GRIP@The Sparks Foundation ¶

Name : Jagadish Rathod

Task 1 : Prediction using Supervised Machine Learning

In this section we will see how the Python Scikit-Learn library for machine learning can be used to implement regression functions. We will start with simple linear regression involving two variables.

```
* This is a simple linear regression task as it involves just 2 variables.
* You can use R, Python, SAS Enterprise Miner or any other tool
* Data can be found at http://bit.ly/w-data
* What will be predicted score if a student studies for 9.25 hrs/ day?
```

Simple Linear Regression

In this regression task we will 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.

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing Data

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

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
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

Checking missing values

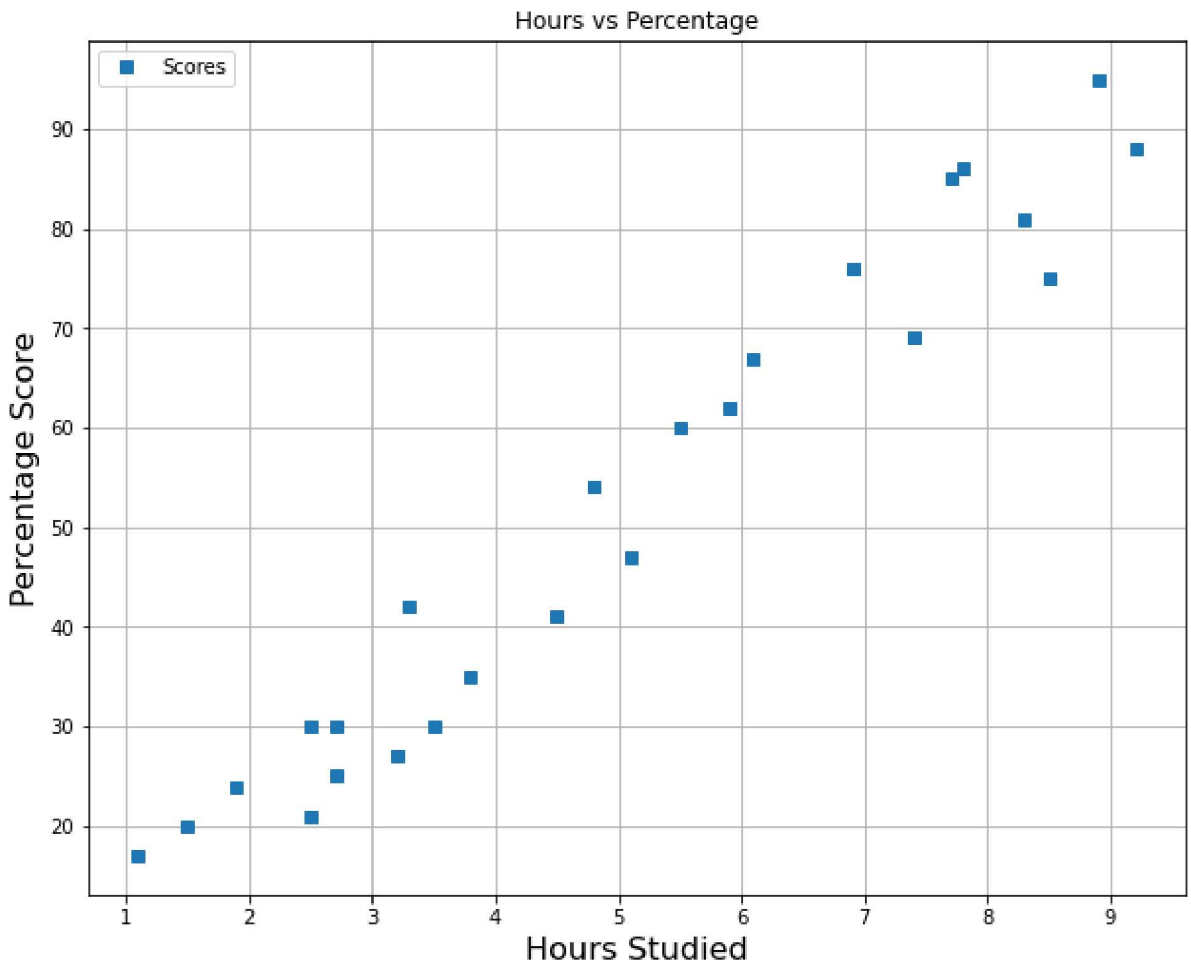
In [3]:
```
s_data.isnull().sum()
```

Out[3]:
```
Hours     0
Scores    0
dtype: int64
```

there is no missing value and duplicate values

Ploting the distribution of scores

In [14]:
```
s_data.plot(x='Hours', y='Scores', style='s', figsize=(10,8))
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied', fontsize=16 )
plt.ylabel('Percentage Score', fontsize=16)
plt.grid(True)
plt.show()
```



Preparing the data

In [15]:
```
X = s_data.iloc[:, :-1].values
y = s_data.iloc[:, 1].values
```

Now that we have our attributes and labels, next step is shuffling and creating train and test dataset using Scikit-Learn.

In [16]:
```
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)
```

Training the model

In [17]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training complete.")
```

```
Training complete.
```

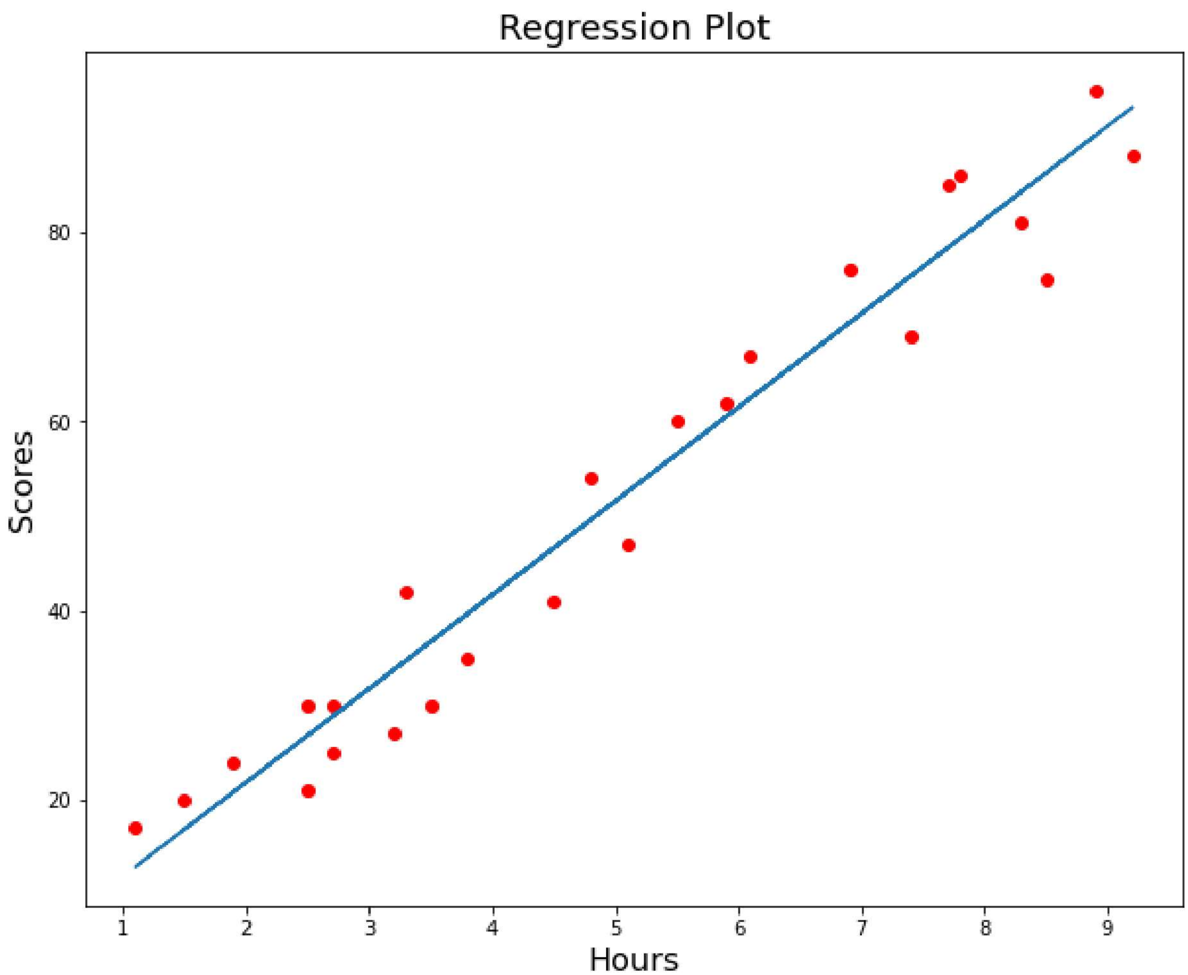
In [18]:
```
print("Coefficient = ", regressor.coef_)
```

```
Coefficient =  [9.91065648]
```

Plotting the Line of Regression

In [19]:
```
# Getting the best fitted line
line = regressor.coef_*X+regressor.intercept_

