TASK 1: Prediction Using Supervised ML¶
Aim is to predict the score of a student if he/she studies for 9.25 hrs/day¶
Author: Kirti Jain¶
Import the Dataset¶

In [88]:
```
#importing the relevant libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [64]:
```
data= pd.read_csv("https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_scores.csv")
data
```

Out[64]:

	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

	Hours	Scores
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 [65]:
```
#specify rows and coloumn
data.shape
```

Out[65]:
```
(25, 2)
```

In [66]:
```
#name of columns
data.columns
```

Out[66]:
```
Index(['Hours', 'Scores'], dtype='object')
```

In [67]:
```
#check for null values
data.isnull()
```

Out[67]:

	Hours	Scores
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False

	Hours	Scores
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False
19	False	False
20	False	False
21	False	False
22	False	False
23	False	False
24	False	False

In [68]:
```
data.isnull().sum()
```

Out[68]:
```
Hours     0
Scores    0
dtype: int64
```

In [69]:

```
data.describe()
```

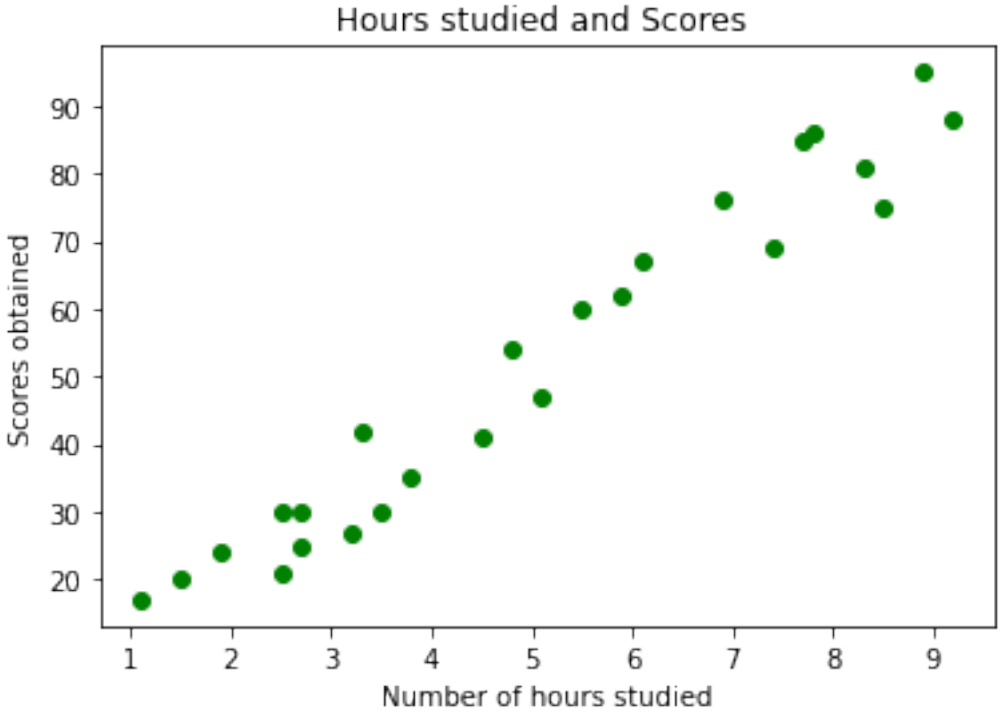
Out[69]:

	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

Visualize and Analyse the Dataset¶

In [70]:

```
#Scatter plot of Number of hours studied and score obtained
plt.scatter(data['Hours'],data['Scores'],color='green')
plt.title('Hours studied and Scores ')
plt.xlabel('Number of hours studied')
plt.ylabel('Scores obtained')
plt.show()
```



In [71]:

```
#check the correlation between two coloumns
data.corr()
```

Out[71]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

From the above graph and data, we can clearly see that there is a positive linear relation between the number of hours studied and scores.

Prepare the Data¶

In [72]:
```
#divide the data into input and output
x=data.iloc[:, :1].values
y=data.iloc[:, 1:].values
```

In [73]:
```
#Hours studied
x
```

Out[73]:
```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [74]:
```
#scores obtained
y
```

Out[74]:
```
array([[21],
       [47],
       [27],
       [75],
       [30],
       [20],
       [88],
       [60],
```

```
       [81],
       [25],
       [85],
       [62],
       [41],
       [42],
       [17],
       [95],
       [30],
       [24],
       [67],
       [69],
       [30],
       [54],
       [35],
       [76],
       [86]], dtype=int64)
```

Design and Train the Machine Learning Model¶

In [75]:
```
#split the data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x,y,test_size=0.3,random_state=42)
```

In [76]:
```
from sklearn.linear_model import LinearRegression
model= LinearRegression()
model.fit(x_train, y_train)
```

Out[76]:
```
LinearRegression()
```

In [77]:
```
model.coef_
```

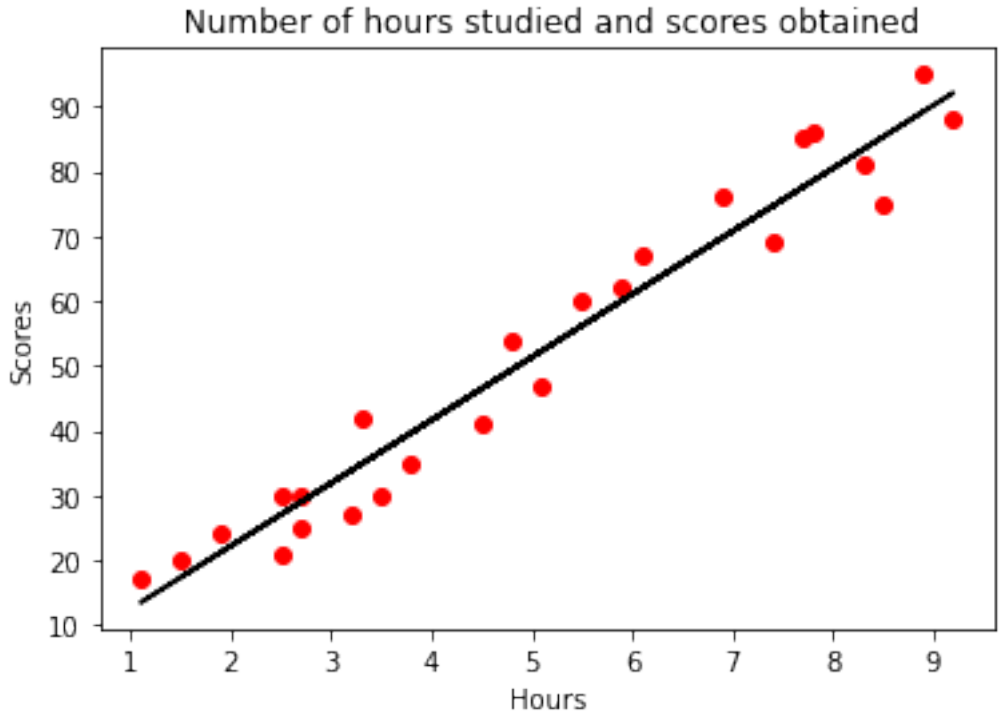
Out[77]:
```
array([[9.71054094]])
```

In [78]:
```
model.intercept_
```

Out[78]:
```
array([2.79419668])
```

Visualize the Model¶

In [79]:
```
#plotting the regression line
#y=mx+c
regression_line= model.coef_*x+model.intercept_
plt.scatter(x,y,color='red')
plt.plot(x,regression_line,color='black')
plt.title(' Number of hours studied and scores obtained')
plt.xlabel('Hours')
plt.ylabel('Scores')
plt.show()
```



Make Predictions¶

In [89]:

```
print(x_test)
y_pred= model.predict(x_test)
```

```
[[8.3]
 [2.5]
 [2.5]
 [6.9]
 [5.9]
 [2.7]
 [3.3]
 [5.1]]
```

In [87]:

```
results= pd.DataFrame({'Actual scores': y_test.ravel(),'Predicted scores':
y_pred.ravel()})
results
```

Out[87]:

	Actual scores	Predicted scores
0	81	83.391686
1	30	27.070549
2	21	27.070549
3	76	69.796929
4	62	60.086388
5	25	29.012657
6	42	34.838982

	Actual scores	Predicted scores
7	47	52.317955

In [82]:
```
Hours=9.25
result= model.predict([[9.25]])
print('The predicted score is ', result)
```
```
The predicted score is  [[92.61670034]]
```

Evaluate the Model¶

In [86]:
```
from sklearn import metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, y_pred))
print('R-2r:',metrics.r2_score(y_test, y_pred))
```
```
Mean Absolute Error: 4.499999999999998
Mean Squared Error: 23.61945761415174
R-2r: 0.9487647340257012
```