Seany Campos Cortés 21690072

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

In [86]:
```
# Importa pandas, numpy
import pandas as pd  # Manejo de datos
import numpy as np   # Operaciones numéricas

# Importa scikilearn y métricas
import sklearn # Biblioteca de machine learning
from sklearn.model_selection import train_test_split #función train_test_split para dividir los datos en conjuntos de entrenamiento y prueba
from sklearn.linear_model import LinearRegression #modelo de regresión lineal
from sklearn.metrics import mean_squared_error #error cuadrático medio para evaluar el rendimiento del modelo

# Importa API's para visualización de datos
import matplotlib.pyplot as plt #visualización de datos en gráficos
import seaborn as sns #para gráficos estadísticos más estilizados
```

In [87]:
```
# Carga el archivo CSV a una dataframe (DF) y visualiza los primeros
df = pd.read_csv('advertising.csv')

##df.head() devuelve las primeras 5 filas del df
```

```
##print(df.head()) imprime las primeras filas
print(df.head())
```

```
      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
```

In [88]:
```
# Visualiza la dimensión del DF

#.shape devuelve una tupla (filas, columnas)
print(f'Dimensiones del dataset: {df.shape}')
```

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

In [89]:
```
# Visualiza la información del DF

## muestra un resumen del df, sobre las columnas, tipos de datos y valores no nulos
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
```

In [90]:
```
# Visualiza las estadísticas generales del DF
df.describe() ## se utiliza para obtener un resumen estadístico de las columnas
```

Out[90]:

	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

In [91]:
```
# 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

## df.isnull() devuelve un df donde los valores son nulos (NaN)
## sum() se utiliza para contar cuántos valores nulos existen en cada columna
print(df.isnull().sum()) # método to_string para visualizar todo el df
```

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

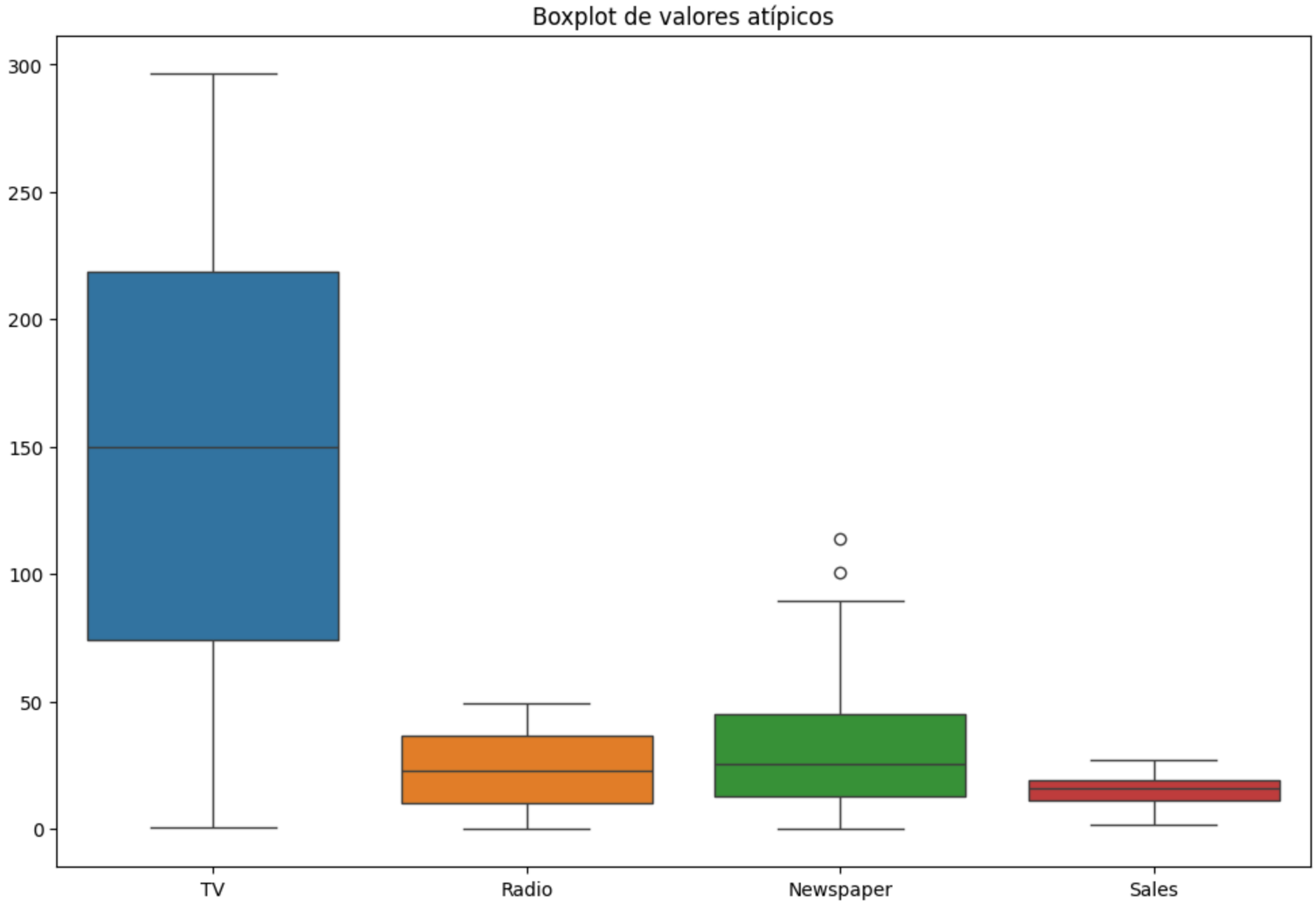
Contesta en esta misma celda: ¿Cuál es tu opinión de acuerdo a los resultados anteriores, para realizar la limpieza de datos?

