LINEAR REGRESSION IN PYTHON WITH SCIKIT LEARN

Predicting the percentage of students based on the number of study hours using supervised machine learning algorithm,linear regression.

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

In [6]:
```
#reading and storing the dataframe into the variable
df = pd.read_csv('http://bit.ly/w-data')
df.head()
```

Out[6]:

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

Checking if the dataset has any null values

In [36]:
```
df.isnull()
```

Out[36]:

	Hours	Scores
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False
19	False	False
20	False	False
21	False	False
22	False	False
23	False	False
24	False	False

Since our dataset doesnot have any null values we dont have to handle any null values

Understanding our dataset

In [38]:
```
df.describe()
```

Out[38]:

	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 [47]:
```
df.mean()
```

Out[47]:
```
Hours      5.012
Scores    51.480
dtype: float64
```

We can understand about the mean hours and scores in our dataset

In [104…
```
X = df.iloc[:, :-1].values    # Hours => Independent Variable,
y = df.iloc[:, 1].values     # Scores => Dependent Variable,
X
```

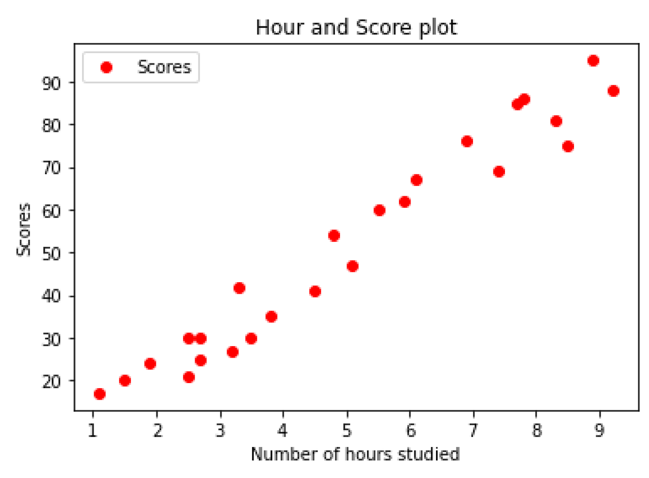
Out[104…
```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [105…
```
y
```

Out[105…
```
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)
```

Plotting the independent and dependent variable in the graph to check the relationship

In [14]:
```
df.plot(x='Hours',y='Scores',style='or')
plt.title('Hour and Score plot')
plt.xlabel('Number of hours studied')
plt.ylabel('Scores')
plt.show()
```



This shows that the number of hours of the student and the Scores achieved are directly proportional and thus give us a positve correlation between the variables.

Splitting the data into training and testing groups using train_test_split

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.3, random_state=0)
#we can choose the test size based on our choice ,here the test_size taken is 0.3
```

Since we have split our data into training and testing sets , we can try and implement linear regression to predict the score a student can acquire if he studies for 9.5 hours

Implementing Linear Regression on the training dataset

In [20]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
```

Out[20]:
```
LinearRegression()
```

Since the line is formed by the intercept and the slope , we need to find the values for the intercept and the slope in in order to plot the graph.A linear regression line has an equation of the form y = mX + c, where X is the explanatory variable and y is the dependent variable. The slope of the line is m, and c is the intercept (the value of y when x = 0).

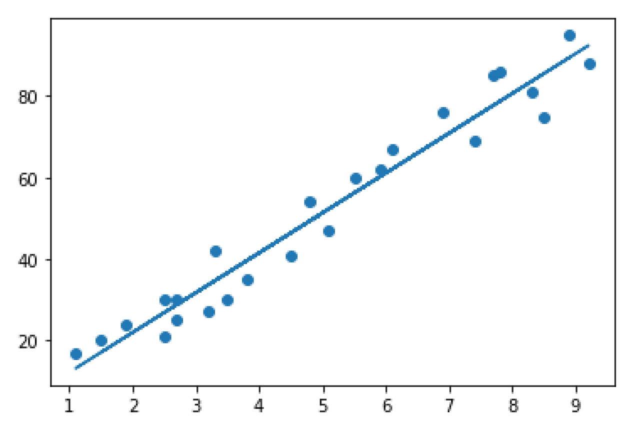
In [25]:
```
intercept = print(reg.intercept_)
slope = print(reg.coef_)
```

```
2.370815382341881
[9.78856669]
```

Since the equation of a line is given as y = mx + c ,the cofficient of our line here is the slope that is, our slope here is 9.78 and the intercept is found to be as 2.37 or in other words the line intersects the 'y-axis' at 2.37

In [115…
```
line_equation = reg.coef_*X + reg.intercept_

# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line_equation);
plt.show()
```



In [64]:
```
print(X_test) # Testing data - In Hours
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]
 [3.8]
 [1.9]
 [7.8]]
```

In [66]:
```
output_pred = reg.predict(X_test) # Predicting the scores obtained from the training data
```

Predictions with the help of the trained data.

In order to find out how well our model has predicted and better data comprehension ,let us compare the predicted scores, the actual scores, their difference and the percentage of difference.

In [108…
```
# Comparing Actual values and the  Predicted values
diff_df = pd.DataFrame({'Actual': y_test, 'Predicted': output_pred,})
diff_df
new_diff = diff_df
new_diff['Difference']=new_diff['Actual']-new_diff['Predicted']
new_diff['Difference_Percent']=(new_diff['Difference']/new_diff['Actual'])*100
new_diff
```

Out[108…

	Actual	Predicted	Difference	Difference_Percent
0	20	17.053665	2.946335	14.731673
1	27	33.694229	-6.694229	-24.793440
2	69	74.806209	-5.806209	-8.414795
3	30	26.842232	3.157768	10.525893
4	62	60.123359	1.876641	3.026841
5	35	39.567369	-4.567369	-13.049625
6	24	20.969092	3.030908	12.628783
7	86	78.721636	7.278364	8.463214

In [95]:
```
hours = [[9.25]]
own_pred = reg.predict(hours)
own_pred
```

Out[95]:
```
array([92.91505723])
```

So it is understood that if a student learns for 9.5 hours,the model predicts it to have score of 92.9

Measuring the performance of the model

The performance of the linear regression model is to be measured using Mean Absolute Error and Root Mean Square Error.This evaluation will determine how well our model has been performing and how close are our results to the actual results from the training data

In [99]:
```
#Evaluating the metrics using root mean squared error
from sklearn.metrics import mean_squared_error
expected = new_diff['Actual']
predicted = new_diff['Predicted']
# calculating residues
errors = mean_squared_error(expected, predicted, squared=False)
print(errors)
```

```
4.792191274636315
```

In [116…
```
#evaluating the metrics from mean absolute error
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.419727808027652
```

On comparision with both the evaluation metrics, it is understood that the linear regression model for the student percentage prediction performs better with MAE metric