CS 584: Machine Learning

Pradyot Mayank
A20405826
Assignment 1

Question 1 (40 points)

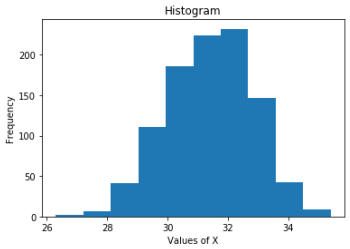
Write a Python program to calculate the density estimator of a histogram. Use the field *x* in the NormalSample.csv file.

a) (5 points) According to Izenman (1991) method, what is the recommended bin-width for the histogram of x?

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math


data = pd.read_csv("E:\\Local Disk D\\IIT-C\Sem 4\\CS 584 Machine Learning\\Homeworks\\Homework 1\\NormalSample.csv")

df = pd.DataFrame(data)

noOfobservations=df['x']

N=df.x.count()

# print(df)
print(N)
```

```
1001
```

```
In [250]: # Visualization of the variable x
plt.hist(noOfobservations)
plt.title("Histogram")
plt.xlabel("Values of X")
plt.ylabel("Frequency")
plt.show()
```




```
In [91]: #Binwidth According to Izenman (1991) method
median =np.median(noOfobservations)

LowerQuartile=np.percentile(noOfobservations,25)

UpperQuartile=np.percentile(noOfobservations,75)

InterQuartile=UpperQuartile-LowerQuartile

IQR=InterQuartile

binwidth=2*(IQR)*N ** (-1. / 3)

# print("Median ",median)

# print("LowerQuartile ",LowerQuartile)

# print("UpperQuartile ",UpperQuartile)

# print("InterQuartile ",IQR)

print("1(a) Binwidth:", format(binwidth))
```

```
1(a) Binwidth: 0.3998667554864774
```

So, binwidth according to Izenman (1991) method is 0.3998667554864774.

b) (5 points) What are the minimum and the maximum values of the field x?

```
In [92]: #Min and Max Value of field X

         MaxValue=max(noOfobservations)
         MinValue=min(noOfobservations)

         print("1(b)")
         print("MaxValue ",MaxValue)

         print("MinValue ",MinValue)

         1(b)
         MaxValue  35.4
         MinValue  26.3
```

Minimum Value of x= 26.3
Maximum Value of x=35.4

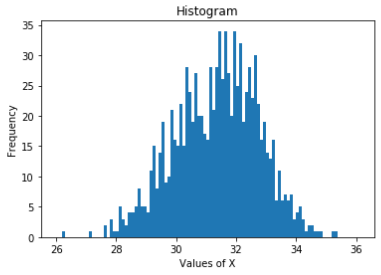
c) (5 points) Let a be the largest integer less than the minimum value of the field x, and b be the smallest integer greater than the maximum value of the field x. What are the values of a and b?

```
In [93]: #Next Maximum and Minimum integer value for X

         a=math.floor(MinValue)
         b=math.ceil(MaxValue)

         print("1(c)")
         print("NextMin ",a)

         print("NextMax ",b)

         1(c)
         NextMin  26
         NextMax  36
```

Value of a=26.
Value of b=36.

d) (5 points) Use h = 0.1, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

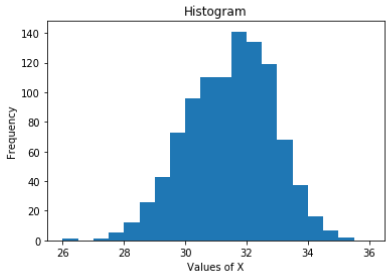
```
In [94]: #h = 0.1, minimum = a and maximum = b

         # plt.hist(noOfobservations)

         print("1(d)")
         binwidth_h=0.1

         plt.hist(noOfobservations, bins=np.arange(a, b + binwidth_h, binwidth_h))

         # plt.hist(noOfobservations, bins=((b-a)/binwidth_h))
         plt.title("Histogram")
         plt.xlabel("Values of X")
         plt.ylabel("Frequency")
         plt.show()

         1(d)
```



e) (5 points) Use h = 0.5, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

```
In [95]: #h = 0.5, minimum = a and maximum = b

         # plt.hist(noOfobservations)

         print("1(e)")
         binwidth_h=0.5

         plt.hist(noOfobservations, bins=np.arange(a, b + binwidth_h, binwidth_h))

         # plt.hist(noOfobservations, bins=np.arange(a, b, binwidth_h))
         plt.title("Histogram")
         plt.xlabel("Values of X")
         plt.ylabel("Frequency")
         plt.show()
```

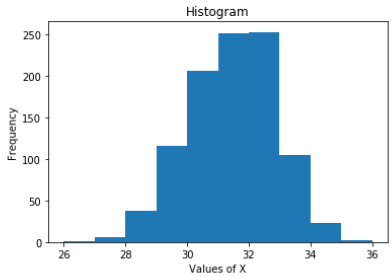
```
1(e)
```



f) (5 points) Use h = 1, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

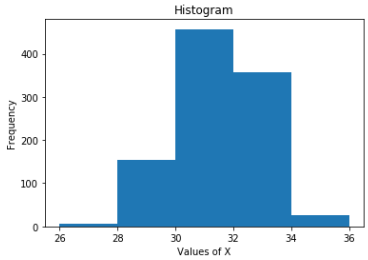
```
In [96]: #h = 1, minimum = a and maximum = b

         # plt.hist(noOfobservations)

         print("1(f)")
         binwidth_h=1

         plt.hist(noOfobservations, bins=np.arange(a, b + binwidth_h, binwidth_h))
         plt.title("Histogram")
         plt.xlabel("Values of X")
         plt.ylabel("Frequency")
         plt.show()
```

```
1(f)
```



g) (5 points) Use h = 2, minimum = a and maximum = a. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

```
In [97]: #h = 2, minimum = a and maximum = b

         # plt.hist(noOfobservations)

         print("1(g)")
         binwidth_h=2

         plt.hist(noOfobservations, bins=np.arange(a, b + binwidth_h, binwidth_h))
         plt.title("Histogram")
         plt.xlabel("Values of X")
         plt.ylabel("Frequency")
         # plt.grid(axis="x")
         # plt.grid(axis="y")
         plt.show()

         1(g)
```



h) (5 points) Among the four histograms, which one, in your honest opinions, can best provide your insights into the shape and the spread of the distribution of the field x? Please state your arguments.

The histogram in part (e) with binwidth=0.5 can best provide the insights into the shape and the spread of the distribution of the field x. As the binwidth calculated in part (a) is 0.4 (upto 1 decimal place). The nearest next binwidth is 0.5 in part (e) and also it provides almost symmetric distribution of data, not skewed left or right.

Question 2 (20 points)

Use in the NormalSample.csv to generate box-plots for answering the following questions.

a) (5 points) What are the five-number summary of x? What are the values of the 1.5 IQR whiskers?

```
In [98]: #Five number summary of the box plot

         Median=np.median(noOfobservations)

         LowerQuartile=np.percentile(noOfobservations,25)

         Q1=LowerQuartile

         UpperQuartile=np.percentile(noOfobservations,75)

         Q3=UpperQuartile

         MaxValue=max(noOfobservations)

         MinValue=min(noOfobservations)

         InterQuartile=UpperQuartile-LowerQuartile

         IQR=InterQuartile


         print("2(a)")
         print("MinValue ",MinValue)

         print("LowerQuartile ",LowerQuartile)

         print("Median ",Median)

         print("UpperQuartile ",UpperQuartile)

         print("MaxValue ",MaxValue)

         2(a)
         MinValue  26.3
         LowerQuartile  30.4
         Median  31.5
         UpperQuartile  32.4
         MaxValue  35.4
```

Five number summary of x:

Minimum Value=26.3
Lower Quartile Q1=30.4
Median =31.5
Upper Quartile Q3=32.4
Maximum Value= 35.4

```
In [99]: # Values of the 1.5 IQR whiskers

         Lowerwhisker=Q1-1.5*IQR

         Upperwhisker=Q3+1.5*IQR

         print("2(a)")

         print("InterQuartile ",IQR)
         print("Lowerwhisker ",Lowerwhisker)

         print("Upperwhisker ",Upperwhisker)


         2(a)
         InterQuartile  2.0
         Lowerwhisker  27.4
         Upperwhisker  35.4
```

Value of whiskers:

Lower Whisker= 27.4
Upper Whisker= 35.4

b) (5 points) What are the five-number summary of x for each category of the group? What are the values of the 1.5 IQR whiskers for each category of the group?

In [152]:
```
#Five-number summary of x for category one of the group

isOne=(df.group==1)

groupOne = data[isOne]['x']

# print(One.head(20))

MinValueOne=min(groupOne)

LowerQuartileOne=np.percentile(groupOne,25)

MedianOne=np.median(groupOne)

UpperQuartileOne=np.percentile(groupOne,75)

MaxValueOne=max(groupOne)

InterQuartileOne=UpperQuartileOne-LowerQuartileOne

LowerwhiskerOne = LowerQuartileOne-1.5*InterQuartileOne

UpperwhiskerOne = UpperQuartileOne+1.5*InterQuartileOne

print("2(b)")
print("Min Value of group One ",MinValueOne)

print("Lower Quartile of group One ",LowerQuartileOne)

print("Median of group One ",Medianone)

print("Upper Quartile of group One ",UpperQuartileOne)

print("Max Value of group One ",MaxValueOne)

print("Lower whisker of group One ",LowerwhiskerOne)

print("Upper whisker of group One ",UpperwhiskerOne)
```

```
2(b)
Min Value of group One  29.1
Lower Quartile of group One  31.4
Median of group One  32.1
Upper Quartile of group One  32.7
Max Value of group One  35.4
Lower whisker of group One  29.449999999999992
Upper whisker of group One  34.650000000000006
```

Five number summary for category One:

Minimum value =29.1
Lower Quartile=31.4
Median =32.1
Upper Quartile= 32.7
Maximum value=35.4

1.5IQR wishkers:

Lower whisker= 29.449999999999992
Upper Whisker= 34.650000000000006

```
In [154]: #Five-number summary of x for category Zero of the group

          isZero=(df.group!=1)

          groupZero = data[isZero]['x']

          MinValueZero=min(groupZero)

          LowerQuartileZero=np.percentile(groupZero,25)

          MedianZero=np.median(groupZero)

          UpperQuartileZero=np.percentile(groupZero,75)

          MaxValueZero=max(groupZero)

          InterQuartileZero=UpperQuartileZero-LowerQuartileZero

          LowerwhiskerZero = LowerQuartileZero-1.5*InterQuartileZero

          UpperwhiskerZero = UpperQuartileZero+1.5*InterQuartileZero


          print("2(b)")
          print("Min Value of group Zero ",MinValueZero)

          print("Lower Quartile of group Zero ",LowerQuartileZero)

          print("Median of group Zero",Medianzero)

          print("Upper Quartile of group Zero ",UpperQuartileZero)

          print("Max Value of group Zero ",MaxValueZero)

          print("Lower whisker of group Zero ",LowerwhiskerZero)

          print("Upper whisker of group Zero ",UpperwhiskerZero)

          2(b)
          Min Value of group Zero  26.3
          Lower Quartile of group Zero  29.4
          Median of group Zero 30.0
          Upper Quartile of group Zero  30.6
          Max Value of group Zero  32.2
          Lower whisker of group Zero  27.599999999999994
          Upper whisker of group Zero  32.400000000000006
```

Five number summary for category Zero:

Minimum value =26.3
Lower Quartile=29.4
Median =30.0
Upper Quartile= 30.6
Maximum value=32.2

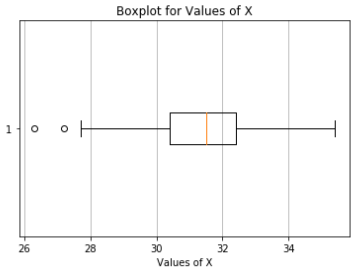
1.5IQR wishkers:

Lower whisker= 27.599999999999994
Upper Whisker= 32.400000000000006

c) (5 points) Draw a boxplot of x (without the group) using the Python boxplot function. Can you tell if the Python's boxplot has displayed the 1.5 IQR whiskers correctly?

```
In [104]: # Visualization values of x using the boxplot

          print("2(c)")
          plt.boxplot(noOfobservations,vert=False)
          plt.title("Boxplot for Values of X")
          plt.xlabel("Values of X")
          plt.grid(axis="x")
          plt.show()
```

```
2(c)
```

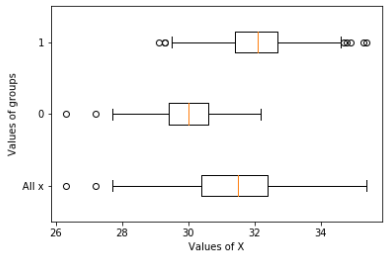


Yes, the above boxplot of value of x has displayed the 1.5IQR whiskers correctly

d) (5 points) Draw a graph where it contains the boxplot of x, the boxplot of x for each category of Group (i.e., three boxplots within the same graph frame). Use the 1.5 IQR whiskers, identify the outliers of x, if any, for the entire data and for each category of Group.
Hint: Consider using the CONCAT function in the PANDA module to append observations.

```
In [141]: #Five number summary of x for each category of the group

          isOne=(df.group==1)

          isZero=(df.group!=1)

          One = data[isOne]['x']
          Zero = data[isZero]['x']

          print("2(d)")
          fig = plt.figure()
          ax = fig.add_subplot(111)
          # ax.boxplot(noOfobservations,vert=False)
          ax.boxplot([noOfobservations,Zero,One],labels=['All x','0', '1'],vert=False)
          # ax.boxplot([One,Zero], labels=['1', '0'],vert=False)

          plt.xlabel("Values of X")
          plt.ylabel("Values of groups")
          plt.show()
```

```
2(d)
C:\Program Files\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:52: FutureWarning: reshape is deprecated and will raise
in a subsequent release. Please use .values.reshape(...) instead
  return getattr(obj, method)(*args, **kwds)
```



Outliers of x for all data:

In [296]:
```
#Outliers for the entire data

# Lowerwhisker  27.4
# Upperwhisker  35.4

print("2(d)")
outliersBelowLowerwhisker=noOfobservations[noOfobservations<Lowerwhisker]

print(outliersBelowLowerwhisker)

outliersAboveUpperwhisker=noOfobservations[noOfobservations>Upperwhisker]

print(outliersAboveUpperwhisker)
```

```
2(d)
70     27.2
295    26.3
Name: x, dtype: float64
Series([], Name: x, dtype: float64)
```

Outliers for group one:

In [169]:
```
#Outliers for the group one

#Lower whisker of group One  29.449999999999992
#Upper whisker of group One  34.650000000000006

print("2(d)")
outliersLowerwhiskerone=groupOne[groupOne<LowerwhiskerOne]

print("Outliers of Lower Whisker for group one \n",outliersLowerwhiskerone)

outliersUpperwhiskerone=groupOne[groupOne>UpperwhiskerOne]

print("Outliers of Upper Whisker for group one \n",outliersUpperwhiskerone)
```

```
2(d)
Outliers of Lower Whisker for group one
 107    29.3
938    29.3
975    29.1
Name: x, dtype: float64
Outliers of Upper Whisker for group one
 30     35.3
297    35.4
812    34.9
846    34.7
907    34.8
Name: x, dtype: float64
```

Outliers for group zero:

In [168]:
```
#Outliers for the group Zero

#Lower whisker of group Zero  27.599999999999994
#Upper whisker of group Zero  32.400000000000006

print("2(d)")
outliersLowerwhiskerzero=groupZero[groupZero<LowerwhiskerZero]

print("Outliers of Lower Whisker for group zero \n",outliersLowerwhiskerzero)

outliersUpperwhiskerzero=groupZero[groupZero>UpperwhiskerZero]

print("Outliers of Upper Whisker for group zero \n",outliersUpperwhiskerzero)
```

```
Outliers of Lower Whisker for group zero
 70     27.2
295    26.3
Name: x, dtype: float64
Outliers of Upper Whisker for group zero
 Series([], Name: x, dtype: float64)
```

Question 3 (40 points)

The data, FRAUD.csv, contains results of fraud investigations of 5,960 cases. The binary variable FRAUD indicates the result of a fraud investigation: 1 = Fraudulent, 0 = Otherwise. The other interval variables contain information about the cases.

1. TOTAL_SPEND: Total amount of claims in dollars
2. DOCTOR_VISITS: Number of visits to a doctor
3. NUM_CLAIMS: Number of claims made recently
4. MEMBER_DURATION: Membership duration in number of months
5. OPTOM_PRESC: Number of optical examinations
6. NUM_MEMBERS: Number of members covered

You are asked to use the Nearest Neighbors algorithm to predict the likelihood of fraud.

a) (5 points) What percent of investigations are found to be fraudulent? Please give your answer up to 4 decimal places.

