Prediction using Supervised ML (Level - Beginner)

by

Priyanshi Khanna

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

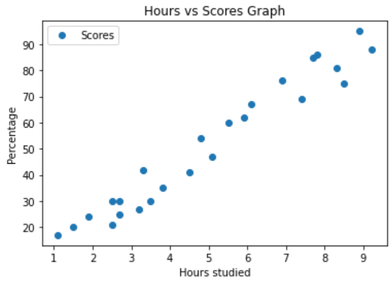
In [3]:
```
#importing the data
data= "http://bit.ly/w-data"
im_data=pd.read_csv(data)
print ("Data has been imported")
im_data.head()
```

```
Data has been imported
```

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]:
```
#Plotting the data points of scores vs hours on a 2-D graph to find any relationship between the two variables
im_data.plot(x="Hours",y="Scores",style="o")
plt.title("Hours vs Scores Graph")
plt.xlabel("Hours studied")
plt.ylabel("Percentage")
plt.show()
```



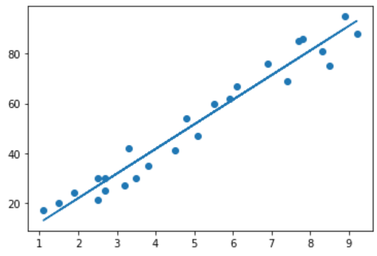
In [5]:
```
#Hence, we observe that there is a positive relationship between the number of hours studied and scores obtained
#WE will devide our data into attributes and labels
x=im_data.iloc[:,:1].values
#x represents the array of values in "Hours" column of the data
y=im_data.iloc[:,1].values
#y represents the array of values in "Scores" column of the data
```

In [6]:
```
#splitting the data into train and test sets using train_test_split() method of Scikit-Learn library
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,random_state=0)
#we have chosen 20% of the data for testing purpose and remaining 80% for training purpose
```

In [7]:
```
#training the model
from sklearn.linear_model import LinearRegression
lm= LinearRegression()
lm.fit(x_train,y_train)
print("Model has been trained")
```

```
Model has been trained
```

In [8]:
```
#Regression line
line= lm.coef_*x + lm.intercept_
#Plotting the test data and Regression line
plt.scatter(x,y)
plt.plot(x,line)
plt.show()
```



In [9]:
```
#Predicting through our trained algorithm
print(x_test)
#testing data in hours
Y=lm.predict(x_test)
#Predicting scores of testing data of hours on the basis of our trained model
```

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

In [10]:
```
#Comparing the actual values to our predicted values
df=pd.DataFrame({"Actual":y_test,"Predicted":Y})
df
```

Out[10]:

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

In [11]:
```
# What will be predicted score if a student studies for 9.25 hrs/ day?
hours=9.25
pred = lm.predict([[hours]])
print ("No. of Hours:{}".format(hours))
print ("Predicted score:{}".format(pred[0]))
```

```
No. of Hours:9.25
Predicted score:93.69173248737538
```

In [12]:
```
from sklearn import metrics
print("Mean Absolute Error:", metrics.mean_absolute_error(y_test,Y))
```

```
Mean Absolute Error: 4.183859899002975
```