Author :- Sambit kumar Nayak

GRIP @ Spark Foundation

TASK 1 :- PREDICTION USING SUPERVISED ML

-Predict the percentage of a student based on the number of study hour.

Dataset :- http://bit.ly/w-data (http://bit.ly/w-data)

import libraries

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

Loading datasets

In [2]:
```
data = pd.read_csv("http://bit.ly/w-data")
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

	Hours	Scores
23	6.9	76
24	7.8	86

In [3]:
```
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 [7]:
```
data.tail()
```

Out[7]:

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

View some statistical details

In [8]:
```
data.describe()
```

Out[8]:

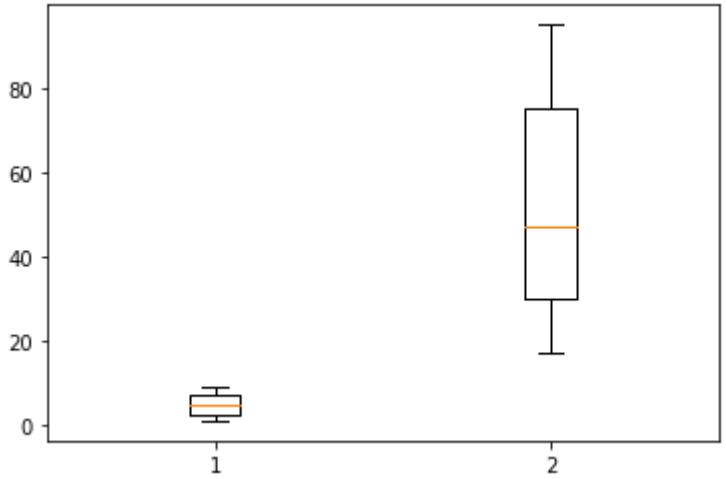
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [9]:
```
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
```

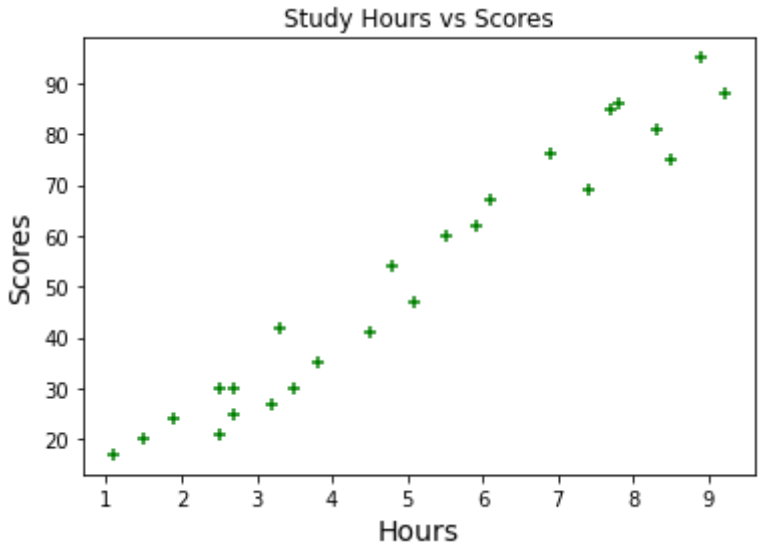
Visualize how given dataset is distributed

In [11]:
```
plt.boxplot(data)
plt.show()
```



Visualize how one variable is affected by another/ relationship between them

In [13]:
```
plt.xlabel("Hours",fontsize = 14)
plt.ylabel("Scores",fontsize = 14)
plt.title("Study Hours vs Scores")
plt.scatter(data.Hours,data.Scores,color = 'green',marker = '+')
plt.show()
```



In [15]:
```
x = data.iloc[:,:-1].values
y = data.iloc[:,1].values
x,y
```

Out[15]:
```
(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]]),
 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))
```

Training the dataset

In [18]:
```
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 [19]:
```
print("shape of x-train : ",x_train.shape)
print("shape of y_train : ", y_train.shape)
print("Test of x_train : ", x_test.shape)
print("Test of y_train : ", y_test.shape)
```

```
shape of x-train :  (20, 1)
shape of y_train :  (20,)
Test of x_train :  (5, 1)
Test of y_train :  (5,)
```

In [21]:
```
from sklearn.linear_model import LinearRegression
linearRegression = LinearRegression()
```

In [23]:
```
linearRegression.fit(x_train,y_train)
```

Out[23]:
```
LinearRegression()
```

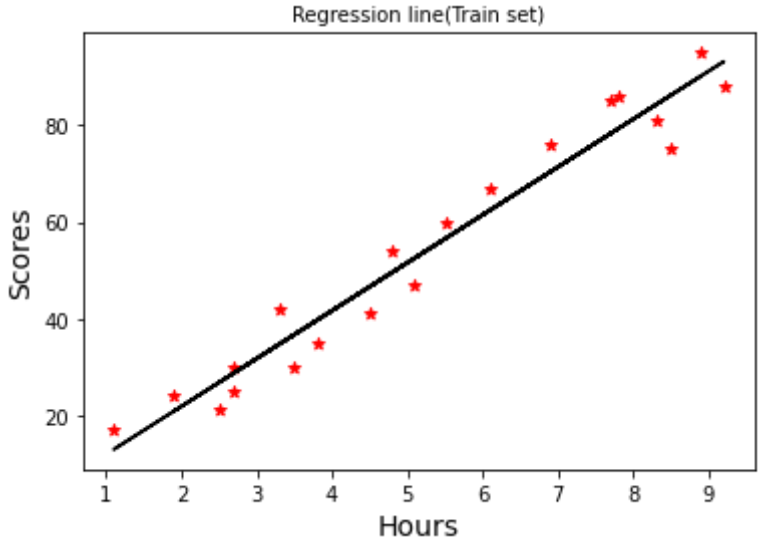
In [24]:
```
print("B0 =",linearRegression.intercept_,"\nB1 =",linearRegression.coef_)
```