```
In [108]: #Percent of the fradulant data

totalData=df.FRAUD.count()

FraudData=(df.FRAUD == 1).sum()

# fraudData=df[df['FRAUD'] == 1].count()

percentFraud=(FraudData/totalData)*100

print("3(a)")

print("Percentage of fradulant data ",round(percentFraud,4))
```

```
3(a)
Percentage of fradulant data  19.9497
```

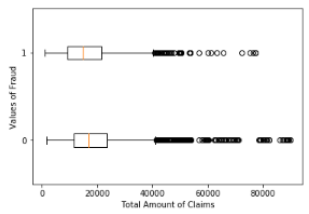
Percent of Fraudulent data =19.9497

b) (5 points) Use the BOXPLOT function to produce horizontal box-plots. For each interval variable, one box-plot for the fraudulent observations, and another box-plot for the non-fraudulent observations. These two box-plots must appear in the same graph for each interval variable.

1. Total Money Spend

```
In [109]: # Visualization of total spend using boxplot

print("3(b)")

isFradulantData=(df.FRAUD==1)

isOtherwiseData=(df.FRAUD!=1)

FradulantData = fraudData[isFradulantData]

OtherwiseData=fraudData[isOtherwiseData]

FradulantData = fraudData[isFradulantData]['TOTAL_SPEND']
OtherwiseData = fraudData[isOtherwiseData]['TOTAL_SPEND']
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot([OtherwiseData,FradulantData], labels=['0', '1'],vert=False)

plt.ylabel("Values of Fraud")
plt.xlabel("Total Amount of Claims")
plt.show()
```

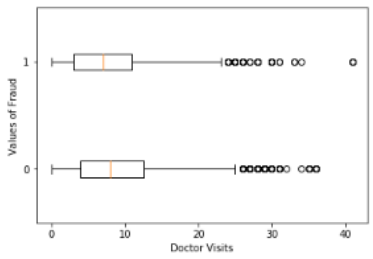
```
3(b)
C:\Program Files\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:52: FutureWarning: reshape is deprecated and will raise
in a subsequent release. Please use .values.reshape(...) instead
  return getattr(obj, method)(*args, **kwds)
```



2. Doctor Visits

In [110]:
```
# Visualization of doctor visits using boxplot

print("3(b)")
isFradulantData=(df.FRAUD==1)

isOtherwiseData=(df.FRAUD!=1)

FradulantData = fraudData[isFradulantData]
OtherwiseData=fraudData[isOtherwiseData]
FradulantData = fraudData[isFradulantData]['DOCTOR_VISITS']
OtherwiseData = fraudData[isOtherwiseData]['DOCTOR_VISITS']
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot([OtherwiseData,FradulantData], labels=['0', '1'],vert=False)
plt.ylabel("Values of Fraud")
plt.xlabel("Doctor Visits")
plt.show()
```

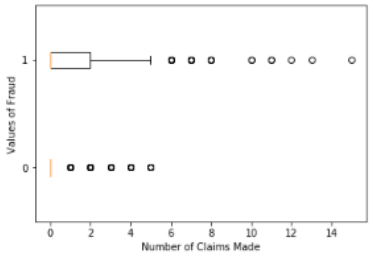
```
3(b)
C:\Program Files\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:52: FutureWarning: reshape is deprecated and will raise
in a subsequent release. Please use .values.reshape(...) instead
  return getattr(obj, method)(*args, **kwds)
```



3. Number of Claims

In [111]:
```
# Visualization of number of claims using boxplot

print("3(b)")
isFradulantData=(df.FRAUD==1)

isOtherwiseData=(df.FRAUD!=1)

FradulantData = fraudData[isFradulantData]

OtherwiseData=fraudData[isOtherwiseData]

