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?

Answer: The recommended bin width according Izenman (1991) method is :

h = 2(IQR)(N^-1/3). After calculating the bin width as per above formula, we get rounded off value h = 0.4

```
#N =  total number of datapoints in the field x
N = len(normal_df)
N_hat = N**(-1/3)
print("N: ", N)
print("N_hat: ", N_hat)

bin_width = 2*IQR*N_hat
h = round(bin_width,1)
print("Recommended bin-width for the histogram of x",h)
```

```
N:  1001
N_hat:  0.09996668887161934
Recommended bin-width for the histogram of x 0.4
```

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

Answer: Minimum value: 26.30

Maximum value: 35.40

```
x = list(normal_df.x)
print("Minimum Value is: ", min(x))
print("Maximum Value is: ", max(x))
#Minimum value : 30.40
#Maximum value : 35.40
```

```
Minimum Value is:  26.3
Maximum Value is:  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?

Answer: 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 favourite graphing tools.

h = 0.1

```
df = pd.read_csv('C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/NormalSample.csv', header = 0)

x = list(df.x)

#Number of bins:
b = 36
a = 26
bin_no = (b-a)/0.1
print("Number of bins: ",bin_no)

h = 0.1

mid01 = h/2

n = np.arange(26.0,36.01, mid01)

mid = []
for i in range(0,len(n),1):
    if i%2 != 0:
        mid.append(n[i])

N = len(x)
Nh_01 = N*h
p = {}
pl = []

for m in mid:
    w = []
    for i in x:
        u = (i-m)/h
        if u > -0.5 and u <= 0.5:
            w.append(1)
    p[m] = sum(w)/Nh_01
    pl.append(sum(w)/Nh_01)
```



```
new_df01 = pd.DataFrame(list(zip(mid, pl)),
                    columns =['Midpoints', 'Density Estimate'])

print("Desnsity  Estimators are: ")
print(new_df01)
import matplotlib.pyplot as plt
plt.figure(figsize=(10,8))
plt.step(mid,pl, where= 'mid', label = 'h=0.1')
plt.legend()
plt.grid(True)
plt.xticks(np.arange(26, 36, 0.5))
plt.xlabel('Mid-points')
plt.ylabel('Density Estimators')
plt.show()
```

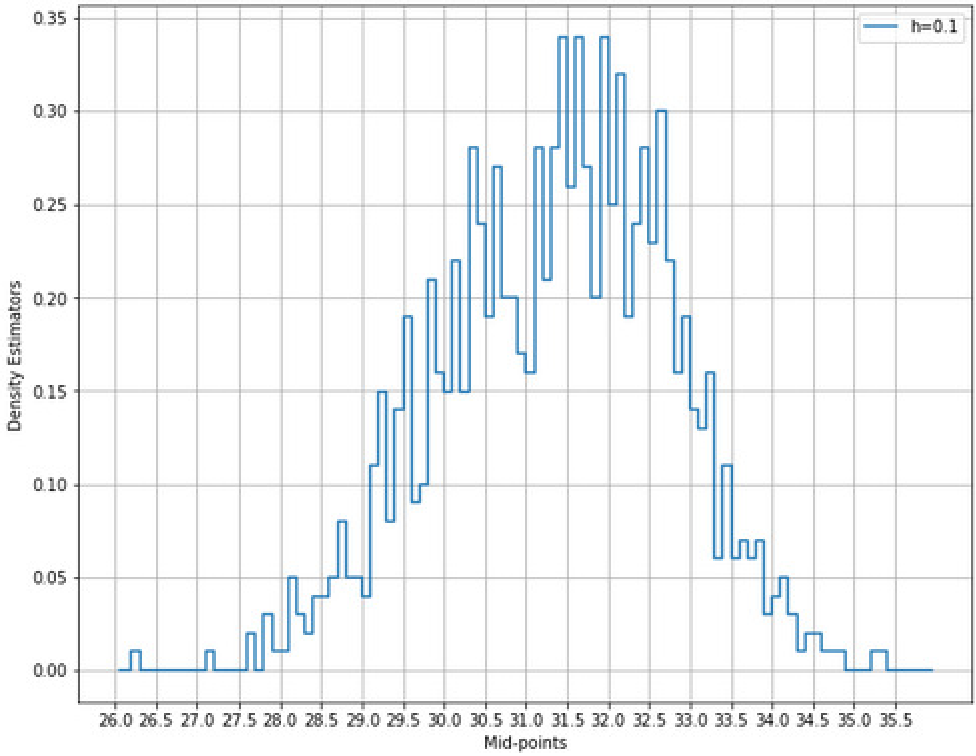
Midpoints, Density Estimators and distribution are as follows:

	Midpoints	Density Estimate
0	26.05	0
1	26.15	0
2	26.25	0.00999001
3	26.35	0
4	26.45	0
5	26.55	0
6	26.65	0
7	26.75	0
8	26.85	0
9	26.95	0
10	27.05	0
11	27.15	0.00999001
12	27.25	0
13	27.35	0
14	27.45	0
15	27.55	0
16	27.65	0.01998002
17	27.75	0
18	27.85	0.02997003
19	27.95	0.00999001
20	28.05	0.00999001
21	28.15	0.04995005
22	28.25	0.02997003
23	28.35	0.01998002
24	28.45	0.03996004
25	28.55	0.03996004
26	28.65	0.04995005
27	28.75	0.07992008
28	28.85	0.04995005
29	28.95	0.04995005
30	29.05	0.03996004
31	29.15	0.10989011
32	29.25	0.14985015
33	29.35	0.07992008
34	29.45	0.13986014
35	29.55	0.18981019
36	29.65	0.08991009
37	29.75	0.0999001
38	29.85	0.20979021
39	29.95	0.15984016
40	30.05	0.14985015
41	30.15	0.21978022
42	30.25	0.14985015
43	30.35	0.27972028
44	30.45	0.23976024
45	30.55	0.18981019
46	30.65	0.26973027
47	30.75	0.1998002
48	30.85	0.1998002
49	30.95	0.16983017
50	31.05	0.15984016
51	31.15	0.27972028

51	31.15	0.27972028
52	31.25	0.20979021
53	31.35	0.27972028
54	31.45	0.33966034
55	31.55	0.25974026
56	31.65	0.33966034
57	31.75	0.26973027
58	31.85	0.1998002
59	31.95	0.33966034
60	32.05	0.24975025
61	32.15	0.31968032
62	32.25	0.18981019
63	32.35	0.23976024
64	32.45	0.27972028
65	32.55	0.22977023
66	32.65	0.2997003
67	32.75	0.21978022
68	32.85	0.15984016
69	32.95	0.18981019
70	33.05	0.13986014
71	33.15	0.12987013
72	33.25	0.15984016
73	33.35	0.05994006
74	33.45	0.10989011
75	33.55	0.05994006
76	33.65	0.06993007

77	33.75	0.05994006
78	33.85	0.06993007
79	33.95	0.02997003
80	34.05	0.03996004
81	34.15	0.04995005
82	34.25	0.02997003
83	34.35	0.00999001
84	34.45	0.01998002
85	34.55	0.01998002
86	34.65	0.00999001
87	34.75	0.00999001
88	34.85	0.00999001
89	34.95	0
90	35.05	0
91	35.15	0
92	35.25	0.00999001
93	35.35	0.00999001
94	35.45	0
95	35.55	0
96	35.65	0
97	35.75	0
98	35.85	0
99	35.95	0

Histogram for h = 0.1



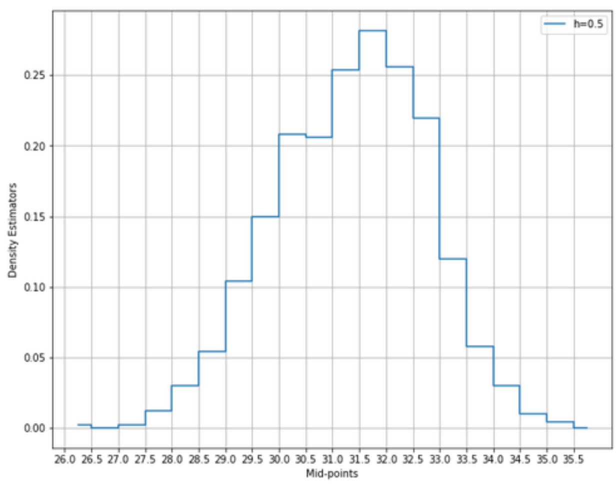
e) (5 points) Use h = 0.5, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favorite graphing tools.

```
h = 0.5
mid05 = h/2

#Number of bins:
b = 36
a = 26
bin_no = (b-a)/h
print("Number of bins: ",bin_no)

n = np.arange(26.0,36.01, mid05)

mid = []
for i in range(0,len(n),1):
    if i%2 != 0:
        mid.append(n[i])

N = len(x)
Nh_05 = N*h
p = {}
pl = []

for m in mid:
    w = []
    for i in x:
        u = (i-m)/h
        if u > -0.5 and u <= 0.5:
            w.append(1)
    p[m] = sum(w)/Nh_05
    pl.append(sum(w)/Nh_05)


new_df05 = pd.DataFrame(list(zip(mid, pl)),
               columns =['Midpoints', 'Density Estimate'])
```

```
print("Desnsity  Estimators are: ")
print(new_df05)
import matplotlib.pyplot as plt
plt.figure(figsize=(10,8))
plt.step(mid,pl, where= 'mid', label = 'h=0.5')
plt.legend()
plt.grid(True)
plt.xticks(np.arange(26, 36, 0.5))
plt.xlabel('Mid-points')
plt.ylabel('Density Estimators')
plt.show()
```

Midpoints, Density Estimators and distribution is as follows:

	Midpoints	Density Estimate
0	26.25	0.001998002
1	26.75	0
2	27.25	0.001998002
3	27.75	0.011988012
4	28.25	0.02997003
5	28.75	0.053946054
6	29.25	0.103896104
7	29.75	0.14985015
8	30.25	0.207792208
9	30.75	0.205794206
10	31.25	0.253746254
11	31.75	0.281718282
12	32.25	0.255744256
13	32.75	0.21978022
14	33.25	0.11988012
15	33.75	0.057942058
16	34.25	0.02997003
17	34.75	0.00999001
18	35.25	0.003996004
19	35.75	0



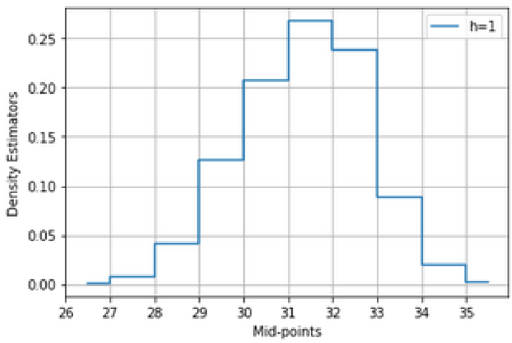
d) (5 points) Use h = 1, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favourite graphing tools.

```
h = 1
mid1 = h/2

#Number of bins:
b = 36
a = 26
bin_no = (b-a)/h
print("Number of bins: ",bin_no)

n = np.arange(26.0,36.01, mid1)

mid = []
for i in range(0,len(n),1):
    if i%2 != 0:
        mid.append(n[i])

N = len(x)
Nh_1 = N*h
p = {}
pl = []

for m in mid:
    w = []
    for i in x:
        u = (i-m)/h
        if u > -0.5 and u <= 0.5:
            w.append(1)
    p[m] = sum(w)/Nh_1
    pl.append(sum(w)/Nh_1)


new_df1 = pd.DataFrame(list(zip(mid, pl)),
               columns =['Midpoints', 'Density Estimate'])
```

```
print("Desnsity  Estimators are: ")
print(new_df1)
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4))
plt.step(mid,pl, where= 'mid', label = 'h=1')
plt.legend()
plt.grid(True)
plt.xticks(np.arange(26, 36,1))
plt.xlabel('Mid-points')
plt.ylabel('Density Estimators')
plt.show()
```

Midpoints, Density Estimators and distribution is as follows:

	Midpoints	Density Estimate
0	26.5	0.000999001
1	27.5	0.006993007
2	28.5	0.041958042
3	29.5	0.126873127
4	30.5	0.206793207
5	31.5	0.267732268
6	32.5	0.237762238
7	33.5	0.088911089
8	34.5	0.01998002
9	35.5	0.001998002

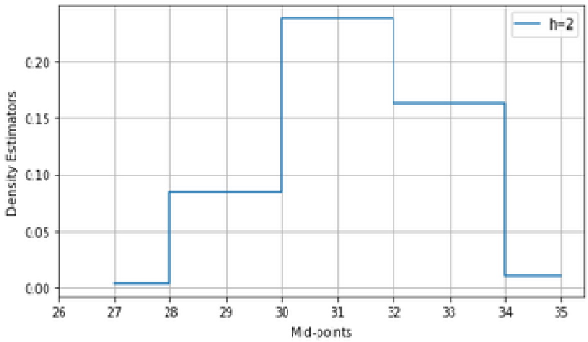


e) (5 points) Use h = 2, minimum = a and maximum = b. List the coordinates of the density estimator. Paste the histogram drawn using Python or your favourite graphing tools.

```
h = 2
mid2 = h/2

#Number of bins:
b = 36
a = 26
bin_no = (b-a)/h
print("Number of bins: ",bin_no)


n = np.arange(26.0,36.01, mid2)

mid = []
for i in range(0,len(n),1):
    if i%2 != 0:
        mid.append(n[i])

N = len(x)
Nh_2 = N*h
p = {}
pl = []

for m in mid:
    w = []
    for i in x:
        u = (i-m)/h
        if u > -0.5 and u <= 0.5:
            w.append(1)
    p[m] = sum(w)/Nh_2
    pl.append(sum(w)/Nh_2)


new_df2 = pd.DataFrame(list(zip(mid, pl)),
               columns =['Midpoints', 'Density Estimate'])
```

```
print("Desnsity  Estimators are: ")
print(new_df2)
import matplotlib.pyplot as plt
plt.figure(figsize=(8,4))
plt.step(mid,pl, where= 'mid', label = 'h=2')
plt.legend()
plt.grid(True)
plt.xticks(np.arange(26, 36, 1))
plt.xlabel('Mid-points')
plt.ylabel('Density Estimators')
plt.show()
```

	Midpoints	Density Estimate
0	27	0.003996004
1	29	0.084415584
2	31	0.237262737
3	33	0.163336663
4	35	0.010989011



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.

Answer: Among 4 histograms with different 'h' values I think h=0.5 gives good spread and distribution. Also it is also closer to the binwidth h = 0.4 that we calculated manually using Izenman (1991) method. The histgram with binwidth h = 0.5, should be a good choice because the data is estimated properly in the bins, i.e the bins are not too thin or unnecessary thick.

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?

Answer:

```
print(normal_df.x.describe())

q1 = np.percentile(normal_df.x, 25)
q3 = np.percentile(normal_df.x, 75)
IQR = q3-q1
print("Q1: ", q1)
print("Q3: ", q3)
print("IQR: ", IQR)
Whisker_lower = q1 - 1.5*(IQR)
Whisker_upper = q3 + 1.5*(IQR)

print("Lower Whisker: ",Whisker_lower)
print("Upper Whisker: ",Whisker_upper)
```

count 1001.000000
mean 31.414585
std 1.397672
min 26.300000
25% 30.400000
50% 31.500000
75% 32.400000
max 35.400000
Name: x, dtype: float64

Values of 1.5 IQR is:
Lower Whisker: 27.4
Upper Whisker: 35.4

FIVE_NUMBER SUMMARY:
Q1 = 30.400000, Q2 = 31.500000, Q3 = 32.400000, MIN = 26.300000, MAX= 35.40000

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?

Answer:

1) Five-number summary of x for category = 0 of the group:

Q1 = 29.400000, Q2 = 30.000000, Q3 =30.600000, MIN =26.300000, MAX= 32.200000

```
#Answer: five-number summary of x for category = 0 of the group:
print("Five-number summary of x for category = 0 of the group:")
df_group0 = normal_df[normal_df.group == 0]
print(df_group0.x.describe())

print("1.5 IQR whiskers for x with category = 0: ")
q10 = np.percentile(df_group0.x, 25)
q30 = np.percentile(df_group0.x, 75)
IQR0 = q30-q10
print("Q1: ", q10)
print("Q3: ", q30)
print("IQR: ", IQR0)
Whisker_lower0 = q10 - 1.5*(IQR0)
Whisker_upper0 = q30 + 1.5*(IQR0)

print("Lower Whisker: ",Whisker_lower0)
print("Upper Whisker: ",Whisker_upper0)
```

Output of .describe() function:

count 315.000000

mean 30.004127

std 0.973935

min 26.300000

25% 29.400000

50% 30.000000

75% 30.600000

max 32.200000

Name: x, dtype: float64

1.5 IQR whiskers for category = 0 of the group:

Lower Whisker: 27.599999999999994

Upper Whisker: 32.400000000000006

2) Five-number summary of x for category = 1 of the group:
Q1 = 31.400000, Q2 = 32.100000, Q3 = 32.700000, MIN = 29.100000, MAX= 35.400000

```
print("\n\nFive-number summary of x for category = 1 of the group:")
df_group1 = normal_df[normal_df.group == 1]
print(df_group1.x.describe())

print("1.5 IQR whiskers for x with category = 1: ")
q11 = np.percentile(df_group1.x, 25)
q31 = np.percentile(df_group1.x, 75)
IQR1 = q31-q11
print("Q1: ", q11)
print("Q3: ", q31)
print("IQR: ", IQR1)
Whisker_lower1 = q11 - 1.5*(IQR1)
Whisker_upper1 = q31 + 1.5*(IQR1)

print("Lower Whisker: ",Whisker_lower1)
print("Upper Whisker: ",Whisker_upper1)
```

Output of .describe() function:
count   686.000000
mean   32.062245
std   1.040236
min   29.100000
25%   31.400000
50%   32.100000
75%   32.700000
max   35.400000
Name: x, dtype: float64

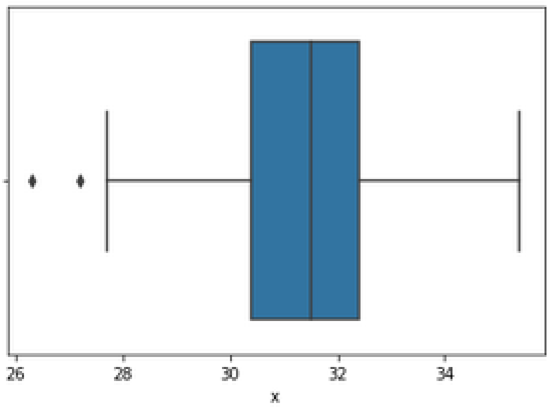
1.5 IQR whiskers for x with category = 1:

Lower Whisker: 29.449999999999992
Upper Whisker: 34.650000000000006

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?

Answer:

```
import seaborn as sns
x_plt = sns.boxplot(x="x" , data = df)
```



Explanation: From the box plot it can be seen that the Python's box plot has displayed the 1.5 IQR whiskers correctly. The package used to plot the box plot is seaborn and the whisker value taken by default is 1.5. This can be verified from manually calculated values. Whiskers manually calculated are: Lower Whisker: 27.599999999999994, Upper Whisker: 32.400000000000006. These values are reflected in the box plot plotted.

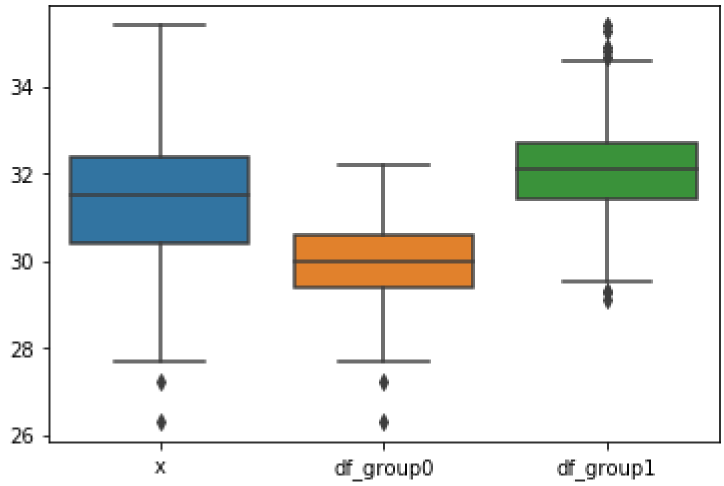
d) (5 points) Draw a graph where it contains the boxplot of x, the boxplot of x for each category of Group (i.e., three boxplots within the same graph frame). Use the 1.5 IQR whiskers, identify the outliers of x, if any, for the entire data and for each category of the group.

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

Answer:

1)

```
import seaborn as sns
from matplotlib import pyplot as plt
import seaborn as sns

x = df.x
df_group0 = normal_df[normal_df.group == 0].x
df_group1 = normal_df[normal_df.group == 1].x

combined_x = pd.concat([x, df_group0, df_group1], axis = 1, keys=['x', 'df_group0', 'df_group1'])

sns.boxplot(data=combined_x)
```



2) Category wise outliers are:

Outliers x are: 27.2, 26.3

Outliers in x for group = 0 are: 27.2, 26.3

Outliers in x for group = 1 are: 35.3, 35.4, 34.9, 34.7, 34.8, 29.3, 29.3, 29.1

```
#Outliers of x,and entire dataframe for each category of the group.

q1 = np.percentile(normal_df.x, 25)
q3 = np.percentile(normal_df.x, 75)
IQR = q3-q1
Whisker_lower = q1 - 1.5*(IQR)
Whisker_upper = q3 + 1.5*(IQR)

#Outliers of x are:
for xx in x:
    if xx > Whisker_upper:
        print("\nOutliers in Upper half of x are: ", xx)
    else:
        pass

for xx in x:
    if xx < Whisker_lower:
        print("\nOutliers in Lower half of x are: ", xx)
    else:
        pass
```

```
Outliers in Lower half of x are:  27.2

Outliers in Lower half of x are:  26.3
```

```python
#Outliers of x,and entire dataframe for group = 0.
df_group0 = normal_df[normal_df.group == 0]
print("1.5 IQR whiskers for x with category = 0: ")
q10 = np.percentile(df_group0.x, 25)
q30 = np.percentile(df_group0.x, 75)
IQR0 = q30-q10
print("Q1: ", q10)
print("Q3: ", q30)
print("IQR: ", IQR0)
Whisker_lower0 = q10 - 1.5*(IQR0)
Whisker_upper0 = q30 + 1.5*(IQR0)

print("Lower Whisker: ",Whisker_lower0)
print("Upper Whisker: ",Whisker_upper0)

# Outliers of x are:
for xx in df_group0.x:
    if xx > Whisker_upper0:
        print("\nOutliers in Upper half of x for group = 0 are: ", xx)
    else:
        pass

for xx in df_group0.x:
    if xx < Whisker_lower0:
        print("\nOutliers in Lower half of x for group = 0 are: ", xx)
    else:
        pass
```

```
1.5 IQR whiskers for x with category = 0:
Q1:  29.4
Q3:  30.6
IQR:  1.2000000000000028
Lower Whisker:  27.599999999999994
Upper Whisker:  32.400000000000006

Outliers in Lower half of x for group = 0 are:  27.2

Outliers in Lower half of x for group = 0 are:  26.3
```

```python
#Outliers of x,and entire dataframe for group = 1.
df_group1 = normal_df[normal_df.group == 1]

print("1.5 IQR whiskers: ")
q11 = np.percentile(df_group1.x, 25)
q31 = np.percentile(df_group1.x, 75)
IQR1 = q31-q11
print("Q1: ", q11)
print("Q3: ", q31)
print("IQR: ", IQR1)
Whisker_lower1 = q11 - 1.5*(IQR1)
Whisker_upper1 = q31 + 1.5*(IQR1)

print("Lower Whisker: ",Whisker_lower1)
print("Upper Whisker: ",Whisker_upper1)


#Outliers of x are:
for xx in df_group1.x:
    if xx > Whisker_upper1:
        print("\nOutliers in Upper half of x for group = 1 are: ", xx)
    else:
        pass

for xx in df_group1.x:
    if xx < Whisker_lower1:
        print("\nOutliers in Lower half of x for group = 1 are: ", xx)
    else:
        pass
```

```
1.5 IQR whiskers:
Q1:  31.4
Q3:  32.7
IQR:  1.3000000000000043
Lower Whisker:  29.449999999999992
Upper Whisker:  34.650000000000006


Outliers in Upper half of x for group = 1 are:  35.3

Outliers in Upper half of x for group = 1 are:  35.4

Outliers in Upper half of x for group = 1 are:  34.9

Outliers in Upper half of x for group = 1 are:  34.7

Outliers in Upper half of x for group = 1 are:  34.8

Outliers in Lower half of x for group = 1 are:  29.3

Outliers in Lower half of x for group = 1 are:  29.3

Outliers in Lower half of x for group = 1 are:  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.

Answer:

```
Fraud_df = pd.read_csv('C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/Fraud.csv')
tot = len(list(Fraud_df.FRAUD))
true_fraud = len(Fraud_df[Fraud_df.FRAUD == 1].FRAUD)
fraud_percent = round((true_fraud/tot)*100,4)
print("Percent of fraudulent data: ",fraud_percent)
```

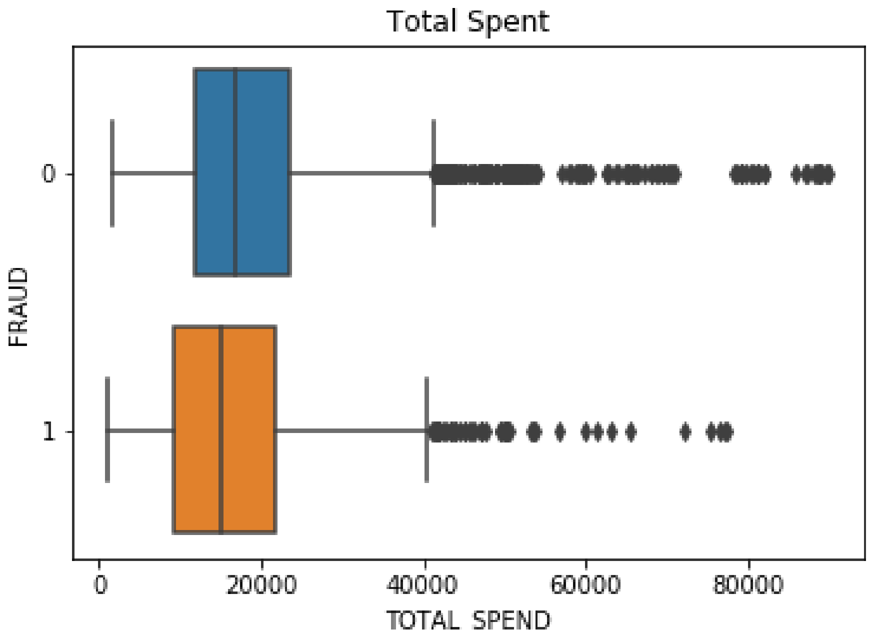
```
Percent of fraudulent data:  19.9497
```

Percent of fraudulent data is 19.9497

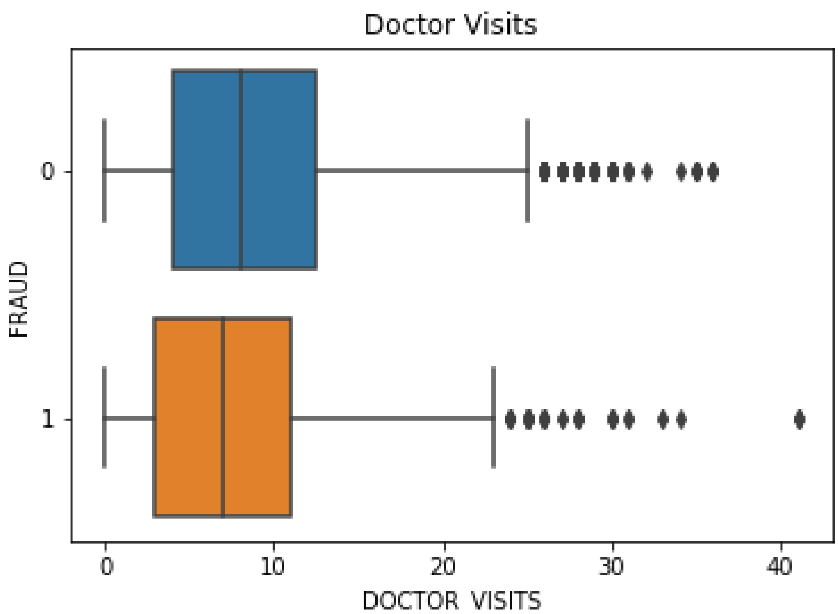
b) (5 points) Use the BOXPLOT function to produce horizontal box-plots. For each interval variable, one box-plot for the fraudulent observations, and another box-plot for the non-fraudulent observations. These two box-plots must appear in the same graph for each interval variable.

Answer:

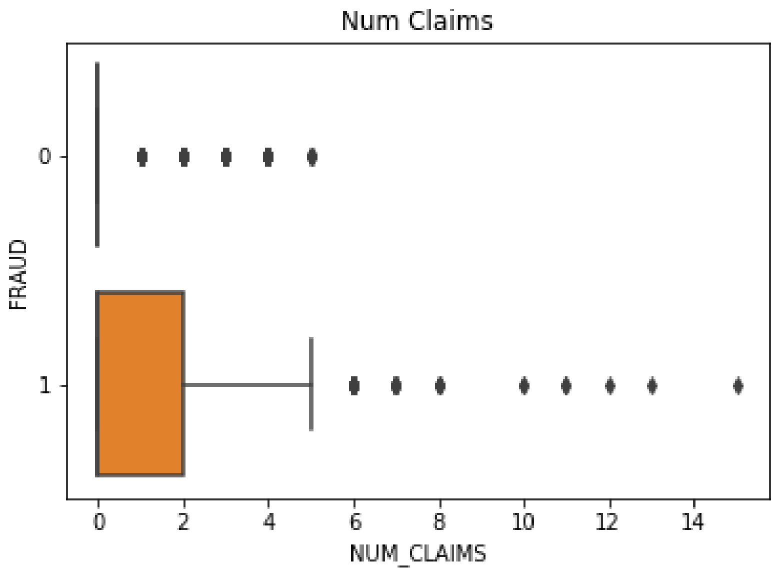
```
import seaborn as sns
sns.boxplot(data = Fraud_df, x = 'TOTAL_SPEND', y = 'FRAUD', orient = 'h')
plt.title("Total Spent")
plt.savefig("C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/TotalSpent.png")
```



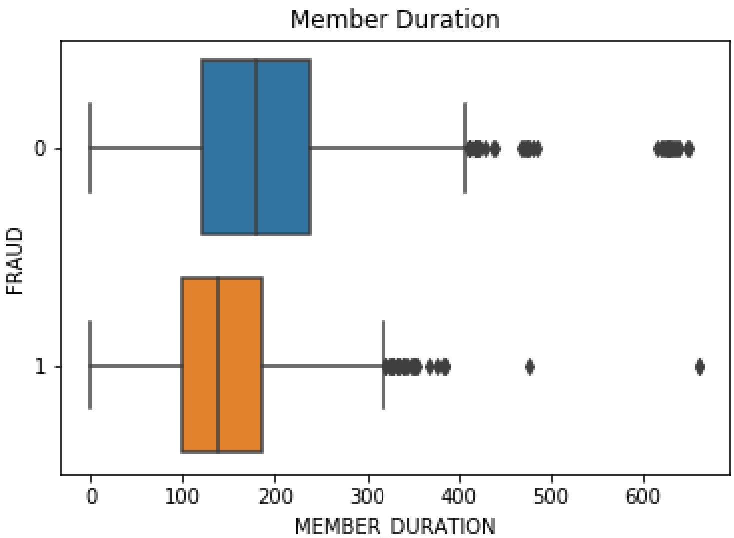

```
import seaborn as sns
sns.boxplot(data = Fraud_df, x = 'DOCTOR_VISITS', y = 'FRAUD', orient = 'h')
plt.title("Doctor Visits")
plt.savefig("C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/DOCTOR_VISITS.png")
```



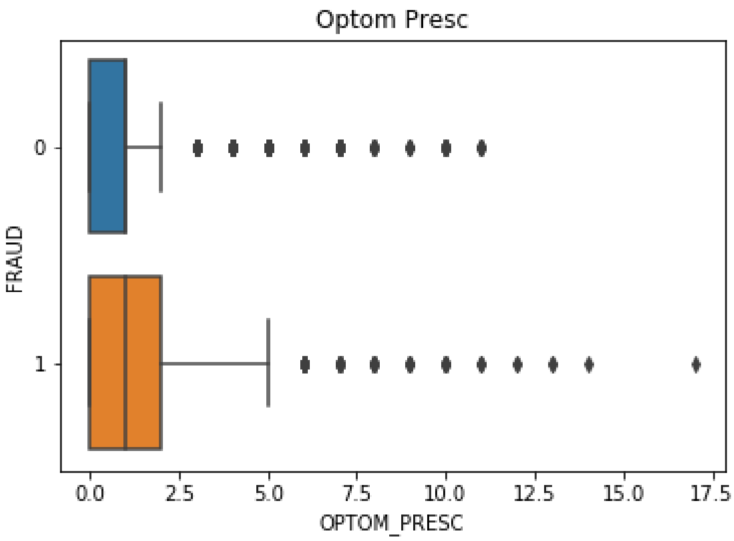
```python
import seaborn as sns
sns.boxplot(data = Fraud_df, x = 'NUM_CLAIMS', y = 'FRAUD', orient = 'h')
plt.title("Num Claims")
plt.savefig("C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/NUM_CLAIMS.png")
```



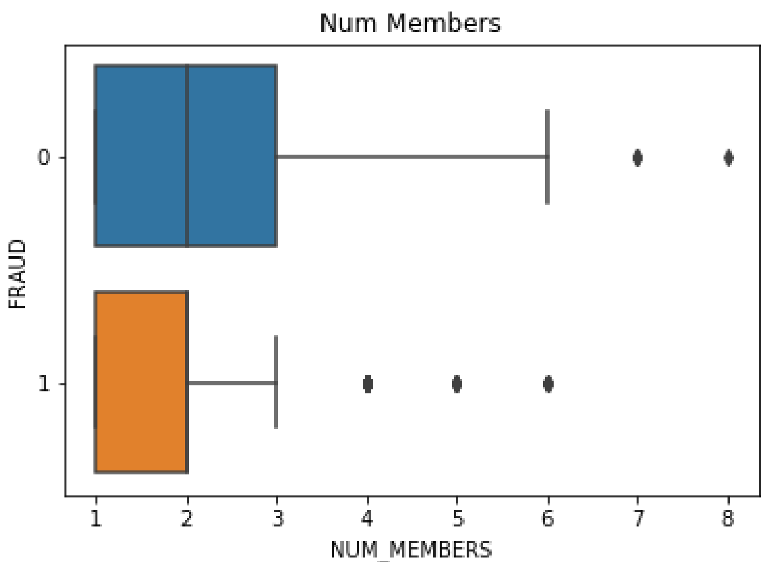

```python
import seaborn as sns
sns.boxplot(data = Fraud_df, x = 'MEMBER_DURATION', y = 'FRAUD', orient = 'h')
plt.title("Member Duration")
plt.savefig("C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/MEMBER_DURATION.png")
```



```python
import seaborn as sns
sns.boxplot(data = Fraud_df, x = 'OPTOM_PRESC', y = 'FRAUD', orient = 'h')
plt.title("Optom Presc")
plt.savefig("C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/OPTOM_PRESC.png")
```




```python
import seaborn as sns
sns.boxplot(data = Fraud_df, x = 'NUM_MEMBERS', y = 'FRAUD', orient = 'h')
plt.title("Num Members")
plt.savefig("C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/NUM_MEMBERS.png")
```



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?

```
Fraud_df_drop = Fraud_df.drop(['CASE_ID','FRAUD'],axis = 1)
x = np.matrix(Fraud_df_drop)
from numpy import linalg as LA
xtx = x.transpose() * x
print("t(x) * x = \n", xtx)

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

# 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)

## Since the eigenvalues of all 6 variables are greater than 1, the dimension is = 6

# Orthonormalize using the orth function
import scipy
from scipy import linalg as LA2

orthx = LA2.orth(x)
print("The orthonormalize x = \n", orthx)

# Check columns of the ORTH function
check = orthx.transpose().dot(orthx)
print("Also Expect an Identity Matrix = \n", check)

# Since the resulting matrix is an Identity Matrix, variables in dataset are orthonormal
```

We have got eigen values of the following six variables greater than 1:
TOTAL_SPEND , DOCOR_VISITS , NUM_CLAIMS , MEMBER_DURATION , OPTOM_PRESC , NUM_MEMBERS . Hence we choose all the 6 variables, and hence we have 6 dimensios.

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

The transformation matrix is as follows:

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]]

The resulting variables are othornormal this can be validated as the dot product of (transpose of orthonormalized matrix) and orthonormalized matrix results in a Identity Matrix as follows:

[[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]]

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.

```
from sklearn.neighbors import KNeighborsClassifier as Knn
from sklearn import metrics

Fraud_data = pd.read_csv('C:/Users/KACHI/Desktop/ALL/1. SEM1/ML/Assignment1/Fraud.csv')
kNNSpec = Knn(n_neighbors=5 , algorithm = 'brute', metric = 'euclidean')
trainData = transf_x
target = Fraud_data.FRAUD
kNNSpec = kNNSpec.fit(trainData, target)
predict = kNNSpec.predict(trainData)
print(metrics.accuracy_score(target, predict))
```

```
0.8778523489932886
```

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

0.8778523489932886. The score function gives 87.7852 accuracy.

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

The score function gives the accuracy of correctly classified data. The value we got is 87.78%, which signifies a good accuracy.

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 neighbours*.

Answer:

```
xt = [[7500,15,3,127,2,2]]* transf
predict = kNNSpec.predict(xt)
neigh = kNNSpec.kneighbors(xt, return_distance=False)
print("The nearest five neighbors are: ",neigh)
print(Fraud_df.iloc[neigh[0][0:]])
```

```
The nearest five neighbors are:  [[ 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
```

The five neighbours are as follows:

[1, 1, 1, 1, 1] and their indices are [588, 2897, 1199, 1246, 886]

The input variables and target values for 5 neighbours are as follows:

We have obtained the input variables which are mapped against the nearest neighbours.

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?

Answer:

Predicted probability of fraudulent is calculated as: Predicted_as_fraud/total_observations As per the result in Q3e) the predicted we have got FRAUD = 1 for all the five neighbours, therefore Predicted_as_fraud = 5 and observations = 5. So, Predicted probability of fraudulent, 5/5 = 1. This probability is much greater than the one we got in Q3a, i.e. 1 > 0.199497. Hence, the classification is fraudulent as per the given criteria and this observation is not misclassified.