UNIVERSIDAD ANDINA DEL CUSCO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Informe Dataset (Entrenamiento y predicción)

ASIGNATURA: INTELIGENCIA ARTIFICIAL

DOCENTE: ESPETIA HUAMANGA HUGO

ESTUDIANTES:

CUSI RONCO JHOEL

HUARACHI PUMACHAPI JUAN ALBERTO

MENDOZA CHOQUEHUILLCA ULISES VALENTY

QUISPE CCOPA EVELYN

CUSCO - PERÚ

2024

1. Limpieza del DataSet

Descripción:

La limpieza de datos es un paso crucial en el análisis de datos y la construcción de modelos predictivos. Consiste en preparar el DataSet eliminando datos incorrectos, incompletos o irrelevantes para asegurar la calidad del análisis y la precisión de los modelos predictivos.

Propósito:

- **Eliminar filas con valores faltantes**: Esto es importante para asegurar que los modelos no se vean afectados por datos incompletos. En tu caso, se eliminan las filas donde faltan los valores de 'Nombre del producto' y 'Marca del producto'.
- **Reemplazar valores no numéricos por NaN**: Convierte los valores no numéricos en la columna 'Precio unitario del producto' en NaN para que puedan ser manejados adecuadamente (en este caso, se reemplazan las monedas y se convierten en valores numéricos).
- **Manejar valores NaN**: Después de reemplazar valores no numéricos, hay que manejar los valores NaN resultantes. En tu código, se reemplazan por la media de la columna para evitar afectar la distribución de datos.
- **Convertir tipos de datos**: Asegura que las columnas estén en el tipo de datos correcto, en este caso, 'Unidades Vendidas' se convierte a entero para evitar problemas durante el análisis.

```python
import pandas as pd
import numpy as np

# Cargar el dataset con encoding adecuado
try:
    data = pd.read_csv('kaggle.csv', encoding='ISO-8859-1')
except UnicodeDecodeError:
    data = pd.read_csv('kaggle.csv', encoding='latin1')

# Eliminar filas con valores faltantes en 'Nombre del producto' y 'Marca del producto'
data = data.dropna(subset=['Nombre del producto', 'Marca del producto'])

# Reemplazar valores no numéricos por NaN en 'Precio unitario del producto'
data['Precio unitario del producto'] = pd.to_numeric(data['Precio unitario del producto'].replace('[\$,]', '', regex=True), errors='coerce')

# Verificar y manejar valores NaN en 'Precio unitario del producto'
print(data['Precio unitario del producto'].isnull().sum())
data['Precio unitario del producto'] = data['Precio unitario del producto'].fillna(data['Precio unitario del producto'].mean())

# Convertir 'Unidades Vendidas' a entero
data['Unidades Vendidas'] = data['Unidades Vendidas'].astype(int)

# Verificar los cambios
print(data.info())
```

```
5
<class 'pandas.core.frame.DataFrame'>
Index: 25454 entries, 0 to 25637
Data columns (total 10 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Fecha                           25454 non-null  int64
 1   ID del producto                 25454 non-null  int64
 2   Nombre del producto             25454 non-null  object
 3    Nombre completo del producto   25454 non-null  object
 4   Marca del producto              25454 non-null  object
 5   Categoría                       25454 non-null  object
 6   Subcategoría                    25454 non-null  object
 7   Etiquetas                       25454 non-null  object
 8   Precio unitario del producto    25454 non-null  float64
 9   Unidades Vendidas               25454 non-null  int64
dtypes: float64(1), int64(3), object(6)
memory usage: 2.1+ MB
None
```

