# CS 584: Machine Learning

Spring 2020 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?

Ans:

**Input**:

```
Q1,Q3= np.percentile(dataset.x, [25 ,75])
IQR=Q3-Q1
N=len(dataset)
h=2*IQR*(N**(-1/3))
print("The recommended bin-width for the histogram of x by using Izenman (1991) method will be given by:",h,"=",round(h,2))
```

**Output:**

```
The recommended bin-width for the histogram of x by using Izenman (1991) method will be given by
0.3998667554864774 = 0.4
```

According to Izenman (1991) method, bin-width can be calculated by:

- ⇨ $\boldsymbol{h = 2(IQR)N^{-1/3}}$
- ⇨ $h$ = 2*2*$(1001)^{-1/3}$
  = 0.3998667554864774
  ≅ **0.4** (Rounded to two decimal places)

---

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

Ans:

**Input:**

```
minimum_x,maximum_x=min(dataset.x),max(dataset.x)
print("Minimum value of field 'x' is: ",minimum_x)
print("Maximum value of field 'x' is: ",maximum_x)
```

**Output:**

```
The recommended bin-width for the histogram of x by using Izenman (1991) method will be given by:
0.3998667554864774 = 0.4
Minimum value of field 'x' is:  26.3
Maximum value of field 'x' is:  35.4
```

I have used **min()** and **max()** function to calculate the minimum and maximum values of the field x. Which are as follows:

- ⇨ Minimum value of the field x = 26.3
- ⇨ Maximum value of the field 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?

Ans:

According to question:

a= largest integer less than the minimum value of the field x

b= smallest integer greater than the maximum value of the field x

**Input:**

```
a=math.floor(minimum_x)
b=math.ceil(maximum_x)
print("The largest integer less than the minimum value of the field x will be: ",a)
print("The smallest integer greater than the maximum value of the field x will be: ",b)
```

**Output:**

```
The largest integer less than the minimum value of the field x will be:  26
The smallest integer greater than the maximum value of the field x will be:  36
```

I have used math.floor() and math.ceil() functions to calculate the values of a and b, Which are as follows:

- ⇨ a= 26
- ⇨ b= 36

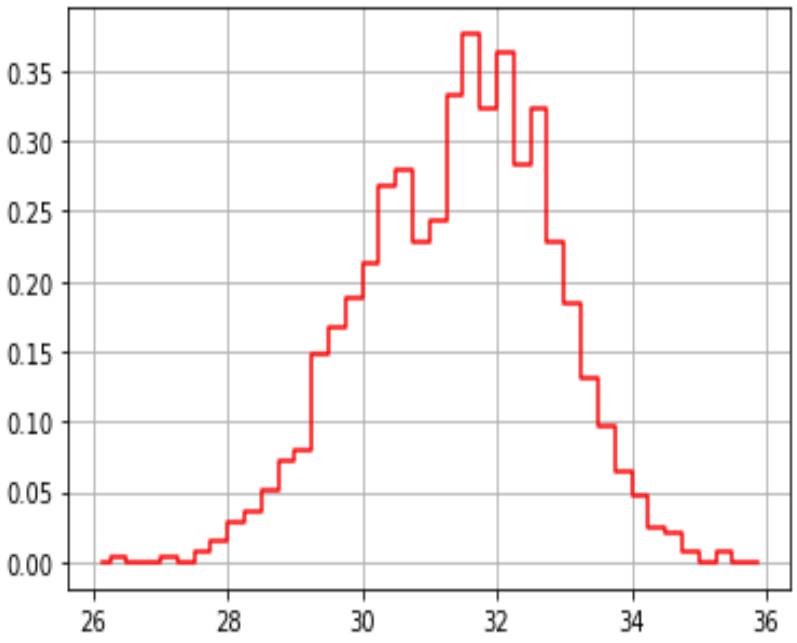
---

d) (5 points) Use h = 0.25, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

Ans:

I have used **step()** function to plot the histogram.

