Task-1

Predicting the percentage of marks based on number of study hours

This is simple Linear regression task in this only two variables are present.

In [1]:

```
## Importing all necessary libraries,required for simple linear regression task
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:

```
## Reading data from url
url = "https://bit.ly/w-data"
student_data = pd.read_csv(url)
student_data.head() ## head function returns by default 5 rows of raw data
```

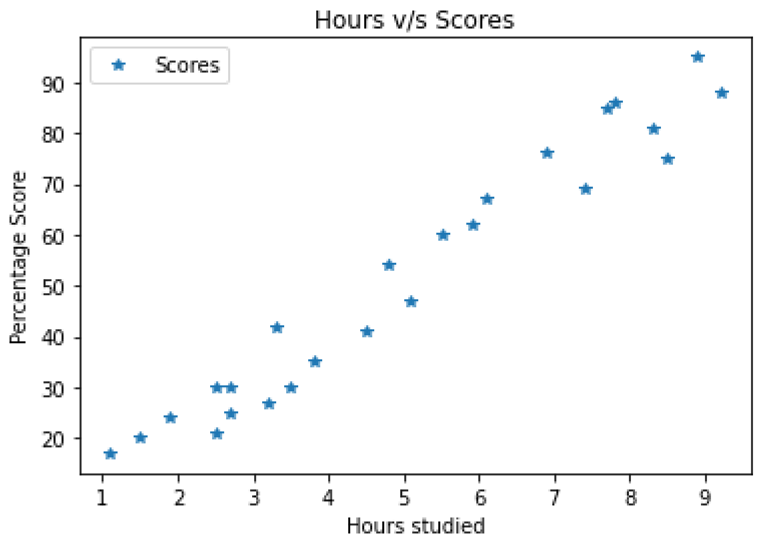
Out[3]:

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

Plot 2D graph based on the data

In [4]:

```
# Plotting the scatter plot to visualize the data
student_data.plot(x = "Hours",y = "Scores",style = '*')
plt.title("Hours v/s Scores")
plt.xlabel("Hours studied")
plt.ylabel("Percentage Score")
plt.show()
```



In above graph we can cleary see that the there is positive linear regression between the no of hours studied and percentage score

Preparing Data

In [5]:

```
X = student_data.iloc[:,:-1].values  ## Here we divided the data into two parts attributes(input) and l
ables(output)
Y = student_data.iloc[:,1].values
```

In [6]:

```
from sklearn.model_selection import train_test_split    ##Here we split the data into two parts trainin
g and testing
X_train, X_test, y_train, y_test = train_test_split(X, Y,
                            test_size=0.2, random_state=0)
```

Train Algorithm For Data

In [7]:

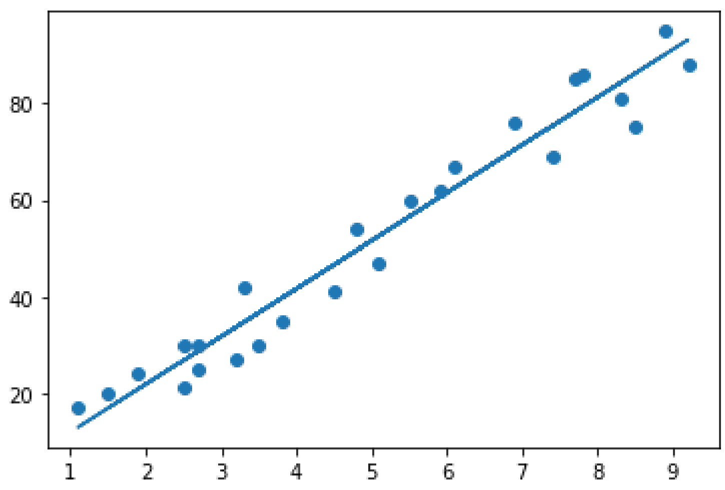
```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(X_train, y_train)
print("Training Complete")
```

```
Training Complete
```

In [8]:

```
line = regression.coef_*X+regression.intercept_  ## Draw line of linear regression

# Plotting for the test data
plt.scatter(X, Y )
plt.plot(X, line);
plt.show()
```



In [9]:

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

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

In [10]:

```
# Comparing Actual vs Predicted
s_data = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
s_data
```

Out[10]:

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

Evaluating the model

Here we use mean square method to evaluate the model

In [11]:

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

```
Mean Absolute Error: 4.183859899002982
```

In [ ]: