ML assignment 1

September 5, 2018

In [10]:
```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from numpy import linalg as LA
import scipy
from scipy import linalg as LA2
from sklearn.neighbors import NearestNeighbors as kNN
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import iqr
x, group = np.loadtxt('NormalSample.csv', delimiter = ',', unpack = True)
print(x)
print(iqr(x))
h = (2*iqr(x))/np.cbrt(x.size)
print("h = ",h)
```

```
[50.91 50.03 50.93 49.67 49.73 50.67 51.28 51.37 49.06 48.27 51.11 50.96
 49.56 49.84 49.67 50.03 51.01 49.49 49.67 50.18 50.97 49.9  49.82 51.09
 51.11 49.05 50.11 51.41 48.63 51.51 49.4  48.79 49.88 49.78 50.26 51.36
 50.11 50.74 50.64 48.84 49.55 50.25 49.64 49.59 49.86 51.58 50.96 50.19
 51.37 49.53 51.51 48.41 50.7  49.31 50.22 49.8  51.82 49.38 50.36 50.61
 50.25 48.36 49.97 50.6  51.72 49.24 48.17 51.71 50.47 49.98 48.57 50.84
 51.07 49.61 49.28 51.58 50.49 51.49 49.33 47.82 49.24 49.3  49.93 51.25
 50.07 48.9  48.59 49.96 51.94 49.35 49.17 50.93 49.91 49.54 49.29 49.23
 50.9  50.22 50.29 50.79]
1.4474999999999998
h =  0.6237088427642294
```

In [11]:
```
u = np.log10(h)
v = np.sign(u) * math.ceil(abs(u))
h = np.power(10,v)
print(h)
```

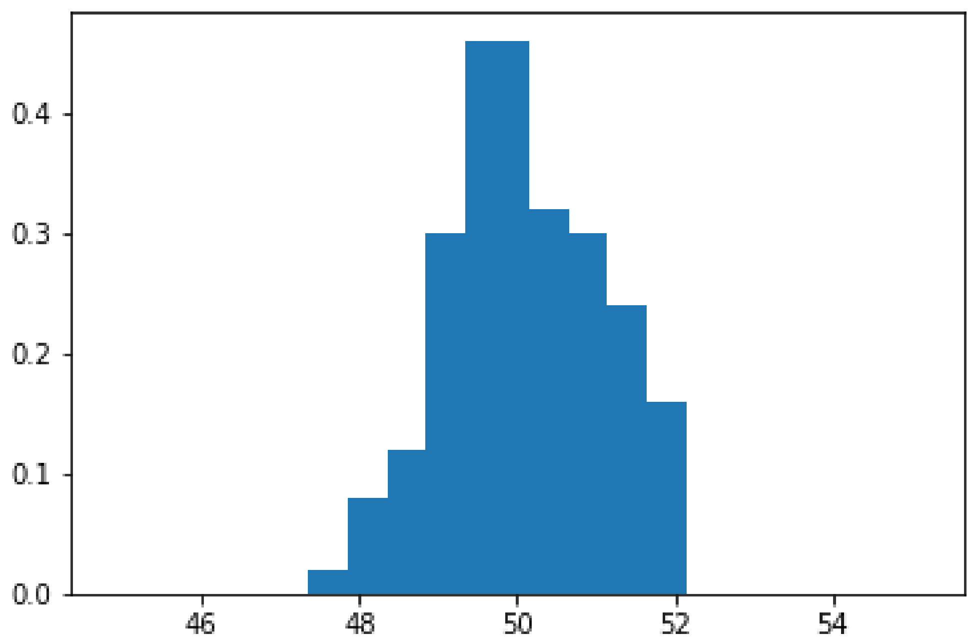
Out[11]: 0.1

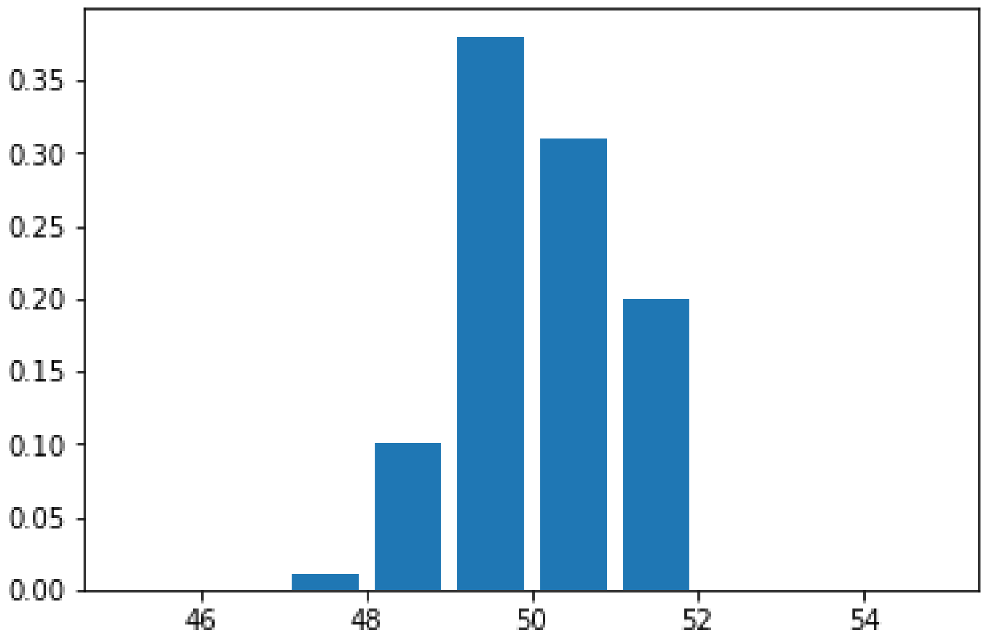
In [12]:
```
N  = x.size
h = 0.5
```

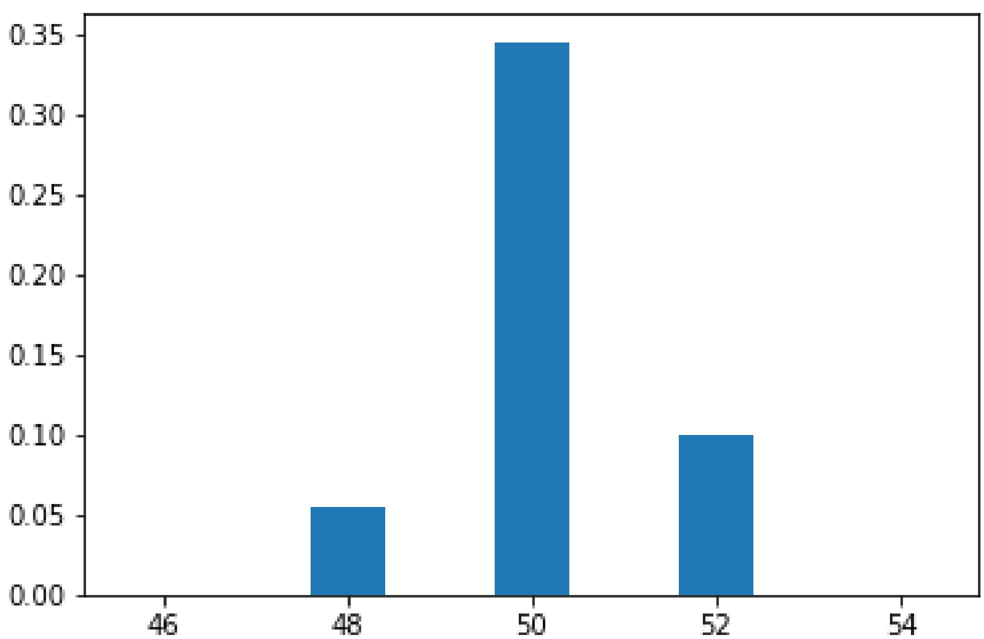
```python
def find_density_estimates(x,h):
    min = 45
    max = 55
    mid_points = [min+h/2]
    for i in range(1,x.size):
        if(mid_points[i-1]+h <= max):
            mid_points.append(mid_points[i-1]+h)
            #print(mid_points[i])
        else: break

    p = []
    for i in range(len(mid_points)):
        w_sum = 0
        u = []
        w = []
        for j in range(x.size):
            u.append((x[j] - mid_points[i])/h)
            if(u[j]>-0.5 and u[j] <= 0.5):
                w.append(1)
            else:
                w.append(0)
            w_sum = w_sum + w[j]
        #print(w_sum)
        p.append(w_sum / (x.size*h))
        del u
        del w
    return p, mid_points

p, mid_points = find_density_estimates(x,h)
print(p)
a = plt.figure(1)
plt.bar(mid_points,p)
a.show
del p

h = 1
p, mid_points = find_density_estimates(x,h)
print(p)
a = plt.figure(2)
plt.bar(mid_points,p)
a.show
del p

h = 2
p, mid_points = find_density_estimates(x,h)
```

```
print(p)
a = plt.figure(3)
plt.bar(mid_points,p)
a.show
del p
del x
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.02, 0.08, 0.12, 0.3, 0.46, 0.32, 0.3, 0.24, 0.16, 0.0, 0.0, 0.0, 0
[0.0, 0.0, 0.01, 0.1, 0.38, 0.31, 0.2, 0.0, 0.0, 0.0]
[0.0, 0.055, 0.345, 0.1, 0.0]
```





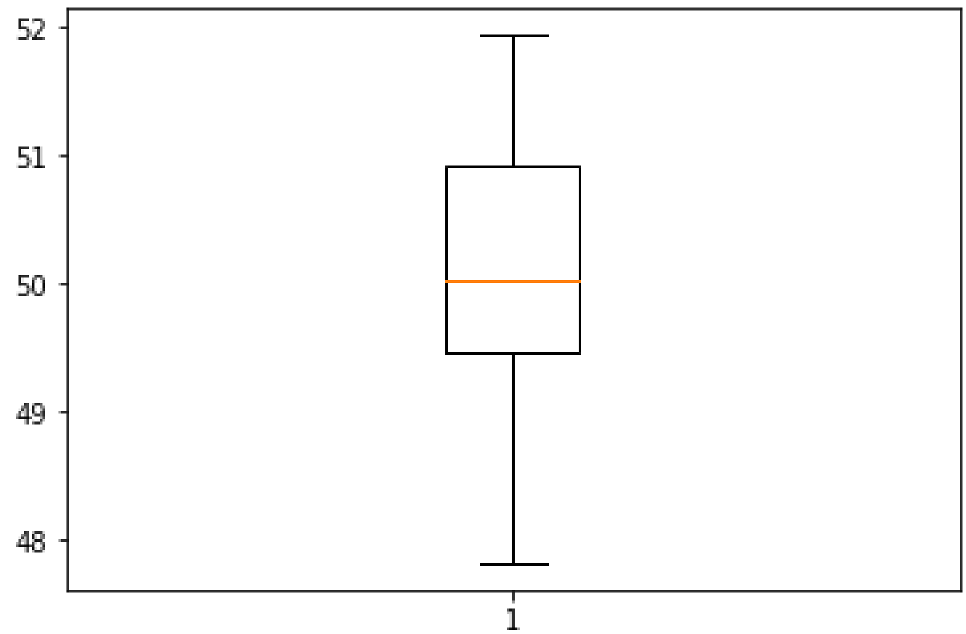


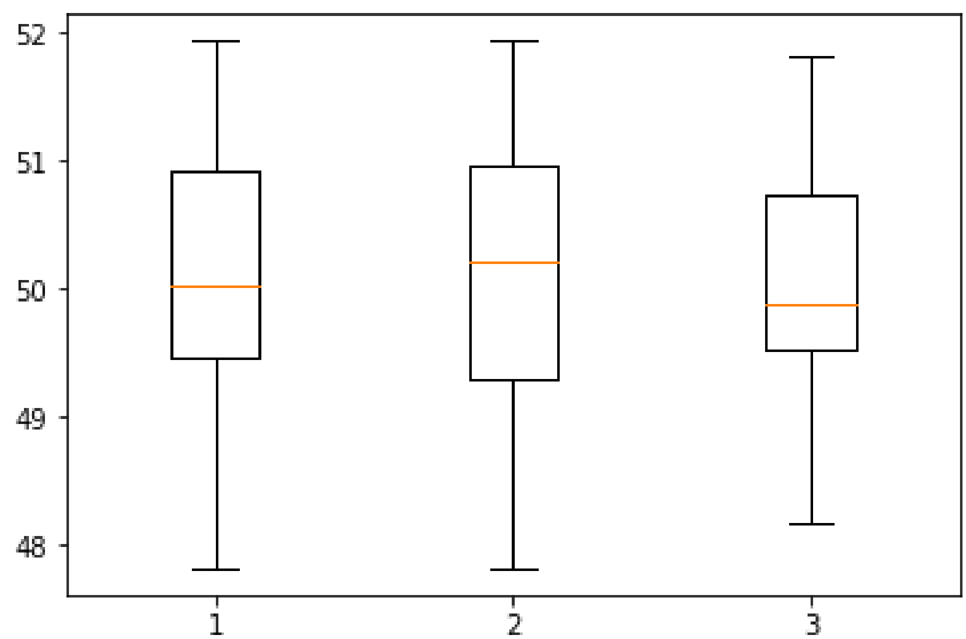

```
In [13]: data = pd.read_csv("NormalSample_labelled.csv")
         col_x = data.iloc[:,0]
         # calculate quartiles
```

```
quartiles = np.percentile(col_x, [25, 50, 75])
# calculate min/max
data_min, data_max = col_x.min(), col_x.max()
print(data_min,quartiles,data_max)

grpBy = data.groupby('Group')
groups = [grpBy.get_group(g) for g in grpBy.groups]

g0_max = groups[0].iloc[:,0].max()
g0_min = groups[0].iloc[:,0].min()
g0_25, g0_50, g0_75 = np.percentile(groups[0].iloc[:,0], [25, 50, 75])
print(g0_min,g0_25,g0_50,g0_75,g0_max)

g1_max = groups[1].iloc[:,0].max()
g1_min = groups[1].iloc[:,0].min()
g1_25, g1_50, g1_75 = np.percentile(groups[1].iloc[:,0], [25, 50, 75])
print(g1_min,g1_25,g1_50,g1_75,g1_max)

box_fig = plt.figure(4)
plt.boxplot(col_x)
box_fig.show()

box_plot = plt.figure(5)
plt.boxplot([col_x, groups[0].iloc[:,0], groups[1].iloc[:,0]],positions = [1,2,3])
box_plot.show()
```

```
47.82 [49.4675 50.03   50.915 ] 51.94
47.82 49.295 50.22 50.96 51.94
48.17 49.53 49.88 50.74 51.82
```

```
c:\users\suhas\pyvenevs\fall\lib\site-packages\matplotlib\figure.py:457: UserWarning: matplotli
  "matplotlib is currently using a non-GUI backend, "
```






```
In [14]: df = pd.read_csv('Fraud.csv')
         # Input the matrix X
         print(" Fraud % = ", (df.iloc[:,1].sum()/df.iloc[:,1].size)*100)
```

