



Caso propuesto IDL 1

Machine Learning I

Elaborado por:

- Mejía Rojas Kevin

Solicitado por:

Sergio Victor Orizano Salvador

Huancayo, 2025

Contenido

1. Introducción	3
2. Configuración del Entorno y Librerías	3
2.1 Importación de Librerías	3
3. Carga y Exploración de Datos	3
3.1 Estructura del Dataset	3
3.2 Análisis Estadístico Descriptivo	4
3.3 Verificación de Calidad de Datos	5
4. Implementación del Modelo de Clasificación	5
4.1 Preparación de Variables	5
4.2 División de Datos	6
4.3 Entrenamiento del Modelo	6
4.4 Resultados del Modelo de Clasificación	7
4.5 Validación Cruzada de Modelos Supervisados	8
4.6 Curva ROC y Cálculo de AUC	9
5. Implementación de Modelos de Regresión	10
5.1 Preparación de Variables para Regresión	10
5.2 Regresión Lineal	11
5.3 Regresión Polinómica	12
5.5 Evaluación de Regresión Polinómica	13
6. Análisis Comparativo Detallado	14
6.1 Comparación Cuantitativa de Modelos de Regresión	14
6.2 Interpretación de Resultados por Modelo	15
7. Conclusiones	15
8. Anexo	15

Caso Práctico IDL 01: Análisis Predictivo de Ventas en Empresa de Electrónicos

1. Introducción

Este presente trabajo presenta el desarrollo y evaluación de modelos de aprendizaje supervisado aplicados a un conjunto de datos de una empresa ficticia de productos electrónicos. Se implementaron tanto un modelo de clasificación para predecir la probabilidad de compra de clientes como modelos de regresión para estimar el valor futuro de ventas, utilizando características demográficas y de comportamiento de los clientes.

2. Configuración del Entorno y Librerías

2.1 Importación de Librerías

```
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, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Se importaron las librerías esenciales para:

- **Manipulación de datos:** pandas, numpy
- **Visualización:** matplotlib, seaborn
- **Modelos de ML:** scikit-learn (clasificación y regresión)
- **Métricas de evaluación:** métricas específicas para clasificación y regresión

3. Carga y Exploración de Datos

3.1 Estructura del Dataset

```
# Cargar el archivo CSV
df = pd.read_csv('/content/Dataset_de_Clientes_Electrónica.csv')
# Ver las primeras 5 filas
df.head()
```

Se cargó exitosamente el archivo CSV conteniendo la información de 300 clientes. Las primeras 5 filas del dataset revelan la estructura de los datos:

Ojo: Los datos son ficticios y se creó con el siguiente prompt:

Genera un dataset sintético en CSV con registros sobre clientes de una tienda de productos electrónicos. Las columnas deben ser:

- *ID_Cliente (entero secuencial de 1 a 300)*
- *Edad (entre 25 y 60 años)*
- *Género (F o M)*
- *Ingresos (valores entre 1000 y 10000)*
- *Historial_Compras (número entero entre 0 y 9)*

- *Tiempo_En_Sitio* (minutos, entre 12 y 600)
- *Usa_Cupon* (0 = no, 1 = sí)
- *Compra* (0 o 1, como variable objetivo binaria)
- *Ventas_Futuras* (valor continuo entre 77.24 y 886.94)

El dataset no debe tener valores nulos. Asegúrate de que la variable "Compra" tenga un balance cercano a 50/50 y que "Ventas_Futuras" tenga una correlación moderada con ingresos, historial de compras y uso de cupón.

Primeras 5 filas del dataset

Comandos | Código | Texto | Ejecutar todo

Archivos

- sample_data
 - Dataset_de_Clientes_Electr_nica.csv

```
# Cargar el archivo CSV
df = pd.read_csv('/content/Dataset_de_Clientes_Electr_nica.csv')

