NAME: Srivally Garimella

Task 1: Prediction Using Supervised ML

In this we will predict the percentage of a student based on the no of study hours

Step 1: Import all the required libraries using the following commands

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Step 2: Read the data from the given URL

In [2]:

```
url = "http://bit.ly/w-data"
s_data = pd.read_csv(url)
print("Data imported successfully")

s_data.head(10)
```

```
Data imported successfully
```

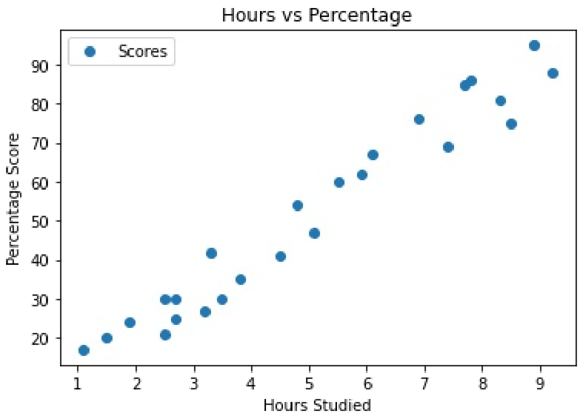
Out[2]:

	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 3: Visualizing the data by plotting a graph

In [3]:

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



It can be observed that there is a positive linear relation between the no of hours studied and the percentage scored

.

Step 4: Divide the data into "attributes" (inputs) and "labels" (outputs)

In [4]:

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

Step 5: Split this data into training and test sets

Step 5: Split this data into training and test sets

In [5]:

```
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)
```

Step 6: Training the Algorithm

In [6]:

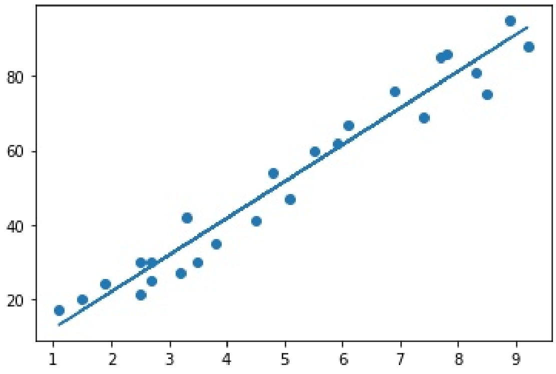
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training complete.")
```

```
Training complete.
```

Step 7: Plotting the regression line for the test data

In [7]:

```
line = regressor.coef_*X+regressor.intercept_
plt.scatter(X, y)
plt.plot(X, line);
plt.show()
```



Prediction

In [8]:

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

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

In [9]:

```
# Comparison between Actual and Predicted
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 [11]:

```
# You can also test with your own data
hours = 9.25
test= np.array([hours])
test=test.reshape(-1,1)
own_pred = regressor.predict(test)
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(own_pred[0]))
```

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

Step 8: Evaluation of Model using Mean square Error

In [13]:

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

```
Mean Absolute Error: 4.183859899002975
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js