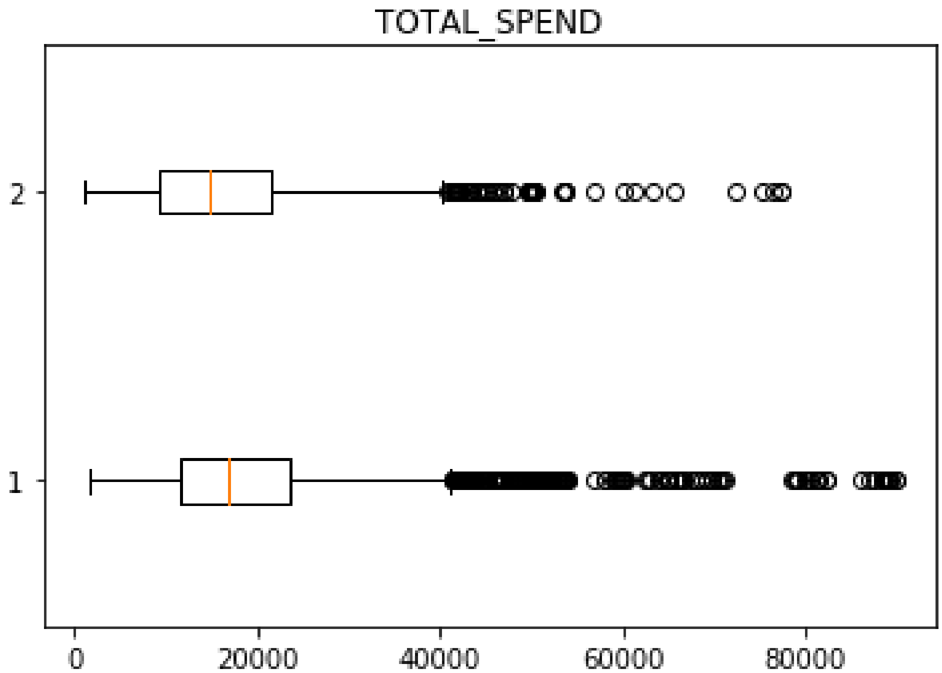
```
fraud_grpBy = df.groupby('FRAUD')
fraud_grps = [fraud_grpBy.get_group(g) for g in fraud_grpBy.groups]

box_plot = plt.figure(6)
plt.boxplot([fraud_grps[0].iloc[:,2],fraud_grps[1].iloc[:,2]],positions = [1,2],vert =
plt.title('TOTAL_SPEND')
box_plot.show()

box_plot = plt.figure(7)
plt.boxplot([fraud_grps[0].iloc[:,3],fraud_grps[1].iloc[:,4]],positions = [1,2],vert =
plt.title('DOCTOR_VISITS')
box_plot.show()

box_plot = plt.figure(8)
plt.boxplot([fraud_grps[0].iloc[:,4],fraud_grps[1].iloc[:,4]],positions = [1,2],vert =
plt.title('NUM_CLAIMS')
box_plot.show()

box_plot = plt.figure(9)
plt.boxplot([fraud_grps[0].iloc[:,5],fraud_grps[1].iloc[:,5]],positions = [1,2],vert =
plt.title('MEMBER_DURATION')
box_plot.show()

box_plot = plt.figure(10)
plt.boxplot([fraud_grps[0].iloc[:,6],fraud_grps[1].iloc[:,6]],positions = [1,2],vert =
plt.title('OPTOM_PRESC')
box_plot.show()

box_plot = plt.figure(11)
plt.boxplot([fraud_grps[0].iloc[:,7],fraud_grps[1].iloc[:,7]],positions = [1,2],vert =
plt.title('NUM_MEMBERS')
box_plot.show()

df_vals = df.values
x = np.delete(df_vals,[0,1],1)

print("Input Matrix = \n", x)

print("Number of Dimensions = ", x.ndim)

print("Number of Rows = ", np.size(x,0))
print("Number of Columns = ", np.size(x,1))

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

# Eigenvalue decomposition
```

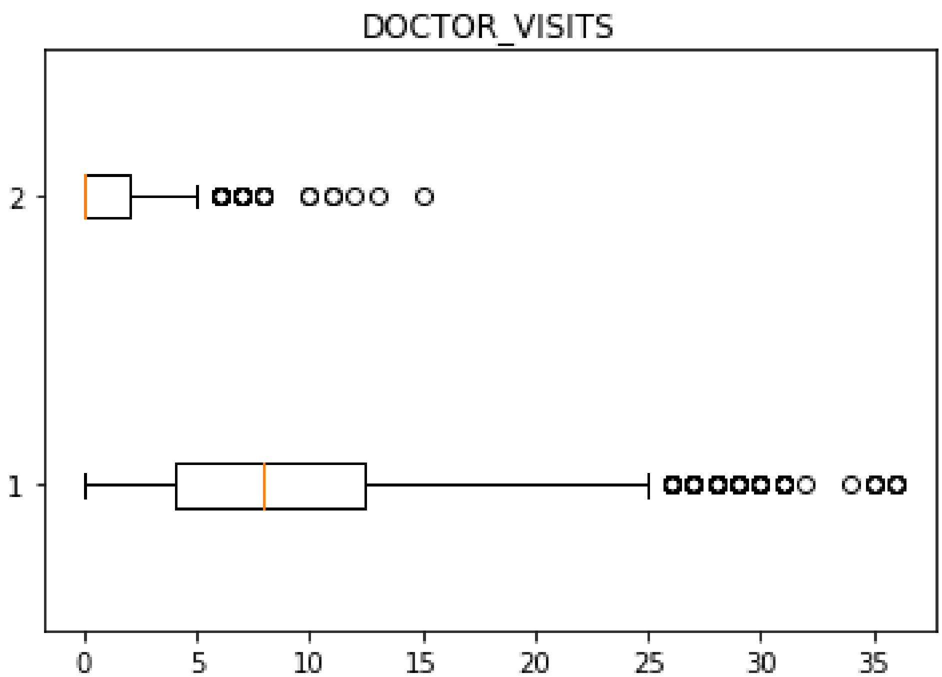
```python
evals, evecs = LA.eigh(xtx)
print("Eigenvalues of x = \n", evals)
print("Eigenvectors of x = \n",evecs)

# Here is the transformation matrix
transf = evecs.dot(LA.inv(np.sqrt(np.diagflat(evals))));
print("Transformation Matrix = \n", transf)

# Here is the transformed X
#transf_x = x * transf;
transf_x = np.matmul(x,transf)
print("The Transformed x = \n", transf_x)

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

# Orthonormalize using the orth function

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", np.round(check))
```