For **h=0.25**, there are total **40** results for density estimator whose coordinates are calculated as follows:




Histogram for h=0.25

| | mid_points | density |
|---|---|---|
| 0 | 26.125 | 0.000000 |
| 1 | 26.375 | 0.003996 |
| 2 | 26.625 | 0.000000 |
| 3 | 26.875 | 0.000000 |
| 4 | 27.125 | 0.003996 |
| 5 | 27.375 | 0.000000 |
| 6 | 27.625 | 0.007992 |
| 7 | 27.875 | 0.015984 |
| 8 | 28.125 | 0.027972 |
| 9 | 28.375 | 0.035964 |
| 10 | 28.625 | 0.051948 |
| 11 | 28.875 | 0.071928 |
| 12 | 29.125 | 0.079920 |
| 13 | 29.375 | 0.147852 |
| 14 | 29.625 | 0.167832 |
| 15 | 29.875 | 0.187812 |
| 16 | 30.125 | 0.211788 |
| 17 | 30.375 | 0.267732 |
| 18 | 30.625 | 0.279720 |
| 19 | 30.875 | 0.227772 |
| 20 | 31.125 | 0.243756 |
| 21 | 31.375 | 0.331668 |
| 22 | 31.625 | 0.375624 |
| 23 | 31.875 | 0.323676 |
| 24 | 32.125 | 0.363636 |
| 25 | 32.375 | 0.283716 |
| 26 | 32.625 | 0.323676 |
| 27 | 32.875 | 0.227772 |
| 28 | 33.125 | 0.183816 |

| | | |
|---|---|---|
| 29 | 33.375 | 0.131868 |
| 30 | 33.625 | 0.095904 |
| 31 | 33.875 | 0.063936 |
| 32 | 34.125 | 0.047952 |
| 33 | 34.375 | 0.023976 |
| 34 | 34.625 | 0.019980 |
| 35 | 34.875 | 0.007992 |
| 36 | 35.125 | 0.000000 |
| 37 | 35.375 | 0.007992 |
| 38 | 35.625 | 0.000000 |
| 39 | 35.875 | 0.000000 |

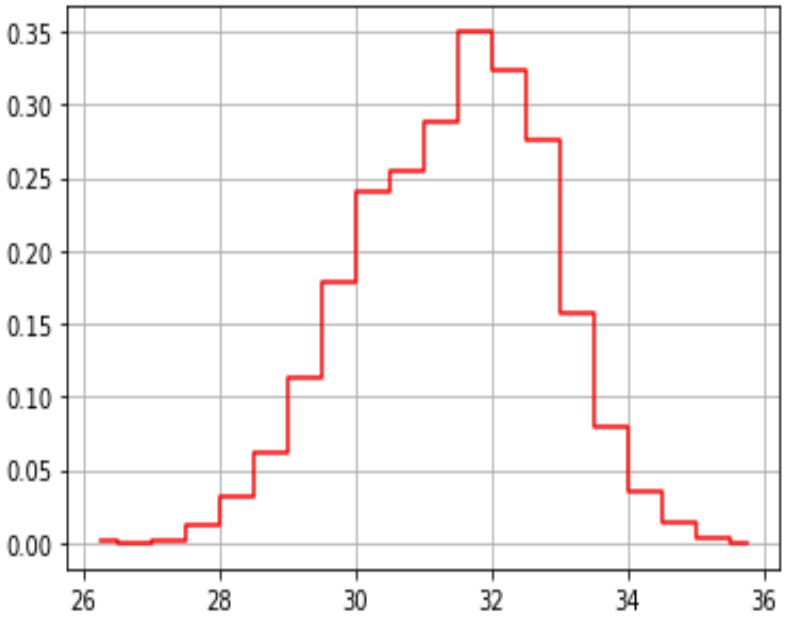
---

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.

Ans:

For **h=0.5**, there are total **20** results for density estimator whose coordinates are calculated as follows:

| | mid_points | density |
|---|---|---|
| 0 | 26.25 | 0.001998 |
| 1 | 26.75 | 0.000000 |
| 2 | 27.25 | 0.001998 |
| 3 | 27.75 | 0.011988 |
| 4 | 28.25 | 0.031968 |
| 5 | 28.75 | 0.061938 |
| 6 | 29.25 | 0.113886 |
| 7 | 29.75 | 0.177822 |
| 8 | 30.25 | 0.239760 |
| 9 | 30.75 | 0.253746 |
| 10 | 31.25 | 0.287712 |
| 11 | 31.75 | 0.349650 |
| 12 | 32.25 | 0.323676 |
| 13 | 32.75 | 0.275724 |
| 14 | 33.25 | 0.157842 |
| 15 | 33.75 | 0.079920 |
| 16 | 34.25 | 0.035964 |
| 17 | 34.75 | 0.013986 |
| 18 | 35.25 | 0.003996 |
| 19 | 35.75 | 0.000000 |



