Task 1- Prediction using supervised ML

Author - Sanyukta Khatdeo

To predict the percentage of the students based on the number of hours they studied

In [2]:
```
# importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```
#Reading data
url = "https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_scores.csv"
data = pd.read_csv(url)
print("**Data Imported**")
data
```

Data Imported

Out[4]:

	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 [6]:
```
data.shape
```

Out[6]:
```
(25, 2)
```

In [7]:
```
data.describe()
```

Out[7]:

	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 [8]:
```
# Check if there is any null value in the Dataset
data.isnull == True
```

Out[8]:
```
False
```

there is no null in the dataset hence we can now visualize

Visualization and Analysis of Dataset

In [39]:
```
#plotting of distribution of scores and number of hours of study on 2D graph

data.plot(x='Hours', y='Scores', style='o')
plt.title('no. of Hours studied Vs Scores of Students')
plt.xlabel('no. of Hours studied')
plt.ylabel('Scores of students')
plt.show()
```

In [20]:
```
#no. of hours studied = x variable
#scores = y variable
X = data.iloc[:, :-1].values
Y = data.iloc[:, 1].values
```

In [21]:
```
#view X variable
X
```

Out[21]:
```
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 [22]:
```
#view Y variable
Y
```

Out[22]:
```
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 [ ]:
```

```

In [25]:
```
#for splitting data into training and test sets

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)
```

In [27]:
```
X_train
```

Out[27]:
```
array([[3.8],
       [1.9],
       [7.8],
       [6.9],
       [1.1],
       [5.1],
       [7.7],
       [3.3],
       [8.3],
       [9.2],
       [6.1],
       [3.5],
       [2.7],
       [5.5],
       [2.7],
       [8.5],
       [2.5],
       [4.8],
       [8.9],
       [4.5]])
```

In [28]:
```
X_test
```

Out[28]:
```
array([[1.5],
       [3.2],
       [7.4],
       [2.5],
       [5.9]])
```

In [29]:
```
Y_train
```

Out[29]:
```
array([35, 24, 86, 76, 17, 47, 85, 42, 81, 88, 67, 30, 25, 60, 30, 75, 21,
       54, 95, 41], dtype=int64)
```

In [30]:
```
Y_test
```

Out[30]:
```
array([20, 27, 69, 30, 62], dtype=int64)
```

Training of Machine Learning model(Alogorithm)

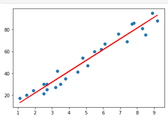
In [31]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,Y_train)
print("****Training of ML model is Completed****")
```

```
****Training of ML model is Completed****
```

Visualizing the Model

In [34]:
```
#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()
```



Making Predictions

In [35]:
```
print(X_test)  #testing data - in hours
Y_pred = regressor.predict(X_test)  #predicting the scores
```

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

In [36]:
```
#Comparing actual Vs Predicted Data
df = pd.DataFrame({'Actual':Y_test, 'Predicted':Y_pred})
df
```

Out[36]:

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

In [37]:
```
#testing with our own custom data
#score of student if he/she studies for 9.25 hrs/day

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
```

Evaluating the Model

In [38]:
```
#mean absolute error:

from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, Y_pred))

#max error:
print('Max Error:', metrics.max_error(Y_test, Y_pred))

#mean squared error:
print('Mean Squared Error:', metrics.mean_squared_error(Y_test, Y_pred))
```

```
Mean Absolute Error: 4.183859899002975
Max Error: 6.732260779489849
Mean Squared Error: 21.5987693072174
```

In [ ]:
```

```