Se puede ver que al parecer no hay valores nulos en las columnas del df, por lo tanto no es necesario hacer una limpieza de datos en cuanto a valores faltantes

In [ ]:
```

```

In [92]:
```
# Identifica si existen valor atípicos (outliers) en las 3 variables independientes que se estan estudiando. Utilza seaborn
# para realizar las gráficas de caja
```

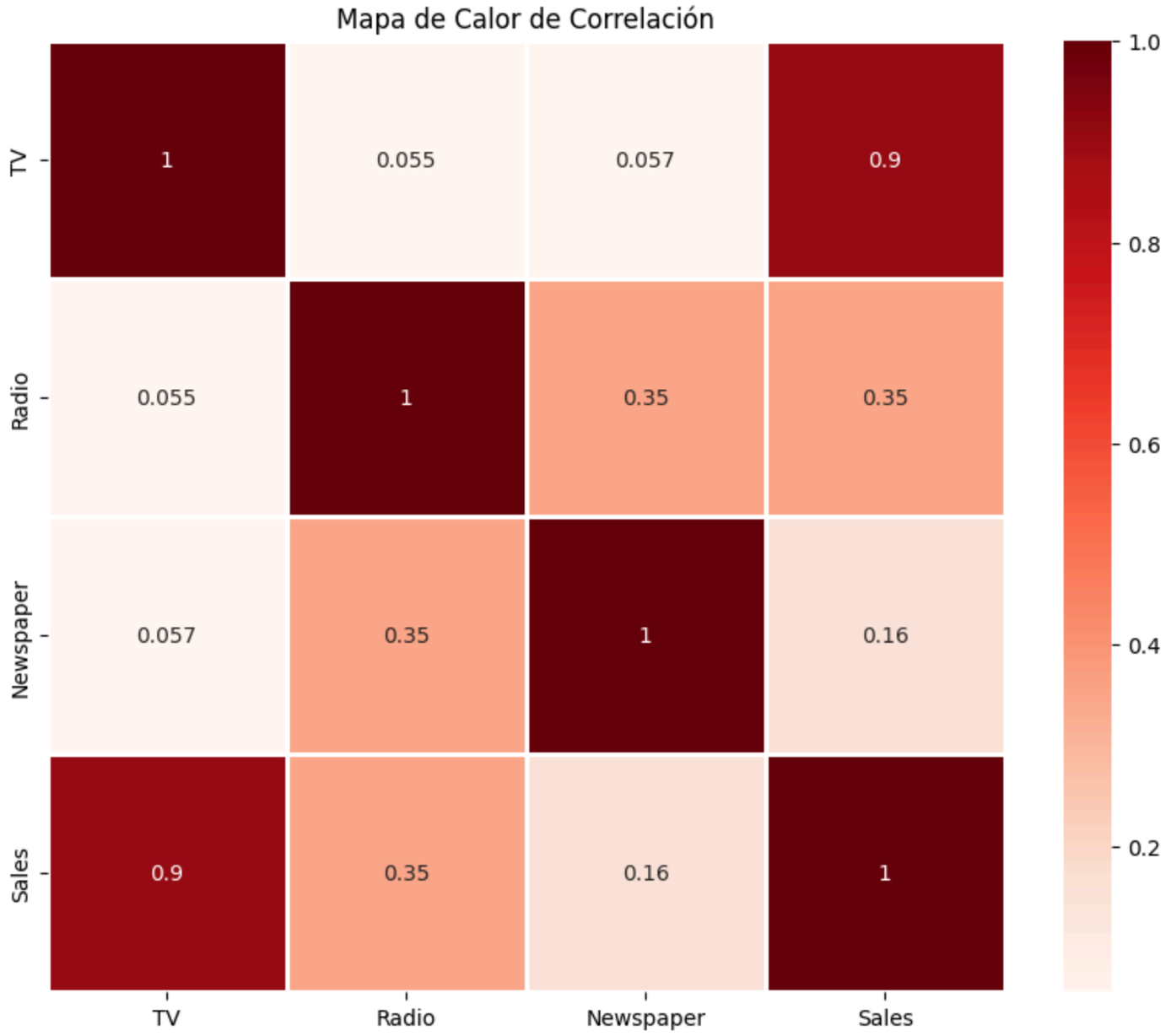
```
#figsize ajusta el tamaño de la figura del gráfico
plt.figure(figsize=(12, 8))

#genera grafica de caja para las columnas numéricas
#data le indica a seaborn que los datos que se van a graficar vienen del df
sns.boxplot(data=df)

# título de la gráfica
plt.title('Boxplot de valores atípicos')

# muestra la gráfica
plt.show()
```


Boxplot de valores atípicos
300
250
200
150
100
50
0
TV
Radio
Newspaper
Sales

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

La variable Newspaper tiene algunos valores atípicos a comparación a las demás variables, lo que podría cambiar los resultados. Pero si el modelo usa una regresión lineal se puede proceder eliminando o corregiendo esos valores para no tener problemas

In [93]:
```
# Crear un mapa de calor (heatmap) para visualizar la correlación entre las variables independientes y dependiente
# Utiliza seaborn para graficar

# Calcular la matriz de correlación

#Calcula la correlación entre todas las columnas numéricas
#
correlation_matrix = df.corr() # para determinar la relación entre las variables numéricas

# Graficar el mapa de calor
#annot=True muestra los valores los valores numéricos en cada celda
#cmap='Reds' es la paleta de colores rojiza
#linewidths=0.5 es el grosor de las líneas que separan las celdas

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='Reds', linewidths=0.9)
plt.title('Mapa de Calor de Correlación')
plt.show()
```


Mapa de Calor de Correlación
TV
Radio
Newspaper
Sales
1
0.055
0.057
0.9
0.055
1
0.35
0.35
0.057
0.35
1
0.16
0.9
0.35
0.16
1
1.0
0.8
0.6
0.4
0.2

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

La variable TV es la que tiene una mayor relación con las ventas ya que tiene un valor de 0.9, mientras que Radio tiene un valor de 0.35 y Newspaper tiene un valor de 0.16

Construyendo el modelo

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

In [94]:
```
# Importa sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [95]:
```
# Crea los conjuntos X y Y para el modelo de regresión lineal simple. Te recomiendo que uses la siguiente nomenclatura
# Y = df['Sales']

# se crea el conjunto X a partir de la columna TV
#X es variable independiente y contiene los valores del gasto en publicidad en TV
# .values se utiliza para obtener los valores del df como un array
X = df['TV'].values

# se crea el conjunto Y a partir de la columna Sales
# Y es la variable dependiente y contiene los valores de las ventas
# .values se utiliza para obtener los valores del df como un array
Y = df['Sales'].values
```

In [96]:
```
# Haz el reshape correspondiente al conjunto X antes de dividir los datos

# muestra las dimensiones del conjunto de datos X antes de realizar el reshape(cambiar la forma de un array)
print(X.shape)

