Assignment 1 Sales data Simple Linear Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv(r"D:\Supervised Machine Learning lab (SMLL)\1\
Assignment 1 SALES.csv",header=None,sep=r'\s+')
print(df.head())
```

```
      0     1
0  12.0  15.0
1  20.5  16.0
2  21.0  18.0
3  15.5  27.0
4  15.3  21.0
```

```
num_rows=df.shape[0]
print(num_rows)
```

```
36
```

```
df.columns=['Advertising','Sales']
print(df.head())
```

```
   Advertising  Sales
0         12.0   15.0
1         20.5   16.0
2         21.0   18.0
3         15.5   27.0
4         15.3   21.0
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Advertising  36 non-null     float64
 1   Sales        36 non-null     float64
dtypes: float64(2)
memory usage: 708.0 bytes
None
```

```
print(df.describe())
```

```
       Advertising      Sales
count    36.000000  36.000000
mean     24.255556  28.527778
std       6.185118  18.777625
min      12.000000   1.000000
25%      20.300000  15.750000
50%      24.250000  23.000000
75%      28.600000  41.000000
max      36.500000  65.000000
```

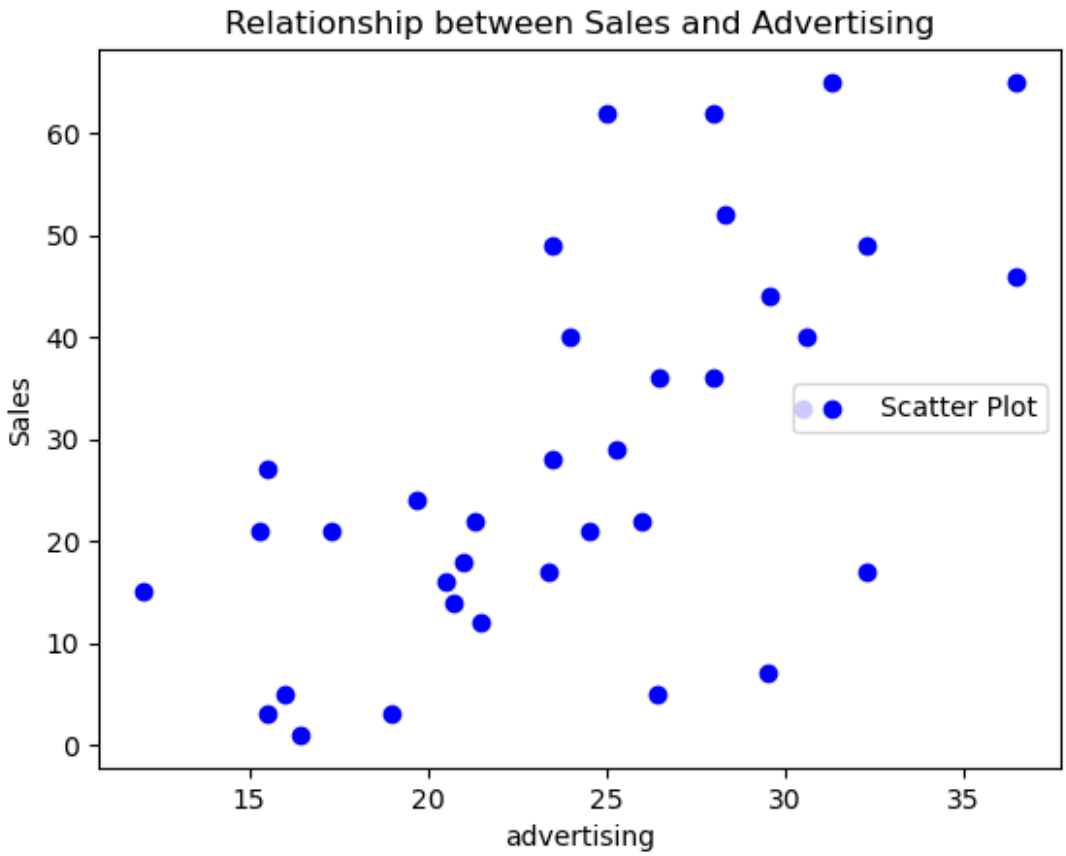
```
print(df.describe(include='all'))
```