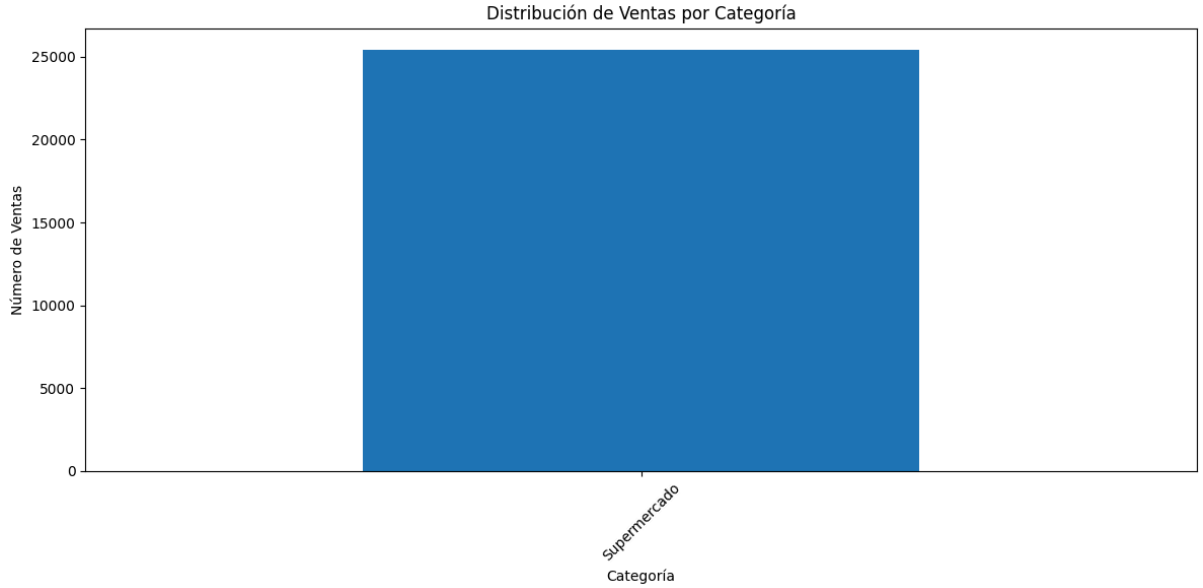
2. Análisis del DataSet

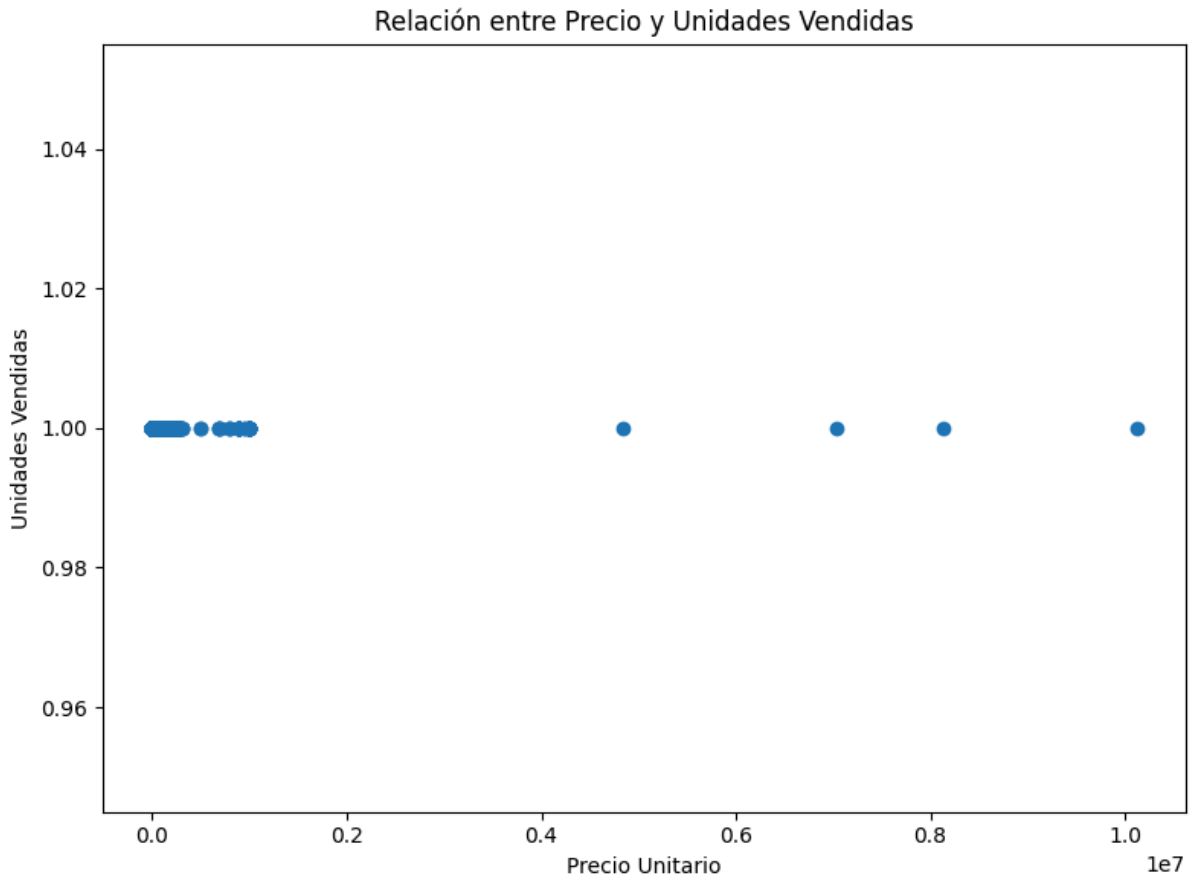
Propósito: Comprender las características y la estructura del dataset para informar el desarrollo de los modelos de predicción.

