Task--2 Prediction using supervised machine learning

Objective: Predict the percentage of an student based on no. of study hours.

Import all libraries ¶

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
C:\Users\HP\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarnin
g: A NumPy version >=1.16.5 and <1.23.0 is required for this version of Sc
iPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Upload and Read the dataset

In [2]:
```
df=pd.read_csv('Scores.csv')
```

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
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 [5]:
```
1 df.shape
```

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

Data Information

There are 25 entries are avaliable in this data set and 2 columns are there.

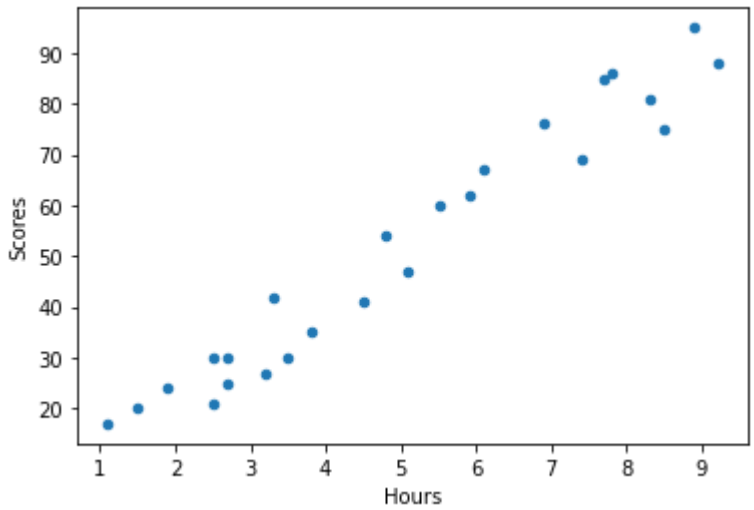
In [6]:
```
1 df.describe()
```

Out[6]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

Visulazation

In [7]:
```
1 df.plot(kind='scatter',x='Hours',y='Scores')
2 plt.show()
```



Correlation matrix

In [8]:
```
1 df.corr(method='pearson')
```

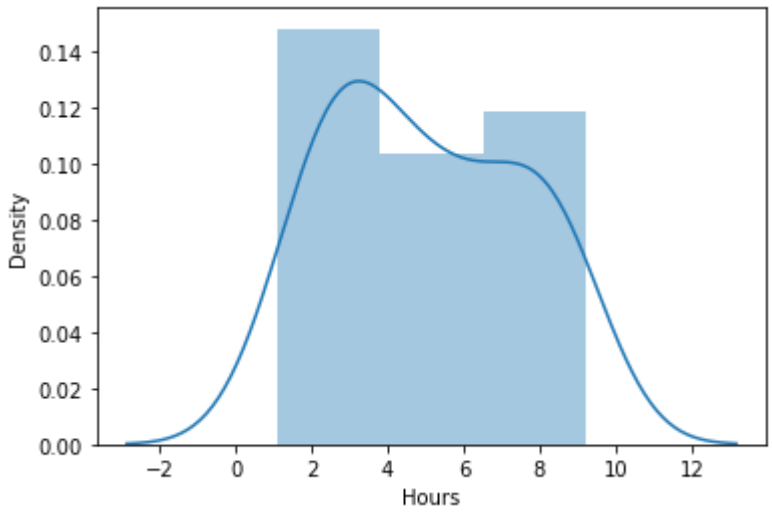
Out[8]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [9]:
```
1 hours=df['Hours']
2 scores=df['Scores']
```

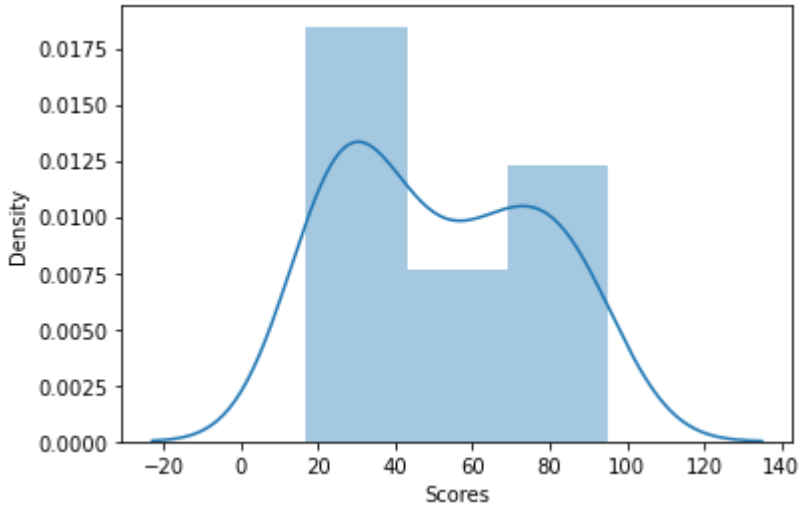
In [10]:
```
1 sns.distplot(hours)
```

Out[10]: <AxesSubplot:xlabel='Hours', ylabel='Density'>



In [11]:
```
1 sns.distplot(scores)
```

Out[11]: <AxesSubplot:xlabel='Scores', ylabel='Density'>



In [12]:
```
1 #seprate X and y
2 #x.......2d
3 #y.......1d
```

Seprate the X and y

In [13]:
```
#x=df[['Hours']]
x=df.iloc[:,:1]
```

In [14]:
```
x
```

Out[14]:

	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

In [15]:
```
x.ndim
```

Out[15]: 2

In [16]:
```
y=df['Scores']
y
```

Out[16]:
```
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
```

Target Column

Y is target column in this data set

Linear regression

Split the data in to train and test

In [17]:
```
#For spliting to data between training and testing
from sklearn.model_selection import train_test_split
```

In [18]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_st
```

In [19]:
```
1 xtrain
```

Out[19]:

	Hours
10	7.7
18	6.1
19	7.4
4	3.5
2	3.2
20	2.7
6	9.2
7	5.5
22	3.8
1	5.1
16	2.5
0	2.5
15	8.9
24	7.8
23	6.9
9	2.7
8	8.3
12	4.5
11	5.9
5	1.5

In [20]:
```
1 ytrain
```

Out[20]:
```
10    85
18    67
19    69
4     30
2     27
20    30
6     88
7     60
22    35
1     47
16    30
0     21
15    95
24    86
23    76
9     25
8     81
12    41
11    62
5     20
Name: Scores, dtype: int64
```

In [21]:
```
#build ml model
#step1:import model
from sklearn.linear_model import LinearRegression
```

In [22]:
```
#step2: create an instance of a model /init a model
lr=LinearRegression()
lr
```

Out[22]:
```
LinearRegression
LinearRegression()
```

In [23]:
```
#step3: Train a model
lr.fit(xtrain,ytrain)#in linear regression finding out best fit line by
```

Out[23]:
```
LinearRegression
LinearRegression()
```

In [24]:
```
#step4:predict values:
ypre=lr.predict(xtest)
```

In [25]:
```
ypre
```

Out[25]:
```
array([ 9.97026179, 32.98470004, 18.33914843, 87.38246316, 48.67636248])
```

In [26]:
```
xtest
```

Out[26]:

	Hours
14	1.1
13	3.3
17	1.9
3	8.5
21	4.8

In [27]:
```
ytest
```

Out[27]:
```
14    17
13    42
17    24
3     75
21    54
Name: Scores, dtype: int64
```

Comparing actual vs predicted

In [39]:
```
actual_predicted=pd.DataFrame({'Target':ytest,'Predicted':ypre})
actual_predicted
```

Out[39]:

	Target	Predicted
14	17	9.970262
13	42	32.984700
17	24	18.339148
3	75	87.382463
21	54	48.676362

Model Evaluation

In [40]:
```
#evaluate a model
from sklearn.metrics import r2_score#to find accuracy of the data
```

In [41]:
```
r2_score(ytest,ypre)
```

Out[41]:
```
0.8421031525243527
```

In [42]:
```
#predict new observation
lr.predict([[4.4]])
```

Out[42]:
```
array([44.49191916])
```

In [43]:
```
lr.coef_#slope
```

Out[43]:
```
array([10.46110829])
```

In [32]:
```
lr.intercept_#inrecept
```

Out[32]:
```
-1.5369573315500702
```

Conclusion

After build the model we get 84% of r2_score.

In [ ]:
```

```