#plotting the best fitted line on the graph
plt.figure(figsize= (10,8))
plt.title('Regression Plot', size = 18)
plt.scatter(X, y,color="red", label="data points")
plt.xlabel(xlabel='Hours',fontsize=16)
plt.ylabel(ylabel='Scores',fontsize=16)
plt.plot(X, line);
plt.show()
```



Making Predictions

In [20]:
```
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]]
```

Comparing the Predicted with the Actual values

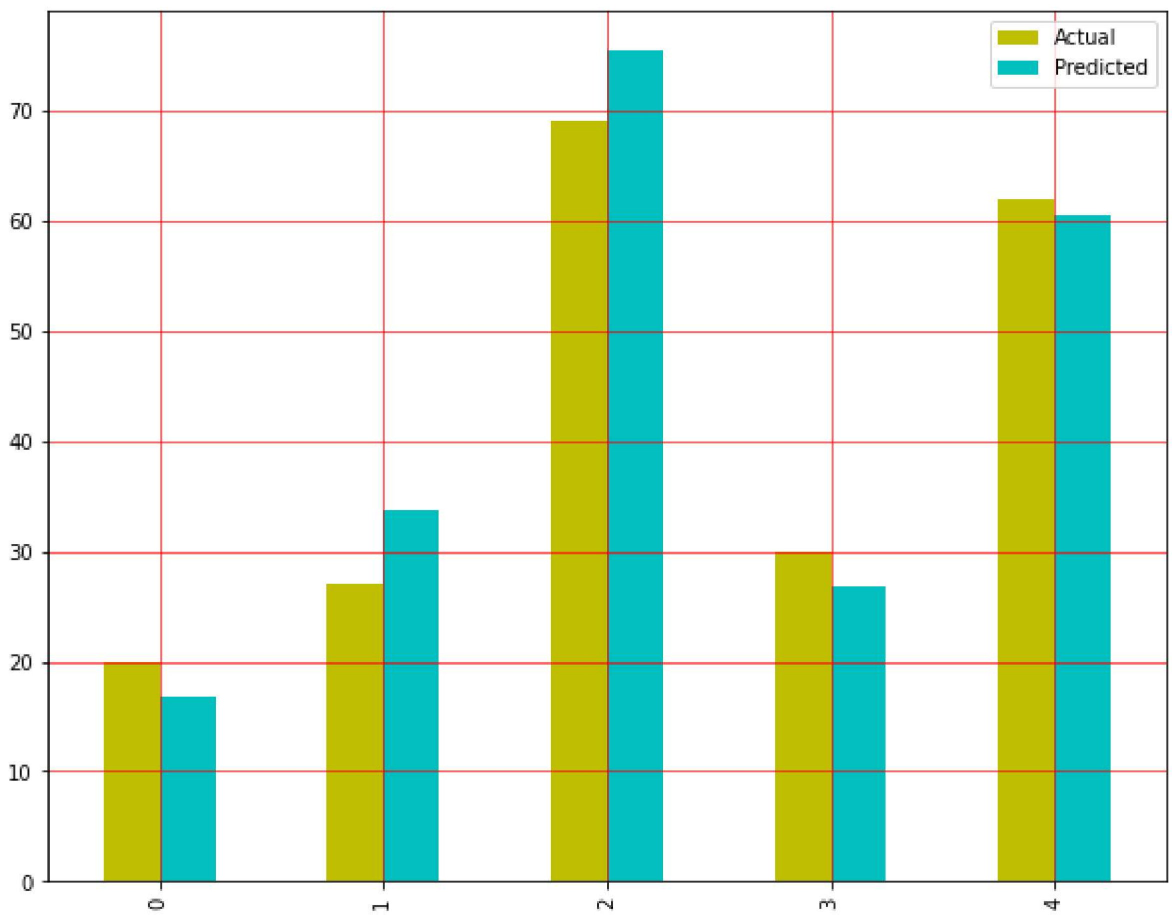
In [21]:
```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[21]:

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

Visually Comparing the Predicted with the Actual values

In [38]:
```
df.plot(kind='bar',figsize=(10,8), color='yc')
plt.grid(which='major', linewidth='0.6', color='red')
plt.grid(which='minor', linewidth='0.6', color='blue')
plt.show()
```



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, we have chosen the mean square error. There are many such metrics.

In [39]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```

Small value of Mean absolute error states that the chances of error or wrong forecasting through the model are very less

Result / Conclusion :

Testing the model to predict the percentage of student if he/she studies for 9.25 hrs/day.

In [40]:
```
hours = 9.25
pred = regressor.predict(np.array(hours).reshape(-1,1))
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(pred[0]))
```

```
No of Hours = 9.25
Predicted Score = 93.69173248737538
```

According to the regression model if a student studies for 9.25 hours a day he/she is likely to score 93.89

After evaluating the performance of model it can be concluded that the model did good prediction by predicting the student percentage as 93.69173248737538 % when student studies for 9.25 hours.

Thank you!