GRIP OCTOBER

THE SPARKS FOUNDATION

TASK 1: Prediction using Supervised ML

This is the task 1 performed by Saksham Sharma in the intership #GRIPOCTOBER under THE SPARKS FOUNDATION

Simple Linear Regression

In this regression task we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied. This is a simple linear regression task as it involves just two variables.

Problem Statement

What will be the predicted score when the student studies for 9.25 hrs/day

Loading and analyzing the data

In [105]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [106]:

```
data=pd.read_excel(r'D:/Saksham 5th sem/Sparks internship tasks/Task1dataset.xlsx')
data.head()
```

Out[106]:

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

In [107]:

```
data.describe()
```

Out[107]:

	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

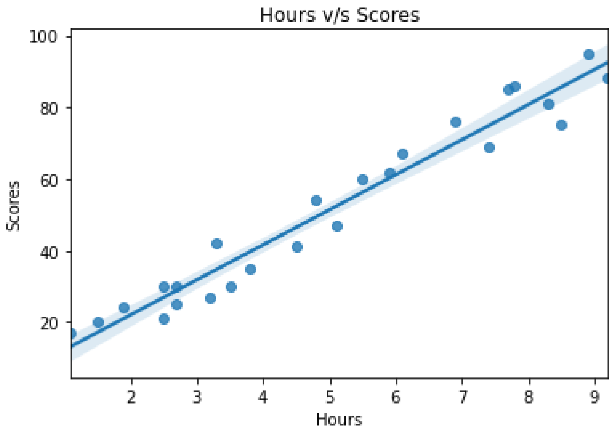
In [108]:

```
x=np.array(data[['Hours']])
y=np.array(data['Scores'])
```

In [109]:

```
sns.regplot(x = 'Hours', y = 'Scores', data = data)
plt.title("Hours v/s Scores")
plt.ylabel("Scores")
```

```
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.

Training the model

After loading the data, the next step is to train the regression model with the scores of the data

In [173]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.27,random_state=0)
```

In [174]:

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

Out[174]:

```
LinearRegression()
```

In [175]:

```
print("Coefficient: ", slr.intercept_)
print("Constant: ", slr.coef_)
```

```
Coefficient:  1.932204253151646
Constant:  [9.94167834]
```

In [176]:

```
print(x_test)
y_pred_slr= slr.predict(x_test)
print("Prediction for test set: {}".format(y_pred_slr))
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]
 [3.8]
 [1.9]]
Prediction for test set: [16.84472176 33.74557494 75.50062397 26.7864001  60.58810646 39.71058194
 20.8213931 ]
```

Regression Equation y=slr.coef *x+slr.intercept*

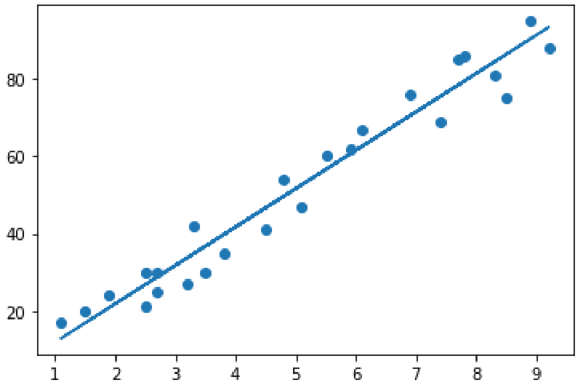
In [177]:

```
slr_diff = pd.DataFrame({'Actual value': y_test, 'Predicted value': y_pred_slr})
print(slr_diff.head())
```

```
   Actual value  Predicted value
0            20        16.844722
1            27        33.745575
2            69        75.500624
3            30        26.786400
4            62        60.588106
```

In [178]:

```
ysol=slr.coef_*x+slr.intercept_
# Plotting for the test data
plt.scatter(x,y)
plt.plot(x,ysol)
plt.show()
```



Output for x = 9.25 hours a day

Prediction using in-built function

In [179]:

```
hours = np.array(9.25)
res_pred = slr.predict(hours.reshape(-1,1))
print('Predicted Score:', res_pred, sep='\n')
```

Predicted Score:
[93.89272889]

Prediction using regression line

In [180]:

```
hours=9.25
score=slr.coef_*hours+slr.intercept_
print('Predicted Score:', score, sep='\n')
```

Predicted Score:
[93.89272889]

Evaluating the 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 [181]:

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