GRIP @ THE SPARKS FOUNDATION

TASK : 1 - PREDICTION USING SUPERVISED ML

NAME : JANE MARIA JOSE

AIM :
To predict the percentage of a student based on the number study hours and to determine the predicted score if a student studies for 9.25hrs/day

Importing required libraries

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

Reading data from remote link

In [3]:
```
url = "http://bit.ly/w-data"
s_data = pd.read_csv(url)
print("Data imported successfully")

s_data.head(10)
```

```
Data imported successfully
```

Out[3]:

	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

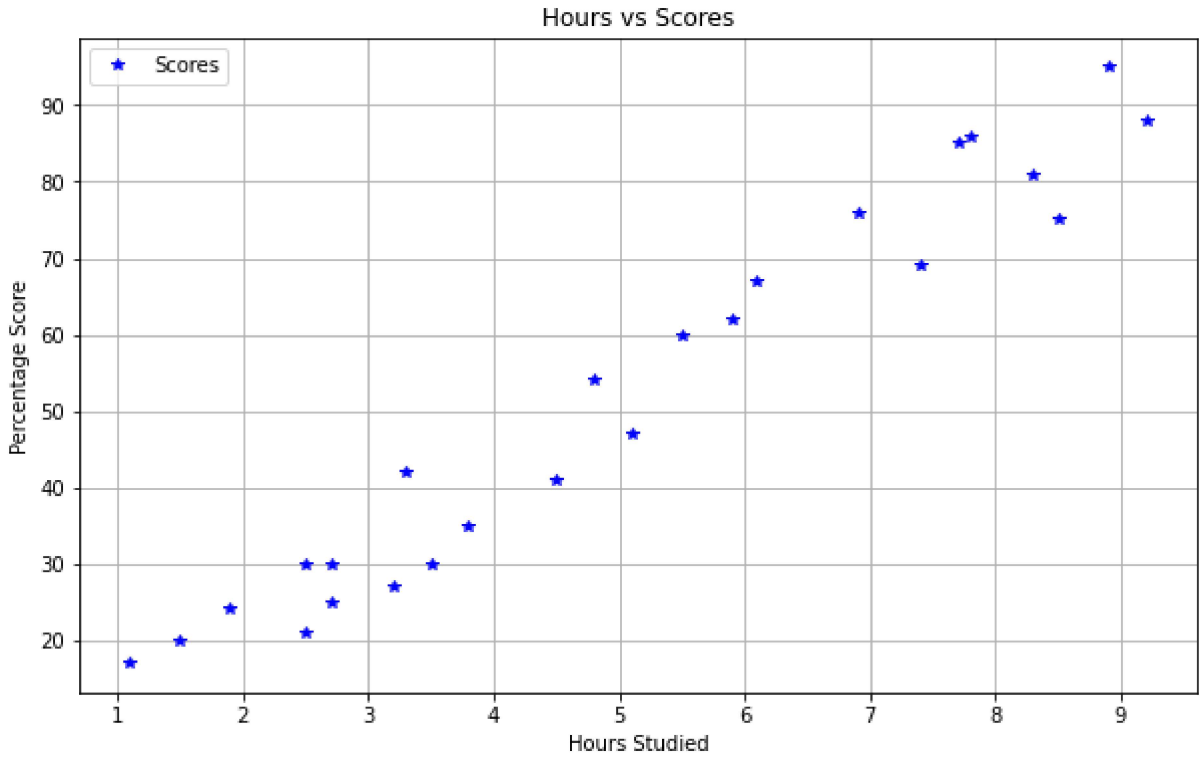
In [4]:
```
s_data.describe()
```

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 [11]:
```
# SCATTER PLOT OF HOURS VS SCORES

plt.rcParams["figure.figsize"]=(10,6)
s_data.plot(x='Hours',y='Scores',style='*',color='blue')
plt.title('Hours vs Scores')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```

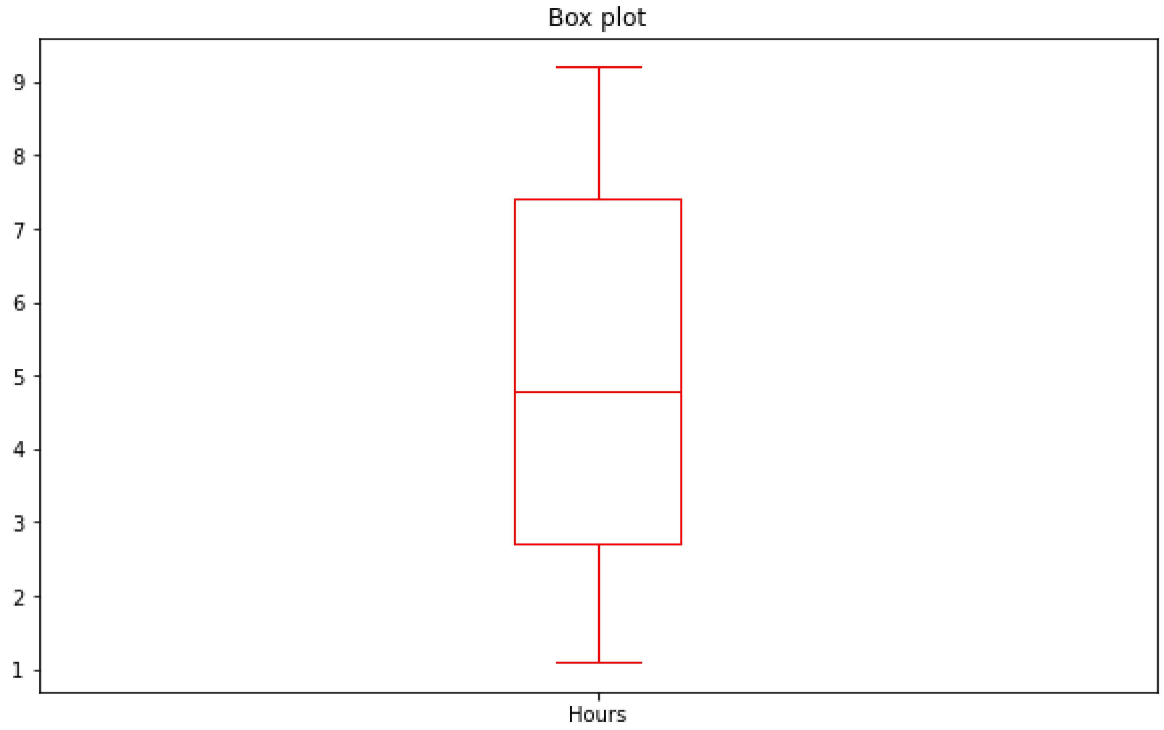


From the above scatter plot, there is a positive relation between Hours studied and percentage score of the student

In [13]:
```
# BOX PLOT OF NO. OF HOURS STUDIED

s_data['Hours'].plot.box(color="red")
plt.title("Box plot")
```

Out[13]: Text(0.5, 1.0, 'Box plot')

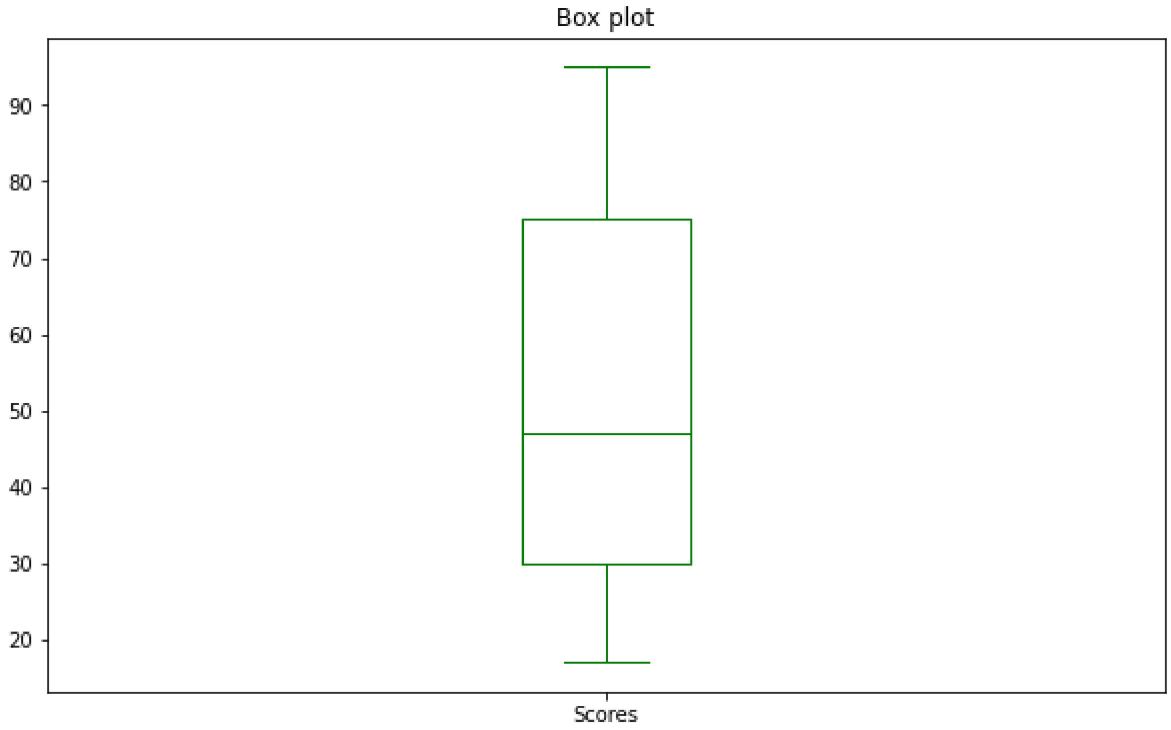


From the above box plot, we can see the median study hours per day by a student is almost 5 hours. It is also clear that there is no outliers. Since median is not equal to mean, it is not normally distributed.

In [14]:
```
# BOX PLOT OF PERCENTAGE SCORE

s_data['Scores'].plot.box(color="green")
plt.title("Box plot")
```

Out[14]: Text(0.5, 1.0, 'Box plot')

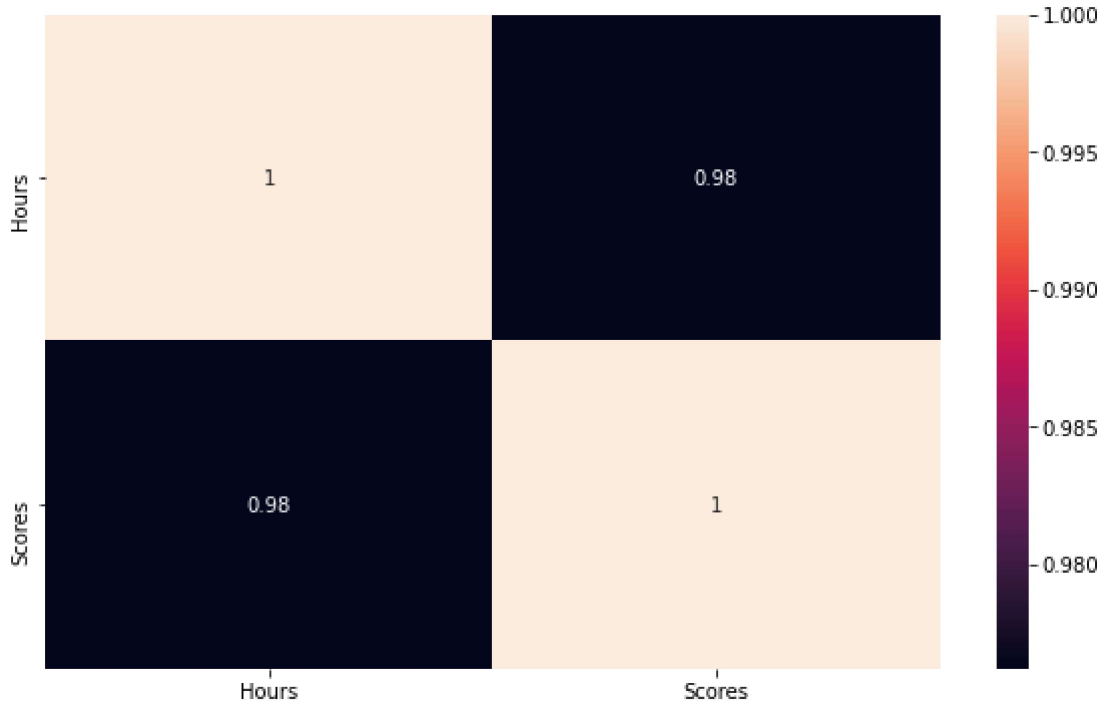


From the above box plot, the median percentage score of a student is aproximately 48% and there is no outliers. Since mean and median is not equal, it is not normally distributed.

In [15]:
```
# HEAT MAP

print("The correlation Heat map is:")
corrMatrix=s_data.corr()
sn.heatmap(corrMatrix,annot=True)
plt.show()
```

```
The correlation Heat map is:
```



The correlation coefficient is 0.98 which implies that the number of hours studied and percentage score of a student are highly positively correlated

Preparing the data

The next step is to divide the data into "attributes" (inputs) and "labels" (outputs)

In [17]:
```
X=s_data.iloc[:, :-1].values
y=s_data.iloc[:, 1].values
```

Now that we have our attributes and labels, the next step is to split this data into training and test sets. We will do this by using Scikit-Learn's built-in train_test_split() method

Spitting data into training and test sets

In [18]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

In [19]:
```
print("Dimension of training set of Scores = ",X_train.ndim)
print("Dimension of training set of Hours = ",y_train.ndim)
```

```
Dimension of training set of Scores =  2
Dimension of training set of Hours =  1
```

Training the Algorithm

We have split our data into training and testing sets and now we proceed to train our algorithm.

In [21]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)

print("The training is complete")
```

```
The training is complete
```

Making prediction on training set and checking the RMSE

In [22]:
```
from sklearn.metrics import r2_score
y_pred=model.predict(X_test)
r2_score(y_test,y_pred)
```

Out[22]:
```
0.9454906892105356
```

In [23]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred,squared=False)
```

Out[23]:
```
4.6474476121003665
```

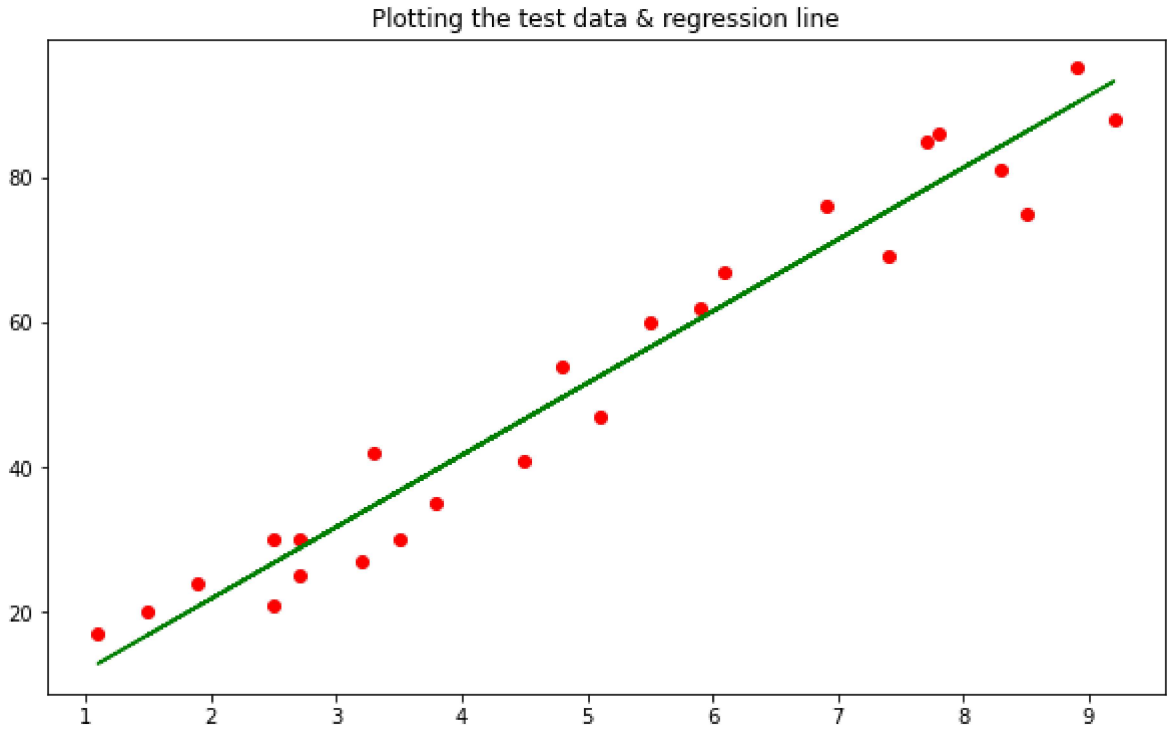
The model has an accuracy score of 0.9454 which implies the model is 94.54% good fit with RMSE value of 4.64%

In [25]:
```
# PLOTTING THE REGRESSION LINE

line = model.coef_*X + model.intercept_

# PLOTTING FOR THE TEST DATA

plt.scatter(X,y,color="red")
plt.plot(X,line,color="green")
plt.title("Plotting the test data & regression line")
plt.show()
```



Prediction

In [26]:
```
# What will be the predicted score if a student studies for 9.25 hrs/day?

hours=9.25
model.predict([[hours]])
```

Out[26]:
```
array([93.69173249])
```

The predicted score if a student studies for 9.25hrs/day is approximately 93.69%

Conclusion

I was successful in completing the prediction using Supervised Machine Learning and is now able to evaluate the model's performance.