```
# Import necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

```
# Importing the dataset
df = pd.read_csv('/content/advertising.csv')
```

```
df.head(10)
```

	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
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6

```
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
```

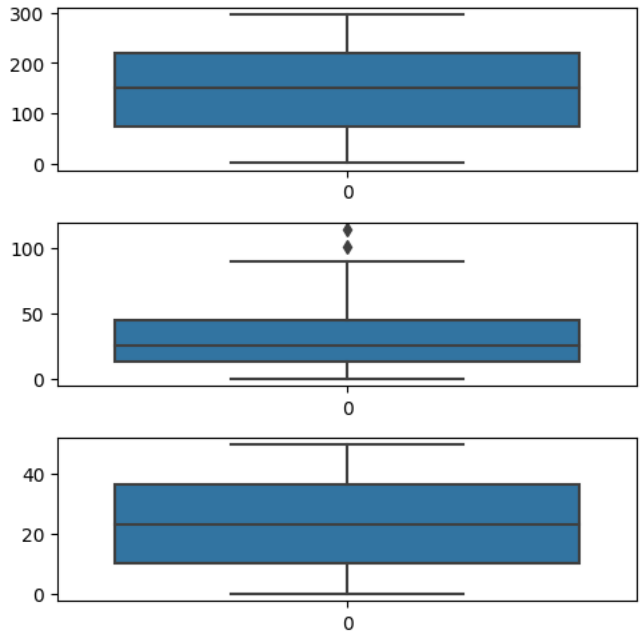
```
print(df.shape)
```

```
(200, 4)
```

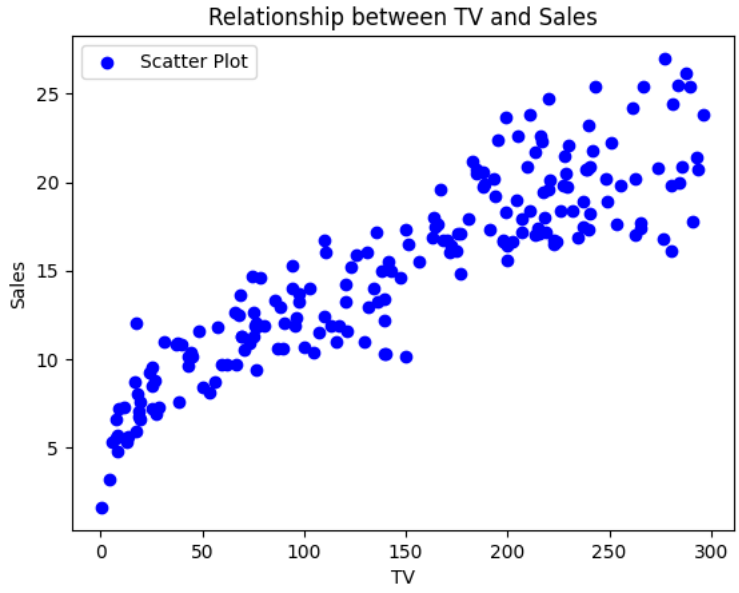
```
# View descriptive statistics

print(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
```

```
# Outlier Analysis
#tight_layout help to fit everything in figure
fig, axs = plt.subplots(3, figsize = (5,5))
plt1 = sns.boxplot(df['TV'], ax = axs[0])
plt2 = sns.boxplot(df['Newspaper'], ax = axs[1])
plt3 = sns.boxplot(df['Radio'], ax = axs[2])
plt.tight_layout()
```




```
# Declare feature variable and target variable

X = df['TV']
y = df['Sales']


# Plot scatter plot between X and y

plt.scatter(X, y, color = 'blue', label='Scatter Plot')
plt.title('Relationship between TV and Sales')
plt.xlabel('TV')
plt.ylabel('Sales')
plt.legend()
plt.show()
```




```
# Print the dimensions of X and y
print(X.shape)
print(y.shape)
```

```
(200,)
(200,)
```

```
X=np.array(X)
y=np.array(y)


# Reshape X and y
```

```
X = X.reshape(-1,1)
y = y.reshape(-1,1)



# Print the dimensions of X and y after reshaping

print(X.shape)
print(y.shape)
```

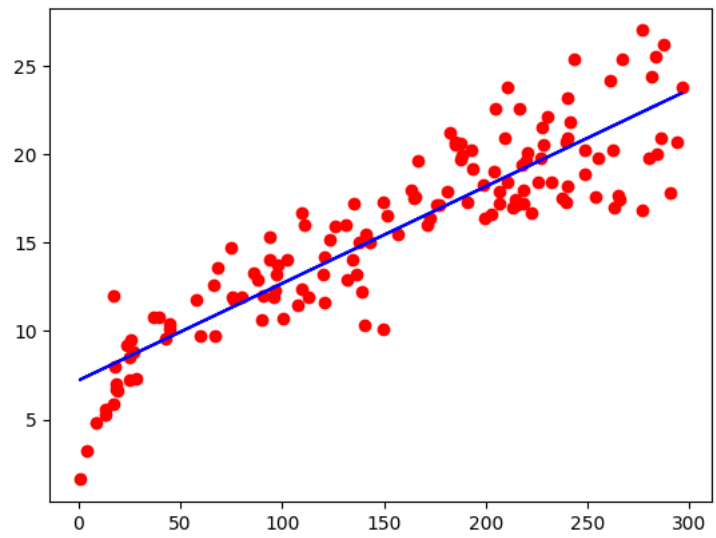
```
(200, 1)
(200, 1)
```

```
# Split X and y into training and test data sets
#random_state--the set of data does not change

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.30, random_state=42)


# Print the dimensions of X_train,X_test,y_train,y_test
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(140, 1)
(140, 1)
(60, 1)
(60, 1)
```

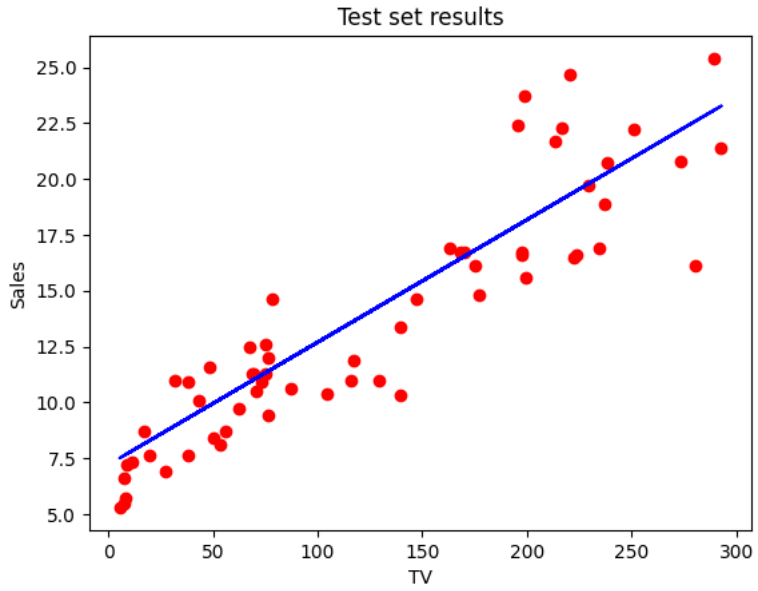
```
# Fit the linear model

# Instantiate the linear regression object lm
from sklearn.linear_model import LinearRegression
lm = LinearRegression()


# Train the model using training data sets
lm.fit(X_train,y_train)


# Predict on the test data
y_pred=lm.predict(X_test)


# Visualising the Training set results
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, lm.predict(X_train), color = 'blue')
```

```
[<matplotlib.lines.Line2D at 0x7a6fe1c08d00>]
```




```
# Visualising the Test set results
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_test, lm.predict(X_test), color = 'blue')
plt.title('Test set results')
plt.xlabel('TV')
```

```
plt.ylabel('Sales')
plt.show()
```




```
# Compute model slope and intercept

slope = lm.coef_
intercept = lm.intercept_,
print("Estimated model slope:" , slope)
print("Estimated model intercept:" , intercept)
```

```
Estimated model slope: [[0.05483488]]
Estimated model intercept: (array([7.20655455]),)
```

```
X_new = [[200]]
```

```
lm.predict(X_new)
```

```
array([[18.17353131]])
```