Data Science and Business Analytic

Name : URVI DALAL

Task 1 : Prediction using Supervised ML

Objective :-

Predict the percentage of an student based on the no. of study hours.</b>

Importing all libraries required in this notebook

In [2]:

```
import pandas as pd                                     #for data
 manipulation and working with csv files
import numpy as np                                      #for numer
ical manipulation
import matplotlib.pyplot as plt                         #for polti
ng graphs
%matplotlib inline
from sklearn.model_selection import train_test_split    #for split
ing dataset
from sklearn.linear_model import LinearRegression       #for linea
r regression
```

Reading data from remote link

In [6]:

```
#Reading data the dataset to perform operation
url = "http://bit.ly/w-data"
df = pd.read_csv(url)
print("Data Successfully loaded")
```

```
Data Successfully loaded
```

In [7]:

```
#Reading Dataset
df
```

Out[7]:

	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 [9]:

```
#checking the shape of the dataset
df.shape
```

Out[9]:

```
(25, 2)
```

In [10]:

```
#Reading the first 10 observation
df.head(10)
```

Out[10]:

	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 [11]:

```
#Reading the last 10 observation
df.tail(10)
```

Out[11]:

	Hours	Scores
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 [12]:
```
#checking numerical data
df.describe()
```

Out[12]:

	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 [14]:
```
#checking correlation between columns
df.corr()
```

Out[14]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

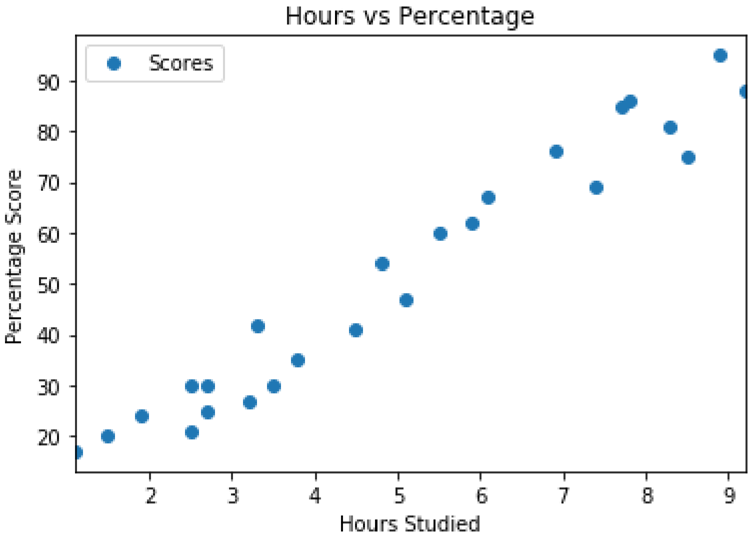
In [16]:
```
#checking the null values in the dataset
df.isnull().sum()
```

Out[16]:
```
Hours     0
Scores    0
dtype: int64
```

Now,let us plot a graph using matplotlib to understand the relation between columns.

In [18]:

```
# Plotting the distribution of scores
df.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



From the graph above, we can clearly see that there is a positive linear relation between the number of hours studied and percentage of score.
Hence we can say that the Percentage scores increases as the number of studied hours increases

Preparing the data

We are going to divide this dataset columns into "attributes"(inputs) and "labels" (Outputs).

In [24]:

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

In [25]:

```
X
```

Out[25]:

```
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 [26]:

```
y
```

Out[26]:

```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 1
7, 95, 30,
       24, 67, 69, 30, 54, 35, 76, 86], dtype=int64)
```

In [27]:

```
# Splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size=0.2, random_state=0)
```

```
In [28]: X_train
```

```
Out[28]: 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 [29]: X_test
```

```
Out[29]: array([[1.5],
                [3.2],
                [7.4],
                [2.5],
                [5.9]])
```

```
In [30]: y_train
```

```
Out[30]: array([35, 24, 86, 76, 17, 47, 85, 42, 81, 88, 67, 30, 25, 60, 3
         0, 75, 21,
                54, 95, 41], dtype=int64)
```

```
In [31]: y_test
```

```
Out[31]: array([20, 27, 69, 30, 62], dtype=int64)
```