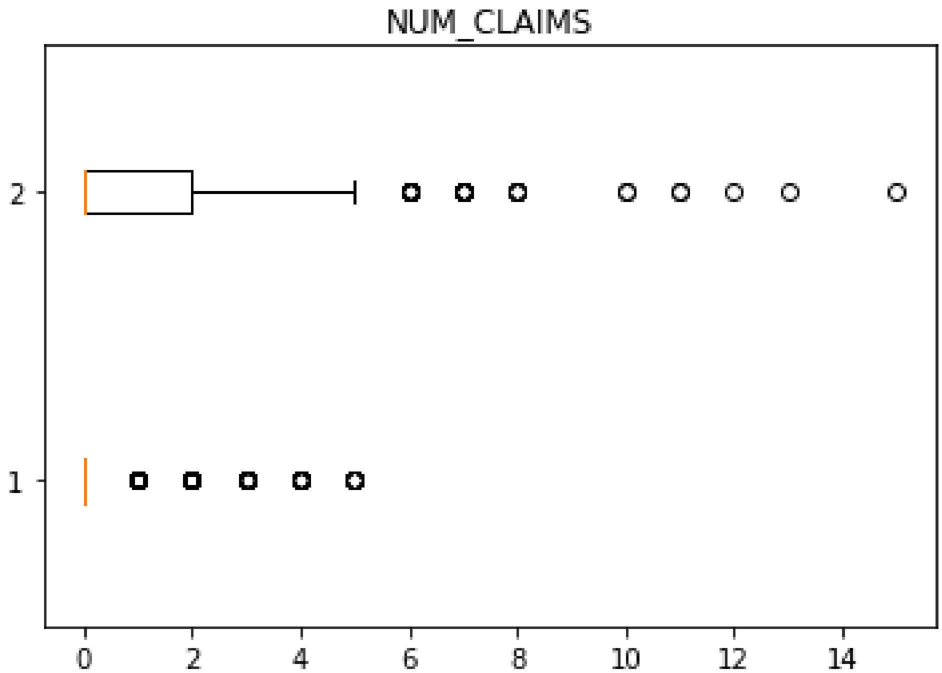
```
 Fraud % =  19.949664429530202
Input Matrix =
 [[ 1100    11     0    94     1     2]
 [ 1300     7     2   122     0     1]
 [ 1500     4     0   149     1     3]
 ...
 [89200    15     0   212     0     2]
 [89800    14     0   214     0     2]
 [89900    15     0   220     0     1]]
Number of Dimensions =  2
Number of Rows =  5960
Number of Columns =  6
t(x) * x =
 [[2812184770000    1040176400      42913200   20404919400     134771800
     220035900]
 [   1040176400        788159         23809      10264845         57654
        106717]
 [     42913200         23809          7922        448090          3459
          4765]
 [  20404919400      10264845        448090     232422585       1163391
       2121127]
```

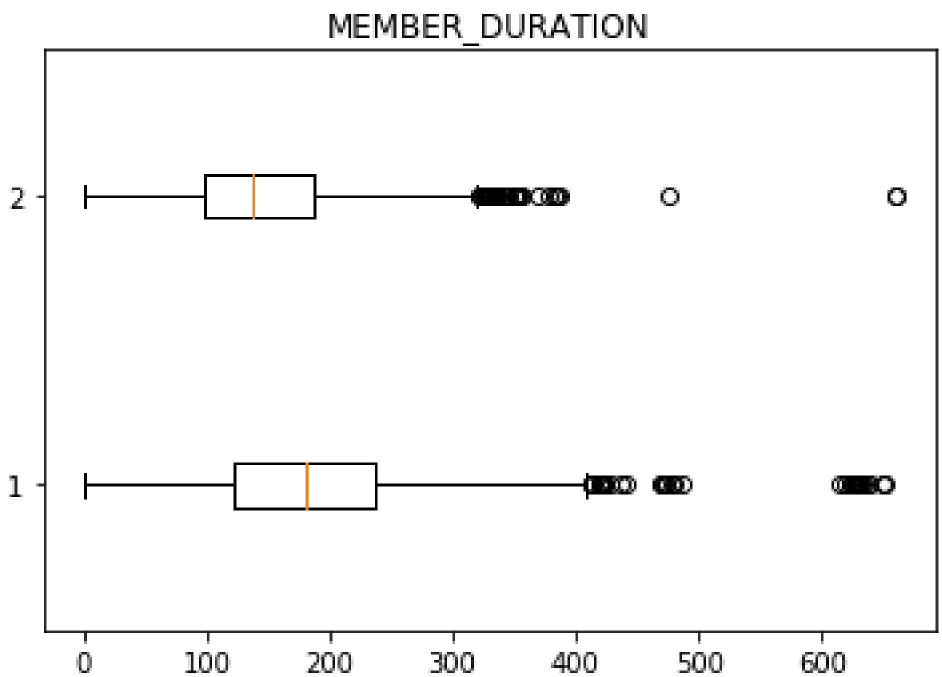
```
 [    134771800         57654          3459       1163391         24460
          13581]
 [    220035900        106717          4765       2121127         13581
          29423]]
Eigenvalues of x =
 [6.84728061e+03 8.38798104e+03 1.80639631e+04 3.15839942e+05
 8.44539131e+07 2.81233324e+12]
Eigenvectors of x =
 [[-5.37750046e-06 -2.20900379e-05  3.62806809e-05 -1.36298664e-04
  -7.26453432e-03  9.99973603e-01]
 [ 6.05433402e-03 -2.69942162e-02  1.27528313e-02  9.99013423e-01
   3.23120126e-02  3.69879256e-04]
 [-9.82198935e-01  1.56454700e-01 -1.03312781e-01  1.14463687e-02
   1.62110700e-03  1.52596881e-05]
 [ 1.59310591e-04 -4.91894718e-03  3.11864824e-03 -3.25018102e-02
   9.99428355e-01  7.25592222e-03]
 [ 6.90939783e-02 -2.10615119e-01 -9.75101628e-01  6.26672294e-03
   2.19857585e-03  4.79234486e-05]
 [ 1.74569737e-01  9.64577791e-01 -1.95782843e-01  2.73038995e-02
   6.21788707e-03  7.82430481e-05]]
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 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]]
Expect an Identity Matrix =
```

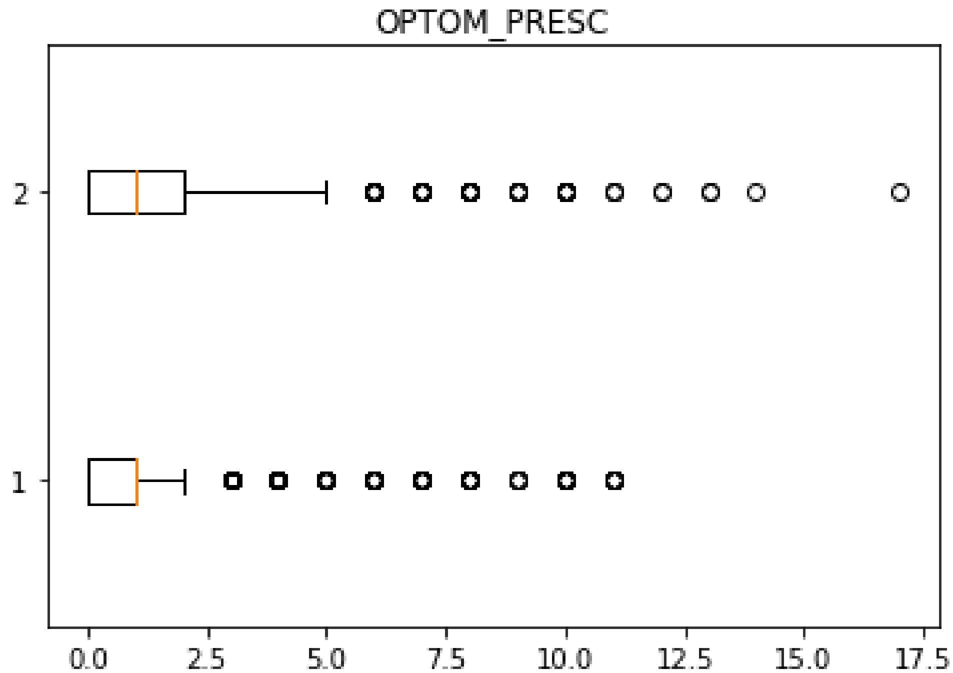
```
 [[ 1. -0.  0.  0.  0. -0.]
 [-0.  1. -0. -0. -0.  0.]
 [ 0. -0.  1.  0. -0. -0.]
 [ 0. -0.  0.  1.  0. -0.]
 [ 0. -0. -0.  0.  1. -0.]
 [-0.  0. -0. -0. -0.  1.]]
The orthonormalize x =
 [[-6.56324665e-04  9.39352141e-03  1.39590283e-02 -6.64664861e-03
   1.02081629e-02 -5.96859502e-03]
 [-7.75702220e-04  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]]
Also Expect an Identity Matrix =
 [[ 1. -0.  0. -0.  0. -0.]
 [-0.  1. -0.  0. -0. -0.]
 [ 0. -0.  1. -0. -0. -0.]
 [-0.  0. -0.  1. -0.  0.]
 [ 0. -0. -0. -0.  1.  0.]
 [-0. -0. -0.  0.  0.  1.]]


c:\users\suhas\pyvenevs\fall\lib\site-packages\matplotlib\figure.py:457: UserWarning: matplotl
  "matplotlib is currently using a non-GUI backend, "
```

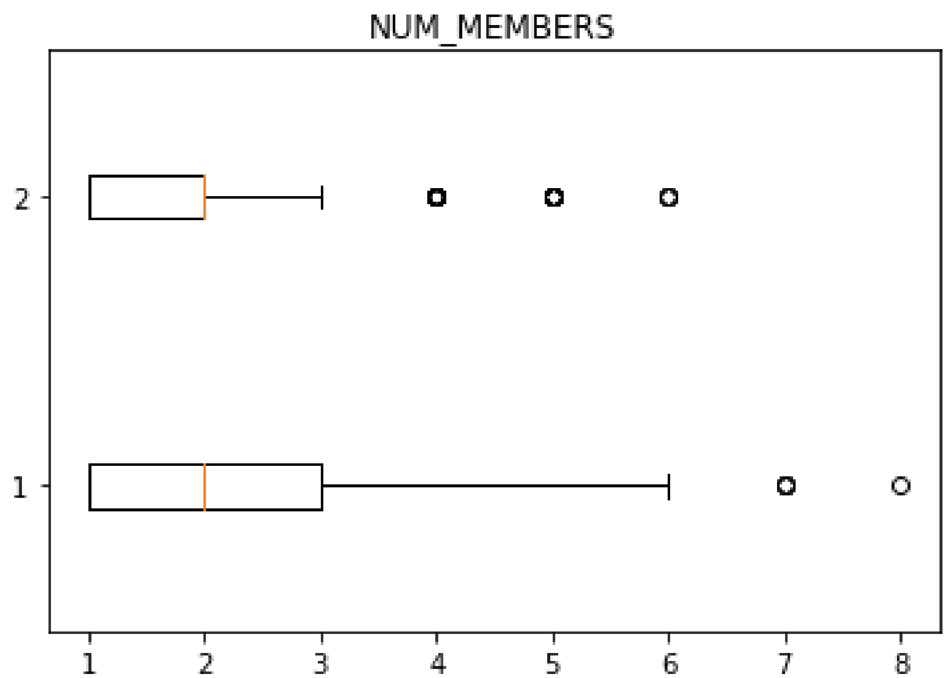

TOTAL_SPEND
2
1
0
20000
40000
60000
80000


DOCTOR_VISITS
2
1
0
5
10
15
20
25
30
35


NUM_CLAIMS
2
1
0
2
4
6
8
10
12
14


MEMBER_DURATION
2
1
0
100
200
300
400
500
600






```
In [15]: kNNSpec = kNN(n_neighbors = 5, algorithm = 'brute', metric = 'euclidean')
```

```
#trainData = df[['TOTAL_SPEND', 'DOCTOR_VISITS', 'NUM_CLAIMS', 'MEMBER_DURATION', 'OP
#trainData.describe()
trainData = transf_x
# Build nearest neighbors
nbrs = kNNSpec.fit(trainData)
distances, indices = nbrs.kneighbors(trainData)

# Find the nearest neighbors of these focal observations
#focal = [[7500, 15, 3, 127, 2, 2]]     # Mercedes-Benz_271
sample = [[7500, 15, 3, 127, 2, 2]]
#sample.reshape(1,-1)
transf_samp = np.matmul(sample,transf)

myNeighbors = nbrs.kneighbors(transf_samp, return_distance = False)
print("My Neighbors = \n", myNeighbors)

# Perform classification
target = df[['FRAUD']]


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

# See the classification result
class_result = nbrs.predict(trainData)
#print(class_result)

# See the classification probabilities
#class_prob = nbrs.predict_proba(trainData)
class_prob = nbrs.predict_proba(transf_samp)
print(class_prob)

accuracy = nbrs.score(trainData, target)
print(accuracy)
print(df.iloc[588,:])
print(df.iloc[2897,:])
print(df.iloc[1199,:])
print(df.iloc[1246,:])
print(df.iloc[886,:])
```

```
My Neighbors =
 [[ 588 2897 1199 1246  886]]


c:\users\suhas\pyvenevs\fall\lib\site-packages\ipykernel_launcher.py:24: DataConversionWarning


[[0. 1.]]
0.8778523489932886
```

```
CASE_ID             589
FRAUD                 1
TOTAL_SPEND        7500
DOCTOR_VISITS        15
NUM_CLAIMS            3
MEMBER_DURATION     127
OPTOM_PRESC           2
NUM_MEMBERS           2
Name: 588, dtype: int64
CASE_ID             2898
FRAUD                  1
TOTAL_SPEND        16000
DOCTOR_VISITS         18
NUM_CLAIMS             3
MEMBER_DURATION      146
OPTOM_PRESC            3
NUM_MEMBERS            2
Name: 2897, dtype: int64
CASE_ID             1200
FRAUD                  1
TOTAL_SPEND        10000
DOCTOR_VISITS         16
NUM_CLAIMS             3
MEMBER_DURATION      124
OPTOM_PRESC            2
NUM_MEMBERS            1
Name: 1199, dtype: int64
CASE_ID             1247
FRAUD                  1
TOTAL_SPEND        10200
DOCTOR_VISITS         13
NUM_CLAIMS             3
MEMBER_DURATION      119
OPTOM_PRESC            2
NUM_MEMBERS            3
Name: 1246, dtype: int64
CASE_ID             887
FRAUD                 1
TOTAL_SPEND        8900
DOCTOR_VISITS        22
NUM_CLAIMS            3
MEMBER_DURATION     166
OPTOM_PRESC           1
NUM_MEMBERS           2
Name: 886, dtype: int64
```