THE SPARKS FOUNDATION

Data Science and Buisness Analytics Tasks

NAME : M.NIVETHITHAA

TASK 1 - Prediction using Supervised ML

DATA AUDIT

Importing Libraries

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

Displaying Raw Dataset

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
23	6.9	76

	Hours	Scores
24	7.8	86

First five rows of the dataset

In [3]:
```
data.head(5)
```

Out[3]:

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

Last five rows of the dataset

In [4]:
```
data.tail(5)
```

Out[4]:

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

Shape of the dataset

In [5]:
```
data.shape
```

Out[5]:
```
(25, 2)
```

Columns present in the dataset

In [6]:
```
data.columns
```

Out[6]:
```
Index(['Hours', 'Scores'], dtype='object')
```

Summary of the dataset

In [7]:
```
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
```

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

Checking Datatypes

In [9]:
```
data.dtypes
```

Out[9]:
```
Hours     float64
Scores      int64
dtype: object
```

Checking missing values

In [10]:
```
data.isna().sum()
```

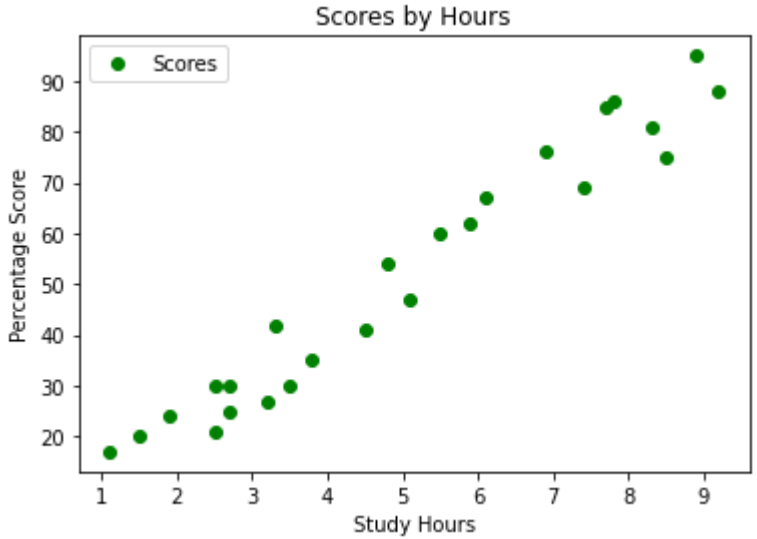
Out[10]:
```
Hours     0
Scores    0
dtype: int64
```

No Missing values

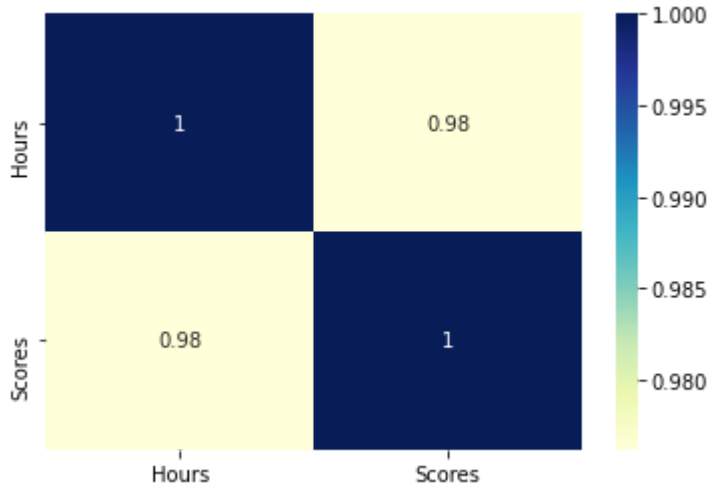
PLOTTING

Plotting Scores by Hours

In [11]:
```
data.plot(x='Hours', y='Scores', style='o',color="green")
plt.title('Scores by Hours')
plt.xlabel('Study Hours')
plt.ylabel('Percentage Score')
plt.show()
```



In [12]:
```
sns.heatmap(data.corr(), cmap="YlGnBu", annot = True)
plt.show()
```



CORRELATION

In [13]:
```
data.corr()
```

Out[13]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

LINEAR REGRESSION

1)Predict the percentage of Student based on number of study hours

Preparing the Data

Create X and Y

In [14]:
```
X = data.iloc[:, :-1].values  #until last column
y = data.iloc[:, 1].values    #last column
```

Create Train and Test sets

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7,
                                                    test_size = 0.3, random_state = 100)
```

In [16]:
```
X_train
```

Out[16]:
```
array([[9.2],
       [4.5],
       [3.5],
       [7.8],
       [2.5],
       [5.1],
       [2.7],
       [1.1],
       [1.9],
       [6.1],
       [3.2],
       [7.7],
       [2.5],
       [8.9],
       [5.5],
       [8.5],
       [8.3]])
```

In [17]:
```
y_train
```

Out[17]:
```
array([88, 41, 30, 86, 21, 47, 30, 17, 24, 67, 27, 85, 30, 95, 60, 75, 81],
      dtype=int64)
```

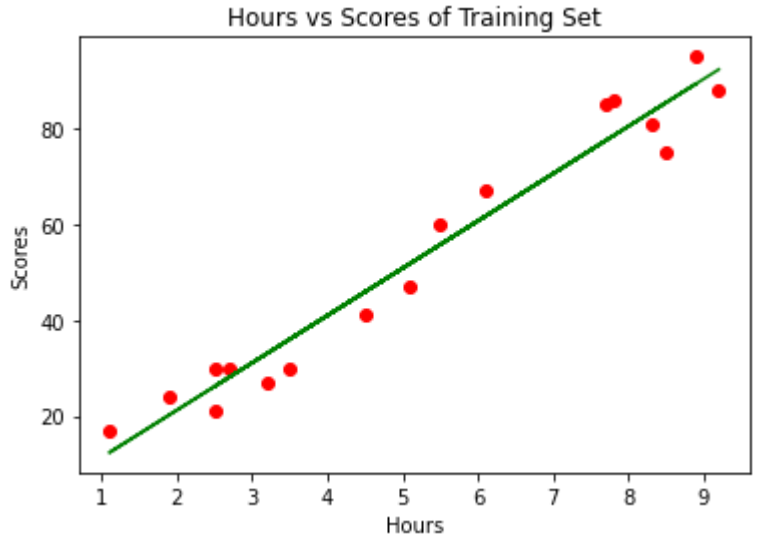
In [18]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

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

Visualization

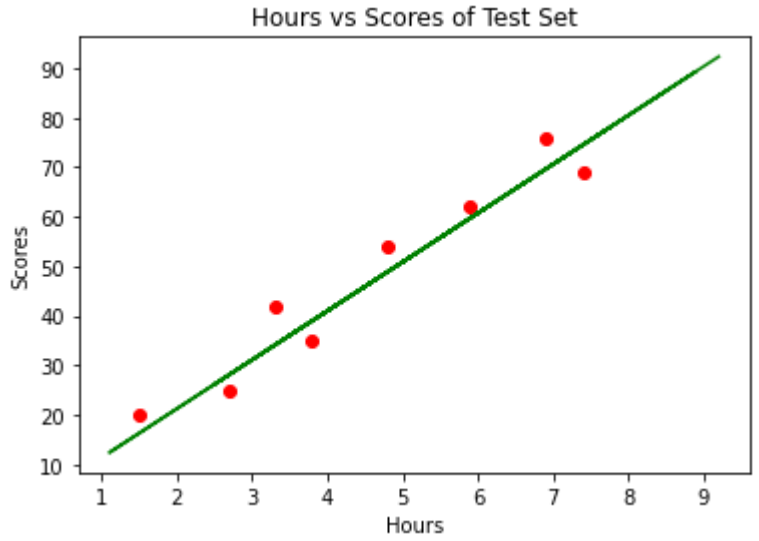
Visualizing the Training set results

In [19]:
```
v_train = plt
v_train.scatter(X_train, y_train, color='red')
v_train.plot(X_train, model.predict(X_train), color='green')
v_train.title('Hours vs Scores of Training Set')
v_train.xlabel('Hours')
v_train.ylabel('Scores')
v_train.show()
```



Visualizing the Test set

In [20]:
```
v_test = plt
v_test.scatter(X_test, y_test, color='red')
v_test.plot(X_train, model.predict(X_train), color='green')
v_test.title('Hours vs Scores of Test Set')
v_test.xlabel('Hours')
v_test.ylabel('Scores')
v_test.show()
```



Predicting the Data - Scores by Hours

In [21]:
```
print(X_test)
y_pred = model.predict(X_test)
y_pred
```

```
[[2.7]
 [3.8]
 [3.3]
 [5.9]
 [1.5]
 [7.4]
 [6.9]
 [4.8]]
```

Out[21]:
```
array([28.14877107, 39.00765694, 34.07179972, 59.73825724, 16.30271375,
       74.54582888, 69.60997167, 48.87937137])
```

Comparing Actual and Predicted Data

In [22]:
```
res = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
res
```

Out[22]:

	Actual	Predicted
0	25	28.148771
1	35	39.007657
2	42	34.071800
3	62	59.738257
4	20	16.302714
5	69	74.545829
6	76	69.609972
7	54	48.879371

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

In [23]:
```
hour = 9.25
y_pred = model.predict([[hour]])
y_pred
```

Out[23]:
```
array([92.80850057])
```

Result : **If a student student studies for 9.25/day he can score 92.80850057 marks as predicted**

Evaluate the Regression Model

In [24]:
```
X_addC = sm.add_constant(X)
result = sm.OLS(y, X_addC).fit()
```

R Square - Square of the Correlation Coefficient

In [25]:
```
print('R Square:',result.rsquared)
```

```
R Square: 0.9529481969048356
```

Adjacent R Square - Statistics based on the number of independent variables

In [26]:
```
print('Adjacent R Square:',result.rsquared_adj)
```

```
Adjacent R Square: 0.9509024663354806
```