Data sience internshala

AND OPERATOR

a=0

b=1

a and b

o/p: 0

<u>or ,</u>

a=1

b=1

a and b

o/p:1

or,

```
In [11]: 3 and 5
Out[11]: 5

In [12]: 0 and 3
Out[12]: 0

In [13]: 4 and 0
Out[13]: 0

In [14]: 7 and 8
Out[14]: 8

In [ ]:
```

Or,

3<5 and 5>3

o/p: True

or,

3<5 and 4>7

o/p:

False

Or,

8<5 and 4>7

o/p

False

OR OPERATOR,

a=0

b=1

a or b

O/P:1

```
In [18]: 1 or 1
Out[18]: 1

In [19]: 0 or 1
Out[19]: 1

In [21]: 1 or 0
Out[21]: 1

In [23]: 0 or 0
Out[23]: 0

In [25]: 3 or 5
Out[25]: 3

In [26]: 4 or 0
Out[26]: 4
```

```
In [27]: 0 or 3
Out[27]: 3

In [28]: 3<5 or 8<7
Out[28]: True

In [29]: 9<5 or 5>9
Out[29]: False
```

NOT OPERATOR:

```
In [30]: not 0 # it flip the boolean value, the boolean value of 0 is false
         #so not 0 return trrue

Out[30]: True

In [31]: not 3# the boolean value of 3 is true

Out[31]: False

In [32]: not 'shubha' #the boolean value of 'shubha' is true

Out[32]: False
```

not 3<5

o/p:

False

<u>Arrithmatic operator:</u>

shubha'+str(20)

o/p:

'shubha20'

<u>Comparison operator:</u>

5*3<6*4

o/p: True

or,

5*3>6*4

o/p: False

<u>how variable work in python:</u>

```
a=7
b=a
a=3
print('a=',a)
print('b=',b)
```

o/p:

a= 3

b= 7

variable naming rule:

```
In [45]: 5=a
         print(a)

  File "<ipython-input-45-2306000d461e>", line 1
    5=a
       ^
SyntaxError: can't assign to literal
```

```
In [46]: a=5
         print(a)

5

In [ ]: 
```

Conditional statement:

```
x=7
if (x>5):
    print('large')
elif(x>3):
    print('medium')
else:
    print('low')
```

o/p:

large

or,

```
y=int(input('Enter the number:'))
if(y>90):
    print('Grade A')
elif(y>60):
    print('Grade B')
else:
    print('Grade F')
```

o/p:

```
Enter the number:92
Grade A
```

o/r:

```
y=int(input('Enter the number:'))
if(y>90):
    print('Grade A')
elif(y>60):
    print('Grade B')
else:
    print('Grade F')
```

o/p:

```
Enter the number:62
Grade B
```

Looping statement:

```
In [53]: for i in range(5):
             print('python is awesom')

python is awesom
python is awesom
python is awesom
python is awesom
python is awesom
```

```
In [54]: for i in range(2.0):
             print('shubha')

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-54-be8a6fa03be9> in <module>
----> 1 for i in range(2.0):
      2     print('shubha')

TypeError: 'float' object cannot be interpreted as an integer
```

```
In [55]: for i in range '':
             print(i)

  File "<ipython-input-55-15465e4850e0>", line 1
    for i in range '':
                    ^
SyntaxError: invalid syntax
```

Print odd number

```
#print all the odd number between 10,20
for i in range(11,20):
    if (i%2!=0):
        print('odd number','and value of  is ',i)
```

or,

```
odd number and value of  is  11
odd number and value of  is  13
odd number and value of  is  15
odd number and value of  is  17
odd number and value of  is  19
```

greater of two:

```
def greater_of_two(x,y):
    if (x>y):
        print(x)
    else:
        print(y)
```

or,

```
greater_of_two(9,12)
o/p:12
```

or,

```
def greater_of_two(x,y):
    if (x>y):
        greater=x
    else:
        greater=y
    return greater
```

\

List:

list3=[1,'python',2,'is',3,'awesom']

list3[-5:-1]

o/p:

```
['python', 2, 'is', 3]
```

Extend keyword:

list3=[1,'python',2,'is',3,'awesom']

list3.extend([4,5])

list3

o/p:

```
[1, 'python', 2, 'is', 3, 'awesom', 4, 5]
```

Append keyword:

list3=[1,'python',2,'is',3,'awesom']

list3.append([4,5])

list3

o/p:

```
[1, 'python', 2, 'is', 3, 'awesom', [4, 5]]
```

Or,

list3=[1,'python',2,'is',3,'awesom']

list3.append([4,5])

list3[6]

o/p:

```
[4, 5]
```

Remove from a list:

list3=[1,'python',2,'is',3,'awesom']

list3.remove(2)

list3

o/p:

```
[1, 'python', 'is', 3, 'awesom']
```

Removing by index:

```
list3=[1,'python',2,'is',3,'awesom']
del list3[3]# removing by index
list3
```

o/p:

```
[1, 'python', 2, 3, 'awesom']
```

Looping of a list:

```
list3=[1,'python',2,'is',3,'awesom']
for i in list3:
    print(i)
```

o/p:

```
1
python
2
is
3
awesom
```

or,

```
list3=[1,'python',2,'is',3,'awesom']
for i in range (len(list3)):
    print(list3[i])
```

o/p:

```
1
python
2
is
3
Awesome
```

Or,

```
marks=[1,2,3,4]
for marks in marks:
    print(marks+1)
```

o/p
:

```
2
3
```

4
5

Dictionary;

```
dict={'suresh':[42,25],'ramesh':[45,52],'brijesh':[58,32]}
dict['suresh']
```

o/p:

```
[42, 25]
```

Element added to dictionary:

Add a single element:

```
dict={'suresh':[42,25],'ramesh':[45,52],'brijesh':[58,32]}
dict['shubha']=[26,25]
dict
```

o/p:

```
{'suresh': [42, 25],
 'ramesh': [45, 52],
 'brijesh': [58, 32],
 'shubha': [26, 25]}
```

Add multiple element:

```
dict={'suresh':[42,25],'ramesh':[45,52],'brijesh':[58,32]}
dict.update({'shubha':26,'rajesh':58})
dict
```

o/p:

```
{'suresh': [42, 25],
 'ramesh': [45, 52],
 'brijesh': [58, 32],
 'shubha': 26,
 'rajesh': 58}
```

Or,

```
dict={'suresh':[42,25],'ramesh':[45,52],'brijesh':[58,32]}
dict.update({'shubha':[26,25],'rajesh':[58,25]})
dict
```

o/p;

```
{'suresh': [42, 25],
 'ramesh': [45, 52],
 'brijesh': [58, 32],
 'shubha': [26, 25],
 'rajesh': [58, 25]}
```

Delete from dictionary:

```
dict={'suresh':[42,25],'ramesh':[45,52],'brijesh':[58,32]}
dict.update({'shubha':[26,25],'rajesh':[58,25]})
del dict['ramesh']
dict
```

o/p:

```
{'suresh': [42, 25],
 'brijesh': [58, 32],
 'shubha': [26, 25],
 'rajesh': [58, 25]}
```

Excess value from dictionary:

```
dict={'a':2,'b':3,'c':4}
items=list(dict.items())
print(items)
i=[(keys,values)for keys,values in items]
print(i)
```

o/p:

```
[('a', 2), ('b', 3), ('c', 4)]
[('a', 2), ('b', 3), ('c', 4)]
```

```
import pandas as pd

df=pd.read_csv(r'C:\Users\Shubhamay\Documents\team_score.csv')

print(df.head())
```

o/p:

```
name  score
0  shubha     56
1   sujit     65
2    babu     20
3   jiten     45
4  sourav     18
```

Reading excel file:

```
import pandas as pd

df=pd.read_excel(r'C:\Users\Shubhamay\Documents\salary_me.xlsx')
```

```
print(df.head())
```

o/p:

```
  name  salary
0  shubha   40000
1   mahim   20000
2     ram   50000
3   shyam   72000
```

No.of row of a table:

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\team_score.csv')
df.shape[0]
```

o/p:

```
8
```

No of column of a table:

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\team_score.csv')
df.shape[1]
```

o/p:

```
2
```

Shape of a dataframe:

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
casts.shape# it has 75001 rows and 6 columns
```

o/p:

```
(75001, 6)
```

All columns from a dataframe:

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
casts.columns
```

o/p:

```
Index(['title', 'year', 'name', 'type', 'character', 'n'], dtype='object')
```

Selecting one column:

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
casts['year'].head()
```

o/p:

```
0    2015
1    1985
2    2017
3    2015
4    2015
Name: year, dtype: int64
```

Selecting multiple column:

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
print(casts[['name','type']].head())
```

o/p:

```
       name   type
0  Buffy #1  actor
1    Homo $  actor
2   $hutter  actor
3   $hutter  actor
4   $hutter  actor
```

Or,

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
print(casts[{'name','type'}].head())
```

o/p:

```
       name   type
0  Buffy #1  actor
1    Homo $  actor
2   $hutter  actor
3   $hutter  actor
4   $hutter  actor
```

First five row

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
print(casts.iloc[:5])
```

o/p:

```
                 title  year      name   type              character     n
0       Closet Monster  2015  Buffy #1  actor                Buffy 4  31.0
1      Suuri illusioni  1985    Homo $  actor                 Guests  22.0
2  Battle of the Sexes  2017   $hutter  actor        Bobby Riggs Fan  10.0
```

```
3  Secret in Their Eyes  2015   $hutter  actor          2002 Dodger Fan   NaN
4           Steve Jobs  2015   $hutter  actor  1988 Opera House Patron   NaN
```

Select last five row:

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
print(casts.iloc[-5:])
```

o/p:

```
                         title  year                         name     type  \
74996  Mia fora kai ena... moro  2011     Penelope Anastasopoulou  actress
74997         The Magician King  2004       Tiannah Anastassiades  actress
74998        Festival of Lights  2010             Zoe Anastassiou  actress
74999                Toxic Tutu  2016             Zoe Anastassiou  actress
75000           Fugitive Pieces  2007  Anastassia Anastassopoulou  actress

                    character     n
74996      Popi voulkanizater  11.0
74997  Unicycle Race Attendant   NaN
74998       Guidance Counselor  20.0
74999        Demon of Toxicity   NaN
75000             Laundry Girl  25.0
```

Select 26 th row to 32th row

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
print(casts.iloc[26:32])
```

o/p:

```
                                              title  year  \
26                     The LXD: The Secrets of the Ra  2011
27                                        Todo x Sara  2014
28                                    Barrio Gangsters  2009
29  Cornmeal, Gunpowder, Ham Hocks and Guitar Strings  2015
30                     The LXD: The Secrets of the Ra  2011
31                                  Pelotazo nacional  1993

                          name   type   character     n
26        Jesse 'Casper' Brown  actor       Fangz   NaN
27          Gil 'Colibri'Viera  actor     Llavero   NaN
28   Marcelino 'Dibujo' Torres  actor      Dibujo   NaN
29      Donnie 'Dicky' Clemson  actor  Jeb Kinney   NaN
30  Dondraico 'Draico' Johnson  actor       Umbra   NaN
31              F?lix 'El Gato'  actor   Rebolledo  12.0
```

Select all row and two column:

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
```

```
print(casts.iloc[:,:2])
```

o/p:

```
                          title  year
0                Closet Monster  2015
1               Suuri illusioni  1985
2           Battle of the Sexes  2017
3          Secret in Their Eyes  2015
4                    Steve Jobs  2015
...                         ...   ...
74996  Mia fora kai ena... moro  2011
74997         The Magician King  2004
74998        Festival of Lights  2010
74999                Toxic Tutu  2016
75000           Fugitive Pieces  2007

[75001 rows x 2 columns]
```

Select customer with sex male:

```
import pandas as pd
data=pd.read_csv(r'C:\Users\Shubhamay\Documents\Module Test 1.csv')
print(data[data['Gender']=='Male'])
```

o/p:

```
      Loan_ID Gender Married Dependents     Education Self_Employed  \
0    LP001002   Male      No          0      Graduate            No
1    LP001003   Male     Yes          1      Graduate            No
2    LP001005   Male     Yes          0      Graduate           Yes
3    LP001006   Male     Yes          0  Not Graduate            No
4    LP001008   Male      No          0      Graduate            No
..        ...    ...     ...        ...           ...           ...
607  LP002964   Male     Yes          2  Not Graduate            No
608  LP002974   Male     Yes          0      Graduate            No
610  LP002979   Male     Yes         3+      Graduate            No
611  LP002983   Male     Yes          1      Graduate            No
612  LP002984   Male     Yes          2      Graduate            No

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5849                0.0         NaN             360.0
1               4583             1508.0       128.0             360.0
2               3000                0.0        66.0             360.0
3               2583             2358.0       120.0             360.0
4               6000                0.0       141.0             360.0
..               ...                ...         ...               ...
607             3987             1411.0       157.0             360.0
608             3232             1950.0       108.0             360.0
610             4106                0.0        40.0             180.0
611             8072              240.0       253.0             360.0
612             7583                0.0       187.0             360.0

     Credit_History Property_Area Loan_Status
```

```
0              1.0         Urban           Y
1              1.0         Rural           N
2              1.0         Urban           Y
3              1.0         Urban           Y
4              1.0         Urban           Y
..             ...           ...         ...
607            1.0         Rural           Y
608            1.0         Rural           Y
610            1.0         Rural           Y
611            1.0         Urban           Y
612            1.0         Urban           Y

[489 rows x 13 columns]
```

<u>Customer with sex female:</u>

```
import pandas as pd
data=pd.read_csv(r'C:\Users\Shubhamay\Documents\Module Test 1.csv')
print(data[data['Gender']=='Female'])
```

o/p:

```
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
17   LP001036  Female      No          0      Graduate            No
29   LP001087  Female      No          2      Graduate           NaN
37   LP001112  Female     Yes          0      Graduate            No
45   LP001137  Female      No          0      Graduate            No
48   LP001146  Female     Yes          0      Graduate            No
..        ...     ...     ...        ...           ...           ...
587  LP002917  Female      No          0  Not Graduate            No
600  LP002949  Female      No         3+      Graduate           NaN
604  LP002959  Female     Yes          1      Graduate            No
609  LP002978  Female      No          0      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
17              3510                0.0        76.0             360.0
29              3750             2083.0       120.0             360.0
37              3667             1459.0       144.0             360.0
45              3410                0.0        88.0               NaN
48              2645             3440.0       120.0             360.0
..               ...                ...         ...               ...
587             2165                0.0        70.0             360.0
600              416            41667.0       350.0             180.0
604            12000                0.0       496.0             360.0
609             2900                0.0        71.0             360.0
613             4583                0.0       133.0             360.0

     Credit_History Property_Area Loan_Status
17              0.0         Urban           N
29              1.0     Semiurban           Y
37              1.0     Semiurban           Y
45              1.0         Urban           Y
48              0.0         Urban           N
..              ...           ...         ...
```

```
587             1.0     Semiurban           Y
600             NaN         Urban           N
604             1.0     Semiurban           Y
609             1.0         Rural           Y
613             0.0     Semiurban           N

[112 rows x 13 columns]
```

Name 2nd row and third column

```
import pandas as pd
casts = pd.read_csv(r'C:\Users\Shubhamay\Documents\cast.csv')
casts.iloc[1,2]
```

o/p:

'Homo $'

Select only dependent and education:

```
import pandas as pd
data=pd.read_csv(r'C:\Users\Shubhamay\Documents\Module Test 1.csv')
data.loc[:,['Dependents','Education']]
```

o/p:

ut[38]:

	Dependents	Education
0	0	Graduate
1	1	Graduate
2	0	Graduate
3	0	Not Graduate
4	0	Graduate
...	...	...
609	0	Graduate
610	3+	Graduate
611	1	Graduate
612	2	Graduate
613	0	Graduate

Select dependents,education,selfemployed

```
import pandas as pd
data=pd.read_csv(r'C:\Users\Shubhamay\Documents\Module Test 1.csv')
data.iloc[:,3:6]
```

t[40]:

	Dependents	Education	Self_Employed
0	0	Graduate	No
1	1	Graduate	No
2	0	Graduate	Yes
3	0	Not Graduate	No
4	0	Graduate	No
...	...	...	...
609	0	Graduate	No
610	3+	Graduate	No
611	1	Graduate	No
612	2	Graduate	No
613	0	Graduate	Yes

614 rows × 3 columns

Favourite subject :

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
df['favourite subject'].mode()
```

o/p:

```
0    physics
dtype: object
```

here is two mode:

```
dict={'no':[2,4,6,2,7,8,7,6,7,2]}
df=pd.DataFrame(dict)
df['no'].mode()#because 2,7 occur for same time
```

o/p:

```
0    2
1    7
dtype: int64
```

mean marks of a data sate:

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
```

```
df['marks'].mean()
```

o/p:

75.36363636363636

Mean of first 10 prime number:

```
dict={'no':[2,3,5,7,11,13,17,19,23,29]}
df=pd.DataFrame(dict)
df['no'].mean()
```

o/p:

12.9

Median:

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
Q2=df['marks'].median()
Q1=df['marks'].quantile(.25)
Q3=df['marks'].quantile(.75)
Q4=df['marks'].quantile(1.0)
print('first quartile:',Q1)
print('median/second quartile:',Q2)
print('3rd quartile:',Q3)
print('highest_marks/fourth quartile:',Q4)
```

o/p:

```
first quartile: 67.25
median/second quartile: 76.5
3rd quartile: 85.0
highest_marks/fourth quartile: 91.0
```

median of first 20 whole number:

```
dict={'no':[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]}
df=pd.DataFrame(dict)
df['no'].median()
```

o/p:

9.5

Range and IQR(inter quartile range):

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
p1=df['marks'].max()
p2=df['marks'].min()
range_data=p1-p2
print('range of dataset is:',range_data)
Q1=df['marks'].quantile(0.25)
Q2=df['marks'].quantile(.75)
```

```
iqr=Q2-Q1
print('\n')
print('IQR of the data set:',iqr)
```

o/p:

range of dataset is: 34

IQR of the data set: 17.75

Or,

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\new_fav.csv')#with outlier
p1=df['marks'].max()
p2=df['marks'].min()
range_data=p1-p2
print('range of dataset is:',range_data)# range is susceptive to
outlier,so,it is not robust
Q1=df['marks'].quantile(0.25)
Q2=df['marks'].quantile(.75)
iqr=Q2-Q1
print('\n')
print('IQR of the data set:',iqr)#it is not
```

o/p:

range of dataset is: 9713

IQR of the data set: 18.5

<u>Variance:</u>

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
mean_data=df['marks'].mean()
print('mean of the data set:',mean_data)
difference=df['marks']-mean_data
print('\n')
print(difference)
```

or,

mean of the data set: 75.71428571428571

```
0     15.285714
1      9.285714
2     11.285714
3    -13.714286
4     -0.714286
```

```
5     -8.714286
6     -7.714286
7      2.285714
8     -2.714286
9      2.285714
10   -10.714286
11    13.285714
12   -18.714286
13     9.285714
Name: marks, dtype: float64
```

Or,

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
mean_data=df['marks'].mean()
#print('mean of the data set:',mean_data)
difference=df['marks']-mean_data
print('\n')
#print(difference)
squre_of_difference=(difference)**2
print(squre_of_difference)
```

o/p:

```
0     233.653061
1      86.224490
2     127.367347
3     188.081633
4       0.510204
5      75.938776
6      59.510204
7       5.224490
8       7.367347
9       5.224490
10    114.795918
11    176.510204
12    350.224490
13     86.224490
Name: marks, dtype: float64
```

Or,

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
mean_data=df['marks'].mean()
#print('mean of the data set:',mean_data)
difference=df['marks']-mean_data
print('\n')
#print(difference)
squre_of_difference=(difference)**2
#print(squre_of_difference)
```

```
variance=(squre_of_difference).mean()
print('variance of the data set:',variance)
```

o/p:

variance of the data set: 108.34693877551021

<u>calculating variance with sigle command:</u>

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
variance=df['marks'].var(ddof=0)
print('variance of the data set:',variance)
```

o/p:

variance of the data set: 108.34693877551021

<u>calculating variance:</u>

```
dict={'no':[7, 6, 8, 4, 2, 7, 6, 7, 6, 5]}
df=pd.DataFrame(dict)
df['no'].var(ddof=0)
```

o/p:

2.7600000000000002

Or,

```
dict={'no':[7, 6, 8, 4, 2, 7, 6, 7, 6, 5]}
df=pd.DataFrame(dict)
df['no'].var()
```

o/p:

3.066666666666667

<u>Calculation of standerd deviation:</u>

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
mean_data=df["marks"].mean()
difference=df["marks"]-mean_data
square_of_difference=(difference)**2
variance=square_of_difference.mean()
std=(variance)**(1/2)
print("standerd deviation of the dataset:",std)
```

o/p:

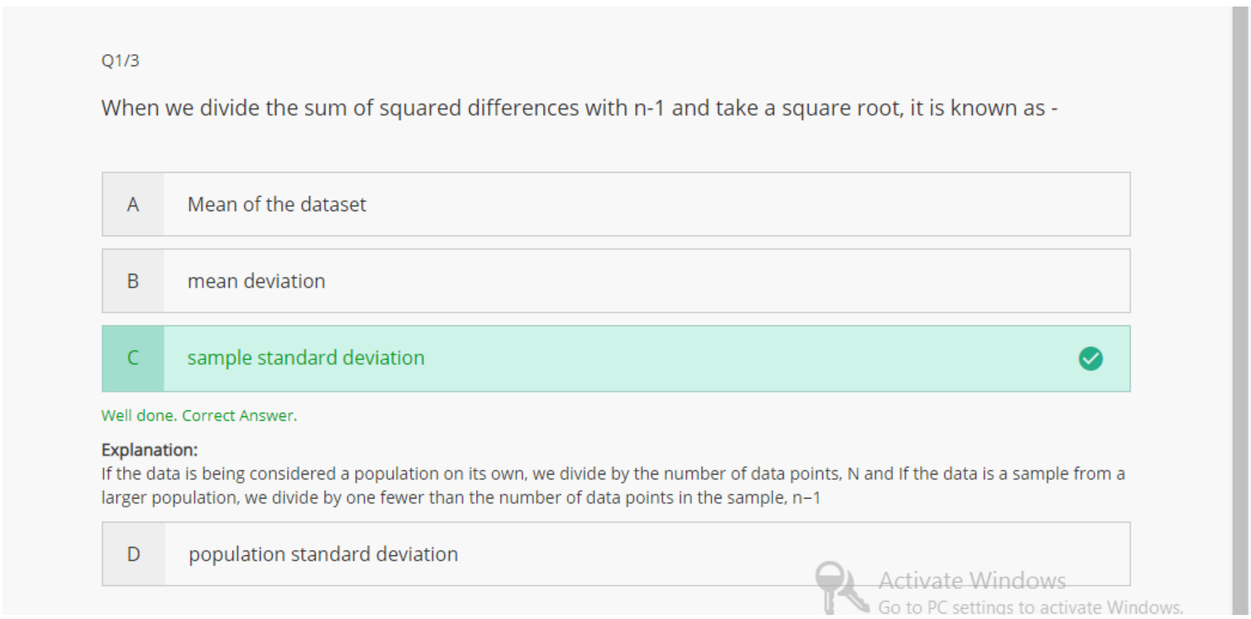
```
standerd deviation of the dataset: 10.408983561112498
```

with direct formula:

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
std=df["marks"].std(ddof=0)
print("standerd deviation of the dataset:",std)
```

o/p:

```
standerd deviation of the dataset: 10.408983561112498
```



One standerd deviation:

```
dict={'no':[180, 313, 101, 255, 202, 198, 109, 183, 181, 113, 171, 165,
318, 145, 131, 145, 226, 113, 268, 108]}
df=pd.DataFrame(dict)
mean_data=df['no'].mean()
print('mean of the dataset:',mean_data)
print('\n')
std=df['no'].std(ddof=0)
print('standerd deviation of the dataset:',std)
add=mean_data+std
print('\n')
print('addition of mean data and standerd deviation',add)
sub=mean_data-std
print('\n')
print('substraction of mean data and standerd deviation',sub)
```

How many values, in the below series, fall within one standard deviation of the mean?

180, 313, 101, 255, 202, 198, 109, 183, 181, 113, 171, 165, 318, 145, 131, 145, 226, 113, 268, 108

A	9
B	10
C	11

Well done. Correct Answer.

Explanation:

n this question, you have to find the number of points that falls in the range within one standard deviation of the mean. So, you can first calculate the mean and the standard deviation of the given data. It comes out to be 181.25 and 64.65 respectively. Now, to calculate one standard deviation above means, you add the standard deviation to the mean once. So, when you add 64.65 (standard deviation) with 181.25 (mean), you approximately get 245, and to calculate one standard deviation below means, you subtract the standard deviation from the mean once. So, when you subtract 64.65 (standard deviation) from 181.25 (mean), you approximately get 117. Hence the range that we get is 117-245. Hence you need to count the numbers which are within this range. You have 11 numbers in this range, so your answer is 11

n this question, you have to find the number of points that falls in the range within one standard deviation of the mean. So, you can first calculate the mean and the standard deviation of the given data. It comes out to be 181.25 and 64.65 respectively. Now, to calculate one standard deviation above means, you add the standard deviation to the mean once. So, when you add 64.65 (standard deviation) with 181.25 (mean), you approximately get 245, and to calculate one standard deviation below means, you subtract the standard deviation from the mean once. So, when you subtract 64.65 (standard deviation) from 181.25 (mean), you approximately get 117. Hence the range that we get is 117-245. Hence you need to count the numbers which are within this range. You have 11 numbers in this range, so your answer is 11

<u>feeequency table of the dataset:</u>

```
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
freq_data=df['favourite subject'].value_counts()
print('freeqency of the data set:','\n',freq_data)
```

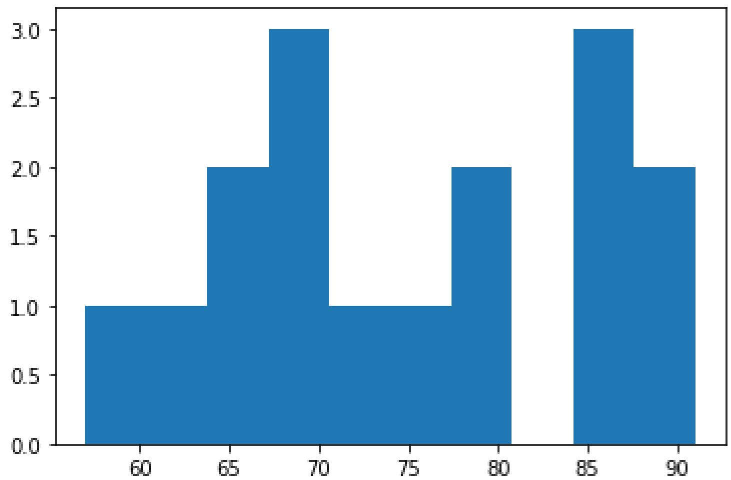
o/p:

```
freeqency of the data set:
 English       4
chemistry     4
physics       4
biology       2
math          1
bangla        1
Name: favourite subject, dtype: int64
```

<u>Histogram: use for ploting the frequency:</u>

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
plt.hist(x='marks',data=df)
plt.show()
```

o/p:

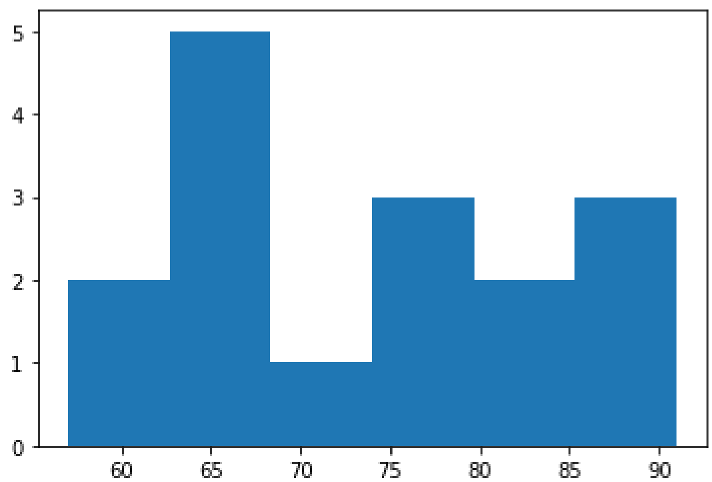


From histogram we observ that lot of student score between (67.5-7) and (85-87.5)

Or,

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')
plt.hist(x='marks',data=df,bins=6,)
plt.show()
```

o/p:



From this histogram we came into conclusion that a lot of student score between 65-70

Q1/3

What is the difference between bar and histogram charts?

A Histogram is for continuous variables whereas Bar graph is for categorical variables correct

Well done. Correct Answer.

Explanation:
We use histogram for continuous variable and bar graph

B Bar charts display frequency whereas Histogram display percentages

C Histogram doesn't show the entire range

D Bar chart is symmetric whereas Histogram is skewed

Z-Factor:

We often encounter a problem that we have to find probability between two points under a standard normal distribution by calculating area under the curve. Which generally is calculated using integral calculus. But for the sake of simplicity, the statisticians defined a z-table, with the help of which the area under the standard normal distribution can be calculated. Note that using the left t-table, which gives the area to the left of z.

In order to use the z table, let us look at the structure of the z table:

along the column is the first decimal of the z value and on the x axis is the second decimal of the z value.

To use this table, we need to locate the first and the second decimal of the z value on the x axis and the y axis respectively and through their intersection, the area under the curve from the left is obtained.

given z = 1.47

for this,

1.We will first locate 1.4 in the y axis,

2.Then locate 0.07 in the x axis.

the corresponding value in the table is the area to the left of z.

As we can see the corresponding value is 0.92, which means that 92% of the area is covered.

given z = 1.47

We have to calculate the Area to the RIGHT of the z value.

Calculate the area to the left of z using the procedure in example 1 and referring to the table, let us name it k.

We know that the area of Standard Normal Distribution is 1.

Therefore the Area to the Right of the z(Shaded Region) value can be calculated as:
Area = 1 - k

the area is (1 - 0.92) = 0.08, which means that 8% of the area is covered.

Calculating the Area between the two z Values: z1(left) and z2(right) respectively.
where z1 = 0.52 and z2 = 1.47

In order to calculate this area, we must calculate the area to the left of both z1 and z2 and then subtract z1 from z2, what is left is the area we need.

Area = z2 - z1
Area = 0.92 - 0.69
Area = 0.23

Which means that 23% of the area is covered.

So we now know how to use the z tables for all the possible problems without needing to use integrals.

Q1/4

Confidence Interval defines

A A range of values in which the sample statistic may lie

B A range of values in which the standard deviation of the sample may lie

C A range of values in which the population statistic may lie correct

Well done. Correct Answer.

Explanation:
Confidence interval defines a region where there is the highest chances of population statistic to lie. Hence option c is correct

D A range of values in which the sample mean may lie

When testing the hypothesis, which value separates the critical region from the noncritical region in a normal curve?

A:Critical value

A critical value is the point (or points) on the scale of the test statistic beyond which we reject the null hypothesis, and is derived from the level of significance of the test.

Type 1 error:

a type I error leads one to conclude that a supposed effect or relationship exists when in fact it doesn't. Examples of type I errors include a test that shows a patient to have a disease when in fact the patient does not have the disease, a fire alarm going on indicating a fire when in fact there is no fire, or an experiment indicating that a medical treatment should cure a disease when in fact it does not. So, it is the incorrect rejection of true null hypothesis

Q3/3

A result was said to be significant at the 5% level. What does this imply?

A The null hypothesis is probably wrong

B The null hypothesis is probably correct

C The result would be unexpected if the null hypothesis were true

Well done. Correct Answer.

D None of the above

Computing one-sample t-test:

```
#lets conduct a sample ttest to check if the mean\

#of the sample insect is similar to the mean of earlier insect is 6.09\

#we have degree of freedom that is 28 and taking significance level\

#to be 0.05,the t-critical value comes out to be 2.048

import pandas as pd

import scipy.stats as stats

from scipy.stats import ttest_1samp

file=pd.read_csv(r'C:\Users\Shubhamay\Documents\one sample.csv')
```

data=file['length of insect']

t_statistic, _ =ttest_1samp(data,6.09)

print(t_statistic)

#t-static>t-critical value.we can reject null hypothesis

o/p:

6.969339710044977

Two sample test:

```
#calculating t-statistic  and p-value using two sample test
import pandas as pd
import scipy.stats as stats
from scipy.stats import ttest_ind
file=pd.read_csv(r'C:\Users\Shubhamay\Documents\two sample.csv')
t_statsitic,p_value=ttest_ind(file['Hauz Khas'],file['Defence 
Colony'],equal_var=False)
print(p_value)



#as the p_value is less than 0.05,we can reject the null value
```

One sample test:

```
#lets conduct a sample ttest to check if the mean\
#of the sample insect is similar to the mean of earlier insect is 6.09\
#we have degree of freedom that is 28 and taking significance level\
#to be 0.05,the t-critical value comes out to be 2.048
import pandas as pd
import scipy.stats as stats
from scipy.stats import ttest_1samp
file=pd.read_csv(r'C:\Users\Shubhamay\Documents\one sample.csv')
data=file['length of insect']
t_statistic =ttest_1samp(data,6.09)
print(t_statistic)
#t-static>t-critical value.we can reject null hypothesis


o/p:

Ttest_1sampResult(statistic=6.969339710044977, 
pvalue=0.000935448462639376)
```

Paired t-test:

```
#compute paired t-test
import pandas as pd
```

```
import scipy.stats as stats
from scipy.stats import ttest_rel
file=pd.read_csv(r'C:\Users\Shubhamay\Documents\paired t-test.csv')
#print(file)
t_statistic,p_value=stats.ttest_rel(file['Error using
typewritter'],file['Error using a Compute'])
print(p_value)
#as the p-value is less than 0.05. we can reject the null hypothesis(mean
error typewritter=mean error from computer)
```

o/p:

0.18562245418957538

Chi squre test:

```
import pandas as pd
import scipy.stats as stats
from scipy.stats import chisquare
file=pd.read_csv(r'C:\Users\Shubhamay\Documents\chi squre test.csv')
t_statistic,p_value=stats.chisquare(file['observed'],file['expected'])
print(p_value)
#p_value is slightly more than 0.05 so we fail to reject null value
```

o/p:

0.08887585044058065

Calculation of co relation separately:

```
import pandas as pd

data=pd.read_csv(r'C:\Users\Shubhamay\Documents\corelation.csv')

print(data[['item_weight','item_MRP']].corr())

print('\n')

print(data[['item_MRP','item_outlet_sale']].corr())

print('\n')

print(data[['item_weight','item_outlet_sale']].corr())
```

o/p:

```
              item_weight  item_MRP
item_weight       1.00000   0.45758
item_MRP          0.45758   1.00000
```

```
                  item_MRP  item_outlet_sale
item_MRP          1.000000          0.835806
item_outlet_sale  0.835806          1.000000
```

```
                  item_weight  item_outlet_sale
item_weight          1.000000          0.138755
item_outlet_sale     0.138755          1.000000
```

find correlation together:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
data=pd.read_csv(r'C:\Users\Shubhamay\Documents\corelation.csv')

print (data.corr())
```

o/p:

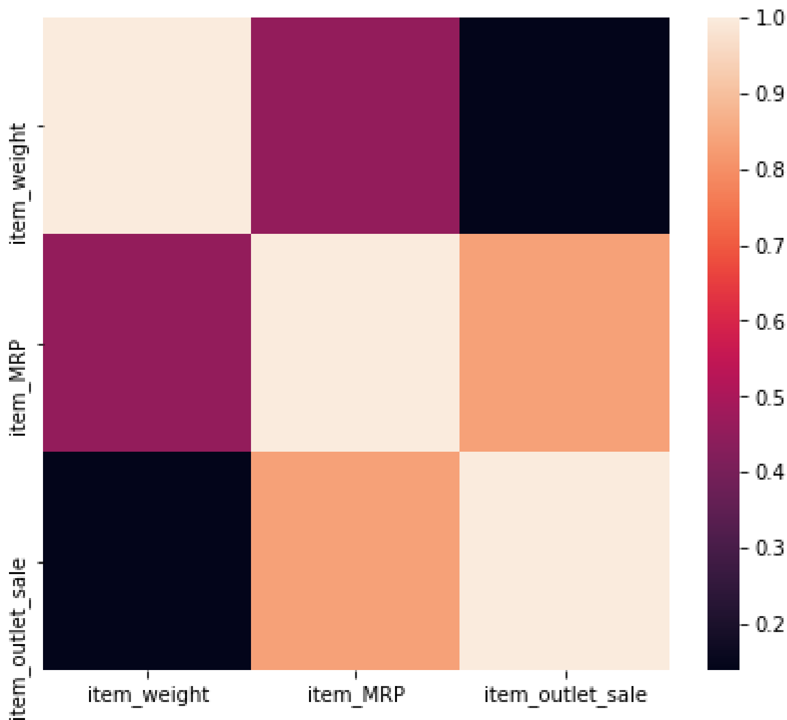
```
                  item_weight  item_MRP  item_outlet_sale
item_weight          1.000000  0.457580          0.138755
item_MRP             0.457580  1.000000          0.835806
item_outlet_sale     0.138755  0.835806          1.000000

#the correlation between item_weight and item_mrp is 0.457580
#the correlation between item_mrp and item_outletsale is 0.138755
#the correlation between item _outlet_sale and item MRP is 0.835806
```

Plotting co relation chart:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

corr=data.corr()
```

```
plt.figure(figsize=(7,6))

sns.heatmap(corr)

plt.show()
```

o/p:



<u>Identification of continuous or discreate variable in pandas:</u>

```
import pandas as pd

df=pd.read_csv('data_python.csv')

df.dtypes

#categorical variable stord as object

#continious variable stored as int float
```

o/p:

```
Loan_ID                  object
Gender                   object
Married                  object
Dependents               object
Education                object
Self_Employed            object
ApplicantIncome           int64
CoapplicantIncome       float64
LoanAmount              float64
Loan_Amount_Term        float64
Credit_History          float64
Property_Area            object
Loan_Status              object
dtype: object
```

NB: The difference between the third and first quartile is known as the inter quartile range.

Univariate analysis for continuous variable

1.tabular format:

```
import pandas as pd

df=pd.read_csv('data_python.csv')

df.describe()
```

]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

2.Graphical method

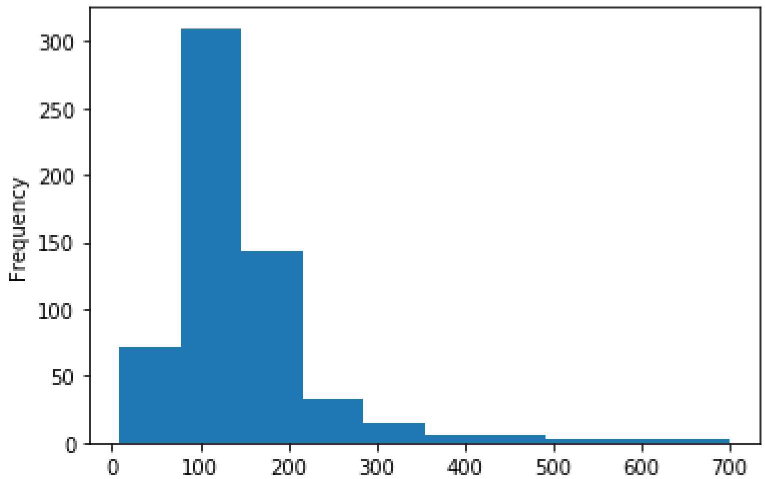
Plotting histogram

```
import pandas as pd

df=pd.read_csv('data_python.csv')
```

```
#histogram of loan amount

df['LoanAmount'].plot.hist()
```

op:

```
<matplotlib.axes._subplots.AxesSubplot at 0x46dc2df688>
```

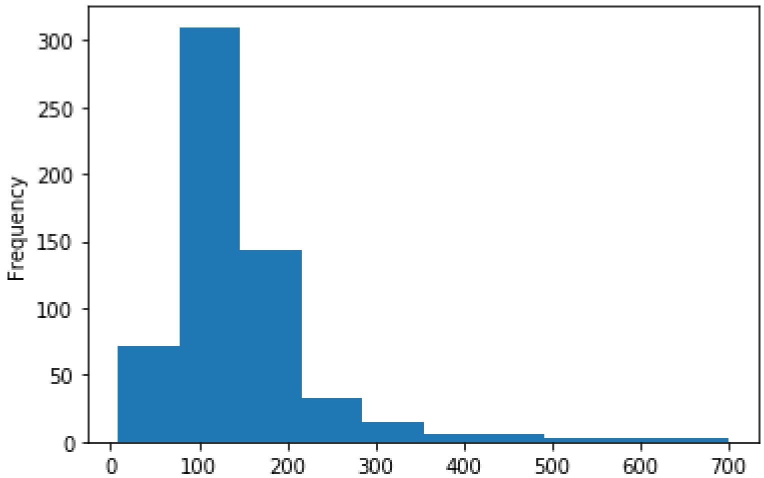


Or,

```
import pandas as pd

import matplotlib.pyplot as plt

df=pd.read_csv('data_python.csv')

#histogram of loan amount

plt.hist(x=df['LoanAmount'])

plt.show()
```

o/p:

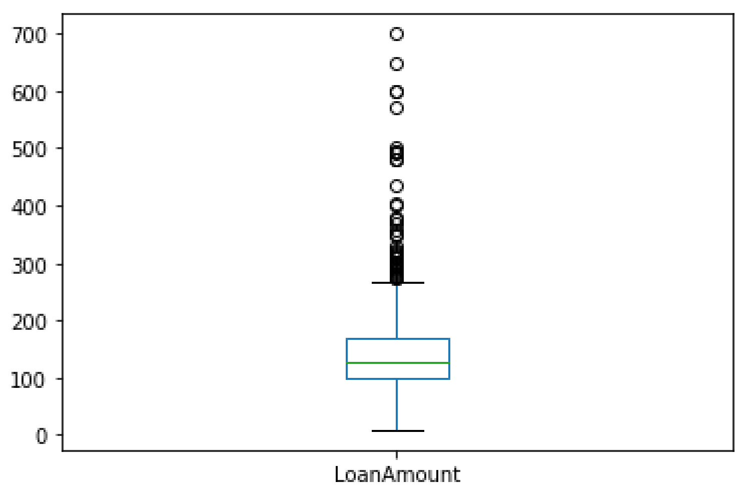
<matplotlib.axes._subplots.AxesSubplot at 0x46dc2df688>



Plotting boxplot:

```
import pandas as pd

df=pd.read_csv('data_python.csv')

#plotting box_plot of loan amount

df['LoanAmount'].plot.box()
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x46dc38db08>



Univariate analysis for categorical variable:

Tabular method:

a.creating a frequency table for a categorical variable sex

```
import pandas as pd

df=pd.read_csv('data_python.csv')

#creating freequency table for caregorical variable sex

df['Gender'].value_counts()
```

o/p:

```
Male      489
Female    112
Name: Gender, dtype: int64
```

CREATING A PERCENTAGE FREQUENCY TABLE FOR GENDER:

```
import pandas as pd
df=pd.read_csv('data_python.csv')
#creating  freequency% table  for caregorical variable sex
df['Gender'].value_counts()/len(df['Gender'])
```

o/p:

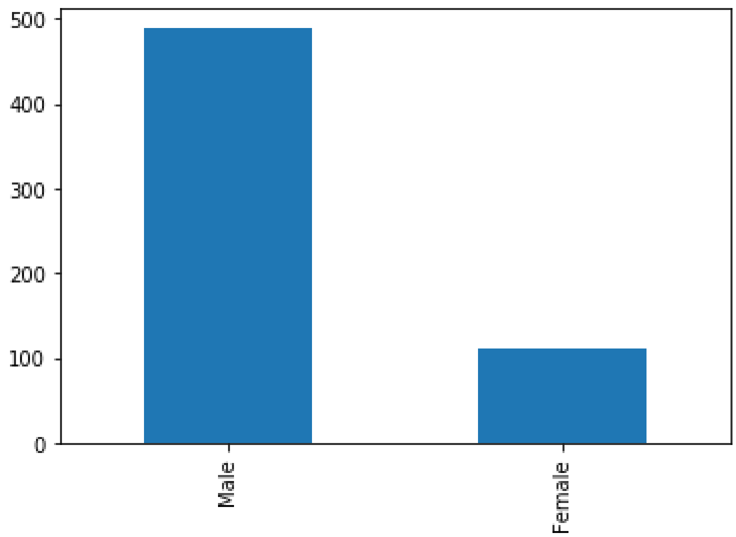
```
Male      0.796417
Female    0.182410
Name: Gender, dtype: float64
```

GRAPHICA METHOD:

```
import pandas as pd

df=pd.read_csv('data_python.csv')

#creating bar plot freequency  for caregorical variable sex

df['Gender'].value_counts().plot.bar()
```

o/p:

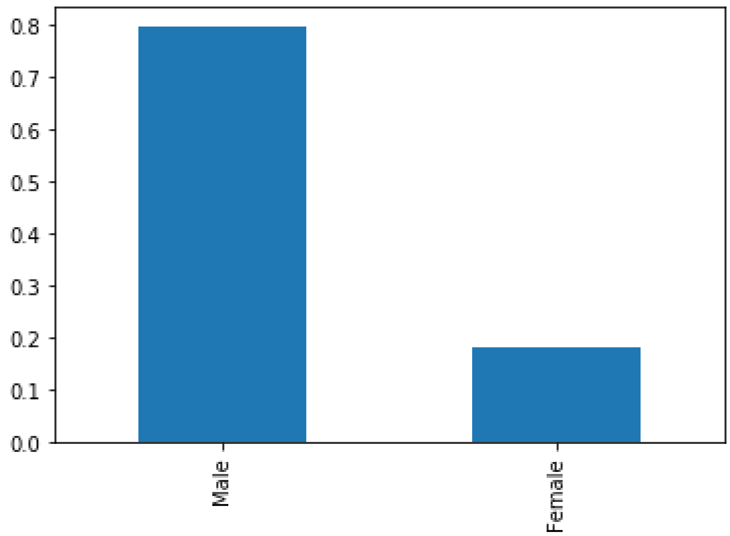
```
<matplotlib.axes._subplots.AxesSubplot at 0x46dc758c88>
```



CREATING A BAR PLOT OF PERCENTAGE FREEQUENCY OF GENDRER:

```
import pandas as pd

df=pd.read_csv('data_python.csv')

#creating  barplot freequency%   for caregorical variable sex

(df['Gender'].value_counts()/len(df['Gender'])).plot.bar()
```

: <matplotlib.axes._subplots.AxesSubplot at 0x46dc81fac8>



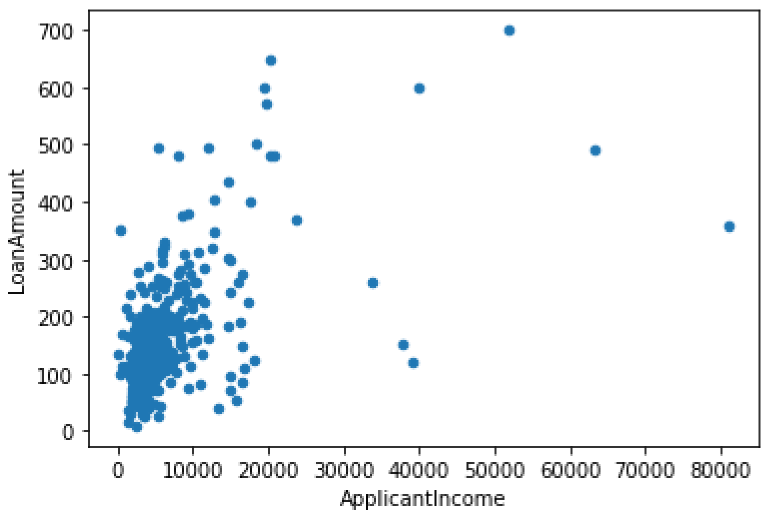
BIVERIATE ANALYSIS:

1.CONTINIOUS-CONTINIOUS VARIABLE:

Scatter plot

```
import pandas as pd

import matplotlib.pyplot as plt

df=pd.read_csv('data_python.csv')

df.plot.scatter('ApplicantIncome','LoanAmount')
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x8ae20ee308>



Correlation between applicant income and loan amount in tabular form

```
import pandas as pd

import matplotlib.pyplot as plt

df=pd.read_csv('data_python.csv')

print(df[['LoanAmount','ApplicantIncome']].corr())
```

o/p:

	LoanAmount	ApplicantIncome
LoanAmount	1.000000	0.570909
ApplicantIncome	0.570909	1.000000

Correlation between all continuous variable:

```
import pandas as pd

import matplotlib.pyplot as plt

df=pd.read_csv('data_python.csv')

df.corr()
```

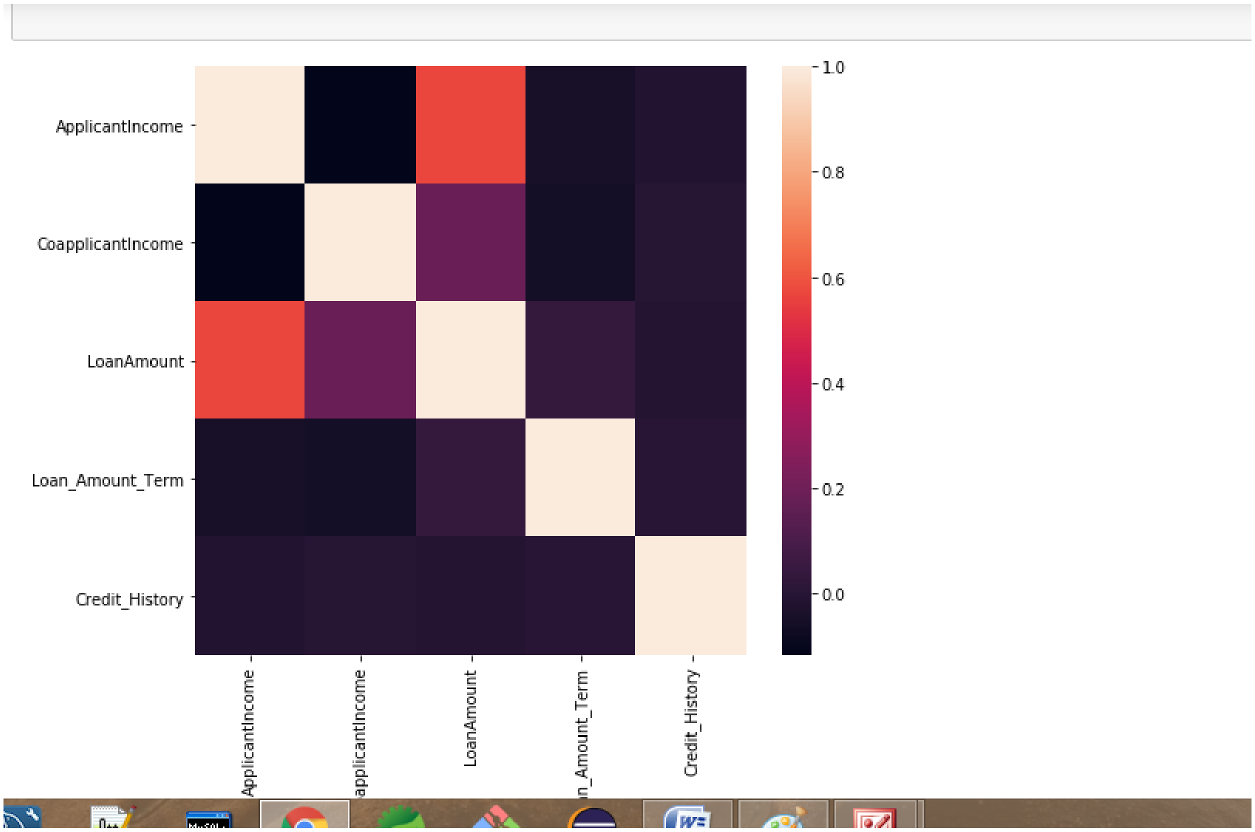
o/p:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306	-0.014715
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878	-0.002056
LoanAmount	0.570909	0.188619	1.000000	0.039447	-0.008433
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000	0.001470
Credit_History	-0.014715	-0.002056	-0.008433	0.001470	1.000000

Heatmap to find correlation between all data:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df=pd.read_csv('data_python.csv')

corr=df.corr()

plt.figure(figsize=(8,7))

sns.heatmap(corr)

plt.show()
```

o/p:



Correlation between applicant income and loan amount:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df=pd.read_csv('data_python.csv')

df['ApplicantIncome'].corr(df['LoanAmount'])
```

o/p:

0.5709090389885667

BIVARIATE ANALYSIS OF CONTINIOOUS CATEGORICAL VARIABLE:

Tabular method

Find the mean income of male and female:

```
import pandas as pd

import matplotlib.pyplot as plt

df=pd.read_csv('data_python.csv')

#FIND THE MEAN INCOME OF FEMALE AND MALE

df.groupby('Gender')['ApplicantIncome'].mean()
```

o/p:

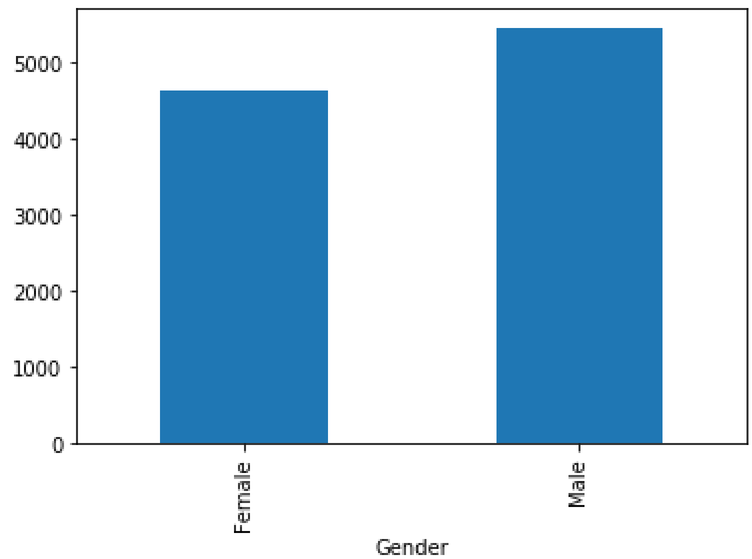
```
Gender
Female    4643.473214
Male      5446.460123
Name: ApplicantIncome, dtype: float64
```

GRAPHICAL REPRESENTATION:

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('data_python.csv')
#FIND THE MEAN INCOME OF FEMALE AND MALE USING BARPLOT
df.groupby('Gender')['ApplicantIncome'].mean().plot.bar()
```

o/p:

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x8ae50efb08>



Two t sample test of mean income of male and female:

```
import pandas as pd

import matplotlib.pyplot as plt

import scipy.stats as stants

from scipy.stats import ttest_ind

df=pd.read_csv('data_python.csv')

male=df[df['Gender']=='Male']

female=df[df['Gender']=='Female']

t_statistic,p_value=ttest_ind(male['ApplicantIncome'],female['ApplicantIncome'],nan_po
licy='omit')

print(t_statistic)

#as p_value>0.05, we can say that mean income of male and female stistically
indifferent
```

o/p

1.3232838129163134

Bivariate analysis of categorical vs categorical variable:

Two way table:

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stants
from scipy.stats import ttest_ind
df=pd.read_csv('data_python.csv')
m=pd.crosstab(df['Gender'],df['Self_Employed'])
print(m)
```

o/p:

```
Self_Employed   No  Yes
Gender
Female           89   15
Male            402   63
```

Chi-squre test of contingency:

```
import pandas as pd

import matplotlib.pyplot as plt

import scipy.stats as stants

from scipy.stats import ttest_ind

from scipy.stats import chi2_contingency

df=pd.read_csv('data_python.csv')

chi2_contingency(pd.crosstab(df['Gender'],df['Self_Employed']))

# p_value 0.93 very much greater than 0.05,so there no effect of gendeder on
unemployment
```

o/p:

```
(0.005893176199177382,
 0.9388088831667365,
 1,
 array([[ 89.74340949,  14.25659051],
        [401.25659051,  63.74340949]]))
```

IDENTIFYING MISSING VALUE:

```
import pandas as pd
df=pd.read_csv('data_python.csv')
print(df.shape)
print('\n')
df.describe()
```

o/p:

(614, 13)

Out[4]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In shape of the data baseit show there is showing 614 rows but in loan_amount_term there is 600 count show 14 value is missing

Missing value for all varia ble:

```
import pandas as pd

df=pd.read_csv('data_python.csv')

df.isnull()
```

o/p:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	False	False	False	False	False	False	False	False	True	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
609	False	False	False	False	False	False	False	False	False	False	False
610	False	False	False	False	False	False	False	False	False	False	False
611	False	False	False	False	False	False	False	False	False	False	False
612	False	False	False	False	False	False	False	False	False	False	False
613	False	False	False	False	False	False	False	False	False	False	False

614 rows × 13 columns

Or,

```
import pandas as pd

df=pd.read_csv('data_python.csv')

df.isnull().sum()
```

o/p:

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

in gender there is 13 value missing and self_employed there is 32 value missing.

```
import pandas as pd
df=pd.read_csv('data_python.csv')
#DROP ALL ROW WITH MISSING VALUE
df.dropna()#HERE DROPNA() KEYWORD DROP ALL THE MISSING VALUE
#TO CONFIRM THAT WE USE
df.dropna().isnull().sum()
#one thing we have to confirm here that
#here df.dropna() keyword keep the copy of df database with no null
#to change in main df data use
#d=f=df.dropna()
```

o/p:

```
Loan_ID              0
Gender               0
```

```
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

TO DELATE ROW WITH ALL MISSING VALUE

```
import pandas as pd
df=pd.read_csv('data_python.csv')
#DROP  ROW WITH ALL ENTRIES ARE MISSING VALUE
df.dropna(how='all')
# TO CHEAK THE SHAPE
df.dropna(how='all').shape
# here is  row show no row are deleted
```

O/P:

```
(614, 13)
```

Drop the column awith missing value

```
import pandas as pd
df=pd.read_csv('data_python.csv')
#DROP COLUMN WITH MISSING VALUE
df.dropna(axis=1)
#to cheack the shape
df.dropna(axis=1).shape
```

o/p:

```
(614, 6)
```

TO DROP THE COLUMN WHICH CONTAIN ALL THE MISSING VALUE

```
import pandas as pd
df=pd.read_csv('data_python.csv')
#TO DROP COLUMN WITH ALL THE ENTRIES CONTAIN MISSING VALUE
df.dropna(axis=1,how='all')
#TO CHEAK THE SHAPE
df.dropna(axis=1,how='all').shape
#here row column with all the missing value so,shape has no difference
with original data
```

O/P:

```
(614, 13)
```

FILL ALL THE MISSING VALUE WITH 0:

```
#FILL ALL THE MISSING VALUE WITH
import pandas as pd
df=pd.read_csv('data_python.csv')
df.fillna(0)
```

O/P:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histor
0	LP001002	Male	No	0	Graduate	No	5849	0.0	0.0	360.0	1.
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.
...	...	...	...	...	...	...	...	...	...	...	.
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.

FILL THE MISSING VALUE OF A PERTICULAR COLUMN

```
#FILL THE MISSING VALUE OF THE PERTICULAR COLOUMN WITH 0
import pandas as pd
df=pd.read_csv('data_python.csv')
df['LoanAmount'].fillna(0)
#fill all the missing value OF LOANAMOUNT
```

O/P:

```
0        0.0
1      128.0
2       66.0
3      120.0
4      141.0
       ...
609     71.0
610     40.0
611    253.0
612    187.0
613    133.0
Name: LoanAmount, Length: 614, dtype: float64
```

FILL THE MISSING VALUE OF A PERTICULAR COLUMN WITH MEAN VALUE of that column

```
#FILL THE MISSING VALUE OF THE PERTICULAR COLOUMN WITH THE MEAN VALUE OF
THAT COLUMN
import pandas as pd
df=pd.read_csv('data_python.csv')
df['LoanAmount'].fillna(df['LoanAmount'].mean())
#fill all the missing value OF LOANAMOUNT

o/p:

0      146.412162
1      128.000000
2       66.000000
3      120.000000
4      141.000000
          ...
609     71.000000
610     40.000000
611    253.000000
612    187.000000
613    133.000000
Name: LoanAmount, Length: 614, dtype: float64
```

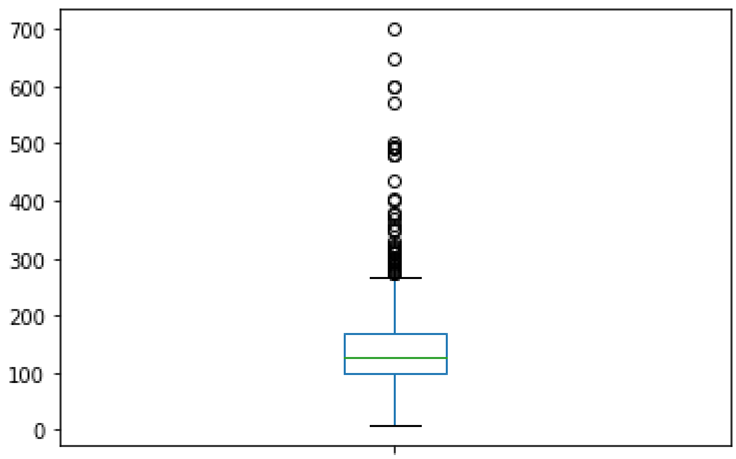
FILL THE MISSING VALue of PERTICULAR COLUMN:

```
#FILL  THE PERTICULAR COLUMN VALUE WITH  MISSING VALUE
import pandas as pd
df=pd.read_csv('data_python.csv')
df['Loan_Amount_Term'].fillna(0)
#fill all the missing value

o/p:

0      360.0
1      360.0
2      360.0
3      360.0
4      360.0
       ...
609    360.0
610    180.0
611    360.0
612    360.0
613    360.0
Name: Loan_Amount_Term, Length: 614, dtype: float64
```

TREATMENT OF UNIVARIATE VOUTLIER

```
# TREETMENT OF OUTLIER IN UNIVARIATE VARIABLE
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('data_python.csv')
df['LoanAmount'].plot.box()
plt.show()
```

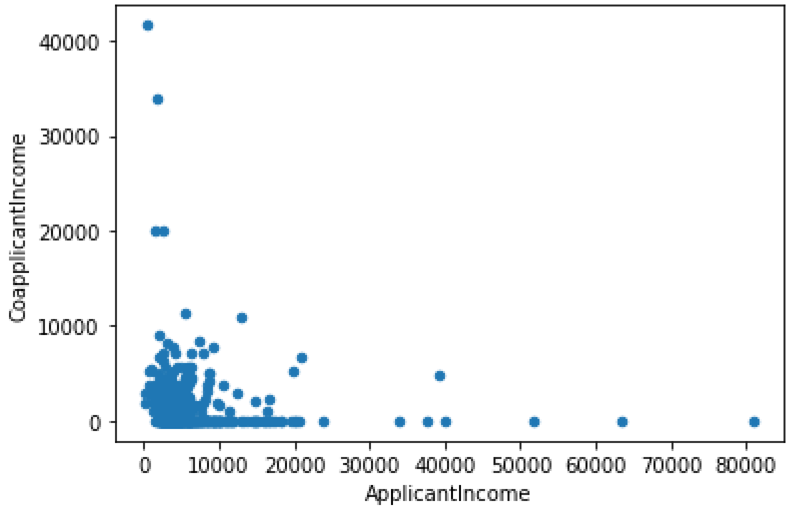
O/P:



TREATE OF OUTLIER IN BI VARIATE VARIABLE;

```
# TREETMENT OF OUTLIER IN BIVARIATE
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('data_python.csv')
df.plot.scatter('ApplicantIncome','CoapplicantIncome')
plt.show()
```

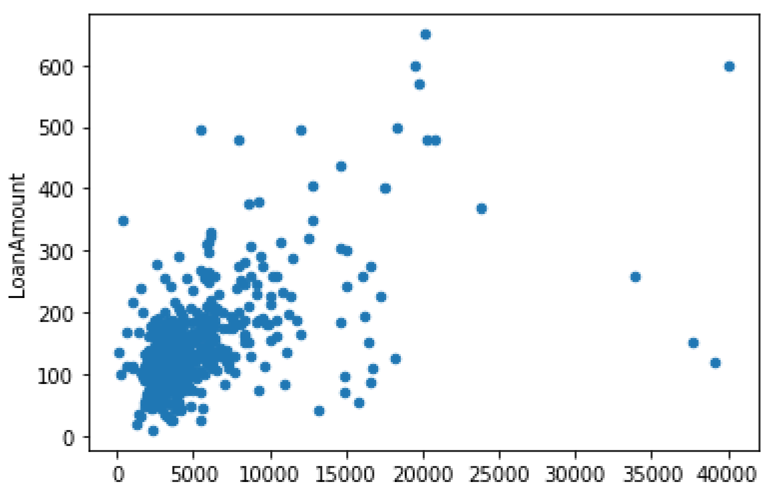
O/P:



Reomoving outlier in bivariate dataset:

```
#REMOVING OUTLIER FROM DATASET
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('data_python.csv')
df=df[df['ApplicantIncome']<40000]
df.plot.scatter('ApplicantIncome','LoanAmount')
#HERE WE REMOVE THE OUR LIER WHICH HAVE INCOME GREATER THAN 40000
#TO VARIFY THIS WE USE SCATTER PLOT
```

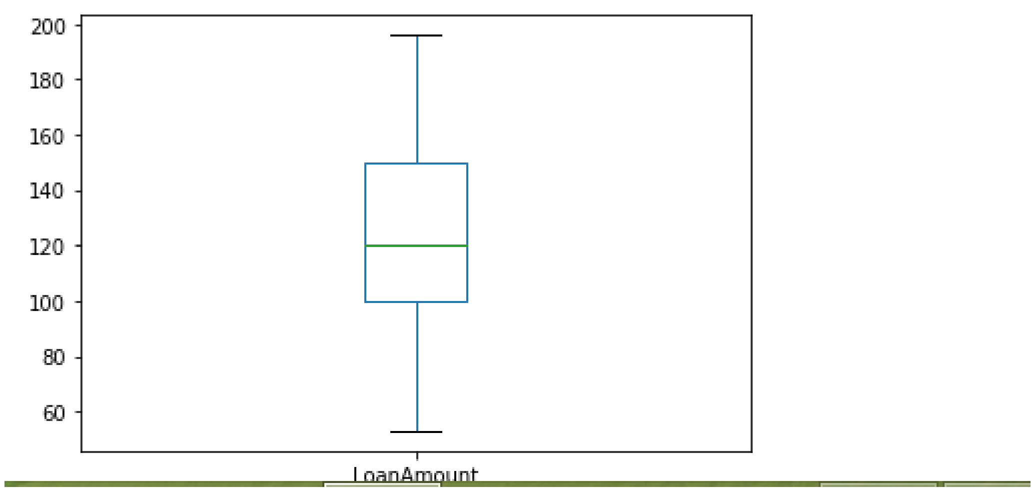
```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x4ceb365888>
```



REMOVING THE OUTLIER IN UNIVARIATE VARIABLE:

```
# TREETMENT OF OUTLIER IN UNIVARIATE
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('data_python.csv')
df=df[df['LoanAmount']<200]
df=df[df['LoanAmount']>50]

df['LoanAmount'].plot.box()
plt.show()
```

O/P:

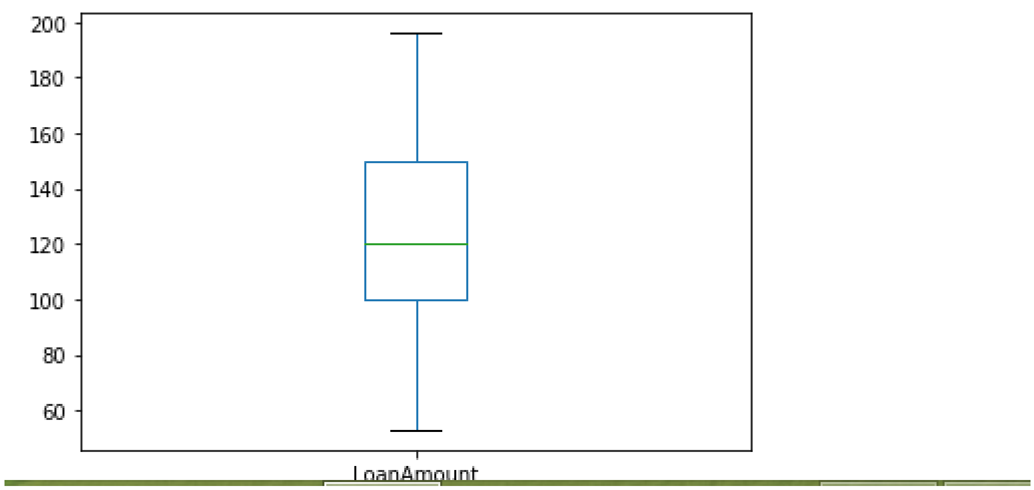


OR,

```
# TREETMENT OF OUTLIER IN BIVARIATE
import pandas as pd
import matplotlib.pyplot as plt
```

```
df=pd.read_csv('data_python.csv')

df['LoanAmount']=df.loc[df['LoanAmount']<200,'LoanAmount']

df['LoanAmount']=df.loc[df['LoanAmount']>50,'LoanAmount']

df['LoanAmount'].plot.box()

plt.show()
```

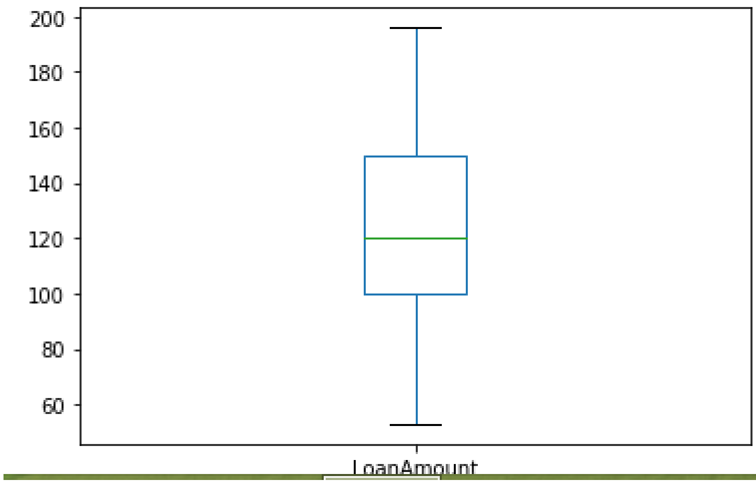
O/P:



REPLACE THE OUTLIER WITH MEAN VALUE

 # TREETMENT OF OUTLIER IN BIVARIATE with mean value

```
import pandas as pd

import matplotlib.pyplot as plt

df=pd.read_csv('data_python.csv')

df.loc[df['LoanAmount']>200,'LoanAmount']=df['LoanAmount'].mean()

df.loc[df['LoanAmount']<50,'LoanAmount']=df['LoanAmount'].mean()

df['LoanAmount'].plot.box()

plt.show()
```

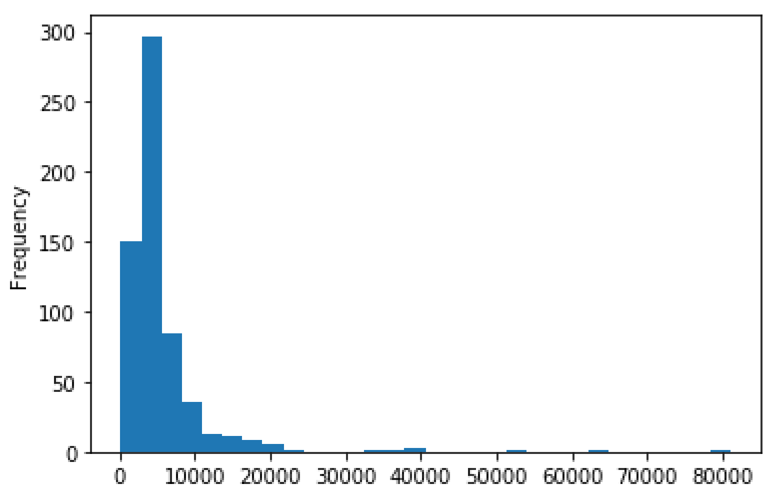


TO CHECK DISTRIBUTION OF A VARIABLE:

```
#TO CHEAK DISTRIBUTION OF A VARIABLE
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\Module Test 1.csv')
df['ApplicantIncome'].plot.hist(bins=30)
```

O/P:

```
55]:  <matplotlib.axes._subplots.AxesSubplot at 0x4ceba04708>
```

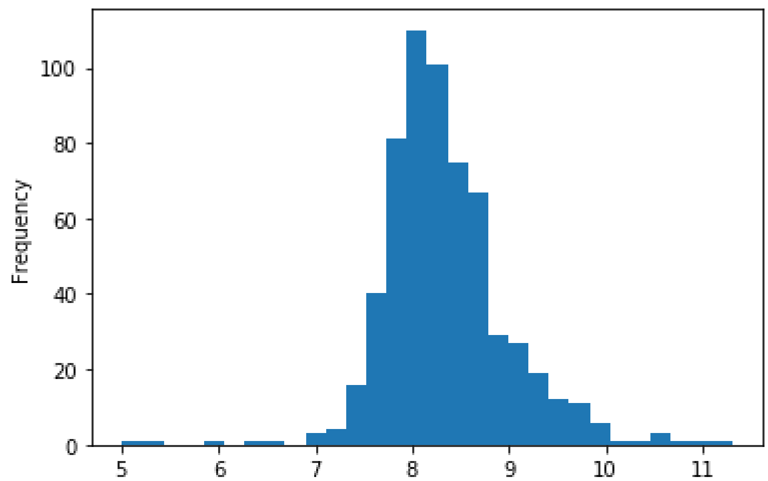


LOG TRANS FORMATION OF A VARIABLE:

```
#LOG OF A VARIABLE TRANS FORMATION OF A VARIABLE
import pandas as pd
import numpy as np
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\Module Test 1.csv')
np.log(df['ApplicantIncome']).plot.hist(bins=30)
```

O/P:

```
Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x4ceb9d1e08>
```

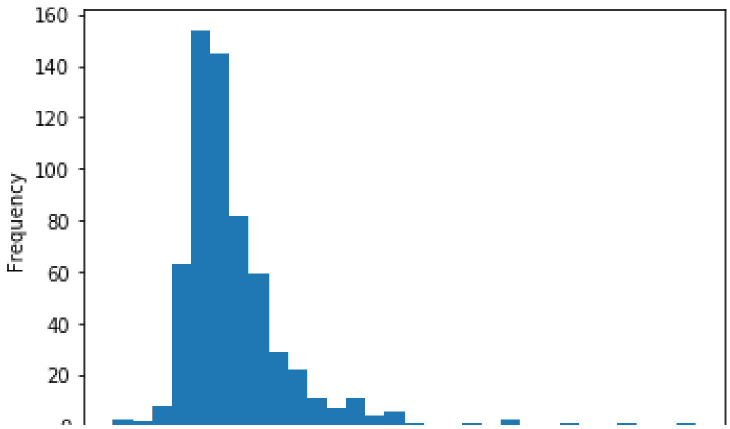


SQRT TRANSFORMATION OF A VARIABLE:

```
import pandas as pd
import numpy as np
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\Module Test 1.csv')
np.sqrt(df['ApplicantIncome']).plot.hist(bins=30)
```

O/P:

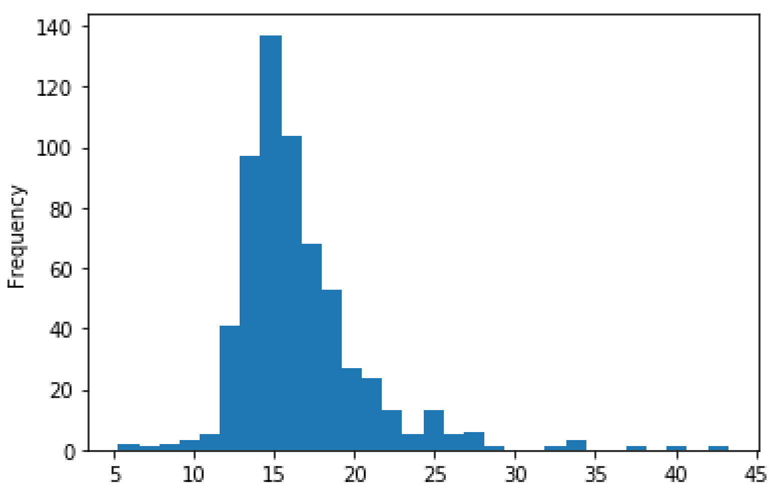
```
Out[68]: <matplotlib.axes._subplots.AxesSubplot at 0x4cebbd1408>
```



CUBE ROOT TRANSFORMATION OF A VARIABLE:

```
import pandas as pd

import numpy as np

df=pd.read_csv(r'C:\Users\Shubhamay\Documents\Module Test 1.csv')

np.power(df['ApplicantIncome'],1/3).plot.hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x4cebd509c8>
```



BIN DATA TRANSFORMATION:

```
# data TRANS FORMATION WITH BIN OF A VARIABLE

import pandas as pd

import numpy as np

df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')

bins=(0,60,80,95)

groups=['good','very good','excellent']

df['type']=pd.cut(df['marks'],bins,labels=groups)

print(df)

#you can see another row type is created cotainind grade
```

O/P:

```
     name  marks favourite subject       type
0   shubha     91           physics  excellent
1    sujit     85           biology  excellent
2      ram     87           physics  excellent
3     jadu     62         chemistry  very good
4    madhu     75         chemistry  very good
5    virat     67           biology  very good
6   sachin     68           physics  very good
7   surwsh     78            bangla  very good
8   mukesh     73           English  very good
9    jibon     78           English  very good
10    babu     65           physics  very good
11    anis     89              math  excellent
12  mukesk     57           English       good
13   patas     85         chemistry  excellent
14   anish     68           English  very good
15   began     68         chemistry  very good
```

<u>You can also cheak the frequency:</u>

```
# data TRANS FORMATION WITH BIN OF A VARIABLE
```

```
import pandas as pd

import numpy as np

df=pd.read_csv(r'C:\Users\Shubhamay\Documents\fav.csv')

bins=(0,60,80,95)

groups=['good','very good','excellent']

df['type']=pd.cut(df['marks'],bins,labels=groups)

df['type'].value_counts()

#you can also cheak the frequency
```

o/p:

```
very good    10
excellent     5
good          1
Name: type, dtype: int64
```

LINEAR REGRESSION

```
#we working on our first linear regression of a big mart
#we want to predict the sale of a big mart
#target is continious variable so we use want to use linear regression
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LinearRegression
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\train.csv')
df=pd.get_dummies(df)
train=df[0:7999]#train dataset
test=df[8000:]# test dataset
# in shape of the data there is 8523 rows
#we take 8000 rows as train dataset and remaining as test dataset
x_train=train.drop('Item_Outlet_Sales',axis=1)#drop target variable from
#train data set
y_train=train['Item_Outlet_Sales']#we seperate dependent and independ
#variable in train data set
x_test=test.drop('Item_Outlet_Sales',axis=1)
test_p=test['Item_Outlet_Sales']#we keep target variable in test data set for
#true prediction
lreg=LinearRegression()# creating object of linear regression
#one thing about linear regression is,it fist create an object then model are
#fitted to that object
x_train=pd.get_dummies(x_train)#dummyfication of train dataset
x_test=pd.get_dummies(x_test)#dummyfication of test data set
```

```
x_train.fillna(0,inplace=True)
x_test.fillna(0,inplace=True)
lreg.fit(x_train,y_train)#fit model to lreg
pred=lreg.predict(x_test)#made prediction on test dataset
lreg.score(x_test,test_p)#compute r squre value of test dataset compare with
#true prediction
lreg.score(x_train,y_train)#compute r squre value of train dataset
#either we can see that model is over fitted to train data set
#nor test dataset is not suitable representative of train data set
rmse_test=np.sqrt(np.mean(np.power((np.array(test_p)-np.array(pred)),2)))
rmse_train=np.sqrt(np.mean(np.power((np.array(y_train)-
np.array(lreg.predict(x_train))),2)))
print('sale prediction by test dataset',rmse_test)
print('\n')
print('sale prediction by train data set',rmse_train)
print(lreg.coef_)
print(lreg.intercept_)
```

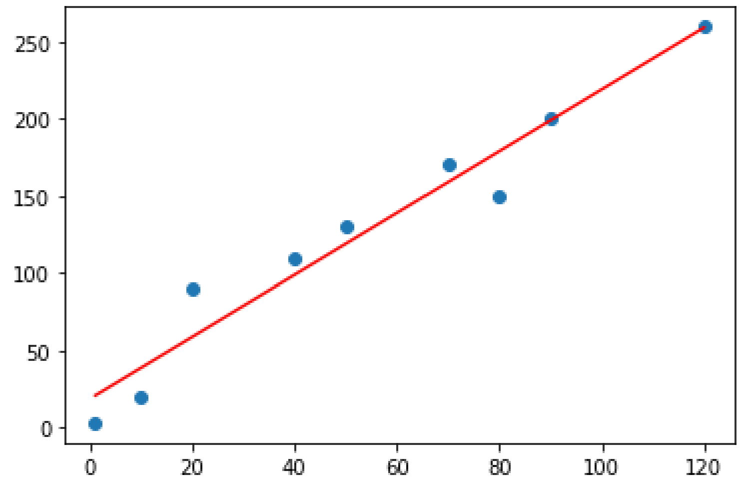
o/p:

```
sale prediction by test dataset 1254.3416428154303


sale prediction by train data set 1015.4723517117602

[-2.67825456e+00 -2.38243415e+02  1.16546920e+00 ...  9.81499251e+13
  2.60305007e+14 -2.35422746e+14]
8.47801690878073e+16
```

simple example of linear regression:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LinearRegression
data=pd.read_csv(r'C:\Users\Shubhamay\Documents\lregsimp.csv')
x=data['x']
y=data['y']
x=np.array(x).reshape(-1,1)
lreg.fit(x,y)
y_pred=lreg.predict(x)
plt.scatter(x,y)
plt.plot(x,y_pred,color='r')
plt.show()
print(lreg.coef_)
print(lreg.intercept_)
#the equation be y=2.004x+18.73
rmse=np.sqrt(np.mean(np.power((np.array(y)-np.array(y_pred)),2)))
print(rmse)
```

o/p:




```
[2.00488411]
18.738971360549
17.8004281629874
```

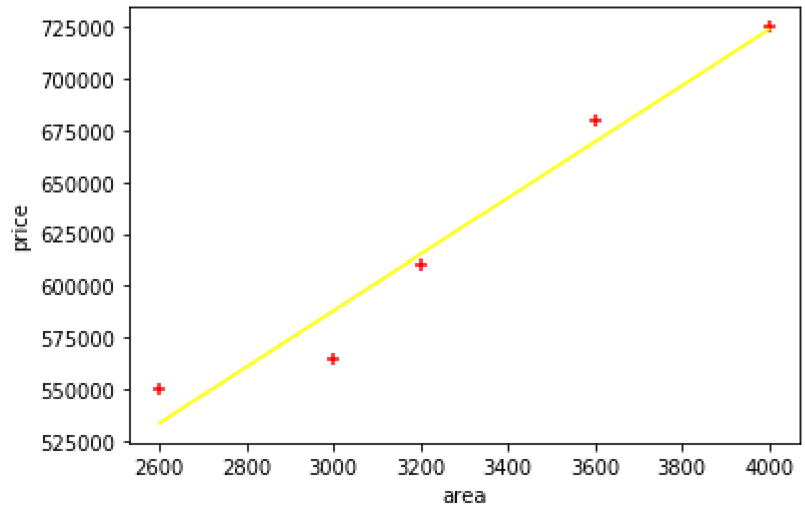
Or,

```
#GIVEN THIS HOME PRICE FIND PRICE OF HOMES WHOSE AREA IS 3300 S FEET AND
5000 SF
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LinearRegression
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\lreg_codebasics.csv')
lreg=LinearRegression()
lreg.fit(df[['area']],df['price'])
print('the value of home with 3300 squre feet:',lreg.predict([[3300]]))
print('the value of home with 5000 squre feet:',lreg.predict([[5000]]))
d=pd.read_csv(r'C:\Users\Shubhamay\Documents\area.csv')
#WE NOW A LOAD A CSV FILE WITH AREA
#AND NOW PREDICT THE PRICE AF THOSE AREA
p=lreg.predict(d)
d['predicted price']=p
print('\n')
print(d)# WE CAN SEE THE NEW TABLE WITH THE PREDICTED VALUE
d.to_csv('prediction.csv',index=False)
#NEW SV FILE NAME PREDICTION IS CREATED
#THE ADDRESS OF PREDICTION IS C:\Users\Shubhamay
plt.scatter(df['area'],df['price'],color='red',marker='+')
plt.plot(df['area'],lreg.predict(df[['area']]),color='yellow')
plt.xlabel('area')
plt.ylabel('price')
plt.show()
#we want to see how our scatter plot look like with predicted vale
```

o/p:

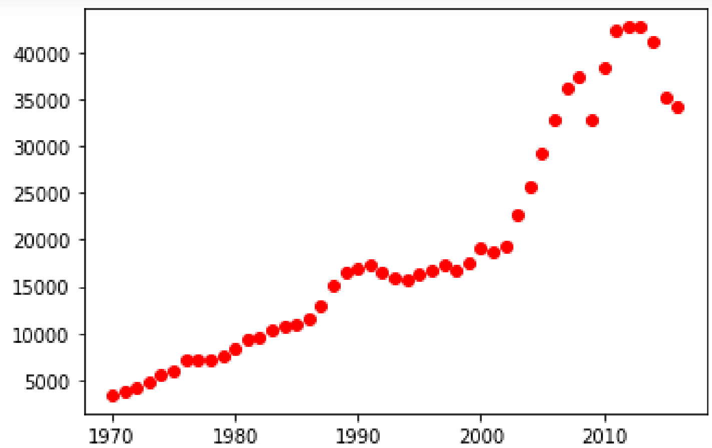
```
the value of home with 3300 squre feet: [628715.75342466]
```

```
the value of home with 5000 squre feet: [859554.79452055]


   area  predicted price
0  1000    316404.109589
1  2000    452191.780822
2  3500    655873.287671
3  3600    669452.054795
4  4000    723767.123288
```

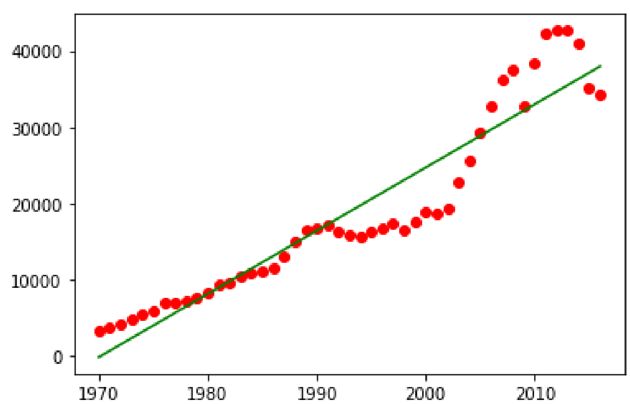


Or,

```
import pandas as p
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LinearRegression
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\canada_per_capita_income.csv
')
plt.scatter(df['year'],df['per capita income (US$)'],marker='o',color='r')
plt.show()
lreg=LinearRegression()
lreg.fit(df[['year']],df['per capita income (US$)'])
#income in 2020
print(lreg.predict([[2020]]))
#how much our prediction is correct.
print('\n')
print(lreg.score(df[['year']],df['per capita income (US$)']))
plt.scatter(df['year'],df['per capita income (US$)'],marker='o',color='r')
plt.plot(df.year,lreg.predict(df[['year']]),color='green')
plt.show()
```




```
[41288.69409442]


0.890916917957032
```



LOGISTIC REGRESSION

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LogisticRegression
df=pd.read_csv(r'F:\Chand\book,
resume,psim\PSIM\Books\Books\python\titanic.csv')
df['Survived'].value_counts()
df=pd.get_dummies(df)#convert to numerical figure instead of categorical
figure
df.fillna(0,inplace=True)
logreg=LogisticRegression(solver='liblinear',multi_class='ovr')
train=df[0:699]
test=df[700:890]
x_train=train.drop('Survived',axis=1)
y_train=train['Survived']
x_test=test.drop('Survived',axis=1)
```

```
test_p=test['Survived']
logreg.fit(x_train,y_train)
pred=logreg.predict(x_test)
print('accuracy of test dataset',logreg.score(x_test,test_p))
print('\n')
print('accuracy of train dataset',logreg.score(x_train,y_train))
```

o/p:

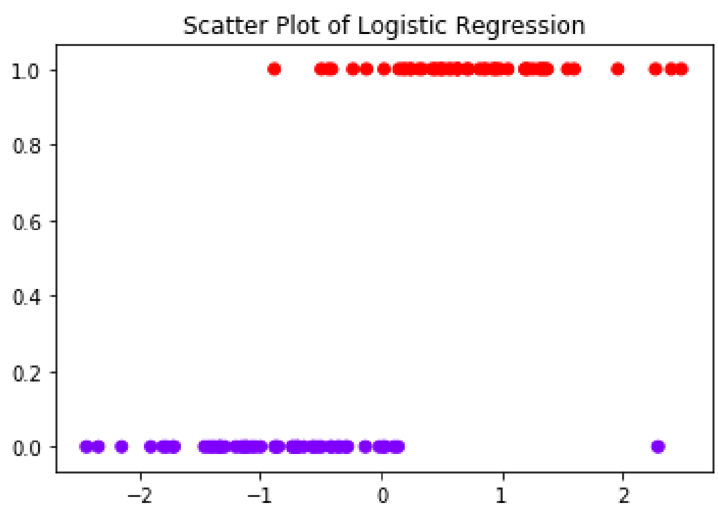
```
accuracy of test dataset 0.8210526315789474


accuracy of train dataset 0.9227467811158798
```

or,

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
# Generate and dataset for Logistic Regression
x, y = make_classification(
    n_samples=100,
    n_features=1,
    n_classes=2,
    n_clusters_per_class=1,
    flip_y=0.03,
    n_informative=1,
    n_redundant=0,
    n_repeated=0
)
# Create a scatter plot
plt.scatter(x, y, c=y, cmap='rainbow')
plt.title('Scatter Plot of Logistic Regression')
plt.show()
# Split the dataset into training and test dataset
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1)
# Create a Logistic Regression Object, perform Logistic Regression
log_reg = LogisticRegression(solver='liblinear')
log_reg.fit(x_train, y_train)
# Show to Coeficient and Intercept
```

```
print(log_reg.coef_)
print(log_reg.intercept_)
# Perform prediction using the test dataset
y_pred = log_reg.predict(x_test)
# Show the Confusion Matrix
confusion_matrix(y_test, y_pred)
#We can deduce from the confusion matrix that:

# True positive: 13 (upper-left) – Number of positives we predicted correctly
# True negative: 11(lower-right) – Number of negatives we predicted correctly
# False positive: 1 (top-right) – Number of positives we predicted wrongly
# False negative:  0(lower-left) – Number of negatives we predicted wrongly
```

o/p:




```
[[2.89125688]]
[0.28784129]
```

```
9]: array([[ 9,  3],
       [ 3, 10]], dtype=int64)
```

Or,

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\lpgreg_codebasics.csv')
plt.scatter(df['age'],df['bought_insurance'],marker='+',color='r')
```

```
x_train, x_test, y_train, y_test =
train_test_split(df[['age']],df['bought_insurance'],train_size=0.9)
#here df[['age']] is x and df['bought_insurance'] is y
#to make x multidimensional we use double bracket in age
print(x_test)
print('\n')
print(x_train)
print('\n')
print(y_test)
print('\n')
print(y_train)
print('\n')
log_reg = LogisticRegression(solver='liblinear')
log_reg.fit(x_train, y_train)
y_pred = log_reg.predict(x_test)
print(y_pred)
#from prediction we can see that older person bougt life insurance
# as 21   0
  #  45   1
    #50   1
log_reg.score(x_test,y_test)
print(log_reg.score(x_train,y_train))
#here we see that test is more fitted to the model
```

o/p:

```
    age
20   21
23   45
24   50



    age
10   18
21   26
17   58
2    47
25   54
11   28
4    46
26   23
0    22
7    60
1    25
```

```
16   25
9    61
14   49
8    62
18   19
6    55
3    52
15   55
22   40
5    56
13   29
19   18
12   27
```

```
20    0
23    1
24    1
Name: bought_insurance, dtype: int64
```

```
10    0
21    0
17    1
2     1
25    1
11    0
4     1
26    0
0     0
7     1
1     0
16    1
9     1
14    1
8     1
18    0
6     0
3     0
15    1
22    1
5     1
13    0
19    0
```

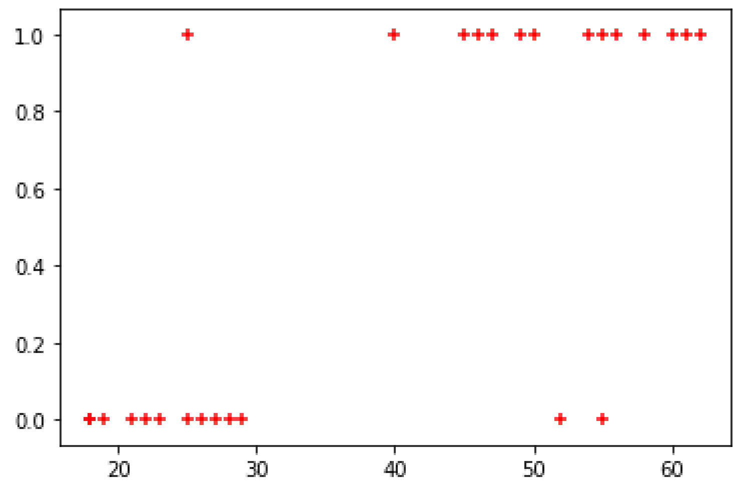
```
12    0
Name: bought_insurance, dtype: int64
```

```
[0 1 1]
```

Out[58]:

```
1.0
```

```
0.875
```



<u>NB: Mean-Squared-Error is used for LINEAR regression problems. For classification problems(LOGISTIC REGRESSION), we use AUC-ROC, Accuracy and Logloss.</u>

<u>DICISION TREE:</u>

*Top-most node of a decision tree is the root node and the bottom-most node is called the leaf node.

*Entropy value lies between 0 and 1

Or

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
df=pd.read_csv(r'F:\Chand\book, resume,psim\PSIM\Books\Books\python\data cleaned.csv')
x=df.drop(['Survived'],axis=1)
y=df['Survived']
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=101,stratify=y)
#by satisfy=y statement we can split into node which have equal percentage of x and y
#we can varify it by train_y.value_counts()/len(train) and
test_y.value_counts()/len(test)
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
print(clf.score(x_train,y_train))
print(clf.score(x_test,y_test))
```

o/p:

```
0.9880239520958084
0.7533632286995515
```

Or

```
#model to find individuals salary wheater it is greter than 100k or not=
#predict of to find the salary of a employee abc pharm,sales ececutive and
#bachelor
import pandas as pd
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\DecisionTree.csv')
inputs=df.drop(['salary_more_than_100k'],axis=1)
target=df['salary_more_than_100k']
print(inputs)
print('\n')
print(target)
#convert class lavel to encoder
from sklearn.preprocessing import LabelEncoder
le_company=LabelEncoder()
le_job=LabelEncoder()
le_degree=LabelEncoder()
inputs['company_n']=le_company.fit_transform(inputs['company'])
inputs['job_n']=le_company.fit_transform(inputs['job'])
inputs['degree_n']=le_company.fit_transform(inputs['degree'])
print('\n')
print(inputs)
inputs_n=inputs.drop(['company','job','degree'],axis=1)
print('\n')
print(inputs_n)
from sklearn.tree import DecisionTreeClassifier
dec=DecisionTreeClassifier()
dec.fit(inputs_n,target)
print(dec.score(inputs_n,target))
print(dec.predict([[0,2,0]]))
```

o/p:

```
       company                  job     degree
0       google      sales executive  bachelors
1       google      sales executive     master
2       google     business manager  bachelors
3       google     business manager     master
4       google  computer programming  bachelors
5       google  computer programming     master
6   abc pharma      sales executive     master
7   abc pharma  computer programming  bachelors
8   abc pharma     business manager  bachelors
9   abc pharma     business manager     master
10    facebook      sales executive  bachelors
11    facebook      sales executive     master
12    facebook     business manager  bachelors
13    facebook     business manager     master
14    facebook  computer programming  bachelors
```

```
15    facebook  computer programming     master


0     0
1     0
2     1
3     1
4     0
5     1
6     0
7     0
8     0
9     1
10    1
11    1
12    1
13    1
14    1
15    1
Name: salary_more_than_100k, dtype: int64


       company                   job     degree  company_n  job_n
degree_n
0       google       sales executive  bachelors          2      2
0
1       google       sales executive     master          2      2
1
2       google      business manager  bachelors          2      0
0
3       google      business manager     master          2      0
1
4       google  computer programming  bachelors          2      1
0
5       google  computer programming     master          2      1
1
6   abc pharma       sales executive     master          0      2
1
7   abc pharma  computer programming  bachelors          0      1
0
8   abc pharma      business manager  bachelors          0      0
0
9   abc pharma      business manager     master          0      0
1
10    facebook       sales executive  bachelors          1      2
0
11    facebook       sales executive     master          1      2
1
12    facebook      business manager  bachelors          1      0
0
13    facebook      business manager     master          1      0
1
14    facebook  computer programming  bachelors          1      1
0
```

```
15     facebook  computer programming     master           1       1
1


    company_n  job_n  degree_n
0           2      2         0
1           2      2         1
2           2      0         0
3           2      0         1
4           2      1         0
5           2      1         1
6           0      2         1
7           0      1         0
8           0      0         0
9           0      0         1
10          1      2         0
11          1      2         1
12          1      0         0
13          1      0         1
14          1      1         0
15          1      1         1
1.0
[0]
```

K-MEANS

Q2/3

What will be the centroid of the following points: (1,4), (2,0), (0,2)?

A (1,2)

Well done. Correct Answer.

Explanation:
Centroid will be : (1+2+0)/3 , (4+0+2)/3 = (1,2)

B (2,1)

C (3,6)

D (6,3)

K-Means uses euclidean distance and minimize it to assign an object to a cluster.

Or,

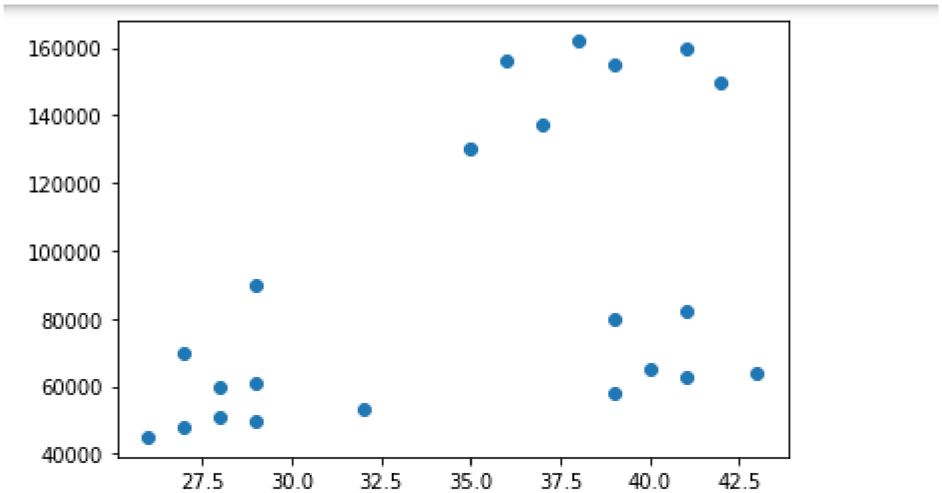
Kmeans algorithm

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

df=pd.read_csv(r'C:\Users\Shubhamay\Documents\salariesme.csv')

plt.scatter(df['age'],df['income'])

plt.show()

from sklearn.cluster import KMeans

from sklearn.preprocessing import MinMaxScaler

km=KMeans(n_clusters=3)

y_predict=km.fit_predict(df[['age','income']])

df['cluster']=y_predict

print(df)

df1=df[df['cluster']==0]

df2=df[df['cluster']==1]

df3=df[df['cluster']==2]


plt.scatter(df1['age'],df1['income'],color='green')

plt.scatter(df2['age'],df2['income'],color='red')

plt.scatter(df3['age'],df3['income'],color='yellow')


plt.xlabel('age')

plt.ylabel('income')
```

```
plt.show()

scaler=MinMaxScaler()

scaler.fit(df[['income']])

df[['income']]=scaler.transform(df[['income']])

scaler.fit(df[['age']])

df[['age']]=scaler.transform(df[['age']])

#after scalling

km=KMeans(n_clusters=3)

y_predict=km.fit_predict(df[['age','income']])

df['cluster']=y_predict

print(df)

df1=df[df['cluster']==0]

df2=df[df['cluster']==1]

df3=df[df['cluster']==2]



plt.scatter(df1['age'],df1['income'],color='green')

plt.scatter(df2['age'],df2['income'],color='red')

plt.scatter(df3['age'],df3['income'],color='yellow')

plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='b
lack',marker='+')


plt.xlabel('age')

plt.ylabel('income')

plt.show()

#elbow plot
```

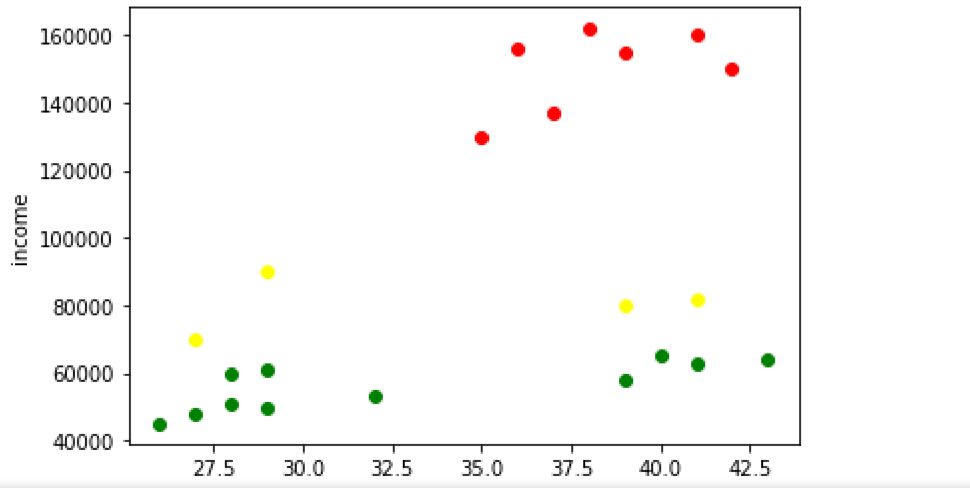
```
sse=[]
for k in range(1,10):
    km=KMeans(n_clusters=k)
    km.fit(df[['age','income']])
    sse.append(km.inertia_)
plt.plot(sse)
plt.show()
```

o/p:

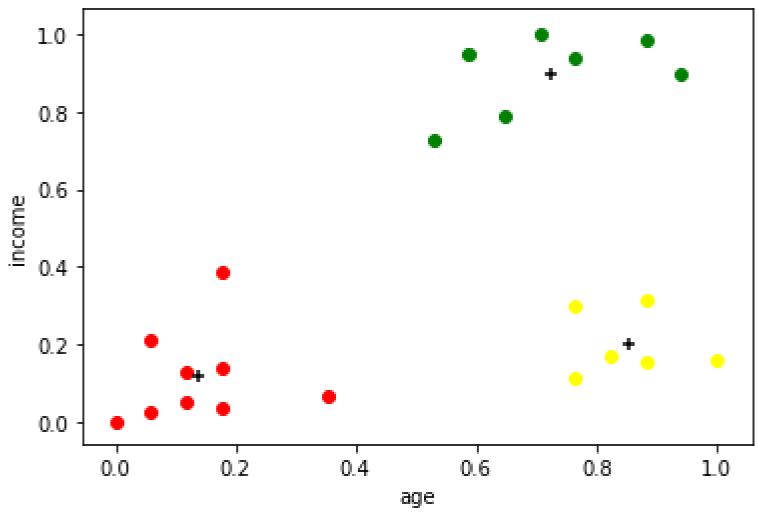


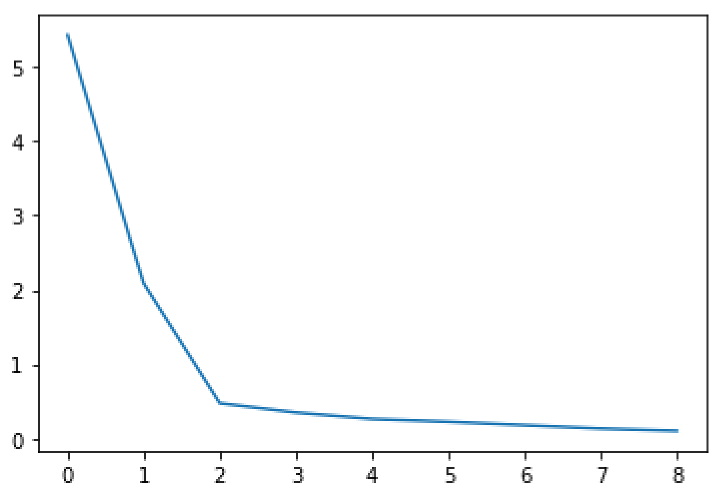
	name	age	income	cluster
0	rob	27	70000	2
1	micheal	29	90000	2
2	mohan	29	61000	0
3	ismail	28	60000	0
4	kory	42	150000	1
5	goutam	39	155000	1
6	david	41	160000	1
7	andrea	38	162000	1
8	brad	36	156000	1
9	angelina	35	130000	1
10	donaid	37	137000	1
11	tom	26	45000	0
12	arnoid	27	48000	0

13	jared	28	51000	0
14	stark	29	49500	0
15	ranbir	32	53000	0
16	dipika	40	65000	0
17	priyanka	41	63000	0
18	nick	43	64000	0
19	alia	39	80000	2
20	sid	41	82000	2
21	abdul	39	58000	0



	name	age	income	cluster
0	rob	0.058824	0.213675	1
1	micheal	0.176471	0.384615	1
2	mohan	0.176471	0.136752	1
3	ismail	0.117647	0.128205	1
4	kory	0.941176	0.897436	0
5	goutam	0.764706	0.940171	0
6	david	0.882353	0.982906	0
7	andrea	0.705882	1.000000	0
8	brad	0.588235	0.948718	0
9	angelina	0.529412	0.726496	0
10	donaid	0.647059	0.786325	0
11	tom	0.000000	0.000000	1
12	arnoid	0.058824	0.025641	1
13	jared	0.117647	0.051282	1
14	stark	0.176471	0.038462	1
15	ranbir	0.352941	0.068376	1
16	dipika	0.823529	0.170940	2
17	priyanka	0.882353	0.153846	2
18	nick	1.000000	0.162393	2
19	alia	0.764706	0.299145	2
20	sid	0.882353	0.316239	2
21	abdul	0.764706	0.111111	2





Or,

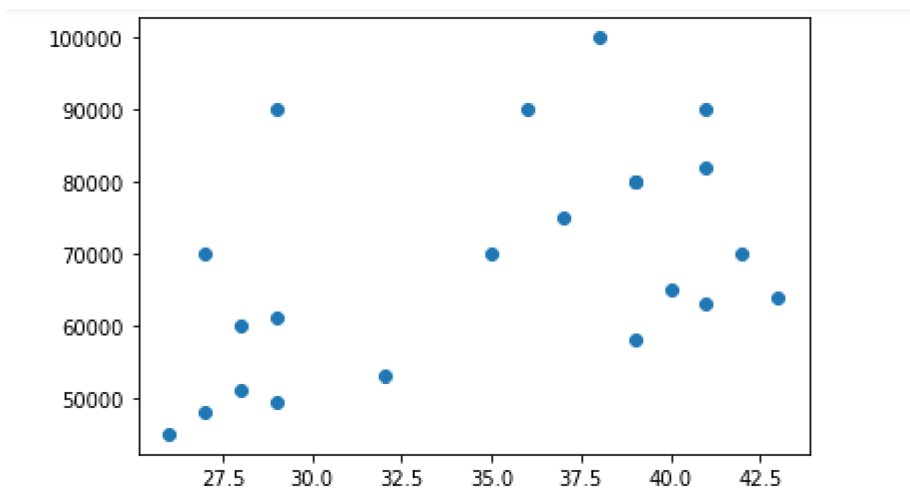
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

df=pd.read_csv(r'C:\Users\Shubhamay\Documents\salariesme.csv')
```

```
plt.scatter(df['age'],df['income'])

plt.show()

from sklearn.cluster import KMeans

from sklearn.preprocessing import MinMaxScaler

km=KMeans(n_clusters=5)

y_predict=km.fit_predict(df[['age','income']])

df['cluster']=y_predict

print(df)

df1=df[df['cluster']==0]

df2=df[df['cluster']==1]

df3=df[df['cluster']==2]

df4=df[df['cluster']==3]

df5=df[df['cluster']==5]


plt.scatter(df1['age'],df1['income'],color='green')

plt.scatter(df2['age'],df2['income'],color='red')

plt.scatter(df3['age'],df3['income'],color='yellow')

plt.scatter(df4['age'],df4['income'],color='blue')

plt.scatter(df5['age'],df5['income'],color='brown')


plt.xlabel('age')

plt.ylabel('income')


plt.show()

scaler=MinMaxScaler()

scaler.fit(df[['income']])

df[['income']]=scaler.transform(df[['income']])
```

```
scaler.fit(df[['age']])

df[['age']]=scaler.transform(df[['age']])

#after scalling

km=KMeans(n_clusters=5)

y_predict=km.fit_predict(df[['age','income']])

df['cluster']=y_predict

print(df)

df1=df[df['cluster']==0]

df2=df[df['cluster']==1]

df3=df[df['cluster']==2]

df4=df[df['cluster']==3]

df5=df[df['cluster']==5]



plt.scatter(df1['age'],df1['income'],color='green')

plt.scatter(df2['age'],df2['income'],color='red')

plt.scatter(df3['age'],df3['income'],color='yellow')

plt.scatter(df4['age'],df4['income'],color='blue')

plt.scatter(df5['age'],df5['income'],color='brown')

plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='b
lack',marker='+')


plt.xlabel('age')

plt.ylabel('income')

plt.show()

#elbow plot

sse=[]
```

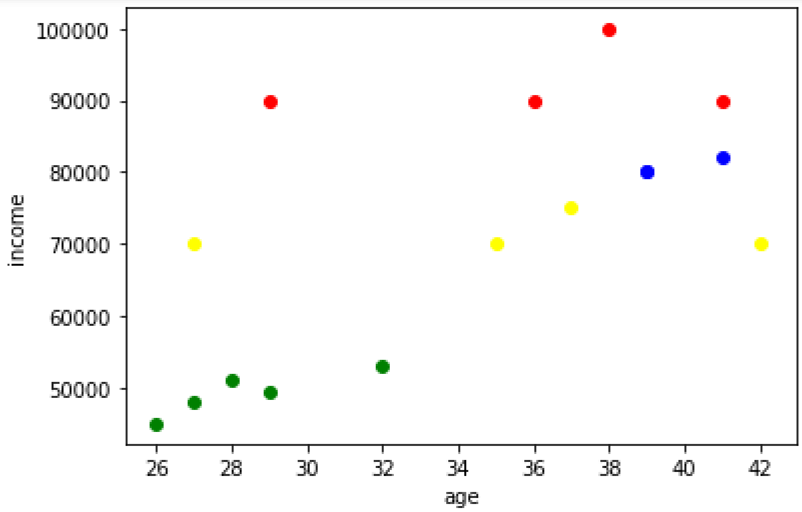
```
for k in range(1,10):
    km=KMeans(n_clusters=k)
    km.fit(df[['age','income']])
    sse.append(km.inertia_)
plt.plot(sse)
plt.show()
```

o/p:

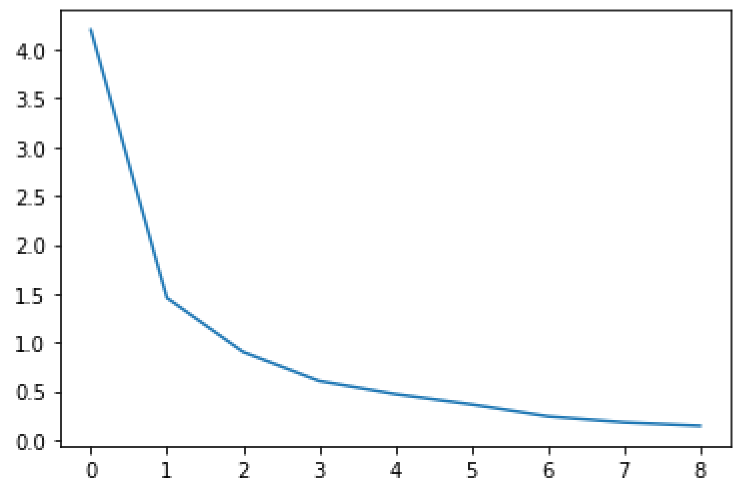


	name	age	income	cluster
0	rob	27	70000	2
1	micheal	29	90000	1
2	mohan	29	61000	4
3	ismail	28	60000	4
4	kory	42	70000	2
5	goutam	39	80000	3
6	david	41	90000	1
7	andrea	38	100000	1
8	brad	36	90000	1
9	angelina	35	70000	2
10	donaid	37	75000	2
11	tom	26	45000	0
12	arnoid	27	48000	0
13	jared	28	51000	0
14	stark	29	49500	0
15	ranbir	32	53000	0

16	dipika	40	65000	4
17	priyanka	41	63000	4
18	nick	43	64000	4
19	alia	39	80000	3
20	sid	41	82000	3
21	abdul	39	58000	4



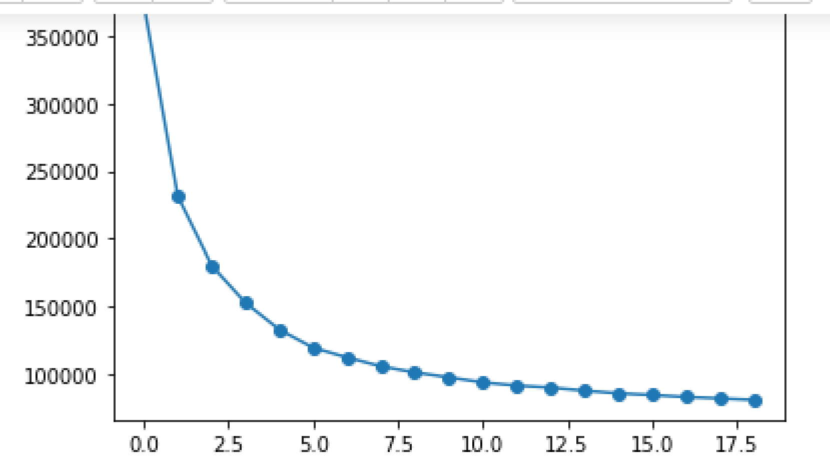
	name	age	income	cluster
0	rob	0.058824	0.454545	3
1	micheal	0.176471	0.818182	3
2	mohan	0.176471	0.290909	0
3	ismail	0.117647	0.272727	0
4	kory	0.941176	0.454545	2
5	goutam	0.764706	0.636364	4
6	david	0.882353	0.818182	1
7	andrea	0.705882	1.000000	1
8	brad	0.588235	0.818182	1
9	angelina	0.529412	0.454545	4
10	donaid	0.647059	0.545455	4
11	tom	0.000000	0.000000	0
12	arnoid	0.058824	0.054545	0
13	jared	0.117647	0.109091	0
14	stark	0.176471	0.081818	0
15	ranbir	0.352941	0.145455	0
16	dipika	0.823529	0.363636	2
17	priyanka	0.882353	0.327273	2
18	nick	1.000000	0.345455	2
19	alia	0.764706	0.636364	4
20	sid	0.882353	0.672727	4
21	abdul	0.764706	0.236364	2

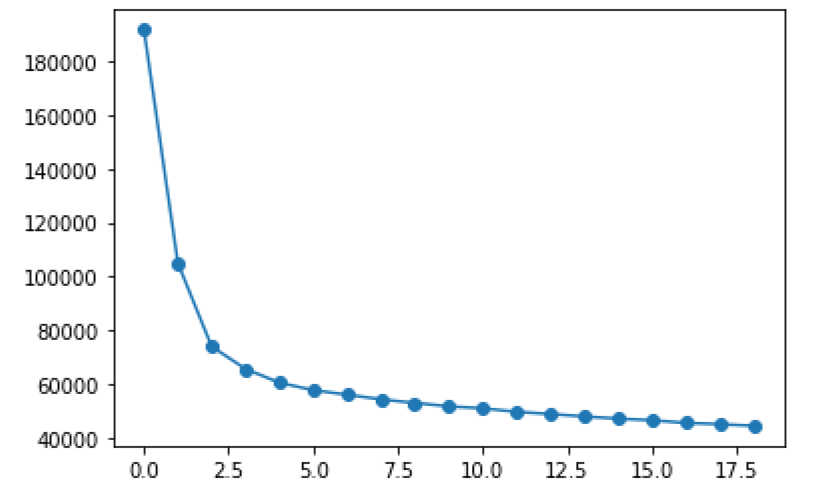


Or,

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.cluster import KMeans

df=pd.read_csv(r'F:\Chand\book,
resume,psim\PSIM\Books\Books\python\Kmeans_E3\E3\kmeans\student_evalua
tion.csv')

#start with cluster value of two

km=KMeans(n_clusters=2)

km.fit(df)

pred=km.predict(df)

#with out scalling

sse=[]

for i in range(1,20):

    km=KMeans(n_clusters=i)
```

```
    km.fit(df)
    sse.append(km.inertia_)
plt.plot(sse,marker='o')
plt.show()
#with scalling
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
data_scaler=scaler.fit_transform(df)
sse_scaled=[]
for i in range(1,20):
    km=KMeans(n_clusters=i)
    km.fit(data_scaler)
    sse_scaled.append(km.inertia_)
plt.plot(sse_scaled,marker='o')
plt.show()
```





Remaining code:

```
#from the elbow plot scaleddataset we can see that best cluster will
be for 4

#now apply kmeans for k=4

km=KMeans(n_clusters=4)

km.fit(data_scaler)

prediction=km.predict(data_scaler)

prediction
```

o/p:

```
array([3, 3, 2, ..., 2, 1, 1])
```

remaining code:

```
# to cheak our data are properly cluster or not

frame=pd.DataFrame(data_scaler)

frame['cluster']=prediction

frame.loc[frame['cluster']==2,:]

#from frame dataset we can see that we made a good clustering
```

<u>Logistic regression example(code basics):</u>

Download employee retaintion dataset

1.Now do some exploratory data analysis to figure out which variable have direct and clear impact on employee relation.

2.Plot bar chart showing impact of employee salary on retaionsion.

3.Plot bar chart showing correlation between department and employee retaintion

4.now build logistic regression model using variable that are narrowed down in step1.

5.,mesure the accuracy of the model

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.linear_model import LogisticRegression

df=pd.read_csv(r'C:\Users\Shubhamay\Documents\HR.csv')

print(df.groupby('left').mean())

#for this data exploration we can conclude that satisfaction level,

#average_monthly_hour,time_spend_company,work_accident

#promotion_last_5yeras and salary depend more in retaintion

pd.crosstab(df['salary'],df['left']).plot(kind='bar')
```

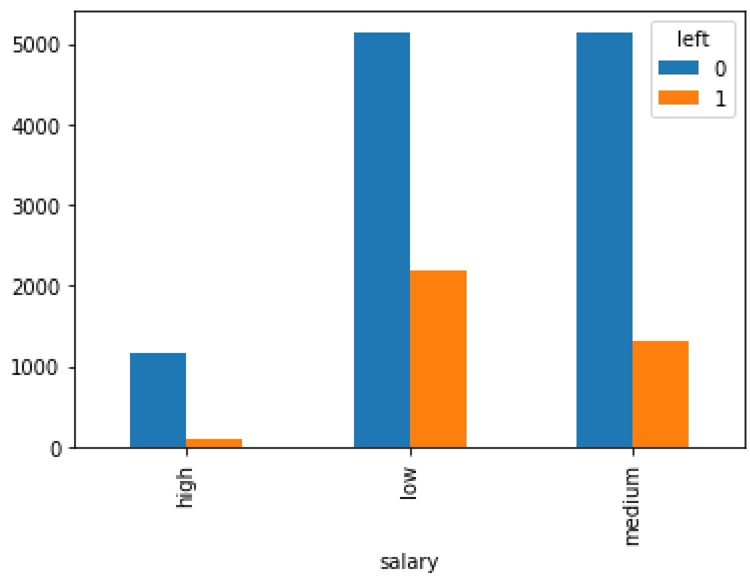
```
plt.show()

pd.crosstab(df['Department'],df['left']).plot(kind='bar')

plt.show()

subdf =
df[['satisfaction_level','average_montly_hours','promotion_last_5years
','salary']]

salary_dummies = pd.get_dummies(subdf['salary'], prefix='salary')

with_salary_dummies=pd.concat([subdf,salary_dummies],axis='columns')

with_salary_dummies.drop('salary',axis='columns',inplace=True)

X=with_salary_dummies

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(X,df['left'],train_size
=0.3)

logreg=LogisticRegression(solver='liblinear',multi_class='ovr')

logreg.fit(x_train,y_train)

logreg.predict(x_test)

file=pd.read_csv(r'C:\Users\Shubhamay\Documents\logpredict.csv')

predictbyme=logreg.predict(file)

file['predict']=predictbyme

print(' test dataset prediction',logreg.score(x_test,y_test))

logreg.score(x_train,y_train)

file.to_csv('logprediction.csv')

#location of this file C:\Users\Shubhamay
```

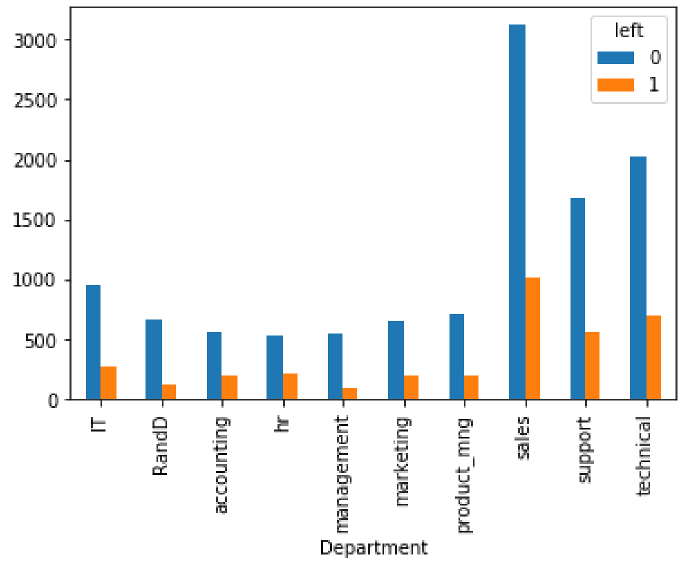
o/p:

```
      satisfaction_level  last_evaluation  number_project  \
left
0               0.666810         0.715473        3.786664
```

```
1                 0.440098         0.718113        3.855503

      average_montly_hours  time_spend_company  Work_accident  \
left
0                 199.060203            3.380032       0.175009
1                 207.419210            3.876505       0.047326

      promotion_last_5years
left
0                    0.026251
1                    0.005321
```

1 0.005321





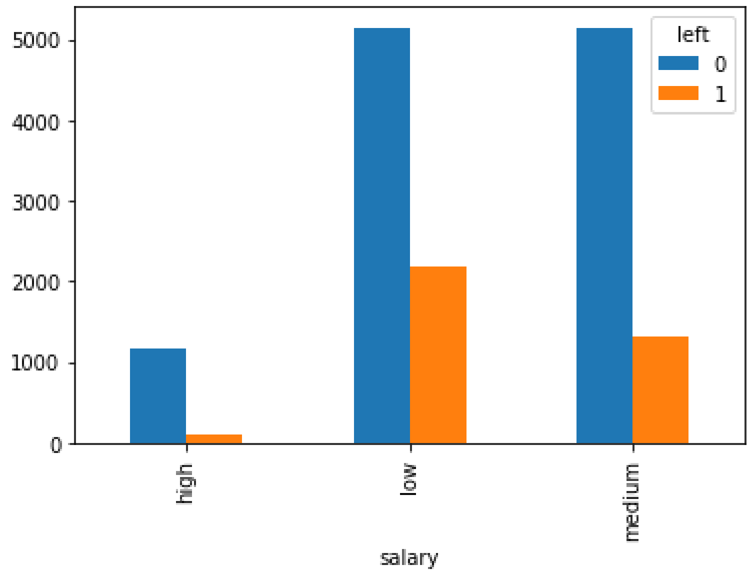

```
test dataset prediction 0.7773333333333333
```

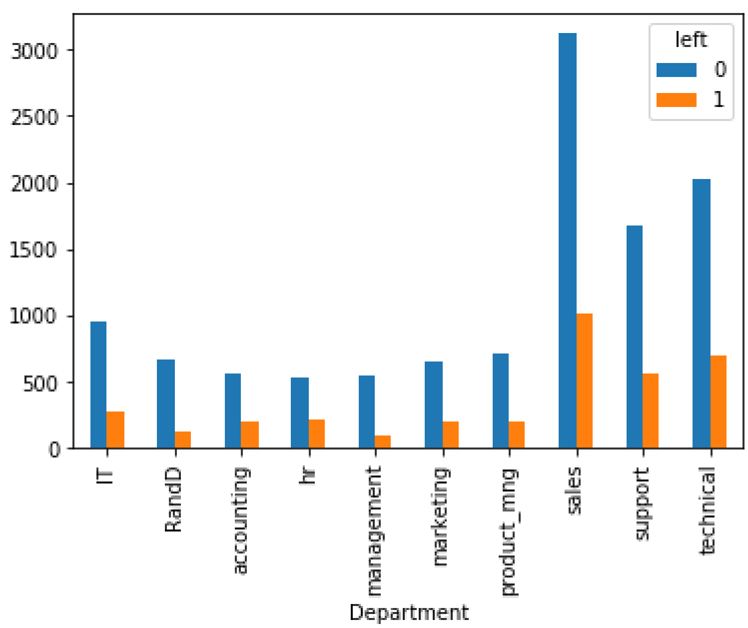
Same prediction by decision tree:

```
import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.linear_model import LogisticRegression

df=pd.read_csv(r'C:\Users\Shubhamay\Documents\HR.csv')

print(df.groupby('left').mean())

#for this data exploration we can conclude that satisfaction level,

#average_monthly_hour,time_spend_company,work_accident

#promotion_last_5yeras and salary depend more in retaintion

pd.crosstab(df['salary'],df['left']).plot(kind='bar')

plt.show()

pd.crosstab(df['Department'],df['left']).plot(kind='bar')

plt.show()

subdf =
df[['satisfaction_level','average_montly_hours','promotion_last_5years
','salary']]
```

```
salary_dummies = pd.get_dummies(subdf['salary'], prefix='salary')

with_salary_dummies=pd.concat([subdf,salary_dummies],axis='columns')

with_salary_dummies.drop('salary',axis='columns',inplace=True)

X=with_salary_dummies

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(X,df['left'],train_size
=0.3)

from sklearn.tree import DecisionTreeClassifier

dr=DecisionTreeClassifier()

dr.fit(x_train,y_train)

print(dr.predict(x_test))

print(dr.score(x_train,y_train))

file=pd.read_csv(r'C:\Users\Shubhamay\Documents\logpredict.csv')

predictbyme=dr.predict(file)

file['predict']=predictbyme

dr.score(x_train,y_train)

file.to_csv('logpredictionbydecisiontree.csv')

#location of this file C:\Users\Shubhamay
```

1 0.003321






```
[0 0 0 ... 0 0 0]
0.9895532340520116
```

Out[8]: `array([0, 0, 0, 0, 0, 1], dtype=int64)`

Decision tree example by codebaseics:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split
```

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split
```

```
df=pd.read_csv(r'F:\Chand\book, resume,psim\PSIM\Books\Books\python\titanic.csv')

subdf=df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Embarked'],axis=
1)

subdf['Age']=subdf['Age'].fillna(subdf['Age'].mean())

subdf=pd.get_dummies(subdf)

inputs=subdf.drop('Survived',axis=1)

target=subdf['Survived']

x_train,x_test,y_train,y_test=train_test_split(inputs,target,train_size=0.8)

len(x_test)

dt=DecisionTreeClassifier()

dt.fit(x_train,y_train)

print(dt.predict(x_test))

print('\n')

print(dt.score(x_test,y_test))
```

or,

```
[0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 1 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0
 0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1
 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 0
 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0
 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0]
0.8268156424581006
```

or:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

df=pd.read_csv(r'F:\Chand\book,
resume,psim\PSIM\Books\Books\python\titanic.csv')
subdf=df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Emba
rked'],axis=1)
subdf['Age']=subdf['Age'].fillna(subdf['Age'].mean())
inputs=subdf.drop('Survived',axis=1)
target=subdf['Survived']
inputs['Sex']=inputs['Sex'].map({'male':1,'female':2})
x_train,x_test,y_train,y_test=train_test_split(inputs,target,train_size=0.
8)
len(x_test)
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
print(dt.predict(x_test))
print('\n')
print(dt.score(x_test,y_test))
```

o/p:

```
[1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0
 0 0 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0
 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0]


0.8156424581005587
```

Or,

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

df=pd.read_csv(r'F:\Chand\book,
resume,psim\PSIM\Books\Books\python\titanic.csv')
subdf=df.drop(['PassengerId','Name','SibSp','Parch','Ticket','Cabin','Emba
rked'],axis=1)
subdf['Age']=subdf['Age'].fillna(subdf['Age'].mean())
inputs=subdf.drop('Survived',axis=1)
target=subdf['Survived']
inputs['Sex']=inputs['Sex'].map({'male':1,'female':2})
x_train,x_test,y_train,y_test=train_test_split(inputs,target,train_size=0.
8)
len(x_test)
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
print(logreg.predict(x_test))
print('\n')
print(logreg.score(x_test,y_test))
```

o/p:

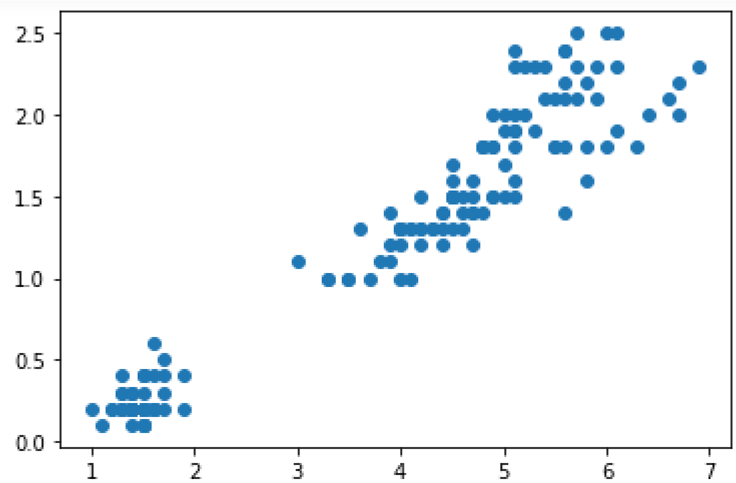
```
[0 1 0 1 1 0 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0 0
 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 0 1 0 1 1 1 1
 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1 1 0
 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0
 1 1 1 0 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1 1 0 0 0]


0.8212290502793296
```

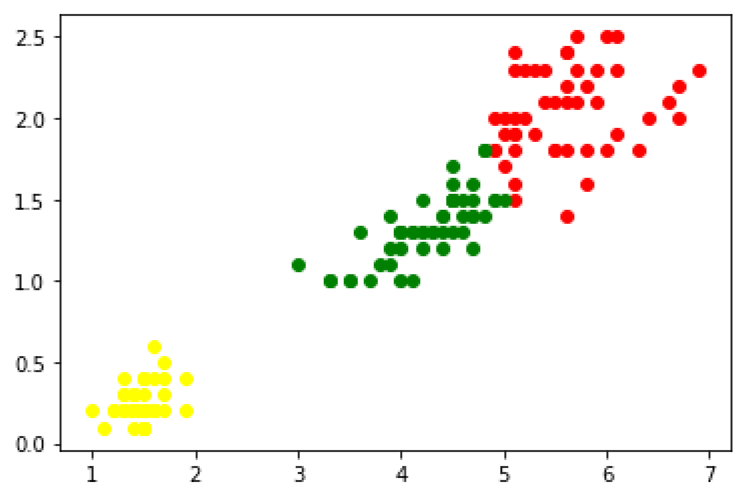
KMeans by codebasics

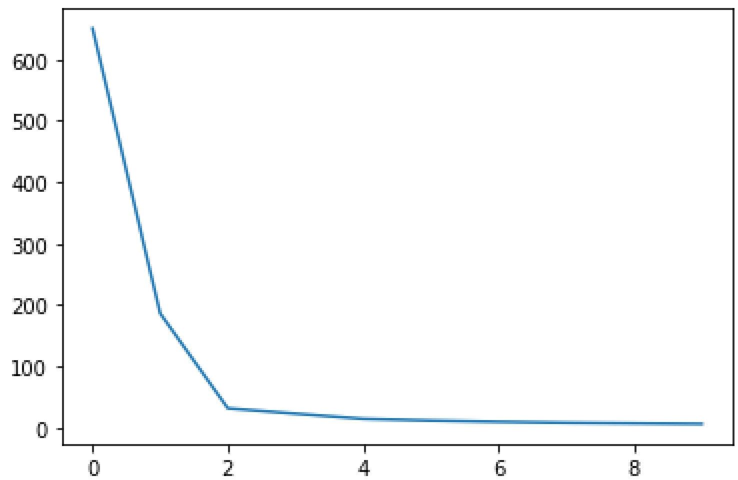
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.cluster import KMeans
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\irish.csv')
subdf=df.drop(['sepal_length','sepal_width','species'],axis=1)
plt.scatter(subdf['petal_length'],subdf['petal_width'])
plt.show()
km=KMeans(n_clusters=3)
yp=km.fit_predict(subdf)
subdf['cluster']=yp
subdf1=subdf[subdf['cluster']==0]
subdf2=subdf[subdf['cluster']==1]
subdf3=subdf[subdf['cluster']==2]
plt.scatter(subdf1['petal_length'],subdf1['petal_width'],color='red')
plt.scatter(subdf2['petal_length'],subdf2['petal_width'],color='yellow')
plt.scatter(subdf3['petal_length'],subdf3['petal_width'],color='green')
plt.show()
#elbow plot
sse=[]
```

```
for k in range(1,11):
    km=KMeans(n_clusters=k)
    km.fit(subdf)
    sse.append(km.inertia_)
plt.plot(sse)
plt.show()
```

o/p:



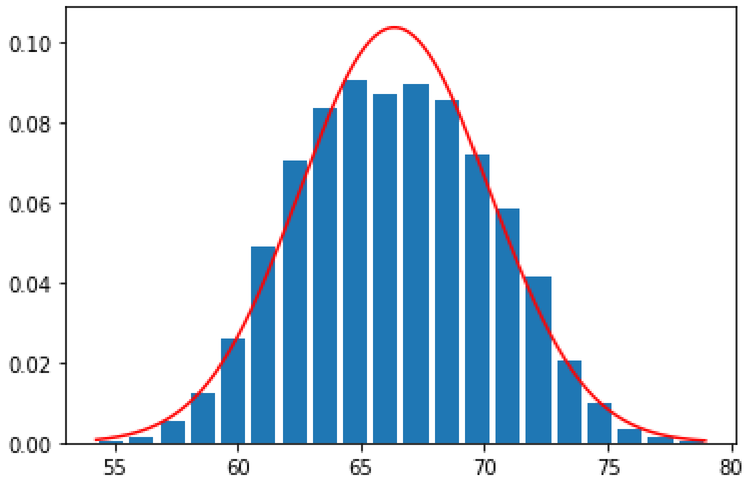




Drawing a belcurve:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

from scipy.stats import norm


df=pd.read_csv(r'C:\Users\Shubhamay\Downloads\heights.csv')

subdf=df.drop('Weight',axis=1)

plt.hist(subdf.Height,bins=20,rwidth=0.8,density=True)

rng=np.arange(subdf.Height.min(),subdf.Height.max(),0.1)

plt.plot(rng,norm.pdf(rng,subdf['Height'].mean(),subdf['Height'].std()),color='red')

plt.show()
```

o/p:



Removing outlier:

```
#removing outlier from the weight-height table

upper_limit=subdf.Height.mean()+3*subdf.Height.std()

lower_limit=subdf.Height.mean()-3*subdf.Height.std()

print(upper_limit)

print('\n')

print(lower_limit)

#dataset with outlier

datasetwith_outlier=subdf[(subdf.Height>upper_limit)|(subdf.Height<lower_limit)]

print(datasetwith_outlier)

print('\n')

#dataset without outlier

dataset_without_outlier=subdf[(subdf.Height<upper_limit)&(subdf.Height>lower_limit)]

print(dataset_without_outlier)
```

o/p:

```
77.91014411714076


54.82497539250136
      Gender     Height   Z-Score
994     Male  78.095867  3.048271
1317    Male  78.462053  3.143445
2014    Male  78.998742  3.282934
3285    Male  78.528210  3.160640
3757    Male  78.621374  3.184854
```

```
6624  Female  54.616858 -3.054091
9285  Female  54.263133 -3.146027


      Gender     Height   Z-Score
0       Male  73.847017  1.943964
1       Male  68.781904  0.627505
2       Male  74.110105  2.012343
3       Male  71.730978  1.393991
4       Male  69.881796  0.913375
...      ...        ...       ...
9995  Female  66.172652 -0.050658
9996  Female  67.067155  0.181830
9997  Female  63.867992 -0.649655
9998  Female  69.034243  0.693090
9999  Female  61.944246 -1.149651

[9993 rows x 3 columns]
```

<u>Removing outlier with z score:</u>

```
#removing outlier with z-score

#create coumn with z-score

subdf['Z-Score']=(subdf.Height-subdf.Height.mean())/subdf.Height.std()

print(subdf)

print('\n')



#filter out of sample which have the sd 3 or above and -3 or below

#data of outlier

o=subdf[(subdf['Z-Score']>3)|(subdf['Z-Score']<-3)]

print(o)

print('\n')

#data without outlier

wo=subdf[(subdf['Z-Score']<3) & (subdf['Z-Score']>-3)]

print(wo)
```

o/p:

```
      Gender     Height   Z-Score
0       Male  73.847017  1.943964
1       Male  68.781904  0.627505
```

```
2       Male  74.110105  2.012343
3       Male  71.730978  1.393991
4       Male  69.881796  0.913375
...      ...        ...       ...
9995  Female  66.172652 -0.050658
9996  Female  67.067155  0.181830
9997  Female  63.867992 -0.649655
9998  Female  69.034243  0.693090
9999  Female  61.944246 -1.149651

[10000 rows x 3 columns]


      Gender     Height   Z-Score
994     Male  78.095867  3.048271
1317    Male  78.462053  3.143445
2014    Male  78.998742  3.282934
3285    Male  78.528210  3.160640
3757    Male  78.621374  3.184854
6624  Female  54.616858 -3.054091
9285  Female  54.263133 -3.146027


      Gender     Height   Z-Score
0       Male  73.847017  1.943964
1       Male  68.781904  0.627505
2       Male  74.110105  2.012343
3       Male  71.730978  1.393991
4       Male  69.881796  0.913375
...      ...        ...       ...
9995  Female  66.172652 -0.050658
9996  Female  67.067155  0.181830
9997  Female  63.867992 -0.649655
9998  Female  69.034243  0.693090
9999  Female  61.944246 -1.149651

[9993 rows x 3 columns]
```

FIND OUT Z-SCORE OF DIFFERENCE CONIDENCE LEVEL

```
#find z-score of different confidence level
from scipy.stats import norm
import numpy as np
#95%confidence level
#1-0.95=0.05,0.05/2=0.025,0.95+0.025=0.975
z95=norm.ppf(0.975)
print(z95)
print('\n')
#99% confidence level
z99=norm.ppf(0.995)
print(z99)
print('\n')
#90% confidence level
z90=norm.ppf(0.95)
```

```
print(z90)
```

o/p:

1.959963984540054

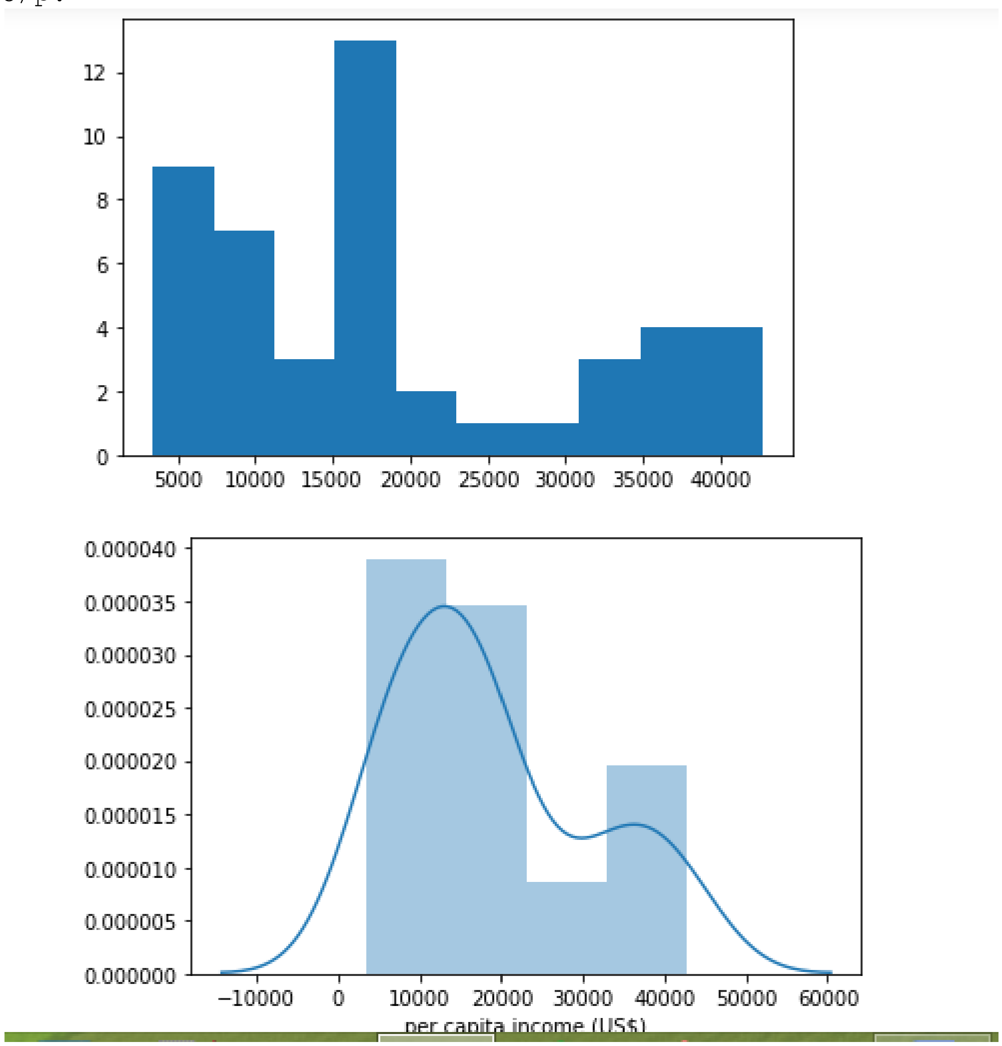
2.5758293035489004

1.6448536269514722

Find population mean of 95% interval

```
#find population mean which lies between 95% of interval
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv(r'C:\Users\Shubhamay\Documents\canada_per_capita_income.csv
')
plt.hist(df['per capita income (US$)'])
plt.show()
sns.distplot(df['per capita income (US$)'])
plt.show()
smalldf=df.drop('year',axis=1)
sigma=smalldf['per capita income (US$)'].std()
x_bar=smalldf['per capita income (US$)'].mean()
n=len(smalldf)
con_coef=.95
alpha=1-con_coef


from scipy.stats import norm
#z_score value for 95% CL
z_score=norm.ppf(.975)
#find z_interval
z_interval=norm.interval(.95)
standerd_error=sigma/np.math.sqrt(n)
CI_lower=x_bar-(z_score*standerd_error)
CI_upper=x_bar+(z_score*standerd_error)
Print(CI_lower,CI_upper)
#of population distribution

#taking 20 sample
sample=np.random.choice(df.index,20)
df_sample=smalldf.loc[sample]
x_bar_sample=df_sample['per capita income (US$)'].mean()
sigma_sample=df_sample['per capita income (US$)'].std()
z_score_sample=norm.ppf(0.975)
SE_sample=sigma_sample/np.sqrt(20)
CI_sample_lower_limit=x_bar_sample-(z_score_sample*SE_sample)
CI_sample_upper_limit=x_bar_sample+(z_score_sample*SE_sample)
Print(CI_sample_lower_limit,CI_sample_upper_limit)
```

o/p:



(15479.540995195799, 22360.73313161271)

(13954.937273904005, 23354.944571595996)# it is ci upper and lower limit of sample so it change continuously

One sample t-test

```
#ONE SAMPLE T-TEST
#WHEATHER MEAN OF THE SAMPLE AND POPULATION ARE DIFFERENT
#NULL HYPOTHESIS- THERE IS NO DIFFERENCE
#ALTERNATE HYPOTHESIS -THERE IS DIFFERENCE
import numpy as np
from scipy.stats import ttest_1samp
import pandas as pd
ages=[10,20,35,50,28,40,55,18,16,55,30,25,43,18,\
      30,28,14,24,16,17,32,25,26,27,65,18,43,23,21,20,19,70]
```

```
ages_mean=np.mean(ages)#mean  of population mean
#take sample of size 10
sample_size=10
ages_sample=np.random.choice(ages,sample_size)
tstatistic,p_value=ttest_1samp(ages_sample,ages_mean)
p_value
#as it greater than 0.05 so we can accept null hypothes

#so there is actully  no difference no difference between sample mean and population mean


o/p:
.11859170122950405
```

Or,

```
#IS CLASS A AGE DIFFERENT FROM SCHOOL AGE
#NUUL HYPOTHESIS: THERE IS NO DIFFERNCE
#ALTERNETE HYPOTHESIS :THERE IS  DIFFERENCE
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import math
from scipy.stats import ttest_1samp
school_ages=stats.poisson.rvs(loc=18,mu=35,size=1500)
classA_ages=stats.poisson.rvs(loc=18,mu=30,size=60)
t_statistic,p_value=ttest_1samp(classA_ages,school_ages.mean())
p_value
#P_VALUE ISMUCHMORE LESS THAN O.05 SO WE CAN REJECT NULL HYPOTHESIS

o/p:

6.893119211384918e-11
```

TWO SAMPLE T_TEST:

```
# WE TAKE TWO SAMPLE CLASS a AND B
#FIND IF THE MEAN OF TWO SAMPLE IS DIFFERENT FROM SCHOOL AGE MEAN
#NUUL HYPOTHESIS: THERE IS NO DIFFERNCE
#ALTERNETE HYPOTHESIS :THERE IS  DIFFERENCE
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import math
from scipy.stats import ttest_ind
school_ages=stats.poisson.rvs(loc=18,mu=35,size=1500)
classA_ages=stats.poisson.rvs(loc=18,mu=30,size=60)
classB_ages=stats.poisson.rvs(loc=18,mu=33,size=60)
t_statistic,p_value=ttest_ind(classA_ages,classB_ages,equal_var=False)
p_value
#P_VALUE ISMUCHMORE LESS THAN O.05 SO WE CAN REJECT NULL HYPOTHESIS
```

o/p:
0.006692318822061351

PAIRED T _TEST

```
#weight some time befor
weight1=[20,24,25,26,28,24,31,35,26,24,26,28,37,36,34,35]
#weight some time after
weight2=weight1+stats.norm.rvs(scale=5,loc=-1.25,size=16)
#two weight are from same kids
#we want to cheak if ther is any statisticle difference or not
#null hypothesis:there is no difference
#alternate hypo thesis there is difference
from scipy.stats import ttest_rel
_,p_value=ttest_rel(weight1,weight2)
p_value
#P_VALUE IS GREATER THAN 0.05 SO WE ACCEPT NULL HYPOTHESIS
```

o/p:

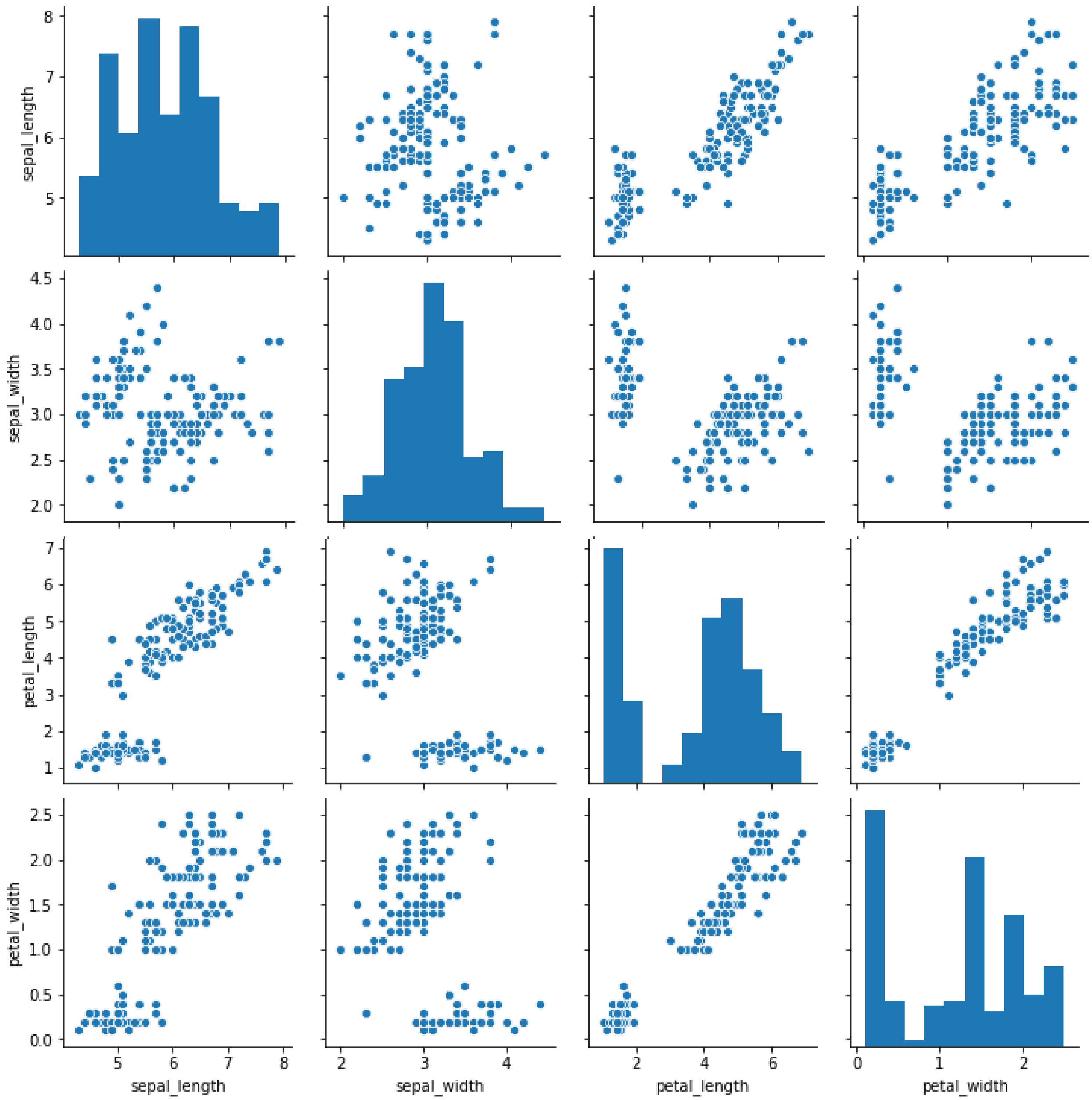
0.38536838714511656

Co relation between data set:

```
import seaborn as sns
df=sns.load_dataset('iris')
print(df)
print(df.corr())
sns.pairplot(df)
plt.show()
```

o/p:

```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
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]
              sepal_length  sepal_width  petal_length  petal_width
sepal_length      1.000000    -0.117570      0.871754     0.817941
sepal_width      -0.117570     1.000000     -0.428440    -0.366126
petal_length      0.871754    -0.428440      1.000000     0.962865
petal_width       0.817941    -0.366126      0.962865     1.000000
```



Import image in python

```
from IPython.display import Image

import matplotlib.pyplot as plt


Image(filename='F:\Chand\sog and pic\ooty pic\DSC_0020.JPG')
```

o/p:

Calculation of qualtile:

```
import numpy as np

a=np.random.randint(7,10,20)

Q1=np.percentile(a,25)

Q2=np.percentile(a,50)

Q3=np.percentile(a,75)

print('first quartile:',Q1)

print('2nd quartile:',Q2)

print('third quartile:',Q3)

print('inter quartile range:',(Q3-Q1))
```

o/p:

```
first quartile: 7.0
2nd quartile: 8.0
third quartile: 9.0
inter quartile range: 2.0
```

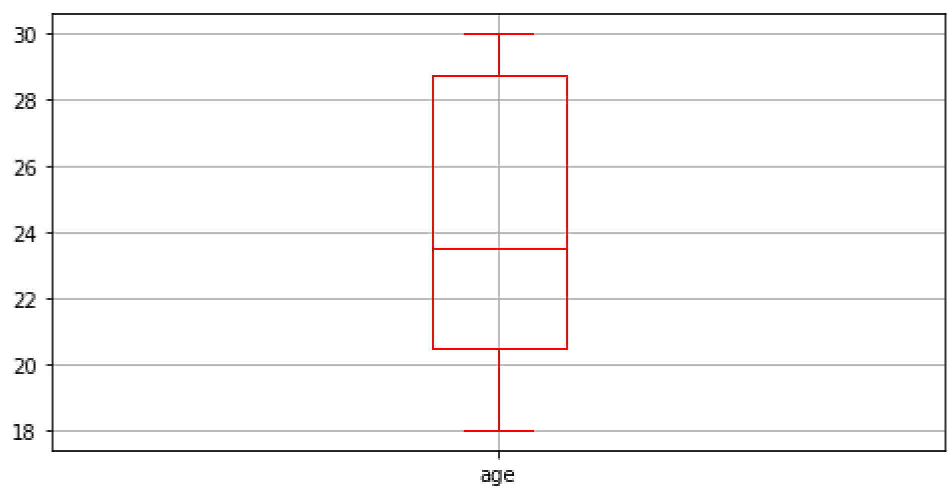
creating a dataframe and ploting boxplot:

```
import numpy as np

import pandas as pd
```

```
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize']=(8,4)

df=pd.DataFrame(dict(id=range(6),age=np.random.randint(18,31,6)))

print(df)

df.boxplot(column='age',return_type='axes',color='r')

plt.show()
```

o/p

```
   id  age
0   0   30
1   1   30
2   2   20
3   3   22
4   4   25
5   5   18
```



Kurtosis and skewedness of the distribution:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

plt.rcParams['figure.figsize']=(8,4)

df=pd.DataFrame(dict(age=[12,15,18,21,19,16,25,14]))

print(df)

p=df.age.skew()

print('skewedness of distribution:',p)
```

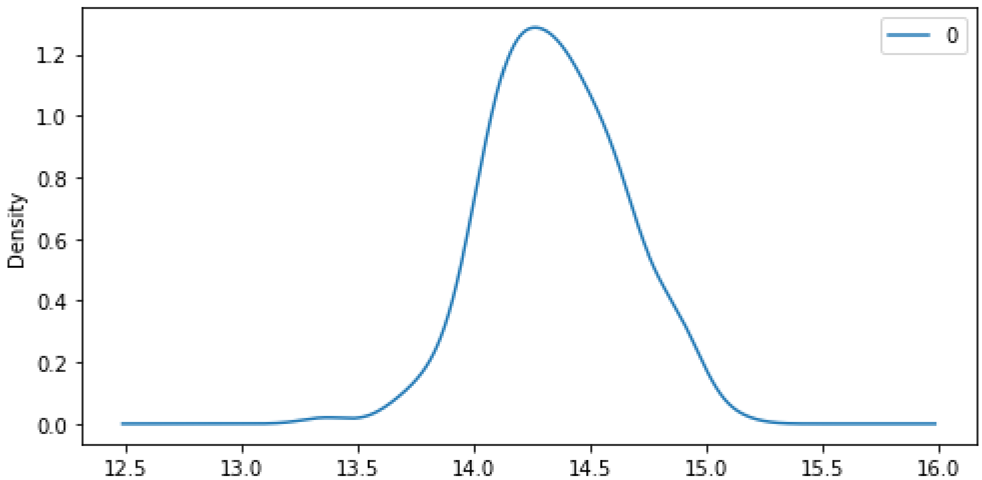
```
d=df.age.kurt()

print('kurtosis of distribution:',d)
```

o/p:

```
   age
0   12
1   15
2   18
3   21
4   19
5   16
6   25
7   14
skewedness of distribution: 0.6282887050234865
kurtosis of distribution: 0.09857565170652993
```

central limit theorem:

```
#population dataset

population=np.random.randint(10,20,1000)

estimate=[]

for i in range(200):

    sample=np.random.choice(population,100)

    estimate.append(sample.mean())

#here we first create a population

#then take 200 sample from there each having 100 values

# and plot means of each didtribution

#it approch to the normal distribution

# it is defination of central limit therem

pd.DataFrame(estimate).plot(kind='density')
```



0
Density
1.2
1.0
0.8
0.6
0.4
0.2
0.0
12.5
13.0
13.5
14.0
14.5
15.0
15.5
16.0