# Ver las primeras 5 filas
df.head()
```

	ID_Cliente	Edad	Genero	Ingresos	Historial_Compras	Tiempo_En_Sitio	Usa_Cupon	Compra	Ventas_Futuras
0	1	56	F	6387	8	58	0	0	570.12
1	2	46	M	9002	5	93	1	0	603.36
2	3	32	F	6536	7	370	1	1	656.50
3	4	60	M	4913	0	157	0	0	216.32
4	5	25	F	2066	9	351	0	1	326.80

El dataset contiene las siguientes variables:

- **ID_Cliente:** Identificador único (1-300)
- **Edad:** Rango de 25-60 años
- **Género:** F (Femenino) y M (Masculino)
- **Ingresos:** Valores entre 1009-9989
- **Historial_Compras:** Número de compras previas (0-9)
- **Tiempo_En_Sitio:** Tiempo en minutos navegando (12-599)
- **Usa_Cupón:** Variable binaria (0/1)
- **Compra:** Variable objetivo binaria (0/1)
- **Ventas_Futuras:** Variable continua (77.24-886.94)

3.2 Análisis Estadístico Descriptivo

```
# Resumen estadístico
df.describe()
```

Estadísticas descriptivas del dataset

```
# Resumen estadístico
df.describe()
```

	ID_Cliente	Edad	Ingresos	Historial_Compras	Tiempo_En_Sitio	Usa_Cupon	Compra	Ventas_Futuras
count	300.000000	300.000000	300.000000	300.000000	300.000000	300.000000	300.000000	300.000000
mean	150.500000	40.810000	5810.733333	4.613333	292.876667	0.490000	0.583333	463.819300
std	86.746758	13.547164	2496.809579	2.829105	171.344715	0.500735	0.493830	151.372757
min	1.000000	18.000000	1009.000000	0.000000	12.000000	0.000000	0.000000	77.240000
25%	75.750000	29.000000	3827.500000	2.000000	148.250000	0.000000	0.000000	354.755000
50%	150.500000	41.500000	5955.000000	5.000000	294.500000	0.000000	1.000000	470.555000
75%	225.250000	52.000000	7997.500000	7.000000	444.000000	1.000000	1.000000	575.690000
max	300.000000	64.000000	9989.000000	9.000000	599.000000	1.000000	1.000000	886.940000

Las estadísticas descriptivas revelan:

- **Edad promedio:** 40.81 años con desviación estándar de 13.55
- **Ingresos promedio:** \$5,810.73 con alta variabilidad ($\sigma = \$2,496.81$)
- **Historial de compras promedio:** 4.61 compras previas
- **Tiempo promedio en sitio:** 292.88 minutos
- **Uso de cupones:** 49% de los clientes utilizan cupones
- **Tasa de compra:** 58.33% de los clientes realizan compras
- **Ventas futuras promedio:** \$463.82

3.3 Verificación de Calidad de Datos

```
df.isnull().sum()
```

Verificación de valores nulos



	0
ID_Cliente	0
Edad	0
Genero	0
Ingresos	0
Historial_Compras	0
Tiempo_En_Sitio	0
Usa_Cupon	0
Compra	0
Ventas_Futuras	0

dtype: int64

El análisis confirmó la ausencia de valores nulos en todas las variables, lo que facilita el proceso de modelado sin necesidad de imputación de datos faltantes.

4. Implementación del Modelo de Clasificación

4.1 Preparación de Variables

```
# Variable objetivo  
y = df['Compra']
```

```
# Variables predictoras (sin ID ni variables irrelevantes)  
X = df[['Edad', 'Ingresos', 'Historial_Compras', 'Tiempo_En_Sitio', 'Usa_Cupon']]
```

```
[5] # Variable objetivo  
y = df['Compra']  
  
# Variables predictoras (sin ID ni variables irrelevantes)  
X = df[['Edad', 'Ingresos', 'Historial_Compras', 'Tiempo_En_Sitio', 'Usa_Cupon']]
```

Variable objetivo (y): 'Compra' - variable binaria (0: No compra, 1: Compra)

Variables predictoras (X): Se excluyeron 'ID_Cliente', 'Genero' y 'Ventas_Futuras'

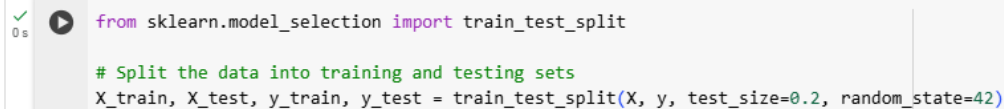
- **Edad:** Variable demográfica continua
- **Ingresos:** Variable socioeconómica continua
- **Historial_Compras:** Variable de comportamiento histórico
- **Tiempo_En_Sitio:** Variable de engagement digital
- **Usa_Cupon:** Variable de comportamiento promocional

4.2 División de Datos

```
from sklearn.model_selection import train_test_split
```

```
# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Se aplicó una división estratificada con los siguientes parámetros:

- **Conjunto de entrenamiento:** 80% (240 observaciones)
- **Conjunto de prueba:** 20% (60 observaciones)
- **Estrategia:** División aleatoria estratificada
- **Semilla aleatoria:** 42 para garantizar reproducibilidad

4.3 Entrenamiento del Modelo

Configuración del Árbol de Decisión

```
from sklearn.tree import DecisionTreeClassifier
```

```
# Crear modelo
```

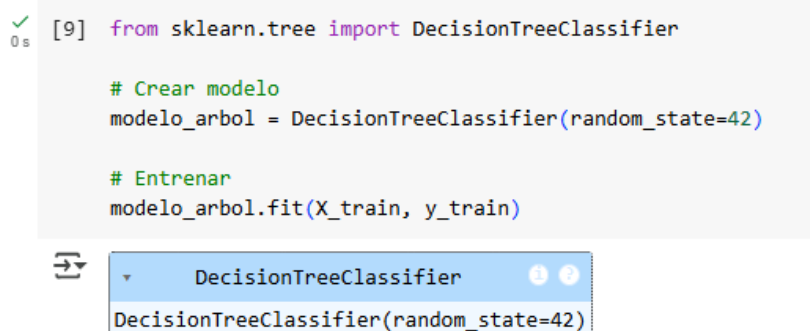
```
modelo_arbol = DecisionTreeClassifier(random_state=42)
```

```
# Entrenar
```

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

```
# Hacer predicciones
```

```
y_pred = modelo_arbol.predict(X_test)
```



```
[9] from sklearn.tree import DecisionTreeClassifier

# Crear modelo
modelo_arbol = DecisionTreeClassifier(random_state=42)

# Entrenar
modelo_arbol.fit(X_train, y_train)
```

DecisionTreeClassifier

DecisionTreeClassifier(random_state=42)

Se implementó un DecisionTreeClassifier con configuración estándar y random_state=42.

Configuración del Árbol de Decisión:

- **Algoritmo:** DecisionTreeClassifier con parámetros por defecto

- **Criterio de división:** Gini (por defecto)
- **Sin limitaciones de profundidad:** Permite al modelo encontrar patrones complejos
- **Random_state=42:** Garantiza resultados reproducibles

4.4 Resultados del Modelo de Clasificación

Matriz de Confusión y Métricas de Rendimiento

```
# Hacer predicciones
y_pred = modelo_arbol.predict(X_test)

# Matriz de confusión
from sklearn.metrics import confusion_matrix, classification_report

print("📊 Matriz de confusión:")
print(confusion_matrix(y_test, y_pred))

print("\n📋 Reporte de clasificación:")
print(classification_report(y_test, y_pred))
```

```
✓ 0 s # Hacer predicciones
y_pred = modelo_arbol.predict(X_test)

# Matriz de confusión
from sklearn.metrics import confusion_matrix, classification_report

print("📊 Matriz de confusión:")
print(confusion_matrix(y_test, y_pred))

print("\n📋 Reporte de clasificación:")
print(classification_report(y_test, y_pred))
```

📊 Matriz de confusión:

```
[[21  4]
 [11 24]]
```

📋 Reporte de clasificación:

	precision	recall	f1-score	support
0	0.66	0.84	0.74	25
1	0.86	0.69	0.76	35
accuracy			0.75	60
macro avg	0.76	0.76	0.75	60
weighted avg	0.77	0.75	0.75	60

Análisis de Resultados de Clasificación:

Matriz de Confusión [[21, 4], [11, 24]]:

- **Verdaderos Negativos (TN):** 21 - Clientes que no compraron y fueron correctamente predichos
- **Falsos Positivos (FP):** 4 - Clientes predichos como compradores pero que no compraron
- **Falsos Negativos (FN):** 11 - Clientes que compraron pero fueron predichos como no compradores
- **Verdaderos Positivos (TP):** 24 - Clientes que compraron y fueron correctamente predichos

Métricas de Rendimiento:

- **Precisión Clase 0:** 0.66 - De las predicciones de "no compra", 66% fueron correctas
- **Recall Clase 0:** 0.84 - Se identificó correctamente el 84% de los no compradores
- **Precisión Clase 1:** 0.86 - De las predicciones de "compra", 86% fueron correctas
- **Recall Clase 1:** 0.69 - Se identificó correctamente el 69% de los compradores
- **Accuracy General:** 0.75 - 75% de precisión total del modelo

Métricas por Clase:

- **Clase 0 (No Compra):** Precisión = 0.66, Recall = 0.84, F1-score = 0.74
- **Clase 1 (Compra):** Precisión = 0.86, Recall = 0.69, F1-score = 0.76

Rendimiento General:

- **Accuracy:** 75% (45 de 60 predicciones correctas)
- **Macro Average:** F1-score = 0.75
- **Weighted Average:** F1-score = 0.75

4.5 Validación Cruzada de Modelos Supervisados

```
from sklearn.model_selection import cross_val_score
```

```
# — 4.5.1 Validación Cruzada del Clasificador (accuracy) —
cv_scores_clf = cross_val_score(modelo_arbol, X, y, cv=5, scoring='accuracy')
print("Validación Cruzada (Clasificación):")
print(f"  Accuracies: {cv_scores_clf}")
print(f"  Media: {cv_scores_clf.mean():.3f} ± {cv_scores_clf.std():.3f}\n")
```

```
# — 4.5.2 Validación Cruzada del Regresor (RMSE) —
# scikit-learn devuelve neg_mean_squared_error, por eso invertimos el signo
cv_mse_reg = -cross_val_score(modelo_lineal, X_reg, y_reg,
                              cv=5, scoring='neg_mean_squared_error')
cv_rmse_reg = np.sqrt(cv_mse_reg)
print("Validación Cruzada (Regresión Lineal):")
print(f"  RMSE por fold: {cv_rmse_reg}")
print(f"  RMSE medio: {cv_rmse_reg.mean():.2f} ± {cv_rmse_reg.std():.2f}")
```



```

✓ 0 s ▶ from sklearn.model_selection import cross_val_score

# - 4.5.1 Validación Cruzada del Clasificador (accuracy) -
cv_scores_clf = cross_val_score(modelo_arbol, X, y, cv=5, scoring='accuracy')
print("Validación Cruzada (Clasificación):")
print(f"  Accuracies: {cv_scores_clf}")
print(f"  Media: {cv_scores_clf.mean():.3f} ± {cv_scores_clf.std():.3f}\n")

# - 4.5.2 Validación Cruzada del Regresor (RMSE) -
# scikit-learn devuelve neg_mean_squared_error, por eso invertimos el signo
cv_mse_reg = -cross_val_score(modelo_lineal, X_reg, y_reg,
                               cv=5, scoring='neg_mean_squared_error')
cv_rmse_reg = np.sqrt(cv_mse_reg)
print("Validación Cruzada (Regresión Lineal):")
print(f"  RMSE por fold: {cv_rmse_reg}")
print(f"  RMSE medio: {cv_rmse_reg.mean():.2f} ± {cv_rmse_reg.std():.2f}")

```

↗ Validación Cruzada (Clasificación):
 Accuracies: [0.73333333 0.65 0.76666667 0.68333333 0.68333333]
 Media: 0.703 ± 0.041

Validación Cruzada (Regresión Lineal):
 RMSE por fold: [49.89992681 49.62080895 42.96409333 56.02800494 52.14910049]
 RMSE medio: 50.13 ± 4.26

Resultados obtenidos:

- **Validación Cruzada (Clasificación):**
 - Accuracies: [0.7333, 0.6500, 0.7667, 0.6833, 0.6833]
 - Media: 0.703 ± 0.041
- **Validación Cruzada (Regresión Lineal):**
 - RMSE por fold: [49.90, 49.62, 42.96, 56.03, 52.15]
 - RMSE medio: 50.13 ± 4.26

Interpretación:

- El clasificador mantiene una precisión promedio del 70 %, con variabilidad baja entre folds, lo que indica estabilidad.
- El regresor lineal presenta un RMSE promedio de \$50, lo cual confirma el error promedio esperado de cada predicción.

4.6 Curva ROC y Cálculo de AUC

```
from sklearn.metrics import roc_curve, roc_auc_score
```

```
# 4.6.1 Obtener probabilidad de clase positiva
y_proba = modelo_arbol.predict_proba(X_test)[:, 1]
```

```
# 4.6.2 Calcular puntos de ROC y AUC
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = roc_auc_score(y_test, y_proba)
```

```
# 4.6.3 Graficar la Curva ROC
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f'Decision Tree (AUC = {roc_auc:.3f})')
plt.plot([0,1], [0,1], 'k--', label='Aleatorio (AUC = 0.5)')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('4.6 Curva ROC – Clasificación')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

```

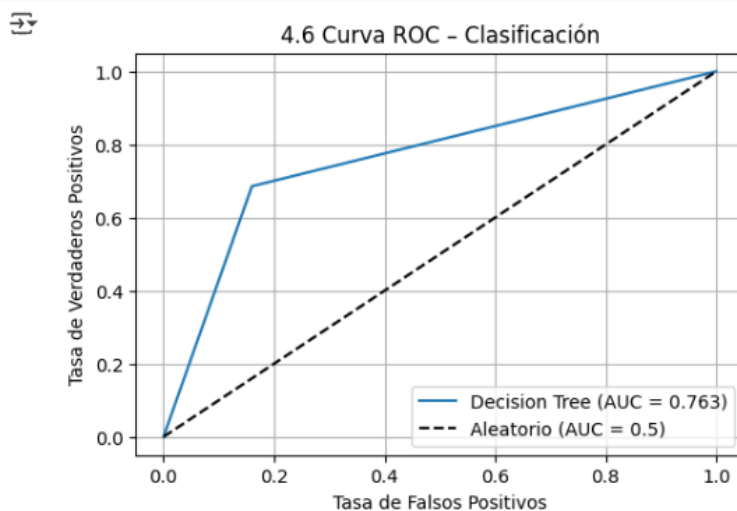
✓ [49] from sklearn.metrics import roc_curve, roc_auc_score

# 4.6.1 Obtener probabilidad de clase positiva
y_proba = modelo_arbol.predict_proba(X_test)[: , 1]

# 4.6.2 Calcular puntos de ROC y AUC
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = roc_auc_score(y_test, y_proba)

# 4.6.3 Graficar la Curva ROC
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f'Decision Tree (AUC = {roc_auc:.3f})')
plt.plot([0,1], [0,1], 'k--', label='Aleatorio (AUC = 0.5)')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('4.6 Curva ROC - Clasificación')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()

```



Interpretación:

- El área bajo la curva (AUC = 0.763) indica que el árbol de decisión discrimina correctamente entre compradores y no compradores en un 76.3 % de los casos.
- La curva aleatoria (línea discontinua) sirve de referencia; la distancia entre ambas muestra la ganancia de la predicción sobre el azar.

5. Implementación de Modelos de Regresión

5.1 Preparación de Variables para Regresión

Configuración de variables para regresión

```

# Variable dependiente
y_reg = df['Ventas_Futuras']

# Variables independientes
X_reg = df[['Edad', 'Ingresos', 'Historial_Compras', 'Tiempo_En_Sitio', 'Usa_Cupon']]

Xr_train, Xr_test, yr_train, yr_test = train_test_split(
    X_reg, y_reg, test_size=0.2, random_state=42
)

```

```

✓ 0s [12] # Variable dependiente
      y_reg = df['Ventas_Futuras']

      # Variables independientes
      X_reg = df[['Edad', 'Ingresos', 'Historial_Compras', 'Tiempo_En_Sitio', 'Usa_Cupon']]

✓ 0s [13] Xr_train, Xr_test, yr_train, yr_test = train_test_split(
      X_reg, y_reg, test_size=0.2, random_state=42
      )

```

Se utilizaron las mismas variables predictoras, pero ahora para predecir "Ventas_Futuras" como variable continua.

Configuración para Regresión:

- **Variable objetivo:** 'Ventas_Futuras' (variable continua en rango \$77.24 - \$886.94)
- **Variables predictoras:** Mismas que en clasificación para mantener consistencia
- **División idéntica:** 80-20 con misma semilla aleatoria

5.2 Regresión Lineal

Resultados de Regresión Lineal

