Author: Partha Sarathi Samal

Task:1 Prediction using Supervised Machine Learning.

GRIP @ The Sparks Foundation

In this regression model task I tried to predict the percentage of marks that a student is expected to score based upon the number of hours they studied.

This is a simple linear regression task as it involves just two variables.

Technical Stack : Scikit-Learn, Numpy Array,Pandas,Matplotlib

Step-1 Importing all the libraries required in this notebook

In [1]:

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

Step-2 Reading the data from source

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(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

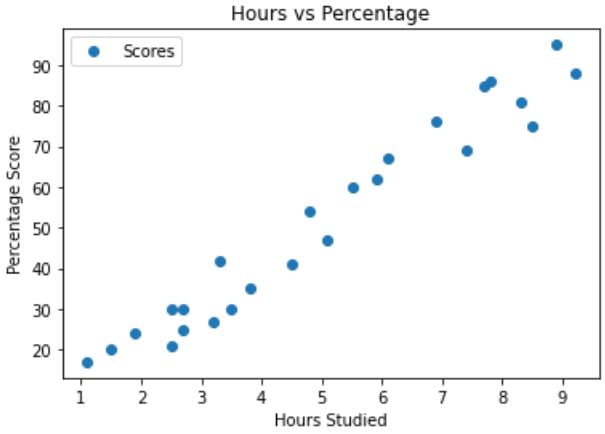
Step-3 Input Data Visualization

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 [3]:

In [3]:

```
# 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.

Step-4 Preparing the data

The next step is to divide the data into "attributes" (inputs) and "labels" (outputs)

In [4]:

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

Now that we have our attributes and labels, the next step is to split this data into training and test sets. We'll do this by using Scikit-Learn's built-in train_test_split() method:

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-5 Training the Algorithm

We have split our data into training and testing sets, and now is finally the time to train our algorithm.

In [6]:

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

print("Training complete.")
```

Training complete.

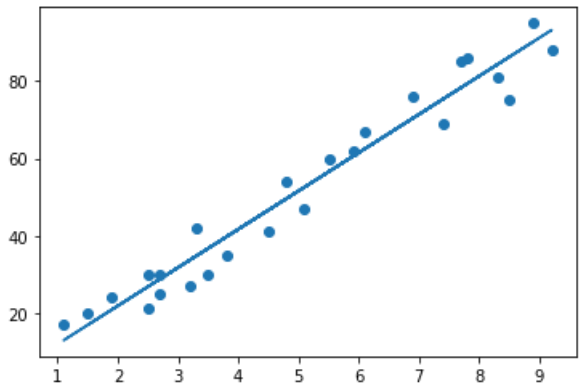
Step-6 Plotting the regression line

In [7]:

```
# Plotting the regression line
```

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

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



Step-7 Making Predictions

Now that we have trained our algorithm, it's time to make some predictions.

In [8]:

```
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]]
```

Step-8 Comparing the actual result with the Predicted model result

In [9]:

```
# Comparing Actual vs 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 [13]:

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

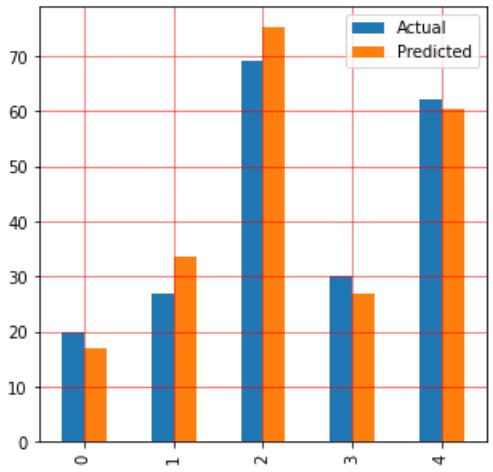
```
Training Score: 0.9515510725211552
Test Score: 0.9454906892105356
```

In [14]:

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

df.plot(kind= 'bar' ,figsize=(5,5))
```

```
plt.grid(which='major', linewidth='0.5' , color='red')
plt.grid(which='minor', linewidth='0.5' , color='blue')
plt.show()
```



In [15]:

```
# Testing the model with our 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-9 Evaluating the model

The final step is to evaluate the performance of algorithm. This step is particularly important to compare how well different algorithms perform on a particular dataset. For simplicity here,different errors have been calculated to compare the model performance and predict the accuracy.

In [17]:

```
from sklearn import metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:',metrics.mean_absolute_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: 4.183859899002975
Root Mean Squared Error: 4.6474476121003665
R-2: 0.9454906892105356

R-2 gives the score of model fit and in this case we have R-2 =0.9454906892105356 which is actually a great score for this model.

Step-10 Conclusion:

I was successfully able to carry out the Prediction using Supervised ML task and was able to eavaluate the model's performance on various parameters.

In [ ]:

```
                *****Thank You*****
```