PRANAV SHARAN

Task 1 Prediction using Supervised ML

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.

We create a linear regression module to help us achieve this as the task involves only two variables.

Importing all required libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Loading the dataset

In [3]:
```
df = pd.read_csv("student_scores.csv")
print("Data imported successfully")

df.head(10)
```

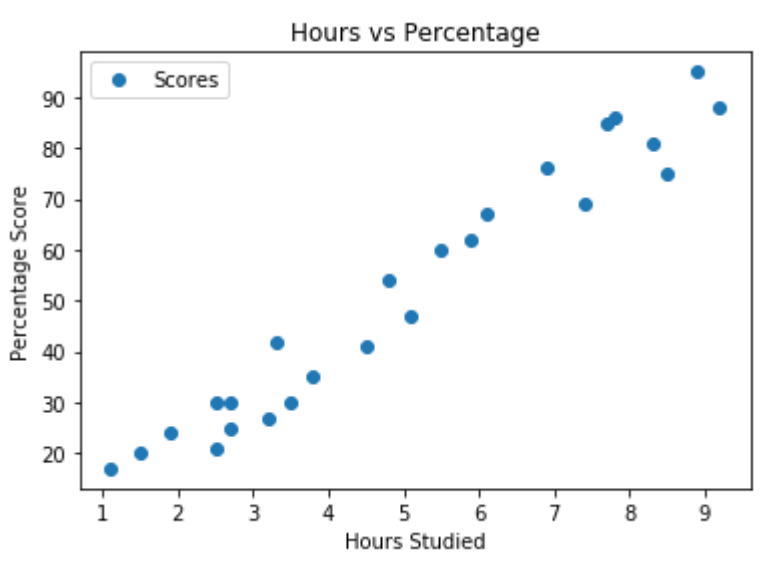
```
Data imported successfully
```

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

We plot our data points on 2-D graph to see if we can find any relationship between the Independent Variable and Dependent Variable. We use matplotlib to plot the graph.

In [4]:
```
df.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



We can see that there is a positive linear relation between the number of hours studied and percentage of score.

Preparing the data

In [5]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, 1].values
```

Now that we have our attributes and labels, the next step is to split this data into training and test sets. We'll do this by using Scikit-Learn's built-in train_test_split() method.

In [6]:
```
from sklearn.model_selection import train_test_split

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

Fitting the Model

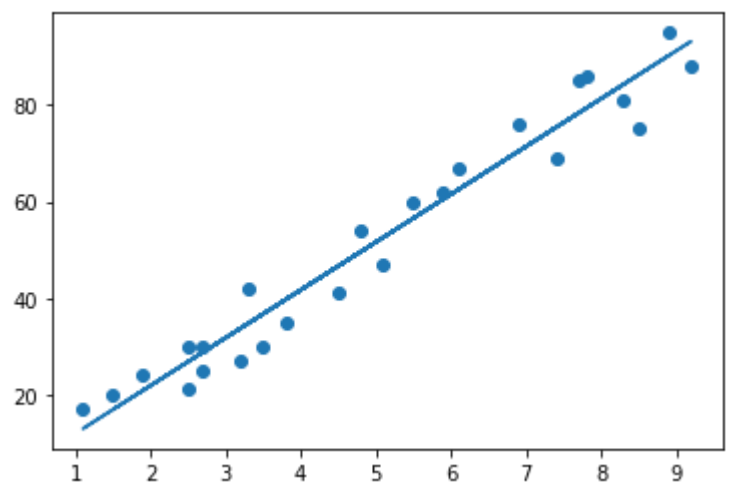
We have split our data into training and testing sets, and now is finally the time to train our algorithm.

In [8]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[8]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [9]:
```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_

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



Making Predictions

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

In [10]:
```
y_pred = regressor.predict(X_test)
df_pred = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df_pred
```

Out[10]:

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

In [14]:
```
hours = 9.25
own_pred = regressor.predict([[hours]])
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(own_pred[0]))
```

```
No of Hours = 9.25
Predicted Score = 93.69173248737538
```

Evaluating the model

In [15]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```

In [ ]:
```

```