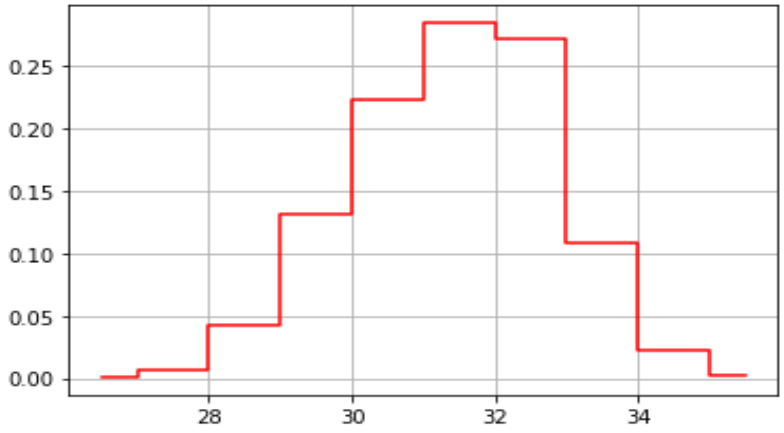

Histogram for h=0.5

---

*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.*

Ans:

For **h=1**, there are total **10** results for density estimator whose coordinates are calculated as follows:




Histogram for h= 1

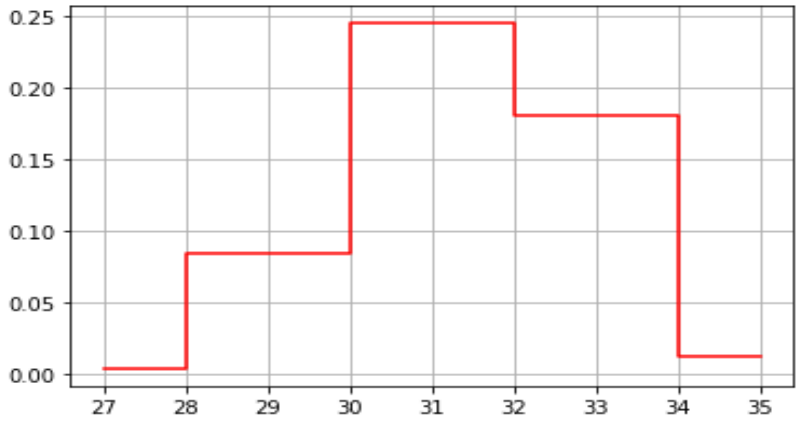
| | mid_points | density |
|---|---|---|
| 0 | 26.5 | 0.000999 |
| 1 | 27.5 | 0.006993 |
| 2 | 28.5 | 0.042957 |
| 3 | 29.5 | 0.131868 |
| 4 | 30.5 | 0.222777 |
| 5 | 31.5 | 0.284715 |
| 6 | 32.5 | 0.271728 |
| 7 | 33.5 | 0.107892 |
| 8 | 34.5 | 0.022977 |
| 9 | 35.5 | 0.001998 |

---

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

Ans:

For **h=2**, there are total **5** results for density estimator whose coordinates are calculated as follows:




Histogram for h= 2

| | mid_points | density |
|---|---|---|
| 0 | 27.0 | 0.003996 |
| 1 | 29.0 | 0.084915 |
| 2 | 31.0 | 0.245255 |
| 3 | 33.0 | 0.180320 |
| 4 | 35.0 | 0.012488 |

---

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.

Ans:

In my opinion, the histogram with h=0.5 is the best among the four in terms of shape and spread of distribution of field 'x'. This is because:

- ⇨ First, the bin width calculated using Izenman(1991) method comes out to be approximately 0.4 which is closest to the bin width of value 0.5.
- ⇨ From the visualization we can see that for h=0.25, density estimator curve tries to accommodate most of the data points which results in most detailed curve but it is not smooth. For h=1 or h=2 density estimator curve tries to accommodate only fewer data points, which makes It difficult to analyze how data points are spread where exactly the min and max are located, which results in less detailed curve.
- ⇨ However, for h=0.5 histogram shows the right amount of datapoints that are required to understand how the data are spread and where the maximum lies within data. Hence, we can say that for h=0.5, histogram provides best insights about the shape and the spread of the distribution of the field x

==========

## Question 2 (20 points)

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

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

Ans:

**Input:**

```
print(dataset.describe())
data=list(dataset.x)
Q1,Q3= np.percentile(data, [25 ,75])
IQR=Q3-Q1
l_whisker=Q1-1.5*IQR
u_whisker=Q3+1.5*IQR
print("Lower Whisker",l_whisker)
print("Upper Whisker",u_whisker)
```

**Output:**

```
                x
count  1001.000000
mean     31.414585
std       1.397672
min      26.300000
25%      30.400000
50%      31.500000
75%      32.400000
max      35.400000
Lower Whisker 27.4
Upper Whisker 35.4
```

I have used **describe()** function to compute the five-number summary of x and its results are as follows:

| | |
|---|---|
| **Min** (minimum value) | **26.300000** |
| **25%** (first quartile value) | **30.400000** |
| **50%** (median value) | **31.500000** |
| **75%** (third quartile value) | **32.400000** |
| **Max** (Maximum value) | **35.400000** |

The values of the 1.5 IQR whiskers can be calculated as follows:

- ⇨ Lower Whisker= **Q1-1.5*IQR** = 27.4
- ⇨ Upper Whisker= **Q3+1.5*IQR** = 35.4

---

b) (5 points) What is 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?