```
       Advertising      Sales
count    36.000000  36.000000
mean     24.255556  28.527778
std       6.185118  18.777625
min      12.000000   1.000000
25%      20.300000  15.750000
50%      24.250000  23.000000
75%      28.600000  41.000000
max      36.500000  65.000000
```

```
#independent and dependent variables
X=df['Advertising'].values
y=df['Sales'].values
X
```

```
array([12. , 20.5, 21. , 15.5, 15.3, 23.5, 24.5, 21.3, 23.5, 28. , 24.
,
       15.5, 17.3, 25.3, 25. , 36.5, 36.5, 29.6, 30.5, 28. , 26. ,
21.5,
       19.7, 19. , 16. , 20.7, 26.5, 30.6, 32.3, 29.5, 28.3, 31.3,
32.3,
       26.4, 23.4, 16.4])
```

```
#plot scatter plot for X and y
plt.scatter(X,y,color='blue',label='Scatter Plot')
plt.title('Relationship between Sales and Advertising')
plt.xlabel('advertising')
plt.ylabel('Sales')
plt.legend(loc=5)
plt.show()
```




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

```
(36,)
(36,)
```

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

```
X
```

```
array([[12. ],
       [20.5],
       [21. ],
       [15.5],
       [15.3],
       [23.5],
       [24.5],
       [21.3],
       [23.5],
       [28. ],
       [24. ],
       [15.5],
       [17.3],
```

```
       [25.3],
       [25. ],
       [36.5],
       [36.5],
       [29.6],
       [30.5],
       [28. ],
       [26. ],
       [21.5],
       [19.7],
       [19. ],
       [16. ],
       [20.7],
       [26.5],
       [30.6],
       [32.3],
       [29.5],
       [28.3],
       [31.3],
       [32.3],
       [26.4],
       [23.4],
       [16.4]])
```

y

```
array([[15.],
       [16.],
       [18.],
       [27.],
       [21.],
       [49.],
       [21.],
       [22.],
       [28.],
       [36.],
       [40.],
       [ 3.],
       [21.],
       [29.],
       [62.],
       [65.],
       [46.],
       [44.],
       [33.],
       [62.],
       [22.],
       [12.],
       [24.],
       [ 3.],
```

```
       [ 5.],
       [14.],
       [36.],
       [40.],
       [49.],
       [ 7.],
       [52.],
       [65.],
       [17.],
       [ 5.],
       [17.],
       [ 1.]])
```

```
from sklearn.model_selection import train_test_split

# Assuming X and y are already defined (e.g., as NumPy arrays or
pandas DataFrames/Series)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(28, 1)
(28, 1)
(8, 1)
(8, 1)
```

```
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)

