Name:- Albina Rebello

Task 1 :- Prediction using Supervised ML

Question:- What will be predicted score if a student studies for 9.25 hrs/ day?

In [1]:
```
#Importing the required libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [3]:
```
#Importing the dataset for the analysis
data=pd.read_csv("http://bit.ly/w-data")
```

In [4]:
```
print(data)
```

```
   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 [5]:
```
data.head(5)
```

Out[5]:

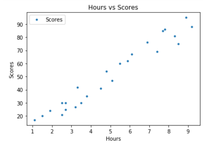
	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [6]:
```
#To get description of the data
data.describe()
```

Out[6]:

	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 [7]:
```
#Plot the distribution of scores and no.of hours of study
data.plot(x="Hours",y="Scores",style="-")
plt.title("Hours vs Scores")
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.show()
```



In [8]:
```
#Dividing into y(dependent) and x(independent) variable

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

In [10]:
```
#Splitting the entire datasets into training and testing datasets

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)
```

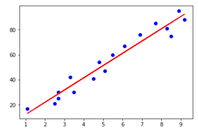
In [11]:
```
#To fit the model using Linear Regression

from sklearn.linear_model import LinearRegression
lreg=LinearRegression()
lreg.fit(x_train,y_train)
```

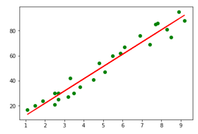
Out[11]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [12]:
```
#To print the values of intercept and coefficient of the fitted model using Linear Regression
print("Coefficient:")
print(lreg.coef_)
print("Intercept:")
```

```
Coefficient:
[9.78856669]
Intercept:
2.370815382341881
```

In [13]:
```
#To Visualise the model
#On the train dataset
line=lreg.coef_*x+lreg.intercept_
plt.scatter(x_train,y_train,color="blue")
plt.plot(x,line,color="red")
plt.show()
#On the entire dataset
line=lreg.coef_*x+lreg.intercept_
plt.scatter(x,y,color="green")
plt.plot(x,line,color="red")
plt.show()
```





In [14]:
```
#Making Prediction
y_predict=lreg.predict(x_test)
y_predict
```

Out[14]:
```
array([17.05366541, 33.69422878, 74.80620886, 26.8422321 , 60.12335883,
       39.56736879, 20.96909209, 78.72163554])
```

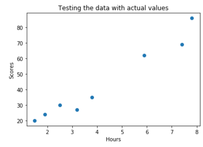
In [15]:
```
#Making a table of actual and predicted Scores and comparing it in test dataset
Df=pd.DataFrame({'Actual':y_test,'Predicted':y_predict})
Df
```

Out[15]:

	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

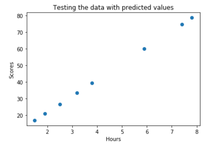
In [17]:
```
#Visualizing the graphs of actual and predicted y values

plt.scatter(x_test,y_test)
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Testing the data with actual values")
```

Out[17]:
```
Text(0.5,1,'Testing the data with actual values')
```



In [18]:
```
plt.scatter(x_test,y_predict)
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Testing the data with predicted values")
```

Out[18]:
```
Text(0.5,1,'Testing the data with predicted values')
```



In [19]:
```
#To find predicted score when no of hours studied is 9.25

hrs=np.array([9.25])
hrs=hrs.reshape(-1,1)
pred=lreg.predict(hrs)
print("Predicted Score for 9.25 hours of study:")
print(pred)
```

```
Predicted Score for 9.25 hours of study:
[92.91505723]
```

In [20]:
```
#Evaluation of the linear regression model

from sklearn import metrics
m=metrics.mean_absolute_error(y_test,y_predict)
print("Mean Absolute Error",m)
```

```
Mean Absolute Error 4.419727808027652
```

In [ ]:
```
from sklearn import metrics
s=metrics.mean_squared_error(y_test,y_predict)
print("Mean Squared Error",s)
```

Thankyou!