Task 1

Prediction using Supervised Machine Learning

Author: Abin Johnson

Submitted to : The Sparks Foundation

Importing and assesing dataset

In [4]:
```
##Importing important libraries---
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [5]:
```
link =  "http://bit.ly/w-data"
set1 = pd.read_csv(link)
print("Data is successfully imported")
set1
```

```
Data is successfully imported
```

Out[5]:

	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

	Hours	Scores
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]:
```
# To get percentiles,mean,std,max,count of the given dataset, let's use describe met

set1.describe()
```

Out[4]:

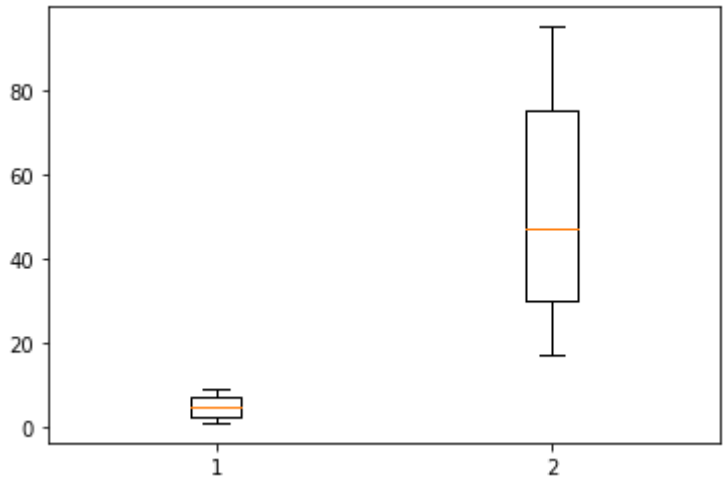
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [5]:
```
set1.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
```

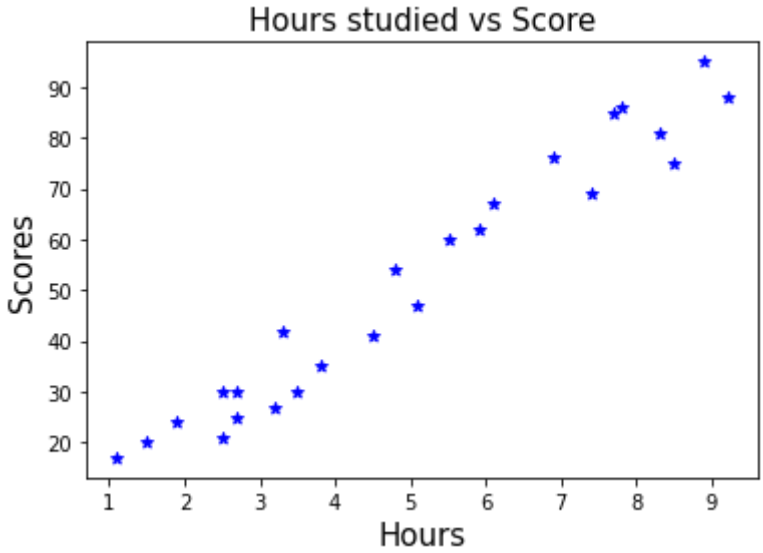
Visualization of the data

In [6]:
```
import seaborn as sns
plt.boxplot(set1)
plt.show()
```



In [7]:

```
## Analyzing the data with Scatter plot
plt.xlabel('Hours',fontsize=15)
plt.ylabel('Scores',fontsize=15)
plt.title('Hours studied vs Score', fontsize=15)
plt.scatter(set1.Hours,set1.Scores,color='blue',marker='*')
plt.show()
```



Analysis of Scatterplot: As we can see in this Scatterplot, Scores and Hours are POSITIVELY RELATED. This implies that if a student studies more hours, more marks will be attained by the students.

In [10]:

```
# We can use iloc function to retrieve a particular value belonging to a row and col
# So let's see each coordinates.

X = set1.iloc[:,:-1].values
X
Y = set1.iloc[:,1].values
Y
print("X coordinates are:", X,"Y Coordinates are:",Y)
```

```
X coordinates are: [[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]] Y Coordinates are: [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]
```

Training, Testing and Splitting of the Dataset

In [11]:

```
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
# We splitted our data using 80-20 rule.(test_size = 0.2)

print("X trained data shape = ",X_train.shape)
print("X test data shape =",X_test.shape)
print("Y train data shape = ",Y_train.shape)
print("Y test data shape = ",Y_test.shape)
```

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

Linear Regression of Training Data Set

In [12]:

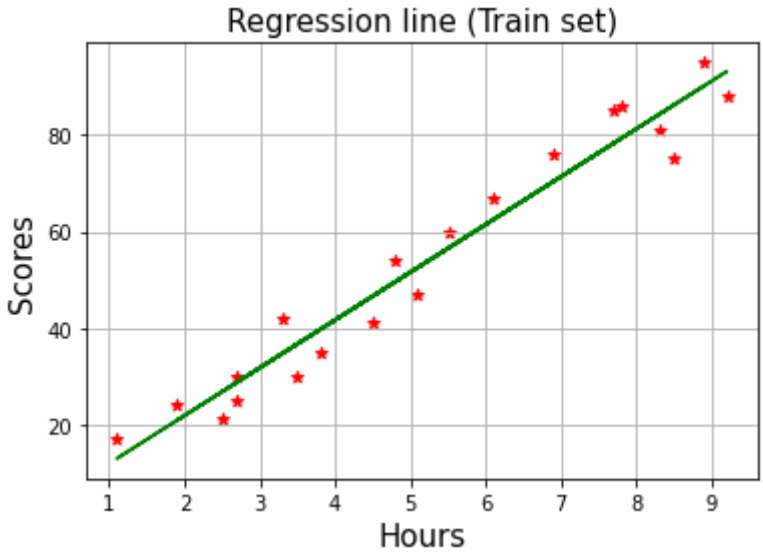
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()

# Let's train our Algorithm
lr.fit(X_train,Y_train)
print("B0 =",lr.intercept_,"\nB1 =",lr.coef_)
## B0 is Intercept & Slope of the line is B1.,"
```

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

In [14]:

```
# Plotting the regression line
Y0 = lr.intercept_ + lr.coef_*X_train
plt.scatter(X_train,Y_train,color='red',marker='*')
plt.plot(X_train,Y0,color='green')
plt.xlabel("Hours",fontsize=15)
plt.ylabel("Scores",fontsize=15)
plt.title("Regression line (Train set)",fontsize=15)
plt.grid()
plt.show()
```



Linear Regression Analysis of Test Data

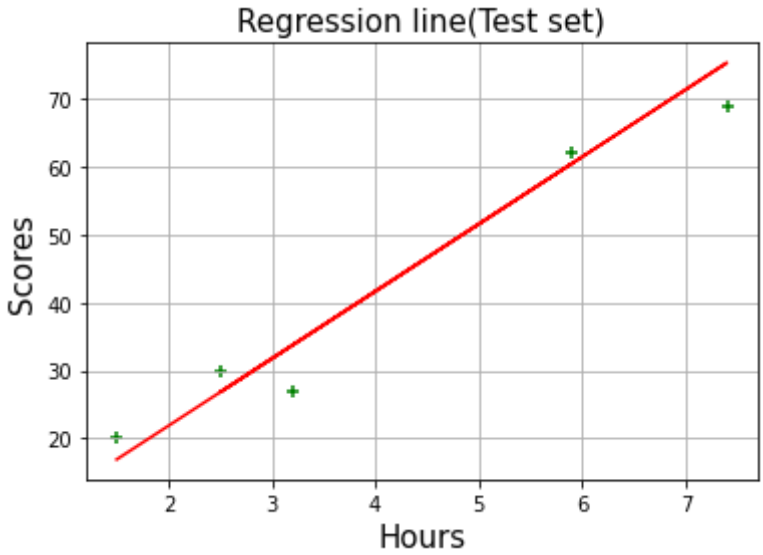
In [15]:
```
Y_pred=lr.predict(X_test)##predicting the Scores for test data
print(Y_pred)
```

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

In [16]:
```
Y_test
```

Out[16]:
```
array([20, 27, 69, 30, 62], dtype=int64)
```

In [17]:
```
#plotting line on test data
plt.plot(X_test,Y_pred,color='red')
plt.scatter(X_test,Y_test,color='green',marker='+')
plt.xlabel("Hours",fontsize=15)
plt.ylabel("Scores",fontsize=15)
plt.title("Regression line(Test set)",fontsize=15)
plt.grid()
plt.show()
```



Comparison of Actual and Predicted Values

In [18]:
```
```

```
Y_test1 = list(Y_test)
prediction=list(Y_pred)
df_compare = pd.DataFrame({ 'Actual':Y_test1,'Result':prediction})
df_compare
```

Out[18]:

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

Finding the Accuracy (Goodness of Fit)

In [19]:

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

Out[19]: 0.9454906892105356

This shows that our model is 94% accurate, that is its a best fitted model

Predicting the Error

In [20]:

```
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
```

Predicting the Scores

In [21]:

```
Prediction_score = lr.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]
```

Inference

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

As shown above, we can conclude that if a student studies for 9.25hrs a day, he may secure approximately 93.69% marks.