# imprime el contenido del conjunto X
print(X)
```

```
(200,)
[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   7.3 136.2 210.8 210.7
  53.5 261.3 239.3 102.7 131.1  69.   31.5 139.3 237.4 216.8 199.1 109.8
  26.8 129.4 213.4  16.9  27.5 120.5   5.4 116.   76.4 239.8  75.3  68.4
 213.5 193.2  76.3 110.7  88.3 109.8 134.3  28.6 217.7 250.9 107.4 163.3
 197.6 184.9 289.7 135.2 222.4 296.4 280.2 187.9 238.2 137.9  25.   90.4
  13.1 255.4 225.8 241.7 175.7 209.6  78.2  75.1 139.2  76.4 125.7  19.4
 141.3  18.8 224.  123.1 229.5  87.2   7.8  80.2 220.3  59.6   0.7 265.2
   8.4 219.8  36.9  48.3  25.6 273.7  43.  184.9  73.4 193.7 220.5 104.6
  96.2 140.3 240.1 243.2  38.   44.7 280.7 121.  197.6 171.3 187.8   4.1
  93.9 149.8  11.7 131.7 172.5  85.7 188.4 163.5 117.2 234.5  17.9 206.8
 215.4 284.3  50.  164.5  19.6 168.4 222.4 276.9 248.4 170.2 276.7 165.6
 156.6 218.5  56.2 287.6 253.8 205.  139.5 191.1 286.   18.7  39.5  75.5
  17.2 166.8 149.7  38.2  94.2 177.  283.6 232.1]
```

In [97]:

```
#Cambia la forma de X para que sea un sola columna
# -1 para ajustar el número de filas automáticamente
# 1 indica que debe haber solo una columna

X = X.reshape(-1,1)

# muestra las dimensiones de X después de cambiar su forma, numero de filas y columnas que tiene

print(X.shape)

# se muestra los datos de X después de haber sido reorganizados, será  una sola columna con todos los valores
print(X)
```

```
(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]
[  7.3]
[136.2]
[210.8]
[210.7]
[ 53.5]
[261.3]
[239.3]
[102.7]
[131.1]
[ 69. ]
[ 31.5]
[139.3]
[237.4]
[216.8]
[199.1]
[109.8]
[ 26.8]
[129.4]
[213.4]
[ 16.9]
[ 27.5]
[120.5]
[  5.4]
[116. ]
[ 76.4]
[239.8]
[ 75.3]
```

```
[ 68.4]
[213.5]
[193.2]
[ 76.3]
[110.7]
[ 88.3]
[109.8]
[134.3]
[ 28.6]
[217.7]
[250.9]
[107.4]
[163.3]
[197.6]
[184.9]
[289.7]
[135.2]
[222.4]
[296.4]
[280.2]
[187.9]
[238.2]
[137.9]
[ 25. ]
[ 90.4]
[ 13.1]
[255.4]
[225.8]
[241.7]
[175.7]
[209.6]
[ 78.2]
[ 75.1]
[139.2]
[ 76.4]
[125.7]
[ 19.4]
[141.3]
[ 18.8]
[224. ]
[123.1]
[229.5]
```

```
[ 87.2]
[  7.8]
[ 80.2]
[220.3]
[ 59.6]
[  0.7]
[265.2]
[  8.4]
[219.8]
[ 36.9]
[ 48.3]
[ 25.6]
[273.7]
[ 43. ]
[184.9]
[ 73.4]
[193.7]
[220.5]
[104.6]
[ 96.2]
[140.3]
[240.1]
[243.2]
[ 38. ]
[ 44.7]
[280.7]
[121. ]
[197.6]
[171.3]
[187.8]
[  4.1]
[ 93.9]
[149.8]
[ 11.7]
[131.7]
[172.5]
[ 85.7]
[188.4]
[163.5]
[117.2]
[234.5]
[ 17.9]
```

```
[206.8]
[215.4]
[284.3]
[ 50. ]
[164.5]
[ 19.6]
[168.4]
[222.4]
[276.9]
[248.4]
[170.2]
[276.7]
[165.6]
[156.6]
[218.5]
[ 56.2]
[287.6]
[253.8]
[205. ]
[139.5]
[191.1]
[286. ]
[ 18.7]
[ 39.5]
[ 75.5]
[ 17.2]
[166.8]
[149.7]
[ 38.2]
[ 94.2]
[177. ]
[283.6]
[232.1]]
```

In [98]:
```python
#dividir los datos en entrenamiento y prueba, considera el 20% de los datos para testing

# train_test_split divide los datos en conjuntos de entrenamiento y prueba

#X_train, X_test son las entradas que tomará el modelo
#Y_train, Y_test son las salidas que el modelo va a predecir

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

```
# se muestra las dimensiones de los conjuntos de entrenamiento y prueba (filas  y columnas)

print('X_train:',X_train.shape) # muestra el tamaño de X_train (entradas de entrenamiento)
print('Y_train:',Y_train.shape) # muestra el tamaño de Y_train (salidas de entrenamiento)
print('X_test:',X_test.shape) # muestra el tamaño de X_test (entradas de prueba)
print('Y_test',Y_test.shape) # muestra el tamaño de Y_test (salidas de prueba)
```

```
X_train: (160, 1)
Y_train: (160,)
X_test: (40, 1)
Y_test (40,)
```

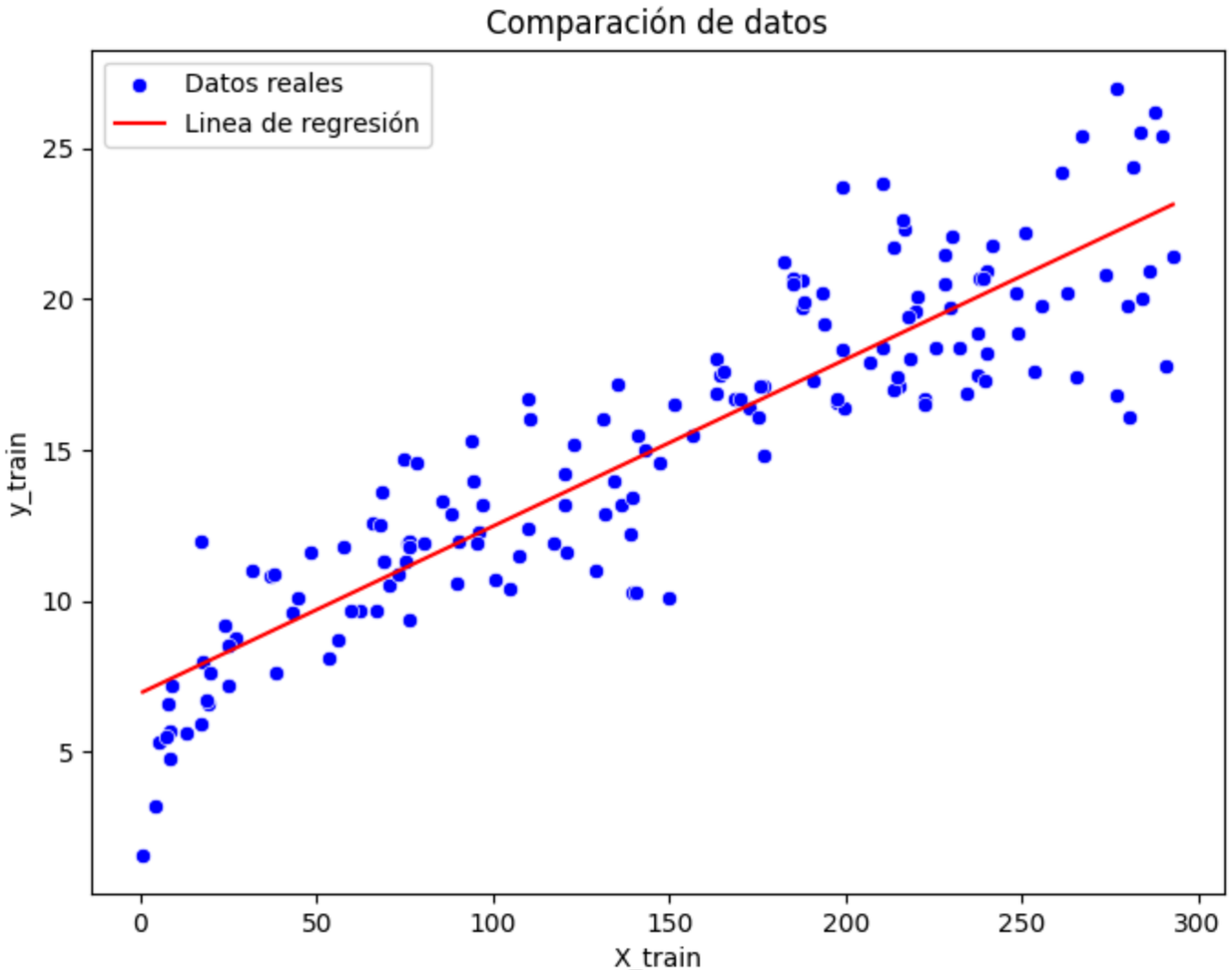
In [99]:
```
# Instancia el modelo LinearRegression()

# reg guarda el modelo de regresión lineal
# LinearRegression intenta encontrar la relación entre las entradas y lo que queremos predecir
reg = LinearRegression()

# Entrena el modelo con el método fit y los conjuntos de entrenamiento (x_train, y_train)

# fit para entrenar o ajustar el modelo con los datos que le vamos a dar
reg.fit(X_train, Y_train)

# Imprime la pendiente, bias y coeficientes resultados del entrenamiento

# reg.coef_ dice cuánto cambia la variable de ventas como 'Sales' cuando cambia una de las otras variables
print('La pendiente es: ', reg.coef_)

# es el valor inicial cuando no hay algun cambio en ninguna de las variables
print('El bias es: ', reg.intercept_)

# reg.score es qué tan bien el modelo está ajustado a los datos
print('Coeficiente de determinación: ', reg.score(X_train, Y_train))
```

```
La pendiente es:  [0.05525601]
El bias es:  6.947681941981351
Coeficiente de determinación:  0.8127188643286971
```

Visualiza el desempeño del entrenamiento

In [100…

```
# Redimensiona el conjunto X de entrenamiento

#X_train es un conjunto de datos de entrenamiento, es una sola columna
#flatten convierte esa columna en una lista
#x_flat es lista de los mismos datos de X_train, pero en una sola lista

x_flat = X_train.flatten()


# Realiza la prediccion sobre el conjunto X de entrenamiento

#reg es el modelo de regresión lineal que se habia entrenado
# predict para hacer una predicción sobre los datos de entrenamiento
# y_hat son los valores que el modelo predice

y_hat = reg.predict(X_train)

# Con seaborn haz una grafica de dispersión para comparar los conjuntos

# el tamaño de la gráfica
plt.figure(figsize=(8, 6))

# scatterplot es la gráfica de dispersión
# cada punto representa el gasto en TV  que es x_flat y las ventas con Y_train
sns.scatterplot(x=x_flat, y=Y_train, label="Datos reales", color="blue")

#lineplot que es la línea de regresión
# la linea representa lo que se predice que es y_hat en relacion del gasto en TV con x_flat
sns.lineplot(x=x_flat, y=y_hat, label="Linea de regresión", color="red")

plt.title("Comparación de datos")
plt.xlabel("X_train")
plt.ylabel("y_train")
plt.show()
```



Evaluación del modelo

In [101…
```
# Importa las métricas mean_squared_error y r2_score
from sklearn.metrics import mean_squared_error, r2_score
```

In [102…
```
# Haz la prediccion utilizando el conjunto de prueba
```

```
# hace la predicción utilizando X_test
# y_pred es el resultado, son las nuevas predicciones del modelo
y_pred = reg.predict(X_test)

#calculo el error cuadratico medio y R2

# se calcula el error entre los valores reales y los que fueron predecidos
mse = (mean_squared_error(Y_test, y_pred))

# muestra el error cuadrático medio con solo 2 decimales (.2f)
print(f"Error cuadrático medio (MSE): {mse:.2f}")

# se calcula el coeficiente de determinación, que tan bien el modelo predice los datos
r2_test = r2_score(Y_test, y_pred)

# muestra el r2 con solo 2 decimales (.2f)
print(f"Coeficiente de determinación (R² Score): {r2_test:.2f}")
```

```
Error cuadrático medio (MSE): 5.71
Coeficiente de determinación (R² Score): 0.81
```

Contesta en esta misma celda: ¿Que deduces de las métricas anteriores, es confiable tu modelo?

Creo que mi modelo es confiable y funcional, pero aún así tal vez se le pueden hacer mejoras como por ejemplo usar alguna de las otras variables dependientes para tener mejores resultados