FradulantData = fraudData[isFradulantData]['NUM_CLAIMS']
OtherwiseData = fraudData[isOtherwiseData]['NUM_CLAIMS']
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot([OtherwiseData,FradulantData], labels=['0', '1'],vert=False)

plt.ylabel("Values of Fraud")
plt.xlabel("Number of Claims Made")
plt.show()
```

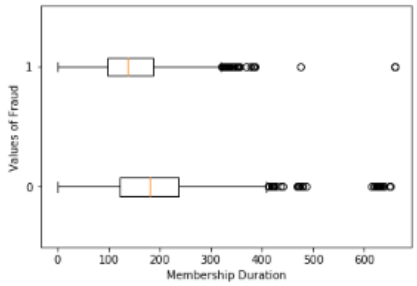
```
3(b)
C:\Program Files\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:52: FutureWarning: reshape is deprecated and will raise
in a subsequent release. Please use .values.reshape(...) instead
  return getattr(obj, method)(*args, **kwds)
```



4. Membership Duration

In [112]:
```
# Visualization of membership duration using boxplot

print("3(b)")
isFradulantData=(df.FRAUD==1)

isOtherwiseData=(df.FRAUD!=1)

FradulantData = fraudData[isFradulantData]
OtherwiseData=fraudData[isOtherwiseData]

FradulantData = fraudData[isFradulantData]['MEMBER_DURATION']
OtherwiseData = fraudData[isOtherwiseData]['MEMBER_DURATION']
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot([OtherwiseData,FradulantData], labels=['0', '1'],vert=False)
plt.ylabel("Values of Fraud")
plt.xlabel("Membership Duration")
plt.show()
```

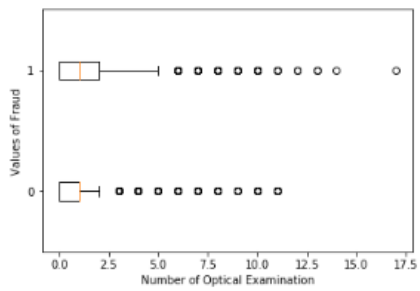
```
3(b)
C:\Program Files\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:52: FutureWarning: reshape is deprecated and will raise
in a subsequent release. Please use .values.reshape(...) instead
  return getattr(obj, method)(*args, **kwds)
```



5. Optical examination

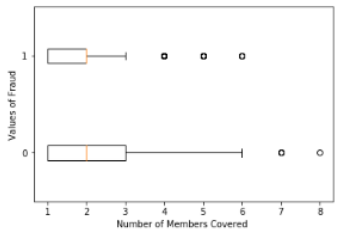
In [113]:
```
# Visualization of optical examination using boxplot

print("3(b)")

isFradulantData=(df.FRAUD==1)

isOtherwiseData=(df.FRAUD!=1)

FradulantData = fraudData[isFradulantData]
OtherwiseData=fraudData[isOtherwiseData]

FradulantData = fraudData[isFradulantData]['OPTOM_PRESC']
OtherwiseData = fraudData[isOtherwiseData]['OPTOM_PRESC']
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot([OtherwiseData,FradulantData], labels=['0', '1'],vert=False)
plt.ylabel("Values of Fraud")
plt.xlabel("Number of Optical Examination")
plt.show()
```

```
3(b)
C:\Program Files\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:52: FutureWarning: reshape is deprecated and will raise
in a subsequent release. Please use .values.reshape(...) instead
  return getattr(obj, method)(*args, **kwds)
```



6. Number of Members

```
In [114]: # Visualization of number of members using boxplot

          print("3(b)")

          isFradulantData=(df.FRAUD==1)

          isOtherwiseData=(df.FRAUD!=1)

          FradulantData = fraudData[isFradulantData]
          OtherwiseData=fraudData[isOtherwiseData]

          FradulantData = fraudData[isFradulantData]['NUM_MEMBERS']
          OtherwiseData = fraudData[isOtherwiseData]['NUM_MEMBERS']

          fig = plt.figure()
          ax = fig.add_subplot(111)
          ax.boxplot([OtherwiseData,FradulantData], labels=['0', '1'],vert=False)
          plt.ylabel("Values of Fraud")
          plt.xlabel("Number of Members Covered ")
          plt.show()

          3(b)

          C:\Program Files\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:52: FutureWarning: reshape is deprecated and will raise
          in a subsequent release. Please use .values.reshape(...) instead
            return getattr(obj, method)(*args, **kwds)
```



c) (10 points) Orthonormalize interval variables and use the resulting variables for the nearest neighbor analysis. Use only the dimensions whose corresponding eigenvalues are greater than one.

i. (5 points) How many dimensions are used?

