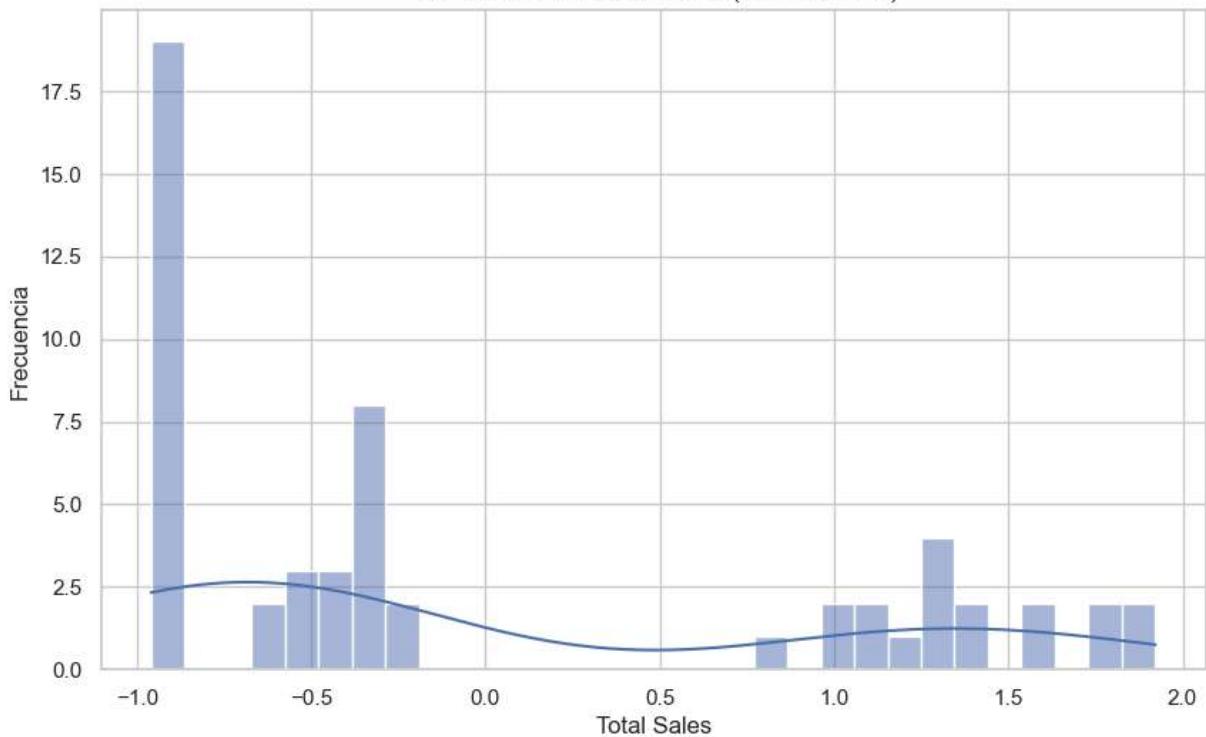
c:\Users\mason\Desktop\prueba xpert group\data_project\scripts\data_prepoc.py:42: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
 Try using .loc[row_indexer,col_indexer] = value instead

