GRIP: The Sparks Foundation

Data Science and Business Analytics Internship Task 1: Prediction Using Supervised Machine Learning

Predict the percentage of an student based on the number of study hours Python 3 Jupyter Notebook Linear Regression Chetana Thorat

Step 1: Import the Dataset Step 2: Visualize and Analyze the Dataset Step 3: Preparation of Data Step 4: Design and Train the Machine Learning Model Step 5: Visualize the Model Step 6: Make Predictions Step 7: Evaluate the model

In [45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [46]:
```
# Reading data from remote link using the url
url = "http://bit.ly/w-data"
student_data = pd.read_csv(url)

print("Data imported successfully")
student_data
```

```
Data imported successfully
```

Out[46]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [47]:
```
student_data.shape

#here we can see that there are 25 rows and 2 colums in the dataset
```

Out[47]:
```
(25, 2)
```

In [48]:
```
student_data.describe()
```

Out[48]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [53]:
```
student_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null    float64
 1   Scores  25 non-null    int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

In [54]:
```
student_data.isnull().sum()

#Here,we can see that there are no NULL VALUES in the dataset that can affect the training of our algorithm
```

Out[54]:
```
Hours     0
Scores    0
dtype: int64
```

In [79]:
```
student_data.corr(method = 'pearson')
```

Out[79]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [80]:
```
student_data.corr(method='spearman')
```

Out[80]:

	Hours	Scores
Hours	1.000000	0.971891
Scores	0.971891	1.000000

In [81]:
```
# Plotting the distribution of scores and number of hours studied on a 2D graph

student_data.plot(x='Hours', y='Scores', style='ro')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```

In [56]:
```
#we are extracting values of Hours Data into variable X and the values of Score Data into variable Y
x = student_data.iloc[:, :-1].values
y = student_data.iloc[:, 1].values
```

In [57]:
```
#number of hours studied
x
```

Out[57]:
```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [58]:
```
#Score obtained
y
```

Out[58]:
```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95, 30,
       24, 67, 69, 30, 54, 35, 76, 86], dtype=int64)
```

In [59]:
```
#We now split the data into train and test dataset using Scikit-Learn's --built-in train_test_split()

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=0)
```

In [60]:
```
#no of rows in training dataset (x-colums)
x_train.shape
```

Out[60]:
```
(20, 1)
```

In [61]:
```
#no of rows in training dataset (y-colums)
y_train.shape
```

Out[61]:
```
(20,)
```

In [62]:
```
#no of rows in testing dataset (x-colums)
x_test.shape
```

Out[62]:
```
(5, 1)
```

In [63]:
```
#no of rows in testing dataset (y-colums)
y_test.shape
```

Out[63]:
```
(5,)
```

In [64]:
```
x_train
```

Out[64]:
```
array([[3.8],
       [1.9],
       [7.8],
       [6.9],
       [1.1],
       [5.1],
       [7.7],
       [3.3],
       [8.3],
       [9.2],
       [6.1],
       [3.5],
       [2.7],
       [5.5],
       [2.7],
       [8.5],
       [2.5],
       [4.8],
       [8.9],
       [4.5]])
```

In [65]:
```
y_train
```

Out[65]:
```
array([35, 24, 86, 76, 17, 47, 85, 42, 81, 88, 67, 30, 25, 60, 30, 75, 21,
       54, 95, 41], dtype=int64)
```

In [66]:
```
x_test
```

Out[66]:
```
array([[1.5],
       [3.2],
       [7.4],
       [2.5],
       [5.9]])
```

In [67]:
```
y_test
```

Out[67]:
```
array([20, 27, 69, 30, 62], dtype=int64)
```

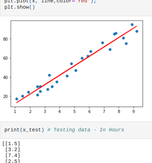
In [68]:
```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(x_train, y_train)
print("Training complete")
```

```
Training complete
```

In [69]:
```
# Plotting the regression line (y=mx+c)
line = regressor.coef_*x+regressor.intercept_

# Plotting for the test data
plt.scatter(x, y)
plt.plot(x, line,color='red');
plt.show()
```



In [70]:
```
print(x_test) # Testing data - In Hours
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [71]:
```
print(y_test) # Testing data - In percentage
```

```
[20 27 69 30 62]
```

In [72]:
```
# Predicting the scores
y_pred = regressor.predict(x_test)
print(y_pred)
```

```
[16.88414476 33.73226078 75.357018   26.79480124 60.49103328]
```

In [73]:
```
# Comparing Actual vs Predicted
compdata = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
compdata
```

Out[73]:

	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [74]:
```
#Testing with the custom of 9 hrs/day

hours = 9.25
own_pred = regressor.predict([[hours]])
print("No of Hours = {hours}")
print("Predicted Score = {own_pred[0]}")
```

```
No of Hours = 9.25
Predicted Score = 93.69173248737538
```

In [75]:
```
from sklearn import metrics

print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```

In [76]:
```
print('Max Error:',metrics.max_error(y_test, y_pred))
```

```
Max Error: 6.732260779489842
```

In [77]:
```
print('Mean Squared Error:',metrics.mean_squared_error(y_test, y_pred))
```

```
Mean Squared Error: 21.5987693072174
```

In [ ]:
```

```