GRIP - The Sparks Foundation

TASK - Prediction using Supervised ML

This Dataset is about the study hours and scores of students

Importing required Libraries

In [1]:

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

Loading the Dataset from remote link

In [5]:

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

data.head(10)
```

```
Data imported successfully
```

Out[5]:

	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

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

In [8]:

```
data.describe()
```

Out[8]:

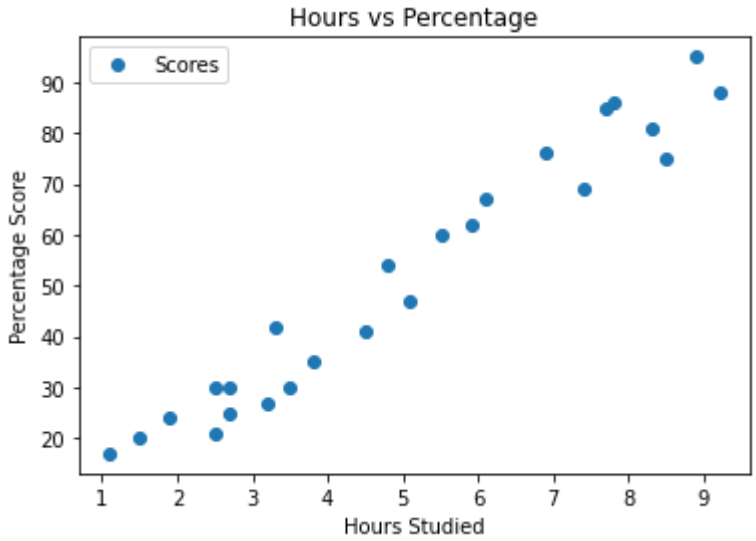
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

Plotting The Distribution Of Scores

Let's plot our data points on 2-D graph to eyeball our dataset and see if we can manually find any relationship between the data. We can create the plot with the following script:

In [9]:

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

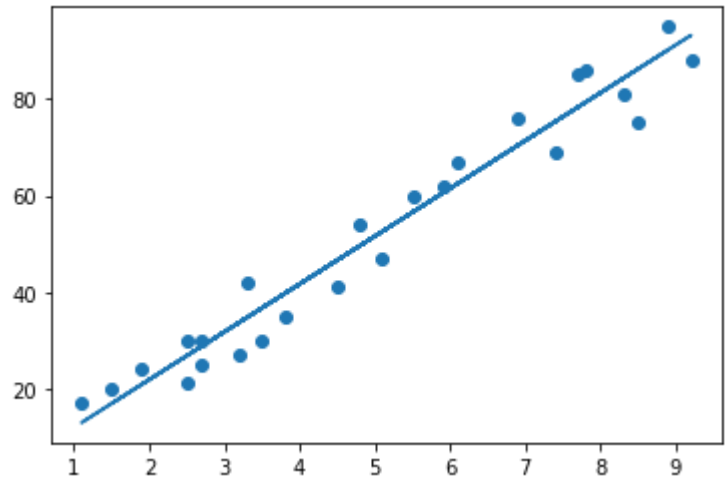


Plotting The Regression Line

In [44]:

```
line = regressor.coef_*X+regressor.intercept_

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



Splitting The Data

In [45]:

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

In [32]:

```
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)
```

Training the Model

In [33]:

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

print("Training complete.")
```

```
Training complete.
```

Predicting the Values

In [34]:

```
y_pred = regressor.predict(X_test)
```

Comparing the Result

In [40]:

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

Out[40]:

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

Finding MAE

In [42]:

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

```
Mean Absolute Error: 4.183859899002975
```

Calculating Accuracy

In [43]:

```
regressor.score(X_test, y_test)
```

Out[43]:

```
0.9454906892105356
```

Predicted score if a student studies for 9.25 hrs/ day

In [46]:

```
x= np.array(9.25)
x_new = x.reshape(1,-1)
regressor.predict(x_new)
```

Out[46]:

```
array([93.69173249])
```

By - Shivangi Chauhan

In [ ]:

```

```