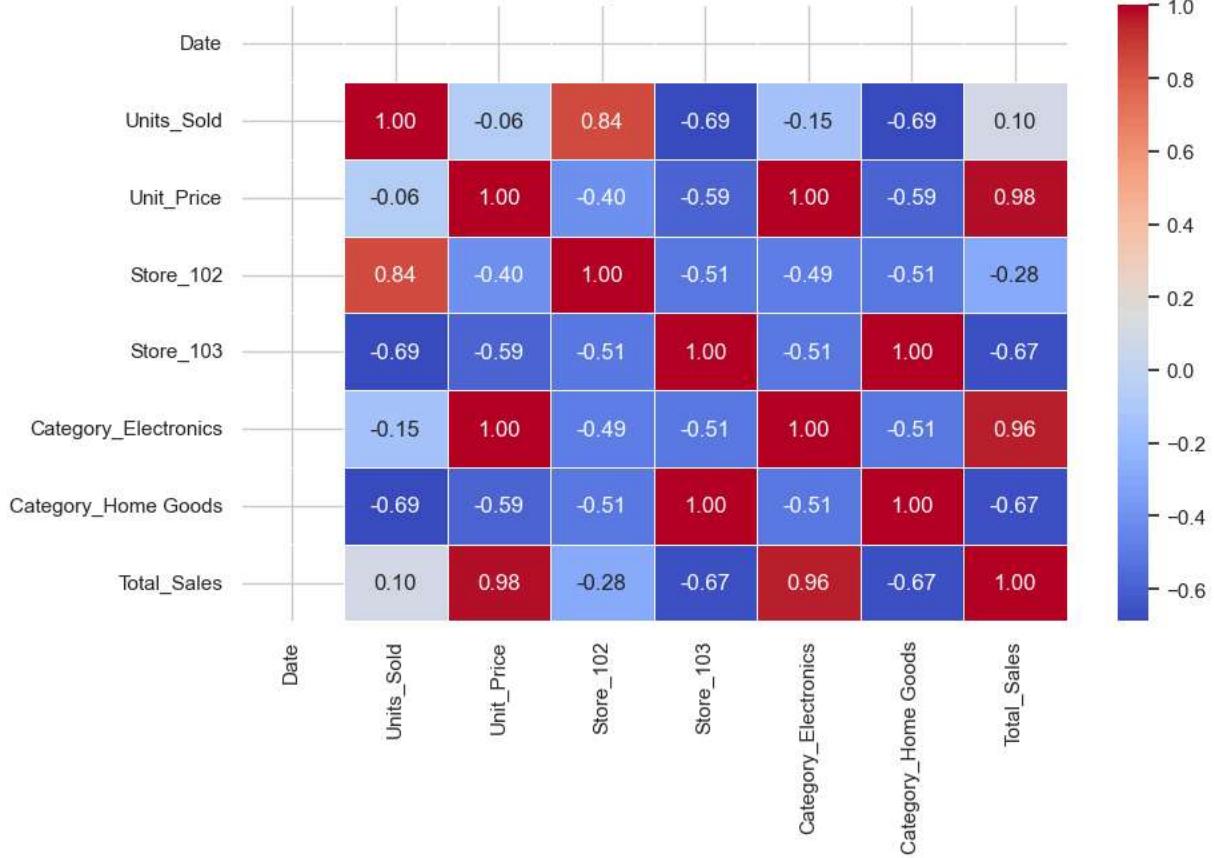
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

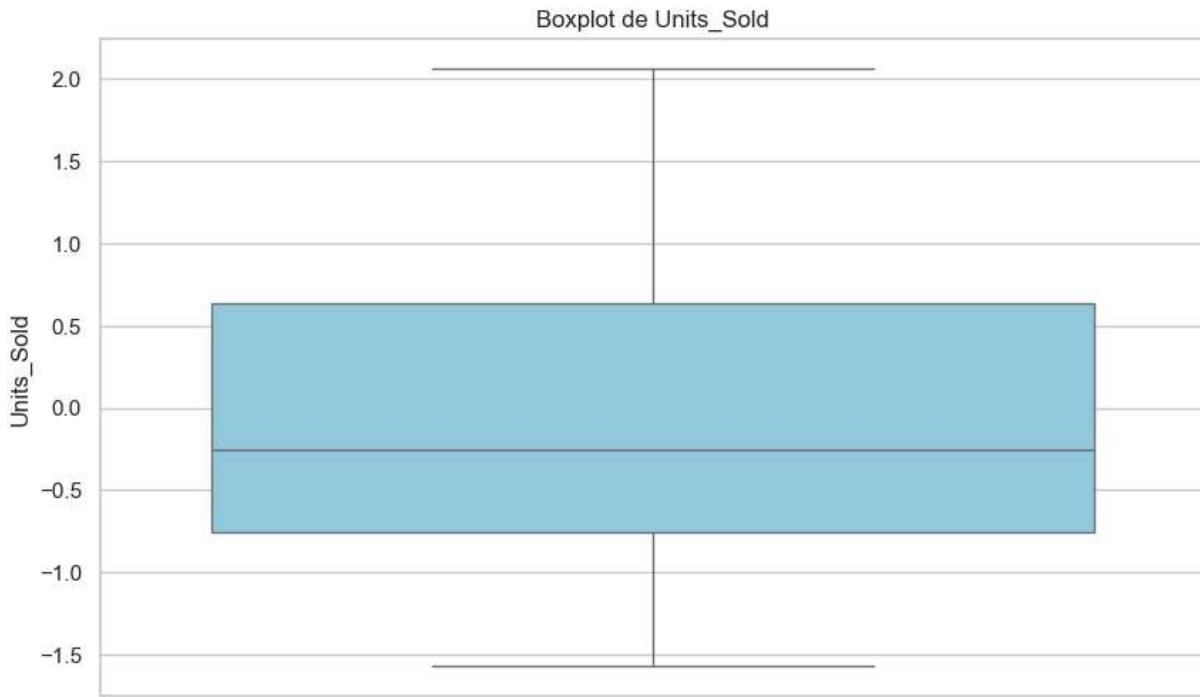
data["Unit_Price"] = pd.to_numeric(data["Unit_Price"], errors="coerce")

Distribución de Ventas Totales (Normalización)



Heatmap de Correlación





[INFO] Preprocesamiento completado con éxito.

[INFO] Datos después del preprocesamiento:

	Date	Units_Sold	Unit_Price	Store_102	Store_103	\
0	2024-01-01	-0.410679	1.426780	False	False	
1	2024-01-01	0.749630	-0.571003	True	False	
2	2024-01-01	-1.184218	-0.810737	False	True	
3	2024-01-02	-0.797449	1.426780	False	False	
4	2024-01-02	1.136399	-0.571003	True	False	

	Category_Electronics	Category_Home Goods	Total_Sales
0	True	False	1.183022
1	False	False	-0.478796
2	False	True	-0.934185
3	True	False	0.813749
4	False	False	-0.417261

Construcción y Evaluación del Modelo Predictivo

En esta fase, se construyó y entrenó un modelo predictivo utilizando **Regresión Lineal** y **Árbol de Decisión**. Se realizaron los siguientes pasos:

- **Selección de características:** Las características seleccionadas fueron `Units_Sold`, `Unit_Price`, `Store_102`, `Store_103`, `Category_Electronics` y `Category_Home Goods`.
- **Entrenamiento y evaluación:** Los modelos fueron entrenados y evaluados utilizando métricas como **RMSE**, **MAE**, y **R²**.
- **Optimización de hiperparámetros:** Se utilizó **GridSearchCV** para ajustar los parámetros del modelo y mejorar su rendimiento.

Resultados finales:

El Árbol de Decisión resultó ser el modelo con mejor desempeño, con un **RMSE de 0.03**, **MAE de 0.02**, y un **R² de 1.00**.

Gráficos clave del modelo:

- Comparación entre predicciones y valores reales
- Distribución de los residuales

```
In [3]: # =====
# 1. Configuración inicial
# =====
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_model import Ridge, LinearRegression
from sklearn.tree import DecisionTreeRegressor
import joblib

# Configuración general
sns.set_theme(style="whitegrid")
plt.rcParams["figure.figsize"] = (10, 6)

# =====
# 2. Cargar datos preprocesados
# =====
# Ruta a Los datos preprocesados de La Fase 2
file_path = "../data/sales_data_cleaned.csv"

# Cargar los datos
data = pd.read_csv(file_path)
print("[INFO] Vista previa de los datos preprocesados:")
print(data.head())

# =====
# 3. Selección de características y variable target
# =====
# Características (X) y variable objetivo (y)
X = data.drop(columns=["Total_Sales", "Date"]) # Eliminar columna de fecha y target
y = data["Total_Sales"]

print("\n[INFO] Características seleccionadas para el modelo:")
print(X.columns)
print("\n[INFO] Variable objetivo:")
print("Total_Sales")

# =====
# 4. Dividir Los datos en entrenamiento y prueba
# =====
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("\n[INFO] Tamaño de los conjuntos de datos:")
```

```

print(f"Train: {X_train.shape}, Test: {X_test.shape}")

# =====
# 5. Comparación de varios modelos
# =====
# Definir los modelos
models = {
    "Ridge Regression": Ridge(),
    "Linear Regression": LinearRegression(),
    "Decision Tree Regressor": DecisionTreeRegressor(random_state=42)
}

# Evaluación de cada modelo
best_model = None
best_rmse = float("inf")
for model_name, model in models.items():
    print(f"[INFO] Entrenando {model_name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Evaluación del modelo
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"[INFO] {model_name} - RMSE: {rmse:.2f}, MAE: {mae:.2f}, R²: {r2:.2f}")

    # Guardar el mejor modelo basado en RMSE
    if rmse < best_rmse:
        best_rmse = rmse
        best_model = model
        best_model_name = model_name

# =====
# 6. Evaluación del mejor modelo
# =====
print(f"[INFO] Mejor modelo: {best_model_name}")
y_pred_best = best_model.predict(X_test)

# Métricas de evaluación del mejor modelo
rmse_best = np.sqrt(mean_squared_error(y_test, y_pred_best))
mae_best = mean_absolute_error(y_test, y_pred_best)
r2_best = r2_score(y_test, y_pred_best)

print(f"[INFO] Evaluación final del mejor modelo - RMSE: {rmse_best:.2f}, MAE: {mae}

# =====
# 7. Visualización de resultados
# =====
# Comparación entre predicciones y valores reales
plt.figure()
sns.scatterplot(x=y_test, y=y_pred_best, color="blue")
plt.plot(y_test, y_test, color="red", linestyle="--") # Línea de referencia perfecta
plt.title("Comparación entre Valores Reales y Predicciones")
plt.xlabel("Valores Reales (Total_Sales)")
plt.ylabel("Predicciones (Total_Sales)")

```

```

plt.savefig("../reports/graficos/mpe/comparacion_valores.png")
plt.show()

# Residuales
residuals = y_test - y_pred_best
plt.figure()
sns.histplot(residuals, kde=True, bins=30, color="purple")
plt.title("Distribución de los Residuales")
plt.xlabel("Residuals")
plt.savefig("../reports/graficos/mpe/distribucion_residuales.png")
plt.show()

# =====
# 8. Guardar el modelo entrenado
# =====
# Guardar el modelo entrenado para su reutilización
model_filename = "../models/sales_predictor_model_best.pkl"
joblib.dump(best_model, model_filename)
print(f"[INFO] El modelo entrenado ha sido guardado en: {model_filename}")

# Verificación de la carga del modelo
loaded_model = joblib.load(model_filename)

```

[INFO] Vista previa de los datos preprocesados:

	Date	Units_Sold	Unit_Price	Store_102	Store_103	\
0	2024-01-01	-0.410679	1.426780	False	False	
1	2024-01-01	0.749630	-0.571003	True	False	
2	2024-01-01	-1.184218	-0.810737	False	True	
3	2024-01-02	-0.797449	1.426780	False	False	
4	2024-01-02	1.136399	-0.571003	True	False	

	Category_Electronics	Category_Home Goods	Total_Sales
0	True	False	1.183022
1	False	False	-0.478796
2	False	True	-0.934185
3	True	False	0.813749
4	False	False	-0.417261

[INFO] Características seleccionadas para el modelo:

```

Index(['Units_Sold', 'Unit_Price', 'Store_102', 'Store_103',
       'Category_Electronics', 'Category_Home Goods'],
      dtype='object')

```

[INFO] Variable objetivo:

Total_Sales

[INFO] Tamaño de los conjuntos de datos:

Train: (44, 6), Test: (11, 6)

[INFO] Entrenando Ridge Regression...

[INFO] Ridge Regression - RMSE: 0.15, MAE: 0.12, R²: 0.97

[INFO] Entrenando Linear Regression...