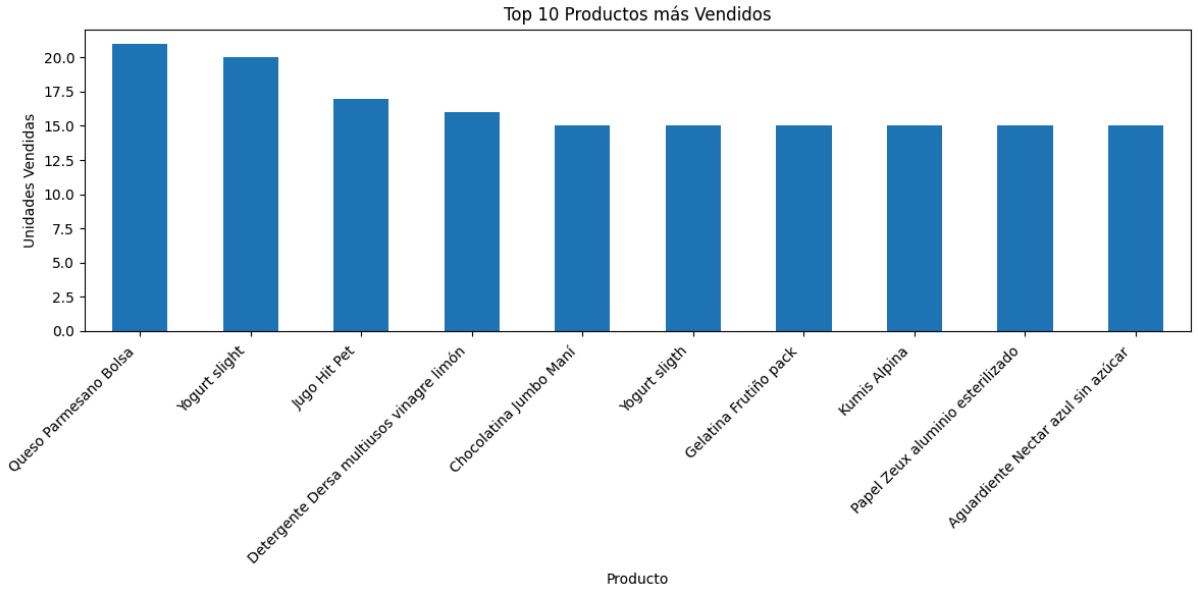
Descripción:

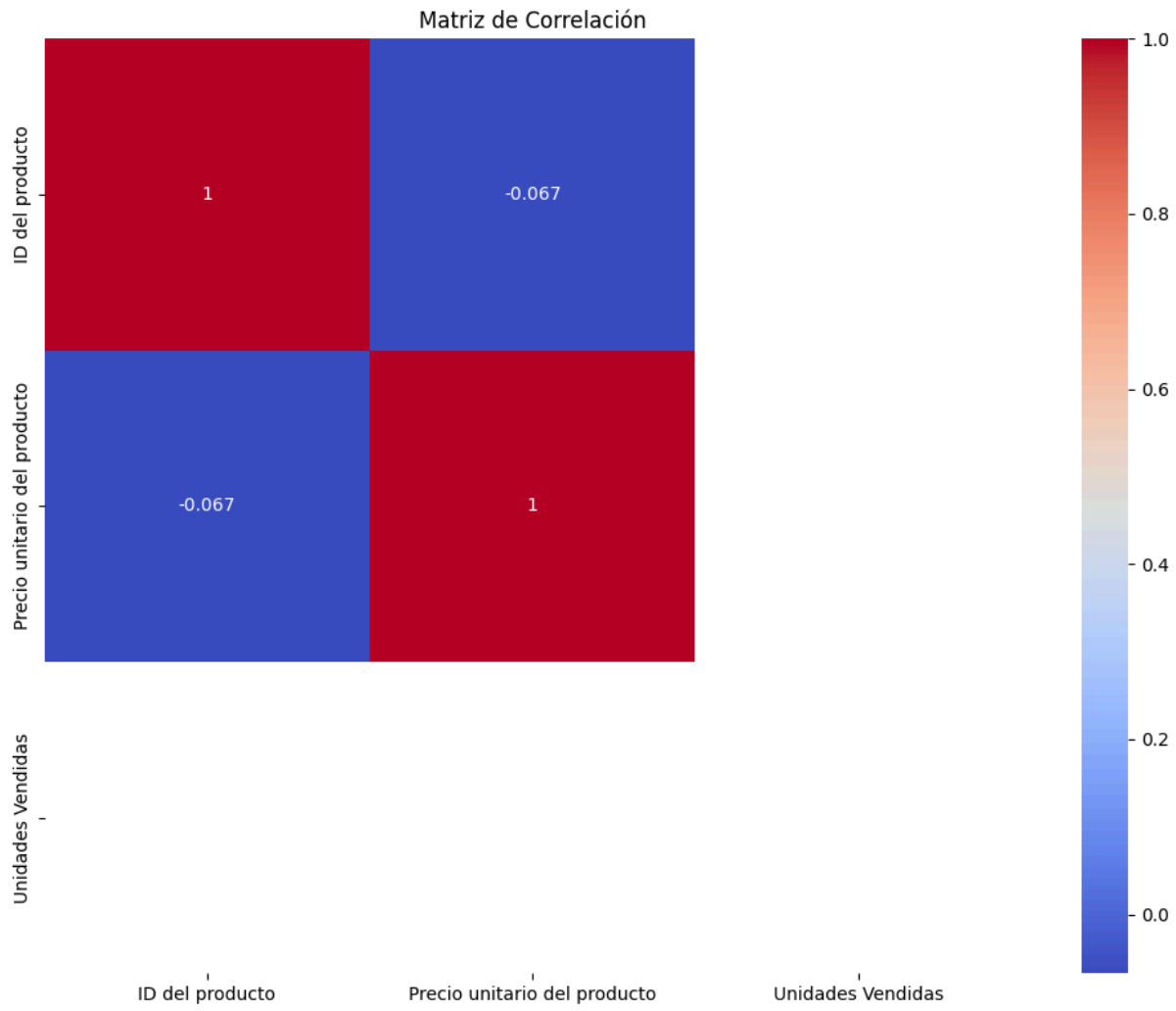
- **Exploración de datos:** Investigar la distribución de variables, estadísticas descriptivas y posibles correlaciones.
- **Visualización:** Utilizar gráficos para explorar relaciones entre variables, como histogramas para la distribución de precios y ventas, y diagramas de dispersión para observar correlaciones.

