CS 584-04: Machine Learning

Fall 2019 Assignment 1

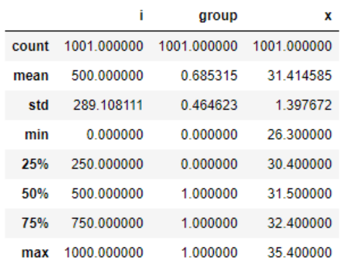
Question 1 (40 points)

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

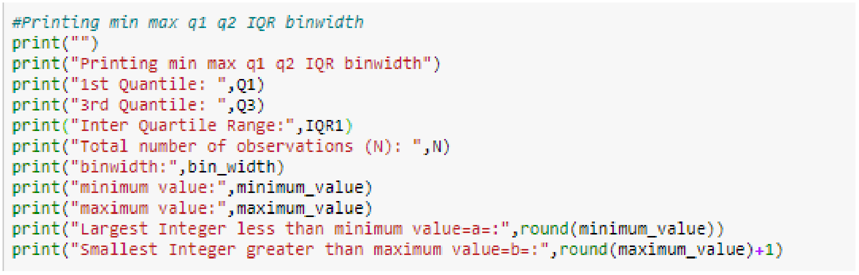
a) (5 points) According to Izenman (1991) method, what is the recommended bin-width for the histogram of x?

Ans:

1. We considered NormalSample.csv file read it as dataframe and using describe() method, we found following,

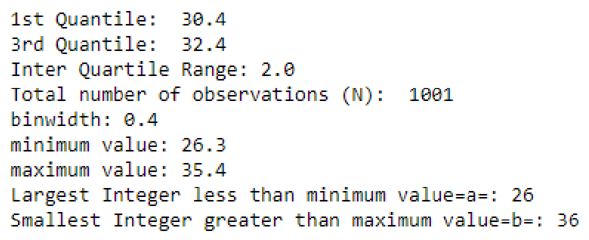


	i	group	x
count	1001.000000	1001.000000	1001.000000
mean	500.000000	0.685315	31.414585
std	289.108111	0.464623	1.397672
min	0.000000	0.000000	26.300000
25%	250.000000	0.000000	30.400000
50%	500.000000	1.000000	31.500000
75%	750.000000	1.000000	32.400000
max	1000.000000	1.000000	35.400000



```python
#Printing min max q1 q2 IQR binwidth
print("")
print("Printing min max q1 q2 IQR binwidth")
print("1st Quantile: ",Q1)
print("3rd Quantile: ",Q3)
print("Inter Quartile Range:",IQR1)
print("Total number of observations (N): ",N)
print("binwidth:",bin_width)
print("minimum value:",minimum_value)
print("maximum value:",maximum_value)
print("Largest Integer less than minimum value=a=:",round(minimum_value))
print("Smallest Integer greater than maximum value=b=:",round(maximum_value)+1)
```

o/p:



```
1st Quantile:  30.4
3rd Quantile:  32.4
Inter Quartile Range: 2.0
Total number of observations (N):  1001
binwidth: 0.4
minimum value: 26.3
maximum value: 35.4
Largest Integer less than minimum value=a=: 26
Smallest Integer greater than maximum value=b=: 36
```

min	Max	Q1	Q3	Q2
26.300000	**35.400000**	**30.400000**	**32.400000**	**31.500000**

2. According to Izenman(1991) method, we can calculate bin-width using following formula,

$$h = 2(IQR)N^{-1/3}$$

h = 2*2 *(1001)^-1/3

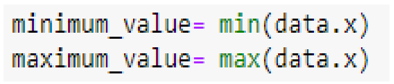
binwidth=h = 0.4 (rounded to the two decimal places)

----

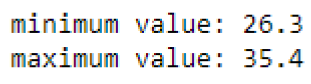
b) (5 points) What are the minimum and the maximum values of the field x?

Ans:

1. By using min() and max() function we can get, minimum and maximum values as follows,
   Minimum value=26.3
   Maximum value=35.4



```python
minimum_value= min(data.x)
maximum_value= max(data.x)
```



```
minimum value: 26.3
maximum value: 35.4
```

2. Also, using describe() function we can get these values.

----

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:

As per logic to calculate a and b following code was implemented, from which we got,

a=26 and b=36

```
minimum_value= min(data.x)
maximum_value= max(data.x)
a=round(minimum_value)
b=round(maximum_value+1)
print(a,b)
```

```
26 36
```

```
minimum value: 26.3
maximum value: 35.4
Largest Integer less than minimum value=a=: 26
Smallest Integer greater than maximum value=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.

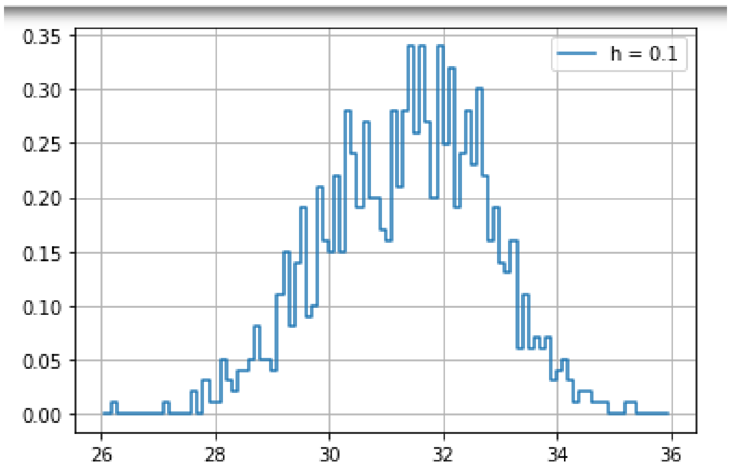
Ans:

All histograms are plotted using step() function.

List of coordinates of density estimator for h=0.1: number of bins=100

Histogram for h=0.1: there are 100 results of desity estimator so providing sceenshot for only first few results. For complete answer please refer code.

```
    midpoints  Probability_density
0       26.05              0.00000
1       26.15              0.00000
2       26.25              0.00999
3       26.35              0.00000
4       26.45              0.00000
5       26.55              0.00000
6       26.65              0.00000
7       26.75              0.00000
8       26.85              0.00000
9       26.95              0.00000
10      27.05              0.00000
11      27.15              0.00999
12      27.25              0.00000
13      27.35              0.00000
```



----

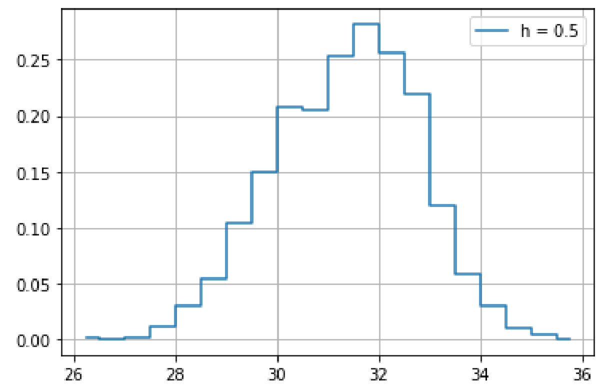
e) (5 points) Use h = 0.5, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

Ans:

List of coordinates of density estimator for h=0.5: number of bins=20

	midpoints	Probability_density
0	26.25	0.001998
1	26.75	0.000000
2	27.25	0.001998
3	27.75	0.011988
4	28.25	0.029970
5	28.75	0.053946
6	29.25	0.103896
7	29.75	0.149850
8	30.25	0.207792
9	30.75	0.205794
10	31.25	0.253746
11	31.75	0.281718
12	32.25	0.255744
13	32.75	0.219780
14	33.25	0.119880
15	33.75	0.057942
16	34.25	0.029970
17	34.75	0.009990
18	35.25	0.003996
19	35.75	0.000000

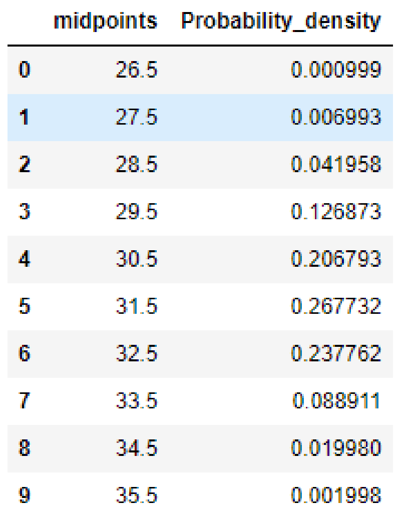
Histogram for h=0.5:



----

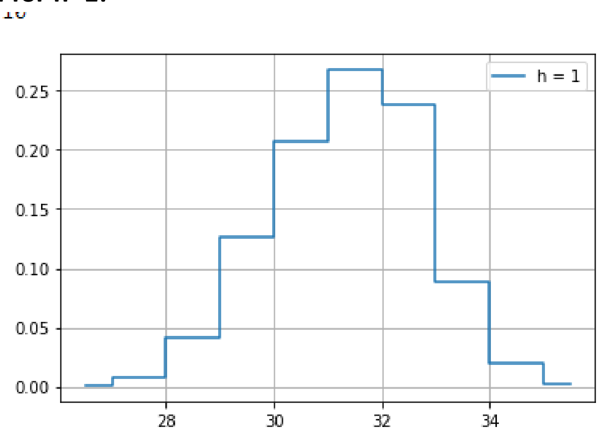
f) (5 points) Use h = 1, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

List of coordinates of density estimator for h=1: number of bins=10



	midpoints	Probability_density
0	26.5	0.000999
1	27.5	0.006993
2	28.5	0.041958
3	29.5	0.126873
4	30.5	0.206793
5	31.5	0.267732
6	32.5	0.237762
7	33.5	0.088911
8	34.5	0.019980
9	35.5	0.001998

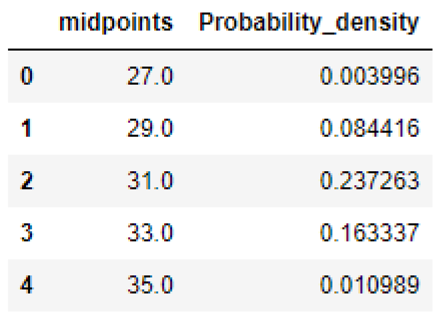
Histogram for h=1:



----

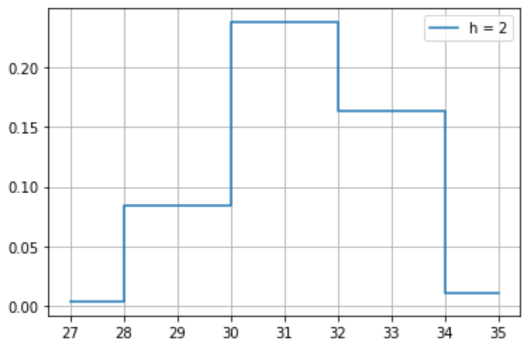
g) (5 points) Use h = 2, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

List of coordinates of density estimator for h=2: number of bins=5



	midpoints	Probability_density
0	27.0	0.003996
1	29.0	0.084416
2	31.0	0.237263
3	33.0	0.163337
4	35.0	0.010989

Histogram 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.
Ans:

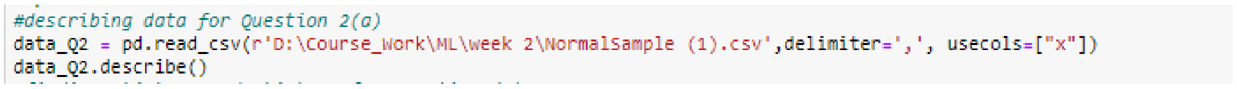
1. According to me, amongst all 4 box plots best plot in terms of shape and spread of distribution of field x is one with **h=0.5**.
2. Explanation:
   I. Primarily, the bin width we calculated by Izenman(1991) method is 0.4 which is closer to this plot's bin width.
   II. From visualization we can see the spread of graph is ideal as compared to other plots it is not wide or narrow.
   III. Also, the graph gives fair estimation of probability densities.
   IV. Besides, the histogram shows an approximately symmetric density with a single mode.
   V. Also, as we discussed in class, there is tradeoff between binwidth and details. Hence, plot of h=0.1 is not good as there are so many details given. In plots of h=1 and h=2 bin width is high there are less details.

Question 2 (20 points)

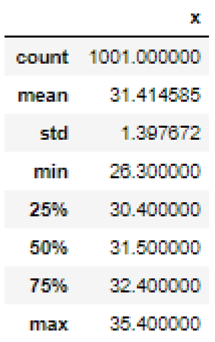
Use in the NormalSample.csv to generate box-plots for answering the following questions.

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

1. Using describe() function we can get five-number summary of x:



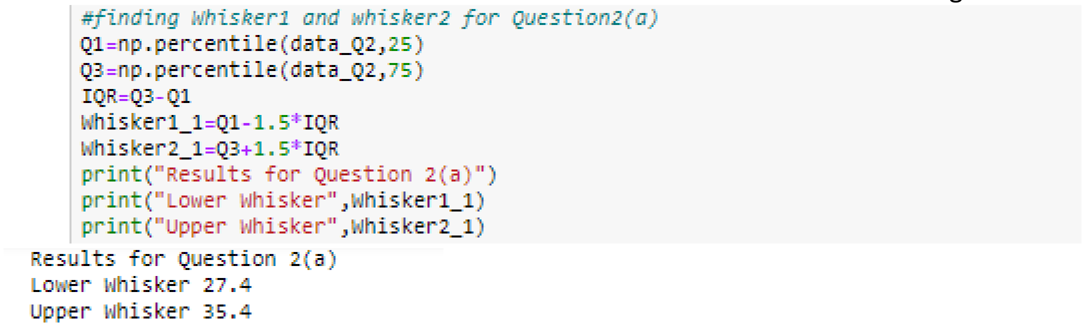
```
#describing data for Question 2(a)
data_Q2 = pd.read_csv(r'D:\Course_Work\ML\week 2\NormalSample (1).csv',delimiter=',', usecols=["x"])
data_Q2.describe()
```



```
                 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
```

Min	max	Q1	Q3	Q2
26.300000	**35.400000**	**30.400000**	**32.400000**	**31.500000**

2. For the values of 1.5 IQR Whiskers, we know that we can calculate it using following formulas, same logic is been implemented while coding
   I. Lower Whisker= Q1 -1.5*IQR
   II. Upper Whisker= Q3 +1.5*IQR
   Where, IQR = |Q3-Q1|



```
#finding Whisker1 and whisker2 for Question2(a)
Q1=np.percentile(data_Q2,25)
Q3=np.percentile(data_Q2,75)
IQR=Q3-Q1
Whisker1_1=Q1-1.5*IQR
Whisker2_1=Q3+1.5*IQR
print("Results for Question 2(a)")
print("Lower Whisker",Whisker1_1)
print("Upper Whisker",Whisker2_1)
```

```
Results for Question 2(a)
Lower Whisker 27.4
Upper Whisker 35.4
```

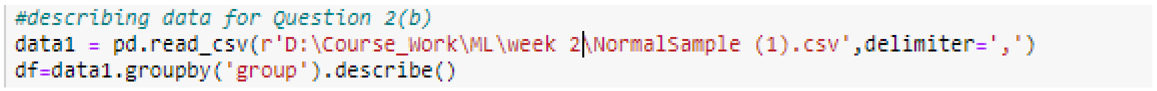
Lower Whisker	27.4
Upper Whisker	35.4

----

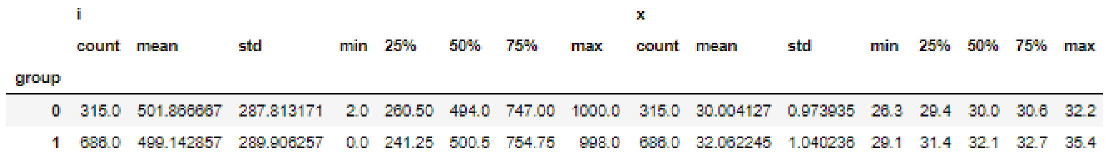
b) (5 points) What is the five-number summary of x for each category of the group? What are the values of the 1.5 IQR whiskers for each category of the group?

Ans:

1. Using describe() function we can get five-number summary of x for each group:



```
#describing data for Question 2(b)
data1 = pd.read_csv(r'D:\Course_Work\ML\week 2\NormalSample (1).csv',delimiter=',')
df=data1.groupby('group').describe()
```

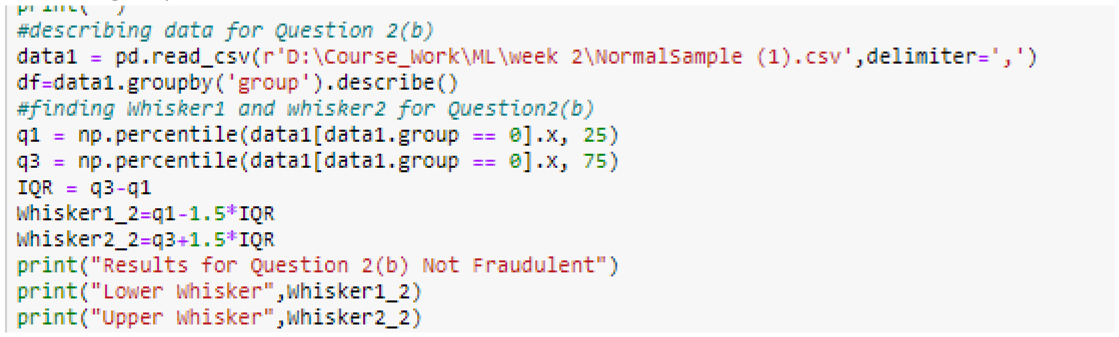


	i								x							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
group																
0	315.0	501.866667	287.813171	2.0	260.50	494.0	747.00	1000.0	315.0	30.004127	0.973935	26.3	29.4	30.0	30.6	32.2
1	686.0	499.142857	289.906257	0.0	241.25	500.5	754.75	998.0	686.0	32.062245	1.040236	29.1	31.4	32.1	32.7	35.4

Group0	**min**	**Max**	**Q1**	**Q3**	**Q2**
	26.3	**32.2**	**29.4**	**30.6**	**30**
Group1	**min**	**Max**	**Q1**	**Q3**	**Q2**
	29.1	**35.4**	**31.4**	**32.7**	**32.1**

2. For the values of 1.5 IQR Whiskers, we know that we can calculate it using following formulas, same logic is been implemented while coding
   I. Lower Whisker= Q1 -1.5*IQR
   II. Upper Whisker= Q3 +1.5*IQR
   Where, IQR = |Q3-Q1|

A. Whiskers for group 0:



```
print(" ")
#describing data for Question 2(b)
data1 = pd.read_csv(r'D:\Course_Work\ML\week 2\NormalSample (1).csv',delimiter=',')
df=data1.groupby('group').describe()
#finding Whisker1 and whisker2 for Question2(b)
q1 = np.percentile(data1[data1.group == 0].x, 25)
q3 = np.percentile(data1[data1.group == 0].x, 75)
IQR = q3-q1
Whisker1_2=q1-1.5*IQR
Whisker2_2=q3+1.5*IQR
print("Results for Question 2(b) Not Fraudulent")
print("Lower Whisker",Whisker1_2)
print("Upper Whisker",Whisker2_2)
```

Output:

```
Results for Question 2(b) group 0
Lower Whisker 27.599999999999994
Upper Whisker 32.400000000000006
```

Lower Whisker	27.599
Upper Whisker	32.400

B. Whiskers for group1:

```
q1 = np.percentile(data1[data1.group == 1].x, 25)
q3 = np.percentile(data1[data1.group == 1].x, 75)
IQR = q3-q1
Whisker1_3=q1-1.5*IQR
Whisker2_3=q3+1.5*IQR
print("Results for Question 2(b) Fraudulent")
print("Lower Whisker",Whisker1_3)
print("Upper Whisker",Whisker2_3)
```

Output:

```
Results for Question 2(b) group 1
Lower Whisker 29.449999999999992
Upper Whisker 34.650000000000006
```

Lower Whisker	29.449
Upper Whisker	34.650

----

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?

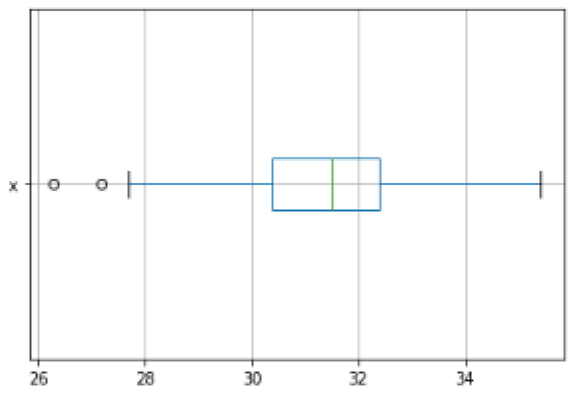
Ans:

Code: for plotting box plot boxplot()function of matplotlib is used

```
#plotting box plot Question 2c
print("plotting box plot Question 2c")
boxplot = data.boxplot(column='x',vert=False)
```

Output:

```
Results for Question 2(b) Fraudulent
```



Results for Upper and lower whisker of x is:

Lower Whisker	27.4
Upper Whisker	35.4

From box plot we can see that both whiskers are almost similar. So, yes 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:

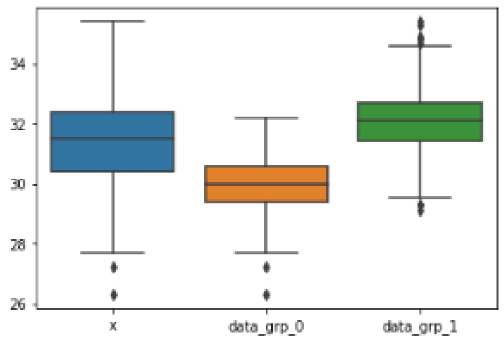
1. For plotting this box plot I used, concat function of pandas and plotted graph using seaborn package's boxplot() function.
2. Using concat function I just concatenated 3 data frames of having data of entire x, x with non-fraudulent group and x with fraudulent group and then plotting same using boxplot() function.
3. The vertical lines above and below box represent 1.5 IQR whiskers.

Code:

```
#question 2d
print("plotting box plot Question 2d combined")
x = data_Q2.x
data_grp_0= data1[data1.group==0].x
data_grp_1= data1[data1.group==1].x
Concatination_=pd.concat([x,data_grp_0,data_grp_1],axis=1, keys=['x','data_grp_0','data_grp_1'])
plt.figure(figsize=(6, 6))
sns.boxplot(data=Concatination_)
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2dd3c6a69b0>
```



4. There are some outliers present for entire data and each group as per my observation, As we can see from boxplot above outliers are represented by small circles in boxplot. Outliers are plotted above and below 1.5IQR whiskers.

   I. For entire data: there are 2 outliers present which are less than lower whisker. There are no outliers above upper whisker.
   Ans:
   Code:

```
#Outliers in 2d
print("outliers of x for entire data are as follows")
xl = list(x)
for i in xl:
    if i < Whisker1_1:
        print("Outliers less than 1st Whisker are:")
        print(i)
    elif i>Whisker2_1:
        print("Outliers greater than 2st Whisker are:")
        print(i)
```

   o/p:

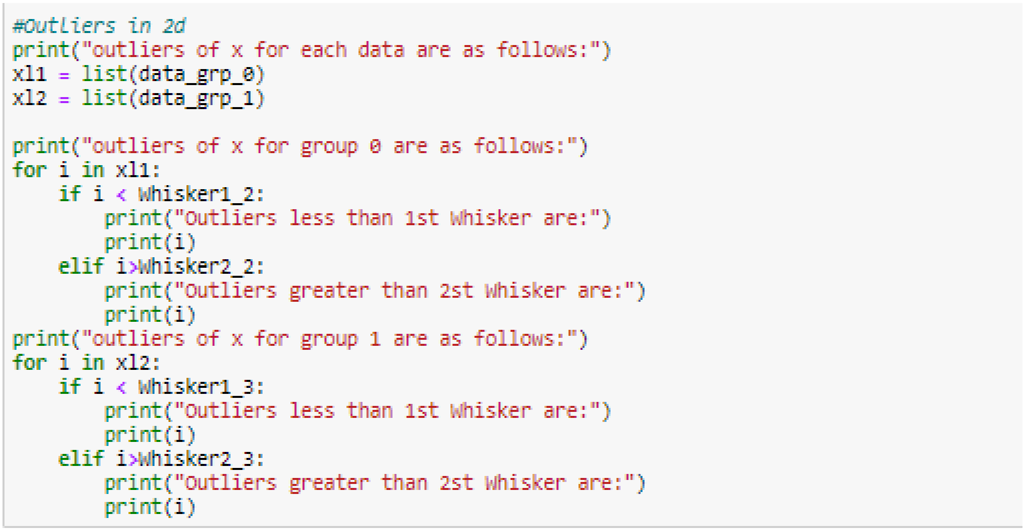
```
outliers of x for entire data are as follows
Outliers less than 1st Whisker are:
27.2
Outliers less than 1st Whisker are:
26.3
```

Lower Whisker outliers	27.2, 26.3

Upper Whisker outliers	NA

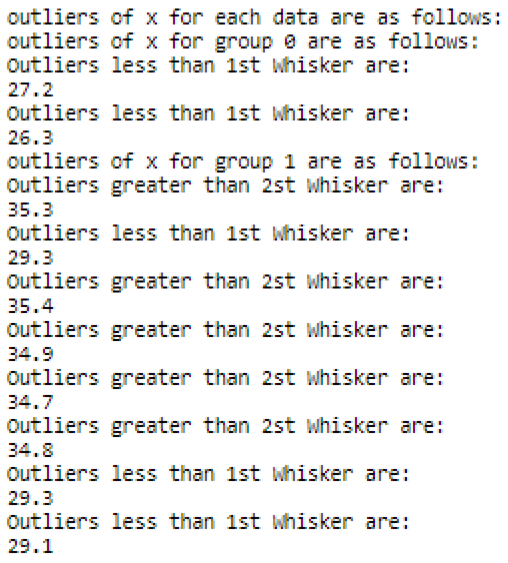
II. For each group:

Code:



```
#Outliers in 2d
print("outliers of x for each data are as follows:")
xl1 = list(data_grp_0)
xl2 = list(data_grp_1)

print("outliers of x for group 0 are as follows:")
for i in xl1:
    if i < Whisker1_2:
        print("Outliers less than 1st Whisker are:")
        print(i)
    elif i>Whisker2_2:
        print("Outliers greater than 2st Whisker are:")
        print(i)
print("outliers of x for group 1 are as follows:")
for i in xl2:
    if i < Whisker1_3:
        print("Outliers less than 1st Whisker are:")
        print(i)
    elif i>Whisker2_3:
        print("Outliers greater than 2st Whisker are:")
        print(i)
```

o/p:



```
outliers of x for each data are as follows:
outliers of x for group 0 are as follows:
Outliers less than 1st Whisker are:
27.2
Outliers less than 1st Whisker are:
26.3
outliers of x for group 1 are as follows:
Outliers greater than 2st Whisker are:
35.3
Outliers less than 1st Whisker are:
29.3
Outliers greater than 2st Whisker are:
35.4
Outliers greater than 2st Whisker are:
34.9
Outliers greater than 2st Whisker are:
34.7
Outliers greater than 2st Whisker are:
34.8
Outliers less than 1st Whisker are:
29.3
Outliers less than 1st Whisker are:
29.1
```

i. For group=0

Lower Whisker outliers	27.2, 26.3
Upper Whisker outliers	NA

ii. For group=1

Lower Whisker outliers	29.1,29.3,29.3
Upper Whisker outliers	35.3,35.4,34.9,34.7,34.8

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.

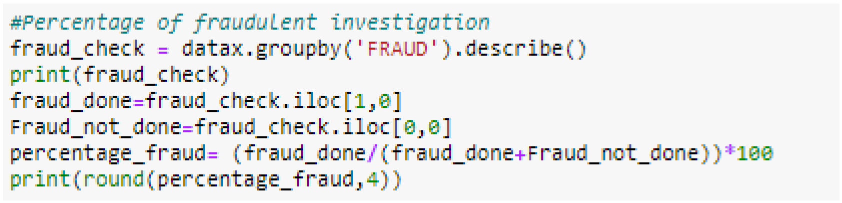
a) (5 points) What percent of investigations are found to be fraudulent? Please give your answer up to 4 decimal places.

Ans:

For this, first I described data using describe() function and grouped it based on FRAUD column. Then I fetched data from describe for fraud done and fraud not done. After that calculated percentage fraud.

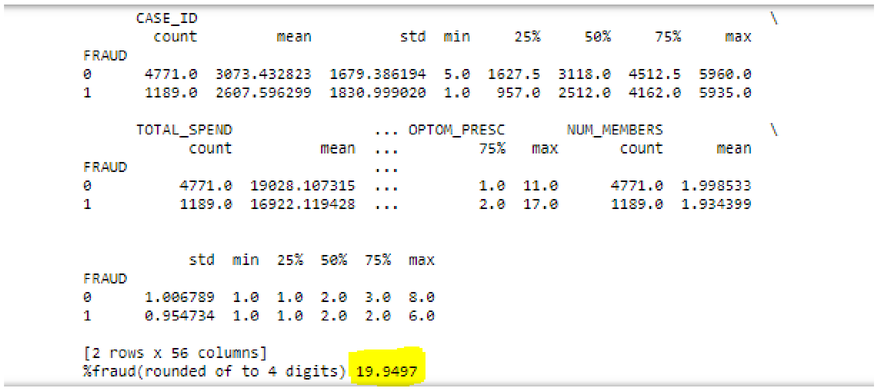
%fraud=[fraud_done/(fraud_done+fraud_not_done)]*100

Code:



```
#Percentage of fraudulent investigation
fraud_check = datax.groupby('FRAUD').describe()
print(fraud_check)
fraud_done=fraud_check.iloc[1,0]
Fraud_not_done=fraud_check.iloc[0,0]
percentage_fraud= (fraud_done/(fraud_done+Fraud_not_done))*100
print(round(percentage_fraud,4))
```

O/p:



```
         CASE_ID                                                           \
           count         mean          std  min     25%     50%     75%     max
FRAUD
0         4771.0  3073.432823  1679.386194  5.0  1627.5  3118.0  4512.5  5960.0
1         1189.0  2607.596299  1830.999020  1.0   957.0  2512.0  4162.0  5935.0

         TOTAL_SPEND               ... OPTOM_PRESC        NUM_MEMBERS           \
               count         mean  ...         75%   max        count      mean
FRAUD                              ...
0             4771.0  19028.107315  ...         1.0  11.0       4771.0  1.998533
1             1189.0  16922.119428  ...         2.0  17.0       1189.0  1.934399


               std  min  25%  50%  75%  max
FRAUD
0         1.006789  1.0  1.0  2.0  3.0  8.0
1         0.954734  1.0  1.0  2.0  2.0  6.0

[2 rows x 56 columns]
%fraud(rounded of to 4 digits) 19.9497
```

Percentage fraud=19.9497%

----

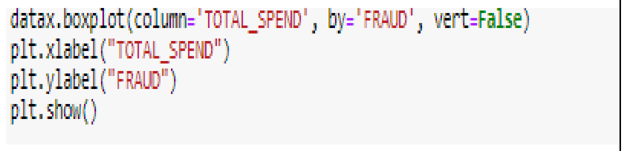
b) (5 points) Use the BOXPLOT function to produce horizontal box-plots. For each interval variable, one box-plot for the fraudulent observations, and another box-plot for the non-fraudulent observations. These two box-plots must appear in the same graph for each interval variable.

Ans:

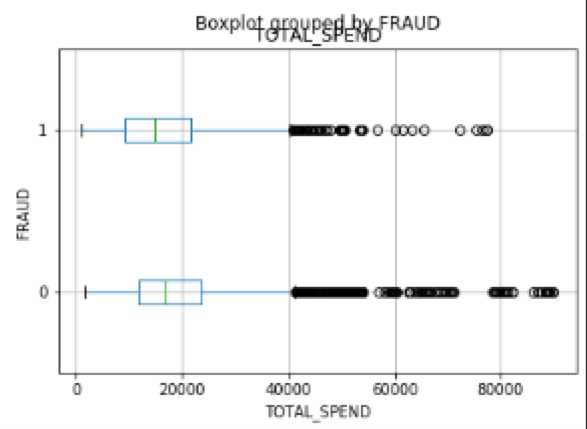
Plotted boxplots using boxplot()function of matplotlib.

Interval variable	code	Plot obtained

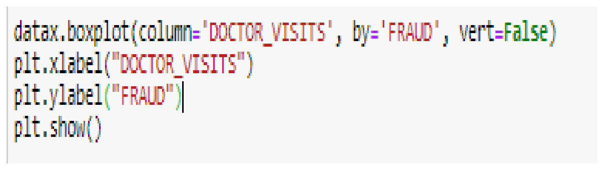
TOTAL_SPEND



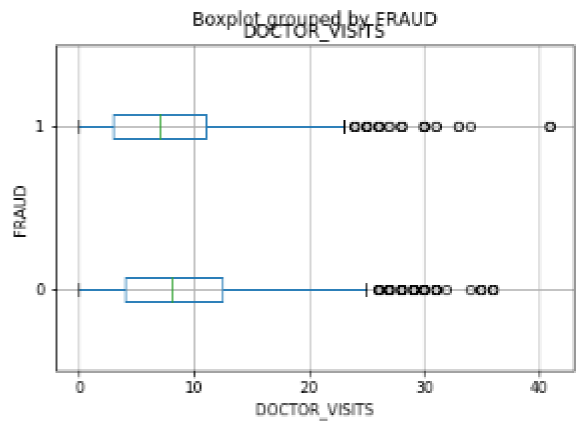
```
datax.boxplot(column='TOTAL_SPEND', by='FRAUD', vert=False)
plt.xlabel("TOTAL_SPEND")
plt.ylabel("FRAUD")
plt.show()
```



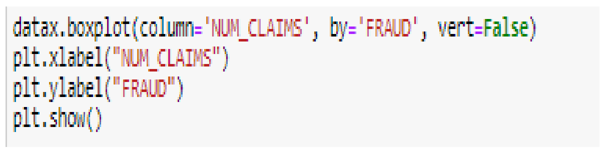
DOCTOR_VISITS



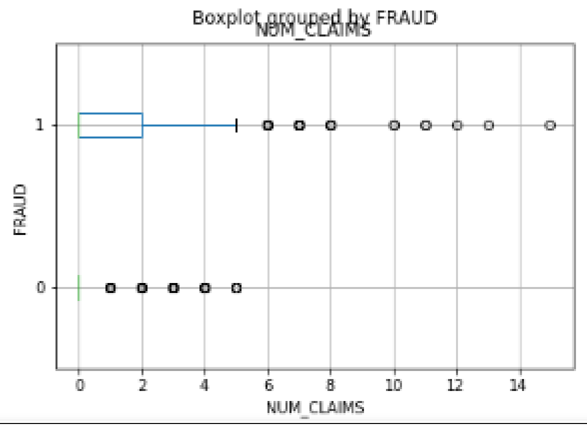
```
datax.boxplot(column='DOCTOR_VISITS', by='FRAUD', vert=False)
plt.xlabel("DOCTOR_VISITS")
plt.ylabel("FRAUD")
plt.show()
```



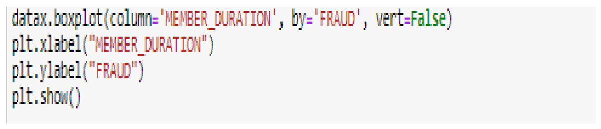
NUM_CLAIMS



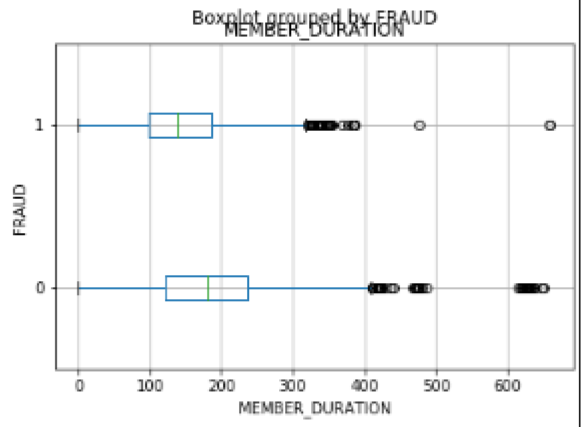
```
datax.boxplot(column='NUM_CLAIMS', by='FRAUD', vert=False)
plt.xlabel("NUM_CLAIMS")
plt.ylabel("FRAUD")
plt.show()
```

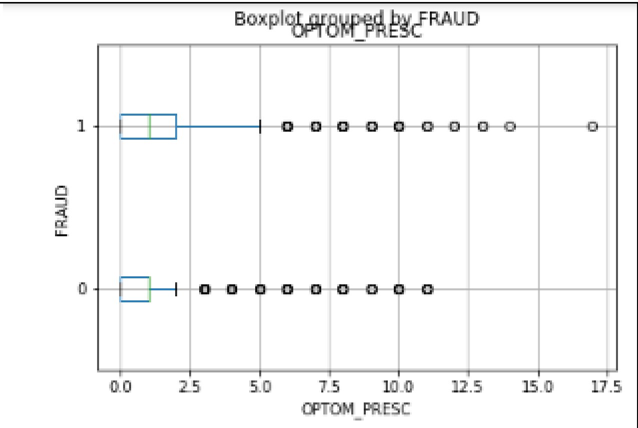


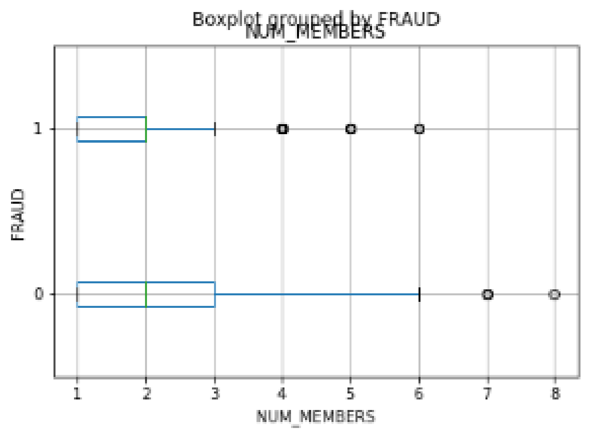
MEMBER_DURATION



```
datax.boxplot(column='MEMBER_DURATION', by='FRAUD', vert=False)
plt.xlabel("MEMBER_DURATION")
plt.ylabel("FRAUD")
plt.show()
```



OPTOM_PRESC	`datax.boxplot(column='OPTOM_PRESC', by='FRAUD', vert=False)`
	`plt.xlabel("OPTOM_PRESC")`
	`plt.ylabel("FRAUD")`
	`plt.show()`
		

NUM_MEMBERS	`datax.boxplot(column='NUM_MEMBERS', by='FRAUD', vert=False)`
	`plt.xlabel("NUM_MEMBERS")`
	`plt.ylabel("FRAUD")`
	`plt.show()`
		

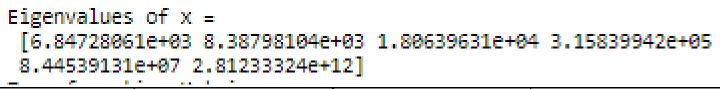
----

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?

Ans: **6 dimensions**

- After calculating eigen values, I found that eigen value for each dimension is greater than one so we need to consider all **6 dimensions** for further calculations.
- If any of the eigen values had value less than 1 I would have dropped it. Eigen values obtained are as follows,



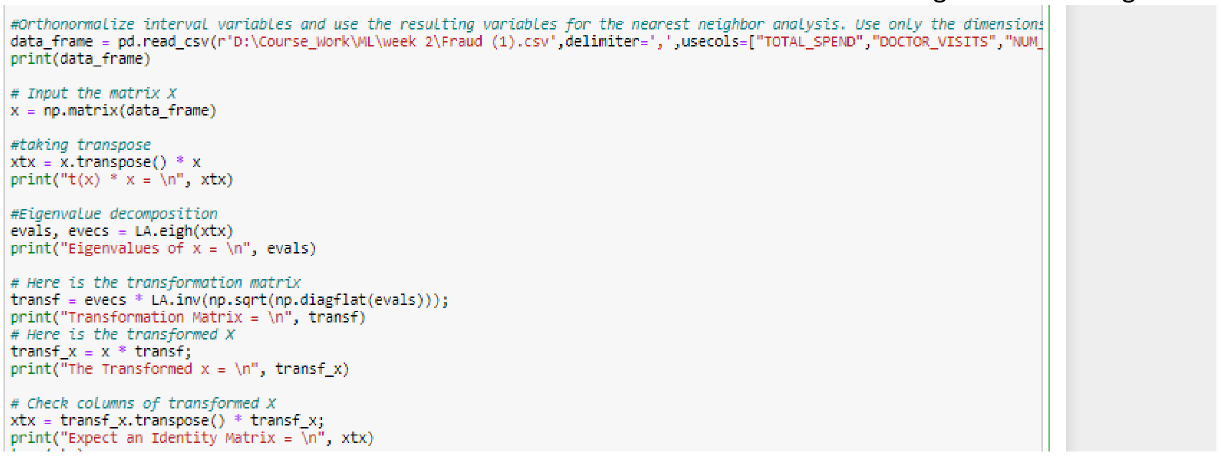
```
Eigenvalues of x =
 [6.84728061e+03 8.38798104e+03 1.80639631e+04 3.15839942e+05
 8.44539131e+07 2.81233324e+12]
```

Eigen Values	6.84728061e+03	8.38798104e+03	1.80639631e+04	3.15839942e+05	8.44539131e+07	2.81233324e+12

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

Ans:

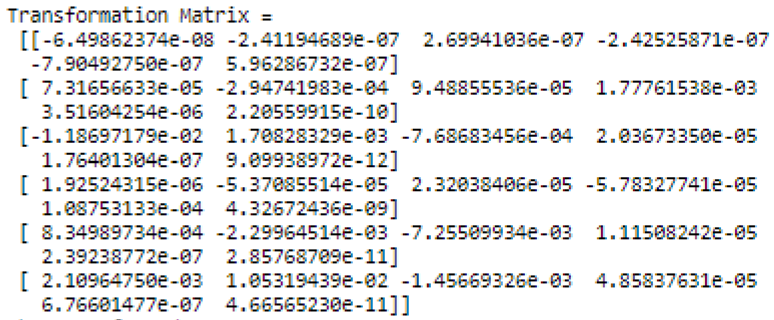
1. **Transformation matrix dimension:6X6**



```
#Orthonormalize interval variables and use the resulting variables for the nearest neighbor analysis. Use only the dimensions
data_frame = pd.read_csv(r'D:\Course_Work\ML\week 2\Fraud (1).csv',delimiter=',',usecols=["TOTAL_SPEND","DOCTOR_VISITS","NUM_
print(data_frame)

# Input the matrix X
x = np.matrix(data_frame)

#taking transpose
xtx = x.transpose() * x
print("t(x) * x = \n", xtx)

#Eigenvalue decomposition
evals, evecs = LA.eigh(xtx)
print("Eigenvalues of x = \n", evals)

# Here is the transformation matrix
transf = evecs * LA.inv(np.sqrt(np.diagflat(evals)));
print("Transformation Matrix = \n", transf)
# Here is the transformed X
transf_x = x * transf;
print("The Transformed x = \n", transf_x)

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

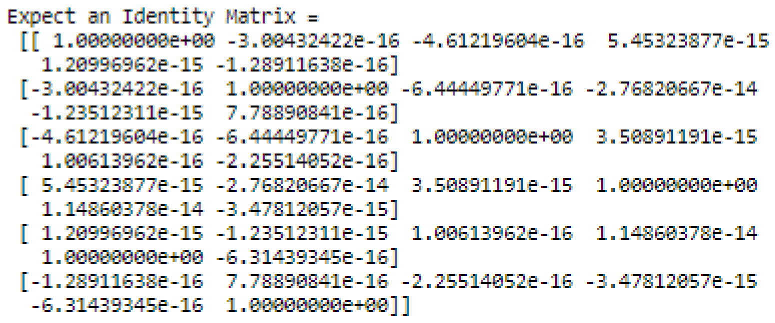
Code:

o/p for 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]]
```

2. x matrix is of size 5960X6 when we multiply it with transformation matrix which is of size 6X6 will get a matrix having dimensions 5960X6. When we multiply transpose of transformation matrix with transformation matrix will get matrix with dimensions 6X6 and it is a identity matrix which proves that 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]]
```

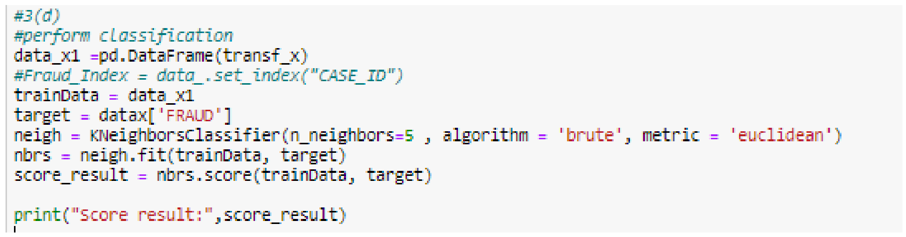
----

d) (10 points) Use the NearestNeighbors module to execute the Nearest Neighbors algorithm using exactly <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
   ii. (5 points) Explain the meaning of the score function return value.

Ans

   i. Code:



```
#3(d)
#perform classification
data_x1 =pd.DataFrame(transf_x)
#Fraud_Index = data_.set_index("CASE_ID")
trainData = data_x1
target = datax['FRAUD']
neigh = KNeighborsClassifier(n_neighbors=5 , algorithm = 'brute', metric = 'euclidean')
nbrs = neigh.fit(trainData, target)
score_result = nbrs.score(trainData, target)

print("Score result:",score_result)
```

Score return value is : Score result**: 0.8778523489932886**

ii. Score function return value which represents fractions of observations which are correctly classified (in short accuracy).

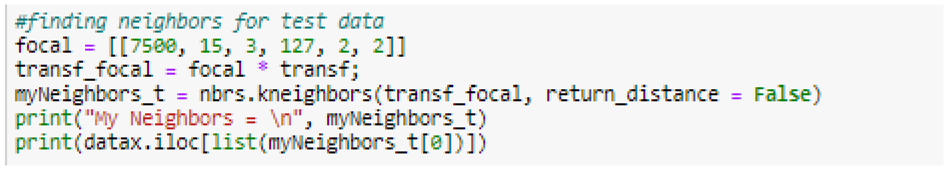
Misclassification rate = 1 – score return value
= 1 - 0.8778523489932886
= 0.1221476510067114=12.21%

---

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*.

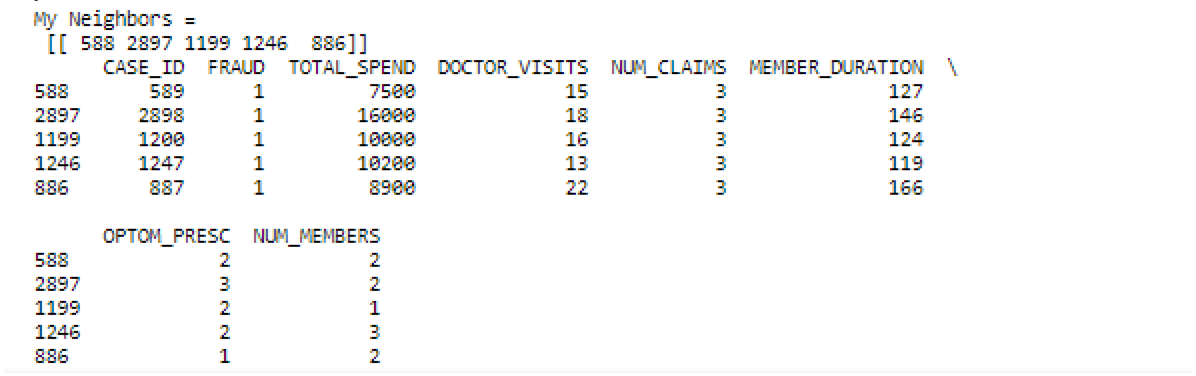
Ans:

Code:



```
#finding neighbors for test data
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(datax.iloc[list(myNeighbors_t[0])])
```

o/p



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

      OPTOM_PRESC  NUM_MEMBERS
588             2            2
2897            3            2
1199            2            1
1246            2            3
886             1            2
```

Explaination:

i. So the indices of 5 neighbors are [[588, 2897, 1199 ,1246, 886]]

ii. And 5 neighbors are as follows, CASE_ID is providing corresponding values to indices which we have found(ex 588 in indices represents to 589).

Indices	CASE_ID	FRAUD	TOTAL_SPEND	DOCTOR_VISIT	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

---

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:

- For all 5 neighbors' values of predicted probability of fraudulent i.e. FRAUD=1 is 1. This we can see from table in 3(e).
- Probability of Fraudulent=
  observations predicted having probability as fraudulent/ total number of observations
- in this case, Probability of Fraudulent= 5/5 i.e 1
- As 1> predicted probability of fraudulent of Q3(a)i.e. 0.199497, hence from given criteria the observations will be predicted as fraudulent too and are not misclassified

----