Author: Agnim Chakraborty

Task 1: Prediction using Supervised ML

GRIP - April' 21 @ The Sparks Foundation

In this task, we are going to apply Linear Regression for predicting student's score percentage based on the number of study hours.

Dataset has been provided for the same

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

Step 1: Importing the required libraries

In [1]:

```
#Importing Libraries
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
%matplotlib inline
```

Step 2: Reading data from the source

In [2]:

```
#Reading the dataset

url = "http://bit.ly/w-data"
student_data = pd.read_csv(url)
print("Data Imported Successfully")
```

```
Data Imported Successfully
```

In [3]:

```
#Display top 5 data
student_data.head()
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [4]:

```
student_data.shape
```

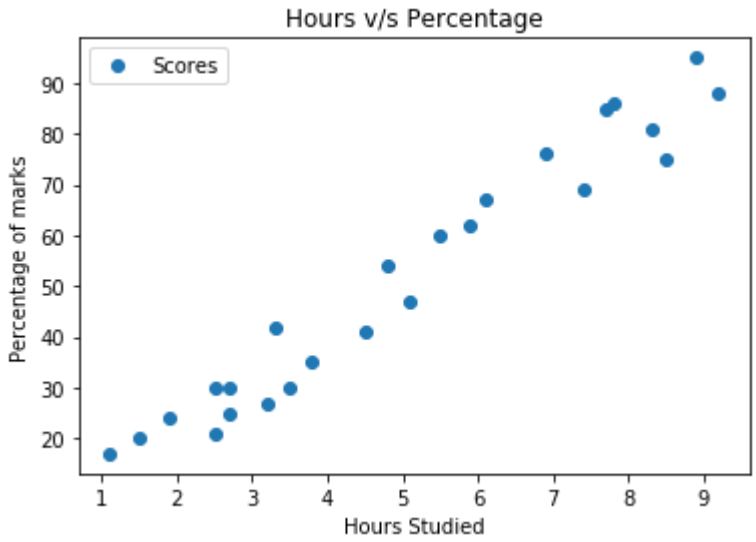
Out[4]:

```
(25, 2)
```

Step 3: Plotting input data for visualization

In [5]:

```
student_data.plot(x= 'Hours',y= 'Scores',style= 'o')
plt.title('Hours v/s Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage of marks')
plt.show()
```



From the graph above we can clearly observe a strong linear relationship between the number of hours studied and percentage of score. And we can imagine a straight line.

Step 4: Data Preprocessing

We divide the dataset into features and labels

In [6]:

```
X = student_data.iloc[:, :-1].values
Y = student_data.iloc[:, 1].values
```

Step 5: Model Training

Let's split the data into training and testing sets, and train the algorithm

In [7]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
lr = LinearRegression()
lr.fit(X_train.reshape(-1,1), Y_train)
print("Model Training Completed.")
```

```
Model Training Completed.
```

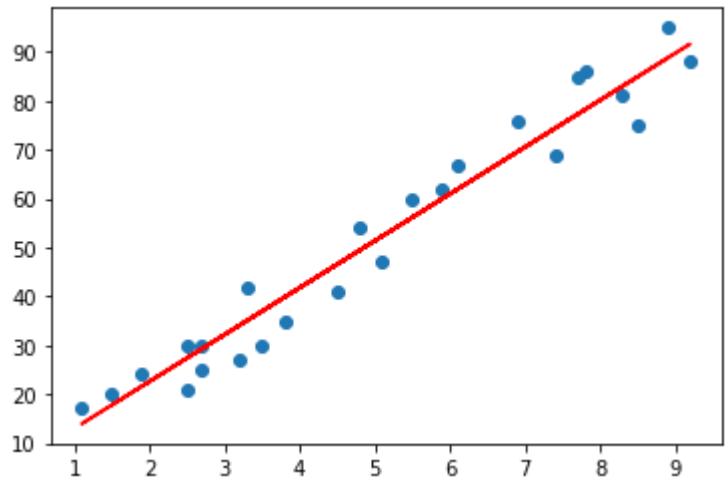
Step 6: Plotting the Line of Regression

Visualizing the line of regression as training is complete

In [8]:

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

# Plotting for the Test Data
plt.scatter(X, Y)
plt.plot(X, line,color='red');
plt.show()
```



Step 7: Making Predictions based on the trained Algorithm

The algorithm is trained, let's test the model by making some predictions.

In [9]:

```
prediction = lr.predict(X_test)
print('Predicted Successfully')
```

```
Predicted Successfully
```

In [10]:

```
prediction
```

Out[10]:

```
array([35.05455971, 78.22421045, 34.09523414, 52.32242001, 56.15972229,
       39.85118757, 29.29860628, 13.94939713])
```

Step 8: Comparing both Actual and Predicted Model results

In [11]:

```
df = pd.DataFrame({'Actual': Y_test, 'Predicted': prediction})
df
```

Out[11]:

	Actual	Predicted
0	42	35.054560
1	86	78.224210
2	27	34.095234
3	47	52.322420
4	60	56.159722
5	35	39.851188
6	25	29.298606
7	17	13.949397

In [12]:

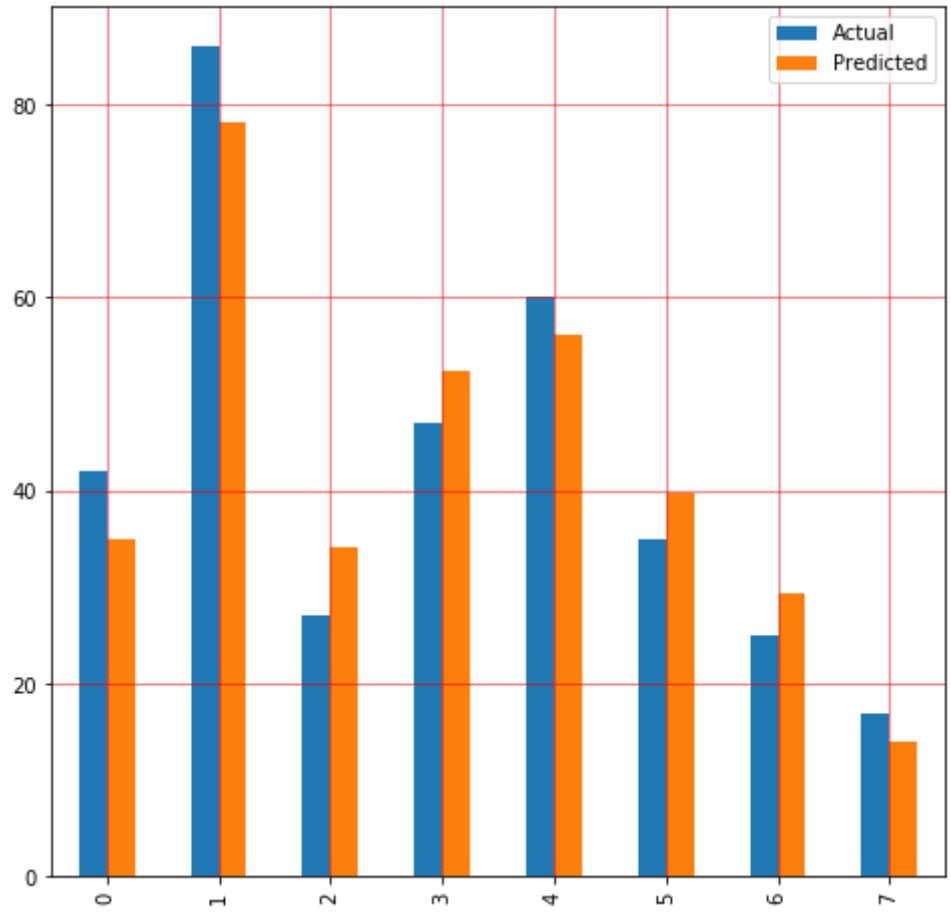
```
#Estimating Training and Testing Scores
print("Training Score:",lr.score(X_train,Y_train))
print("Testing Score:",lr.score(X_test,Y_test))
```

```
Training Score: 0.9565348480964992
Testing Score: 0.9270024140612362
```

In [13]:

```
# Plotting the Bar graph to see the difference between the actual and predicted value

df.plot(kind='bar',figsize=(8,8))
plt.grid(which='major', linewidth='0.5', color='red')
plt.grid(which='minor', linewidth='0.5', color='blue')
plt.show()
```



Step 9: Evaluating the Model

Now, let's evaluate the performance of algorithm. This step is important to compare how well algorithms perform on a particular dataset. Here different errors have been calculated to compare the model performance and predict the accuracy.

In [15]:

```
from sklearn import metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(Y_test, prediction))
print('Mean Squared Error:', metrics.mean_squared_error(Y_test, prediction))
```

```
Mean Absolute Error: 5.397444801586888
Mean Squared Error: 31.679811710143202
```

Step 10: Solution

This is the final step to predict or test the model with our own data

In [16]:

```
hours = 9.25
test = np.array([hours])
test = test.reshape(-1, 1)
predicted_score = lr.predict(test)
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(predicted_score[0]))
```

```
No of Hours = 9.25
Predicted Score = 92.1344312434442
```

Conclusion

I was successfully able to carry-out Prediction using Supervised ML task and was able to evaluate the model's performance. Thus I can conclude that, If a student studies for 9.25 hours/day, he is expected to score 92.134 marks.

Thank You. Have a nice day ahead.

In [ ]:

```

```