Name : Gadha T

Task 1 :

PREDICTION USING SUPERVISED MACHINE LEARNING

GRIP @ The Sparks Foundation

In this task, Im trying to predict the score percentage of students based upon the number of hours they studied.

Importing all libraries required in this notebook

In [22]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
%matplotlib inline
```

Step 1 : Reading data from the source

In [11]:

```
url = r"https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_scores.csv
s_data = pd.read_csv(url)
print("The data imported successfully")

s_data.head(10)
```

```
Data imported successfully
```

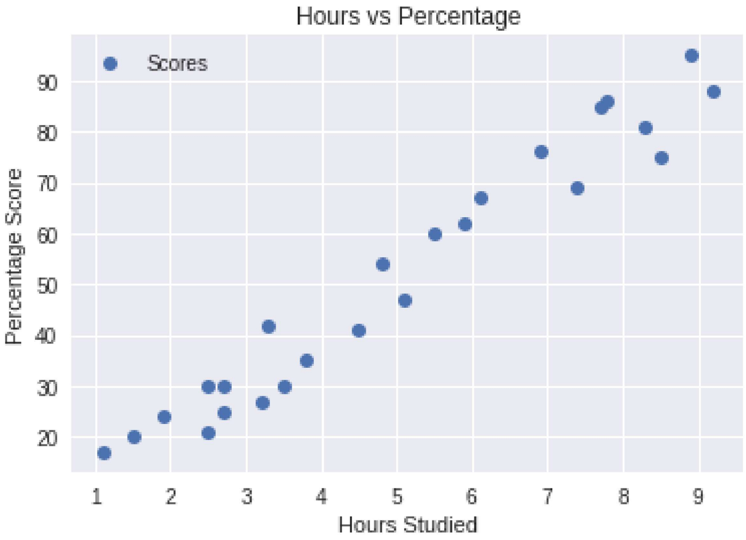
Out[11]:

	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

Step 2 : The distribution plot of scores

In [0]:

```
s_data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



The graph obtained clearly shows that there is a positive linear relation between the number of hours studied and percentage of score.

Step 3: Preparing the data

The next step divides the data into "attributes" (inputs) and "labels" (outputs).

In [18]:

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

Step 4 : Training the Model

Here, we split this data into training and test sets. And then, We'll do this by using Scikit-Learn's built-in train_test_split() method:

In [26]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size=0.2, random_state=0)
```

Training the Algorithm

We have split our data into training and testing sets, and now is finally the time to train our algorithm.

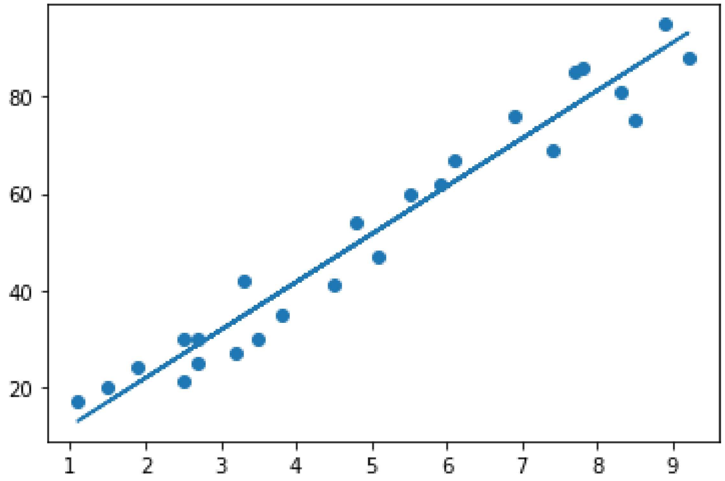
In [25]:

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training is completed.")
```

```
Training is completed.
```

Step 5 : Plotting the regression line

In [15]:
```
line = regressor.coef_*X+regressor.intercept_

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



Step 6 : Making Predictions

Now that we have trained our algorithm, it's time to make some predictions.

In [8]:

```
#Testing data - In Hours
print(X_test)

#Predicting the scores
y_pred = regressor.predict(X_test)
```

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

Step 6 : Comparing the Actual model vs the Predicted model

In [9]:

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

Out[9]:

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

In [27]:

```
#Estimating training and testing the score

print("Training Score: ",regressor.score(X_train,y_train))
print("Test Score: ",regressor.score(X_test,y_test))
```

```
Training Score:  0.9515510725211552
Test Score:  0.9454906892105356
```

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 [23]:

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

```
Mean Absolute Error: 4.183859899002975
```

Conclusion

I was succesful in completing the Prediction using Supervised Machine leraning and is now able to evaluate the model's performance.