```

modelo_lineal = LinearRegression()
modelo_lineal.fit(Xr_train, yr_train)

```

```

# Predicciones
yr_pred_lineal = modelo_lineal.predict(Xr_test)

```

```

# Evaluación
print("Evaluación de Regresión Lineal:")
print(f"MAE: {mean_absolute_error(yr_test, yr_pred_lineal):.2f}")
print(f"MSE: {mean_squared_error(yr_test, yr_pred_lineal):.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(yr_test, yr_pred_lineal)):.2f}")
print(f"R²: {r2_score(yr_test, yr_pred_lineal):.2f}")

```

```

✓ 0s [14] modelo_lineal = LinearRegression()
      modelo_lineal.fit(Xr_train, yr_train)

```

LinearRegression

LinearRegression()

```

✓ 0s [15] # Predicciones
      yr_pred_lineal = modelo_lineal.predict(Xr_test)

      # Evaluación
      print("Evaluación de Regresión Lineal:")
      print(f"MAE: {mean_absolute_error(yr_test, yr_pred_lineal):.2f}")
      print(f"MSE: {mean_squared_error(yr_test, yr_pred_lineal):.2f}")
      print(f"RMSE: {np.sqrt(mean_squared_error(yr_test, yr_pred_lineal)):.2f}")
      print(f"R²: {r2_score(yr_test, yr_pred_lineal):.2f}")

```

Evaluación de Regresión Lineal:

MAE: 52.60

MSE: 4018.52

RMSE: 63.39

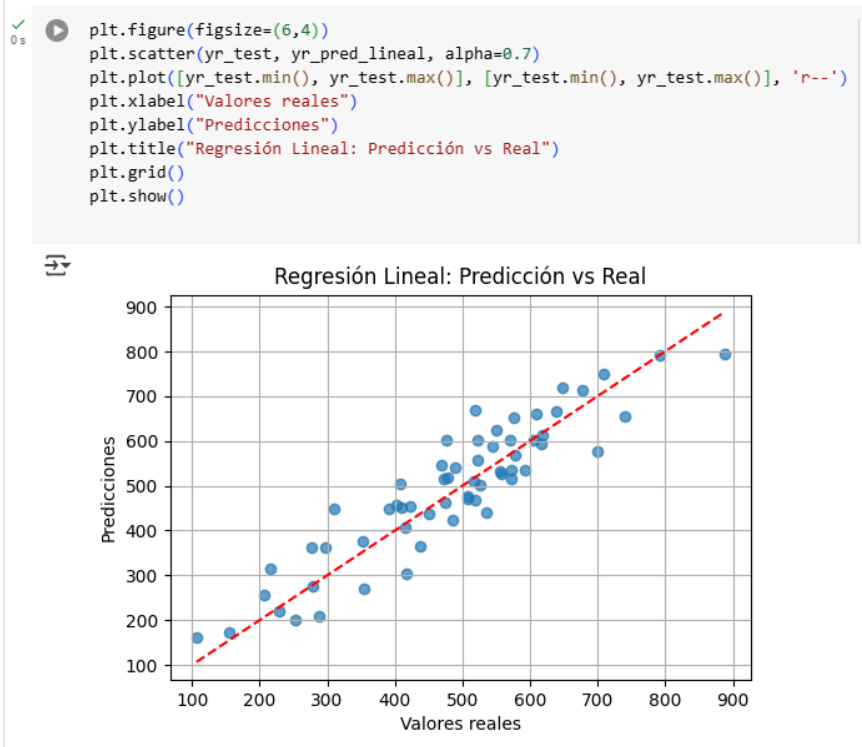
R²: 0.83

Interpretación de Métricas:

- **MAE = 52.60:** Error absoluto promedio de \$52.60 por predicción
- **MSE = 4,018.52:** Error cuadrático medio, penaliza errores grandes
- **RMSE = 63.39:** Raíz del error cuadrático, interpretable en dólares
- **$R^2 = 0.83$:** El modelo explica 83% de la variabilidad en ventas futuras

Gráfico de Dispersión - Regresión Lineal