```
In [118]: #Eigen values and Eigenvectors
          evals, evecs = la2.eigh(transposeMatrix)
          print("3(c)(i)")
          print("Eigenvalues of transposeMatrix = \n\n", evals)
          print("Eigenvectors of transposeMatrix = \n\n",evecs)

          3(c)(i)
          Eigenvalues of transposeMatrix =

           [6.84728061e+03 8.38798104e+03 1.80639631e+04 3.15839942e+05
           8.44539131e+07 2.81233324e+12]
          Eigenvectors of transposeMatrix =

           [[-5.37750046e-06 -2.20900379e-05  3.62806809e-05 -1.36298664e-04
            -7.26453432e-03  9.99973603e-01]
           [ 6.05433402e-03 -2.69942162e-02  1.27528313e-02  9.99013423e-01
              3.23120126e-02  3.69879256e-04]
           [-9.82198935e-01  1.56454700e-01 -1.03312781e-01  1.14463687e-02
              1.62110700e-03  1.52596881e-05]
           [ 1.59310591e-04 -4.91894718e-03  3.11864824e-03 -3.25018102e-02
              9.99428355e-01  7.25592222e-03]
           [ 6.90939783e-02 -2.10615119e-01 -9.75101628e-01  6.26672294e-03
              2.19857585e-03  4.79234486e-05]
           [ 1.74569737e-01  9.64577791e-01 -1.95782843e-01  2.73038995e-02
              6.21788707e-03  7.82430481e-05]]
```

As we can see from the above screenshot, six dimensions are used.
All the eigenvalues are greater than one.

ii. (5 points) Please provide the transformation matrix? You must provide proof that the resulting variables are actually orthonormal.

Transformation Matrix:

```
In [119]: #Transformation matrix

          print("3(c)(ii)")
          transformationMatrix = evecs * la2.inv(np.sqrt(np.diagflat(evals)))
          print("Transformation Matrix = \n\n", transformationMatrix)
```

```
3(c)(ii)
Transformation Matrix =

 [[-6.49862374e-08 -2.41194689e-07  2.69941036e-07 -2.42525871e-07
  -7.90492750e-07  5.96286732e-07]
 [ 7.31656633e-05 -2.94741983e-04  9.48855536e-05  1.77761538e-03
   3.51604254e-06  2.20559915e-10]
 [-1.18697179e-02  1.70828329e-03 -7.68683456e-04  2.03673350e-05
   1.76401304e-07  9.09938972e-12]
 [ 1.92524315e-06 -5.37085514e-05  2.32038406e-05 -5.78327741e-05
   1.08753133e-04  4.32672436e-09]
 [ 8.34989734e-04 -2.29964514e-03 -7.25509934e-03  1.11508242e-05
   2.39238772e-07  2.85768709e-11]
 [ 2.10964750e-03  1.05319439e-02 -1.45669326e-03  4.85837631e-05
   6.76601477e-07  4.66565230e-11]]
```

Proof that the resulting variables are orthonormal:

```
In [123]: # Identity Matrix to prove the the matrix is orthonormalization


          print("3(c)(ii)")
          xtx = transf_im.transpose()*transf_im

          # print(np.shape(xtx))

          print("Expect an Identity Matrix = \n\n", xtx)
```

```
3(c)(ii)
Expect an Identity Matrix =

 [[ 1.00000000e+00 -3.00432422e-16 -4.61219604e-16  5.45323877e-15
   1.20996962e-15 -1.28911638e-16]
 [-3.00432422e-16  1.00000000e+00 -6.44449771e-16 -2.76820667e-14
  -1.23512311e-15  7.78890841e-16]
 [-4.61219604e-16 -6.44449771e-16  1.00000000e+00  3.50891191e-15
   1.00613962e-16 -2.25514052e-16]
 [ 5.45323877e-15 -2.76820667e-14  3.50891191e-15  1.00000000e+00
   1.14860378e-14 -3.47812057e-15]
 [ 1.20996962e-15 -1.23512311e-15  1.00613962e-16  1.14860378e-14
   1.00000000e+00 -6.31439345e-16]
 [-1.28911638e-16  7.78890841e-16 -2.25514052e-16 -3.47812057e-15
  -6.31439345e-16  1.00000000e+00]]
```

d) (10 points) Use the NearestNeighbors module to execute the Nearest Neighbors algorithm using exactly five neighbors and the resulting variables you have chosen in c). The KNeighborsClassifier module has a score function.

i. (5 points) Run the score function, provide the function return value.

```
In [124]: # Nearest Neighbors module

          from sklearn.neighbors import KNeighborsClassifier

          #Transform data as traindata

          trainData = transf_im


          targetData = df['FRAUD']


          KNeighbor = KNeighborsClassifier(n_neighbors=5 , algorithm = 'brute', metric = 'euclidean')
          nbrs = KNeighbor.fit(trainData, targetData)

          print(nbrs)
```

```
KNeighborsClassifier(algorithm='brute', leaf_size=30, metric='euclidean',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
In [125]: score=nbrs.score(trainData,np.array(targetData))

          print("3(d)(i)")
          print(score)

          3(d)(i)
          0.8778523489932886
```

The score function return value is **0.8779** up to four decimal places.

ii. (5 points) Explain the meaning of the score function return value.

Score function tells us the accuracy of our model which we have trained on our training data.

e) (5 points) For the observation which has these input variable values: TOTAL_SPEND = 7500, DOCTOR_VISITS = 15, NUM_CLAIMS = 3, MEMBER_DURATION = 127, OPTOM_PRESC = 2, and NUM_MEMBERS = 2, find its **five** neighbors. Please list their input variable values and the target values. *Reminder: transform the input observation using the results in c) before finding the neighbors*.

```
In [224]: # Observation of input variables

          inputVariables = pd.DataFrame(columns=["TOTAL_SPEND", "DOCTOR_VISITS", "NUM_CLAIMS", "MEMBER_DURATION", "OPTOM_PRESC", "OPTOM_PRE
                                  data=[[7500,15,3,127,2,2]])



          print("3(e)")
          inputMatrix=np.matrix(inputVariables)

          print(inputMatrix)

          transInputMatrix = inputMatrix * transformationMatrix;

          print(transInputMatrix)


          myNeighbors = nbrs.kneighbors(transInputMatrix, return_distance = False)
          print("Nearest Neighbors = \n\n", myNeighbors)

          3(e)
          [[7500   15    3  127    2    2]]
          [[-0.02886529  0.00853837 -0.01333491  0.0176811   0.00793805  0.0044727 ]]
          Nearest Neighbors =

           [[ 588 2897 1199 1246  886]]
```

```
In [239]: #Values of all the target values
          #Since the index starts from 0 so we subtract 1 from each neighbour
          print("3(e)")
          targetData[[588-1, 2897-1 ,1199-1, 1246-1 , 886-1]]

Out[239]: 587     1
          2896    1
          1198    1
          1245    0
          885     0
          Name: FRAUD, dtype: int64
```

```
In [181]: #Values of all the input
          #Since the index starts from 0 so we subtract 1 from each neighbour
          print("3(e)")
          print(intervalMatrix[588-1])
          print(intervalMatrix[2897-1])
          print(intervalMatrix[1199-1])
          print(intervalMatrix[1246-1])
          print(intervalMatrix[886-1])

          [7500    6    4  345    1    1]
          [16000     3     0   190     0     3]
          [10000    15     2   109     3     1]
          [10200     1     0   105     1     1]
          [8900   18    0  280    0    1]
```

f) (5 points) Follow-up with e), what is the predicted probability of fraudulent (i.e., FRAUD = 1)? If your predicted probability is greater than or equal to your answer in a), then the observation will be classified as fraudulent. Otherwise, non-fraudulent. Based on this criterion, will this observation be misclassified?

```
In [244]: nbrs.predict(transInputMatrix)

          prediction=nbrs.predict(transInputMatrix)

          print("3(f)")
          print(prediction)

          3(f)
          [1]
```

```
In [247]: class_proba=nbrs.predict_proba(inputMatrix)
          print("3(f)")
          print(class_proba)

          3(f)
          [[0.8 0.2]]
```

The predicted probability of fraudulent is 20% or 0.2.
No, this observation is not misclassified.