# X_intercept
a = lm.intercept_
a
```

```
array([-12.45519409])
```

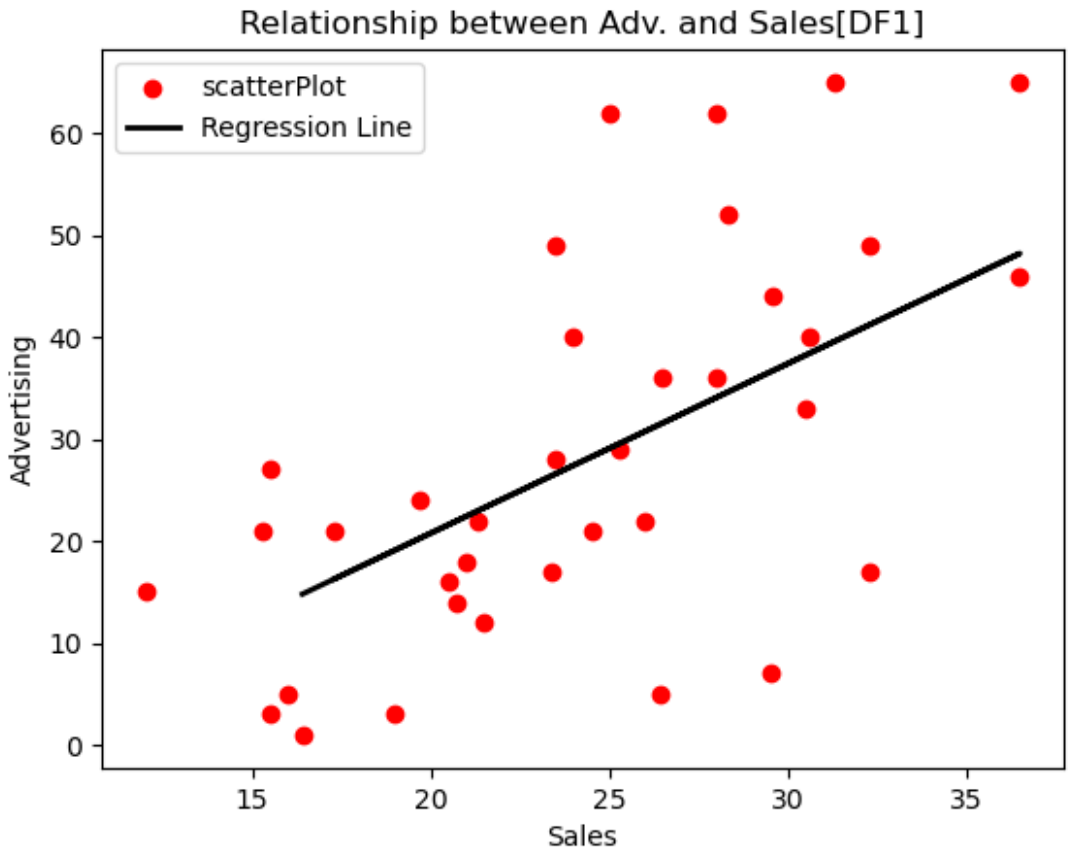
```
# Y_intercept
b = lm.coef_
print((b))
```

```
[[1.66205855]]
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)

# Root Mean Squared Error

rmse = np.sqrt(mse)
print("RMSE value {:.4}".format(rmse))
# Calculate Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)

# Calculate R-squared
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-squared: {r2}")
```

```
RMSE value 12.82
Mean Squared Error: 164.3461539052813
Mean Absolute Error: 9.981297920558294
R-squared: 0.5875233784338038
```

```
#plotting the regression line
plt.scatter(X,y,color='red', label='scatterPlot')
plt.plot(X_test, y_pred, color = 'black', linewidth = 2, label
='Regression Line')
plt.title('Relationship between Adv. and Sales[DF1]')
plt.xlabel('Sales')
plt.ylabel('Advertising')
plt.legend() #ignored loc{loc specifies where the label='scatterPlot'
is going to be located}
plt.show()
```



MyCar Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df_mycar=pd.read_csv(r"D:\Supervised Machine Learning lab (SMLL)\1\
mycar.csv")
print(df_mycar.head())
```

```
   Unnamed: 0  speed  dist
0           1      4     2
1           2      4    10
2           3      7     4
3           4      7    22
4           5      8    16
```

```
df_mycar.shape
```

```
(50, 3)
```

```
df_mycar.head()
```

```
   Unnamed: 0  speed  dist
0           1      4     2
1           2      4    10
2           3      7     4
3           4      7    22
4           5      8    16
```

```
num_cols=df_mycar.shape[1]
print(f"The no of columns are : {num_cols}")
```

```
The no of columns are : 3
```

```
print(df_mycar.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  50 non-null     int64
 1   speed       50 non-null     int64
 2   dist        50 non-null     int64
dtypes: int64(3)
memory usage: 1.3 KB
None
```

```
print(df_mycar.describe())
```

```
       Unnamed: 0      speed        dist
count    50.00000  50.000000   50.000000
mean     25.50000  15.400000   42.980000
std      14.57738   5.287644   25.769377
min       1.00000   4.000000    2.000000
25%      13.25000  12.000000   26.000000
50%      25.50000  15.000000   36.000000
75%      37.75000  19.000000   56.000000
max      50.00000  25.000000  120.000000
```

