Actividad Final

Regresión Lineal Simple

Problema

Construir un modelo que haga predicciones de ventas basado en la invesión financiera de diferentes plataformas de publicidad.

Datos

Utiliza el conjunto de datos CSV del archivo advertising.csv y analiza la relacion que tienen las diferentes plataformas respecto a las ventas.

Leyendo y entendiendo los datos

```
# Importa pandas, numpy
import pandas as pd
import numpy as np

# Importa scikilearn y métricas
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Importa API's para visualización de datos
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Carga el archivo CSV a una dataframe (DF) y visualiza los primeros
df = pd.read_csv('advertising.csv')
print(df.head())
print()
```

```
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
```

```
# Visualiza la dimensión del DF
print(f'Dimensiones del dataset: {df.shape}')
print()
```

```
Dimensiones del dataset: (200, 4)
```

```
# Visualiza la información del DF
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
None
```

```
# Visualiza las estadísticas generales del DF
df.describe()
```

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

Limpiando los datos

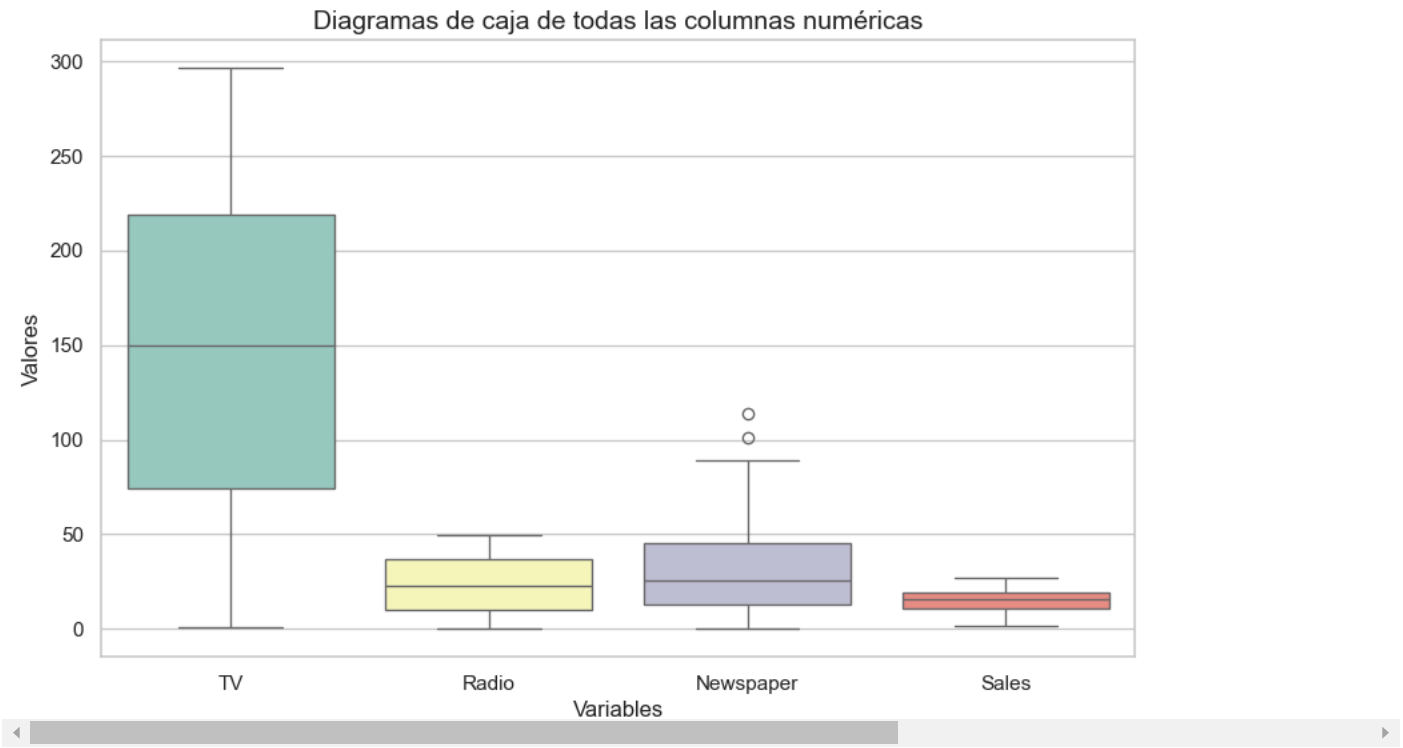
```
# Busca si existen valores nulos en el DF. Utiliza le método isnull() con la función de agregación sum()
# para saber la cantidad de nulos por cada variable del DF
print(df.isnull().sum())
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

Contesta en esta misma celda: ¿Cuál es tu opinión de acuerdo a los resultados anteriores, para realizar la limpieza de datos?

En este caso al no haber valores nulos no es necesario una limpieza pero en el caso de que hubieran podria usarse df.dropna() para eliminar los valores nulos

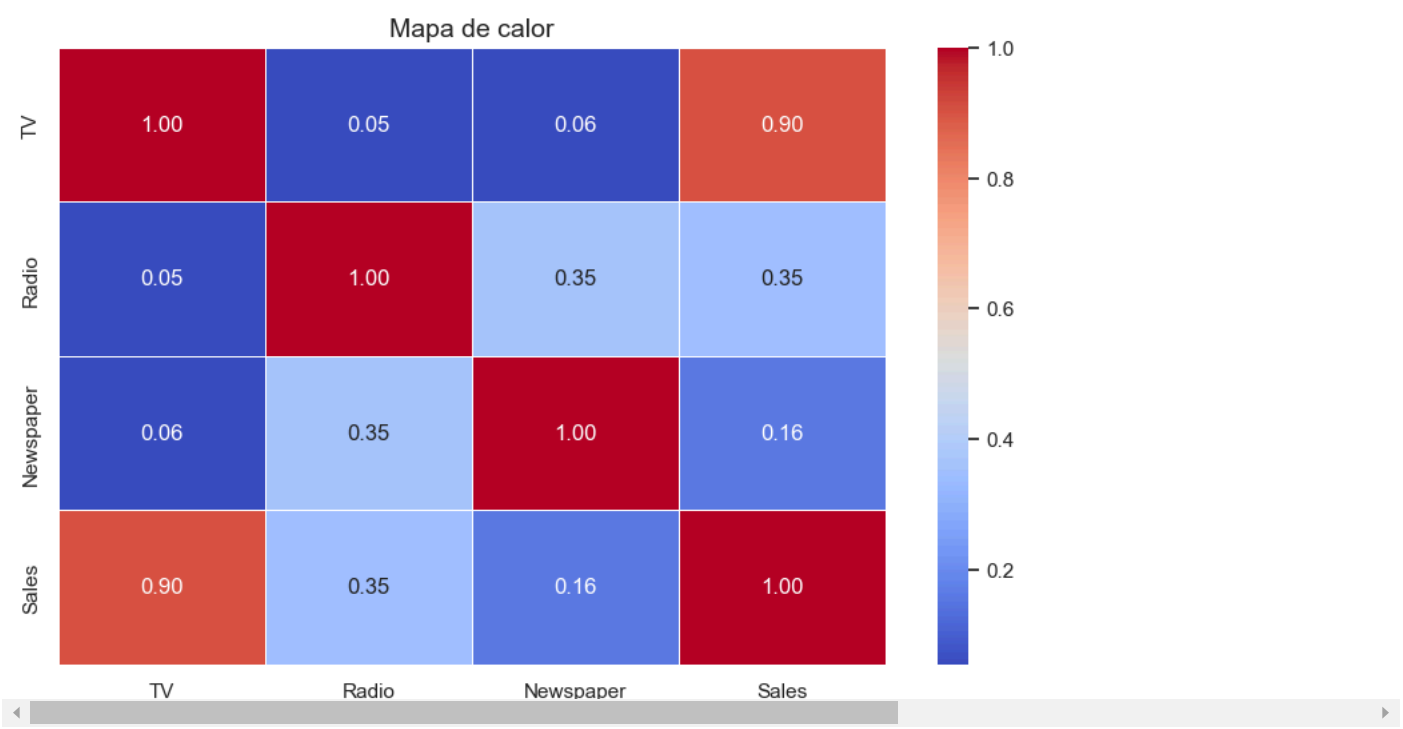
```
# Identifica si existen valor atípicos (outliers) en las 3 variables independientes que se estan estudiando. Utilza seaborn
# para realizar las gráficas de caja

# Configurar el estilo de Seaborn
sns.set_theme(style="whitegrid")

# Crear la figura
plt.figure(figsize=(10, 6))

# Dibujar los boxplots con Seaborn
sns.boxplot(data=df, palette="Set3")

# Configurar etiquetas y título
plt.title("Diagramas de caja de todas las columnas numéricas", fontsize=14)
plt.xlabel("Variables")
plt.ylabel("Valores")

# Mostrar la gráfica
plt.show()
```



Contesta en esta misma celda: ¿Existen valores atípicos que realmente afecten el desempeño del entrenamiento? ¿Cómo procederías de acuerdo a tu respuesta anterior?

Si, Existen valores atipicos y para tratar con ellos y que no afecten al momento del entrenamiento del modelo seria eliminarlos

```
# Crear un mapa de calor (heatmap) para visualizar la correlación entre las variables independietes y dependiente
# Utiliza seaborn para graficar

# Configurar el estilo de Seaborn
sns.set_theme(style="whitegrid")

# Crear la figura
plt.figure(figsize=(10, 6))

# Dibujar el mapa de calor
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

# Configurar título
plt.title("Mapa de calor", fontsize=14)

# Mostrar la gráfica
plt.show()
```



Contesta en esta misma celda: ¿Cuál es la variable que tiene mayor relación con las ventas?.

Es la variable TV

Constuyendo el modelo

Importa las librerias sklearn para el split de los datos y el modelo de regesión lineal

```
# Importa sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression


# Crea los conjutnos X y Y para le modelo de regresión lineal simple. Te recomiendo que uses la siguiente nomenclatura
# Y = df['Sales']
X = df['TV'].values
Y = df['Sales'].values



# Haz el reshape correspondiente al conjutno X antes de dividir los datos
X = X.reshape(-1,1)
print(X.shape)
print(X)

#dividir los datos en entrenamiento y prueba, considera el 20% de los datos para testing
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

```
(200, 1)
[[230.1]
 [ 44.5]
 [ 17.2]
 [151.5]
 [180.8]
 [  8.7]
 [ 57.5]
 [120.2]
 [  8.6]
 [199.8]
 [ 66.1]
 [214.7]
 [ 23.8]
 [ 97.5]
 [204.1]
```

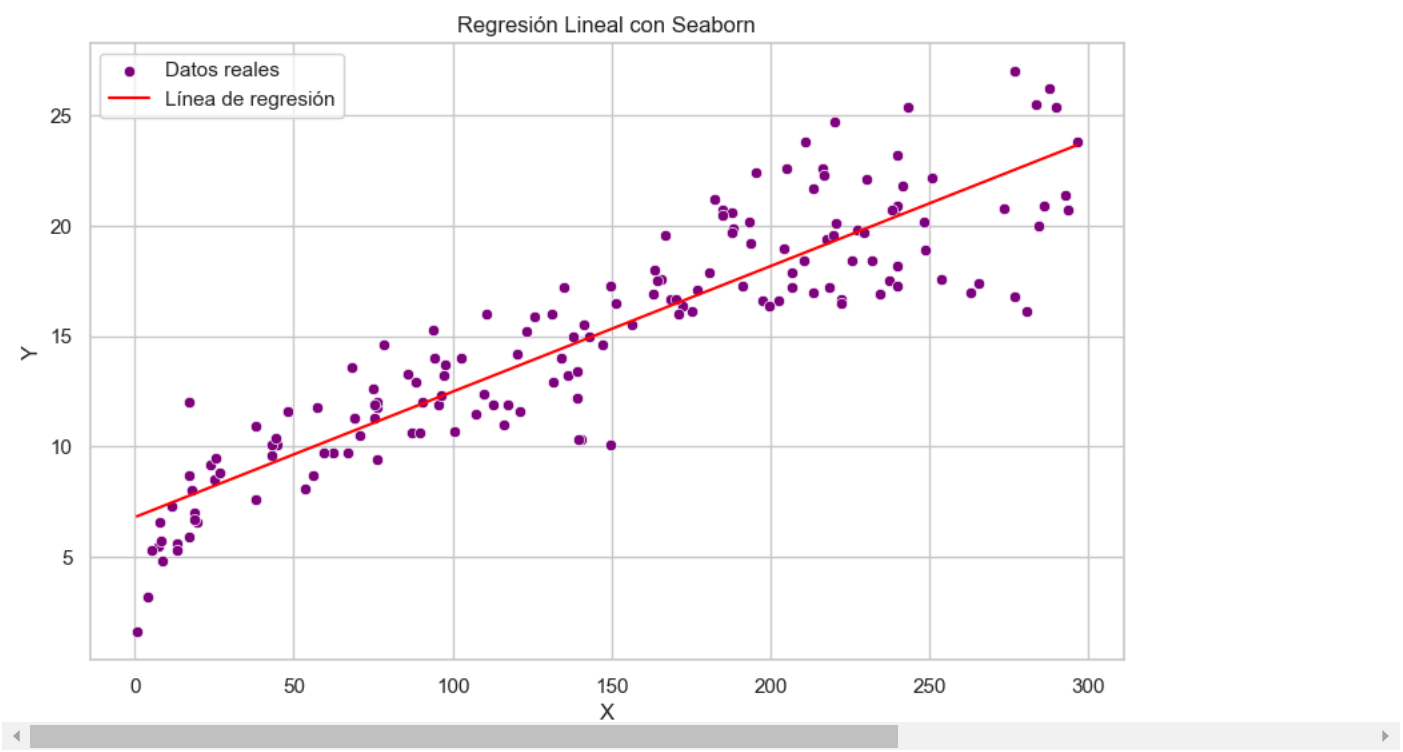
```
[195.4]
[ 67.8]
[281.4]
[ 69.2]
[147.3]
[218.4]
[237.4]
[ 13.2]
[228.3]
[ 62.3]
[262.9]
[142.9]
[240.1]
[248.8]
[ 70.6]
[292.9]
[112.9]
[ 97.2]
[265.6]
[ 95.7]
[290.7]
[266.9]
[ 74.7]
[ 43.1]
[228. ]
[202.5]
[177. ]
[293.6]
[206.9]
[ 25.1]
[175.1]
[ 89.7]
[239.9]
[227.2]
[ 66.9]
[199.8]
[100.4]
[216.4]
[182.6]
[262.7]
[198.9]
```

```
# Instancia el modelo LienarRegression()
reg = LinearRegression()
# Entrnea el modelo con el método fit y los conjutnos de entrenamiento (x_train, y_train)
reg.fit(X_train, Y_train)

# Imprime la pendiente, bias y coeficientes resultados del entrenamiento
print('La pendiente es: ', reg.coef_)
print('El bias es: ', reg.intercept_)
print('Coeficiente de determinacion: ', reg.score(X_train, Y_train))
```

```
La pendiente es:  [0.05686527]
El bias es:  6.7985830426488025
Coeficiente de determinacion:  0.8155707313837735
```

Visualiza el desempeño del entrenamiento

```
# Redimensiona el conjunto X de entrenamiento
x_flat = X_train.flatten()

# Realiza la prediccion sobre el conjunto X de entrenamiento
y_hat = reg.predict(X_train)

# Con seabor haz una grafica de dispersión para comparar los conjuntos
# Graficar regresión lineal con Seaborn
sns.set_theme(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.scatterplot(x=x_flat, y=Y_train, label='Datos reales', color='purple')  # Datos reales
sns.lineplot(x=x_flat, y=y_hat, color='red', label='Línea de regresión')  # Línea de regresión

plt.xlabel('X')  # Etiqueta del eje X
plt.ylabel('Y')  # Etiqueta del eje Y
plt.title('Regresión Lineal con Seaborn')  # Título del gráfico
plt.legend()  # Agregar leyenda
plt.show()  # Mostrar gráfico
```



Evaluación del modelo

```
# Importa las métricas mean_squared_error y r2_score
from sklearn.metrics import mean_squared_error, r2_score
```