LINEAR REGRESSION USING SUPERVISED LEARNING

Stastics of linear regression

In [5]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import  pandas as pd
plt.rcParams['figure.figsize']=(7.0,6.0)

#Reading Data

data=pd.read_csv(r'C:\Users\pragathi s p\OneDrive\Documents\Book1.csv')
print(data.shape)

#viewing data
data.head()
```

```
(25, 2)
```

Out[5]:

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

In [6]:
```
X=data['Hours'].values      # X is independent variable
Y=data['Scores'].values    # Y is dependent variable
```

In [39]:
```
print(data)              #this is how the data looks
```

```
    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 [7]:
```
#Mean of X and Y
mean_x=np.mean(X)
mean_y=np.mean(Y)
n=len(X)

numer=0
denom=0
for i in range(n):
    numer=(X[i]-mean_x)*(Y[i]-mean_y)
    denom=(X[i]-mean_x)**2                    # Y=mX+C
m=numer/denom                                 # m is the slope of the regression
c=mean_y-(m*mean_x)                           # c is constant
print(m,c)
```

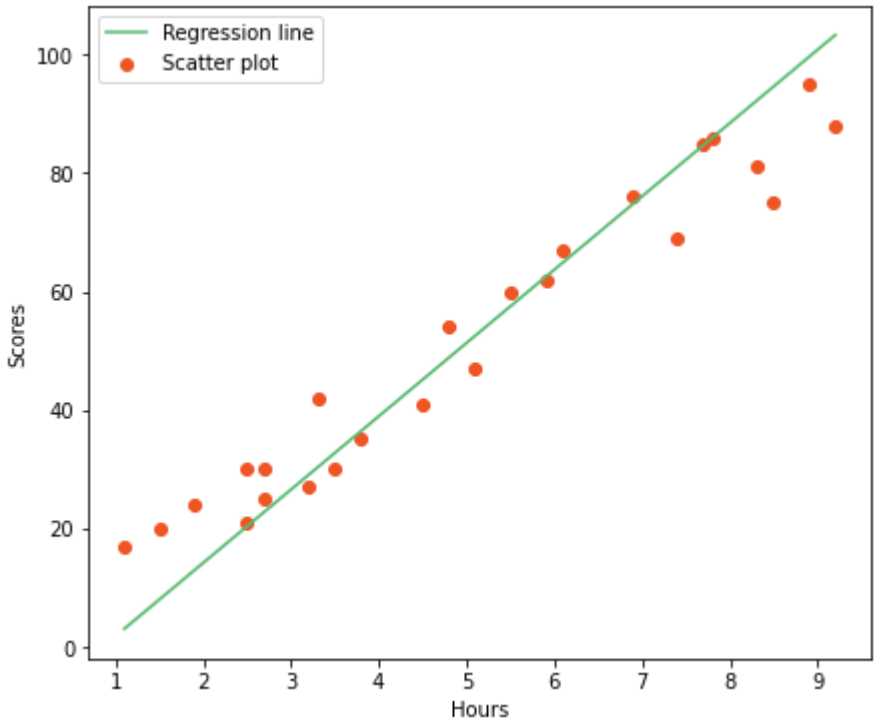
```
12.381635581061692 -10.5767575322812
```

In [8]:
```
max_x=np.max(X)
min_x=np.min(X)

x=np.linspace(min_x, max_x,5)
y= c + (m*x)

plt.plot(x, y, color='#58b970', label='Regression line')
plt.scatter(X,Y, color='#ef5423', label="Scatter plot")

plt.xlabel('Hours')
plt.ylabel('Scores')
plt.legend()
plt.show()
```



In [9]:
```
ss_t=0
ss_r=0
for i in range(n):                                  #calculating R**2
    y_pred=(m*X[i])+c
    ss_t+=(Y[i]-mean_y)**2
    ss_r+=(Y[i]-y_pred)**2
r2=1-(ss_r/ss_t)
print(r2)
```

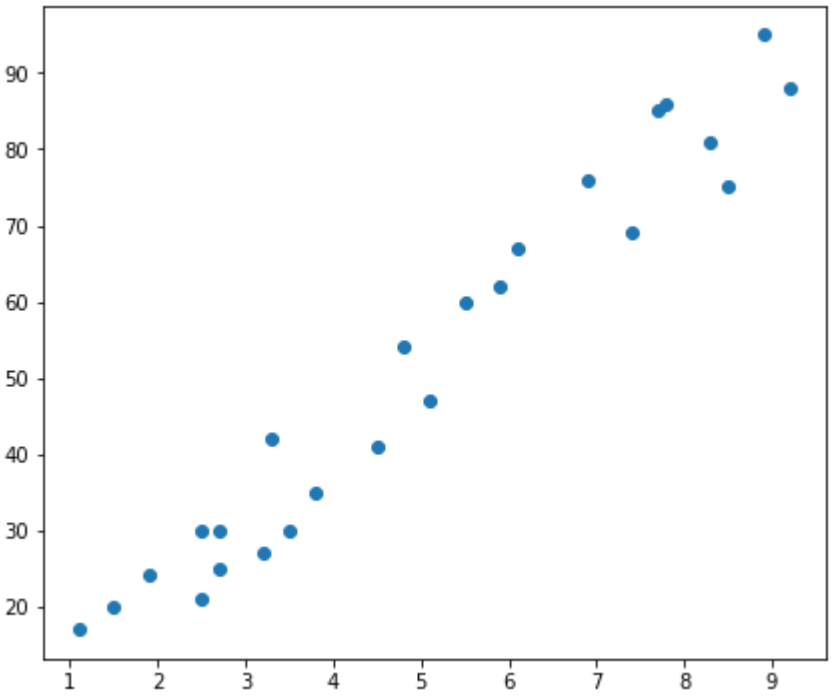
```
0.8852375029281304
```

Prediction using Supervised Machine-learning

Preparation of data

In [10]:
```
plt.scatter(data['Hours'], data['Scores'])
```

Out[10]:
```
<matplotlib.collections.PathCollection at 0x1fa18cf7610>
```



In [11]:
```
X=data.iloc[:,:-1].values  #reshaping
Y=data.iloc[:,1].values
```

In [12]:
```
print(X.ndim)
print(Y.ndim)
print(X.shape)
print(Y.shape)
```

```
2
1
(25, 1)
(25,)
```

In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test=train_test_split(X, Y, test_size=0.34)
```

In [14]:
```
X_train
```

Out[14]:
```
array([[1.1],
       [9.2],
       [3.8],
       [4.8],
       [4.5],
       [1.5],
       [2.7],
       [2.5],
       [5.9],
       [3.3],
       [1.9],
       [7.8],
       [5.5],
       [7.7],
       [3.5],
       [6.1]])
```

In [15]:
```
X_test
```

Out[15]:
```
array([[5.1],
       [3.2],
       [2.7],
       [8.5],
       [8.9],
       [2.5],
       [7.4],
       [6.9],
       [8.3]])
```

In [16]:
```
Y_train
```

Out[16]:
```
array([17, 88, 35, 54, 41, 20, 30, 21, 62, 42, 24, 86, 60, 85, 30, 67],
      dtype=int64)
```

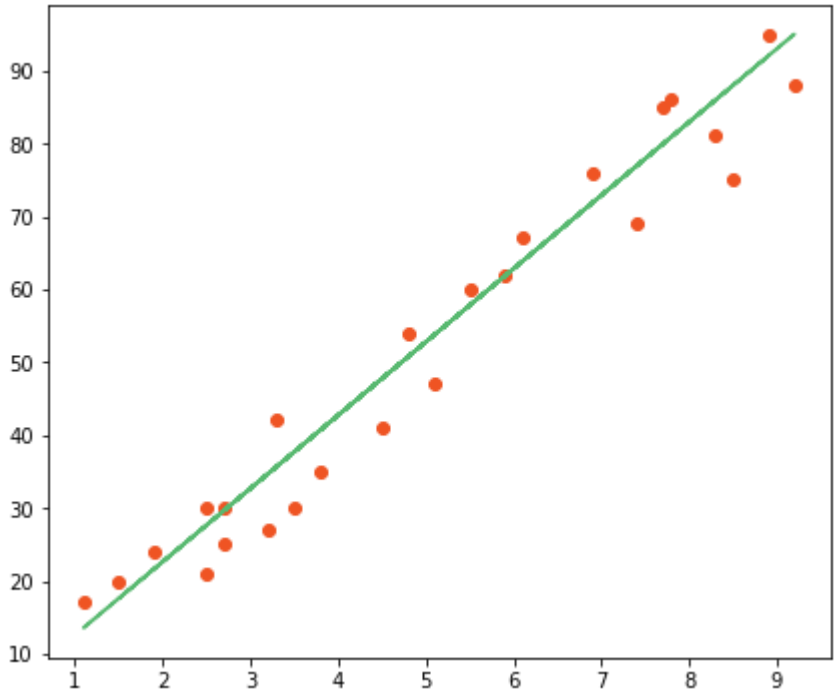
In [17]:
```
Y_test
```

Out[17]:
```
array([47, 27, 25, 75, 95, 30, 69, 76, 81], dtype=int64)
```

In [18]:
```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,Y_train)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```
Yo=regressor.coef_*X+regressor.intercept_
plt.plot(X, Yo, color='#58b970')
plt.scatter(X,Y, color='#ef5423')
plt.show()
```



In [20]:
```
Y_pred=regressor.predict(X_test)
print(Y_pred)
```

```
[53.7844151  34.67765806 29.64956411 87.97545399 91.99792916 27.63832653
 76.91364729 71.88555334 85.96421641]
```

In [21]:
```
Comparision = pd.DataFrame({'Actual': Y_test, 'Predicted': Y_pred})
print(Comparision)
```

```
   Actual  Predicted
0      47  53.784415
1      27  34.677658
2      25  29.649564
3      75  87.975454
4      95  91.997929
5      30  27.638327
6      69  76.913647
7      76  71.885553
8      81  85.964216
```

Predicted score if a student studies for 9.25 hours a day

In [22]:
```
Hours=9.25
pred_result = regressor.predict(np.array(Hours).reshape(-1,1))
print("Number of Hours = {}".format(Hours))
print("Predicted Score = {}".format(pred_result[0]))
```

```
Number of Hours = 9.25
Predicted Score = 95.5175949257103
```

Evaluating the model

In [37]:
```
from sklearn import metrics
y=metrics.mean_absolute_error(Y_test, Y_pred)
print("Mean absolute error= {}".format(y))
```

```
Mean absolute error= 4.3824753409718875
```