CS 584-04: Machine Learning

Spring 2019 Assignment 1 By Chetan Gupta A20378854

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?

Sol: Bin Width Calculation = 0.38461538

Code :-

bin_fd = np.histogram_bin_edges(trainData, bins='fd', range=(26.30, 35.40))

print(bin_fd)

```
In [50]: bin_fd = np.histogram_bin_edges(trainData, bins='fd', range=(26.30, 35.40))
         print(bin_fd)

[26.3        26.69565217 27.09130435 27.48695652 27.8826087  28.27826087
 28.67391304 29.06956522 29.46521739 29.86086957 30.25652174 30.65217391
 31.04782609 31.44347826 31.83913043 32.23478261 32.63043478 33.02608696
 33.42173913 33.8173913  34.21304348 34.60869565 35.00434783 35.4       ]
```

Bin Width Calculation = 0.38461538

```
In [51]: np.diff(bin_fd)

Out[51]: array([0.39565217, 0.39565217, 0.39565217, 0.39565217, 0.39565217,
                0.39565217, 0.39565217, 0.39565217, 0.39565217, 0.39565217,
                0.39565217, 0.39565217, 0.39565217, 0.39565217, 0.39565217,
                0.39565217, 0.39565217, 0.39565217, 0.39565217, 0.39565217,
                0.39565217, 0.39565217, 0.39565217])
```

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

Solu:- `min 26.300000`

`max 35.400000`

```
In :  # Put the descriptive statistics into another dataframe
      trainData_descriptive = trainData.describe()
      print(trainData_descriptive)
      print(trainData.describe())
```

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?

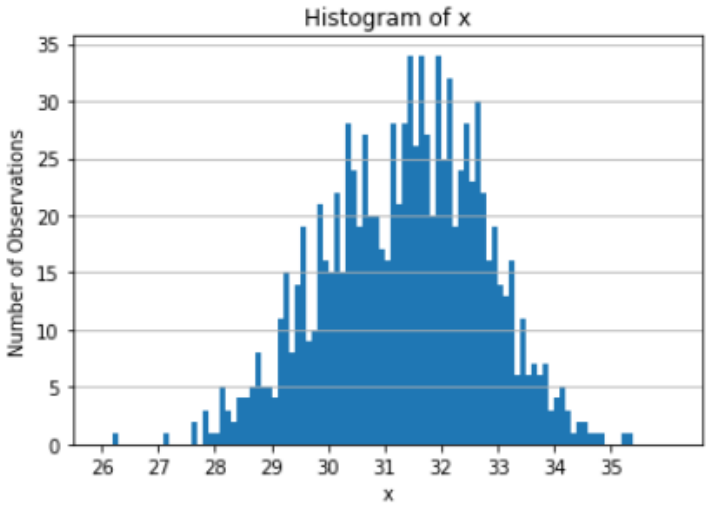
Solution:- a = 26

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.

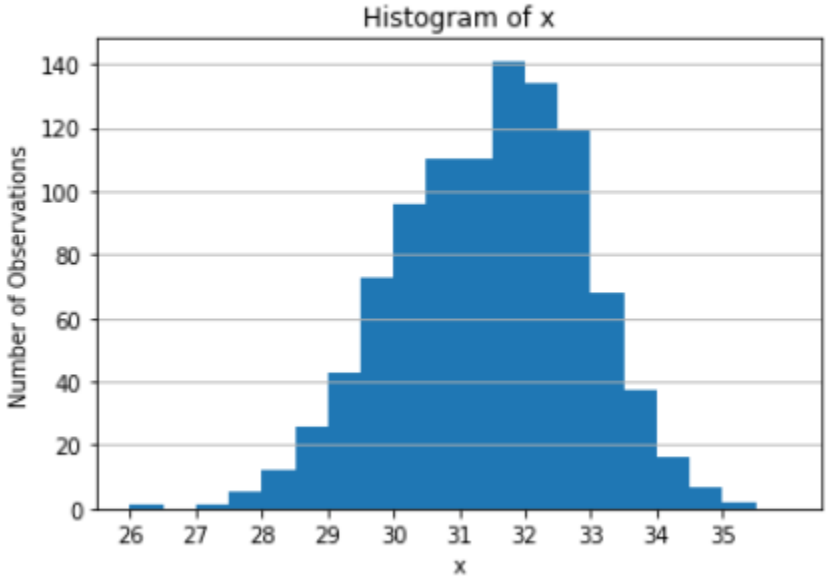
In [99]:
```
# Visualize the histogram of the x variable
for i in bin_list:
    trainData.hist(column='x', bins=np.arange(a, b + i, i))
    plt.title("Histogram of x")
    plt.xlabel("x")
    plt.ylabel("Number of Observations")
    plt.xticks(numpy.arange(26,36,step=1))
    plt.grid(axis="x")
    print('For h = ', i)
    plt.show()
```

```
For h =  0.1
```



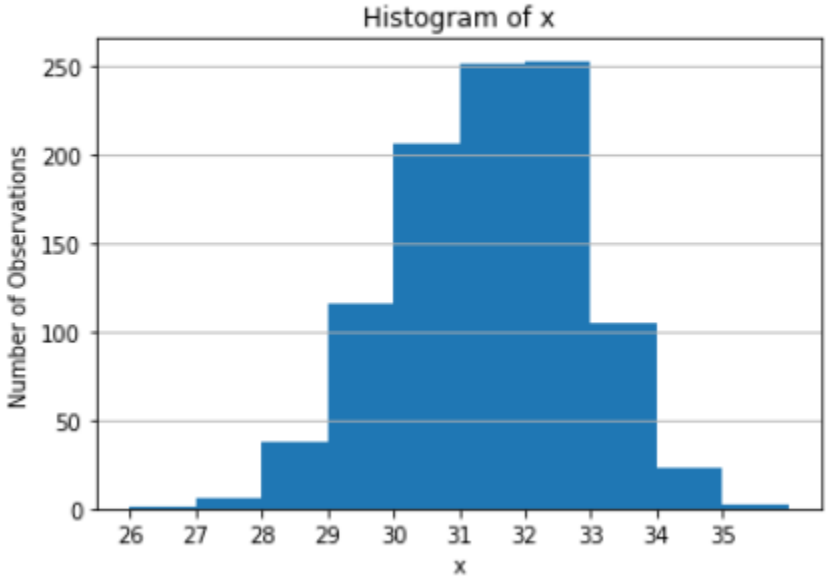
e) **(5 points) Use h = 0.5, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools**.

```
For h =  0.5
```



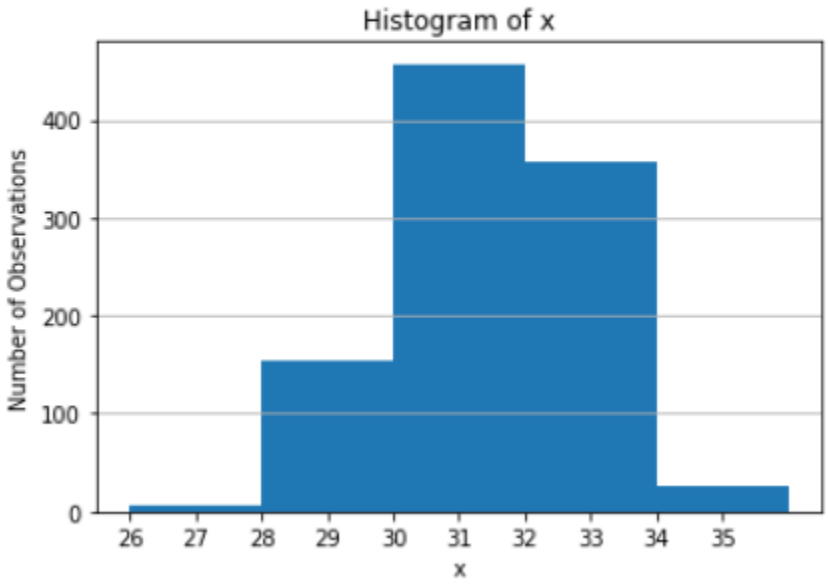
f) (5 points) Use h = 1, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

```
For h =  1
```



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.

For h = 2



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.

A Histogram provides a visual representation so you can see where most of the measurements are located and how spread out they are. A histogram is a plot that lets you discover, and show, the underlying frequency distribution (shape) of a set of continuous data. This allows the inspection of the data for its underlying distribution (e.g., normal distribution), outliers, skewness, etc. In a histogram, it is the area of the bar that indicates the frequency of occurrences for each bin. This means that the height of the bar does not necessarily indicate how many occurrences of scores there were within each individual bin. It is the product of height multiplied by the width of the bin that indicates the frequency of occurrences within that bin. One of the reasons that the height of the bars is often incorrectly assessed as indicating frequency and not the area of the bar is due to the fact that a lot of histograms often have equally spaced bars (bins), and under these circumstances, the height of the bin does reflect the frequency. So as per my analysis, histogram with h = 0.5 is best from all 4 histogram

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?**

min	26.3
25%	30.4
50%	31.5
75%	32.4
max	35.4
iqr	2
upper_whisker	35.4
lower_whisker	27.7

06]:
```
# calculate a 5-number summary
# calculate quartiles
quartiles = percentile(trainData, [25, 50, 75])
print(quartiles)
```

```
[30.4 31.5 32.4]
```

98]:
```
# calculate min/max
data_min, data_max = trainData.min(), trainData.max()
print(data_min[0], data_max[0])
```

```
26.3 35.4
```

```
def iqras(trainData):
    median = np.median(trainData)
    upper_quartile = np.percentile(trainData, 75)
    lower_quartile = np.percentile(trainData, 25)

    iqr = upper_quartile - lower_quartile
    upper_whisker = trainData[trainData<=upper_quartile+1.5*iqr].max()
    lower_whisker = trainData[trainData>=lower_quartile-1.5*iqr].min()
    return (iqr, upper_whisker[0], lower_whisker[0])
```

```
print(iqras(trainData))
```

```
(2.0, 35.4, 27.7)
```

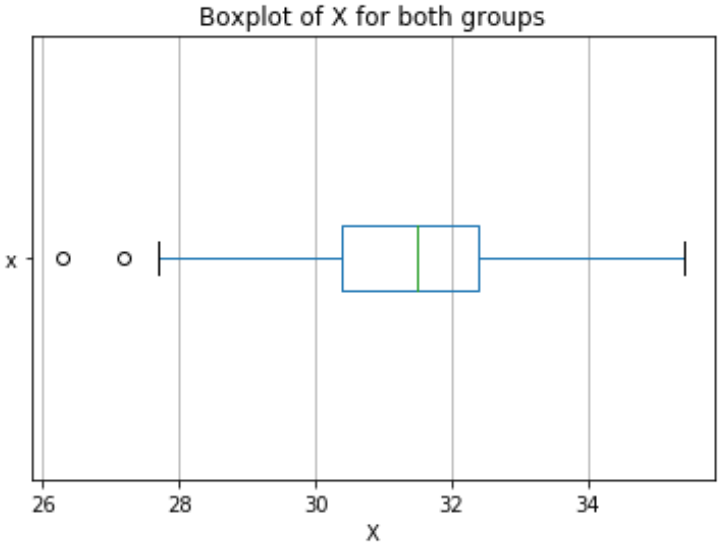
b) (**5 points) What are the five-number summary of x for each category of the group? What are the values of the 1.5 IQR whiskers for each category of the group?**
for 0 Group

min	26.3
25%	29.4
0.5	30
0.75	30.6
max	32.2
iqr	1.2
upper_whisker	32.2
lower_whisker	27.7

