Data Science and Business Analytics Intern

Name : Sarthak Gupta

Task 1 : Prediction Using Supervised ML

Task is to predict the percentage of student based on no. of study hours

GRIPJUN21

In [1]:
```
# Importing all the necesarry libraries needed in our entire analysis

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

In [2]:
```
# loading the dataset to perform further operations on our given data

data_url = "http://bit.ly/w-data"
df = pd.read_csv(data_url)
```

In [3]:
```
# Printing our dataset
df
```

Out[3]:

	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 [4]:
```
# checking the shape of data set

df.shape  # which gives us the result that our dataset contains 25 rows and 2 col
```

Out[4]:
```
(25, 2)
```

In [5]:
```
# Printing the first five observations using head funtion

df.head()
```

Out[5]:

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

In [6]:
```
# Printing the last five observations using tail funtion

df.tail()
```

Out[6]:

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

In [7]:
```
# Process to check if any missing value is present in our data set or not!
df.isna()
```

Out[7]:

	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

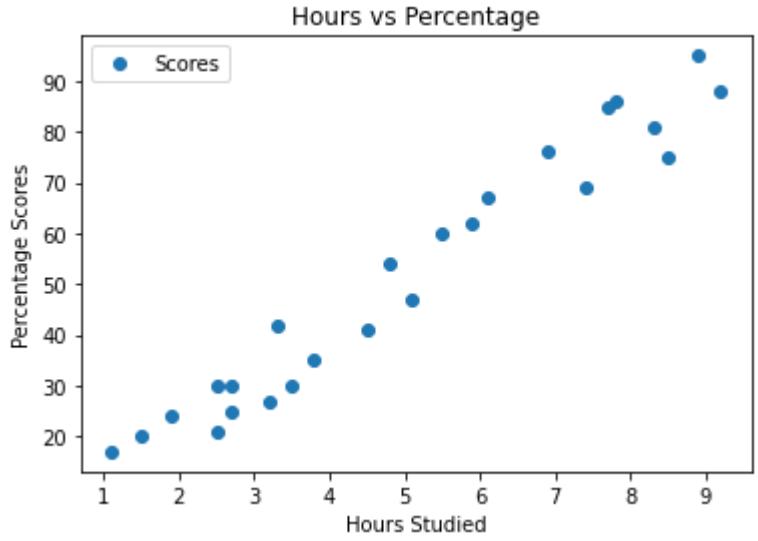
In [8]:
```
# This gives us the inference that no missing values are present in our dataset s

df.isna().sum()
```

Out[8]:
```
Hours     0
Scores    0
dtype: int64
```

In [9]:
```
# Analyzing the scores through plotting the distribution

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



From the above graph we can see that their exists a linear relationship between hours studied and percentage score.

Hence we can infer that Percentage scores increases as the number of study hours increases

In [10]:
```
# Checking the statistical results

df.describe()
```

Out[10]:

	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 [11]:
```
# Dividing the data into features(input) and labels(output)

x = df.iloc[:,:-1].values
y = df.iloc[:,1].values
```

In [12]:
```
x
```

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]:
```
y
```

Out[13]:
```
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 [14]:
```
# Training and test splitting

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 ,
```

Implementing Linear Regression Algorithm

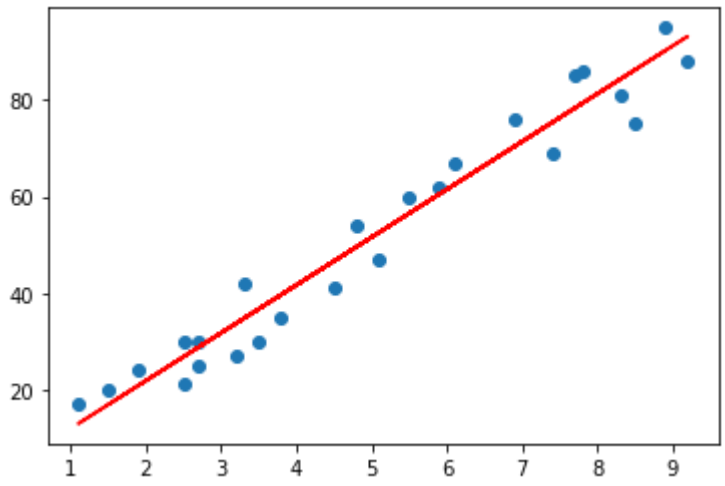
In [15]:
```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train , y_train) # training complete
```

Out[15]:
```
LinearRegression()
```

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

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



In [17]:
```
# Testing our Algorithm
print(X_test)
y_pred = regressor.predict(X_test)
```

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

In [18]:
```
# Creating a data frame of actual and predicted values
data_frame = pd.DataFrame({'Actual' : y_test , 'Predicted' : y_pred})
data_frame
```

Out[18]:

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

In [22]:
```
# Checking the percentage on the given data point(study hours = 9.25)

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

```
No of Hours = [[9.25]]
Predicted Score = 93.69173248737535
```

In [24]:
```
# Checking the performance of algorithm

from sklearn import metrics
print("Mean Absolute error is : " , metrics.mean_absolute_error(y_test , y_pred))
```

```
Mean Absolute error is :  4.183859899002975
```

Conclusion : Hence we concluded that if a student is involved in 9.25 hours of study per day , then their is a possibility that the percentage comes out to be 93.69

In [ ]:
```

```