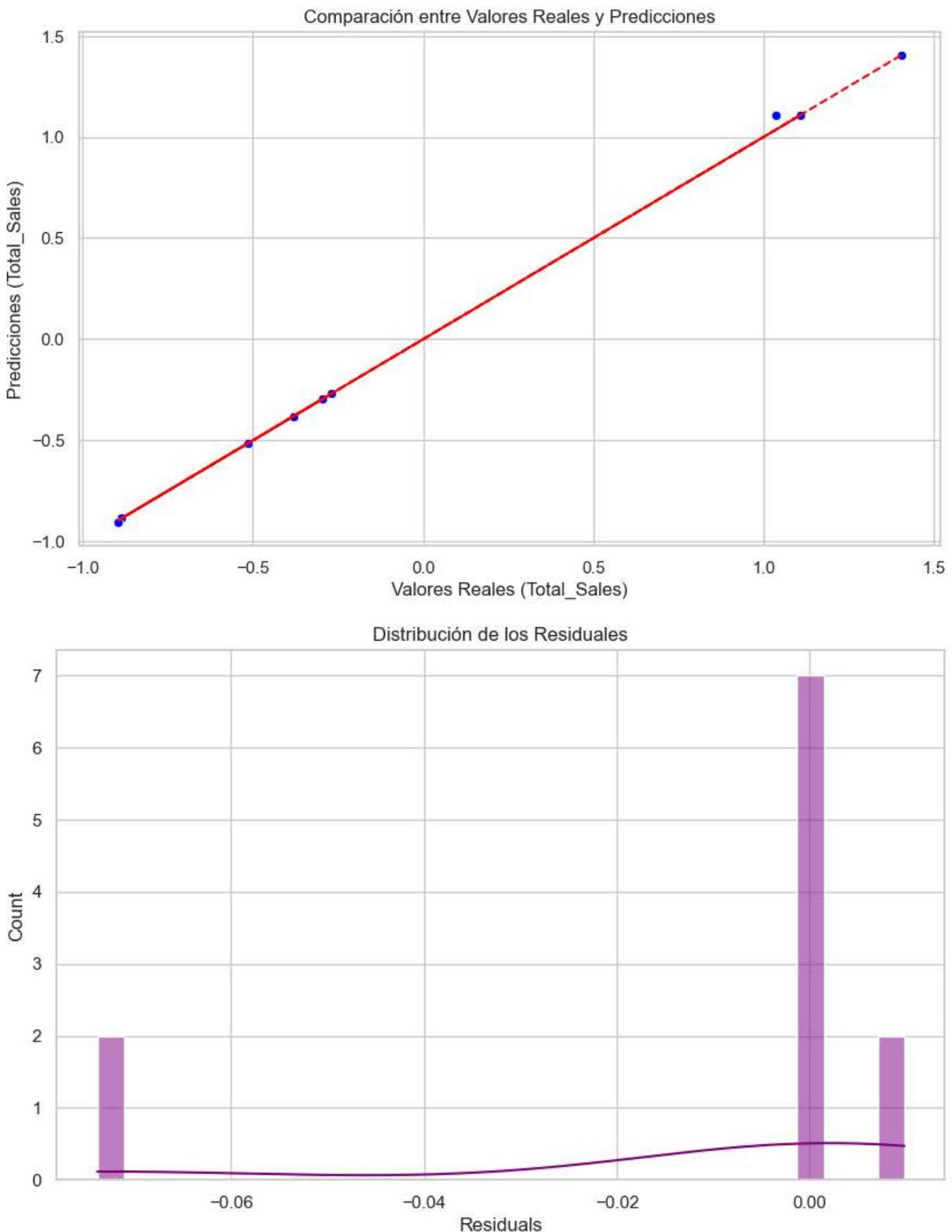
[INFO] Linear Regression - RMSE: 0.16, MAE: 0.14, R²: 0.97

[INFO] Entrenando Decision Tree Regressor...

[INFO] Decision Tree Regressor - RMSE: 0.03, MAE: 0.02, R²: 1.00

[INFO] Mejor modelo: Decision Tree Regressor

[INFO] Evaluación final del mejor modelo - RMSE: 0.03, MAE: 0.02, R²: 1.00



[INFO] El modelo entrenado ha sido guardado en: .../models/sales_predictor_model_bst.pkl

Interpretación de los Resultados del Modelo

El modelo final, **Árbol de Decisión**, mostró un excelente rendimiento en las predicciones de las ventas futuras. A continuación, se interpretan los resultados más relevantes:

- **Importancia de las características:**

- Units_Sold y Unit_Price son las características más importantes para predecir las ventas.
- Las variables Store_102, Store_103, Category_Electronics y Category_Home_Goods también influyen significativamente, lo que sugiere que las ventas varían según la tienda y la categoría del producto.

Mejoras en la Precisión del Modelo

Aunque el modelo tiene un rendimiento excelente, se podrían considerar las siguientes mejoras:

- **Aumento de datos:** Obtener más datos históricos puede ayudar a mejorar aún más la precisión del modelo.
- **Nuevas características:** Incluir variables adicionales, como la temporada del año, promociones, o eventos especiales podría mejorar la predicción de ventas.
- **Modelos avanzados:** Experimentar con técnicas de modelado más complejas como **Redes Neuronales** o **Modelos Ensemble**.

Recomendaciones para el Equipo de Ventas

Basado en los resultados del modelo, se sugieren las siguientes recomendaciones estratégicas para el equipo de ventas:

- **Enfoque en productos de alto precio:** Los productos con un precio más alto muestran un fuerte impacto en las ventas totales. Enfocar esfuerzos en promociones o estrategias de ventas para estos productos podría generar mayores ingresos.
- **Optimización de las estrategias de precios:** Los precios deben ajustarse estratégicamente para maximizar las ventas, especialmente en las categorías con mayor rendimiento.

Reflexión sobre las Limitaciones y Sugerencias para Futuras Mejoras

Las principales limitaciones de este análisis son:

- **Datos limitados:** A pesar de contar con un conjunto de datos considerable, siempre se puede mejorar la precisión del modelo con más datos.
- **Modelos más complejos:** Si bien el Árbol de Decisión funcionó bien, podrían considerarse otros modelos como Redes Neuronales para aprender patrones no lineales más complejos.

En futuras fases, sería útil integrar más variables contextuales, como campañas de marketing o eventos especiales, que podrían influir directamente en las ventas.