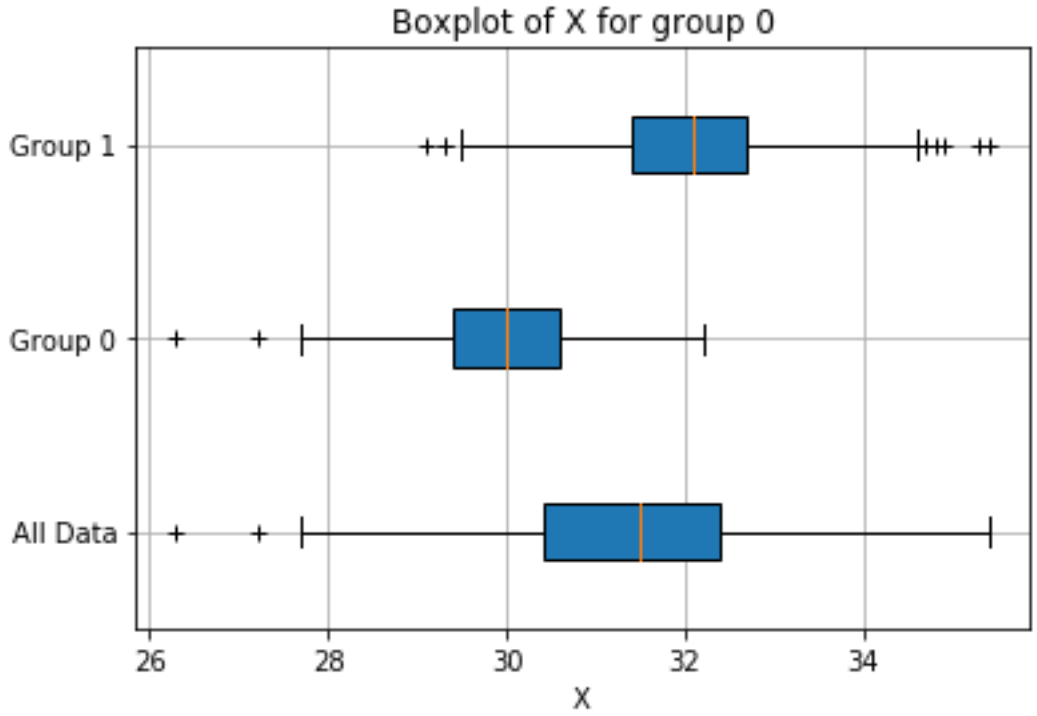
For 1 Group

min	29.1
25%	31.4
0.5	32.1
0.75	32.7
max	35.4
iqr	1.3
upper_whisker	34.6
lower_whisker	29.5

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?**
Yes it has displayed the 1.5 IQR 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.



Yes there are outliers, and all the outliers are marked as "+" signs in the plot.

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.

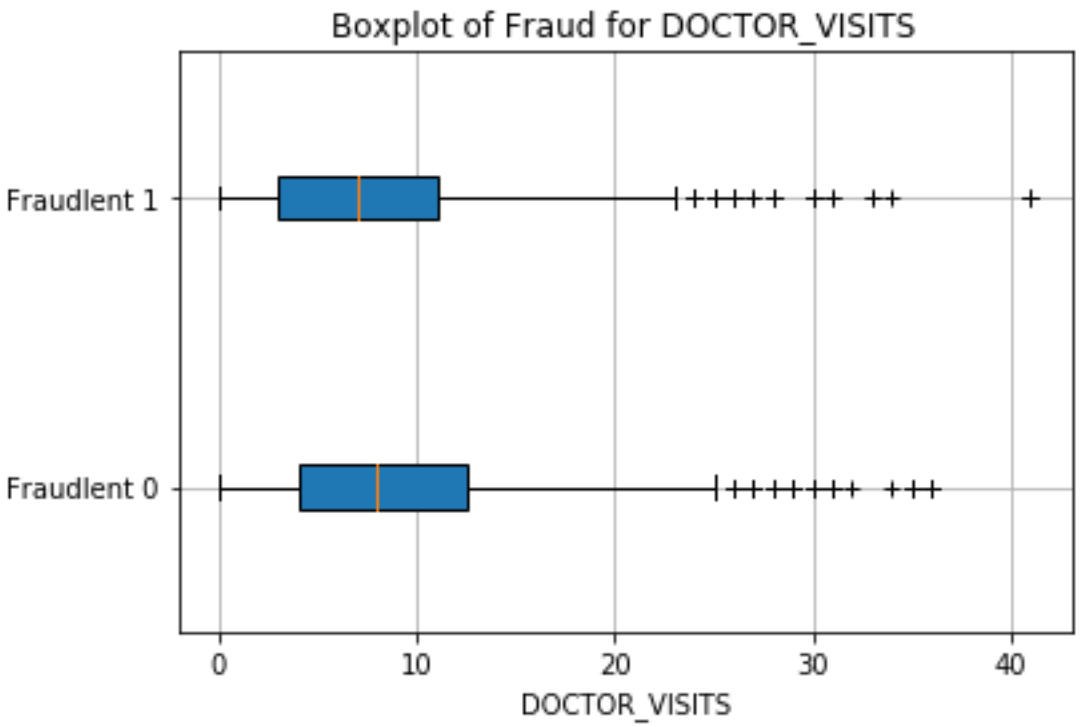
1. **TOTAL_SPEND: Total amount of claims in dollars**
2. **DOCTOR_VISITS: Number of visits to a doctor**
3. **NUM_CLAIMS: Number of claims made recently**
4. **MEMBER_DURATION: Membership duration in number of months**
5. **OPTOM_PRESC: Number of optical examinations**
6. **NUM_MEMBERS: Number of members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

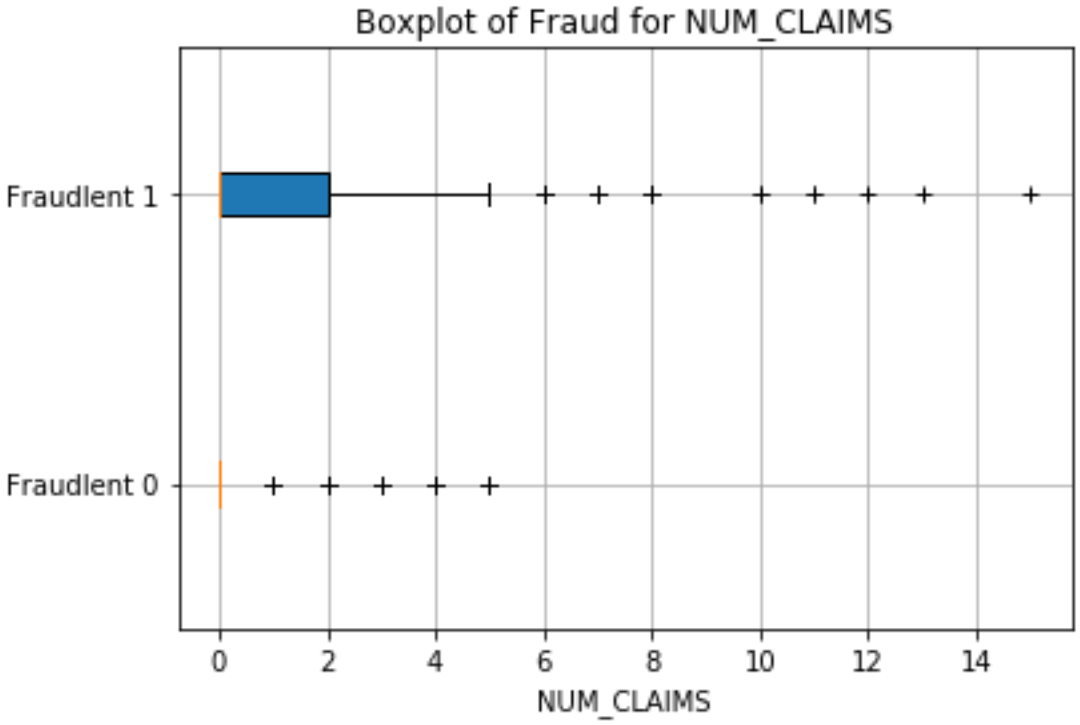
You are asked to use the Nearest Neighbors algorithm to predict the likelihood of fraud.

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

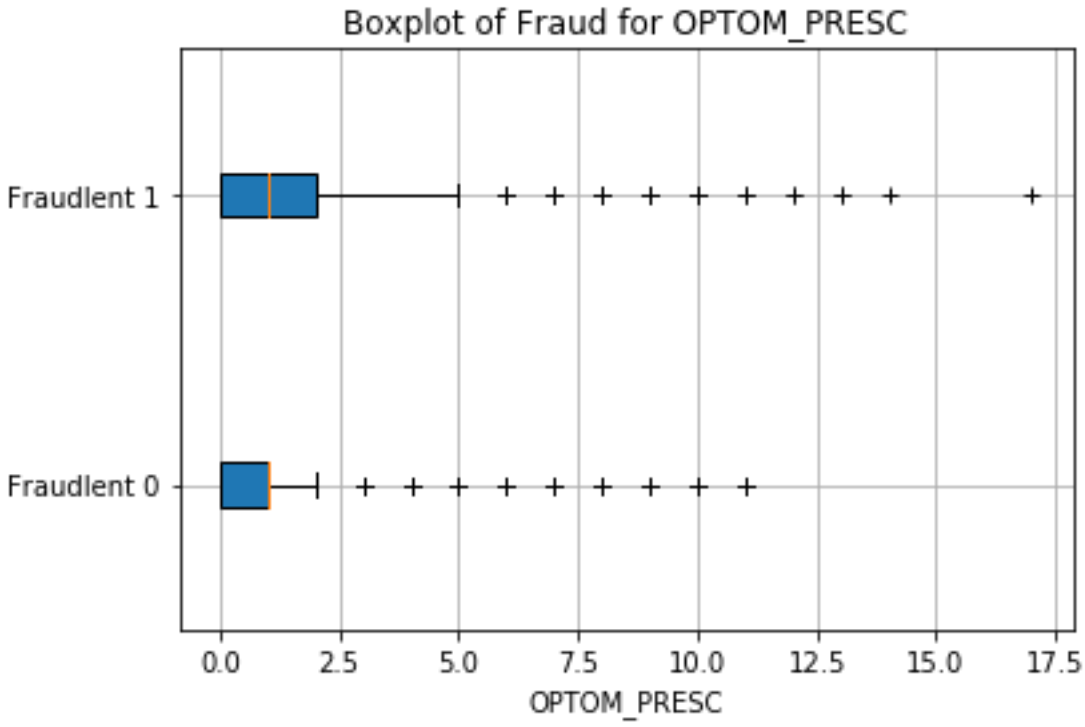
```
percent of investigations are found to be fraudulent
19.949664429530202 %
```

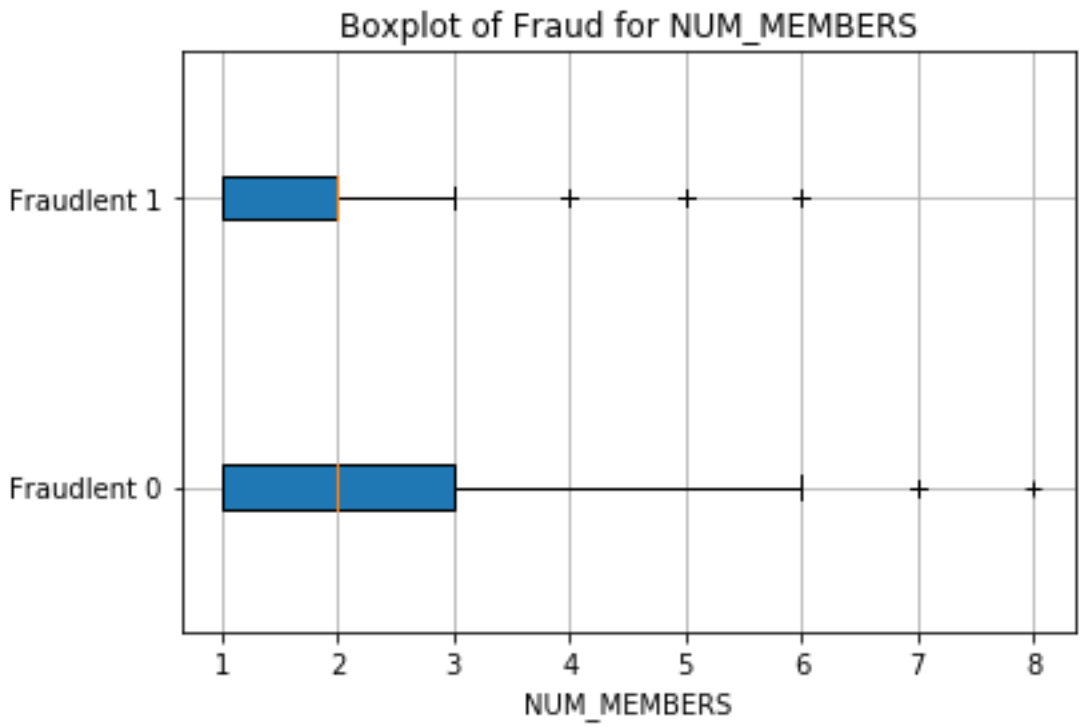
b) **(5 points) Use the BOXPLOT function to produce horizontal box-plots. For each interval variable, one box-plot for the fraudulent observations, and another box-plot for the non-fraudulent observations. These two box-plots must appear in the same graph for each interval variable.**


Boxplot of Fraud for DOCTOR_VISITS
Fraudlent 1
Fraudlent 0
0
10
20
30
40
DOCTOR_VISITS


Boxplot of Fraud for NUM_CLAIMS
Fraudlent 1
Fraudlent 0
0
2
4
6
8
10
12
14
NUM_CLAIMS


Boxplot of Fraud for MEMBER_DURATION
Fraudlent 1
Fraudlent 0
0
100
200
300
400
500
600
X


Boxplot of Fraud for OPTOM_PRESC
Fraudlent 1
Fraudlent 0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
OPTOM_PRESC



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?** `(5960, 6)`

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

Orthonormalize Matrix and Identity Matrix check

```
4]: orthx = LA2.orth(mat)
    print("The orthonormalize x = \n", orthx)
    orthx.shape
```

```
The orthonormalize x =
 [[-6.56324665e-04  9.39352141e-03  1.39590283e-02 -6.64664861e-03
   1.02081629e-02 -5.96859502e-03]
 [-7.75702220e-04  1.22658834e-02  5.16174400e-03  8.51930607e-04
   5.01932025e-03  2.09672310e-02]
 [-8.95075830e-04  1.50348109e-02 -1.71350853e-03 -7.38335310e-03
   1.97528525e-02 -7.64597676e-03]
 ...
 [-5.31896971e-02 -4.74021952e-02 -7.13245766e-03  2.75078514e-02
  -1.62580211e-02  7.18408819e-05]
 [-5.35474776e-02 -4.76625006e-02 -9.17125411e-03  2.76213381e-02
  -1.62154130e-02  1.80147801e-04]
 [-5.36071324e-02 -4.70861917e-02 -7.81347172e-03  2.93391341e-02
  -2.73884697e-02  2.21157680e-03]]
```

```
4]: (5960, 6)
```

```
5]: check = orthx.transpose().dot(orthx)
    print("Also Expect an Identity Matrix = \n", check)
```

```
Also Expect an Identity Matrix =
 [[ 1.00000000e+00 -1.11022302e-16  9.67108338e-17 -7.63278329e-17
   1.99493200e-17 -7.91467586e-18]
 [-1.11022302e-16  1.00000000e+00  1.83447008e-16  2.25514052e-17
  -1.38777878e-17 -3.03576608e-18]
 [ 9.67108338e-17  1.83447008e-16  1.00000000e+00 -6.67868538e-17
  -7.91467586e-18  2.55465137e-17]
 [-7.63278329e-17  2.25514052e-17 -6.67868538e-17  1.00000000e+00
  -9.10729825e-17  1.63660318e-16]
 [ 1.99493200e-17 -1.38777878e-17 -7.91467586e-18 -9.10729825e-17
   1.00000000e+00  3.25748543e-16]
 [-7.91467586e-18 -3.03576608e-18  2.55465137e-17  1.63660318e-16
   3.25748543e-16  1.00000000e+00]]
```

In [60]:
```
# Here is the transformation matrix
transf = evecs * LA.inv(np.sqrt(np.diagflat(evals)));
print("Transformation Matrix = \n", transf)
```

```
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]]
```

In [61]:
```
# Here is the transformed X
transf_x = mat * transf;
print("The Transformed x = \n", transf_x)
```

```
The Transformed x =
 [[ 5.96859502e-03  1.02081629e-02 -6.64664861e-03  1.39590283e-02
   9.39352141e-03  6.56324665e-04]
 [-2.09672310e-02  5.01932025e-03  8.51930607e-04  5.16174400e-03
   1.22658834e-02  7.75702220e-04]
 [ 7.64597676e-03  1.97528525e-02 -7.38335310e-03 -1.71350853e-03
   1.50348109e-02  8.95075830e-04]
 ...
 [-7.18408819e-05 -1.62580211e-02  2.75078514e-02 -7.13245766e-03
  -4.74021952e-02  5.31896971e-02]
 [-1.80147801e-04 -1.62154130e-02  2.76213381e-02 -9.17125411e-03
  -4.76625006e-02  5.35474776e-02]
 [-2.21157680e-03 -2.73884697e-02  2.93391341e-02 -7.81347172e-03
  -4.70861917e-02  5.36071324e-02]]
```

[62]:
```
# Check columns of transformed X
xtx = transf_x.transpose() * transf_x;
print("Expect an Identity Matrix = \n", xtx)
```

```
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 <u>five</u> 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.**
0.8778523489932886

ii. **(5 points) Explain the meaning of the score function return value.**
Score function basically returns the mean accuracy on the given test data and labels. In general, different models have score methods that return different metrics. This is to allow classifiers to specify what scoring metric they think is most appropriate for them (thus, for example, a least-squares regression classifier would have a score method that returns something like the sum of squared errors).

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*.**
Input Variable name:- 'TOTAL_SPEND', 'DOCTOR_VISITS', 'NUM_CLAIMS', 'MEMBER_DURATION', 'OPTOM_PRESC' , 'NUM_MEMBERS'
Target Values:- 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]]
Neighbors through both methods as professor illustrated in class
Case 1

CASE_ID	FRAUD	TOTAL_SPEND	DOCTOR_VISITS	NUM_CLAIMS	MEMBER_DURATION	OPTOM_PRESC	NUM_MEMBERS
588	1	7500	6	4	345	1	1
2897	1	16000	3	0	190	0	3
1199	1	10000	15	2	109	3	1
1246	0	10200	1	0	105	1	1
886	0	8900	18	0	280	0	1

Case 2

CASE_ID	FRAUD	TOTAL_SPEND	DOCTOR_VISITS	NUM_CLAIMS	MEMBER_DURATION	OPTOM_PRESC	NUM_MEMBERS
588	1	7500	6	4	345	1	1
577	1	7500	2	0	147	0	3
582	1	7500	18	2	76	1	2
573	1	7500	21	0	159	1	4
575	1	7500	6	0	147	0	3

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?

```
: focal = [[7500, 15, 3, 127, 2, 2]]

: nbrs.predict(focal*transf)

: array([1], dtype=int64)

: nbrs.predict_proba(focal)

: array([[0.8, 0.2]])
```

It is classified as Flatulent and as per my analysis this is not misclassified because when saw the neighbors, in the both the cases majoring goes with the fraudulent, that in case 1 60% cases are fraudulent and in case 2 100% are fraudulent.