Task-1

Analysis of percentage of an student based on the no. of study hours

`Import the necessary libraries`

In [29]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

`storing the data in a csv file`

In [2]:
```
data=pd.read_csv("http://bit.ly/w-data")
```

In [3]:
```
data
```

Out[3]:

	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

`Description of the dataset`

In [4]:
```
data.head()
```

Out[4]:

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

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

In [6]:
```
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

=> The dataset contains 25 rows and 2 columns

=> The first column denote the number of hour spend by the students.It contains quantitative values.

= >The second column denote the marks obtained by the student.It contains quantiative values.

= >There is zero null values in both columns.

`Plotting the data`

In [7]:
```
data['Hours'].unique()
```

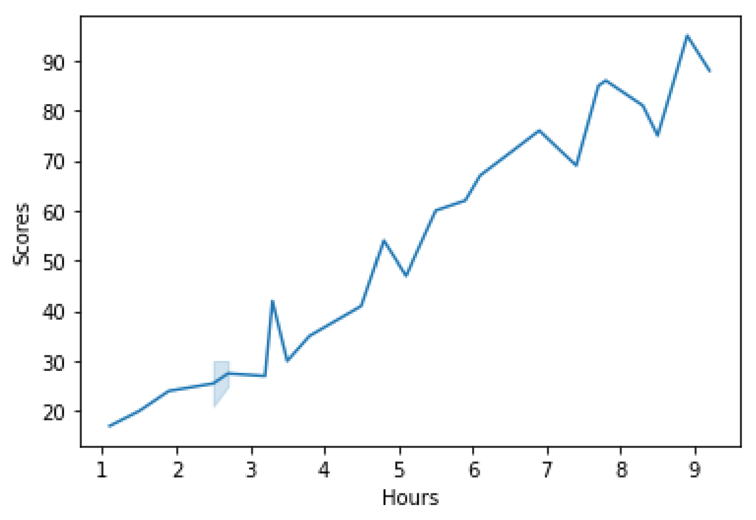
Out[7]:
```
array([2.5, 5.1, 3.2, 8.5, 3.5, 1.5, 9.2, 5.5, 8.3, 2.7, 7.7, 5.9, 4.5,
       3.3, 1.1, 8.9, 1.9, 6.1, 7.4, 4.8, 3.8, 6.9, 7.8])
```

In [8]:
```
data['Scores'].unique()
```

Out[8]:
```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95, 24,
       67, 69, 54, 35, 76, 86], dtype=int64)
```

In [9]:
```
sns.lineplot(data=data, x="Hours", y="Scores")
```

Out[9]:
```
<AxesSubplot:xlabel='Hours', ylabel='Scores'>
```



We can clearly notice a linear relation between hours and scores

`splitting the dataset into train and test dataset`

In [22]:
```
X_data=data.iloc[:, :-1].values
Y_data=data.iloc[:, 1].values
```

In [23]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data, Y_data, test_size=0.2, random_state=0)
```

Applying the Linear Regression Model

In [24]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
y_pred=regressor.predict(X_test)
```

In [26]:
```
y_pred
```

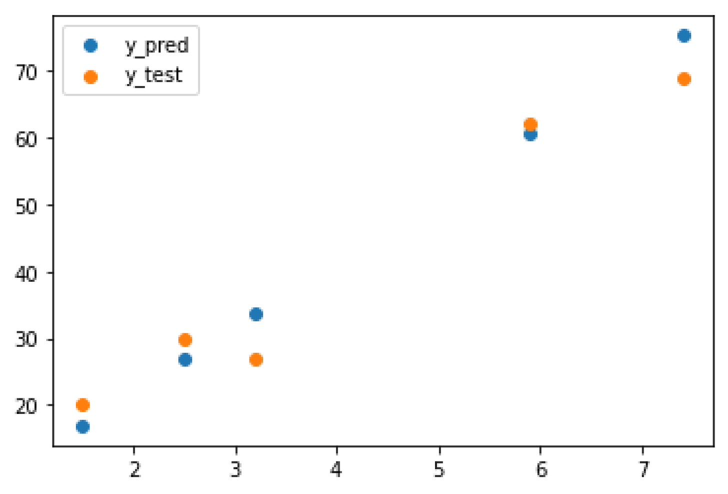
Out[26]:
```
array([16.88414476, 33.73226078, 75.357018  , 26.79480124, 60.49103328])
```

In [27]:
```
y_test
```

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

In [36]:
```
plt.scatter(X_test,y_pred)
plt.scatter(X_test,y_test)
plt.legend(['y_pred','y_test'])
```

Out[36]:
```
<matplotlib.legend.Legend at 0x1e2640a4b20>
```



Evaluating the model

In [33]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 4.183859899002982
Mean Squared Error: 21.598769307217456
Root Mean Squared Error: 4.647447612100373
```

In [ ]: