THE SPARKS FOUNDATION-GRIP

Data Science and Business Analytics Internship

Name- Shrutika Pramanik

Task 1 - Prediction Using ML

Objective - In this task, we need to predict the percentage of a student based on the number of study hours. We also need to find the predicted score if a student studies for 9.25 hours/day.

Simple Linear Regression

Steps

Step 1) Importing Libraries

Step 2) Visualizing the dataset

Step 3) Data Preparation

Step 4) Training the Algorithm

Step 5) Making Predictions

Step 6) Evaluation of the Model

[Dataset URL]- (http://bit.ly/w-data)

Step 1 - Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
# Reading data from remote link

url = "http://bit.ly/w-data"
data = pd.read_csv(url)
print("Data imported successfully")
print(data)
```

```
Data imported successfully
    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
```

The first 5 rows, last 5 rows, shape, description of the data are displayed

In [3]:
```
data.head(#gives the first 5 rows
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [4]:
```
data.tail(#gives the last 5 rows
```

Out[4]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [5]:
```
data.shape #gives the shape of the data
```

Out[5]:
```
(25, 2)
```

In [6]:
```
data.info() #gives information of the data
```

```
<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 [7]:
```
data.describe() #describes the data
```

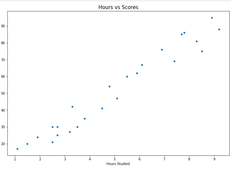
Out[7]:

	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

Let's plot our data points on 2-D graph to eyeball our dataset and see if we can manually find any relationship between the data.

Step 2 - Data Visualization

In [8]:
```
#scatter plot
plt.figure(figsize=(12,8))
sns.scatterplot(x=data.Hours,y=data.Scores)
plt.title('Hours vs Scores',fontdict={'fontsize':15})
plt.xlabel('Hours Studied')
plt.ylabel('Marks Scored')
plt.show()
```



From the above graph, we can clearly see that there is a **positive linear relation** between the number of hours studied and the marks scored.

CORRELATION OF THE DATA

In [10]:
```
data.corr() #gives the correlation of the data
```

Out[10]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

There is a **positive(strong)** correlation between hours studied and marks scored.

Step 3 - Data Preparation

SPLITTING THE DATA

In [11]:
```
#Dividing the DF to independent and dependent variable
x = data['Hours'].values.reshape(-1,1)
y = data['Scores']
```

The x and y values are shown below.

In [12]:
```
print('The values of x are')
x
```

```
The values of x are
```

Out[12]:
```
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 [13]:
```
print('The values of y are')
y
```

```
The values of y are
```

Out[13]:
```
0     21
1     47
2     27
3     75
4     30
5     20
6     88
7     60
8     81
9     25
10    85
11    62
12    41
13    42
14    17
15    95
16    30
17    24
18    67
19    69
20    30
21    54
22    35
23    76
24    86
Name: Scores, dtype: int64
```

The next step is to split the data into training and test sets.

We'll do this by using Scikit-Learn's built-in train_test_split() method:

In [29]:
```
# Splitting the x,y into train and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.20, random_state = 0)
```

Step 4 - Training The Algorithm

We have split our data into training and testing sets and now is the finally time to train the algorithm.

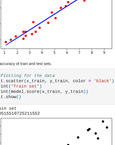
In [15]:
```
# Importing LinearRegression from sklearn
from sklearn.linear_model import LinearRegression

# Creating object and fitting the model
lin_reg = LinearRegression()
model = lin_reg.fit(x_train,y_train)
```

PLOTTING THE REGRESSION LINE

In [16]:
```
# Plotting the regression line
line = model.coef_*x+model.intercept_
```

In [19]:
```
# Plotting for the data
plt.scatter(x, y, color = "red")
plt.plot(x, line, color = "blue")
plt.show()
```

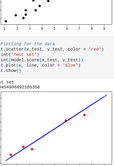
Out[19]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



The accuracy of train and test sets.

In [20]:
```
# Plotting for the data
plt.scatter(x_train, y_train, color = 'black')
print('Train set')
print(model.score(x_train, y_train))
plt.show()
```

```
Train set
0.9515510725211552
```



In [21]:
```
# Plotting for the data
plt.scatter(x_test, y_test, color = 'red')
print('Test set')
print(model.score(x_test, y_test))
plt.plot(x, line, color = "blue")
plt.show()
```

```
Test set
0.9454906892105356
```

Step 5 - Making Predictions

Now that we have trained our algorithm, it's time to make some predictions.

In [22]:
```
# Predicting for test dataset
y_pred = model.predict(x_test)
```

In [23]:
```
# Creating Actual and Predicted dataset
df1 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df1
```

Out[23]:

	Actual	Predicted
5	20	16.884145
2	27	33.732261
19	69	75.357018
16	30	26.794801
11	62	60.491033

SCORE PREDICTION FOR 9.25 HOURS

In [24]:
```
# Testing with your own data
hours = np.array([9.25]) # No. of hours should be mentioned inside array
hours = hours.reshape(-1,1)
own_pred = model.predict(hours)
print("No. of Hours studied by the student = {}".format(float(hours)))
print("Predicted Score = {}".format(round(own_pred[0],2)))
```

```
No. of Hours studied by the student = 9.25
Predicted Score = 92.69
```

So the predicted score of a student studies for 9.25 hours/day is 93.69.

Step 6 - Evaluation Of Model

In [31]:
```
# Model Evaluation

# Importing metrics from sklearn
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# To find Mean Absolute Error(MAE)
mae = (mean_absolute_error(y_test, y_pred))
print('MAE:', mae)

# To find Root Mean Squared Error(RMSE)
rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))
print('RMSE:', rmse)

# To find Coefficient of Determination
r2 = r2_score(y_test, y_pred)
print('R-Square:', r2)
```

```
MAE: 4.183859899002975
RMSE: 4.647447612100365
R-Square: 0.9454906892105356
```

THANK YOU.