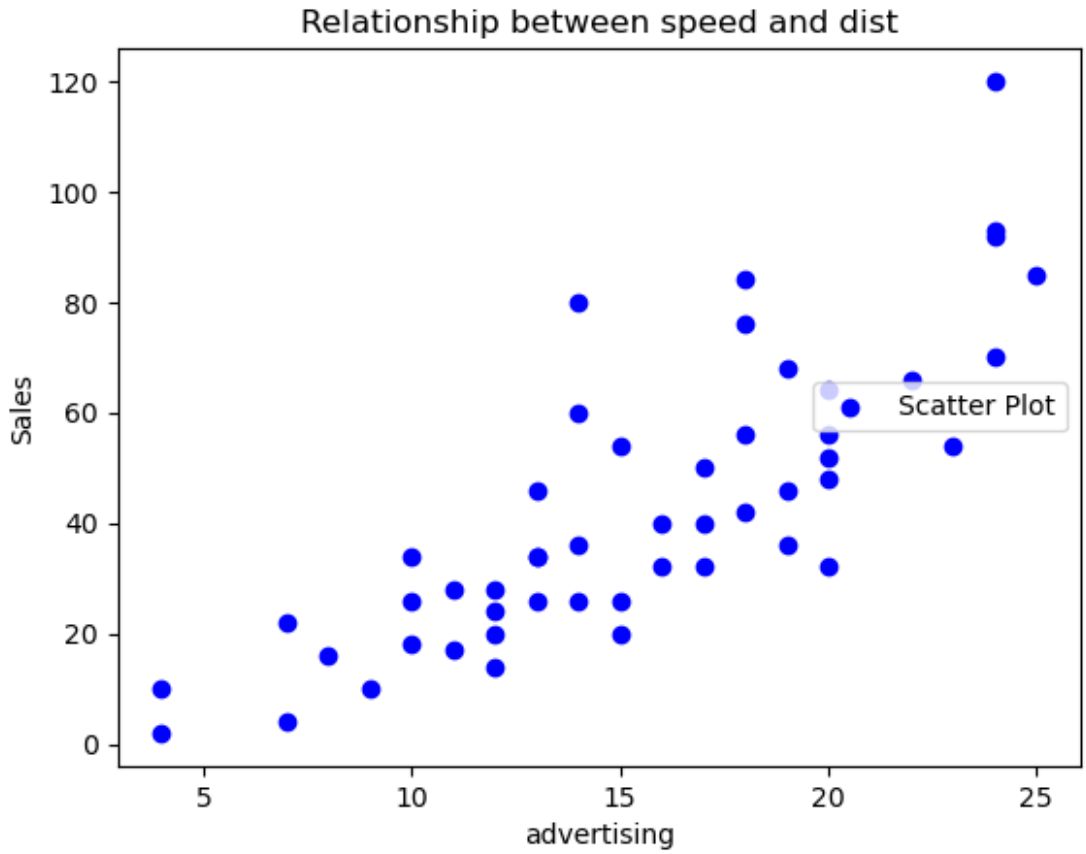
```
print(df_mycar.describe(include='all'))
```

```
       Unnamed: 0      speed        dist
count    50.00000  50.000000   50.000000
mean     25.50000  15.400000   42.980000
std      14.57738   5.287644   25.769377
min       1.00000   4.000000    2.000000
25%      13.25000  12.000000   26.000000
50%      25.50000  15.000000   36.000000
75%      37.75000  19.000000   56.000000
max      50.00000  25.000000  120.000000
```

```
#independent and dependent variables
X=df_mycar['speed'].values
y=df_mycar['dist'].values
X
```

```
array([ 4,  4,  7,  7,  8,  9, 10, 10, 10, 11, 11, 12, 12, 12, 12, 13,
13,
       13, 13, 14, 14, 14, 14, 15, 15, 15, 16, 16, 17, 17, 17, 18, 18,
18,
       18, 19, 19, 19, 20, 20, 20, 20, 20, 22, 23, 24, 24, 24, 24,
25],
      dtype=int64)
```

```
#plot scatter plot for X and y
plt.scatter(X,y,color='blue',label='Scatter Plot')
plt.title('Relationship between speed and dist')
plt.xlabel('advertising')
plt.ylabel('Sales')
plt.legend(loc=5)
plt.show()
plt.savefig('1.png')
```




```
<Figure size 640x480 with 0 Axes>
```

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

```
(50,)
(50,)
```

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

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

```
(50, 1)
(50, 1)
```

```
from sklearn.model_selection import train_test_split

# Assuming X and y are already defined (e.g., as NumPy arrays or
pandas DataFrames/Series)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(35, 1)
(35, 1)
(15, 1)
(15, 1)
```

```
from sklearn.linear_model import LinearRegression

lm_car = LinearRegression()

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

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

X_test
```

```
array([[12],
       [20],
       [17],
       [24],
       [13],
       [24],
       [16],
       [15],
```

```
       [18],
       [14],
       [12],
       [ 8],
       [19],
       [10],
       [ 7]], dtype=int64)
```

```
a = lm_car.intercept_
print(a)
```

```
[-16.01050541]
```

```
PRED = lm_car.predict([[50]])
PRED
```

```
array([[172.10111098]])
```

```
b = lm_car.coef_
print((b))
```

```
[[3.76223233]]
```

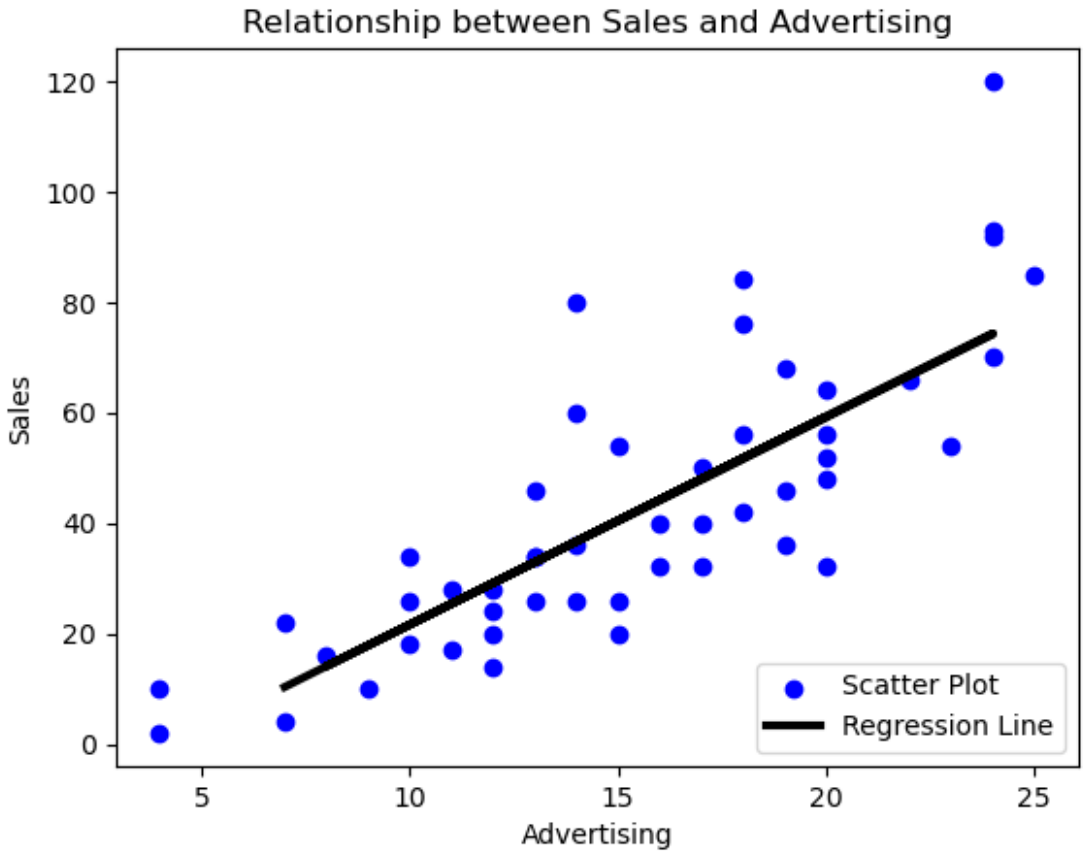
```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)

# Root Mean Squared Error

rmse = np.sqrt(mse)

# Calculate Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)

# Calculate R-squared
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print("RMSE value {:.4}".format(rmse))
print(f"Mean Absolute Error: {mae}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 217.67480016957953
RMSE value 14.75
Mean Absolute Error: 10.525189960856551
R-squared: 0.6806081096217954
```

```
X_test.shape
```

```
(15, 1)
```

```
import matplotlib.pyplot as plt
import numpy as np


# Plot the data
plt.scatter(X, y, color='blue', label='Scatter Plot')
plt.plot(X_test, y_pred, color='black', linewidth=3, label='Regression
Line')
plt.title('Relationship between Sales and Advertising')
plt.xlabel('Advertising')
plt.ylabel('Sales')
plt.legend(loc=4)  # Bottom-right corner
plt.show()
plt.savefig('2.png')
```




```
<Figure size 640x480 with 0 Axes>
```