```
B0 = 2.018160041434662
B1 = [9.91065648]
```

In [26]:
```
Y0 = linearRegression.intercept_ + linearRegression.coef_ * x_train
```

plotting the Regression Line

In [29]:
```
plt.scatter(x_train,y_train,color='red',marker='*')
plt.plot(x_train,Y0,color='black')
plt.xlabel("Hours",fontsize=14)
plt.ylabel("Scores",fontsize=14)
plt.title("Regression line(Train set)",fontsize=10)
plt.show()
```



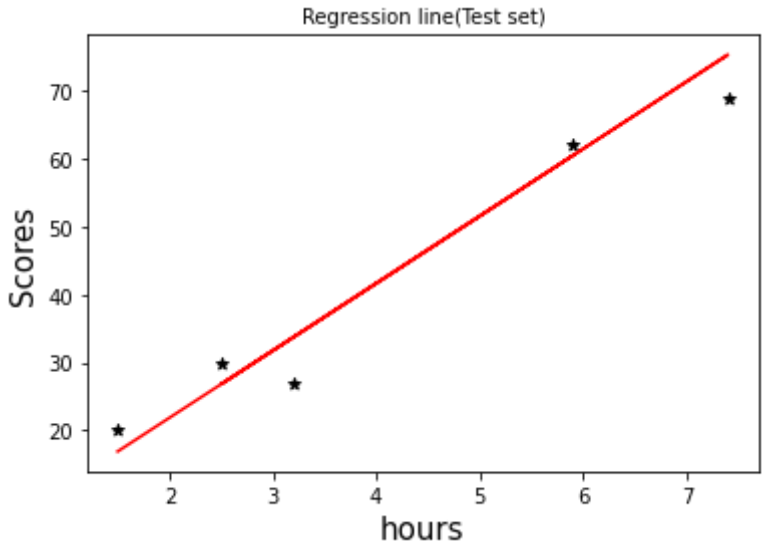
In [30]:
```
y_pred = linearRegression.predict(x_test)
print(y_pred)
```

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

In [31]:
```
y_test
```

Out[31]:
```
array([20, 27, 69, 30, 62], dtype=int64)
```

In [35]:
```
plt.plot(x_test,y_pred,color = 'red')
plt.scatter(x_test,y_test,color = 'black',marker='*')
plt.xlabel("hours",fontsize=15)
plt.ylabel("Scores",fontsize=15)
plt.title("Regression line(Test set)",fontsize=10)
plt.show()
```



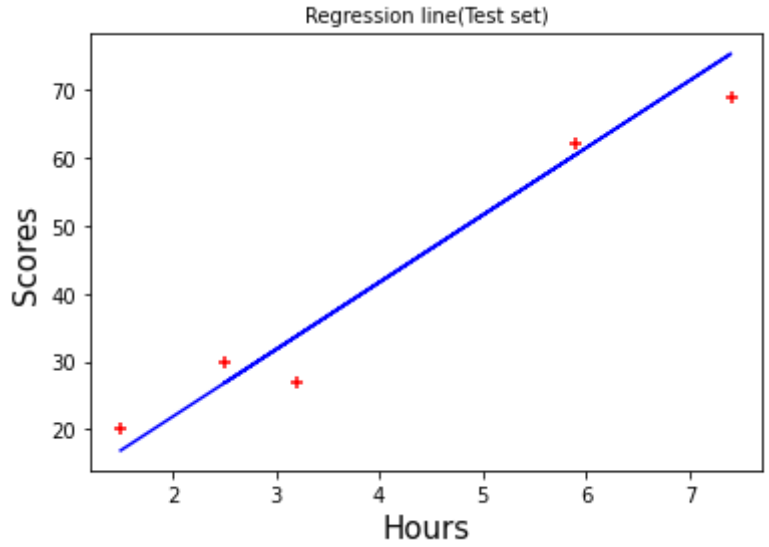
In [36]:
```
y_pred = linearRegression.predict(x_test)
print(y_pred)
```

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

In [37]:
```
y_test
```

Out[37]:
```
array([20, 27, 69, 30, 62], dtype=int64)
```

In [39]:
```
plt.plot(x_test,y_pred,color='blue')
plt.scatter(x_test,y_test,color = 'red',marker = '+')
plt.xlabel("Hours",fontsize=15)
plt.ylabel("Scores",fontsize=15)
plt.title("Regression line(Test set)",fontsize=10)
plt.show()
```



In [44]:
```
y_test1 = list(y_test)
prediction = list(y_pred)
df_compare = pd.DataFrame({'Actual':y_test1,'Result':prediction})
df_compare
```

Out[44]:

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

Calculating the matrices

In [48]:
```
from sklearn import metrics
metrics.r2_score(y_test,y_pred)
```

Out[48]:
```
0.9454906892105354
```

In [53]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
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    =  21.598769307217456
Root Mean Squared Error  = 4.647447612100373
Mean Absolute Error     = 4.647447612100373
```

Answer

In [56]:
```
prediction_score = linearRegression.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]
```

In [ ]:
```

```