To Explore Supervised Machine Learning

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 invoves just two variables.Data can be found at http://bit.ly/w-data (http://bit.ly/w-data)

In [1]:

```
#Importing necessary Libraries

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

In [2]:

```
#Reading the data to a dataframe
url = "http://bit.ly/w-data"
data = pd.read_csv(url)
print("Data imported successfully")

data.head(10)
```

```
Data imported successfully
```

Out[2]:

	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

In [3]:

```
data.head()
```

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.shape
```

Out[4]:

```
(25, 2)
```

In [5]:

```
data.describe()
```

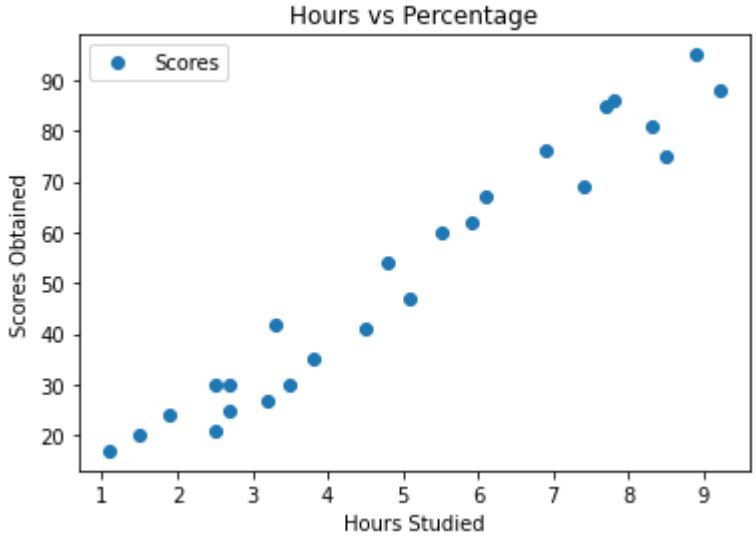
Out[5]:

	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

Here We have to visualize the Data to give a better understanding of the correlation between variables (Since the Dataset is quite small)

In [6]:

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



It is clearly seen that there is Positive linear relation between the number of hours studied and the marks obtained

In [7]:

```
#Now lets divide our data to independent and depedent variables
```

In [8]:

```
X = data.iloc[:, :-1].values
y = data.iloc[:, 1].values
```

In [9]:

```
from sklearn.model_selection import train_test_split
```

In [10]:

```

X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size=0.3, random_state=0)
```

Training the algorithm

In [11]:

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
```

In [12]:

```
reg.fit(X_train,y_train)
```

Out[12]:

```
LinearRegression()
```

In [13]:

```
print("Training is completed")
```

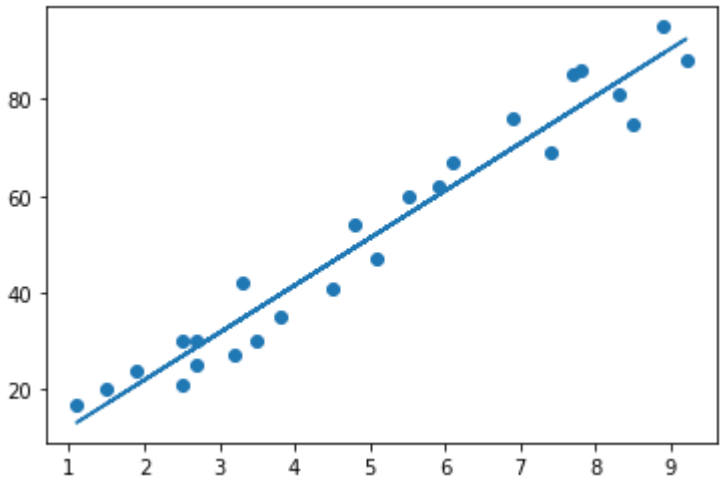
```
Training is completed
```

Now we have to visualize the linear regression(i.e. How line will fit the data)

In [14]:

```
line = reg.coef_*X+reg.intercept_


# Plotting for testing data
plt.scatter(X, y)
plt.plot(X, line);
plt.show()
```



In [15]:

```
#To retrieve the intercept and coefficient

print("Intercept is :")
print(reg.intercept_)
```

```
Intercept is :
2.370815382341881
```

In [16]:

```
print("coefficient is : ")
print(reg.coef_)
```

```
coefficient is :
[9.78856669]
```

This means that for every one unit of change in hours studied, The change in the score is about 9.78%.Or in

simpler words, if a student studies one hour more than they previously studied for an exam,they can expect to achieve an increase of 9.78% in the score achieved by the student previously.

Now lets first make predictions on testing data

In [17]:

```
y_pred = reg.predict(X_test)
```

In [18]:

```
y_pred
```

Out[18]:

```
array([17.05366541, 33.69422878, 74.80620886, 26.8422321 , 60.12335883,
       39.56736879, 20.96909209, 78.72163554])
```

The y_pred is a numpy array that contains all the predicted values for the input values in the X_test series

To compare the actual output values for X_test with the predicted values, execute the following script

In [19]:

```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[19]:

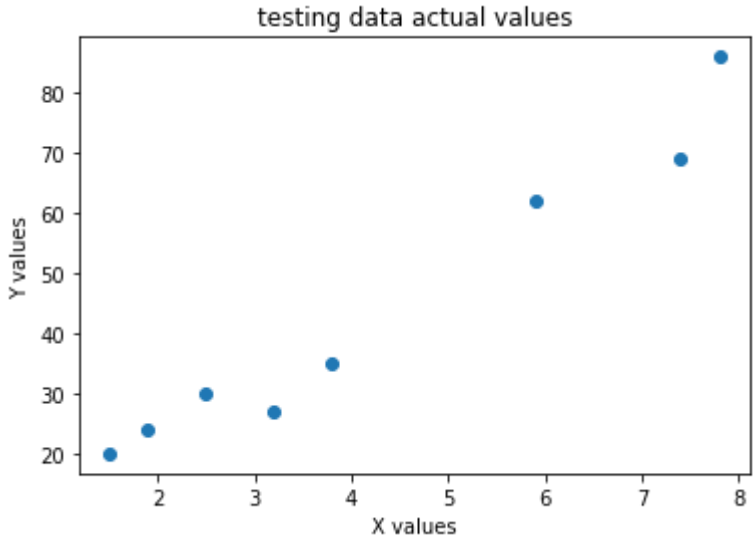
	Actual	Predicted
0	20	17.053665
1	27	33.694229
2	69	74.806209
3	30	26.842232
4	62	60.123359
5	35	39.567369
6	24	20.969092
7	86	78.721636

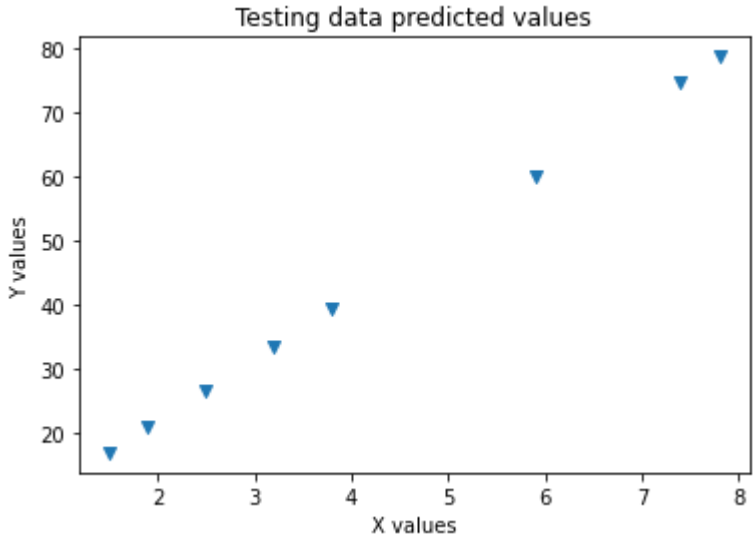
In [20]:

```
# Now lets visualize the predicted and actual values

plt.scatter(X_test,y_test)
plt.xlabel('X values')
plt.ylabel('Y values')
plt.title('testing data actual values')
plt.show()

plt.scatter(X_test,y_pred,marker='v')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.title('Testing data predicted values')
plt.show()
```





In [21]:

```
# You can also test with your own data
hours = [[9.25]]
k_pred = reg.predict(hours)
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(k_pred[0]))
```

```
No of Hours = [[9.25]]
Predicted Score = 92.91505723477056
```