Salary Dataset

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

```
import matplotlib.pyplot as plt
df_sal=pd.read_csv(r"D:\Supervised Machine Learning lab (SMLL)\1\
Salary_dataset.csv")
print(df_sal.head())
```

```
   Unnamed: 0  YearsExperience   Salary
0           0              1.2  39344.0
1           1              1.4  46206.0
2           2              1.6  37732.0
3           3              2.1  43526.0
4           4              2.3  39892.0
```

```
df_sal.head()
```

```
   Unnamed: 0  YearsExperience   Salary
0           0              1.2  39344.0
1           1              1.4  46206.0
2           2              1.6  37732.0
3           3              2.1  43526.0
4           4              2.3  39892.0
```

```
sal = df_sal.iloc[: , 1: ]
sal
```

```
    YearsExperience    Salary
0               1.2   39344.0
1               1.4   46206.0
2               1.6   37732.0
3               2.1   43526.0
4               2.3   39892.0
5               3.0   56643.0
6               3.1   60151.0
7               3.3   54446.0
8               3.3   64446.0
9               3.8   57190.0
10              4.0   63219.0
11              4.1   55795.0
12              4.1   56958.0
13              4.2   57082.0
14              4.6   61112.0
15              5.0   67939.0
16              5.2   66030.0
17              5.4   83089.0
18              6.0   81364.0
19              6.1   93941.0
20              6.9   91739.0
21              7.2   98274.0
22              8.0  101303.0
23              8.3  113813.0
24              8.8  109432.0
```

```
25              9.1  105583.0
26              9.6  116970.0
27              9.7  112636.0
28             10.4  122392.0
29             10.6  121873.0
```

```
X=sal['YearsExperience'].values
y=sal['Salary'].values
X
```

```
array([ 1.2,  1.4,  1.6,  2.1,  2.3,  3. ,  3.1,  3.3,  3.3,  3.8,  4.
,
        4.1,  4.1,  4.2,  4.6,  5. ,  5.2,  5.4,  6. ,  6.1,  6.9,
7.2,
        8. ,  8.3,  8.8,  9.1,  9.6,  9.7, 10.4, 10.6])
```

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

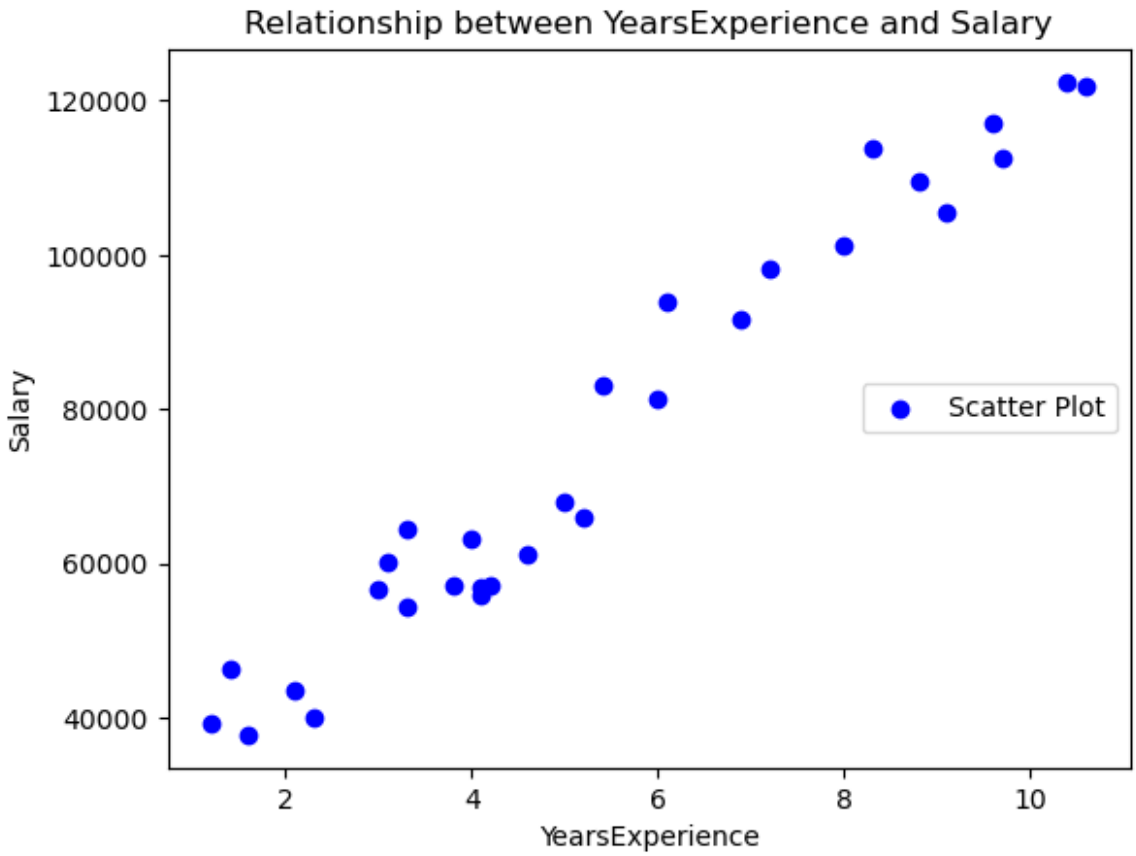
```
(30,)
(30,)
```

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

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

```
(30, 1)
(30, 1)
```

```
#plot scatter plot for X and y
plt.scatter(X,y,color='blue',label='Scatter Plot')
plt.title('Relationship between YearsExperience and Salary')
plt.xlabel('YearsExperience')
plt.ylabel('Salary')
plt.legend(loc=5)
plt.show()
plt.savefig('3.png')
```




```
<Figure size 640x480 with 0 Axes>
```

```
from sklearn.model_selection import train_test_split

# Assuming X and y are already defined (e.g., as NumPy arrays or
pandas DataFrames/Series)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

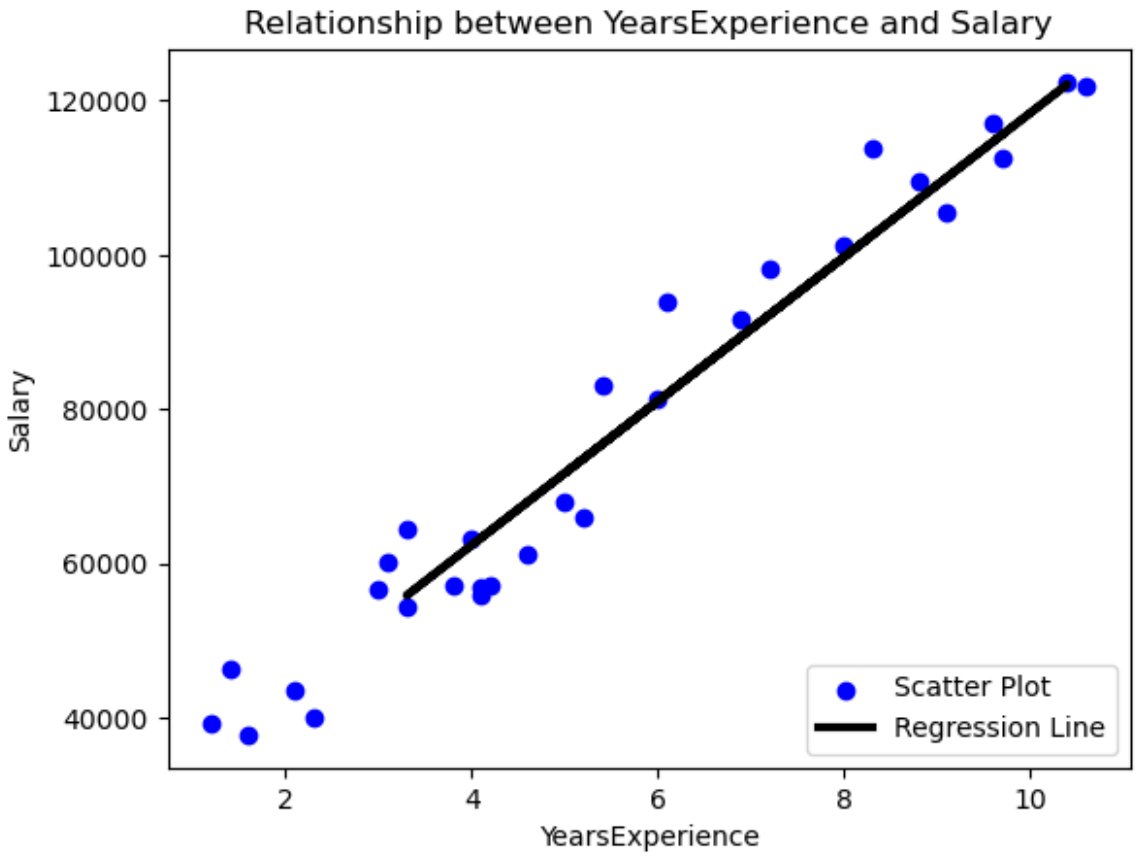
```
(21, 1)
(21, 1)
(9, 1)
(9, 1)
```

```
from sklearn.linear_model import LinearRegression

lm_sal = LinearRegression()

# Train the model using training data sets
```

```
lm_sal.fit(X_train,y_train)

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

# Errors
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
# Mean Square Error
mean_square_error = mean_squared_error(y_test,y_pred)
