Author : Yatam Sumalatha

Task 1 : Prediction using Supervised Machine Learning

GRIP @ The Sparks Foundation

In this regression 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.

In [1]:

```
# Importing the required libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Step 1 - Reading the data from source

In [2]:

```
dataframe = pd.read_csv('https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20st
```

In [3]:

```
dataframe
```

Out[3]:

	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 [4]:

```
dataframe.head(5)
```

Out[4]:

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

In [5]:

```
dataframe.tail(5)
```

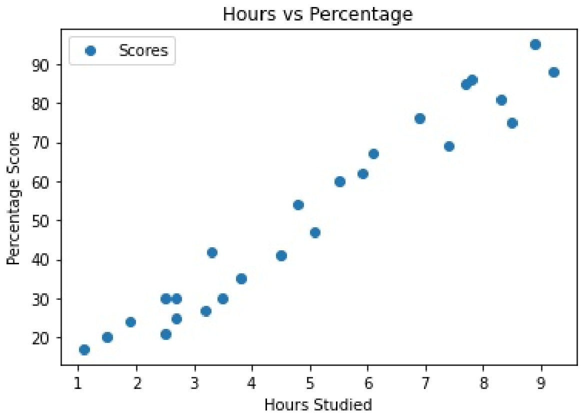
Out[5]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

Step 2 Data Visualization

In [7]:

```
# Plotting the distribution of scores
dataframe.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 we can safely assume a positive linear relation between the number of hours studied and percentage of score.

Step 3 - Data Preprocessing

In [9]:

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

Step 4 - Model Training

Splitting the data into training and testing sets, and training the algorithm.

In [10]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
regressor = LinearRegression()
regressor.fit(X_train.reshape(-1,1), y_train)

print("Training complete.")
```

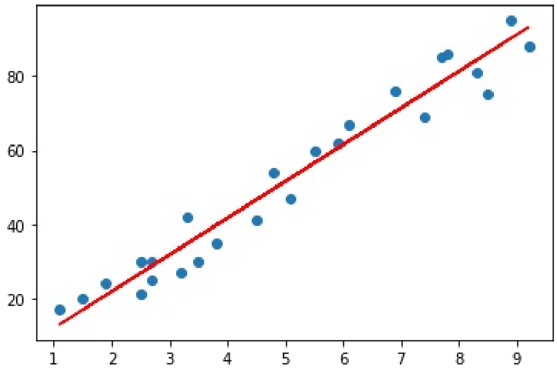
```
Training complete.
```

Step 5 - Plotting the Line of regression

In [11]:

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

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



Step 6 - Making Predictions

In [12]:

```
# Testing data
print(X_test)
# Model Prediction
y_pred = regressor.predict(X_test)
```

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

Step 7 - Comparing Actual result to the Predicted Model result

In [13]:

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

Out[13]:

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

In [14]:

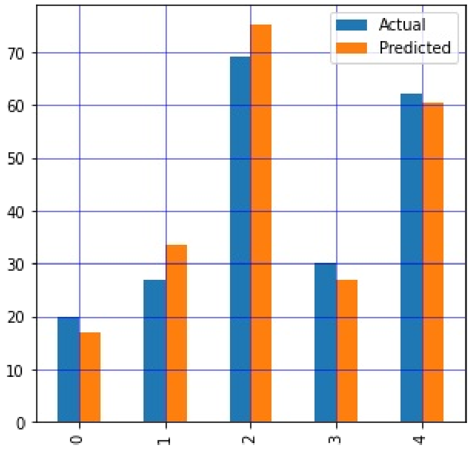
```
print("Training Score:",regressor.score(X_train,y_train))
print("Test Score:",regressor.score(X_test,y_test))
```

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

In [19]:

```
# Plotting the Bar graph

df.plot(kind='bar',figsize=(5,5))
plt.grid(which='major', linewidth='0.5', color='blue')
plt.grid(which='minor', linewidth='0.5', color='red')
plt.show()
```



In [20]:

```
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 8 - Evaluating the model

In [21]:

```
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)))
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
```

Conclusion

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

ThankYou

In [ ]: