FDS OBSERVARTION

230701097

Haresh.R cse-‘b’

1.SEABORN

```
230701097
Haresh R
2 CSE - B
CS2334 FUNDAMENTAL OF DATA SCIENCE
```

[318]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib pyplot as plt
%matplotlib inline
```

[322]:
```
data=pd read_csv( Iris - Iris.csv')
data
```

```
1            4.9 3.0         1.4         0.2 Setosa

2            4.7 3.2         1.3         0.2 Setosa

3            4.6 3.1         1.5         0.2 Setosa

4            5.0 3.6         1.4         0.2 Setosa

..           …           …           …           …        …

145          6.7 3.0         5.2         2.3
                                                  Virginica
146          6.3 2.5         5.0         1.9
                                                  Virginica
147          6.5 3.0         5.2         2.0
                                                  Virginica
148          6.2 3.4         5.4         2.3
                                                  Virginica
149          5.9 3.0         5.1         1.8
                                                  Virginica
[150 rows x 5 columns]
```

[324]:
```
data ınfo()
```

```
<class
 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 150 entries, 0 to
149 Data columns (total 5 columns):
 # Column       Non-Null Count Dtype
--- ------ -------------- ----0 sepal.length
 150 non-null float64
1 sepal.width 150 non-null float64 2
petal.length 150 non-null float64 3
petal.width 150 non-null float64 4
variety    150 non-null object dtypes:
float64(4), object(1) memory usage:
6.0+ KB
```

[326]:
```
data describe()
```

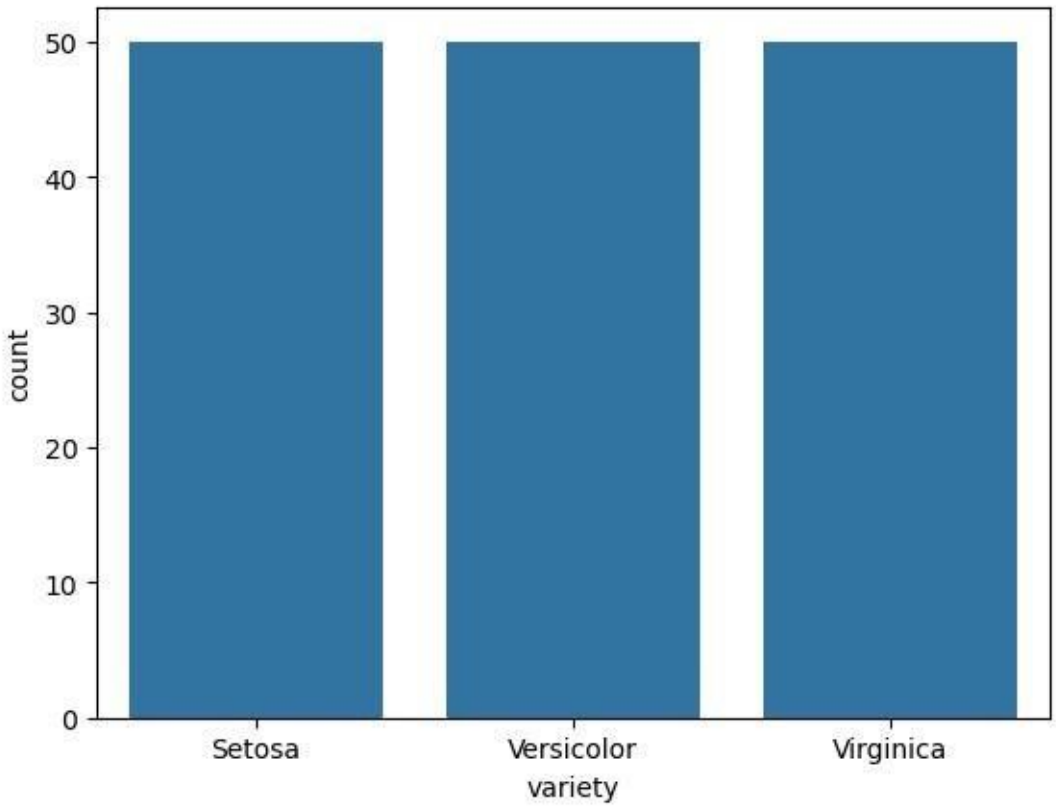
[326]:

	sepal.length	sepal.width	petal.length	petal.width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

[328]:
```
data value_counts( variety )
```

[328]:
```
variety

Setosa       50

Versicolor 50

 Virginica 50

Name: count, dtype: int64
```

[330]:
```
sns countplot(x  variety  data=data,)
plt show()
```




```
[332]: dummies=pd.get_dummies(data.variety)
       FinalDataset=pd.concat([pd.get_dummies(data.variety),da
       ta.iloc[: ↪,[0,1,2,3]]],axis=1) FinalDataset.head()
```

[332]: Setosa Versicolor Virginica sepal.length sepal.width petal.length
\
0 True False False 5.1 3.5 1.4

1 True False False 4.9 3.0 1.4

2 True False False 4.7 3.2 1.3

3 True False False 4.6 3.1 1.5

4 True False False 5.0 3.6 1.4 petal.width 0 0 .....

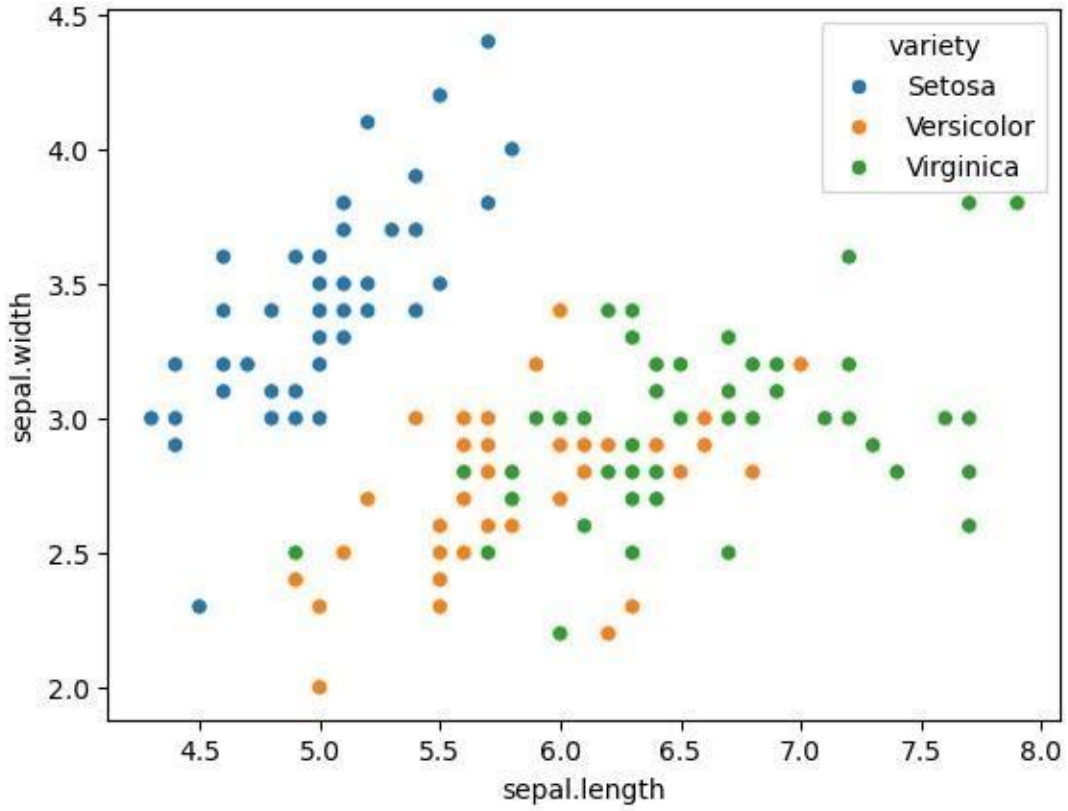
Error! Bookmark not defined. 1 0

..... 10

2 0 .....
11

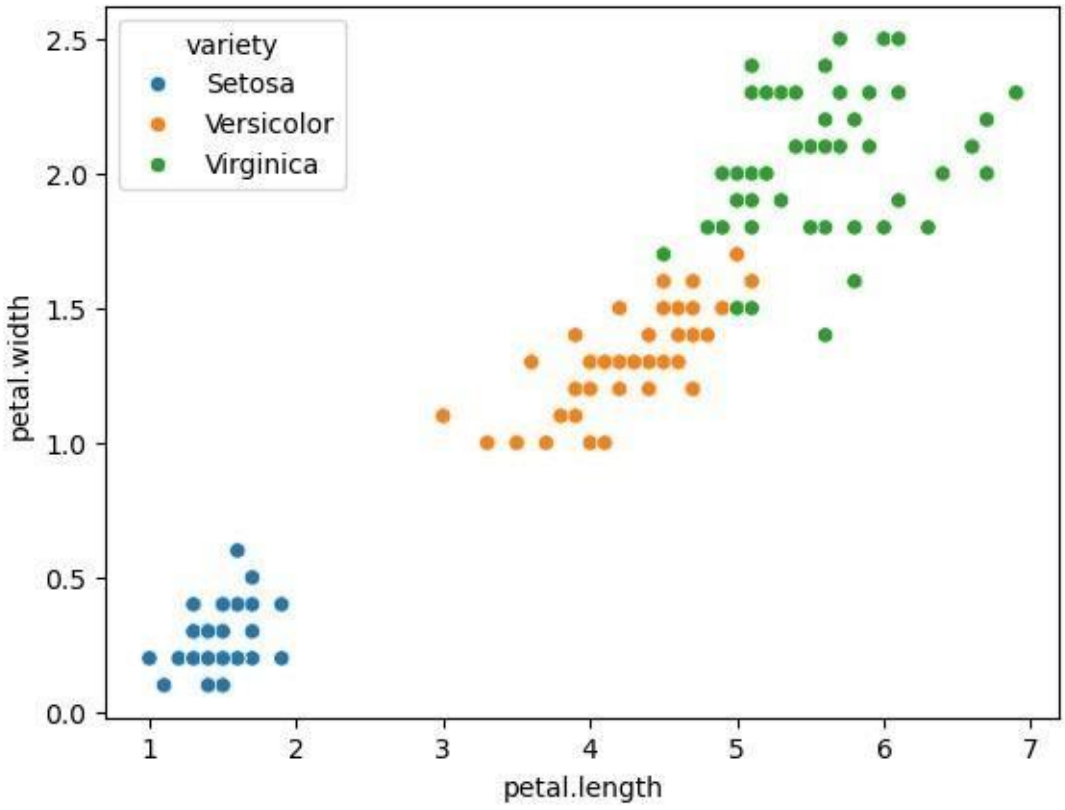
```
3  0 ..........................................................
                                                             2

4  0 ..........................................................
                                                             2
```

[340]:
```
sns.scatterplot(x='sepal.length',y='sepal.width',hue='variety',data=data
)
```
[340]: <Axes: xlabel='sepal.length', ylabel='sepal.width'>

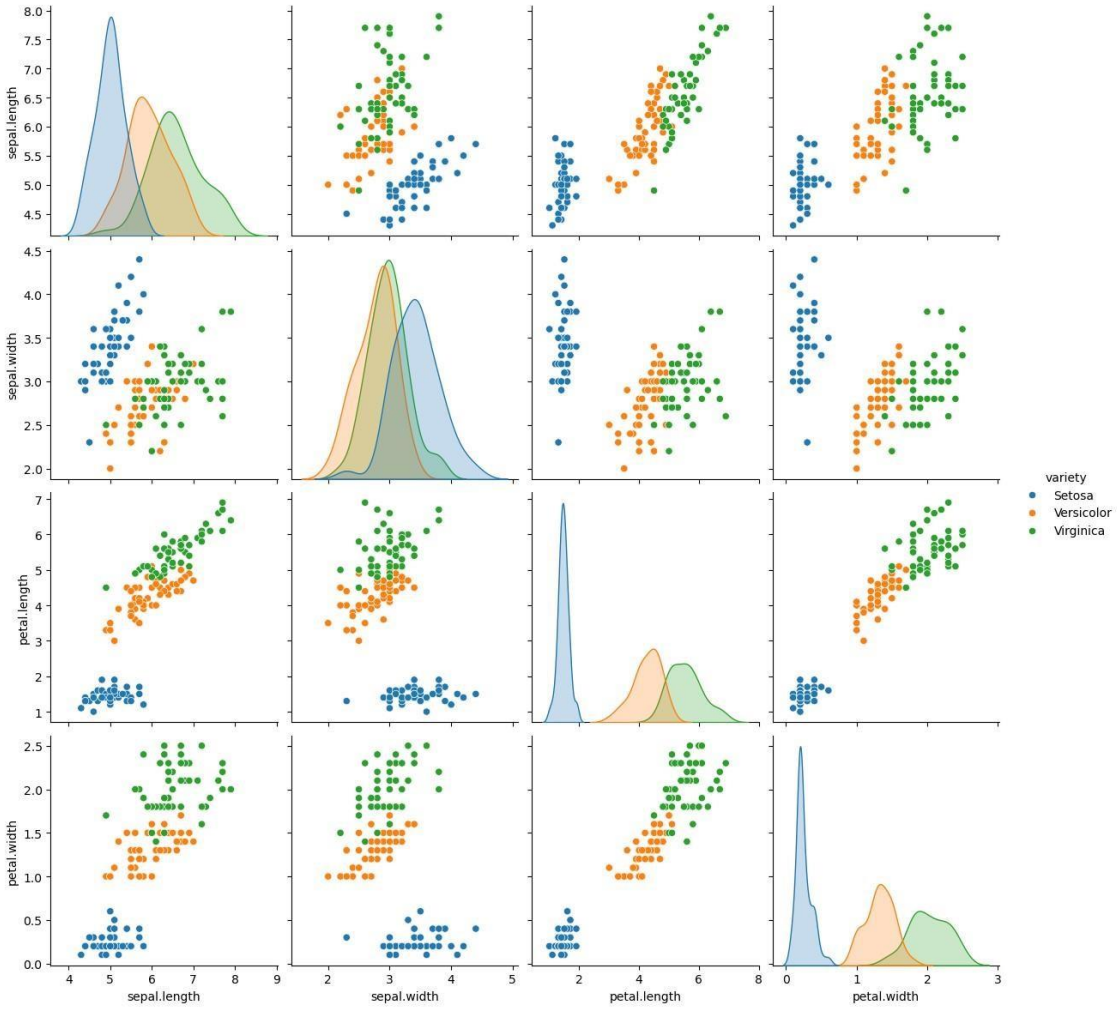


[342]:
```
sns.scatterplot(x='petal.length',y='petal.width',hue='variety',data=data
,)
```

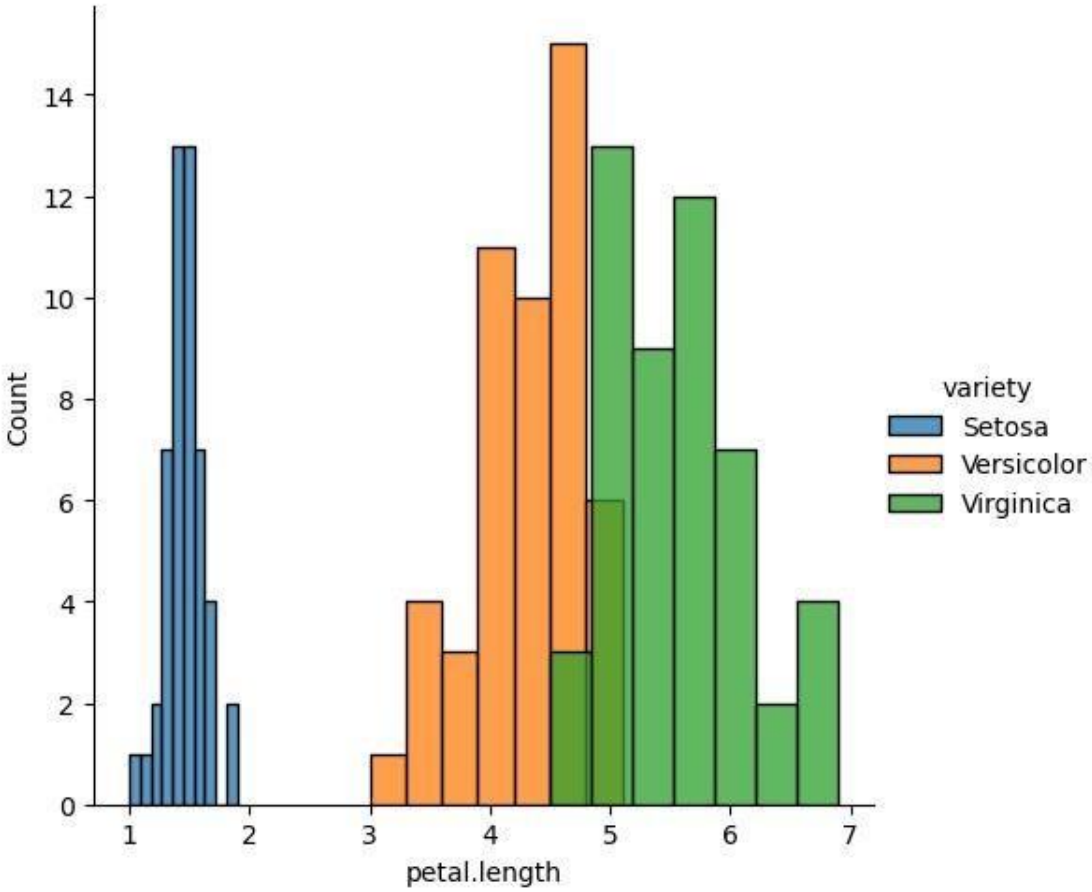
[342]: <Axes: xlabel='petal.length', ylabel='petal.width'>



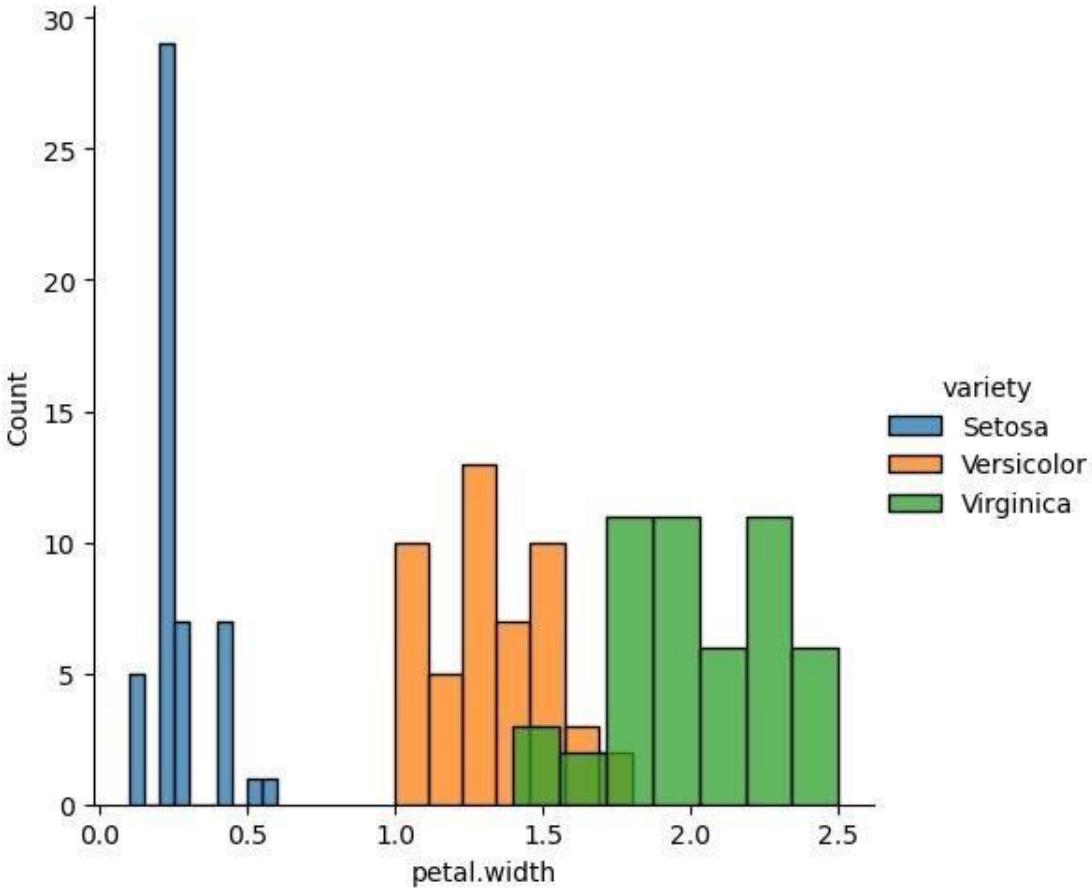
[345]:
```
sns pairplot(data,hue  variety  height=3)
```



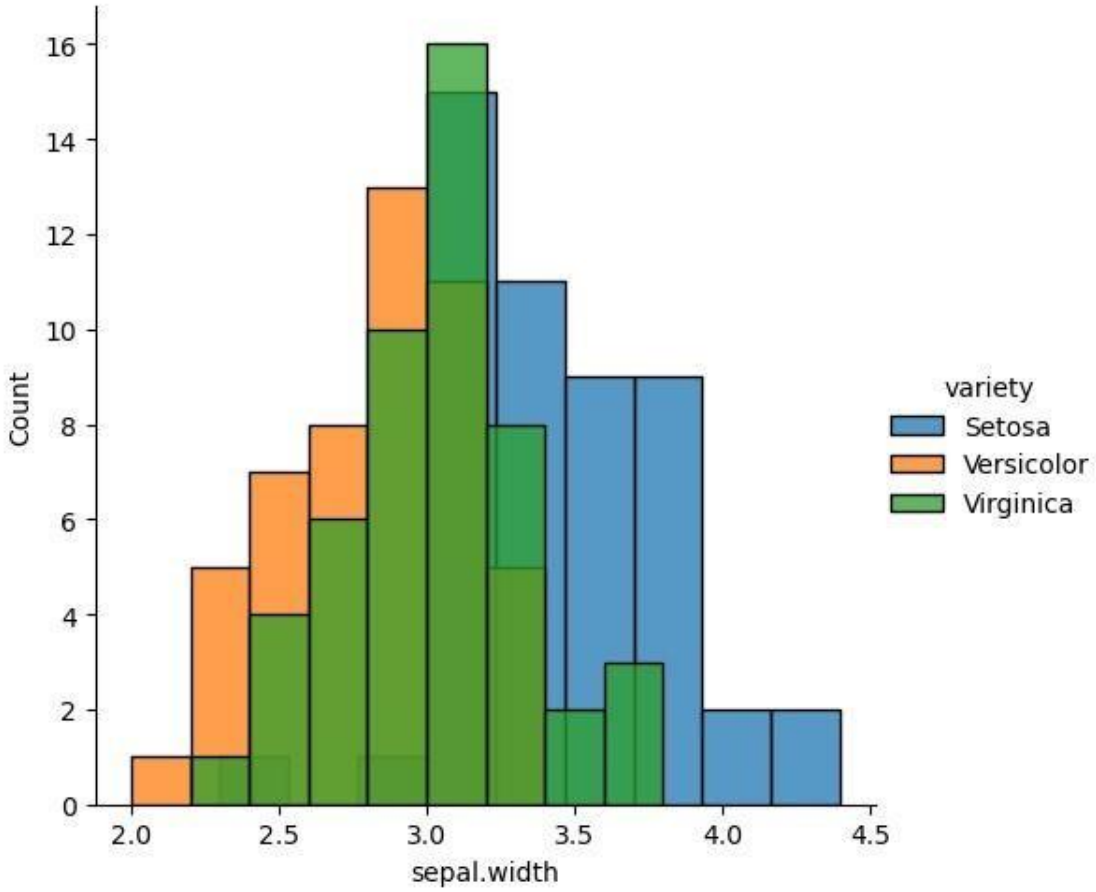
[351]:
```
sns FacetGrid(data,hue= variety ,height=5) map(sns histplot, petal.length )
  ↪add_legend();
plt show();
```



[353]:
```
sns FacetGrid(data,hue= variety ,height=5) map(sns histplot, petal.width )
  ↪add_legend();
plt show();
```



[355]:
```
sns FacetGrid(data,hue= variety ,height=5) map(sns histplot, sepal.width )
  ↪add_legend();
plt show();
```


16
14
12
10
8
6
4
2
0
Count
2.0
2.5
3.0
3.5
4.0
4.5
sepal.width
variety
Setosa
Versicolor
Virginica

[ ]:

2.NUMPY

```
230701097
Haresh R
2 CSE - B
CS2334 FUNDAMENTAL OF DATA SCIENCE
```

```
[7]: import numpy as np
     array=np random randint(1 100,9)
     array

[7]: array([38, 13, 41, 2, 67, 22, 22, 79, 62])
[9]: np sqrt(array)
```

```
[9]: array([6.164414 , 3.60555128, 6.40312424, 1.41421356,
8.18535277,
       4.69041576, 4.69041576, 8.88819442, 7.87400787])
```

```
[11]: array ndim //number of dimension

[11]: 1
[15]: new_array=array reshape(3 3) //changes 1d to 2d
      new_array

[15]: array([[38, 13,
       41], [ 2,
       67, 22],
       [22, 79, 62]])
[17]: new_array ndim

[17]: 2
[19]: new_array ravel() //flattens 2d into 1d

[19]: array([38, 13, 41, 2, 67, 22, 22, 79, 62])
[25]: newm=new_array reshape(3 3)
      newm
```



```
[25]: array([[38, 13,
       41], [ 2,
       67, 22],
```

```
       [22, 79, 62]])
```

[27]:
```
newm[2 1 3]
```

[27]:
```
array([79, 62])
```

[29]:
```
newm[1 2 1 3]
```

[29]:
```
array([[67, 22]])
```

[31]:
```
new_array[0 3 0 0]
```

[31]:
```
array([], shape=(3, 0), dtype=int32)
```

[33]:
```
new_array[0 2 0 1]
```

[33]:
```
array([[38],
       [ 2]])
```

[35]:
```
new_array[0 3 0 1]
```

[35]:
```
array
      ([[38
      ], [
      2],
      [22]])
```

[37]:
```
new_array[1 3]
```

[37]:
```
array([[ 2, 67, 22],
       [22, 79, 62]])
```

[ ]:

3.PANDAS

```
230701097
Haresh R
2 CSE - B
CS2334 FUNDAMENTAL OF DATA SCIENCE
```

```
[3]: 0        1     2
     0  1 Smith 50000
     1  2 Jones 60000
```

```
[5]: df columns=[ Empd   Name   Salary ]
     df
```

```
[5]: Empd Name Salary 0
      1 Smith 50000
     1     2 Jones 60000
```

```
[7]: df ınfo()
```

```
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 2 entries, 0 to
     1 Data columns (total 3
     columns):
      # Column Non-Null Count Dtype
     --- ------ -------------- -----
       0   Empd 2 non-null int64
       1   Name 2 non-null object
       2   Salary 2 non-null     int64 dtypes: int64(2), object(1)
      memory usage: 180.0+ bytes
```

```
[13]: df=pd read_csv( 3_50_Startups.csv )
      df head()
```

```
[13]: R&D Spend Administration Marketing Spend State Profit 0
     165349.20 136897.80 471784.10 New York 192261.83
     1 162597.70       151377.59 443898.53 California 191792.06
     2 153441.51       101145.55 407934.54 Florida 191050.39
     3 144372.41       118671.85 383199.62 New York 182901.99
     4 142107.34       91391.77 366168.42 Florida 166187.94
```

```
[15]: df ınfo()
```

```
     <class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 50 entries, 0 to
49 Data columns (total 5
columns):
 # Column          Non-Null Count Dtype
--- ------  -------------- -----0 R&D
Spend 50 non-null float64
1 Administration 50 non-null float64 2
Marketing Spend 50 non-null float64
3 State     50 non-null object 4 Profit
 50 non-null float64 dtypes: float64(4),
object(1) memory usage: 2.1+ KB
```

[17]:
```
df tail()
```

[17]:
```
 R&D Spend Administration Marketing Spend State Profit 45 1000.23
124153.04 1903.93 New York 64926.08 46 1315.46
115816.21 297114.46 Florida 49490.75
47      0.00     135426.92 0.00 California 42559.73
48      542.05 51743.15 0.00 New York 35673.41 49 0.00
        116983.80 45173.06 California 14681.40
```

[25]:
```
df Profit
```

[25]:
```
 0   192261.83
1    191792.06
2    191050.39
3    182901.99
4    166187.94
5    156991.12
6    156122.51
7    155752.60
8    152211.77
9    149759.96
10   146121.95
11   144259.40
12   141585.52
13   134307.35
14   132602.65
 15    129917.04
 16    126992.93
 17    125370.37
 18    124266.90
 19    122776.86



 20    118474.03
```

```
21    111313.02
22    110352.25
23    108733.99
24    108552.04
25    107404.34
26    105733.54
27    105008.31
28    103282.38
29    101004.64
30    99937.59
31    97483.56
32    97427.84
33    96778.92
34    96712.80
35    96479.51
36    90708.19
37    89949.14
38    81229.06
39    81005.76
40    78239.91
41    77798.83
42    71498.49
43    69758.98
44    65200.33
45    64926.08
46    49490.75
47    42559.73
48    35673.41
49    14681.40
Name: Profit, dtype: float64
```

[27]:
```
type(df Profit)
```

[27]:
```
pandas.core.series.Series
```

[29]:
```
df Profit mean()
```

[29]:
```
112012.63920000002
```

[31]:
```
df Profit median()
```

[31]:
```
107978.19
```

[33]:
```
df Profit mode()
```

[33]:
```
0    14681.40
```

1	35673.41
2	42559.73
3	49490.75
4	64926.08
5	65200.33
6	69758.98
7	71498.49
8	77798.83
9	78239.91
10	81005.76
11	81229.06
12	89949.14
13	90708.19
14	96479.51
15	96712.80
16	96778.92
17	97427.84
18	97483.56
19	99937.59
20	101004.64
21	103282.38
22	105008.31
23	105733.54
24	107404.34
25	108552.04
26	108733.99
27	110352.25
28	111313.02
29	118474.03
30	122776.86
31	124266.90
32	125370.37
33	126992.93
34	129917.04
35	132602.65
36	134307.35
37	141585.52
38	144259.40
39	146121.95
40	149759.96
41	152211.77
42	155752.60
43	156122.51
44	156991.12
45	166187.94

```
46    182901.99
47    191050.39
48    191792.06 49192261.83
 Name: Profit, dtype: float64
```

[35]:
```
df Profit var
```

[35]:
```
<bound method Series.var of 0192261.83
1     191792.06
2     191050.39
3     182901.99
4     166187.94
5     156991.12
6     156122.51
7     155752.60
8     152211.77
9     149759.96
10    146121.95
11    144259.40
12    141585.52
13    134307.35
14    132602.65
15    129917.04
16    126992.93
17    125370.37
18    124266.90
19    122776.86
20    118474.03
21    111313.02
22    110352.25
23    108733.99
24    108552.04
25    107404.34
26    105733.54
27    105008.31
28    103282.38
29    101004.64
30    99937.59
31    97483.56
32    97427.84
33    96778.92
34    96712.80
```

```
35    96479.51
36    90708.19
37    89949.14
38    81229.06
39    81005.76
40    78239.91
41    77798.83
42    71498.49
43    69758.98
44    65200.33
45    64926.08
46    49490.75
47    42559.73
48    35673.41
49    14681.40
Name: Profit, dtype: float64>
```

[37]:
```
df Profit std
```

[37]:
```
<bound method Series.std of 0192261.83
1     191792.06
2     191050.39
3     182901.99
4     166187.94
5     156991.12
6     156122.51
7     155752.60
8     152211.77
9     149759.96
10    146121.95
11    144259.40
12    141585.52
13    134307.35
14    132602.65
15    129917.04
16    126992.93
17    125370.37
18    124266.90
19    122776.86
20    118474.03
21    111313.02
22    110352.25
```

```
23    108733.99
24    108552.04
25    107404.34
26    105733.54
27    105008.31
28    103282.38
29    101004.64
30    99937.59
31    97483.56
32    97427.84
33    96778.92
34    96712.80
35    96479.51
36    90708.19
37    89949.14
38    81229.06
39    81005.76
40    78239.91
41    77798.83
42    71498.49
43    69758.98
44    65200.33
45    64926.08
46    49490.75
47    42559.73
48    35673.41
49    14681.40
Name: Profit, dtype: float64>
```

[39]:
```
df describe()
```

[39]:
```
       R&D Spend Administration Marketing Spend      Profit
count    50.000000     50.000000         50.000000 50.000000
mean   73721.615600 121344.639600 211025.097800
                                   112012.639200
std    45902.256482 28017.802755 122290.310726
                                                  40306.180338
min        0.000000 51283.140000       0.000000 14681.400000
25%    39936.370000 103730.875000 129300.132500
                                                  90138.902500
50%    73051.080000 122699.795000 212716.240000
                                   107978.190000
```

```
75%    101602.800000 144842.180000 299469.085000
                                    139765.977500
max    165349.200000 182645.560000 471784.100000
                                    192261.830000
```

[41]:
```
df describe(include  all )
```

[41]:
```
           R&D Spend Administration Marketing Spend State \
count     50.000000      50.000000    50.000000         50
unique          NaN            NaN           NaN          3
top             NaN            NaN           NaN New York
freq            NaN            NaN           NaN         17
mean    73721.615600  121344.639600  211025.097800        NaN
std     45902.256482   28017.802755  122290.310726        NaN
min         0.000000   51283.140000       0.000000        NaN
25%     39936.370000  103730.875000  129300.132500        NaN
50%     73051.080000  122699.795000  212716.240000        NaN
75%    101602.800000  144842.180000  299469.085000        NaN
max    165349.200000  182645.560000  471784.100000        NaN

              Profit
count      50.000000
unique           NaN
top              NaN
freq             NaN
mean    112012.639200
std      40306.180338
min      14681.400000
25%      90138.902500
50%     107978.190000
75%     139765.977500
max     192261.830000
```

[43]:
```
a=df columns
a
```

[43]:
```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State',
'Profit'], dtype='object')
```

[47]:
```
b=df values
b
```

```
[47]: array([[165349.2, 136897.8, 471784.1, 'New York', 192261.83],
             [162597.7, 151377.59, 443898.53, 'California', 191792.06],
             [153441.51, 101145.55, 407934.54, 'Florida', 191050.39],
             [144372.41, 118671.85, 383199.62, 'New York', 182901.99],
             [142107.34, 91391.77, 366168.42, 'Florida', 166187.94],
             [131876.9, 99814.71, 362861.36, 'New York', 156991.12],
             [134615.46, 147198.87, 127716.82, 'California', 156122.51],
             [130298.13, 145530.06, 323876.68, 'Florida', 155752.6],
             [120542.52, 148718.95, 311613.29, 'New York', 152211.77],
             [123334.88, 108679.17, 304981.62, 'California', 149759.96],
             [101913.08, 110594.11, 229160.95, 'Florida', 146121.95],
             [100671.96, 91790.61, 249744.55, 'California', 144259.4],
             [93863.75, 127320.38, 249839.44, 'Florida', 141585.52],
             [91992.39, 135495.07, 252664.93, 'California', 134307.35],
             [119943.24, 156547.42, 256512.92, 'Florida', 132602.65],
             [114523.61, 122616.84, 261776.23, 'New York', 129917.04],
             [78013.11, 121597.55, 264346.06, 'California', 126992.93],
             [94657.16, 145077.58, 282574.31, 'New York', 125370.37],
             [91749.16, 114175.79, 294919.57, 'Florida', 124266.9],
             [86419.7, 153514.11, 0.0, 'New York', 122776.86],
             [76253.86, 113867.3, 298664.47, 'California', 118474.03],
             [78389.47, 153773.43, 299737.29, 'New York', 111313.02],
             [73994.56, 122782.75, 303319.26, 'Florida', 110352.25],
             [67532.53, 105751.03, 304768.73, 'Florida', 108733.99],
             [77044.01, 99281.34, 140574.81, 'New York', 108552.04],
             [64664.71, 139553.16, 137962.62, 'California', 107404.34],
             [75328.87, 144135.98, 134050.07, 'Florida', 105733.54],
             [72107.6, 127864.55, 353183.81, 'New York', 105008.31],
             [66051.52, 182645.56, 118148.2, 'Florida', 103282.38],
             [65605.48, 153032.06, 107138.38, 'New York', 101004.64],
             [61994.48, 115641.28, 91131.24, 'Florida', 99937.59],
             [61136.38, 152701.92, 88218.23, 'New York', 97483.56],
             [63408.86, 129219.61, 46085.25, 'California', 97427.84],
             [55493.95, 103057.49, 214634.81, 'Florida', 96778.92],
             [46426.07, 157693.92, 210797.67, 'California', 96712.8],
             [46014.02, 85047.44, 205517.64, 'New York', 96479.51],
             [28663.76, 127056.21, 201126.82, 'Florida', 90708.19],
             [44069.95, 51283.14, 197029.42, 'California', 89949.14],
             [20229.59, 65947.93, 185265.1, 'New York', 81229.06],
             [38558.51, 82982.09, 174999.3, 'California', 81005.76],
             [28754.33, 118546.05, 172795.67, 'California', 78239.91],
             [27892.92, 84710.77, 164470.71, 'Florida', 77798.83],
             [23640.93, 96189.63, 148001.11, 'California', 71498.49],
             [15505.73, 127382.3, 35534.17, 'New York', 69758.98],
             [22177.74, 154806.14, 28334.72, 'California', 65200.33],
```

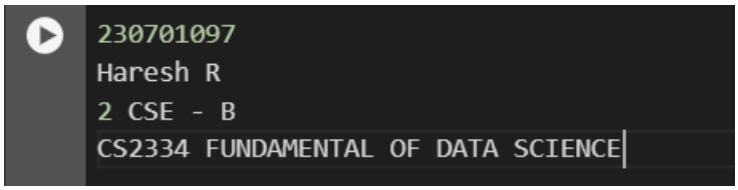
```
       [1000.23, 124153.04, 1903.93, 'New York', 64926.08],
       [1315.46, 115816.21, 297114.46, 'Florida', 49490.75],
       [0.0, 135426.92, 0.0, 'California', 42559.73],
       [542.05, 51743.15, 0.0, 'New York', 35673.41], [0.0,
       116983.8, 45173.06, 'California', 14681.4]],
       dtype=object)
```

[ ]:

[ ]:

[14]:
```
import seaborn as sns
%matplotlib inline
sns displot(array)
```

4.OUTLIER DETECTION



[2]:
```
array([76, 61, 80, 12, 8, 54, 41, 18, 98, 82, 5, 15, 14, 55, 67, 70])
```

[4]:
```
array mean()
```

[4]:
```
47.25
```

[6]:
```
np percentile(array,25)
```

[6]:
```
14.75
```

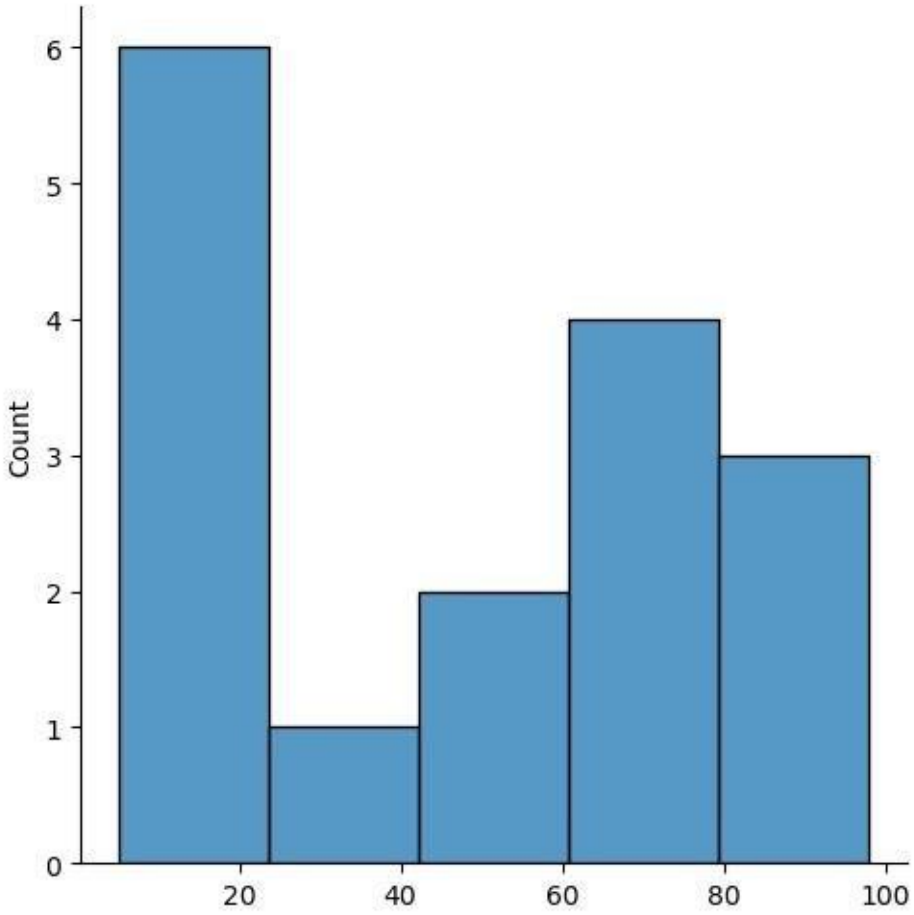
[8]:
```
np percentile(array,75)
```

[8]:
```
71.5
```

[12]:
```
#outliers detection
def outDetection(array):
    sorted(array)
    Q1,Q3=np percentile(array,[25,75])
    IQR=Q3-Q1
    lr=Q1-(1.5*IQR)
    ur=Q3+(1.5*IQR)
    return lr,ur
lr,ur=outDetection(array)
lr,ur
```

[12]:
```
(-70.375, 156.625)
```

[14]:
```
<seaborn.axisgrid.FacetGrid at 0x1d3957026f0>
```

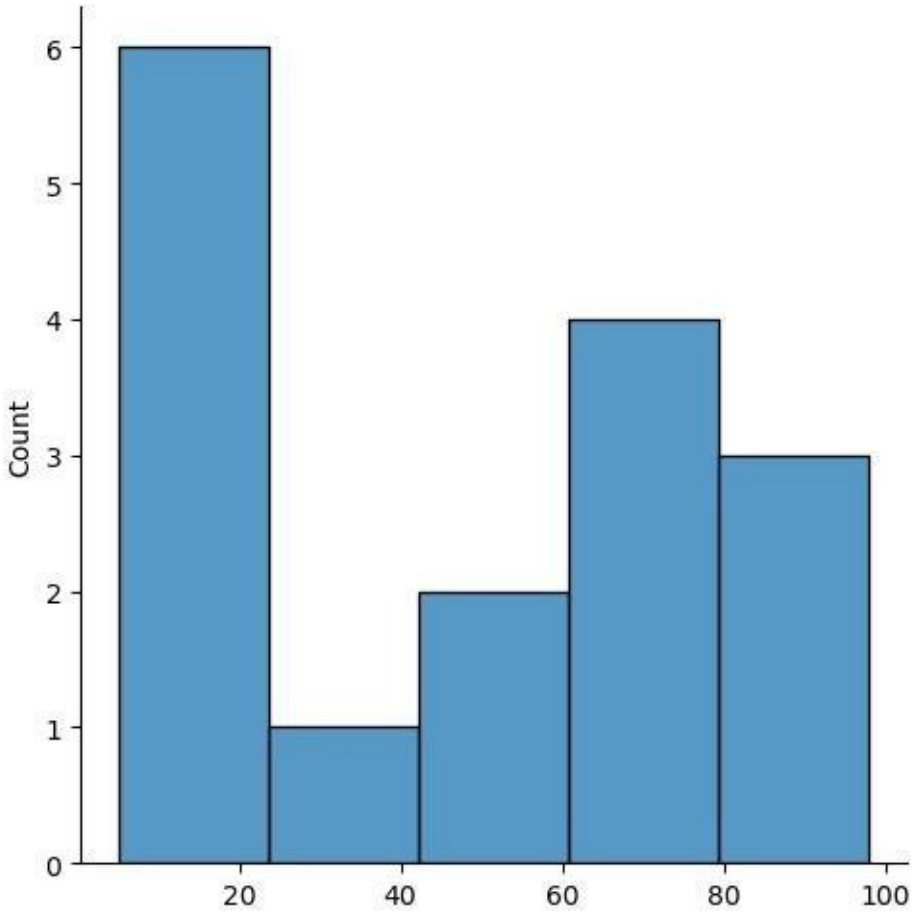


[16]:
```
new_array=array[(array>lr) & (array<ur)]
new_array
```

[16]:
```
array([76, 61, 80, 12, 8, 54, 41, 18, 98, 82, 5, 15, 14, 55, 67,
70])
```

[18]:
```
sns displot(new_array)
```

[18]:
```
<seaborn.axisgrid.FacetGrid at 0x1d390e4be30>
```



[20]:
```
lr1,ur1=outDetection(new_array)
lr1,ur
```

[20]: (-70.375, 156.625)

[25]:
```
final_array=new_array[(new_array>lr1) & (new_array<ur1)]
final_array
```

[25]: array([76, 61, 80, 12, 8, 54, 41, 18, 98, 82, 5, 15, 14, 55, 67, 70])

[ ]:

5 MISSING AND INAPPROPRIATE DATA

[ ]:
```
#EX NO:3
#MISSING AND INAPPROPRIATE DATA 20/08/2024
#ARITRA GUPTA
#230701033
#CSE A
```

[3]:
```
import numpy as np
import pandas as pd
df=pd read_csv("hotel_data_set.csv")
df
```

[3]:
```
    CustomerID Age_Group  Rating(1-5)      Hotel FoodPreference  Bill  \
0            1     20-25 4     Ibis  veg 1300
1            2     30-35 5 LemonTree      Non-Veg 2000
2            3     25-30 6    RedFox     Veg 1322
3            4 20-25 -1 LemonTree Veg 1234
4            5 35+ 3 Ibis Vegetarian 989
5            6 35+ 3 Ibys Non-Veg 1909
6            7 35+ 4 RedFox Vegetarian 1000
7            8 20-25 7 LemonTree Veg 2999
8            9 25-30 2 Ibis Non-Veg 3456
9            9     25-30 2    Ibis Non-Veg 3456
10          10 30-35 5      RedFox     non-Veg -6755

    NoOfPax EstimatedSalary Age_Group.1
0         2 40000 20-25
1         3 59000 30-35
2         2 30000 25-30
3         2 120000      20-25 4 2  45000 35+
5         2 122220      35+
6         -1 21122 35+
7         -10     345673     20-25
8         3 -99999      25-30
9         3 -99999      25-30
10        4 87777 30-35
```

[5]:
```
df duplicated()
```

[5]:
```
0    False
1    False
```

```
2    False
3    False
4    False
5     False
6     False
7     False
8     False
9     True
10    False
dtype: bool
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9
columns):
 # Column          Non-Null Count Dtype
--- ------         -------------- -----
0   CustomerID    11 non-null int64
1   Age_Group     11 non-null object 2 Rating(1-5) 11
    non-null int64
3    Hotel   11 non-null object
4    FoodPreference 11 non-null object
5   Bill 11 non-null int64 6 NoOfPax 11 non-null int64
7 EstimatedSalary 11 non-null int64 8
Age_Group.1 11 non-null object dtypes:
int64(5), object(4) memory usage:
924.0+ bytes
```

[9]:
```
df.drop_duplicates(inplace True)
df
```

[9]:
```
    CustomerID Age_Group Rating(1-5)Hotel FoodPreference Bill \
0           1     20-25         4       Ibis           veg 1300
1           2     30-35          5 LemonTree      Non-Veg 2000
2 3 25-30 6 RedFox Veg 1322
3 4 20-25 -1 LemonTree Veg 1234
4 5 35+ 3 Ibis Vegetarian 989
5     6     35+   3     Ibys  Non-Veg 1909 6    7     35+   4     RedFox
Vegetarian 1000
7           8     20-25 7 LemonTree      Veg 2999
8           9     25-30 2     Ibis  Non-Veg 3456
10         10     30-35          5     RedFox      non-Veg -6755
    NoOfPax EstimatedSalary Age_Group.1
0        2  40000 20-
```

```
                                  25
1         3  59000 30-
                                  35
2         2  30000 25-
                                  30
3         2  120000     20-
                                  25
4         2  45000 35+ 5       2     122220     35+
6         -1 21122 35+
7         -10      345673     20-
                                  25
8         3  -99999     25-
                                  30
10        4            87777      30-
                                  35
```

[11]:
```
len(df)
```

[11]: 10

[13]:
```
index=np array(list(range(0 len(df))))
df set_index(index,inplace=True)
index
```

[13]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

[15]:
```
df drop([ Age_Group.1 ],axis=1,inplace=True)
df
```

[15]:
```
  CustomerID Age_Group Rating(1-5) Hotel FoodPreference Bill
NoOfPax \
           0 1  20-25 4  Ibis veg 13001 2  30-35 5 LemonTree  Non-Veg
           2000                                                    2
           2 3  25-30 6  RedFox  Veg 1322                         2
           3 4  20-25 -1 LemonTree  Veg 1234                      2
           4 5  35+ 3  Ibis Vegetarian 989                       2
           5 6  35+ 3  Ibys Non-Veg 19096 7  35+ 4  RedFox  Vegetarian
           1000                                                    2


                     3

                     -1

           7         8     20-25 7 LemonTree     Veg 2999 -10
```

```
        8          9     25-30 2    Ibis Non-Veg 3456     3

        9          10    30-35 5    RedFox     non-Veg -6755
                          4

   EstimatedSalary
0           40000
1           59000
2           30000
3           120000
4           45000
5           122220 6 21122
7     345673
8     -99999 9 87777 [21]:
df.EstimatedSalary.fillna(round(df.Est
imatedSalary.mean()))
df.NoOfPax.fillna(round(df.NoOfPax.med
ian()))
df['Rating(1-5)'].fillna(round(df['Rating(1-
5)'].median()))

df.Bill.fillna(round(df.Bill.mean())) df
[21]: CustomerID Age_Group Rating(1-5) Hotel FoodPreference Bill NoOfPax
\

0          1     20-25 4    Ibis veg 1300 2

1          2     30-35 5 LemonTree     Non-Veg 2000     3

2          3     25-30 6    RedFox     Veg 1322 2

3          4     20-25 -1 LemonTree    Veg 1234 2

4          5     35+ 3      Ibis Vegetarian 989 2

5          6     35+ 3      Ibys Non-Veg 1909     2

6          7     35+ 4      RedFox     Vegetarian 1000 -1

7          8     20-25 7 LemonTree     Veg 2999 -10

8          9     25-30 2    Ibis Non-Veg 3456     3

9          10 30-35 5  RedFox     non-Veg -6755 4

   EstimatedSalary
0           40000
1           59000
```

```
2           30000
3           120000 4 45000
5     122220 6
 21122
7           345673
8           -99999
9           87777
```

[23]:
```
df Age_Group unique()
```

[23]:
```
array(['20-25', '30-35', '25-30', '35+'], dtype=object)
```

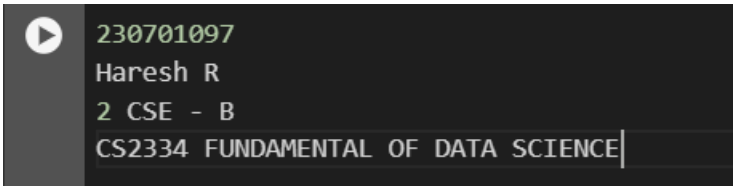
[25]:
```
df Hotel unique()
```

[25]:
```
array(['Ibis', 'LemonTree', 'RedFox', 'Ibys'],
dtype=object)
```

[29]:
```
df Hotel replace([ Ibys ]  Ibis )
```

[29]:
```
         Ibis
0
1        LemonTree
2        RedFox
3        LemonTree
4        Ibis
5        Ibis
6        RedFox
7        LemonTree
8        Ibis
9        RedFox
      Name: Hotel, dtype: object
```

6.DATA PRE PROCESSING




```
1     2 230701002 AAKASH V 44
2     3 230701003 ABHILASH G R    44
3     4 230701004 ABHINAYA LAKSHMI S 48
4     5 230701005 ABHISHEK ROBIN S A 16
      .. …        …                    … …
65   66 230701504 KAAVIYA R 16
66   67 230701507 MAGESH VASAN M 38
67   68 230701510 SARANYA M 44
68   69 230701514 GANESHAN M 14
69   70 230701521 JABARAJ E 9
     [70 rows x 4 columns]
```

[36]:
```
df ınfo()
```

```
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 70 entries, 0 to 69
     Data columns (total 4
     columns):
      # Column Non-Null Count Dtype
     --- ------ -------------- -----
0    SNO     70 non-null      int64
1    RNO     70 non-null      int64
2    NAME 70 non-null   object
3    MARKS 70 non-null  int64
      dtypes: int64(3), object(1) memory
     usage: 2.3+ KB
```

[40]:
```
df MARKS mode()
```

```
[40]: 0   40
      Name: MARKS, dtype: int64
[42]: df MARKS mode()[0]

[42]: 40
[44]: type(df MARKS mode())

[44]: pandas.core.series.Series
[48]: df MARKS fillna(df MARKS mode()[0])

[48]: 0    40
      1    44
      2    44
      3    48
      4    16
..
65    16
66    38
67    44
68    14
69    9
Name: MARKS, Length: 70, dtype: int64
[50]: df MARKS fillna(df MARKS median())

[50]: 0 40
1    44
2    44
3    48
4    16
..
65    16
66    38
67    44
```

```
68    14
69    9
Name: MARKS, Length: 70, dtype: int64
```

[52]:

```
df
```

[52]:

	SNO	RNO	NAME	MARKS
0	1	230701001	AADITYA PARTHA SARATHY	40
1	2	230701002	AAKASH V	44
2	3	230701003	ABHILASH G R	44
3	4	230701004	ABHINAYA LAKSHMI S	48
4	5	230701005	ABHISHEK ROBIN S A	16
..	…	…	…	…
65	66	230701504	KAAVIYA R	16
66	67	230701507	MAGESH VASAN M	38
67	68	230701510	SARANYA M	44
68	69	230701514	GANESHAN M	14
69	70	230701521	JABARAJ E	9

[70 rows x 4 columns]

[54]:

```
pd get_dummies(df NAME)
```

[54]:

	AADITYA PARTHA SARATHY	AAKASH V	ABHILASH G R	ABHINAYA LAKSHMI S	\
0	True	False	False	False	
1	False	True	False	False	
2	False	False	True	False	
3	False	False	False	True	
4	False	False	False	False	
..	…	…	…	…	
65	False	False	False	False	
66	False	False	False	False	

```
67                False     False False      False

68                False     False False      False

69                False     False False      False

ABHISHEK ROBIN S A ABHISHEK S ABINAV S T ABIRAMI K ABISHEK I \
0              False      False      False     False     False

1              False      False      False     False     False

2              False      False      False     False     False

3              False      False      False     False     False

4               True      False      False     False     False

..               …         …          …         …         …

65             False      False      False     False     False

66             False      False      False     False     False

67             False      False      False     False     False

68             False      False      False     False     False

69             False      False      False     False     False
   ABISHEK NATARAJAN … DARSHAN S DAYANITHI V DEEPA S DEEPAK K \
0             False …     False       False   False    False
1             False …     False       False   False    False
2             False …     False       False   False    False
3             False …     False       False   False    False
4             False …     False       False   False    False
..              … …         …           …       …        …

65            False …     False       False   False    False
66            False …     False       False   False    False
67            False …     False       False   False    False
68            False …     False       False   False    False
69            False …     False       False   False    False
GANESHAN M H AKSHITHAA JABARAJ E KAAVIYA R MAGESH VASAN M SARANYA M
 0 False False False False False False

1       False    False False      False      False False

2       False    False False      False      False False
3       False    False False      False      False False
```

```
4       False    False False       False       False False

..        …         …       …        …            …       …

      65      False    False False       True        False False

      66      False    False False       False       True  False 67

                  False False False       False       False True

68       True    False False       False       False False

69       False   False True  False       False False

[70 rows x 69 columns]
```

[56]:
```
df ınfo()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 4
columns):
 #     Column Non-Null Count
Dtype --- ------ -------------- -----
  0 SNO 70 non-null          int64
  1 RNO 70 non-null          int64
  2 NAME 70 non-null         object
  3 MARKS 70 non-null memory int64 dtypes: int64(3), object(1)
 usage: 2.3+ KB
```

[ ]:

7.EDA

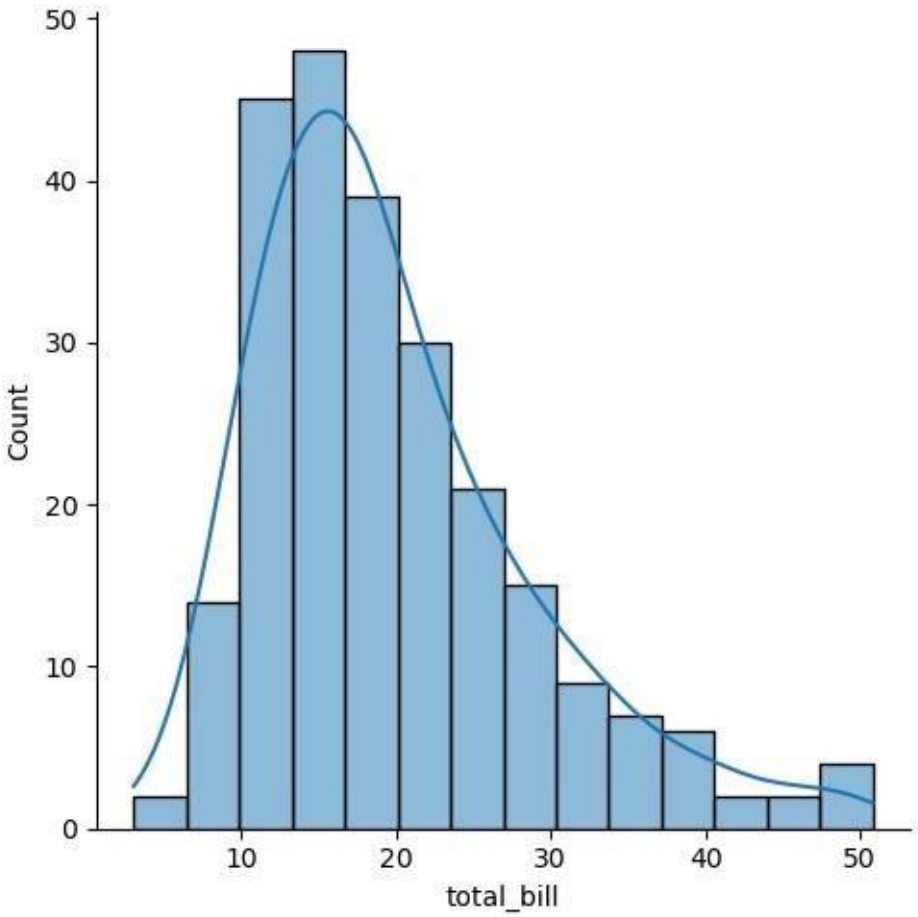
```
[63]: import seaborn as sns
      import pandas as pd
      import numpy as np
      import matplotlib pyplot as plt
      %matplotlib inline
      tips=sns load_dataset( tips )
      tips head()
```

```
[63]:    total_bill   tip     sex smoker  day    time  size
      0       16.99  1.01  Female     No  Sun  Dinner     2
      1       10.34  1.66    Male     No Sun Dinner     3
      2       21.01  3.50    Male     No Sun Dinner     3
```

```
3     23.68 3.31 Male  No Sun Dinner    2 4
24.59 3.61 Female      No Sun Dinner    4
```

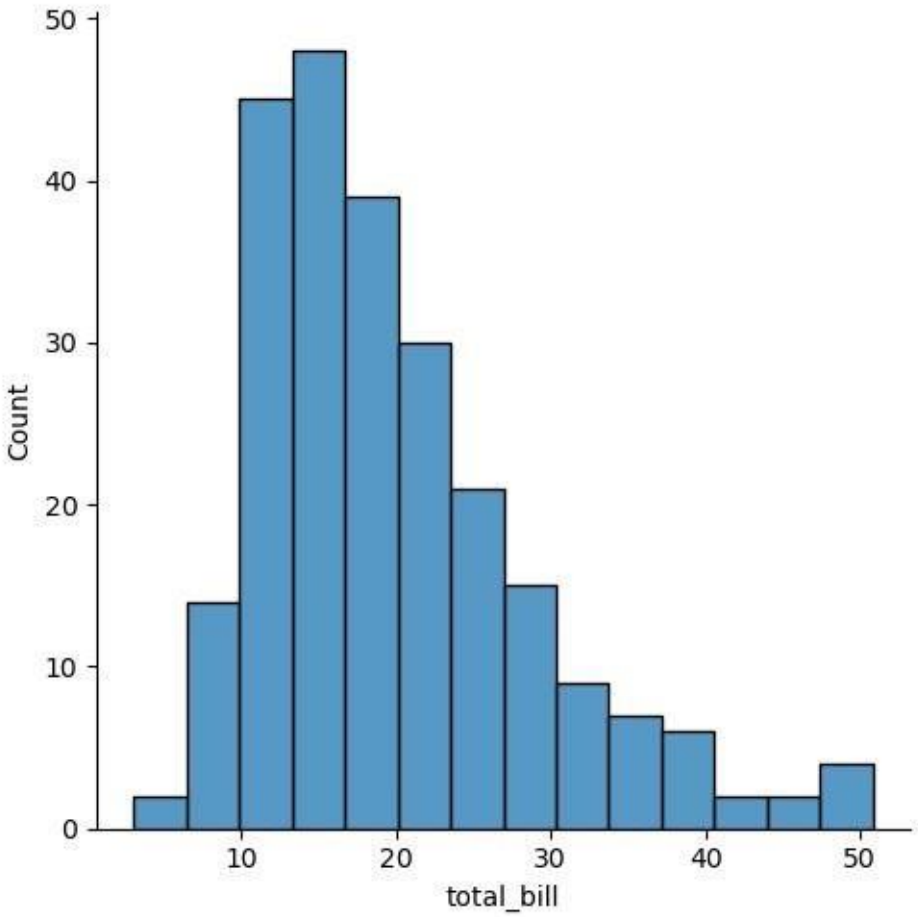
```
[65]: sns displot(tips total_bill,kde True)
```

```
[65]: <seaborn.axisgrid.FacetGrid at 0x229166f4b00>
```



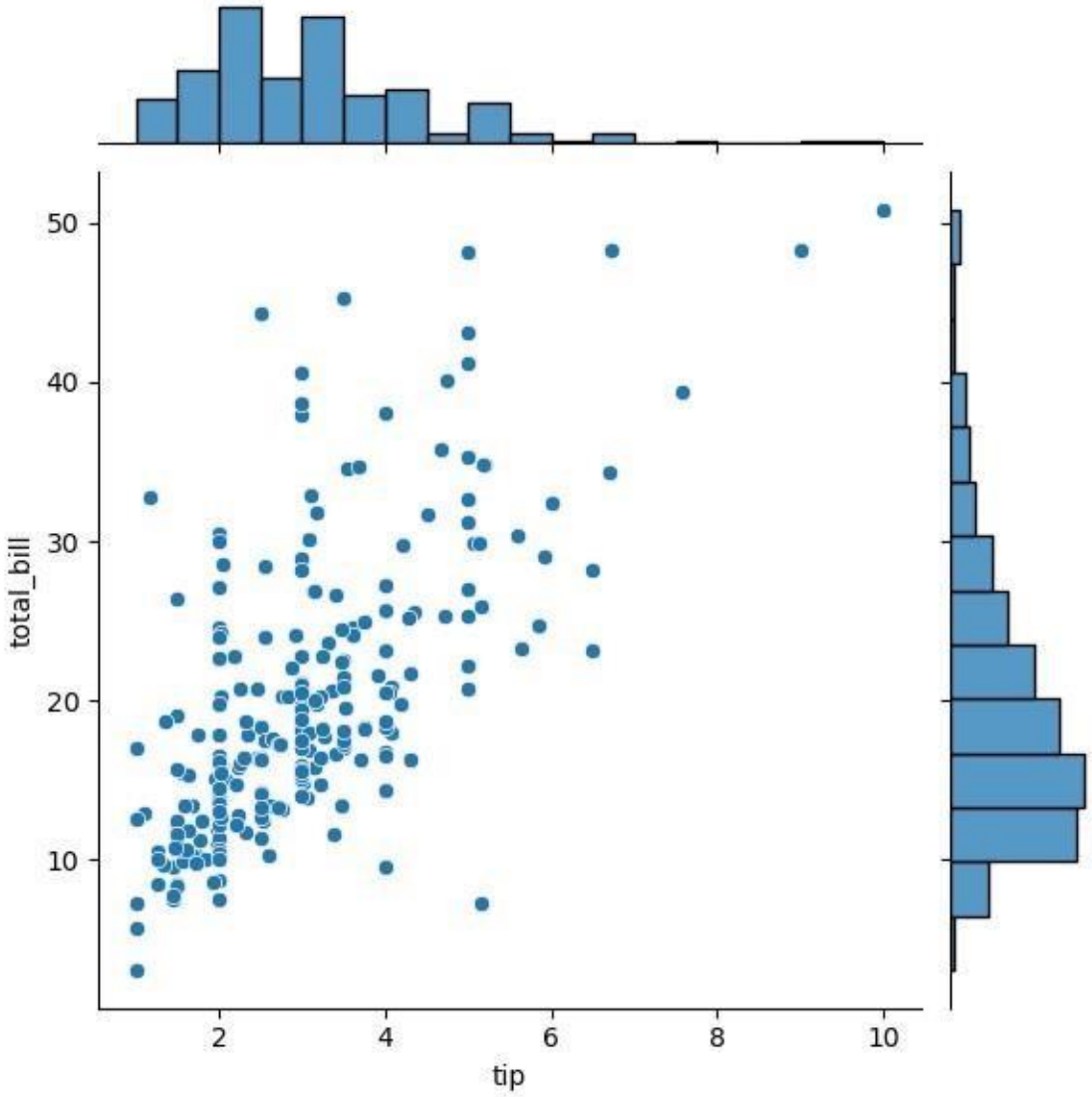
[67]:
```
sns displot(tips total_bill,kde False)
```

[67]: <seaborn.axisgrid.FacetGrid at 0x229183d7b00>



[69]:
```
sns jointplot(x tips tip,y tips total_bill)
```

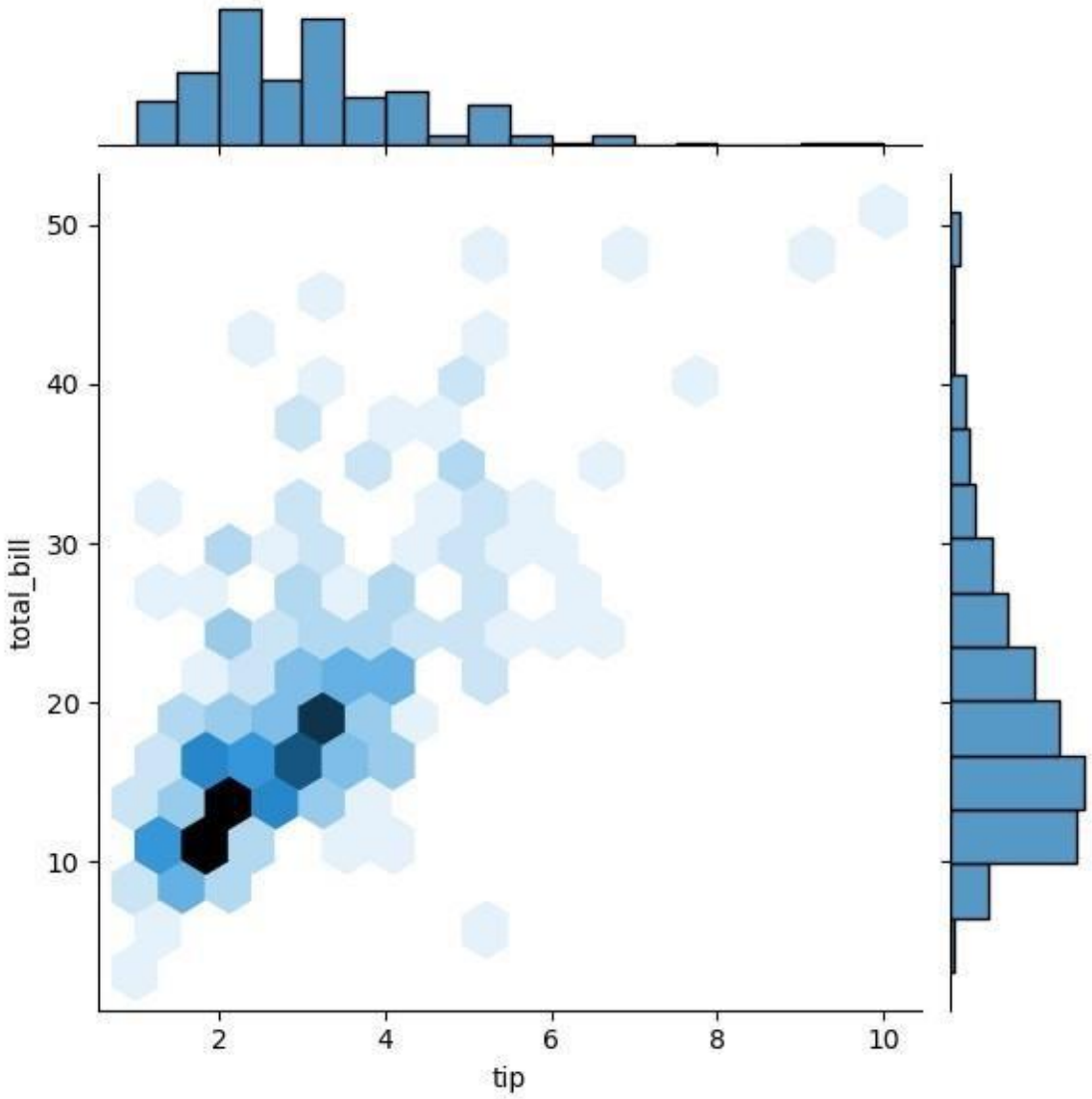
[69]:
```
<seaborn.axisgrid.JointGrid at 0x22911d47650>
```



[71]:

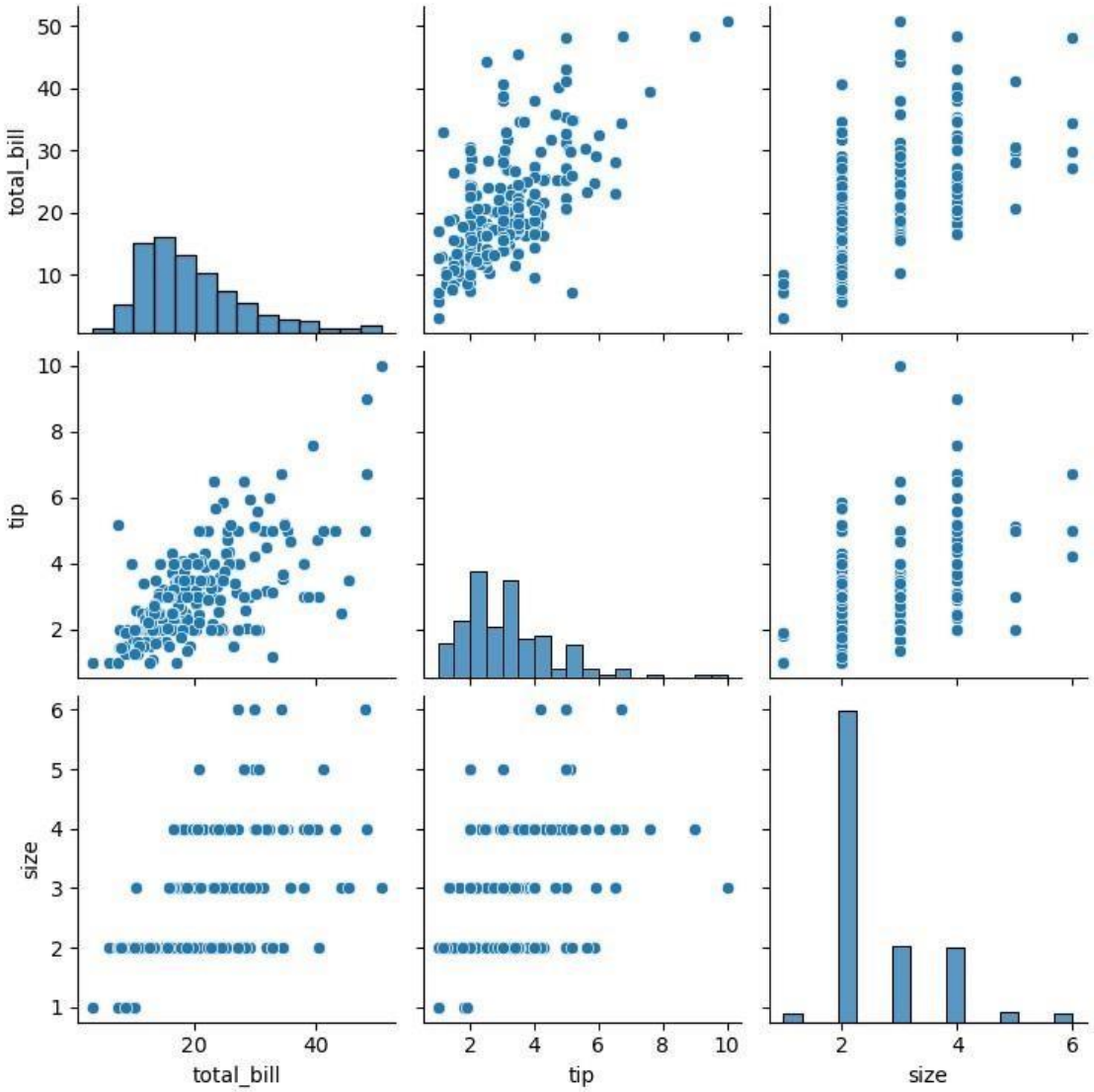
```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex"
)
```

[71]: <seaborn.axisgrid.JointGrid at 0x2291850c6e0>



[73]:
```
sns pairplot(tips)
```

[73]:
```
<seaborn.axisgrid.PairGrid at 0x229184b9e80>
```



[75]:
```
tips time.value_counts()
```

[75]:
```
time Dinner
      176
     Lunch     68
 Name: count, dtype: int64
```

[ ]:

8 RANDOM SAMPLING

[182]:
```
import numpy as np
import matplotlib pyplot as plt
```

[184]:
```
population_mean = 50
population_std = 10
population_size = 100000
population = np random normal(population_mean, population_std, population_size)
```

[186]:
```
sample_sizes = [30  50, 100] # different sample sizes to consider
num_samples = 1000 # number of samples for each sample size
sample_means = {}
for size in sample_sizes:
    sample_means[size] = []
```

[188]:
```
for _ in range(num_samples):
    sample = np random choice(population, size=size, replace=False)
    sample_means[size] append(np mean(sample))
```

[189]:
```
plt figure(figsize=(12, 8))
```

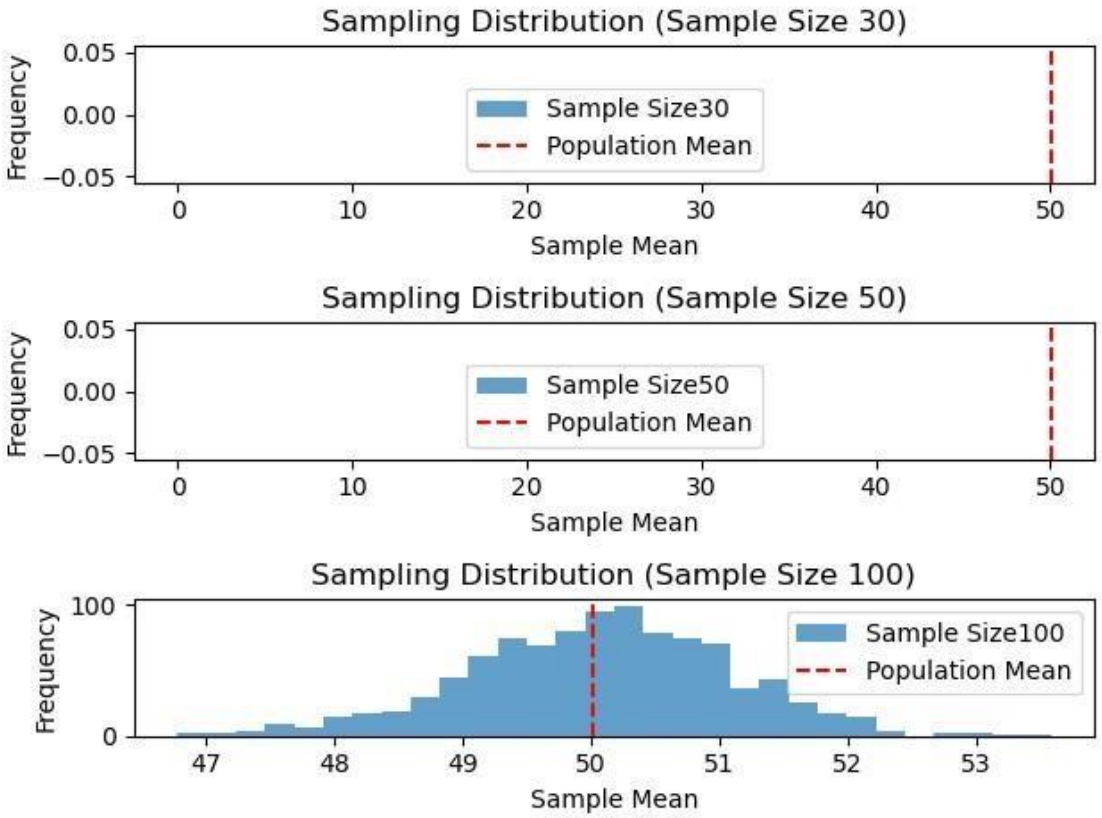
[189]:
```
<Figure size 1200x800 with 0 Axes >
```

<Figure size 1200x800 with 0 Axes>

[190]:
```
for i, size in enumerate(sample_sizes):
    plt.subplot(len(sample_sizes), 1, i+1)
    plt.hist(sample_means[size], bins=30, alpha=0.7, label=f'Sample
    Size{size}') plt.axvline(np.mean(population), color='red',
    linestyle='dashed',
↪linewidth=1.5,label='Population Mean') plt.title(f'Sampling
    Distribution (Sample Size
    {size})') plt.xlabel('Sample Mean') plt.ylabel('Frequency')
```

[ ]:

```
    plt legend()
plt tight_layout()
plt show()
```



[ ]:

[ ]:

[ ]:

9 Z TEST

```
#EX NO:7
#Z-TEST 10/09/2024
#ARITRA GUPTA
#230701033
#CSE A
```

[236]:
```
import numpy as np
import scipy stats as stats
```

[238]:
```
sample_data = np array([152, 148, 151, 149, 147, 153, 150  148, 152,
149 151  150  149  152, 151, 148, 150, 152, 149, 150,148, 153  151,
150  149  152  148  151, 150  153])
```

[240]:
```
population_mean = 150
sample_mean = np mean(sample_data)
sample_std = np std(sample_data, ddof=1)
```

[242]:
```
n = len(sample_data)
z_statistic = (sample_mean - population_mean) / (sample_std /
np sqrt(n))
p_value = 2 * (1 - stats norm cdf(np abs(z_statistic)))
```

[244]:
```
print(f"Sample Mean: {sample_mean:.2f}")
print(f"Z-Statistic: {z_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")
```

```
Sample Mean: 150.20
Z-Statistic: 0.6406
P-Value: 0.5218
```

[246]:
```
alpha = 0.05 if
p_value < alpha:
        print("Reject the null hypothesis: The average weight is
                                                significantly␣
 ↪different from 150 grams.")
else:
   print("Fail to reject the null hypothesis: There is no
    significant
  ↪difference in average weight from 150 grams.")
```

```
Fail to reject the null hypothesis: There is no significant
difference in average weight from 150 grams.
```

[ ]:

[ ]:

[ ]:

10 T TEST

```
[262]: import numpy as np
       import scipy stats as stats

[264]: np random seed(42)
       sample_size = 25
       sample_data = np random normal(loc=102, scale=15, size=sample_size)

[266]: population_mean = 100
       sample_mean = np mean(sample_data)
       sample_std = np std(sample_data, ddof=1)

[268]: n = len(sample_data)
       t_statistic, p_value = stats ttest_1samp(sample_data,population_mean)

[270]: print(f"quot;Sample Mean: {sample_mean:.2f}")
       print(f"T-Statistic: {t_statistic:.4f}")
       print(f"P-Value: {p_value:.4f}")

       quot;Sample Mean: 99.55
       T-Statistic: -0.1577
       P-Value: 0.8760

[272]: alpha = 0.05 if
       p_value < alpha:
               print("Reject the null hypothesis: The average IQ SCORE is
                                                          significantly␣
        ↪different from 100.")
       else:
          print("Fail to reject the null hypothesis: There is no
            significant
          ↪difference in average of IQ Score from 100.")
```

```
Fail to reject the null hypothesis: There is no significant
```

```
difference in average of IQ Score from 100.
```

[

]:

[ ]:

11 ANOVA TEST

```
#EX NO:9
#ANOVA-TEST 08/10/2024
#ARITRA GUPTA
#230701033
#CSE A
```

[302]:
```
import numpy as np
import scipy stats as stats
```

[304]:
```
np random seed(42)
n_plants = 25
growth_A = np random normal(loc=10  scale=2, size=n_plants)
growth_B = np random normal(loc=12  scale=3, size=n_plants)
growth_C = np random normal(loc=15  scale=2.5, size=n_plants)
```

[306]:
```
all_data = np concatenate([growth_A, growth_B, growth_C])
treatment_labels = [ A ] * n_plants + [ B ] * n_plants + [ C ] * n_plants
```

[308]:
```
f_statistic, p_value = stats f_oneway(growth_A, growth_B, growth_C)
```

[310]:
```
print("Treatment A Mean Growth:&amp;quot;, np.mean(growth_A) ")
print("Treatment B Mean Growth:&amp;quot;, np.mean(growth_B) ")
print("Treatment C Mean Growth:&amp;quot;, np.mean(growth_C) ")
print()
print(f"F-Statistic: {f_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")
```

```
Treatment A Mean Growth:&amp;quot;, np.mean(growth_A)
Treatment B Mean Growth:&amp;quot;, np.mean(growth_B)
Treatment C Mean Growth:&amp;quot;, np.mean(growth_C)

F-Statistic: 36.1214 P-
Value: 0.0000
```

[312]:
```
 alpha = 0.05 if
p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference
                                                                    in
    ↪mean growth rates among the three treatments.")
else:
    print("Fail to reject the null hypothesis: There is no
    significant
    ↪difference in mean growth rates among the three treatments.")
```

[ ]:
Reject the null hypothesis: There is a significant difference in mean growth rates among the three treatments.

[314]:
```
if p_value < alpha: from statsmodels.stats.multicomp import pairwise_tukeyhsd tukey_results = pairwise_tukeyhsd(all_data, treatment_labels, alpha=0.05) print("\nTukey's HSD Post-hoc Test:") print(tukey_results)
```

```
Tukey's HSD Post-hoc Test:
Multiple Comparison of Means - Tukey HSD,
FWER=0.05
=============================================
======= group1  group2  meandiff  p-adj  lower
upper reject
 -----------------------------------------------------
A     B 1.4647 0.0877 -0.1683 3.0977 False
   A      C 5.5923      0.0 3.9593 7.2252     True
   B      C 4.1276      0.0 2.4946 5.7605     True
     ---------------------------------------------------- [
```

]:

12 FEATURE SCALING

```
#EX NO:10
#FEATURE SCALING 22/10/2024
#ARITRA GUPTA
#230701033
#CSE A
```

[84]:
```
import numpy as np
import pandas as pd
df=pd read_csv( 2_datasetExample.csv )
df
```

[84]:

	SNO	RNO	NAME	MARKS
0	1	230701001	AADITYA PARTHA SARATHY	40
1	2	230701002	AAKASH V	44
2	3	230701003	ABHILASH G R	44
3	4	230701004	ABHINAYA LAKSHMI S	48
4	5	230701005	ABHISHEK ROBIN S A	16

```
..  …       …                    … …
65   66 230701504 KAAVIYA R 16
66   67 230701507 MAGESH VASAN M 38
67   68 230701510 SARANYA M 44
68   69 230701514 GANESHAN M 14
69   70 230701521 JABARAJ E 9

[70 rows x 4 columns]
```

[86]:
```
df head()
```

[86]:
```
SNO          RNO                   NAME MARKS
0    1 230701001 AADITYA PARTHA SARATHY     40
1    2 230701002 AAKASH V 44
2    3 230701003 ABHILASH G R    44
3    4 230701004 ABHINAYA LAKSHMI S 48
4    5 230701005 ABHISHEK ROBIN S A 16
```

[94]:
```
df MARKS fillna(df MARKS mode()[0])
features=df ıloc[:,: 1] values
df
```

[94]:
```
    SNO       RNO                   NAME MARKS
0    1 230701001 AADITYA PARTHA SARATHY     40
1    2 230701002 AAKASH V 44
2    3 230701003 ABHILASH G R    44
3    4 230701004 ABHINAYA LAKSHMI S 48
4    5 230701005 ABHISHEK ROBIN S A 16
..  …       …                    … …
65   66 230701504 KAAVIYA R 16
66   67 230701507 MAGESH VASAN M 38
67   68 230701510 SARANYA M 44
68   69 230701514 GANESHAN M 14
69   70 230701521 JABARAJ E 9
[70 rows x 4 columns]
```

[ ]:

[98]:
```
label=df.iloc[:,-1].values from sklearn.impute import
    SimpleImputer
age=SimpleImputer(strategy="mean",missing_values=np
.nan)
Salary=SimpleImputer(strategy="mean",missing_values
=np.nan) age.fit(features[:,[1]])
```

[98]:
```
SimpleImputer()
```

[106]:
```
SimpleImputer()
```

[106]:
```
SimpleImputer()
```

[114]:
```
features[:,[1]]=age.transform(features[:,[1]])
features
```

[114]:
```
array([[1, 230701001.0, 'AADITYA PARTHA SARATHY'],
       [2, 230701002.0, 'AAKASH V'],
       [3, 230701003.0, 'ABHILASH G R'],
       [4, 230701004.0, 'ABHINAYA LAKSHMI S'],
       [5, 230701005.0, 'ABHISHEK ROBIN S A'],
       [6, 230701006.0, 'ABHISHEK S'],
       [7, 230701007.0, 'ABINAV S T'],
       [8, 230701008.0, 'ABIRAMI K'],
       [9, 230701009.0, 'ABISHEK I'],
       [10, 230701010.0, 'ABISHEK NATARAJAN'],
       [11, 230701011.0, 'ABOORVAN SHANMUGAPRIYA BABU'],
       [12, 230701012.0, 'ADHAVAN BALAJI N M'],
       [13, 230701013.0, 'ADITHYA J'],
       [14, 230701014.0, 'ADITHYAA SURESH'],
       [15, 230701015.0, 'AISHWARYA A'],
       [16, 230701016.0, 'AISHWARYA M'],
       [17, 230701017.0, 'AJAY SRINIVAS R'],
       [18, 230701018.0, 'AJEESH R R'],
```

[19, 230701019.0, 'AKASH N'],
[20, 230701020.0, 'AKILESH PRASAD I K'],
[21, 230701021.0, 'AKSHAY KUMAR S'],
[22, 230701022.0, 'AKSHAY VENKAT KRISHNA'],
[23, 230701023.0, 'AKSHAYA BALAJI NITHYANANDAN'],
[24, 230701024.0, 'AKSHAYA SRI S'],
[25, 230701025.0, 'H AKSHITHAA'],
[26, 230701026.0, 'ALFRED SAM D'],
[27, 230701027.0, 'AMIRTHAVARSHINI R U'],
[28, 230701028.0, 'ANIRUDH C'],
[29, 230701029.0, 'ANIRUDH S'],
[30, 230701030.0, 'ANU S'],
[31, 230701031.0, 'ARAVINDAN S G'],
[32, 230701032.0, 'ARAVINTHAA S'],
[33, 230701033.0, 'ARITRA GUPTA'],
[34, 230701034.0, 'ARUL JOTHI P'],
[35, 230701035.0, 'ARUL RAJAN S'],
[36, 230701036.0, 'ARUN M C'],
[37, 230701037.0, 'ARUN PRAKASH M'],
[38, 230701038.0, 'ARVIND RAVI'],
[39, 230701039.0, 'ARYA SUBANANTH R K'],
[40, 230701040.0, 'ARYAN SAI VENKAT M'],
[41, 230701041.0, 'ASHISH P SHAJI'],
[42, 230701042.0, 'ASHNA V'],
[43, 230701043.0, 'ASHWIN KUMAR A P'],
[44, 230701044.0, 'ASWINKUMAR J'],
[45, 230701045.0, 'ATCHAYA S'],
[46, 230701046.0, 'ATHIENA RACHEL J'],
[47, 230701047.0, 'ATHIRA D R'],
[48, 230701048.0, 'AWINTHIKA SANTHANAM'],
[49, 230701049.0, 'BALAJI C'],
[50, 230701051.0, 'BERNIEO FATIM A'], [51,
230701052.0, 'BHARATH B'],
[52, 230701053.0, 'BHARATH KUMAR M'], [53,
230701054.0, 'BHARRATH K'],
[54, 230701055.0, 'BHUVANESHWARI K'],
[55, 230701056.0, 'BOOTHALINGESH N'],
[56, 230701057.0, 'BOSEBALA T'],
[57, 230701058.0, 'BRIJITH MANIKANDAN P'],
[58, 230701059.0, 'CHANDNI M N'],
[59, 230701060.0, 'DANIEL LEVE MANICKAM D A'],
[60, 230701061.0, 'DARSHAN M'],
[61, 230701062.0, 'DARSHAN M'],
[62, 230701063.0, 'DARSHAN S'],

```
        [63, 230701064.0, 'DAYANITHI V'],
        [64, 230701065.0, 'DEEPA S'],
        [65, 230701066.0, 'DEEPAK K'],
```



```
        [66, 230701504.0, 'KAAVIYA R'],
        [67, 230701507.0, 'MAGESH VASAN M'],
        [68, 230701510.0, 'SARANYA M'],
        [69, 230701514.0, 'GANESHAN M'],
        [70, 230701521.0, 'JABARAJ E']], dtype=object)
```

[116]:
```
from sklearn preprocessing import OneHotEncoder
oh = OneHotEncoder(sparse_output=False)
Country=oh fit_transform(features[:,[0]])
Country
```

[116]:
```
array([[1., 0., 0., …, 0., 0., 0.],
       [0., 1., 0., …, 0., 0., 0.],
       [0., 0., 1., …, 0., 0., 0.],
       …,
       [0., 0., 0., …, 1., 0., 0.],
       [0., 0., 0., …, 0., 1., 0.],
       [0., 0., 0., …, 0., 0., 1.]])
```

[118]:
```
final_set np concatenate((Country,features[:,[1 2]]) axis=1)
final_set
```

[118]:
```
array([[1.0, 0.0, 0.0, …, 0.0, 230701001.0, 'AADITYA PARTHA
SARATHY'],
       [0.0, 1.0, 0.0, …, 0.0, 230701002.0, 'AAKASH V'],
       [0.0, 0.0, 1.0, …, 0.0, 230701003.0, 'ABHILASH G R'],
       …,
       [0.0, 0.0, 0.0, …, 0.0, 230701510.0, 'SARANYA M'],
       [0.0, 0.0, 0.0, …, 0.0, 230701514.0, 'GANESHAN M'],
       [0.0, 0.0, 0.0, …, 1.0, 230701521.0, 'JABARAJ E']],
       dtype=object)
```

[ ]:

[ ]:

13 LINEAR REGRESSION

```
230701097
Haresh R
2 CSE - B
CS2334 FUNDAMENTAL OF DATA SCIENCE
```

[4]:
```
import numpy as np
import pandas as pd
df=pd read_csv( 4i_salary_data.csv )
df
```

[4]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0

```
26             9.5 116969.0
27             9.6 112635.0 28 10.3
               122391.0
29            10.5 121872.0
```

[6]:
```
df info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 # Column          Non-Null Count Dtype
--- ------ -------------- ----0
 YearsExperience 30 non-null float64
1 Salary    30 non-null float64 dtypes:
float64(2) memory usage: 612.0
bytes
```

[8]:
```
df dropna(inplace True)
```

[10]:
```
df describe()
```

[10]:
```
 YearsExperience Salary count
30.000000      30.000000      mean
5.313333     76003.000000      std
2.837888     27414.429785      min
1.100000     37731.000000      25%
3.200000 56720.750000
50%           4.700000 65237.000000
75%          7.700000 100544.750000
max          10.500000  122391.000000
```

[12]:
```
features=df iloc[:,[0]] values
label=df iloc[:,[1]] values
```

[14]:
```
from sklearn model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.
 ↪2,random_state=0)
```

[16]:
```
from sklearn linear_model import LinearRegression
model=LinearRegression()
model fit(x_train,y_train)
```

[16]:
```
LinearRegression()
```

[18]:
```
model score(x_train,y_train)
```

[18]:
```
 0.9411949620562126
```

[20]:
```
model score(x_test,y_test)
```

[20]:
```
0.988169515729126
```

[22]:
```
import pickle
pickle.dump(model,open( SalaryPred.model , wb ))
```

[24]:
```
model=pickle.load(open( SalaryPred.model , rb ))
```

[26]:
```
yr_of_exp=float(input("Enter Years of Experience: "))
yr_of_exp_NP=np array([[yr_of_exp]])
Salary=model predict(yr_of_exp_NP)
```

```
Enter Years of Experience: 70
```

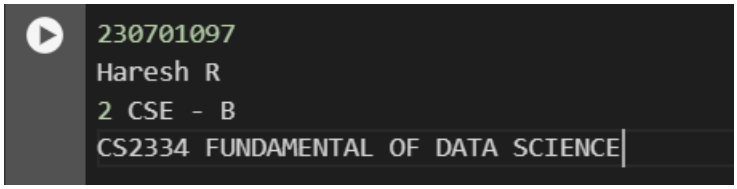
"

[30]:
```
print("Estimated Salary for {} years of experience is {}:

.
    ↪format(yr_of_exp,Salary))
```

```
Estimated Salary for 70.0 years of experience is
                                    [[678660.35802167]]:
```

[ ]:

14 LOGISTIC REGRESSION



[127]:
```
import numpy as np
import pandas as pd
df=pd read_csv( 4ii_Social_Network_Ads.csv )
df
```

[127]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0

```
4 15804002 Male    19          76000        0 0  15624510 Male 19
                                                 19000 0

1   15810944 Male 35        20000 0

2   15668575 Female 26           43000 0

3   15603246 Female 27           57000 0

4   15804002 Male 19        76000 0

..       …      … …              …        …

395 15691863 Female 46           41000 1

396 15706071 Male 51        23000 1

397 15654296 Female 50           20000 1

398 15755018 Male 36        33000 0

399 15594041 Female 49           36000 1

[400 rows x 5 columns]
```

[129]:
```
df head()
```

[129]: User ID Gender Age EstimatedSalary Purchased

[131]:
```
features=df ılloc[:,[2 3]] values
label df ılloc[:,4] values
features
```

```
  [131]: array([[ 19,
19000], [ 35, 20000],
       [    26, 43000],
       [    27, 57000],
       [    19, 76000],
       [    27, 58000],
       [    27, 84000],
       [   32, 150000],
       [    25, 33000],
       [    35, 65000],

       [
       [
       [
       [
       [
```

```
[   26, 80000],
[   26, 52000],
[   20, 86000],
[   32, 18000],
[   18, 82000],
[   29, 80000],
[   47, 25000],
[   45, 26000],
[   46, 28000],
[   48, 29000],
[   45, 22000],
[   47, 49000],
[   48, 41000],
[   45, 22000],
[   46, 23000],
[   47, 20000],
[   49, 28000],
[   47, 30000],
[   29, 43000],
[   31, 18000],
[   31, 74000],
[  27, 137000],
[   21, 16000],
[   28, 44000],
[   27, 90000],
[   35, 27000],

[
[
[
[
[
[
[
[
[   33, 28000],
[   30, 49000],
[   26, 72000],

[
[
[
[
[
```

```
[   27, 31000],
[   27, 17000],
[   33, 51000],
    35, 108000],
    30, 15000],
    28, 84000],
    23, 20000],
    25, 79000],
    27, 54000],
    30, 135000],
    31, 89000],
    24, 32000],
    18, 44000],
    29, 83000],
    35, 23000],
    27, 58000],
[   24, 55000],
[   23, 48000],
[   28, 79000],
[   22, 18000],
[   32, 117000],
[   27, 20000],
[   25, 87000],
[   23, 66000],
[   32, 120000],
[   59, 83000],
[   24, 58000],
[   24, 19000],
[   23, 82000],
[   22, 63000],
[   31, 68000],
[   25, 80000],
```

```
[
[
[
[
[
```

```
[
[
[
[
[

    24, 27000],
    20, 23000],
   33, 113000],
    32, 18000],
   34, 112000],
[   18, 52000],
[   22, 27000],
[   28, 87000],
[   26, 17000],
[   30, 80000],
[   39, 42000],
[   20, 49000],
[   35, 88000],
[   30, 62000],
[  31, 118000],
[   24, 55000],
[   28, 85000],
[   26, 81000],
    35, 50000],
    22, 81000],
    30, 116000],
    26, 15000],
    29, 28000],

[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
     29, 83000],
     35, 44000],
     35, 25000],
     28, 123000],
     35, 73000],
     28, 37000],
     27, 88000],
     28, 59000],
[    32, 86000],
[   33, 149000],
[    19, 21000],
[    21, 72000],
[    26, 35000],
[    27, 89000],
[    26, 86000],
[    38, 80000],
[    39, 71000],
[    37, 71000],
[    38, 61000],
[    37, 55000],
[    42, 80000],
[    40, 57000],
[    35, 75000],
[    36, 52000],
[    40, 59000],
[    41, 59000],
[    36, 75000],
[    37, 72000],
[    40, 75000],

[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   35, 53000],
[   41, 51000],
[   39, 61000],

[   42, 65000],
[   26, 32000],
[   30, 17000],
[   26, 84000],
[   31, 58000],
[   33, 31000],
[   30, 87000],
[   21, 68000],
[   28, 55000],
[   23, 63000],
    20, 82000],
    30, 107000],
    28, 59000],
    19, 25000],
    19, 85000],
    18, 68000],
    35, 59000],
    30, 89000],
    34, 25000],
    24, 89000],
    27, 96000],
    41, 30000],
    29, 61000],
[   20, 74000],
[   26, 15000],
[   41, 45000],

[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   31, 76000],
[   36, 50000],
[   40, 47000],
[   31, 15000],
[   46, 59000],
[   29, 75000],

[   26, 30000],
[   32, 135000],
[   32, 100000],
[   25, 90000],
[   37, 33000],
[   35, 38000],
[   33, 69000],
[   18, 86000],
[   22, 55000],
[   35, 71000],
[   29, 148000],
[   29, 47000],
[   21, 88000],
[   34, 115000],
[   26, 118000],
[   34, 43000],
[   34, 72000],
[   23, 28000],
[   35, 47000],
[   25, 22000],
[   24, 23000],
[   31, 34000],

[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   26, 16000],
[   31, 71000],
[   32, 117000],
    33, 43000],
    33, 60000],
    31, 66000],
    20, 82000],
    33, 41000],
    35, 72000],
    28, 32000],

    24, 84000],
    19, 26000],
    29, 43000],
    19, 70000],
    28, 89000],
    34, 43000],
[   30, 79000],
[   20, 36000],
[   26, 80000],
[   35, 22000],
[   35, 39000],
[   49, 74000],
[   39, 134000],
[   41, 71000],
[   58, 101000],
[   47, 47000],
[   55, 130000],
[   52, 114000],
[   40, 142000],
[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   46, 22000],
[   48, 96000],
[   52, 150000],
[   59, 42000],
[   35, 58000],
[   47, 43000],
[   60, 108000],
[   49, 65000],
[   40, 78000],
[   46, 96000],
[   59, 143000],
[   41, 80000],
[   35, 91000],

[   37, 144000],
[   60, 102000],
[   35, 60000],
[   37, 53000],
[   36, 126000],
[   56, 133000],
[   40, 72000],









[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   42, 80000],
    35, 147000],
    39, 42000],
    40, 107000],
    49, 86000],
    38, 112000],
    46, 79000],
    40, 57000],
    37, 80000],
    46, 82000],
    53, 143000],
    42, 149000],
    38, 59000],
    50, 88000],
[   56, 104000],
[   41, 72000],
[   51, 146000],
[   35, 50000],
[   57, 122000],
[   41, 52000],
[   35, 97000],
[   44, 39000],
[   37, 52000],
[   48, 134000],
[   37, 146000],

[   50, 44000],
[   52, 90000],
[   41, 72000],
[   40, 57000],

[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   58, 95000],
[   45, 131000],
[   35, 77000],
[   36, 144000],
[   55, 125000],
[   35, 72000],
[   48, 90000],
[   42, 108000],
[   40, 75000],
[   37, 74000],
[   47, 144000],
[   40, 61000],
[   43, 133000],
[   59, 76000],
[   60, 42000],
[   39, 106000],
[   57, 26000],
[   57, 74000],
[   38, 71000],
    49, 88000],
    52, 38000],
    50, 36000],
    59, 88000],
    35, 61000],
    37, 70000],
    52, 21000],
    48, 141000],
    37, 93000],


[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
     37, 62000],
     48, 138000],
     41, 79000],
     37, 78000],
[   39, 134000],
[    49, 89000],
[    55, 39000],
[    37, 77000],
[    35, 57000],
[    36, 63000],
[    42, 73000],
[   43, 112000],
[    45, 79000],
[   46, 117000],
[    58, 38000],
[    48, 74000],
[   37, 137000],
[    37, 79000],
[    40, 60000],
[    42, 54000],
[   51, 134000],
[   47, 113000],
[   36, 125000],
[    38, 50000],
[    42, 70000],
[    39, 96000],
[    38, 50000],
[   49, 141000],
[    39, 79000],
[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   39, 75000],
[   54, 104000],
[   35, 55000],

[   45, 32000],
[   36, 60000],
[   52, 138000],
[   53, 82000],
[   41, 52000],
[   48, 30000],
    48, 131000],
    41, 60000],
    41, 72000],
    42, 75000],
    36, 118000],
    47, 107000],
    38, 51000],
    48, 119000],
    42, 65000],
    40, 65000],
    57, 60000],
    36, 54000],
    58, 144000],
[   35, 79000],
[   38, 55000],
[   39, 122000],
[   53, 104000],
[   35, 75000],
[   38, 65000],
[   47, 51000],

[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   47, 105000],
[   41, 63000],
[   53, 72000],
[   54, 108000],
[   39, 77000],
[   38, 61000],

[   38, 113000],
[   37, 75000],
[   42, 90000],
[   37, 57000],
[   36, 99000],
[   60, 34000],
[   54, 70000],
[   41, 72000],
[   40, 71000],
[   42, 54000],
[   43, 129000],
[   53, 34000],
[   47, 50000],
[   42, 79000],
[   42, 104000],
[   59, 29000],
[   58, 47000],
[   46, 88000],
[   38, 71000],
[   54, 26000],



[
[
[
[
[
```

```
[
[
[
[
[
[
[
[
[   60, 46000],
    60, 83000],
    39, 73000],
    59, 130000],
    37, 80000],
    46, 32000],
```

```
[
[
[
[
[
```

```
            [
            [
            [
            [
            [
            [
            [
            [
                46, 74000],
                42, 53000],
                41, 87000],
                58, 23000],
                42, 64000],
                48, 33000],
                44, 139000],
                49, 28000],
            [ 57, 33000],
             [ 56, 60000],
             [ 49, 39000],
             [ 39, 71000],
             [ 47, 34000],
             [ 48, 35000],
             [ 48, 33000],
             [ 47, 23000],
             [ 45, 45000],
             [ 60, 42000],
             [ 39, 59000],
             [ 46, 41000],
             [ 51, 23000],
             [ 50, 20000],
             [ 36, 33000],
            [ 49, 36000]], dtype=int64)
```

[133]:

```
label
```

```
[133]: array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                                          0, 0, 0,
            0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
            0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
            0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
            0, 0,
```

```
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,
       1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
       0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
```

[135]:
```
from sklearn.model_selection import
train_test_split from sklearn.linear_model
import LogisticRegression
```

[141]:
```
for i in range(1,401):
   x_train,x_test,y_train,y_test=train_test_split(features,label,te
   st_size=0.
↪2,random_state=i)
   model=LogisticRegression()
   model.fit(x_train,y_train)
   train_score=model.score(x_train,y_
   train)
```

```
   1, 1, 0, 1, 1, 1, 0, 1], dtype=int64)
```

```
    test_score=model.score(x_test,y_te st) if


    test_score>train_score:


        print("Test {} Train{} Random State {}".
  ↪format(test_score,train_score,i))
```

```
Test 0.9 Train0.840625 Random State 4
Test 0.8625 Train0.85 Random State 5
Test 0.8625 Train0.859375 Random State 6
Test 0.8875 Train0.8375 Random State 7
Test 0.8625 Train0.8375 Random State 9
Test 0.9 Train0.840625 Random State 10
Test 0.8625 Train0.85625 Random State 14
Test 0.85 Train0.84375 Random State 15
Test 0.8625 Train0.85625 Random State 16
Test 0.875 Train0.834375 Random State 18
Test 0.85 Train0.84375 Random State 19
Test 0.875 Train0.84375 Random State 20
Test 0.8625 Train0.834375 Random State 21
Test 0.875 Train0.840625 Random State 22
Test 0.875 Train0.840625 Random State 24
Test 0.85 Train0.834375 Random State 26
Test 0.85 Train0.840625 Random State 27
Test 0.8625 Train0.834375 Random State 30
Test 0.8625 Train0.85625 Random State 31
Test 0.875 Train0.853125 Random State 32
Test 0.8625 Train0.84375 Random State 33
Test 0.875 Train0.83125 Random State 35
Test 0.8625 Train0.853125 Random State 36
Test 0.8875 Train0.840625 Random State 38
Test 0.875 Train0.8375 Random State 39
Test 0.8875 Train0.8375 Random State 42
Test 0.875 Train0.846875 Random State 46
Test 0.9125 Train0.83125 Random State 47
Test 0.875 Train0.83125 Random State 51
Test 0.9 Train0.84375 Random State 54
Test 0.85 Train0.84375 Random State 57
Test 0.875 Train0.84375 Random State 58
Test 0.925 Train0.8375 Random State 61
Test 0.8875 Train0.834375 Random State 65
```

Test 0.8875 Train0.840625 Random State 68
Test 0.9 Train0.83125 Random State 72
Test 0.8875 Train0.8375 Random State 75
Test 0.925 Train0.825 Random State 76
Test 0.8625 Train0.840625 Random State 77
Test 0.8625 Train0.859375 Random State 81
Test 0.875 Train0.8375 Random State 82
Test 0.8875 Train0.8375 Random State 83
Test 0.8625 Train0.853125 Random State 84
Test 0.8625 Train0.840625 Random State 85
Test 0.8625 Train0.840625 Random State 87
Test 0.875 Train0.846875 Random State 88
Test 0.9125 Train0.8375 Random State 90
Test 0.8625 Train0.85 Random State 95
Test 0.875 Train0.85 Random State 99
Test 0.85 Train0.840625 Random State 101
Test 0.85 Train0.840625 Random State 102
Test 0.9 Train0.825 Random State 106
Test 0.8625 Train0.840625 Random State 107 Test 0.85 Train0.834375 Random State 109
Test 0.85 Train0.840625 Random State 111
Test 0.9125 Train0.840625 Random State 112
Test 0.8625 Train0.85 Random State 115
Test 0.8625 Train0.840625 Random State 116 Test 0.875 Train0.834375 Random State 119
Test 0.9125 Train0.828125 Random State 120
Test 0.8625 Train0.859375 Random State 125
Test 0.85 Train0.846875 Random State 128
Test 0.875 Train0.85 Random State 130
Test 0.9 Train0.84375 Random State 133
Test 0.925 Train0.834375 Random State 134
Test 0.8625 Train0.85 Random State 135
Test 0.875 Train0.83125 Random State 138 Test 0.8625 Train0.85 Random State 141
Test 0.85 Train0.846875 Random State 143
Test 0.85 Train0.846875 Random State 146
Test 0.85 Train0.84375 Random State 147 Test 0.8625 Train0.85 Random State 148
Test 0.875 Train0.8375 Random State 150
Test 0.8875 Train0.83125 Random State 151
Test 0.925 Train0.84375 Random State 152
Test 0.85 Train0.840625 Random State 153
Test 0.9 Train0.84375 Random State 154
Test 0.9 Train0.840625 Random State 155

Test 0.8875 Train0.846875 Random State 156
Test 0.8875 Train0.834375 Random State 158
Test 0.875 Train0.828125 Random State 159
Test 0.9 Train0.83125 Random State 161
Test 0.85 Train0.8375 Random State 163
Test 0.875 Train0.83125 Random State 164
Test 0.8625 Train0.85 Random State 169
Test 0.875 Train0.840625 Random State 171
Test 0.85 Train0.840625 Random State 172 Test
0.9 Train0.825 Random State 180
Test 0.85 Train0.834375 Random State 184
Test 0.925 Train0.821875 Random State 186
Test 0.9 Train0.83125 Random State 193
Test 0.8625 Train0.85 Random State 195
Test 0.8625 Train0.840625 Random State 196
Test 0.8625 Train0.8375 Random State 197
Test 0.875 Train0.840625 Random State
198 Test 0.8875 Train0.8375 Random State
199
Test 0.8875 Train0.84375 Random State 200
Test 0.8625 Train0.8375 Random State 202
Test 0.8625 Train0.840625 Random State
203 Test 0.8875 Train0.83125 Random State
206
Test 0.8625 Train0.834375 Random State 211
Test 0.85 Train0.84375 Random State 212
Test 0.8625 Train0.834375 Random State 214
Test 0.875 Train0.83125 Random State 217
Test 0.9625 Train0.81875 Random State 220
Test 0.875 Train0.84375 Random State 221
Test 0.85 Train0.840625 Random State 222
Test 0.9 Train0.84375 Random State 223
Test 0.8625 Train0.853125 Random State 227
Test 0.8625 Train0.834375 Random State 228
Test 0.9 Train0.840625 Random State 229
Test 0.85 Train0.84375 Random State 232
Test 0.875 Train0.846875 Random State 233
Test 0.9125 Train0.840625 Random State 234
Test 0.8625 Train0.840625 Random State 235
Test 0.85 Train0.846875 Random State 236
Test 0.875 Train0.846875 Random State 239
Test 0.85 Train0.84375 Random State
241 Test 0.8875 Train0.85 Random State
242
Test 0.8875 Train0.825 Random State 243
Test 0.875 Train0.846875 Random State 244

Test 0.875 Train0.840625 Random State 245
Test 0.875 Train0.846875 Random State 246
Test 0.8625 Train0.859375 Random State 247
Test 0.8875 Train0.84375 Random State 248
Test 0.8625 Train0.85 Random State 250
Test 0.875 Train0.83125 Random State 251
Test 0.8875 Train0.84375 Random State 252
Test 0.8625 Train0.846875 Random State 255
Test 0.9 Train0.840625 Random State 257 Test 0.8625 Train0.85625
Random State 260
Test 0.8625 Train0.840625 Random State 266
Test 0.8625 Train0.8375 Random State 268
Test 0.875 Train0.840625 Random State 275
Test 0.8625 Train0.85 Random State 276
Test 0.925 Train0.8375 Random State 277
Test 0.875 Train0.846875 Random State 282
Test 0.85 Train0.846875 Random State 283
Test 0.85 Train0.84375 Random State 285
Test 0.9125 Train0.834375 Random State 286
Test 0.85 Train0.840625 Random State 290
Test 0.85 Train0.840625 Random State 291
Test 0.85 Train0.846875 Random State 292
Test 0.8625 Train0.8375 Random State 294
Test 0.8875 Train0.828125 Random State 297
Test 0.8625 Train0.834375 Random State 300
Test 0.8625 Train0.85 Random State 301
Test 0.8875 Train0.85 Random State 302
Test 0.875 Train0.846875 Random State 303
Test 0.8625 Train0.834375 Random State 305
Test 0.9125 Train0.8375 Random State 306
Test 0.875 Train0.846875 Random State 308
Test 0.9 Train0.84375 Random State 311
Test 0.8625 Train0.834375 Random State 313
Test 0.9125 Train0.834375 Random State 314
Test 0.875 Train0.8375 Random State 315
Test 0.9 Train0.846875 Random State 317
Test 0.9125 Train0.821875 Random State 319
Test 0.8625 Train0.85 Random State 321
Test 0.9125 Train0.828125 Random State 322
Test 0.85 Train0.846875 Random State 328
Test 0.85 Train0.8375 Random State 332
Test 0.8875 Train0.853125 Random State 336
Test 0.85 Train0.8375 Random State 337
Test 0.875 Train0.840625 Random State 343
Test 0.8625 Train0.84375 Random State 346
Test 0.8875 Train0.83125 Random State 351

```
Test 0.8625 Train0.85 Random State 352
Test 0.95 Train0.81875 Random State 354
Test 0.8625 Train0.85 Random State 356
Test 0.9125 Train0.840625 Random State 357
Test 0.8625 Train0.8375 Random State 358
Test 0.85 Train0.840625 Random State 362
Test 0.9 Train0.84375 Random State 363
Test 0.8625 Train0.853125 Random State 364
Test 0.9375 Train0.821875 Random State 366
Test 0.9125 Train0.840625 Random State 369
Test 0.8625 Train0.853125 Random State 371 Test 0.925 Train0.834375
Random State 376
Test 0.9125 Train0.828125 Random State 377
Test 0.8875 Train0.85 Random State 378
Test 0.8875 Train0.85 Random State 379
Test 0.8625 Train0.840625 Random State 382
Test 0.8625 Train0.859375 Random State 386
Test 0.85 Train0.8375 Random State 387
Test 0.875 Train0.828125 Random State 388
Test 0.85 Train0.84375 Random State 394
Test 0.8625 Train0.8375 Random State 395
Test 0.9 Train0.84375 Random State 397 Test
0.8625 Train0.84375 Random State 400
```

[143]:
```
x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0
.2,
 ↪random_state=354)
finalModel=LogisticRegression()
finalModel.fit(x_train,y_train)
```

[143]: LogisticRegression()

[145]:
```
print(finalModel.score(x_train,y_train))
print(finalModel.score(x_test,y_test))
```

```
0.81875
0.95
```

[147]:
```
from sklearn.metrics import classification_report
print(classification_report(label,finalModel.predict(features)))
```

	precision	recall	f1-score	support
0	0.85	0.91	0.88	257
1	0.82	0.72	0.77	143

```
    accuracy                           0.84       400
   macro avg       0.84      0.82       0.83       400
weighted           0.84      0.84       0.84       400
avg
```

[ ]: