THE SPARKS FOUNDATION-GRIP

Data Science and Business Analytics Internship

Name - Ruchira Sengupta

Task 1 - Prediction Using Supervised 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

In this regression task we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied. This is a simple linear regression task as it involves just two variables.

IMPORT THE REQUIRED LIBRARIES

In [26]:

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

READ THE DATASET FROM THE URL MENTIONED IN THE GRIP TASK

In [27]:

```
url = "https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_scores.csv"
data = pd.read_csv(url)
print('Data imported successfully')
print(data)#printing the data imported from the url
```

```
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 [28]:

```
data.head()#gives the first 5 rows
```

Out[28]:

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

In [29]:

```
data.tail()#gives the last 5 rows
```

Out[29]:

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

In [30]:

```
data.shape #gives the shape of the data
```

Out[30]:

```
(25, 2)
```

In [31]:

```
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 [32]:

```
data.describe() #describes the data
```

Out[32]:

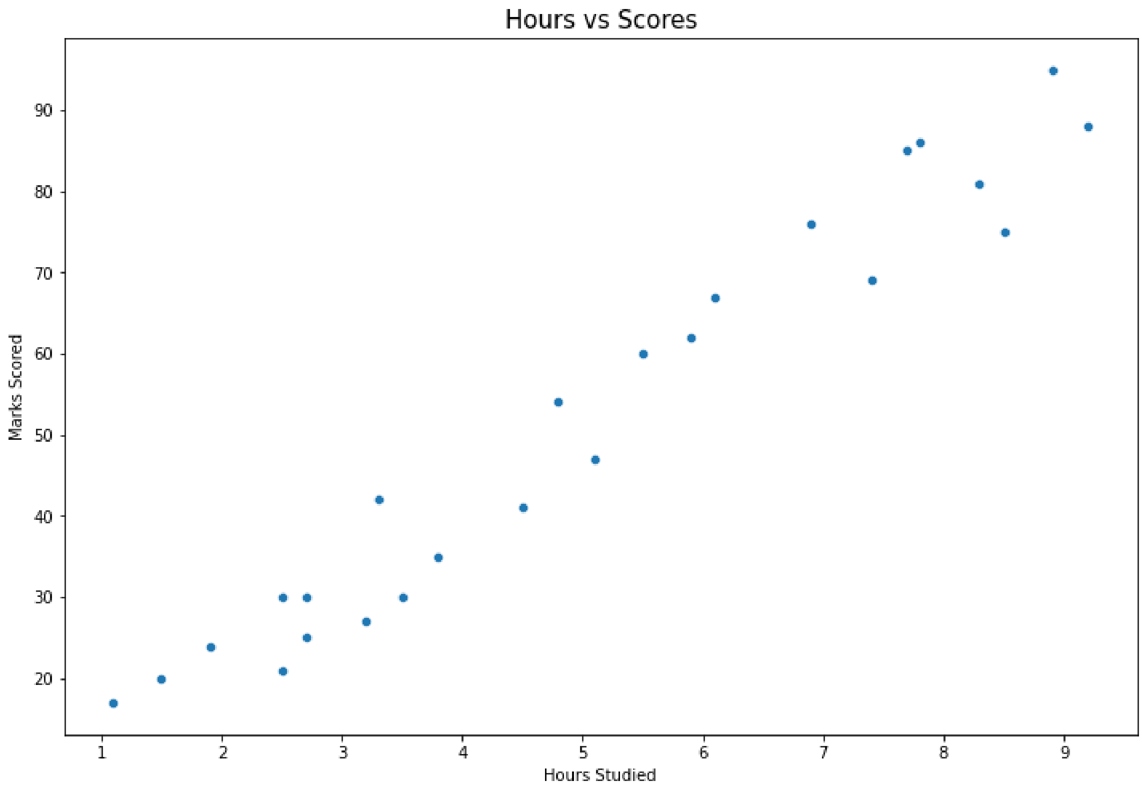
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

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.