	ID del producto	Precio unitario del producto	Unidades Vendidas
count	2.545400e+04	2.545400e+04	25454.0
mean	7.136823e+12	1.737640e+04	1.0
std	1.872465e+12	1.051179e+05	0.0
min	5.244812e+06	0.000000e+00	1.0
25%	7.702010e+12	4.170000e+03	1.0
50%	7.702189e+12	7.390000e+03	1.0
75%	7.703616e+12	1.399000e+04	1.0
max	9.333527e+12	1.011905e+07	1.0




Relación entre Precio y Unidades Vendidas
Unidades Vendidas
1.04
1.02
1.00
0.98
0.96
0.0
0.2
0.4
0.6
0.8
1.0
Precio Unitario
1e7


Top 10 Productos más Vendidos
Unidades Vendidas
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Queso Parmesano Bolsa
Yogurt slight
Jugo Hit Pet
Detergente Dersa multiusos vinagre limón
Chocolatina Jumbo Maní
Yogurt sligth
Gelatina Frutiño pack
Kumis Alpina
Papel Zeux aluminio esterilizado
Aguardiente Nectar azul sin azúcar
Producto

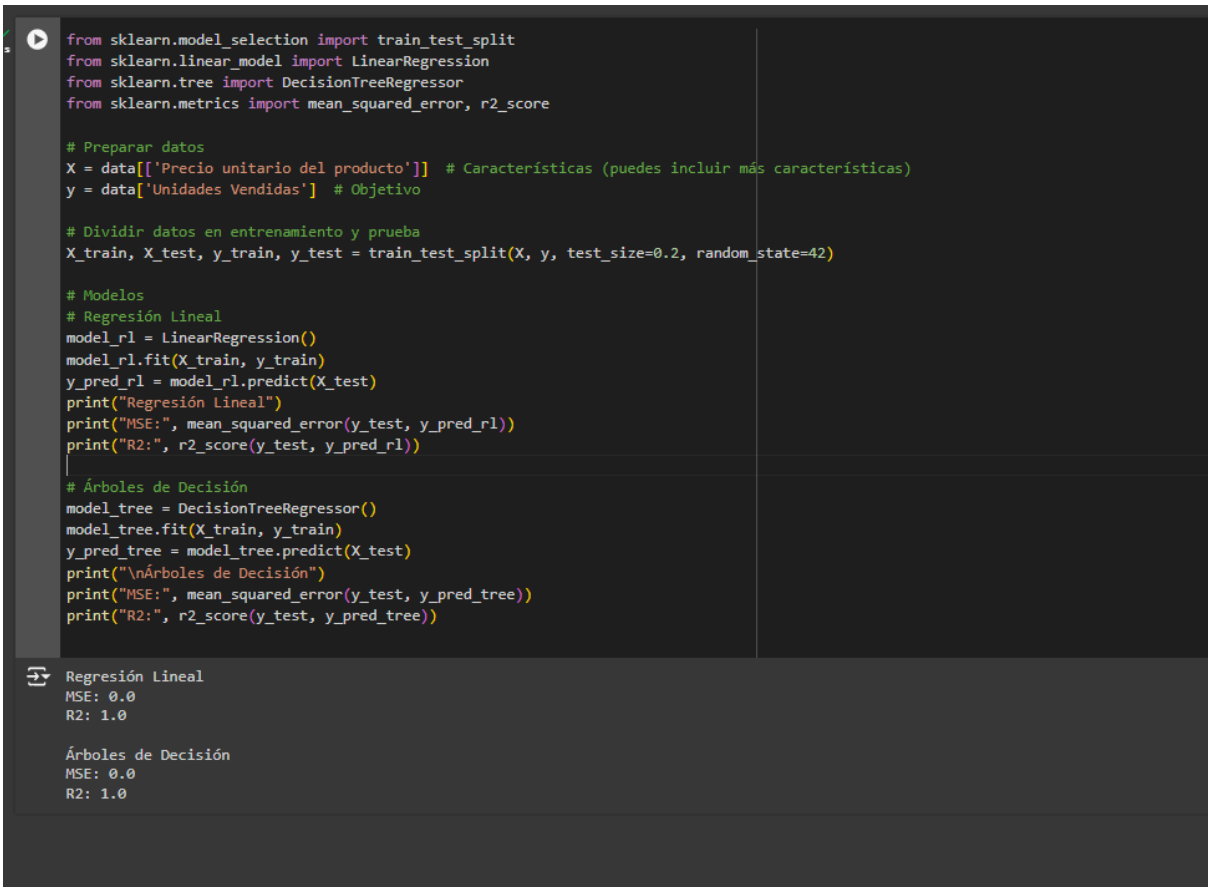


3. Implementación de Modelos de Aprendizaje Automático

Propósito: Utilizar técnicas de aprendizaje automático para construir modelos predictivos que estimen la demanda de productos.

Descripción:

- **Modelo de Regresión Lineal (RL):** Ajustar un modelo de regresión lineal para predecir la demanda basada en características como el precio unitario.
- **Modelo de Regresión Logística (RLog):** Si la variable objetivo es categórica (por ejemplo, alta/ baja demanda), aplicar un modelo de regresión logística.
- **Árboles de Decisión:** Implementar un modelo de árboles de decisión para capturar relaciones no lineales entre las características y la demanda.



```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Preparar datos
X = data[['Precio unitario del producto']]  # Características (puedes incluir más características)
y = data['Unidades Vendidas']  # Objetivo

# Dividir 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)

# Modelos
# Regresión Lineal
model_rl = LinearRegression()
model_rl.fit(X_train, y_train)
y_pred_rl = model_rl.predict(X_test)
print("Regresión Lineal")
print("MSE:", mean_squared_error(y_test, y_pred_rl))
print("R2:", r2_score(y_test, y_pred_rl))

# Árboles de Decisión
model_tree = DecisionTreeRegressor()
model_tree.fit(X_train, y_train)
y_pred_tree = model_tree.predict(X_test)
print("\nÁrboles de Decisión")
print("MSE:", mean_squared_error(y_test, y_pred_tree))
print("R2:", r2_score(y_test, y_pred_tree))
```

```
Regresión Lineal
MSE: 0.0
R2: 1.0

Árboles de Decisión
MSE: 0.0
R2: 1.0
```

4. Implementación de Modelos de Aprendizaje Automático

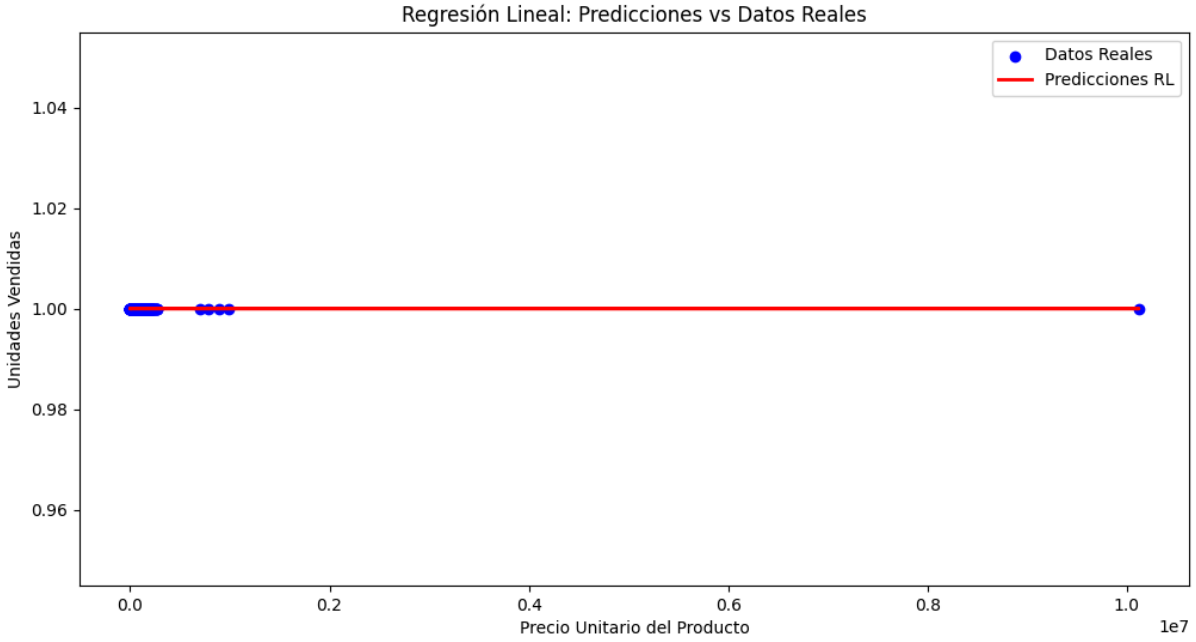
Propósito: implementa y entrena los modelos de Regresión Lineal y Árboles de Decisión, y realiza predicciones.

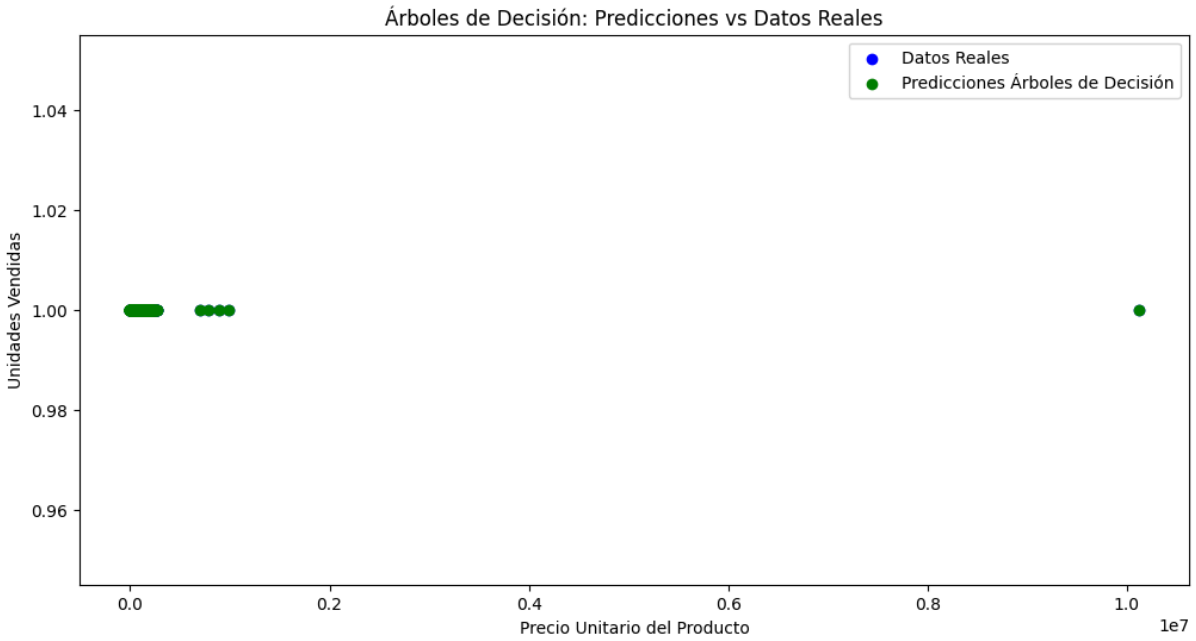
Descripción:

- **Regresión Lineal (RL):** Idealmente, debería tener un MSE bajo y un R2 alto. Si obtienes MSE de 0 y R2 de 1.0, puede indicar un problema con el conjunto de datos (por ejemplo, falta de variabilidad o datos demasiado simples).
- **Árboles de Decisión:** También deberías buscar un MSE bajo y un R2 alto. Los árboles de decisión pueden manejar relaciones no lineales y complejas mejor que la regresión lineal en algunos casos.

```
Comparación de Modelos de Regresión:
                Modelo  MSE   R2
0     Regresión Lineal  0.0  1.0
1  Árboles de Decisión  0.0  1.0

El mejor modelo de regresión es: Regresión Lineal
```





Conclusión

1. **Regresión Lineal:** Suele ser adecuada si hay una relación lineal entre las características y la variable objetivo. Puede ser sensible a valores atípicos y no manejar bien relaciones no lineales.
2. **Árboles de Decisión:** Pueden capturar relaciones no lineales y son más flexibles. Sin embargo, pueden ser propensos a sobreajustarse (overfitting) si no se ajustan adecuadamente.