By : Prajakta Bhavsar

GRIP : The Spark Foundation #GRIP JULY21

Data Science and Business Analytics

Taks 1 : Prediction Using Supervised ML

Objective : What will be predicted score if a student studies for 9.25 hrs/day?

Dataset URL : http://bit.ly/w-data

In [1]:
```
# Importing all libraries required in this notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
# Reading data from remote link
url = "http://bit.ly/w-data"
s_data = pd.read_csv(url)
print("Data imported successfully")

s_data.head(5)
```

```
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

In [3]:
```
s_data.shape
```

Out[3]:
```
(25, 2)
```

In [4]:
```
s_data.describe()
```

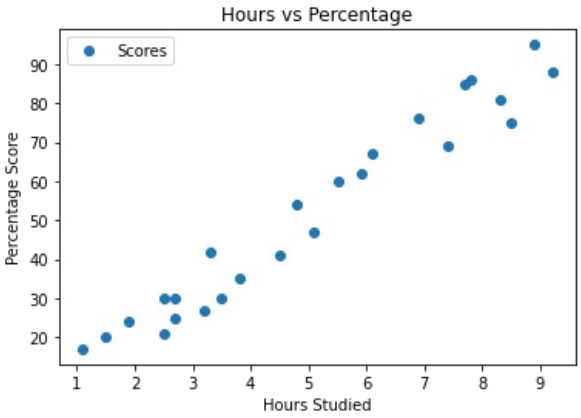
Out[4]:

	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

In [5]:
```
s_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 [6]:
```
# Plotting the distribution of scores
s_data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



From the graph above, we can clearly see that there is a positive linear relation between the number of hours studied and percentage of score.

In [7]:
```
#PREPARING THE DATA:

#Selecting the values of data from the dataframe
x = s_data.iloc[:, :-1].values
y = s_data.iloc[:, 1].values

#Splitting the data values obtained into training and testing samples:
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)
```

In [8]:
```
#TRAINING THE MODEL

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
print("Data Trained Successfully!")
```

```
Data Trained Successfully!
```

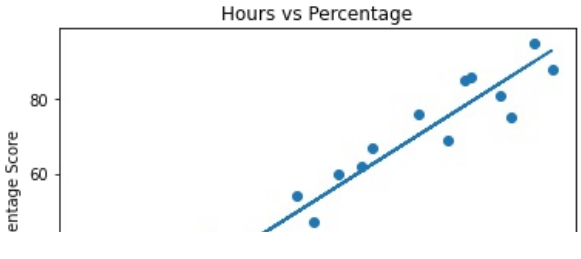
In [9]:
```
#Finding the slope and intercept for the Regression line
slope = regressor.coef_
interc = regressor.intercept_
print("Slope: ", slope,"\nY-Intercept: ", interc)
```

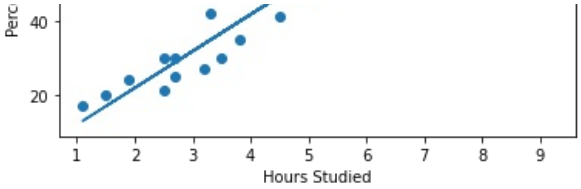
```
Slope:  [9.91065648]
Y-Intercept:  2.018160041434683
```

In [10]:
```
# Plotting the regression line
line = slope*x + interc

# Plotting for the test data
plt.scatter(x, y)
plt.plot(x, line);
plt.title("Hours vs Percentage")
plt.xlabel("Hours Studied")
plt.ylabel("Percentage Score")
plt.show()

print(slope)
```






```
[9.91065648]
```

In [11]:
```
#MAKING PREDICTIONS

print("Hours studied\n", x_test) # Testing data - In Hours
y_pred = regressor.predict(x_test) # Predicting the scores

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

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

Out[11]:

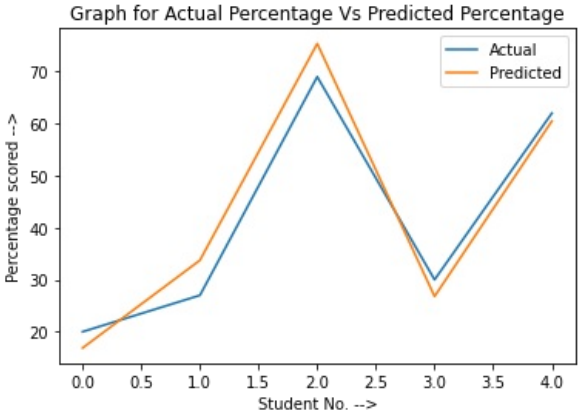
	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [12]:
```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[12]:

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

In [13]:
```
df.plot(kind = "line")
plt.title("Graph for Actual Percentage Vs Predicted Percentage")
plt.xlabel("Student No. -->")
plt.ylabel("Percentage scored -->")
plt.show()
```



Predicting the Percentage of a student who studied for 9.25 hours.

In [14]:
```
# Predict function can be used to calculate the student score if the given amount of time he studied is given.
hours = 9.25
fin_pred = round(regressor.predict([[9.25]]) [0],2)
print("No of Hours studied = {}".format(hours))
print("Predicted Score = {}".format(fin_pred))
```

```
No of Hours studied = 9.25
Predicted Score = 93.69
```

In [15]:
```
#EVALUATING THE MODEL

from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 4.183859899002975
Mean Squared Error: 21.5987693072174
Root Mean Squared Error: 4.6474476121003665
```