print(f"Mean Square Error is : {mean_square_error}")
# Calculate Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error is : {mae}")
# Root Mean Square Error
root_mean_square_error = np.sqrt(mean_square_error)
print(f"Root Mean Square Error is : {root_mean_square_error}")
# Calculate R-squared
r2 = r2_score(y_test, y_pred)
print(f"R2 score is : {r2*100}")
#intecept
X_intercept = lm_sal.intercept_
print(f"X intercept is : {X_intercept}")
Y_intercept = lm_sal.coef_
print(f"Y intercept is : {Y_intercept}")
```

```
Mean Square Error is : 37784662.46621308
Mean Absolute Error is : 5161.328710400178
Root Mean Square Error is : 6146.9230079945755
R2 score is : 94.14466227178215
X intercept is : [24985.53016251]
Y intercept is : [[9339.08172382]]
```

```
import matplotlib.pyplot as plt
import numpy as np

# Plot the data
plt.scatter(X, y, color='blue', label='Scatter Plot')
plt.plot(X_test, y_pred, color='black', linewidth=3, label='Regression
Line')
plt.title('Relationship between YearsExperience and Salary')
plt.xlabel('YearsExperience')
plt.ylabel('Salary')
plt.legend(loc=4)  # Bottom-right corner
plt.show()
plt.savefig('4.png')
```




```
<Figure size 640x480 with 0 Axes>
```