DATA VISUALIZATION

In [33]:

```
#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 graph above, 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 [34]:

```
data.corr() #gives the correlation of the data
```

Out[34]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

There is a positive(strong) correlation between and hours studied and scores.

PREPARING THE DATA

Splitting the data

In [35]:

```
# 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 [36]:

```
print('The values of X are')
X
```

```
The values of X are
```

Out[36]:

```
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 [37]:

```
print('The values of y are')
y
```

```
The values of y are
```

Out[37]:

```
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 this data into training and test sets.

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

In [38]:

```
# Spliting the X,y into train and test

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)
```

TRAINING THE ALGORITHM

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

In [39]:

```
# 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)
```

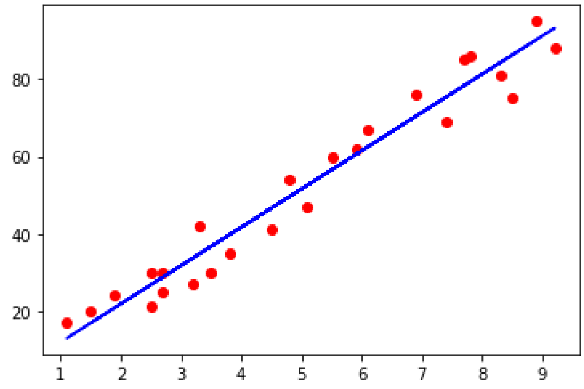
Plotting the regression line

In [40]:

```
# Plotting the regression line
line = model.coef_*X+model.intercept_
```

In [41]:

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

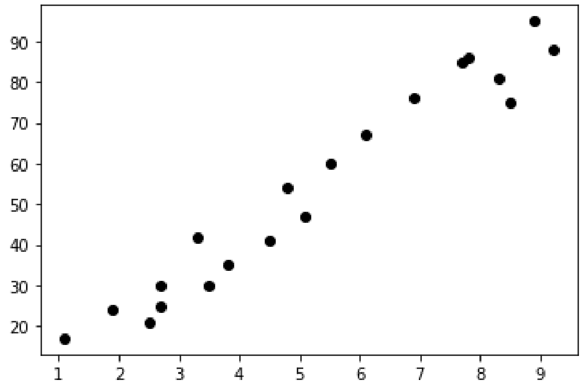


The accuracy of train and test sets.

In [42]:

```
# Plotting for the data
plt.scatter(X_train, y_train,color='black')
print('Train set')
print(model.score(X_train,y_train))
plt.show()
```

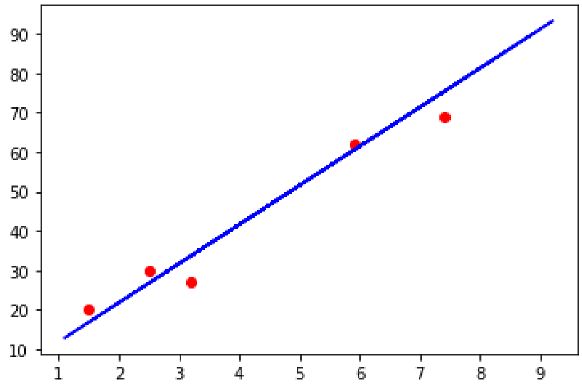
```
Train set
0.9515510725211552
```



In [43]:

```
# 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
```



MAKING PREDICTIONS

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

In [44]:

```
# Predicting for test dataset
y_pred = model.predict(X_test)
```

In [45]:

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

Out[45]:

	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 [46]:

```
# 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 = 93.69
```

So, the predicted score if a student studies for 9.25 hours/day is 93.69

MODEL EVALUATION

In [47]:

```
# Model Evaluation

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

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

# 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: 4.6474476121003665
R-Square: 0.9454906892105356
```

THANK YOU.