GRIP@The Spark Foundation- Data Science & Business Analytics Internship

Author - Samarth Waghmode

Task 1: Prediction using Supervised ML

Dataset used: Student Scores

It can be downloaded through the following link - http://bit.ly/w-data (http://bit.ly/w-data)

Problem Statement(s) :

*** Predict the percentage of a student based on the no. of study hours.

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

Import necessary libraries

In [1]:

```
# Importing Libraries required for data analysis
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

Read the data from Dataset

In [3]:

```
#Reading the data from Dataset
url = "http://bit.ly/w-data"
data = pd.read_csv(url)
```

In [4]:

```
data
```

Out[4]:

	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
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [5]:

```
data.head(10) # top 10 rows
```

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.shape  # view the shape i.e. number of rows, columns
```

Out[6]:

```
(25, 2)
```

In [7]:

```
data.info() #to get the summary of dataframe
```

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

In [8]:

```
data.describe() #Summary Of Statistics
```

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

In [9]:

```
data.size #Size of dataframe which is calculated by number of rows and columns
```

Out[9]:

```
50
```

In [10]:

```
data.dtypes
```

Out[10]:

```
Hours      float64
Scores       int64
dtype: object
```

In [11]:

```
data.corr() #to find correlation
```

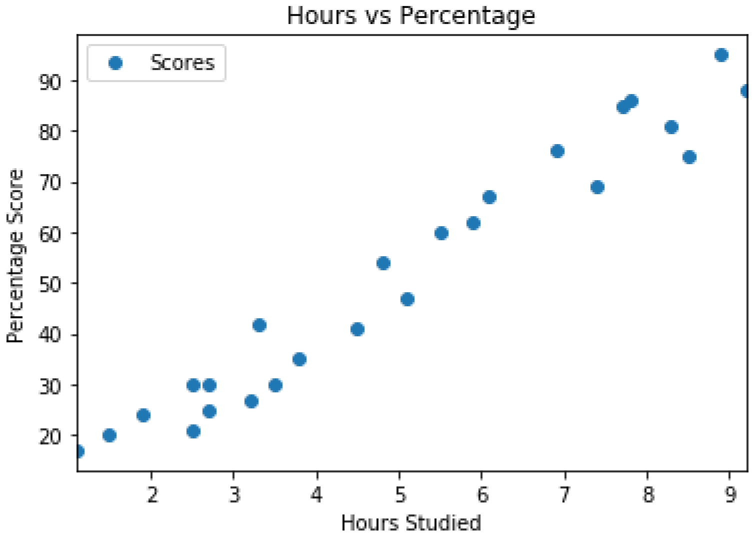
Out[11]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

Visualize the data

In [12]:

```
# Plotting the graph for distribution of scores
data.plot(x='Hours', y='Scores',style='o')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.title('Hours vs Percentage')
plt.show()
```



It is evident from the graph that there is a positive linear relation between the number of hours studied and percentage of score.

Prepare the data

In [13]:

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

In [14]:

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

Train the Algorithm

In [15]:

```
#training the data
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("The data has been trained")
```

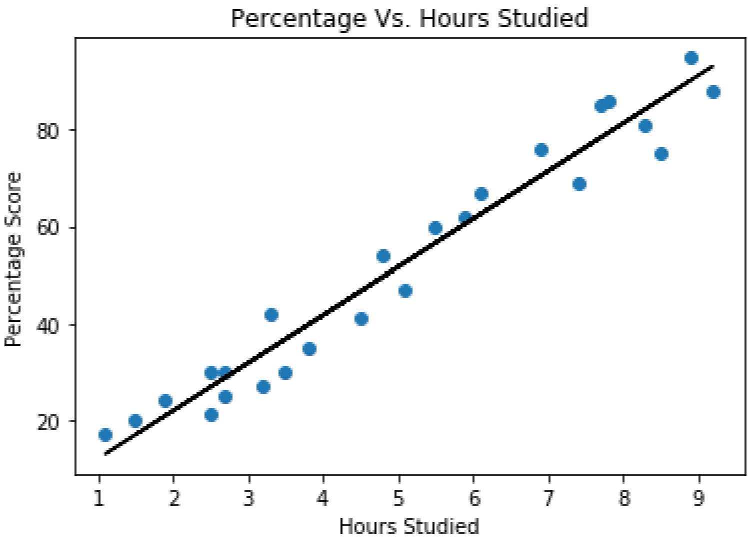
```
The data has been trained
```

In [16]:

```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_
```

In [17]:

```
# Plotting for the test data
plt.scatter(X,y)
plt.plot(X,line, color ='black')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.title('Percentage Vs. Hours Studied')
plt.show()
```



Making Predictions

In [18]:

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

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

In [19]:

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

Compare Actual vs Predicted Score

In [20]:

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

Out[20]:

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

Estimate Training and Test Score

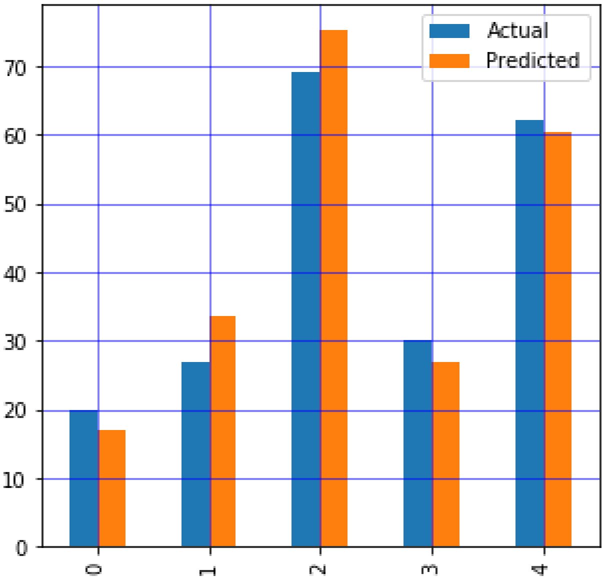
In [21]:

```
# Estimating training and test score
print("Training Score",regressor.score(X_train,y_train))
print("Test Score",regressor.score(X_test,y_test))
```

```
Training Score 0.9515510725211553
Test Score 0.9454906892105356
```

In [22]:

```
# Plotting the bar graph to depict the difference between the actual and predic
ted value
df.plot(kind='bar',figsize=(5,5))
plt.grid(which='major', linewidth='0.5', color='red')
plt.grid(which='major', linewidth='0.5', color='blue')
plt.show()
```



To find Predicted Score if student studies 9.25 hours a day

In [23]:

```
# Predict percent for custom input value for hours
# Q. What will be predicted score if a student studies for 9.25 hrs/ day?
h= 9.25
test= np.array([h])
test = test.reshape(-1,1)
own_pred = regressor.predict(test)
print("No of Hours = {}".format(h))
print("If the student studies for 9.25 hours/day, the score is {}.".format(own_
pred[0]))
```

```
No of Hours = 9.25
If the student studies for 9.25 hours/day, the score is 93.69173248737538.
```

Evaluating the model

In [24]:

```
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)))
print('R-2:', metrics.r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
Mean Squared Error: 21.5987693072174
Root Mean Squared Error: 4.6474476121003665
R-2: 0.9454906892105356
```

In [ ]: