INDEX

34. write a program to demonstrate K-means clustering algorithm.

35. write a program to demonstrate k-medoid clustering algorithm.

PROGRAMS

1.Write a program to Create an array with numeric values and find **mean** of them.

```
import numpy as np
arr=eval(input('enter values as [1,2,3] : '))
arr=np.array(arr)
sum=0
for i in arr:
    sum+=i
print('mean of given values {} is : '.format(arr),sum/len(arr))
```

OUTPUT:

```
enter values as [1,2,3] : [2,3,6,2,8,3,7]
mean of given values [2 3 6 2 8 3 7] is : 4.428571428571429
```

2.Write a program to Create an array with numeric values and find **median** of them.

```
import numpy as np
arr=eval(input('enter values as [1,2,3] : '))
arr=np.array(arr)


def median(arr):
    arr=sorted(arr)
    if len(arr)%2==1:
        l=len(arr)//2
        return arr[l]
    else:
        l=len(arr)//2
        sum=arr[l]+arr[l-1]
```

```
        return sum/2


print('median of given value {} is : '.format(sorted(arr)),median(arr))
```

OUTPUT:

```
enter values as [1,2,3] : [1,5,2,8,4,8,7]
median of given value [1, 2, 4, 5, 7, 8, 8] is :  5
```

3.Write a program to Create an array with numeric values and find **mode** of them.

```
import numpy as np
arr=eval(input('enter values as [1,2,3] : '))
arr=np.array(arr)


def median(arr):
    arr=sorted(arr)
    if len(arr)%2==1:
        l=len(arr)//2
        return arr[l]
    else:
        l=len(arr)//2
        sum=arr[l]+arr[l-1]
        return sum/2


def mean(arr):
    sum=0
    for i in arr:
        sum+=i
    return sum/len(arr)


def mode(arr):
    mode=3*median(arr)-2*mean(arr)
    return mode


print('mode of given val {} is : '.format(arr),mode(arr))
```

OUTPUT:

```
enter values as [1,2,3] : [2,5,3,7,4,8,9]
```

mode of given val [2 5 3 7 4 8 9] is : 4.142857142857142

4.Write a program to read an excel sheet and find **mean,median** and **mode** of Price.

```
import pandas as pd
df=pd.read_excel("income.xlsx",sheet_name='Sheet1')
print('Mean of the values of price are : ',df['Price'].mean())
print('Median of the values of price are : ',df['Price'].median())
print('Mode of the values of price are : ',df['Price'].mode())
```

OUTPUT:

Mean of the values of price are : 16181.758620689656

Median of the values of price are : 16124.5

Mode of the values of price are : 0   15950

Name: Price, dtype: int64

5.write a program to create an array with some np.nan values and replace np.nan values with mean.

```
import numpy as np
v = np.array([1,2, 3, 4,np.nan,5,6,np.nan,np.nan])
print('Before replacing : ',v)
vMean=np.nanmean(v)
for i in range(len(v)):
   if np.isnan(v[i]):
      v[i]=vMean
#v[np.isnan(v)] = np.nanmean(v)
print('After replacing with mean : ',v)
```

OUTPUT:

Before replacing : [1. 2. 3. 4. nan 5. 6. nan nan]

After replacing with mean : [1. 2. 3. 4. 3.5 5. 6. 3.5 3.5]

6. write a program to create an array with some np.nan values and replace np.nan values with median.

```
import numpy as np
v = np.array([1,6, 3, 4,np.nan,5,6,np.nan,np.nan])
print('Before replacing : ',v)
vMean=np.nanmedian(v)
for i in range(len(v)):
   if np.isnan(v[i]):
      v[i]=vMean
#v[np.isnan(v)] = np.nanmean(v)
```

```
print('After replacing with mean : ',v)
```

OUTPUT:

```
Before replacing :  [ 1.  6.  3.  4. nan  5.  6. nan nan]
After replacing with mean :  [1.  6.  3.  4.  4.5 5.  6.  4.5 4.5]
```

7. write a program to create a string array with some np.nan values and replace np.nan values with unknown.

```
import numpy as np
v = np.array(['ab','bc','cd','de',np.nan,'ef','fg',np.nan,np.nan])
print('Before replacing : ',v)
for i in range(len(v)):
    if v[i]==str(np.nan):
        v[i]='unknown'
#v[np.where(v.astype(str)==str(np.nan))]='unknown'
print('After replacing with "unknown" : ',v)
```

OUTPUT:

```
Before replacing :  ['ab' 'bc' 'cd' 'de' 'nan' 'ef' 'fg' 'nan' 'nan']
After replacing with "unknown" :  ['ab' 'bc' 'cd' 'de' 'unknown' 'ef' 'fg' 'unknown' 'unknown']
```

8.write a program to read an excel sheet and replace nan values with mean in age.

```
import pandas as pd
df=pd.read_excel("income.xlsx",sheet_name='Sheet1',\
                 index_col=0,na_values=["????","??"])

print("Age values before replacing with mean : ")
print(df.head(10))
df['Age'].fillna(df['Age'].mean(),inplace=True)
print("Age values after replacing with mean : ")
print(df.head(10))
```

OUTPUT:

```
Age values before replacing with mean :
   Price   Age      KM FuelType  ...  Automatic    CC  Doors  Weight
0  13500   NaN  46986.0   Diesel  ...          0  2000      3    1165
1  13750  23.0  72937.0      NaN  ...          0  2000      3    1165
2  13950   NaN      NaN      NaN  ...          0  2000      3    1165
```

3 14950 26.0 NaN NaN ... 0 2000 3 1165

4 13750 NaN NaN Diesel ... 0 2000 3 1170

5 12950 32.0 61000.0 NaN ... 0 2000 3 1170

6 16900 27.0 NaN Diesel ... 0 2000 3 1245

7 18600 NaN 75889.0 NaN ... 0 2000 3 1245

8 21500 27.0 19700.0 Petrol ... 0 1800 3 1185

9 12950 23.0 71138.0 Diesel ... 0 1900 3 1105

[10 rows x 10 columns]

Age values after replacing with mean :

Price Age KM FuelType ... Automatic CC Doors Weight

0 13500 29.598131 46986.0 Diesel ... 0 2000 3 1165

1 13750 23.000000 72937.0 NaN ... 0 2000 3 1165

2 13950 29.598131 NaN NaN ... 0 2000 3 1165

3 14950 26.000000 NaN NaN ... 0 2000 3 1165

4 13750 29.598131 NaN Diesel ... 0 2000 3 1170

5 12950 32.000000 61000.0 NaN ... 0 2000 3 1170

6 16900 27.000000 NaN Diesel ... 0 2000 3 1245

7 18600 29.598131 75889.0 NaN ... 0 2000 3 1245

8 21500 27.000000 19700.0 Petrol ... 0 1800 3 1185

9 12950 23.000000 71138.0 Diesel ... 0 1900 3 1105

9.write a program to read an excel sheet an d replace nan values with median in KM.

```
import pandas as pd
df=pd.read_excel("income.xlsx",sheet_name='Sheet1',\
        index_col=0,na_values=["????","??"])

print("km values before replacing with median : ")
print(df.head(10))
df['KM'].fillna(df['KM'].median(),inplace=True)
print("km values after replacing with median : ")
print(df.head(10))
```

OUTPUT:

km values before replacing with median :

Price Age KM FuelType ... Automatic CC Doors Weight

```
0  13500   NaN  46986.0   Diesel  ...        0  2000     3    1165
1  13750  23.0  72937.0      NaN  ...        0  2000     3    1165
2  13950   NaN      NaN      NaN  ...        0  2000     3    1165
3  14950  26.0      NaN      NaN  ...        0  2000     3    1165
4  13750   NaN      NaN   Diesel  ...        0  2000     3    1170
5  12950  32.0  61000.0      NaN  ...        0  2000     3    1170
6  16900  27.0      NaN   Diesel  ...        0  2000     3    1245
7  18600   NaN  75889.0      NaN  ...        0  2000     3    1245
8  21500  27.0  19700.0   Petrol  ...        0  1800     3    1185
9  12950  23.0  71138.0   Diesel  ...        0  1900     3    1105

[10 rows x 10 columns]
km values after replacing with median :
   Price   Age      KM FuelType  ...  Automatic    CC  Doors  Weight
0  13500   NaN  46986.0   Diesel  ...        0  2000     3    1165
1  13750  23.0  72937.0      NaN  ...        0  2000     3    1165
2  13950   NaN  36860.5      NaN  ...        0  2000     3    1165
3  14950  26.0  36860.5      NaN  ...        0  2000     3    1165
4  13750   NaN  36860.5   Diesel  ...        0  2000     3    1170
5  12950  32.0  61000.0      NaN  ...        0  2000     3    1170
6  16900  27.0  36860.5   Diesel  ...        0  2000     3    1245
7  18600   NaN  75889.0      NaN  ...        0  2000     3    1245
8  21500  27.0  19700.0   Petrol  ...        0  1800     3    1185
9  12950  23.0  71138.0   Diesel  ...        0  1900     3    1105
```

10.write a program to create an excel sheet and replace nan values with most frequent values in FuelType.

```
import pandas as pd
df=pd.read_excel("income.xlsx",sheet_name='Sheet1',\
            index_col=0,na_values=["????","??"])
print("FuelType values before replacing with specified value : ")
print(df.head(10))
df['FuelType'].fillna(df['FuelType'].value_counts()\
                .index[0],inplace=True)
print("FuelType values after replacing with specified value : ")
print(df.head(10))
```

OUTPUT:

FuelType values before replacing with specified value :

```
   Price   Age      KM FuelType  ...  Automatic    CC  Doors  Weight
0  13500   NaN  46986.0   Diesel  ...          0  2000      3    1165
1  13750  23.0  72937.0      NaN  ...          0  2000      3    1165
2  13950   NaN      NaN      NaN  ...          0  2000      3    1165
3  14950  26.0      NaN      NaN  ...          0  2000      3    1165
4  13750   NaN      NaN   Diesel  ...          0  2000      3    1170
5  12950  32.0  61000.0      NaN  ...          0  2000      3    1170
6  16900  27.0      NaN   Diesel  ...          0  2000      3    1245
7  18600   NaN  75889.0      NaN  ...          0  2000      3    1245
8  21500  27.0  19700.0   Petrol  ...          0  1800      3    1185
9  12950  23.0  71138.0   Diesel  ...          0  1900      3    1105

[10 rows x 10 columns]
```

FuelType values after replacing with specified value :

```
   Price   Age      KM FuelType  ...  Automatic    CC  Doors  Weight
0  13500   NaN  46986.0   Diesel  ...          0  2000      3    1165
1  13750  23.0  72937.0   Petrol  ...          0  2000      3    1165
2  13950   NaN      NaN   Petrol  ...          0  2000      3    1165
3  14950  26.0      NaN   Petrol  ...          0  2000      3    1165
4  13750   NaN      NaN   Diesel  ...          0  2000      3    1170
5  12950  32.0  61000.0   Petrol  ...          0  2000      3    1170
6  16900  27.0      NaN   Diesel  ...          0  2000      3    1245
7  18600   NaN  75889.0   Petrol  ...          0  2000      3    1245
8  21500  27.0  19700.0   Petrol  ...          0  1800      3    1185
9  12950  23.0  71138.0   Diesel  ...          0  1900      3    1105
```

11.write a program to read an excel sheet an replace nan values with mode in metcolor.

```
import pandas as pd
df=pd.read_excel("income.xlsx",sheet_name='Sheet1',\
            index_col=0,na_values=["????","??"])
pd.set_option("display.max_columns",None)
print("MetColor values before replacing with mode : ")
print(df.head(10))
```

```
df['MetColor'].fillna(df['MetColor'].mode()[0],inplace=True)
print("MetColor values after replacing with mode : ")
print(df.head(10))
```

OUTPUT:

```
MetColor values before replacing with mode :
   Price   Age      KM FuelType     HP  MetColor  Automatic    CC  Doors  Weight
0  13500   NaN  46986.0   Diesel   90.0       1.0          0  2000      3    1165
1  13750  23.0  72937.0      NaN   90.0       NaN          0  2000      3    1165
2  13950   NaN      NaN      NaN   90.0       NaN          0  2000      3    1165
3  14950  26.0      NaN      NaN   90.0       0.0          0  2000      3    1165
4  13750   NaN      NaN   Diesel   90.0       NaN          0  2000      3    1170
5  12950  32.0  61000.0      NaN   90.0       0.0          0  2000      3    1170
6  16900  27.0      NaN   Diesel    NaN       NaN          0  2000      3    1245
7  18600   NaN  75889.0      NaN   90.0       1.0          0  2000      3    1245
8  21500  27.0  19700.0   Petrol  192.0       0.0          0  1800      3    1185
9  12950  23.0  71138.0   Diesel    NaN       NaN          0  1900      3    1105

 MetColor values after replacing with mode :
   Price   Age      KM FuelType     HP  MetColor  Automatic    CC  Doors  Weight
0  13500   NaN  46986.0   Diesel   90.0       1.0          0  2000      3    1165
1  13750  23.0  72937.0      NaN   90.0       1.0          0  2000      3    1165
2  13950   NaN      NaN      NaN   90.0       1.0          0  2000      3    1165
3  14950  26.0      NaN      NaN   90.0       0.0          0  2000      3    1165
4  13750   NaN      NaN   Diesel   90.0       1.0          0  2000      3    1170
5  12950  32.0  61000.0      NaN   90.0       0.0          0  2000      3    1170
6  16900  27.0      NaN   Diesel    NaN       1.0          0  2000      3    1245
7  18600   NaN  75889.0      NaN   90.0       1.0          0  2000      3    1245
8  21500  27.0  19700.0   Petrol  192.0       0.0          0  1800      3    1185
9  12950  23.0  71138.0   Diesel    NaN       1.0          0  1900      3    1105
```

12.write a program to create an array and perform smoothing by bin mean on it.

```
import numpy as np
d=[12,2,4,5,7,3,6,8,1,4,7,9,1,6,2,3]
d=np.array(d)
```

```
def sort(d):
    for i in range(len(d)):
        for j in range(len(d)-i-1):
            if d[j]>d[j+1]:
                d[j],d[j+1]=d[j+1],d[j]


def divBin(d):
    global new,r,c
    for i in range(2,len(d)+1):
        if len(d)%i==0:
            new=d.reshape(i,len(d)//i)
            r=i
            c=len(d)//i
            return


def binByMean(new,r,c):
    print('values are replaced with "MEAN" ')
    for i in range(r):
        sum=0
        mean=0
        for j in range(c):
            sum+=new[i,j]
            mean=sum/c
        for j in range(c):
            new[i,j]=round(mean)
sort(d)
print('values after sort : \n',d)
divBin(d)
print('Values after bin : \n',new)
binByMean(new,r,c)
print('after bin by mean : ')
print(new)
```

OUTPUT:

values after sort :

```
[ 1  1  2  2  3  3  4  4  5  6  6  7  7  8  9 12]
Values after bin :
[[ 1  1  2  2  3  3  4  4]
 [ 5  6  6  7  7  8  9 12]]
values are replaced with "MEAN"
[[2 2 2 2 2 2 2 2]
 [8 8 8 8 8 8 8 8]]
```

13.write a program to create an array and perform smoothing by bin median on it.

```
import numpy as np
d=[12,2,4,5,7,3,6,8,1,4,7,9,1,6,2,3]
d=np.array(d)
def sort(d):
    for i in range(len(d)):
        for j in range(len(d)-i-1):
            if d[j]>d[j+1]:
                d[j],d[j+1]=d[j+1],d[j]

def divBin(d):
    global new,r,c
    for i in range(2,len(d)+1):
        if len(d)%i==0:
            new=d.reshape(i,len(d)//i)
            r=i
            c=len(d)//i
            return

def binByMedian(new,r,c):
    print('values are replaced with "MEDIAN" ')
    for i in range(r):
        median=0
        if c%2==1:
            median=new[i,c//2]
        else:
            median=(new[i,c//2-1]+new[i,c//2])/2
```

```
        for j in range(c):
            new[i,j]=round(median)


sort(d)
print('values after sort : \n',d)
divBin(d)
print('Values after bin : \n',new)
binByMedian(new,r,c)
print(new)
```

OUTPUT:

```
values after sort :
 [ 1  1  2  2  3  3  4  4  5  6  6  7  7  8  9 12]
Values after bin :
 [[ 1  1  2  2  3  3  4  4]
 [ 5  6  6  7  7  8  9 12]]
values are replaced with "MEDIAN"
[[2 2 2 2 2 2 2 2]
 [7 7 7 7 7 7 7 7]]
```

14write a program to create an array and perform smoothing by bin boundaries on it.

```
import numpy as np
d=[12,2,4,5,7,3,6,8,1,4,7,9,1,6,2,3]
d=np.array(d)
def sort(d):
    for i in range(len(d)):
        for j in range(len(d)-i-1):
            if d[j]>d[j+1]:
                d[j],d[j+1]=d[j+1],d[j]


def divBin(d):
    global new,r,c
    for i in range(2,len(d)+1):
        if len(d)%i==0:
            new=d.reshape(i,len(d)//i)
            r=i
```

```
        c=len(d)//i
        return
def binByBoundary(new,r,c):
   for i in range(r):
      min1=new[i,0]
      max1=new[i,c-1]
      for j in range(1,c):
         if new[i,j]-min1<max1-new[i,j]:
            new[i,j]=min1
         else:
            new[i,j]=max1
sort(d)
print('values after sort : \n',d)
divBin(d)
print('Values after bin : \n',new)
binByBoundary(new,r,c)
print('values are replaced with boundaries :  ')
print(new)
```

OUTPUT:

```
values after sort :
 [ 1  1  2  2  3  3  4  4  5  6  6  7  7  8  9 12]
Values after bin :
 [[ 1  1  2  2  3  3  4  4]
 [ 5  6  6  7  7  8  9 12]]
values are replaced with boundaries :
[[ 1  1  1  1  4  4  4  4]
 [ 5  5  5  5  5  5 12 12]]
```

15.write a program to read an excel sheet and perform mean_max normalization on price.

```
import pandas as pd

df=pd.read_excel("income.xlsx",sheet_name="Sheet1",index_col=0)
print("Data frame before applying min_max normalization(0 to 1 or -1 to 1) to price attribute\n\n")

def min_max_normal(a):
```

```
    maxv=max(a)
    minv=min(a)
    difv=maxv-minv
    minnv=int(input("enter new min vals i.e 0 or -1 : "))
    maxnv=1 #new max is always 1
    d=((a-minv*(maxnv-minnv))/(difv))+minnv
    return d


print(df.head())
df["Price"]=min_max_normal(df["Price"])
print("\n\nData frame after applying min_max normalization(0 to 1 or -1 to 1) to price attribute\n\n")
print(df.head())
```

OUTPUT:

```
Data frame before applying min_max normalization(0 to 1 or -1 to 1) to price attribute


  Price   Age    KM FuelType  HP MetColor  Automatic    CC  Doors  Weight
0  13500   NaN  46986   Diesel  90      1        0  2000     3    1165
1  13750    23  72937      ??  90      ??       0  2000     3    1165
2  13950  ????    NaN    ????  90     NaN       0  2000     3    1165
3  14950    26    ??     NaN  90      0        0  2000     3    1165
4  13750    ??  ????   Diesel  90    ????       0  2000     3    1170
enter new min vals i.e 0 or -1 : 0


Data frame after applying min_max normalization(0 to 1 or -1 to 1) to price attribute


    Price   Age    KM FuelType  HP MetColor  Automatic    CC  Doors  Weight
0  0.471429   NaN  46986   Diesel  90      1        0  2000     3    1165
1  0.485714    23  72937      ??  90      ??       0  2000     3    1165
2  0.497143  ????    NaN    ????  90     NaN       0  2000     3    1165
3  0.554286    26    ??     NaN  90      0        0  2000     3    1165
```

4 0.485714 ?? ???? Diesel 90 ???? 0 2000 3 1170

16.write a program to read an excel sheet and perform z-score normalization on price.

```
import pandas as pd
import statistics as sts

df=pd.read_excel("income.xlsx",sheet_name="Sheet1",index_col=0)
print('Data frame before applying ZScore normalization to "price" attribute\n\n')

def ZScore_normal(a):
    m=sts.mean(a)
    strd=sts.stdev(a)
    d=(a-m)/strd
    return d

print(df.head())
df["Price"]=ZScore_normal(df["Price"])
print('\n\nData frame after applying ZScore normalization to "price" attribute\n\n')
print(df.head())
```

OUTPUT:

```
Data frame before applying ZScore normalization to "price" attribute


  Price   Age    KM FuelType  HP MetColor  Automatic    CC  Doors  Weight
0  13500   NaN  46986   Diesel  90       1          0  2000      3    1165
1  13750    23  72937       ??  90       ??         0  2000      3    1165
2  13950  ????    NaN    ????  90      NaN         0  2000      3    1165
3  14950    26     ??     NaN  90        0         0  2000      3    1165
4  13750    ??   ????   Diesel  90    ????         0  2000      3    1170

Data frame after applying ZScore normalization to "price" attribute


     Price   Age    KM FuelType  HP MetColor  Automatic    CC  Doors  Weight
0 -0.759612   NaN  46986   Diesel  90       1          0  2000      3    1165
1 -0.688799    23  72937       ??  90       ??         0  2000      3    1165
2 -0.632149  ????    NaN    ????  90      NaN         0  2000      3    1165
```

3 -0.348898 26 ?? NaN 90 0 0 2000 3 1165

4 -0.688799 ?? ???? Diesel 90 ???? 0 2000 3 1170

17.write a program to read an excel sheet and perform decimal scaling normalization on price.

```
import pandas as pd
import statistics as sts

df=pd.read_excel("income.xlsx",sheet_name="Sheet1",index_col=0)
print("Data frame before applying Decimal Scaling normalization to 'price' attribute\n\n")

def min_max_normal(a):
    m=abs(a)
    m=str(max(a))
    l=len(m)
    d=a/10**l
    return d

print(df.head())
df["Price"]=min_max_normal(df["Price"])
print("\n\nData frame after applying Decimal Scaling normalization to 'price' attribute\n\n")
print(df.head())
```

OUTPUT:

Data frame before applying Decimal Scaling normalization to 'price' attribute

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	NaN	46986	Diesel	90	1	0	2000	3	1165
1	13750	23	72937	??	90	??	0	2000	3	1165
2	13950	????	NaN	????	90	NaN	0	2000	3	1165
3	14950	26	??	NaN	90	0	0	2000	3	1165
4	13750	??	????	Diesel	90	????	0	2000	3	1170

Data frame after applying Decimal Scaling normalization to 'price' attribute

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0.1350	NaN	46986	Diesel	90	1	0	2000	3	1165

1 0.1375 23 72937 ?? 90 ?? 0 2000 3 1165

2 0.1395 ???? NaN ???? 90 NaN 0 2000 3 1165

3 0.1495 26 ?? NaN 90 0 0 2000 3 1165

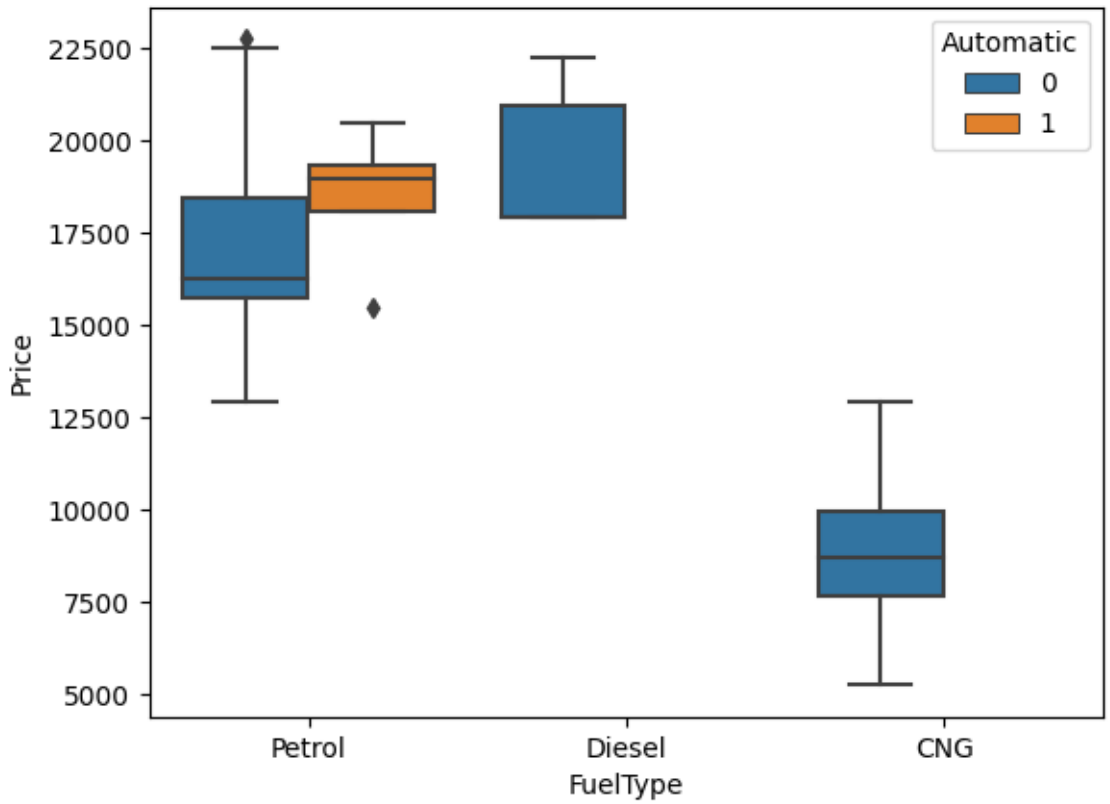
4 0.1375 ?? ???? Diesel 90 ???? 0 2000 3 1170

18.write a program to read an excel sheet and display boxplot with price vs fueltype and automatic.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_excel("income.xlsx",sheet_name='Sheet1',\
            index_col=0,na_values=["????","??"])
df.dropna(axis=0,inplace=True)
sns.boxplot(x="FuelType",y=df['Price'],hue='Automatic',data=df)
plt.show()
```

OUTPUT:

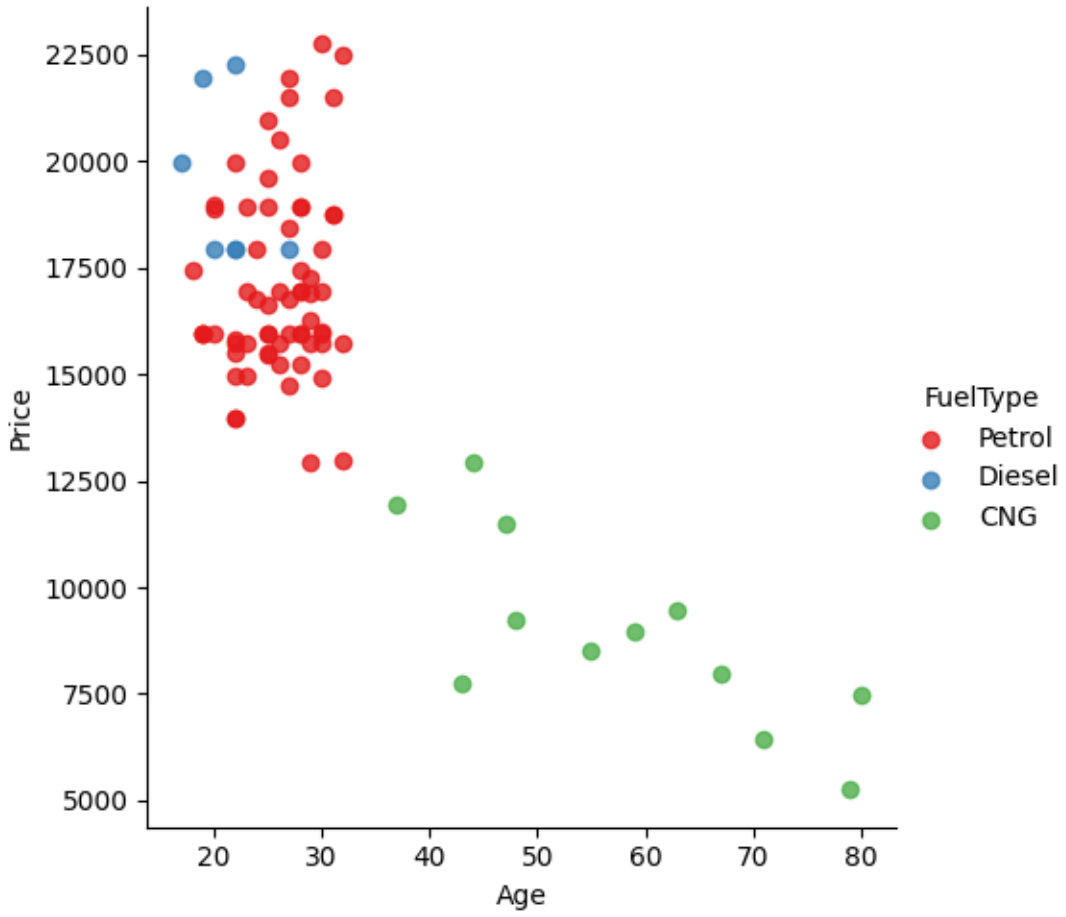


19.write a program to read an excel sheet and display scatter plot with price vs age by fueltype.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_excel("income.xlsx",sheet_name='Sheet1',\
            index_col=0,na_values=["????","??"])
df.dropna(axis=0,inplace=True)
sns.lmplot(x='Age',y='Price',data=df,fit_reg=False,hue=\
        'FuelType',legend=True,palette="Set1")
plt.show()
```

OUTPUT:



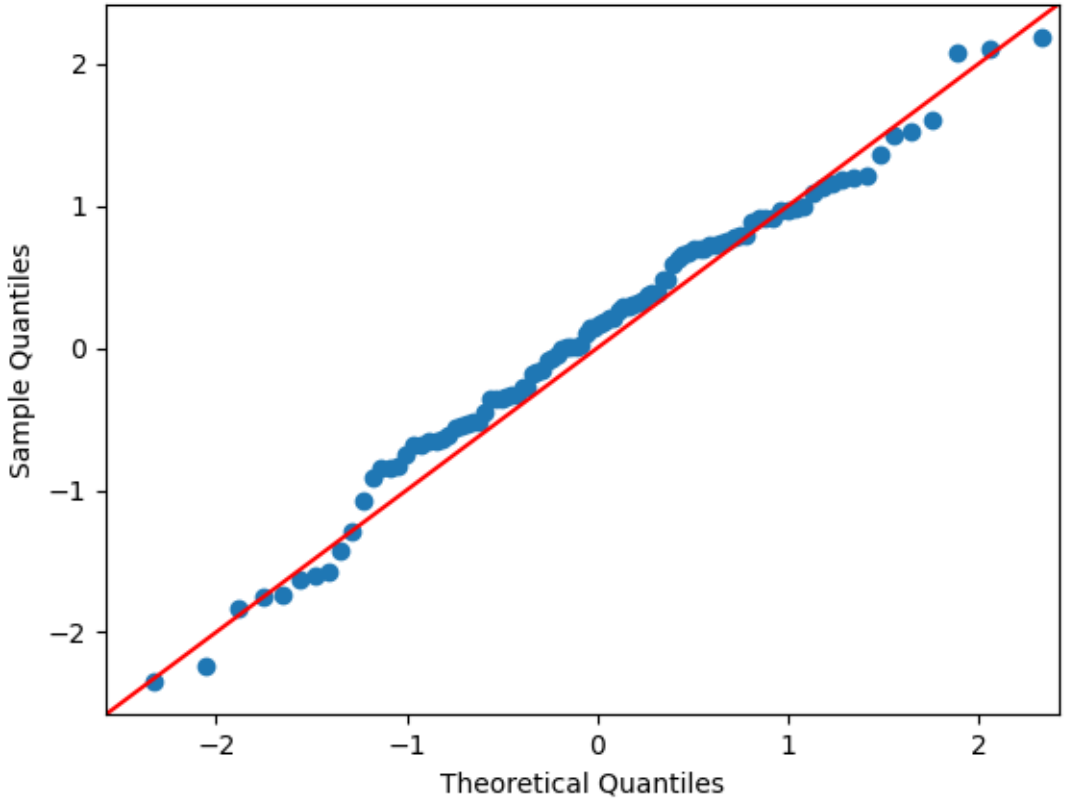
20.write a program to read an excel sheet and display Q-Q plot.

```
import numpy as np
import statsmodels.api as sm
import pylab as py

data_points = np.random.normal(0, 1, 100)
```

```
sm.qqplot(data_points, line ='45')
py.show()
```

OUTPUT:



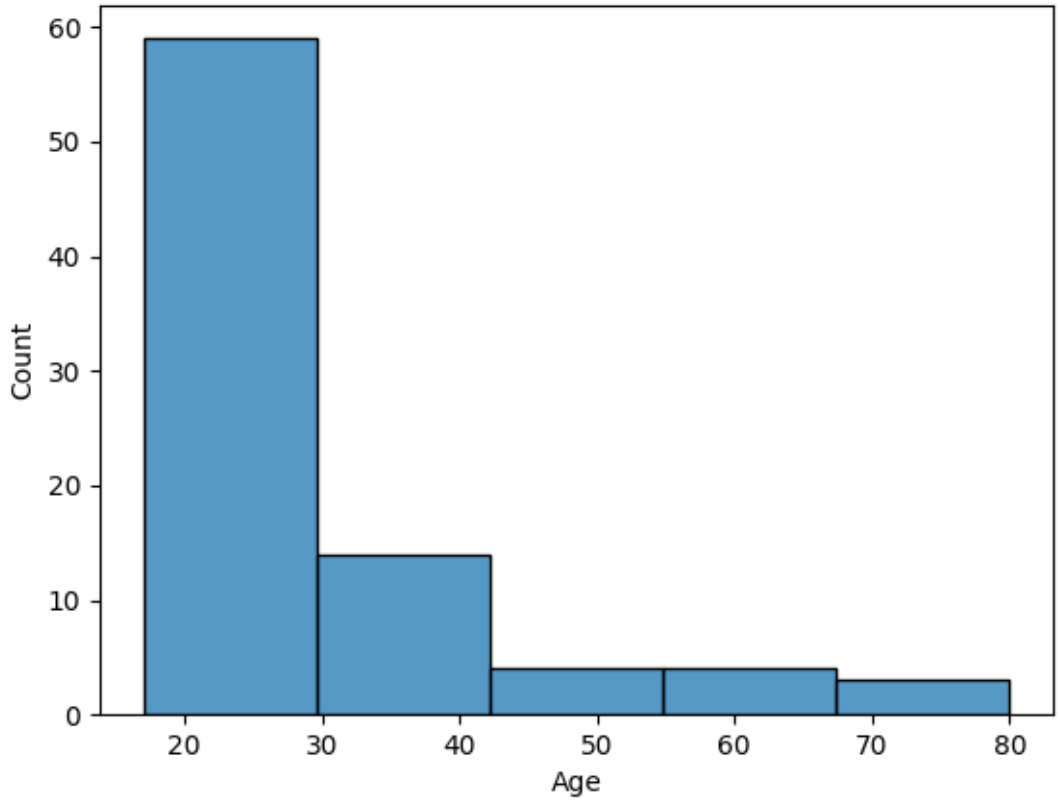
21.write a program to read an excel sheet and display histogram on age.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_excel("income.xlsx",sheet_name='Sheet1',\
        index_col=0,na_values=["????","??"])
df.dropna(axis=0,inplace=True)

sns.histplot(df['Age'],kde=False,bins=5)
plt.show()
```

OUTPUT:

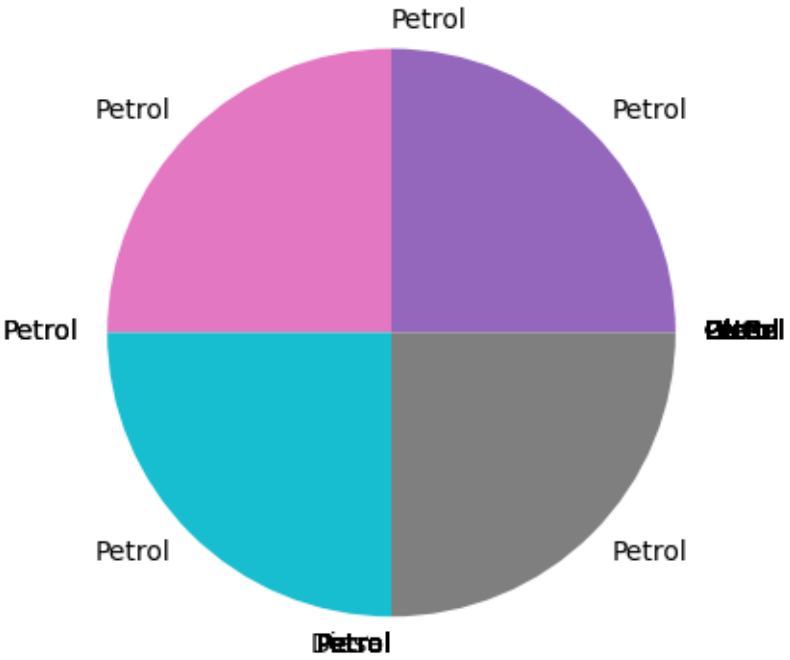


22.write a program to read an excel sheet and display piegraph.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn

df=pd.read_excel("income.xlsx",sheet_name='Sheet1',\
        index_col=0,na_values=["????","??"])
df.dropna(axis=0,inplace=True)

plt.pie(df['Automatic'],labels=df['FuelType'])
plt.show()
```



OUTPUT:

23.write a program to demonstrate PCA.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df=pd.read_excel("income2.xlsx",na_values=["????","??"],sheet_name="Sheet1",index_col=0)
df.dropna(axis=0,inplace=True)
scaler=StandardScaler()
scaler.fit(df)
scaled_data=scaler.transform(df)
pca=PCA(n_components=2)
pca.fit(scaled_data)
x_pca=pca.transform(scaled_data)
print("dimesions of before appaying pca ",scaled_data.shape)
print("dimesions of after appaying pca ",x_pca.shape)
print("the vector values are : ")
```

```
print(x_pca)
```

OUTPUT:

```
dimesions of before appaying pca  (30, 9)
dimesions of after appaying pca  (30, 2)
the vector values are :
[[-0.54978564  1.65891144]
 [-0.51775201  2.63228264]
 [ 0.33207702  1.8193292 ]
 [ 0.43192691  1.86053913]
 [ 0.40171152  2.95129382]
 [ 1.80020386  2.96789749]
 [ 2.7174894  -1.49514211]
 [ 2.49597055 -1.06563866]
 [ 2.13554587 -0.61030502]
 [ 2.2593789  -0.8483611 ]
 [ 2.50727678 -1.32522149]
 [ 2.73723621 -0.98045763]
 [ 2.66109512 -1.12503166]
 [-1.0381793  -1.41835221]
 [-1.32662939 -1.20068875]
 [-0.87258735  0.69206217]
 [-1.04908755  0.96032055]
 [ 0.62398639  1.0275045 ]
 [-0.96693403  0.51648471]
 [-0.78102847 -0.56108031]
 [-0.84066561 -0.65979736]
 [-1.13426457 -0.72430594]
 [-0.74667448 -0.59927733]
 [-0.93521583  0.01644061]
 [-0.3377844   0.0694466 ]
 [-2.24364051 -1.28710528]
 [-2.18494006 -1.05592094]
 [-1.84748754 -1.04376081]
 [-2.23131203 -0.79551529]
```

```
 [-1.49992977 -0.37655098]]
```

24. write a program to demonstrate attribute subset - stepforward selection .

```
import pandas as pd
import numpy as np
import random

df=pd.read_excel("income2.xlsx",na_values=["????","??"],sheet_name="Sheet1",index_col=0)
df.dropna(axis=0,inplace=True)
s=df.columns
ss=list()
for i in s:
    ss.append(i)
    print(df[ss].head(2))
    print()
```

OUTPUT:

```
   Price
0  13500
1  13750

   Price   Age
0  13500  23.0
1  13750  23.0

   Price   Age      KM
0  13500  23.0  46986.0
1  13750  23.0  72937.0

   Price   Age      KM    HP
0  13500  23.0  46986.0  90.0
1  13750  23.0  72937.0  90.0

   Price   Age      KM    HP  MetColor
0  13500  23.0  46986.0  90.0       1.0
1  13750  23.0  72937.0  90.0       1.0
```

```
  Price   Age     KM    HP  MetColor  Automatic
0  13500  23.0  46986.0  90.0     1.0          0
1  13750  23.0  72937.0  90.0     1.0          0

  Price   Age     KM    HP  MetColor  Automatic    CC
0  13500  23.0  46986.0  90.0     1.0          0  2000
1  13750  23.0  72937.0  90.0     1.0          0  2000

  Price   Age     KM    HP  MetColor  Automatic    CC  Doors
0  13500  23.0  46986.0  90.0     1.0          0  2000      3
1  13750  23.0  72937.0  90.0     1.0          0  2000      3

  Price   Age     KM    HP  MetColor  Automatic    CC  Doors  Weight
0  13500  23.0  46986.0  90.0     1.0          0  2000      3    1165
1  13750  23.0  72937.0  90.0     1.0          0  2000      3    1165
```

25. write a program to demonstrate attribute subset – backword elimination.

```
import pandas as pd
import numpy as np
import random

df=pd.read_excel("income2.xlsx",na_values=["????","??"],sheet_name="Sheet1",index_col=0)
df.dropna(axis=0,inplace=True)
s=df.columns
ss=list(s)
for i in s:
    print(df[ss].head(2))
    ss.pop()
    print()
```

OUTPUT:

```
  Price   Age     KM    HP  MetColor  Automatic    CC  Doors  Weight
0  13500  23.0  46986.0  90.0     1.0          0  2000      3    1165
1  13750  23.0  72937.0  90.0     1.0          0  2000      3    1165
```

```
   Price   Age      KM    HP  MetColor  Automatic    CC  Doors
0  13500  23.0  46986.0  90.0      1.0          0  2000      3
1  13750  23.0  72937.0  90.0      1.0          0  2000      3

   Price   Age      KM    HP  MetColor  Automatic    CC
0  13500  23.0  46986.0  90.0      1.0          0  2000
1  13750  23.0  72937.0  90.0      1.0          0  2000

   Price   Age      KM    HP  MetColor  Automatic
0  13500  23.0  46986.0  90.0      1.0          0
1  13750  23.0  72937.0  90.0      1.0          0

   Price   Age      KM    HP  MetColor
0  13500  23.0  46986.0  90.0      1.0
1  13750  23.0  72937.0  90.0      1.0

   Price   Age      KM    HP
0  13500  23.0  46986.0  90.0
1  13750  23.0  72937.0  90.0

   Price   Age      KM
0  13500  23.0  46986.0
1  13750  23.0  72937.0

   Price   Age
0  13500  23.0
1  13750  23.0

   Price
0  13500
1  13750
```

26. write a program to demonstrate attribute subset- combination(stepforward selection and backword elimination).

```
import pandas as pd
import numpy as np
```

```
import random
from itertools import combinations

df=pd.read_excel("income3.xlsx",na_values=["????","??"],sheet_name="Sheet1",index_col=0)
df.dropna(axis=0,inplace=True)
n=int(input("enter no. column combinations in,and its >2and  < %i"%len(df.columns)))
s=df.columns
comb=list(combinations(s,n))
comb=set(comb)
for i in comb:
    i=list(i)
    print(df[i].head(2))
    print()
```

OUTPUT:

```
enter no. column combinations in,and its >2and  < 6 : 5
   Price   Age    HP  MetColor  Automatic
0  13500  23.0  90.0      1.0          0
1  13750  23.0  90.0      1.0          0

   Price    HP  MetColor  Automatic  Doors
0  13500  90.0      1.0          0      3
1  13750  90.0      1.0          0      3

    Age    HP  MetColor  Automatic  Doors
0  23.0  90.0      1.0          0      3
1  23.0  90.0      1.0          0      3

   Price   Age    HP  MetColor  Doors
0  13500  23.0  90.0      1.0      3
1  13750  23.0  90.0      1.0      3

   Price   Age  MetColor  Automatic  Doors
0  13500  23.0      1.0          0      3
1  13750  23.0      1.0          0      3
```

```
  Price   Age    HP  Automatic  Doors
0  13500  23.0  90.0          0      3
1  13750  23.0  90.0          0      3
```

27. write a program to demonstrate simple random Sampling.

```
import pandas as pd
import numpy as np
import random

df=pd.read_excel("income1.xlsx",na_values=["????","??"],sheet_name="Sheet1",index_col=0)
df.dropna(axis=0,inplace=True)
n=int(input('enter no. rows u need and it should be < %i : '%len(df)))
l1=list(df.index.values)
s=random.sample(l1,n)
s.sort()
result = df.loc[s]
print(result)
```

OUTPUT:

```
enter no. rows u need and it should be < 76 : 4
   Price   Age      KM FuelType  ...  Automatic    CC  Doors  Weight
14  22500  32.0  34131.0   Petrol  ...          0  1800      3    1185
76  18750  31.0  25266.0   Petrol  ...          0  1600      5    1130
95  19950  17.0  30351.0   Diesel  ...          0  1995      3    1260
97  15950  19.0  25948.0   Petrol  ...          0  1400      3    1100
```

28. write a program to demonstrate simple random sampling with replacement.

```
import pandas as pd
import numpy as np
import random

df=pd.read_excel("income1.xlsx",na_values=["????","??"],sheet_name="Sheet1",index_col=0)
df.dropna(axis=0,inplace=True)
pd.set_option('display.max_rows',None)
n=int(input('enter no. rows u need and it should be < %i : '%len(df)))
```

```
l1=list(df.index.values)
s=random.sample(l1,n)
print("from sample, replce FuelType Diesel with petrol ")
s.sort()
result = df.loc[s]
result['FuelType']='Petrol'
print(result)
```

OUTPUT:

```
enter no. rows u need and it should be < 76 : 4
from sample, replce FuelType Diesel with petrol
    Price   Age      KM FuelType  ...  Automatic    CC  Doors  Weight
8   21500  27.0  19700.0   Petrol  ...         0  1800      3    1185
31  15750  22.0  35199.0   Petrol  ...         0  1400      3    1100
72  18950  28.0  28817.0   Petrol  ...         0  1598      5    1130
75  16950  23.0  28000.0   Petrol  ...         0  1600      5    1115
```

29. write a program to demonstrate stratified sampling.

```
import pandas as pd
import numpy as np
import random

df=pd.read_excel("income1.xlsx",na_values=["????","??"],sheet_name="Sheet1",index_col=0)
df.dropna(axis=0,inplace=True)
n=int(input('enter no. rows u need and it should be < %i : '%len(df)))
df1=df[df['Doors']==3]
df2=df[df['Doors']==5]
l1=list(df1.index.values)
l2=list(df2.index.values)
s1=random.sample(l1,n//2)
s2=random.sample(l2,n//2)
s=s1+s2
s.sort()
result = df.loc[s]
```

```
print("startified sampling by 'Doors',here we get equal no. of obj of Doors 3,5 ")
print(result)
```

OUTPUT:

```
enter no. rows u need and it should be < 76 : 5
startified sampling by 'Doors',here we get equal no. of obj of Doors 3,5
    Price   Age      KM FuelType  ...  Automatic    CC  Doors  Weight
23  16950  28.0  32220.0   Petrol  ...          0  1600      3    1120
30  12950  29.0   9750.0   Petrol  ...          0  1400      3    1100
51  15750  30.0  57086.0   Petrol  ...          0  1400      5    1110
70  15950  28.0  29206.0   Petrol  ...          0  1400      5    1110
```

30. write a program to demonstrate Implementation of apriori algorithm.

```
import pandas as pd
import numpy as np
import random
from itertools import combinations

def combin(lst,n):
    global comb1
    lst = list(filter(None,lst))
    if len(lst)>1:
        comb1=lst
    if type(lst)==list and len(lst)>0:
        if type(lst[0])==list:
            lst=one(lst)
    lst=set(lst)
    lst=list(lst)
    comb=list(combinations(lst,n))
    for i in range(len(comb)):
        comb[i]=list(comb[i])
    return comb

#for making one list
def one(lst):
```

```
    k=[]
    for i in lst:
        for j in i:
            if str==type(j):
                k.append(j)
    return k
df=pd.read_excel("FPgrowth.xlsx",na_values=["????","??"],sheet_name="Sheet1")
n=int(input('enter support count : '))
lst=df.values

k=one(lst)
temp=list(set(k))#for get unique items
comb=temp

for i in range(1,len(temp)+1):
    if len(comb)<1:
        break
    comb=combin(comb,i)
    d=0
    for j in comb:
        j1=j
        j=set(j)
        c=0
        for k in lst:
            k=set(k)
            if j.issubset(k):
                c+=1
        if c<n:
            for j in range(len(comb[d])):
                for i in comb[d]:
                    comb[d].remove(i)
        d+=1

print("The max combinations of items with %d support count : "%n,comb1)
```

OUTPUT:

enter support count : 2

The max combinations of items with 2 support count : [['B', 'A', 'D'], ['B', 'A', 'C'], ['B', 'D', 'C'], ['A', 'E', 'D'], ['A', 'D', 'C']]

31. write a program to demonstrate FP-Growth construction.

```
import pandas as pd
import numpy as np
import random
from itertools import combinations
import copy

def getValue(ls,v):
    ls=one(ls)
    temp=list(set(ls))
    for i in temp:
        c=0
        for j in ls:
            if i==j:
                c+=1
        if c<n:
            temp.remove(i)
    s=combin(temp,v)
    results.append(s)



def combin(lst,v):
    x=[]
    y=[]
    for i in range(1,len(lst)+1):
        x=list(combinations(lst,i))
        for i in range(len(x)):
            x[i]=list(x[i])
            x[i].append(v)
```

```
            y.append(x[i])
    return y



#for making one list
def one(lst):
    k=[]
    for i in lst:
        for j in i:
            if str==type(j):
                k.append(j)
    return k



df=pd.read_excel("FPgrowth.xlsx",na_values=["????","??"],sheet_name="Sheet1")

n=int(input('enter support count : '))
lst=df.values

k=one(lst)
uniq=list(set(k))#for get unique items
comb=[]
co=[]
slst=[]
dup=[]
results=[]
#checking min count of each unique element
for i in uniq:
    c=0
    for j in k:
        if i==j:
            c+=1
    if c>=n:
        co.append(c)
```

```
        comb.append(i)

#sorting unique elements
for i in range(len(co)-1):
    for j in range(len(co)-1):
        if co[j]<co[j+1]:
            co[j],co[j+1]=co[j+1],co[j]
            comb[j],comb[j+1]=comb[j+1],comb[j]

#sorting of each record of file by unique elements order

for i in lst:
    m=[]
    for j in comb:
        for k in range(len(i)):
            if i[k]==j:
                m.append(j)
    slst.append(m)

dup=copy.deepcopy(slst)

for i in comb[::-1]:
    m=[]
    for j in slst:
        for k in j:
            if i==k:
                j.remove(i)
                m.append(j)
    getValue(m,i)
#making result sets of each value
for i in results:
    for j in i:
        c=0
        for k in dup:
```

```
        if set(j).issubset(k):
            c+=1
    if c<2:
        i.remove(j)

i=len(comb)-1
for j in results:
  j.append(list(comb[i]))
  print("frequent item sets end with %s "%comb[i],j)
  i-=1
```

OUTPUT:

```
enter support count : 2
frequent item sets end with E  [['C', 'E'], ['D', 'E'], ['A', 'E'], ['C', 'A', 'E'], ['D', 'A', 'E'], ['E']]
frequent item sets end with D  [['C', 'D'], ['B', 'D'], ['A', 'D'], ['C', 'B', 'D'], ['C', 'A', 'D'], ['B', 'A', 'D'], ['D']]
frequent item sets end with C  [['B', 'C'], ['A', 'C'], ['B', 'A', 'C'], ['C']]
frequent item sets end with B  [['A', 'B'], ['B']]
frequent item sets end with A  [['A']]
```

32.Write a program to implement Decision tree.

```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier

df=pd.read_excel("DecisionTree.xlsx",na_values=["????","??"],sheet_name="Sheet1")
x=df.iloc[:,:-1]
y=df.iloc[:,5]
#converting to numeric data
labelencoder_x=LabelEncoder()
x=x.apply(LabelEncoder().fit_transform)
regressor=DecisionTreeClassifier()
regressor.fit(x.iloc[:,1:5],y)
m=[]
```

```
m.append(int(input(("choose one of '0,1,2' where 'age' is '21-35,<21,>35' : "))))
m.append(int(input(("choose one of '0,1,2' where 'income' is 'High,Low,Medim' : "))))
m.append(int(input(("choose one of '0,1' where 'gender' is 'Female,Male' : "))))
m.append(int(input(("choose one of '0,1' where 'maritalStatus' is 'Married,Single' : "))))
x_in=np.array(m)
y_pred=regressor.predict([x_in])
print("The buying chance for above details is : ",y_pred)
```

OUTPUT:

```
choose one of '0,1,2' where 'age' is '21-35,<21,>35' : 1
choose one of '0,1,2' where 'income' is 'High,Low,Medim' : 1
choose one of '0,1' where 'gender' is 'Female,Male' : 0
choose one of '0,1' where 'maritalStatus' is 'Married,Single' : 0
The buying chance for above details is :  ['Yes']
```

33. write a program to demonstrate naïve bayes classification.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder

df=pd.read_excel("DecisionTree.xlsx",na_values=["????","??"],sheet_name="Sheet1")
x=df.iloc[:,:-1]
y=df.iloc[:,5]
x1=df.iloc[:,:]
#converting to numeric data
labelencoder_x=LabelEncoder()
x=x.apply(LabelEncoder().fit_transform)
x1=x1.apply(LabelEncoder().fit_transform)
yes=x1['Buys'].value_counts()[1]
no=x1['Buys'].value_counts()[0]
tot=yes+no
def mul(s,g,n,v):
    c=0
    for i in range(len(x1[s])):
        if x1[s][i]==g and x1["Buys"][i]==n:
            c+=1
```

```
    return round(c/v,3)


def predict(ls):
    yy=mul("Age",ls[0],1,yes)*mul("Icome",ls[1],1,yes)*mul("Gender",ls[2],1,yes)*mul("MaritalStatus",ls[3],1,yes)
    nn=mul("Age",ls[0],0,no)*mul("Icome",ls[1],0,no)*mul("Gender",ls[2],0,no)*mul("MaritalStatus",ls[3],0,no)
    py=round(yes/tot,3)
    pn=round(no/tot,3)
    if yy*py>nn*pn:
        val="yes"
    else:
        val="No"
    return val


m=[]
m.append(int(input(("choose one of '0,1,2' where 'age' is '21-35,<21,>35' : "))))
m.append(int(input(("choose one of '0,1,2' where 'income' is 'High,Low,Medim' : "))))
m.append(int(input(("choose one of '0,1' where 'gender' is 'Female,Male' : "))))
m.append(int(input(("choose one of '0,1' where 'maritalStatus' is 'Married,Single' : "))))
print("The buying chance for above details is : ",predict(m))
```

OUTPUT:

```
choose one of '0,1,2' where 'age' is '21-35,<21,>35' : 1
choose one of '0,1,2' where 'income' is 'High,Low,Medim' : 2
choose one of '0,1' where 'gender' is 'Female,Male' : 0
choose one of '0,1' where 'maritalStatus' is 'Married,Single' : 1
The buying chance for above details is :  yes
```

34. write a program to demonstrate K-means clustering algorithm.

```
a=eval(input("enter some values as '[1,2,3,3]' for making '2' clusters : "))
k11=a[0]
k22=a[len(a)-1]
val=True
def mean(ls):
    c=0
    for i in ls:
        c+=i
```

```
    return round(c/len(ls))
while val:
    k1=[]
    k2=[]
    for i in a:
        if abs(i-k11)<abs(i-k22):
            k1.append(i)
        else:
            k2.append(i)
    t1=mean(k1)
    t2=mean(k2)
    if t1==k11 and t2==k22:
        val=False
    else:
        k11,k22=t1,t2
print("final clusters are : ",k1,k2)
```

OUTPUT:

```
enter some values as '[1,2,3,3]' for making '2' clusters : [6,4,8,7,2,4,3,4,5]
final clusters are :  [6, 8, 7] [4, 2, 4, 3, 4, 5]
```

35. write a program to demonstrate k-medoid clustering algorithm.

```
a=eval(input("enter some values as '[2,3,3,4,6,6,7,7,8,7]' for making '2' clusters : "))
b=eval(input("enter some values as '[6,4,8,7,2,4,3,4,5,6]' both same length : "))
k11=0
k22=len(a)-1
val=True
cop=0
while val:
    c=0
    k1=[k11]
    k2=[k22]
    for i in range(0,len(a)):
        if i==k11 or i==k22:
            continue
        x=abs(a[i]-a[k11])+abs(b[i]-b[k11])
```

```
        y=abs(a[i]-a[k22])+abs(b[i]-b[k22])
        if x<y:
            c+=x
            k1.append(i)
        else:
            c+=y
            k2.append(i)
    k11+=1
    if cop==0 or c-cop<=0:
        cop=c
        fin1=k1
        fin2=k2
    else:
        val=False
def results(k):
    res=[]
    for i in k:
        res.append([a[i],b[i]])
    return res
print("1st cluster : ",results(fin1))
print("2nd cluster : ",results(fin2))
```

OUTPUT:

```
enter some values as '[2,3,3,4,6,6,7,7,8,7]' for making '2' clusters : [2,3,3,4,6,6,7,7,8,7]
enter some values as '[6,4,8,7,2,4,3,4,5,6]' both same length : [6,4,8,7,2,4,3,4,5,6]
1st cluster :  [[2, 6], [3, 4], [3, 8], [4, 7]]
2nd cluster :  [[7, 6], [6, 2], [6, 4], [7, 3], [7, 4], [8, 5]]
```