Name: Farza Sageer

Task 1: To predict the percentage of a student based on the number of study hours using supervised machine learning.

In [1]:
```
#Loading the packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
```

In [2]:
```
#Importing the dataset

data = pd.read_excel("Data.xlsx")
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

	Hours	Scores
22	3.8	35
23	6.9	76
24	7.8	86

In [3]:
```
data.describe()
```

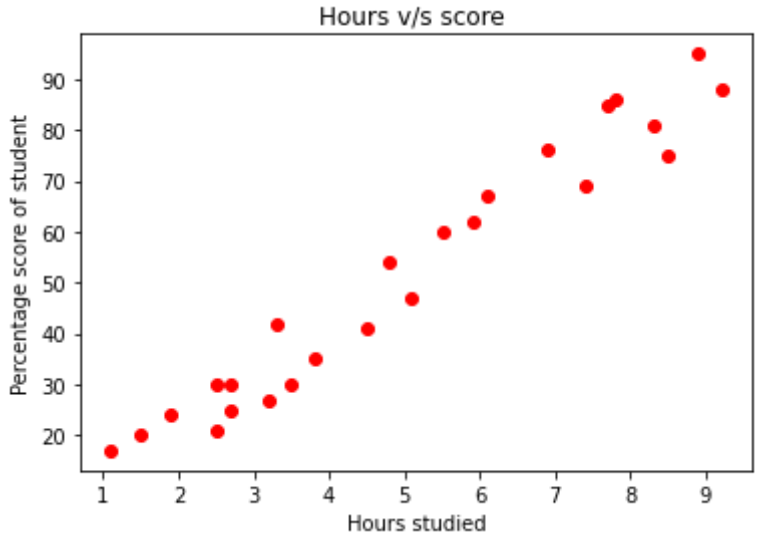
Out[3]:

	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

Fig1: Scatter plot between number of hours studied and the percentage scored

In [4]:
```
x = data['Hours']
y = data['Scores']
plt.xlabel("Hours studied")
plt.ylabel("Percentage score of student")
plt.title("Hours v/s score ")
plt.scatter(x, y, color='red')
```

Out[4]: <matplotlib.collections.PathCollection at 0x1b309346760>



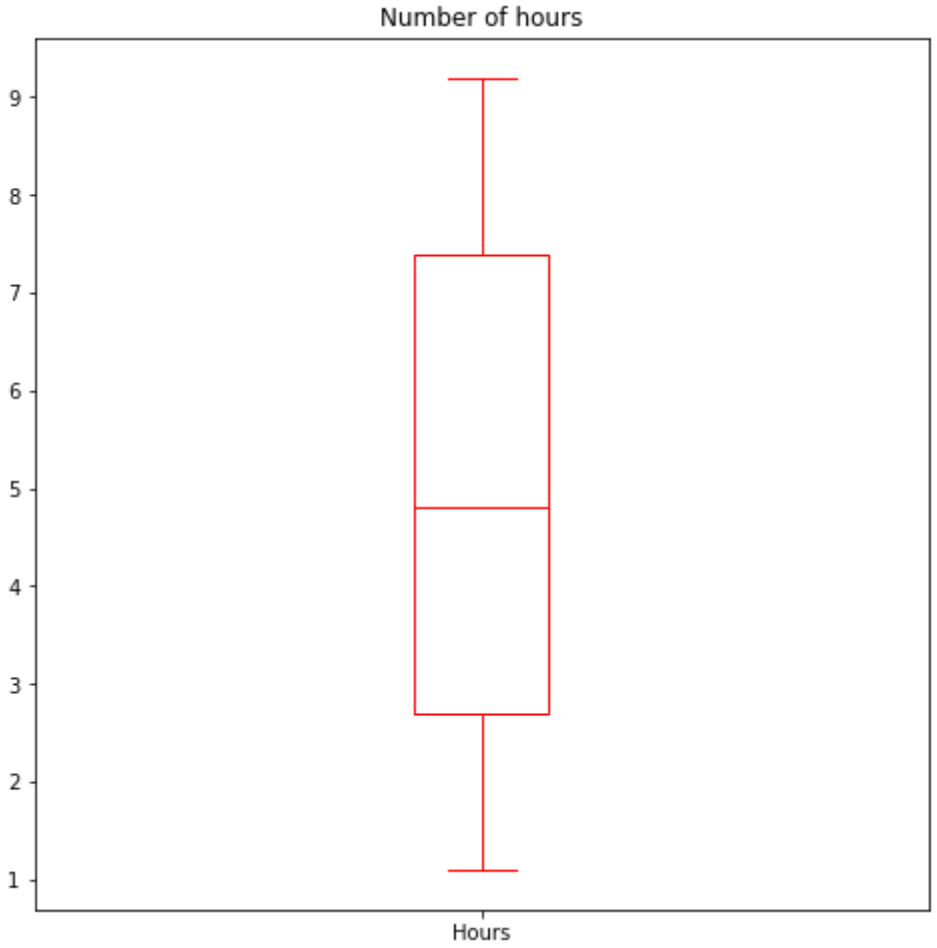
Interpretation: There is a positive correlation between the hours studied

by a student and the percentage scored.

Fig2: Box plot for the number of hours studied

In [5]:
```
data.Hours.plot.box(color="red", figsize=(8,8))
plt.title("Number of hours")
```

Out[5]:
```
Text(0.5, 1.0, 'Number of hours')
```

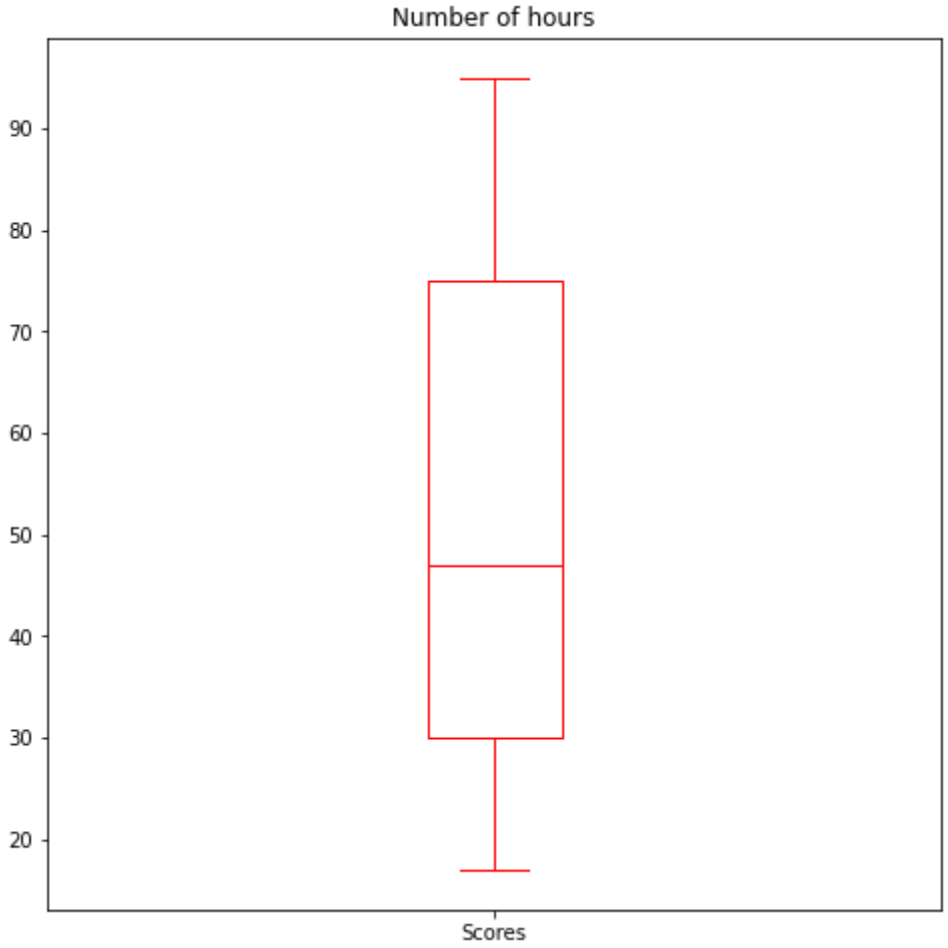


Interpretation: The plot shows that the median hours of study by a student is almost 5 hours per day. It also shows that there is no outliers and that it is not normally distributed since median is not equal to mean.

Fig3: Box plot for scores

In [6]:
```
data.Scores.plot.box(color="red", figsize=(8,8))
plt.title("Number of hours")
```

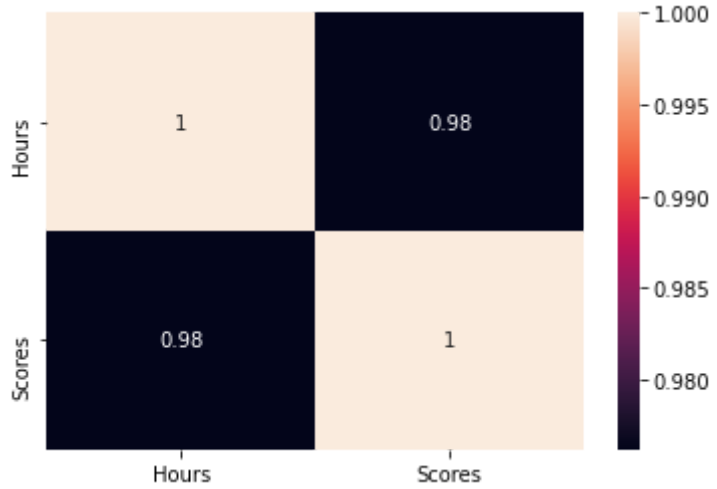
Out[6]:
```
Text(0.5, 1.0, 'Number of hours')
```



Interpretation: The plot shows that the percentage of score by a student is around 48%. It also shows that there is no outliers and that it is not normally distributed since median is not equal to mean.

Fig4: Correlation heat map of number of hours studied and the percentage scored

In [7]:
```
corrmatrix=data.corr()
sn.heatmap(corrmatrix, annot=True)
plt.show()
```



Interpretation: The correlation coefficient obtained is 0.98 which implies that the hours of study and the percentage scored by a student is highly positively correlated.

Data preparation:

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

In [8]:
```
X=data.iloc[:, :-1].values
y=data.iloc[:, 1].values
```

Now we have our attributes and labels, the next step is to split the data into training and test sets using train_test_split() method.

Splitting data into training and testing sets

In [9]:
```
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 [10]:
```
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

In [11]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(X_train, y_train)
print("The training is completed")
```

```
The training is completed
```

Making prediction on training set and checking RMSE

In [12]:
```
from sklearn.metrics import r2_score
y_pred=model.predict(X_test)
r2_score(y_test, y_pred)
```

Out[12]:
```
0.9454906892105356
```

In [13]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred, squared=False)
```

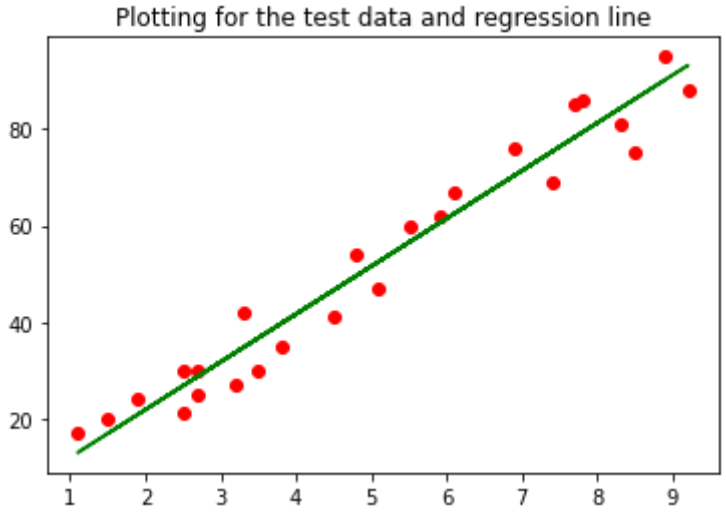
Out[13]:
```
4.6474476121003665
```

Interpretation: The model has an accuracy score of 0.9454. i.e, it is 95 % good fit and its RMSE is 4.64%

Fig5: Plotting the regression line and the test data

In [14]:
```
line=model.coef_*X + model.intercept_
plt.scatter(X, y, color="red")
plt.plot(X, line, color="green")
plt.title("Plotting for the test data and regression line")
```

Out[14]:
```
Text(0.5, 1.0, 'Plotting for the test data and regression line')
```



Prediction:

In [15]:
```
#To predict the of a student if he/ she studies 9.25 hours per day

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

Out[15]:
```
array([93.69173249])
```

Interpretation: A student scores 93.69% if he/ she studies for 9.25 hours per day.