Training our model and implementing linear regression algorithm

In [33]:

```
LinearModel = LinearRegression()  # Instantiating the LinearRegression Model
LinearModel.fit(X_train, y_train) # Plotting the Linear Regression Line
```

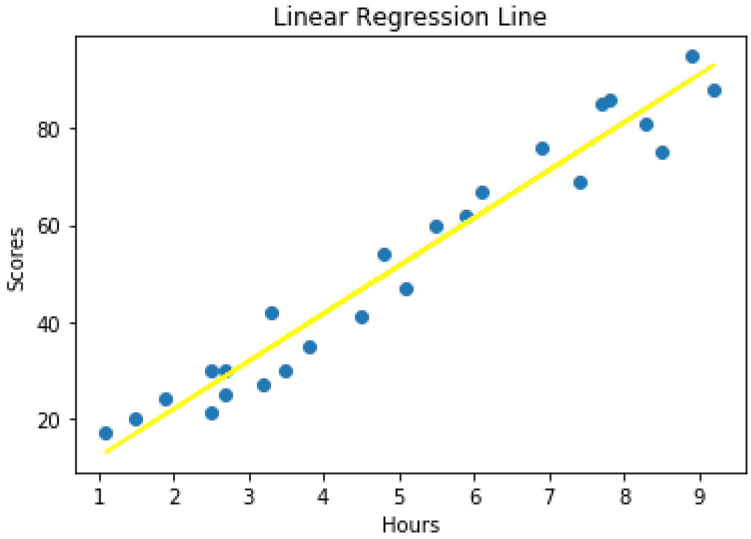
Out[33]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [40]:

```
#Ploting the linear regression line
line = LinearModel.coef_*X+LinearModel.intercept_

# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line, color = "yellow");
plt.xlabel("Hours");
plt.ylabel("Scores");
plt.title("Linear Regression Line")
plt.show()
```



Testing our linear Regression Model

In [41]:

```
print(X_test) # Testing data - In Hours
y_pred = LinearModel.predict(X_test) # Predicting the scores
```

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

In [44]:

```
# Comparing the Actual and Predicted Values by storing them in dataframe

compare= pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
compare
```

Out[44]:

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

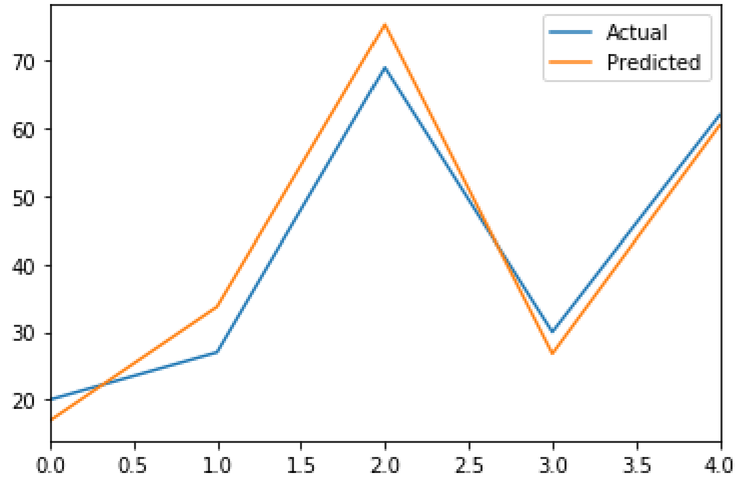
In [51]:

```
# Plotting the graph for actual vs predicted values

compare.plot()
```

Out[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ab075a6808>
```



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

In [56]:

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

```
No of Hours = 9.25
Predicted Score = 93.69173248737539
```

Hence we can concluded that if a student is involved in 9.25 hours per day , then there is a possibility that the percentage comes out to be 93.69173248737539.

Evaluation Of Our Linear Regression Model

The final step is to evaluate the performance of algorithm. This step is particularly important to compare how well different algorithms perform on a particular dataset. For simplicity here, we have chosen the mean square error. There are many such metrics

In [57]:

```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002982
```