Ans:

**Input:**

```
new_dataset=dataset.groupby('group')
print(new_dataset.describe())
data_group_0=dataset[dataset.group == 0].x
data_group_1=dataset[dataset.group == 1].x
```

```
print("For Group 0:")
Q1_0,Q3_0= np.percentile(data_group_0,[25,75])
IQR = Q3_0-Q1_0
l_whisker_0=Q1_0-1.5*IQR
u_whisker_0=Q3_0+1.5*IQR
print("\tLower Whisker",l_whisker_0)
print("\tUpper Whisker",u_whisker_0)

print("For Group 1:")
Q1_1,Q3_1= np.percentile(data_group_1,[25,75])
IQR = Q3_1-Q1_1
l_whisker_1=Q1_1-1.5*IQR
u_whisker_1=Q3_1+1.5*IQR
print("\tLower Whisker",l_whisker_1)
print("\tUpper Whisker",u_whisker_1)
```

**Output:**

```
           i                                       ...      x
       count        mean         std  min     25%  ...   min   25%   50%   75%   max
group                                               ...
0      315.0  501.866667  287.813171  2.0  260.50  ...  26.3  29.4  30.0  30.6  32.2
1      686.0  499.142857  289.906257  0.0  241.25  ...  29.1  31.4  32.1  32.7  35.4

[2 rows x 16 columns]
For Group 0:
        Lower Whisker 27.599999999999994
        Upper Whisker 32.400000000000006
For Group 1:
        Lower Whisker 29.449999999999992
        Upper Whisker 34.650000000000006
```

I have used **describe()** function to compute the five-number summary of x for each category of the group and its results are as follows:

| Group 0 | | Group 1 | |
|---|---|---|---|
| **Min** | **26.3** | **Min** | **29.1** |
| **25%** | **29.4** | **25%** | **31.4** |
| **50%** | **30.0** | **50%** | **32.1** |
| **75%** | **30.6** | **75%** | **32.7** |
| **max** | **32.2** | **max** | **35.4** |

the values of the 1.5 IQR whiskers can be calculated as follows:

- For Group 0

⇨ Lowe Whisker= **Q1-1.5*IQR** = 27.59

⇨ Upper Whisker= **Q3+1.5*IQR** = 32.40

- For Group 1

⇨ Lowe Whisker= **Q1-1.5*IQR** = 29.44

⇨ Upper Whisker= **Q3+1.5*IQR** = 34.65

---

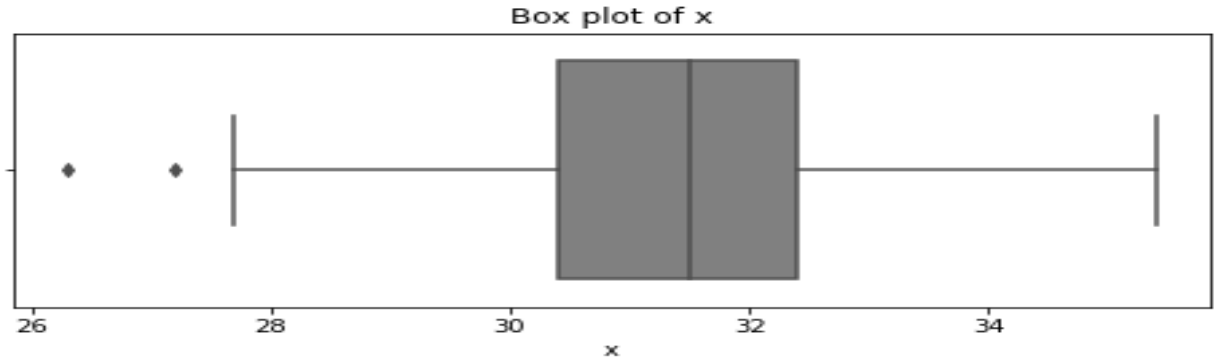
c) (5 points) Draw a boxplot of x (without the group) using the Python boxplot function. Can you tell if the Python's boxplot has displayed the 1.5 IQR whiskers correctly?

Ans:

**Input:**

```
plt.figure(figsize=(8,3))
sns.boxplot(x=dataset.x,color='gray')
plt.title("Box plot of x")
```

**Output:**




From the above box plot we can see that both the Whiskers are almost similar to what we have calculated. Hence, we can say that Python's boxplot 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 the group.

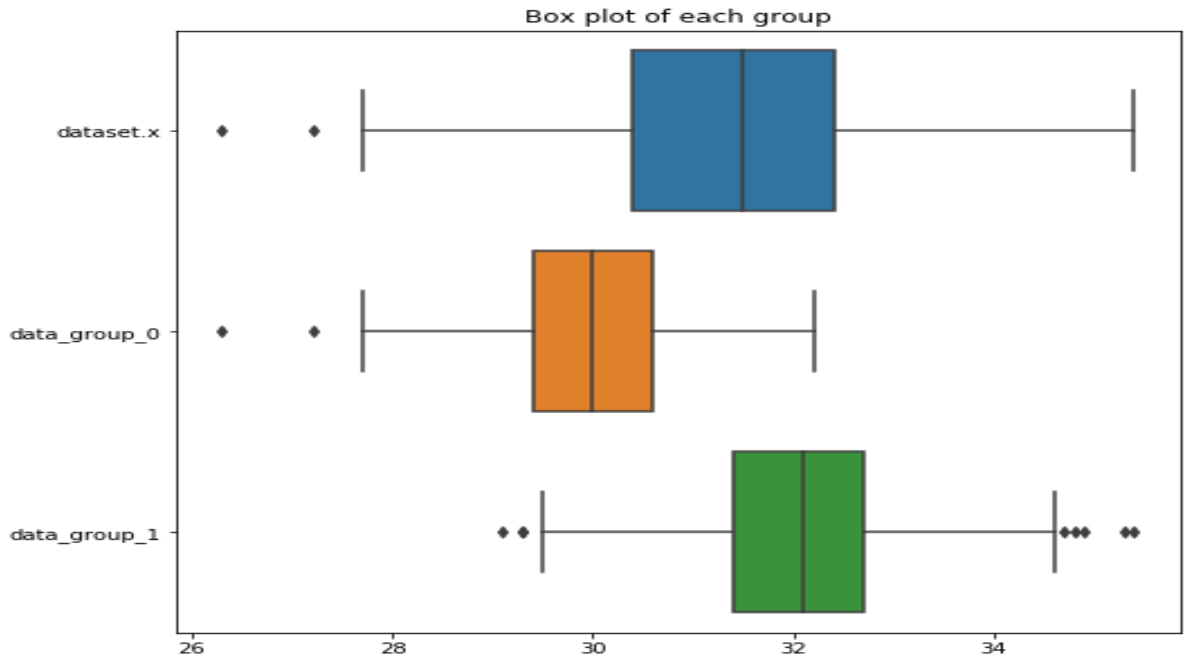
*Hint: Consider using the CONCAT function in the PANDA module to append observations*.

Ans:

**Input:**

```
new_data=pd.concat([dataset.x,data_group_0,data_group_1],axis=1, keys=['dataset.x','data_group_0','data_group_1'])
plt.figure(figsize=(8,8))
sns.boxplot(data=new_data,orient='h')
plt.title("Box plot of each group")
```

**Output:**



As mentioned in the question, I have used **concat()** function in the **Panda** module to append observations and plotted the graph using **boxplot()** function in the **seaborn** module.

**Outlier Identification:**

**Input:**

```
print("outliers of x for entire dataset are as follows:")
for pt in dataset.x:
    if pt < l_whisker or pt > u_whisker:
        print("\t",pt)

print("outliers of x for each group of data are as follows:")
print("\toutliers of x for group 0 are as follows:")
for pt in data_group_0:
    if pt < l_whisker_0 or pt>u_whisker_0:
        print("\t\t",pt)

print("\toutliers of x for group 1 are as follows:")
for pt in data_group_1:
    if pt < l_whisker_1 or pt>u_whisker_1:
        print("\t\t",pt)
```

**Output:**

```
outliers of x for entire dataset are as follows:
         27.2
         26.3
outliers of x for each group of data are as follows:
        outliers of x for group 0 are as follows:
                 27.2
                 26.3
        outliers of x for group 1 are as follows:
                 35.3
                 29.3
                 35.4
                 34.9
                 34.7
                 34.8
                 29.3
                 29.1
```

==========

## 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.

Ans:

Percent of investigations found to be fraudulent is given by:

$$= \frac{Number\ of\ fraudulent\ records}{Total\ number\ of\ records} * 100$$

$$= \frac{1189}{5960} * 100$$

= **19.9497 %**

**Input:**

```
print(dataset.head())
fraud = dataset[dataset.FRAUD == 1]
fraud_percentage=(len(fraud)/len(dataset))*100
print("Percent of investigations are found to be fraudulent is given by: ",round(fraud_percentage, 4))
```

**Output:**

```
Percent of investigations are found to be fraudulent is given by:  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.

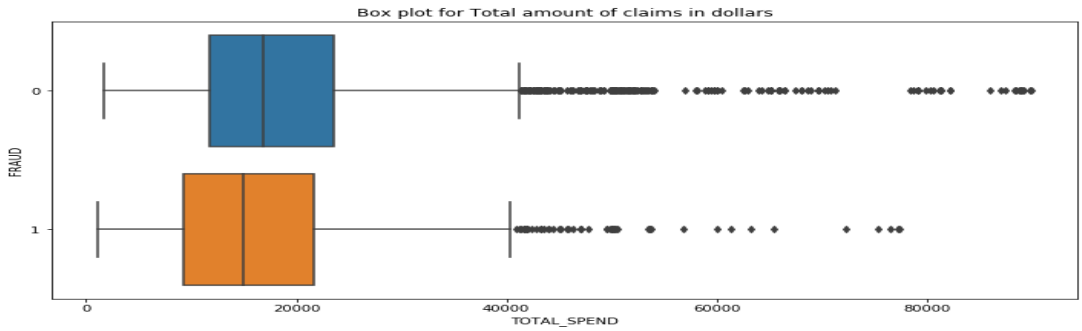
Ans:

⇨ Box plot for TOTAL_SPEND

**Input:**

```
plt.figure(figsize=(12,6))
sns.boxplot(x="TOTAL_SPEND", y="FRAUD", data=dataset,orient='h')
plt.title("Box plot for Total amount of claims in dollars")
```

**Output:**

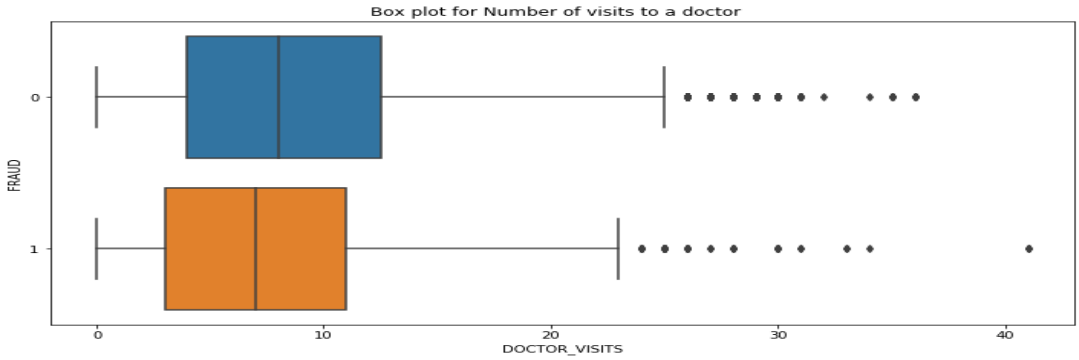


⇨ Box plot for DOCTOR_VISITS

**Input:**

```
plt.figure(figsize=(12,6))
sns.boxplot(x="DOCTOR_VISITS", y="FRAUD", data=dataset,orient='h')
plt.title("Box plot for Number of visits to a doctor  ")
```

**Output:**

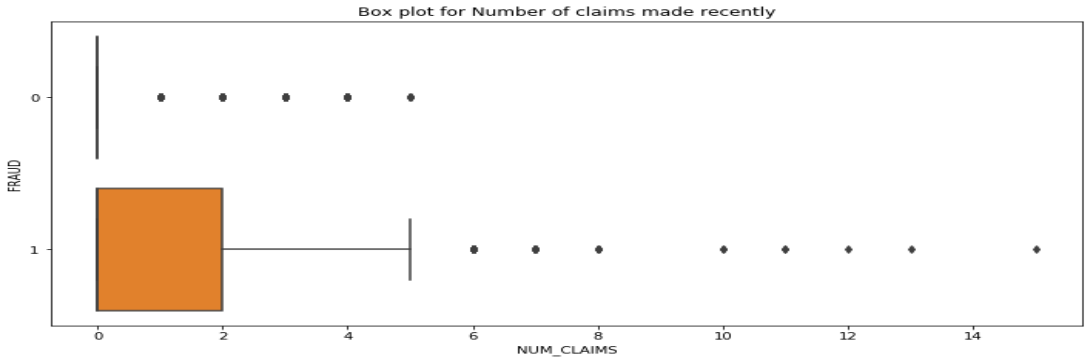



⇨ Box plot for NUM_CLAIMS

**Input:**

```
plt.figure(figsize=(12,6))
sns.boxplot(x="NUM_CLAIMS", y="FRAUD", data=dataset,orient='h')
plt.title("Box plot for Number of claims made recently")
```

**Output:**

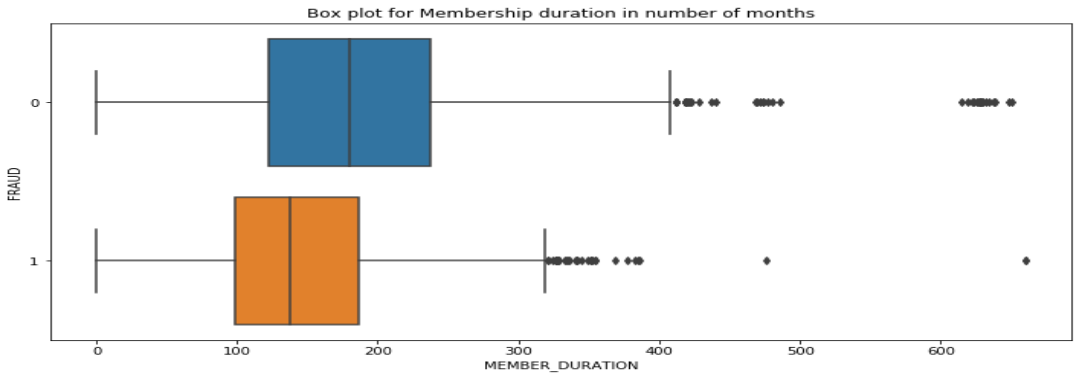



⇨ Box plot for MEMBER_DURATION

**Input:**

```
plt.figure(figsize=(12,6))
sns.boxplot(x="MEMBER_DURATION", y="FRAUD", data=dataset,orient='h')
plt.title("Box plot for Membership duration in number of months")
```

**Output:**

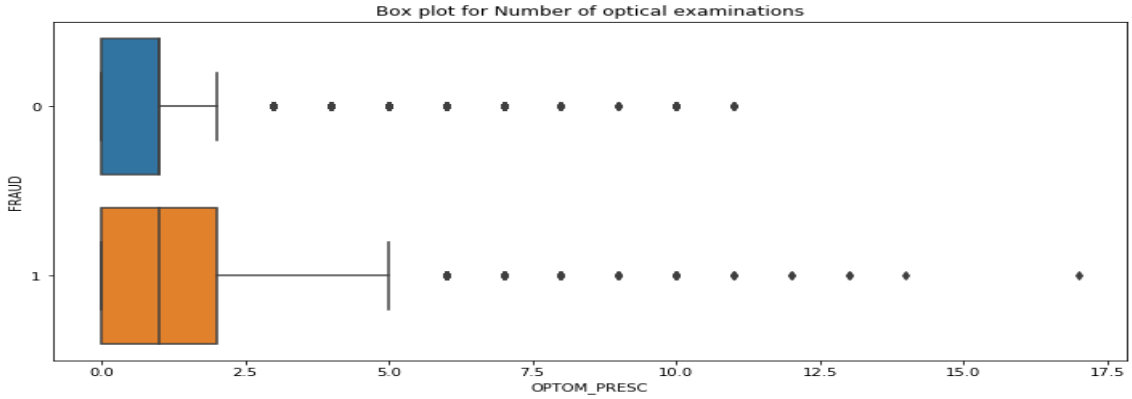


⇨ Box plot for OPTOM_PRESC

**Input:**

```
plt.figure(figsize=(12,6))
sns.boxplot(x="OPTOM_PRESC", y="FRAUD", data=dataset,orient='h')
plt.title("Box plot for Number of optical examinations")
```

**Output:**

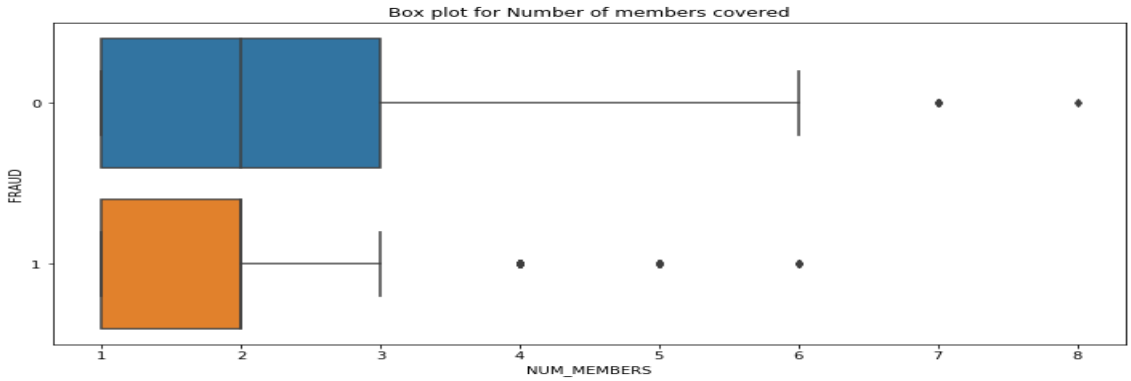



⇨ Box plot for NUM_MEMBERS

**Input:**

```
plt.figure(figsize=(12,6))
sns.boxplot(x="NUM_MEMBERS", y="FRAUD", data=dataset,orient='h')
plt.title("Box plot for Number of members covered")
```

**Output:**




---

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.

Ans:

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

**Output for eigenvalues are:**

```
Eigenvalues of x =
         [6.84728061e+03 8.38798104e+03 1.80639631e+04 3.15839942e+05
 8.44539131e+07 2.81233324e+12]
```

According to question, we should use only those dimensions whose corresponding eigenvalues are greater than one. After calculating eigenvalues of x, I found that all the eigen are greater than 1. So, I have used all the **6 dimensions**.

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

**Input:**

```
data=dataset.drop(['FRAUD','CASE_ID'],axis=1)
# Create a matrix x
x = np.matrix(data)
#Calculate the transpose of x
xtx = x.transpose() * x
print("\nt(x) * x = \n\t", xtx)
#Eigenvalue decomposition
evals, evecs = LA.eigh(xtx)
print("\nEigenvalues of x = \n\t", evals)
print("\nEigenvectors of x = \n\t", evecs)
# Here is the transformation matrix
transf = evecs * LA.inv(np.sqrt(np.diagflat(evals)));
print("\nTransformation Matrix = \n\t", transf)
# Here is the transformed X
transf_x = x * transf;
print("\nThe Transformed x = \n\t", transf_x)
# Check columns of transformed X
xtx = transf_x.transpose() * transf_x;
print("\nExpect an Identity Matrix = \n\t", xtx)
```

**Output:**

**Transformation Matrix:**

```
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]]
```

We can see that after multiplying **transformed matrix** with **transpose of transformed matrix**, we are getting identity matrix which is a proof that the resulting variables are actually orthonormal.

```
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.

Ans:

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

**Input:**

```
#Using KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=5 , algorithm = 'brute', metric = 'euclidean')
nbrs = neigh.fit(trainData, target)
# Score Calculation:
_score = nbrs.score(trainData, target)
print("Score is:",_score)
```

**Output:**

```
Score is: 0.8778523489932886
```

Score Function return value is **0.8778523489932886**

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

Score function return value represents the **Accuracy** of the model i.e., the fraction of observations that are correctly classified by the model.

---

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*.

Ans:

**Input:**

```
pd.set_option('display.expand_frame_repr', False)
focal = [[7500, 15, 3, 127, 2, 2]]
transf_focal = focal * transf;
myNeighbors_t = nbrs.kneighbors(transf_focal, return_distance = False)
print("My Neighbors = \n", myNeighbors_t)
print(dataset.iloc[list(myNeighbors_t[0])])
```

**Output:**

```
My Neighbors =
 [[ 588 2897 1199 1246  886]]
      CASE_ID  FRAUD  TOTAL_SPEND  DOCTOR_VISITS  NUM_CLAIMS  MEMBER_DURATION  OPTOM_PRESC  NUM_MEMBERS
588       589      1         7500             15           3              127            2            2
2897     2898      1        16000             18           3              146            3            2
1199     1200      1        10000             16           3              124            2            1
1246     1247      1        10200             13           3              119            2            3
886       887      1         8900             22           3              166            1            2
```

From the above results, we can see that the indices of five neighbors are: **588, 2897, 1199, 1246 and 886**. So, the neighbors will have CASE_ID: 589, 2898, 1200, 1247 and 887 and are shown in above tables.

---

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?

Ans:

**Input:**

```
class_prob = nbrs.predict_proba(transf_focal)
print("\nPredicted Probability is:",class_prob)
```

**Output:**

```
Predicted Probability is: [[0. 1.]]
```

Here, From the above results, we can see that the predicted probability of fraudulent (i.e., FRAUD = 1) is **100%,** which is greater than **19.9497%**. Hence the observation will be classified as fraudulent and we can say that it is not misclassified.

==========