The Sparks Foundation

Ayush Singh

Function : Data Science and Business Analytics

GRIP Task - 1 : Prediction Using Supervised Machine Learning.

Aim : (a)Predict the percentage of an student based on the no. of study hours.

(b)What will be predicted score if a student studies for 9.25 hrs/ day?

In [3]:

```
#Importing Libraries
import pandas as pd  # For Data Analysis
import numpy as np  # For Data Modification
import matplotlib.pyplot as plt # For Data Visualizatio
import seaborn as sns # For Data Visualization
```

In [4]:

```
# Reading The Data
url='https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20
df=pd.read_csv(url)
df.head()
```

Out[4]:

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

In [5]:

```
df.shape
```

Out[5]:

```
(25, 2)
```

In [6]:

```
df.describe()
```

Out[6]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000

	Hours	Scores
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [7]:

```
df.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
```

In [8]:

```
df.isnull().sum() # Checking for null value
```

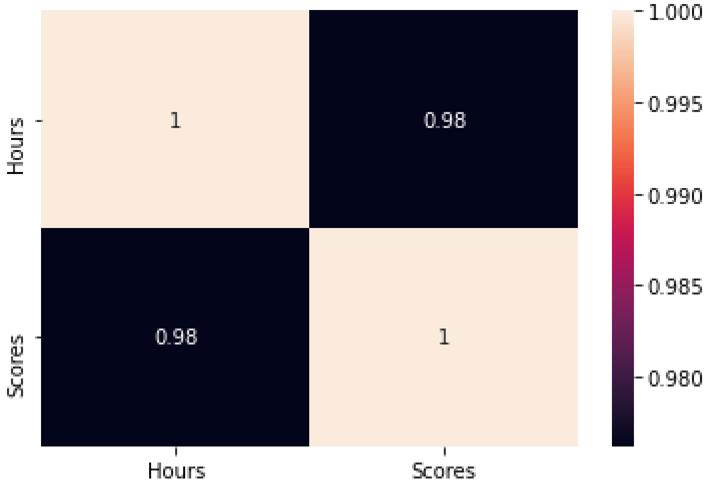
Out[8]:

```
Hours     0
Scores    0
dtype: int64
```

In [9]:

```
# Correlation
cr=df.corr()
print(cr)
print(sns.heatmap(cr,annot=True))
```

```
           Hours    Scores
Hours   1.000000  0.976191
Scores  0.976191  1.000000
AxesSubplot(0.125,0.125;0.62x0.755)
```



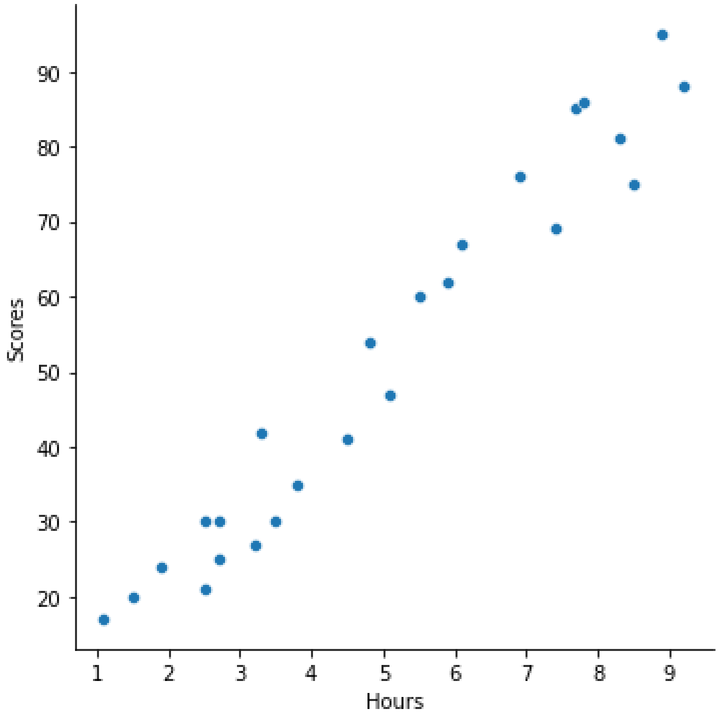
From this we can say that there is very high +ve correlation between Hours and Scores

In [10]:

```
# Vizualization of Dataset
sns.relplot(x='Hours',y='Scores',data=df)
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x1a27fa816a0>
```



from this plot we can say that there is very +ve Linear Relation between Hours studies and Score obtained and therefore the Linear regression model will be the best model for this case.

In [33]:

```
# Data Prepration
x=df[['Hours']]
y=df['Scores']
print(x,y)
```

```
    Hours
0     2.5
1     5.1
2     3.2
3     8.5
4     3.5
5     1.5
6     9.2
7     5.5
8     8.3
9     2.7
10    7.7
11    5.9
12    4.5
13    3.3
14    1.1
15    8.9
16    2.5
17    1.9
18    6.1
19    7.4
20    2.7
21    4.8
22    3.8
23    6.9
24    7.8 0     21
1     47
2     27
3     75
4     30
5     20
```

```
6     88
7     60
8     81
9     25
10    85
11    62
12    41
13    42
14    17
15    95
16    30
17    24
18    67
19    69
20    30
21    54
22    35
23    76
24    86
Name: Scores, dtype: int64
```

In [34]:

```
#Splitting into Train and Test dataset
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/5,random_state=0)
print(x_train,y_train)
```

```
    Hours
22    3.8
17    1.9
24    7.8
23    6.9
14    1.1
1     5.1
10    7.7
13    3.3
8     8.3
6     9.2
18    6.1
4     3.5
9     2.7
7     5.5
20    2.7
3     8.5
0     2.5
21    4.8
15    8.9
12    4.5 22    35
17    24
24    86
23    76
14    17
1     47
10    85
13    42
8     81
6     88
18    67
4     30
9     25
7     60
20    30
3     75
0     21
21    54
15    95
```

```
12    41
Name: Scores, dtype: int64
```

In [35]:

```
#  Linear Regression Model
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

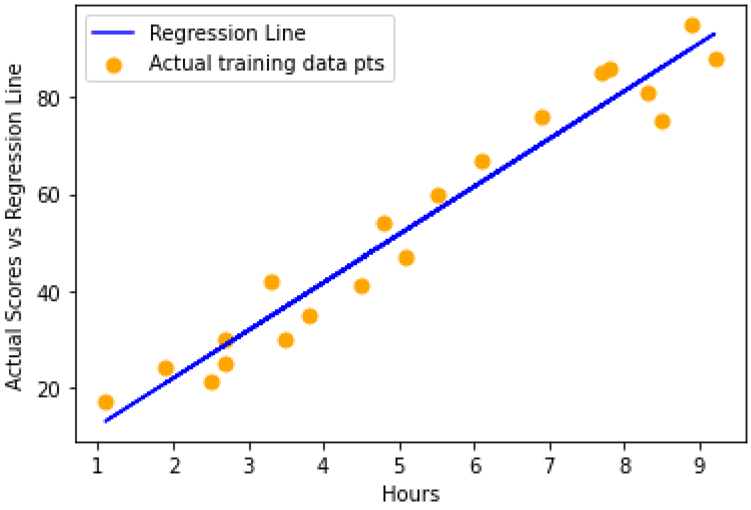
In [36]:

```
# Fitting the Traing Data Set into Linear Regression Model.
lr.fit(x_train,y_train)
print("Training Complete")
```

```
Training Complete
```

In [38]:

```
#Visualizing the Training result
plt.scatter(x_train,y_train,color='orange',marker='o',s=50,label='Actual training data
plt.plot(x_train,lr.predict(x_train),color='blue',label='Regression Line')
plt.legend()
plt.xlabel('Hours')
plt.ylabel('Actual Scores vs Regression Line')
plt.show()
```



In [18]:

```
#Slope(m)
m=lr.coef_
m
```

Out[18]:

```
array([9.91065648])
```

In [19]:

```
#Intercept(c)
c=lr.intercept_
c
```

Out[19]:

```
2.018160041434683
```

In [20]:

```
print("Equation of Regression Line : y=mx+c")
```

```
Equation of Regression Line : y=mx+c
```

In [21]:

```
#Prediction with Test Dataset
```

In [22]:

```
y_pred=lr.predict(x_test)
```

In [23]:

```
# Comparision of Predicted score and Actual score
```

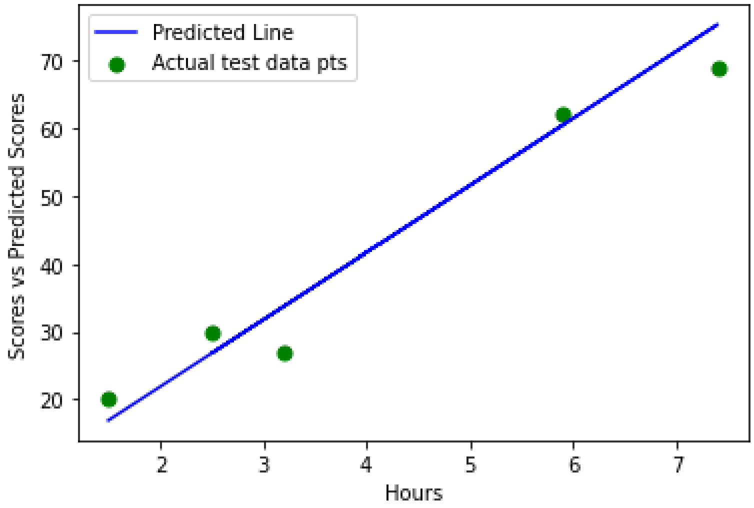
```
dt=pd.DataFrame(np.c_[x_test,y_test,y_pred],columns=['Hours','Scores','Pre-Scores'])
dt
```

Out[23]:

	Hours	Scores	Pre-Scores
0	1.5	20.0	16.884145
1	3.2	27.0	33.732261
2	7.4	69.0	75.357018
3	2.5	30.0	26.794801
4	5.9	62.0	60.491033

In [39]:

```
#visualizing the predicted result
plt.scatter(dt['Hours'],dt['Scores'],marker='o',color='green',s=50,label='Actual test d
plt.plot(dt['Hours'],dt['Pre-Scores'],color='blue',label='Predicted Line')
plt.legend()
plt.xlabel('Hours')
plt.ylabel('Scores vs Predicted Scores')
plt.show()
```



In [25]:

```
# Prediction of score obtained for 9.25 hours of Studies.
```

In [26]:

```
lr.predict([[9.25]])
```

Out[26]:

```
array([93.69173249])
```

In [41]:

```
# We can also find it by using eq of regression line i.e., y=mx+c
y=m*(9.25)+c
print("The marks obtained for 9.25 hrs of studies is :",y)
```

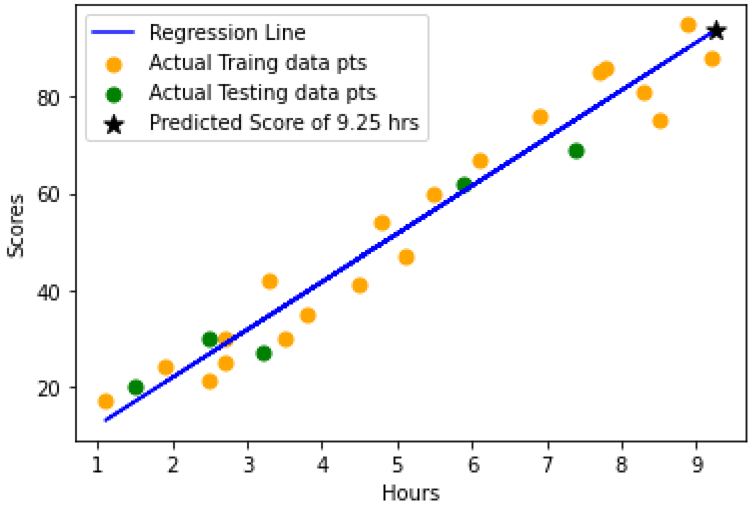
```
The marks obtained for 9.25 hrs of studies is : [93.69173249]
```

In [42]:

```
# Complete Visualization
```

In [43]:

```
plt.scatter(x_train,y_train,color='orange',marker='o',s=50,label='Actual Traing data pt
plt.scatter(dt['Hours'],dt['Scores'],marker='o',color='green',s=50,label='Actual Testin
plt.plot(x_train,lr.predict(x_train),color='blue',label='Regression Line')
plt.scatter(9.25,y,color='black',marker='*',s=100,label='Predicted Score of 9.25 hrs')
plt.legend()
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.show()
```



In [30]:

```
# Modell Evaluation.
```

In [31]:

```
# Calculating the R-Squired Value.
from sklearn.metrics import r2_score
print("R-Squired value is :",r2_score(y_test,y_pred))
```

```
R-Squired value is : 0.9454906892105356
```

In [32]:

```
# Calculating the Mean Absolute Error
from sklearn import metrics
print("Mean absolute error is :",metrics.mean_absolute_error(y_test,y_pred))
```

```
Mean absolute error is : 4.183859899002975
```

As from model evaluation we have received very high R-Squired Value which is :0.9454906892105356 and a low Mean Absolute Error which is : 4.183859899002975, so we can conclude that we are receiving a very good prediction result using this model.

Thank You