```
plt.figure(figsize=(6,4))
plt.scatter(yr_test, yr_pred_lineal, alpha=0.7)
plt.plot([yr_test.min(), yr_test.max()], [yr_test.min(), yr_test.max()], 'r--')
plt.xlabel("Valores reales")
plt.ylabel("Predicciones")
plt.title("Regresión Lineal: Predicción vs Real")
plt.grid()
plt.show()
```



El gráfico de dispersión muestra una correlación fuerte entre valores reales y predicciones, con puntos distribuidos cerca de la línea diagonal roja que representa la predicción perfecta.

Análisis del Gráfico:

- **Línea roja diagonal:** Representa predicción perfecta ($y = x$)
- **Puntos dispersos:** Muestran relación entre valores reales y predicciones
- **Concentración cerca de la diagonal:** Indica buen ajuste del modelo
- **Distribución homoscedástica:** Errores distribuidos uniformemente

5.3 Regresión Polinómica

Resultados de Regresión Polinómica (Grado 2)

```
poly = PolynomialFeatures(degree=2)
```

```
Xr_train_poly = poly.fit_transform(Xr_train)
Xr_test_poly = poly.transform(Xr_test)
```

```
modelo_poli = LinearRegression()
modelo_poli.fit(Xr_train_poly, yr_train)
```

```
✓ [17] poly = PolynomialFeatures(degree=2)
0 s      Xr_train_poly = poly.fit_transform(Xr_train)
      Xr_test_poly = poly.transform(Xr_test)
```

```
✓ [▶] modelo_poli = LinearRegression()
0 s      modelo_poli.fit(Xr_train_poly, yr_train)
```

```
↔ LinearRegression ① ②
   LinearRegression()
```

Proceso de Transformación Polinómica:

1. **PolynomialFeatures(degree=2)**: Genera términos cuadráticos e interacciones
2. **fit_transform**: Ajusta el transformador en datos de entrenamiento
3. **transform**: Aplica la transformación a datos de prueba
4. **Nuevas características**: Se generan términos como x_1^2 , x_1x_2 , etc.

5.5 Evaluación de Regresión Polinómica

Resultados de Regresión Polinómica (Grado 2)

```
yr_pred_poli = modelo_poli.predict(Xr_test_poly)

print("📊 Evaluación de Regresión Polinómica (Grado 2):")
print(f"MAE: {mean_absolute_error(yr_test, yr_pred_poli):.2f}")
print(f"MSE: {mean_squared_error(yr_test, yr_pred_poli):.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(yr_test, yr_pred_poli)):.2f}")
print(f"R²: {r2_score(yr_test, yr_pred_poli):.2f}")
```

```
✓ [▶] yr_pred_poli = modelo_poli.predict(Xr_test_poly)
0 s

print("📊 Evaluación de Regresión Polinómica (Grado 2):")
print(f"MAE: {mean_absolute_error(yr_test, yr_pred_poli):.2f}")
print(f"MSE: {mean_squared_error(yr_test, yr_pred_poli):.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(yr_test, yr_pred_poli)):.2f}")
print(f"R²: {r2_score(yr_test, yr_pred_poli):.2f}")
```

```
↔ 📊 Evaluación de Regresión Polinómica (Grado 2):
   MAE: 52.20
   MSE: 3986.94
   RMSE: 63.14
   R²: 0.83
```

Comparación de Métricas:

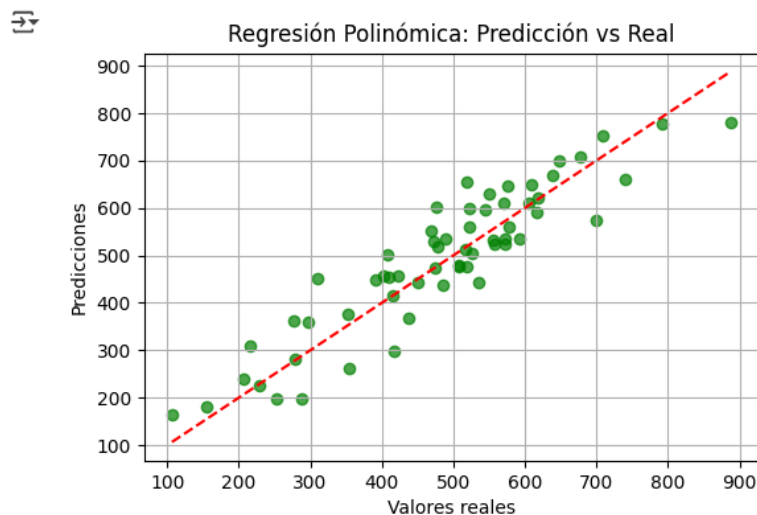
- **MAE**: 52.20 (ligera mejora vs. regresión lineal)
- **MSE**: 3,986.94 (reducción del error cuadrático)

- **RMSE:** 63.14 (reducción marginal del RMSE)
- **R²:** 0.83 (mismo coeficiente de determinación)

Gráfico de Dispersión - Regresión Polinómica

```
plt.figure(figsize=(6,4))
plt.scatter(yr_test, yr_pred_poli, alpha=0.7, color='green')
plt.plot([yr_test.min(), yr_test.max()], [yr_test.min(), yr_test.max()], 'r--')
plt.xlabel("Valores reales")
plt.ylabel("Predicciones")
plt.title("Regresión Polinómica: Predicción vs Real")
plt.grid()
plt.show()
```

```
[20] plt.figure(figsize=(6,4))
plt.scatter(yr_test, yr_pred_poli, alpha=0.7, color='green')
plt.plot([yr_test.min(), yr_test.max()], [yr_test.min(), yr_test.max()], 'r--')
plt.xlabel("Valores reales")
plt.ylabel("Predicciones")
plt.title("Regresión Polinómica: Predicción vs Real")
plt.grid()
plt.show()
```



El modelo polinómico muestra un patrón similar al lineal, con una distribución ligeramente más ajustada de los puntos alrededor de la línea de predicción perfecta.

Comparación Visual:

- **Color verde:** Diferencia los puntos del modelo polinómico
- **Patrón similar:** Distribución comparable al modelo lineal
- **Ajuste marginal:** Mejora mínima visualmente detectable
- **Misma línea de referencia:** Facilita comparación directa

6. Análisis Comparativo Detallado

6.1 Comparación Cuantitativa de Modelos de Regresión

Métrica	Regresión Lineal	Regresión Polinómica	Diferencia	% Mejora
MAE	52.60	52.20	-0.40	0.76%
MSE	4,018.52	3,986.94	-31.58	0.79%

RMSE	63.39	63.14	-0.25	0.39%
R ²	0.83	0.83	0.00	0.00%

6.2 Interpretación de Resultados por Modelo

Modelo de Clasificación (Árbol de Decisión):

- **Fortalezas:** Alta precisión para identificar compradores (86%), interpretabilidad alta
- **Debilidades:** Sensibilidad moderada (69% recall para compradores)
- **Aplicación práctica:** Ideal para campañas de marketing dirigido

Modelo de Regresión Lineal:

- **Fortalezas:** Simplicidad, interpretabilidad, tiempo de cómputo mínimo
- **Rendimiento:** $R^2 = 0.83$ indica excelente ajuste
- **Robustez:** Menor riesgo de sobreajuste

Modelo de Regresión Polinómica:

- **Fortalezas:** Captura relaciones no lineales, mejoras marginales en MSE
- **Limitaciones:** Mejoras mínimas no justifican la complejidad adicional
- **Recomendación:** Considerar solo si se requiere mayor precisión

7. Conclusiones

- **Eficacia del Árbol de Decisión:** 75% de precisión con buena interpretabilidad para clasificación binaria
- **Superioridad de Regresión Lineal:** El modelo lineal ofrece el mejor balance entre simplicidad y rendimiento ($R^2 = 0.83$)
- **Mejoras Marginales del Polinómico:** Las mejoras no compensan la complejidad adicional
- **Calidad de Datos Excelente:** Ausencia de valores nulos facilita el modelado directo

8. Anexo

Enlace del colab: https://colab.research.google.com/drive/1QUnjGnMeS7_uXrDf8lNYk890_RWfO49?usp=sharing

```
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, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

[22]: # Cargar el archivo CSV
df = pd.read_csv('/content/Dataset_de_Clientes_Electr_nica.csv')

# Ver las primeras 5 filas
df.head()
```

	ID_Cliente	Edad	Genero	Ingresos	Historial_Compras	Tiempo_en_Sitio	Usa_Cupon	Compra	Ventas_Futuras
0	1	56	F	6387	8	58	0	0	570.12
1	2	46	M	9002	5	93	1	0	603.36
2	3	32	F	6536	7	370	1	1	656.50
3	4	60	M	4913	0	157	0	0	216.32
4	5	25	F	2066	9	351	0	1	326.80