The Sparks Foundation

Graduate Rotational Internship Program (GRIP) July2021 Batch

Data Science & Business Analytics

Task - 1 Predict the percentage of an student based on the no. of study hours

Prediction using Supervised ML

- 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.

- By: Prathamesh Parab

1.Importing and reading the dataset

In [1]:

```
##Importing important libraries
import pandas as pd # For Handling The Dataset
import numpy as np # For Numerical Operation
import seaborn as sns # For Visualization
import matplotlib.pyplot as plt # For Visualization
%matplotlib inline
```

In [2]:

```python
#Importing & reading the Dataset from remote link
path= r"http://bit.ly/w-data"
Data=pd.read_csv(path)
print('Data Imported Successfully')
Data
```

```
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
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 [3]:

```
## Print the records
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]:

```
Data.head(2)
```

Out[4]:

	Hours	Scores
0	2.5	21
1	5.1	47

In [5]:

```
Data.tail()
```

Out[5]:

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

In [6]:

```
Data.tail(1)
```

Out[6]:

	Hours	Scores
24	7.8	86

In [7]:

```
## Display Data From Selected columns
Data[Data['Scores']>70]
```

Out[7]:

	Hours	Scores
3	8.5	75
6	9.2	88
8	8.3	81
10	7.7	85
15	8.9	95
23	6.9	76
24	7.8	86

In [8]:

```
Data[Data['Scores']<70]
```

Out[8]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
4	3.5	30
5	1.5	20
7	5.5	60
9	2.7	25
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
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

In [9]:

```
## use describe() method so that we can able to see percentiles,mean,std,max,count of t
Data.describe()
```

Out[9]:

	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 [10]:

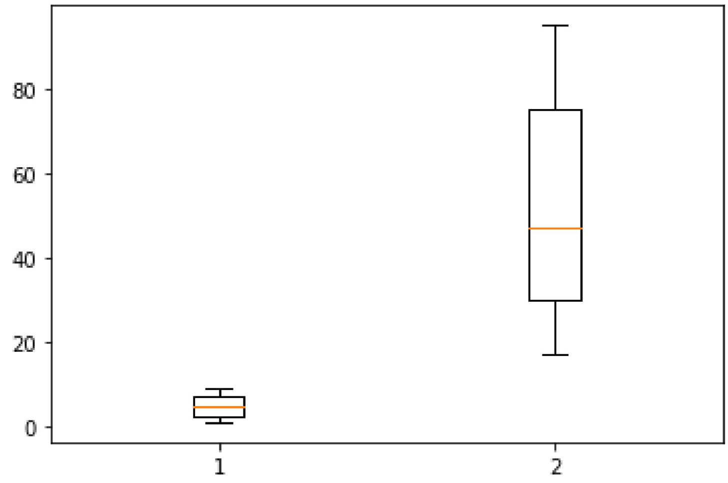
```
## print the full summary of the dataframe .
Data.info()
```

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

2. Visualizing Data

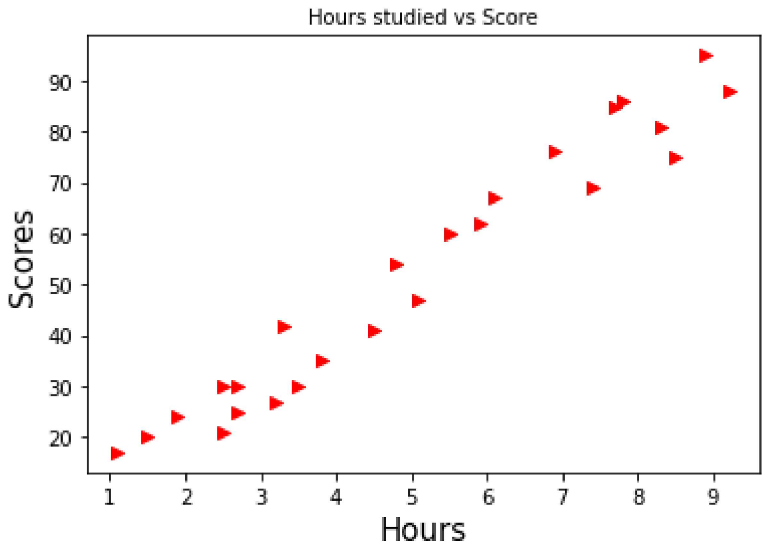
In [11]:

```
##importing libraries for plotting Graphs
import seaborn as sns
plt.boxplot(Data)
plt.show()
```



In [12]:

```
##ploting Scatter plot----
plt.xlabel('Hours',fontsize=15)
plt.ylabel('Scores',fontsize=15)
plt.title('Hours studied vs Score', fontsize=10)
plt.scatter(Data.Hours,Data.Scores,color='r',marker='>')
plt.show()
```



- A scatterplot displays a relationship between two sets of data.
- Notice that the data points are spread out even more in these graphs. The closer the data points lie together to make a line, the higher the correlation.These is Positive Correlation
- In these graphs, there is still a trend in the data, so we would say that the data has a weak or lower correlation.
- As the number of hours of study increase, test scores increase

In [13]:

```
Data.corr()
```

Out[13]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

- Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate).

3. Data Preprocessing

In [27]:

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

Out[27]:

```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [29]:

```
Y
```

Out[29]:

```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95, 30,
       24, 67, 69, 30, 54, 35, 76, 86], dtype=int64)
```

4. Preparing Data and splitting into train and test sets.

In [30]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,random_state = 0,test_size=0.2)
```

In [31]:

```
## We have Splitted Our Data Using 80:20 RULe(PARETO)
print("X train.shape =", X_train.shape)
print("Y train.shape =", Y_train.shape)
print("X test.shape  =", X_test.shape)
print("Y test.shape  =", Y_test.shape)
```

```
X train.shape = (20, 1)
Y train.shape = (20,)
X test.shape  = (5, 1)
Y test.shape  = (5,)
```

5.Training the Model

In [32]:

```
from sklearn.linear_model import LinearRegression
linreg=LinearRegression()
```

In [33]:

```
##Fitting Training Data
linreg.fit(X_train,Y_train)
print("Training algorithm is finished")
```

```
Training algorithm is finished
```

In [34]:

```
print("B0 =",linreg.intercept_,"\nB1 =",linreg.coef_)## β0 is Intercept & Slope of the
```

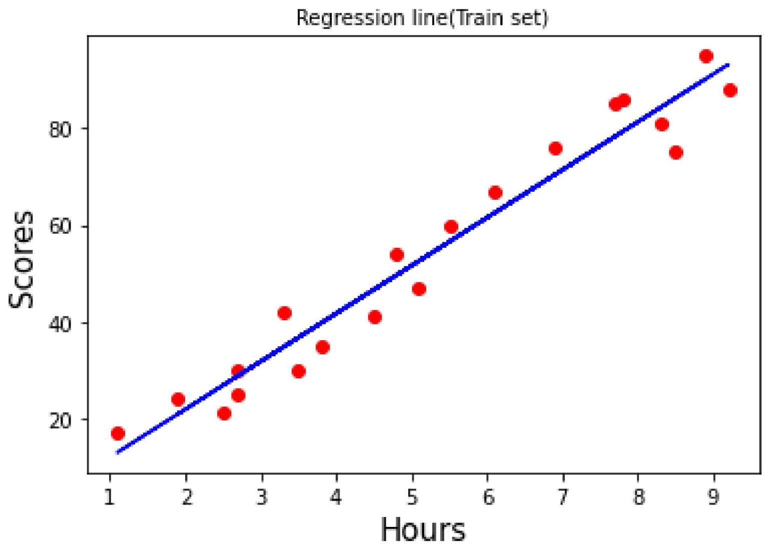
```
B0 = 2.018160041434683
B1 = [9.91065648]
```

In [35]:

```
##plotting the REGRESSION LINE---
Y0 = linreg.intercept_ + linreg.coef_*X_train
```

In [38]:

```
##plotting on train data
plt.scatter(X_train,Y_train,color='r',marker='o')
plt.plot(X_train,Y0,color='b')
plt.xlabel("Hours",fontsize=15)
plt.ylabel("Scores",fontsize=15)
plt.title("Regression line(Train set)",fontsize=10)
plt.show()
```



6. Test the model

In [39]:

```
Y_pred=linreg.predict(X_test)##predicting the Scores for test data
print(Y_pred)
```

```
[16.88414476 33.73226078 75.357018   26.79480124 60.49103328]
```

In [40]:

```
#now print the Y_test.
Y_test
```

Out[40]:

```
array([20, 27, 69, 30, 62], dtype=int64)
```

7. Compare Actual Result vs Predicted Result

In [41]:

```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': Y_test, 'Predicted': Y_pred})
df
```

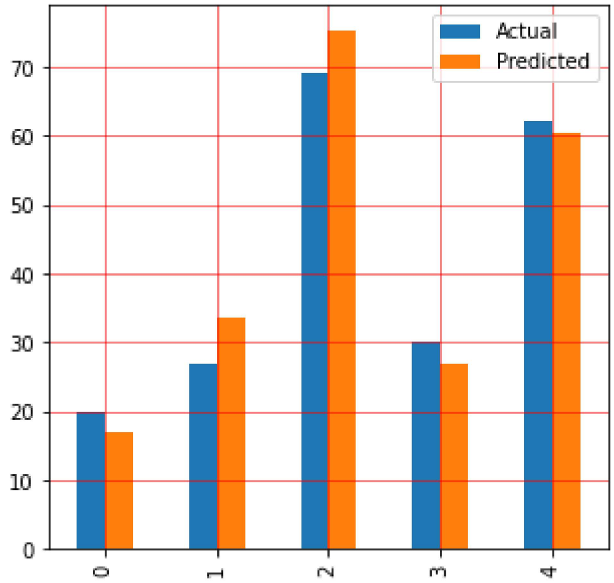
Out[41]:

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

In [47]:

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

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()
```



8. Accuracy Of The Model

In [48]:

```
from sklearn import metrics
metrics.r2_score(Y_test,Y_pred)##Goodness of fit Test
```

Out[48]:

```
0.9454906892105356
```

9. Predict The Error

In [49]:

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [50]:

```
MSE = metrics.mean_squared_error(Y_test,Y_pred)
root_E = np.sqrt(metrics.mean_squared_error(Y_test,Y_pred))
Abs_E = np.sqrt(metrics.mean_squared_error(Y_test,Y_pred))
print("Mean Squared Error      = ",MSE)
print("Root Mean Squared Error = ",root_E)
print("Mean Absolute Error     = ",Abs_E)
```

```
Mean Squared Error      =  21.5987693072174
Root Mean Squared Error =  4.6474476121003665
Mean Absolute Error     =  4.6474476121003665
```

10.Predict The score

In [51]:

```
Prediction_score = linreg.predict([[9.25]])
print("predicted score for a student studying 9.25 hours :",Prediction_score)
```

```
predicted score for a student studying 9.25 hours : [93.69173249]
```

From the above result we can say that if a studied for 9.25 then student will secured 93.69 MARKS

In [ ]:

```

```