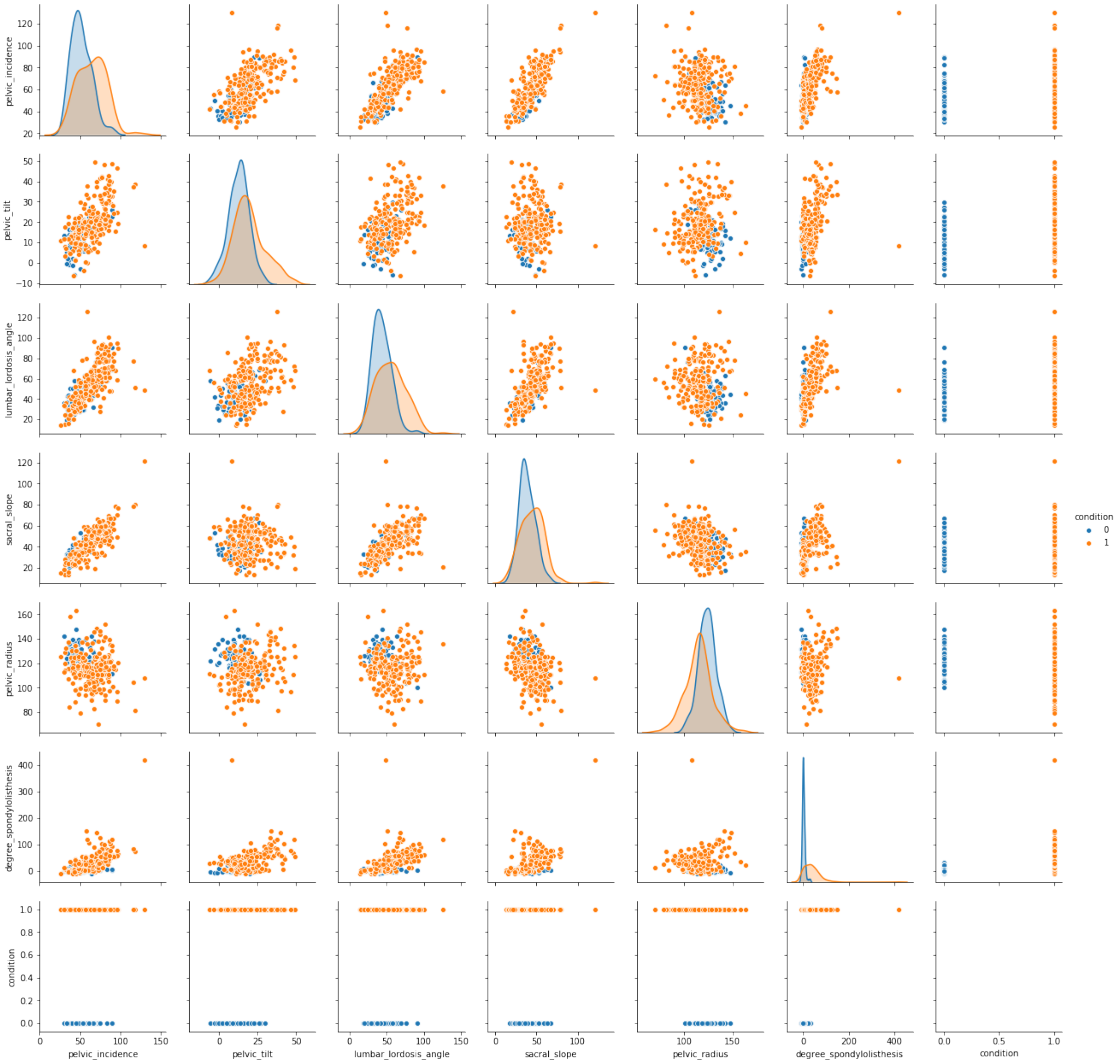
427_hw1

April 23, 2019

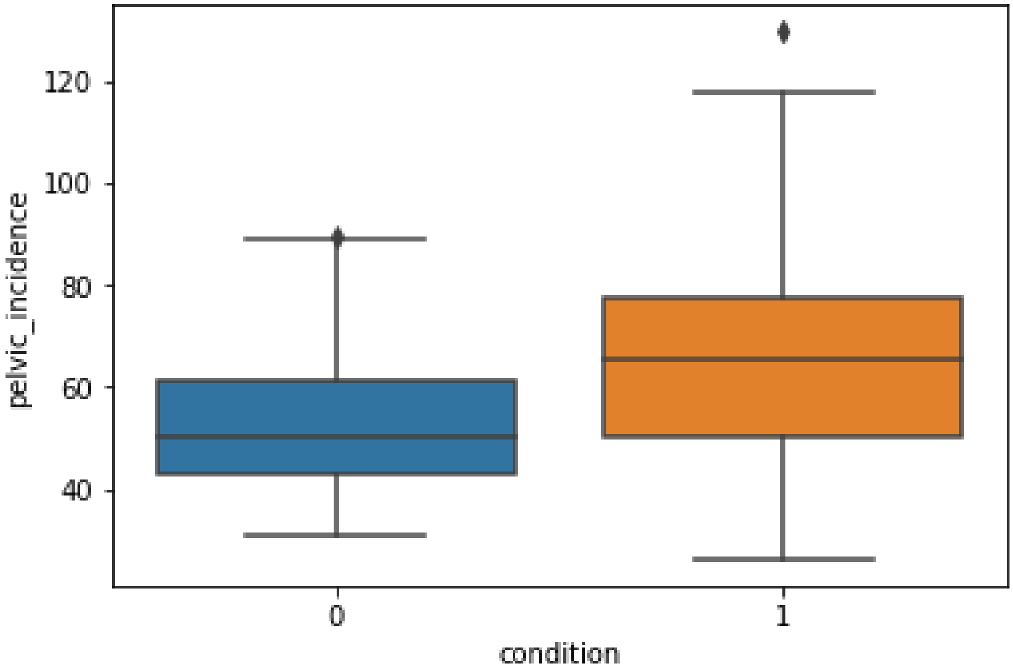
```
In [348]: import pandas as pd
          import matplotlib as plot

In [349]: mydata=pd.read_csv('vertebral.csv', delimiter = ' ')

In [350]: mydata = mydata.replace("AB", 1)
          mydata = mydata.replace("NO",0)

In [351]: incidence = mydata['pelvic_incidence']
          tilt = mydata['pelvic_tilt']
          angle = mydata['lumbar_lordosis_angle']
          slope = mydata['sacral_slope']
          radius = mydata['pelvic_radius']
          degree = mydata['degree_spondylolisthesis']
          condition = mydata['condition']

In [352]: import seaborn as sns
          sns.pairplot(mydata, hue = 'condition')
```

```
/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tu
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
/anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py:488: RuntimeWarning: in
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/anaconda3/lib/python3.7/site-packages/statsmodels/nonparametric/kdetools.py:34: RuntimeWarning
  FAC1 = 2*(np.pi*bw/RANGE)**2
/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:83: RuntimeWarning: invalid va
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

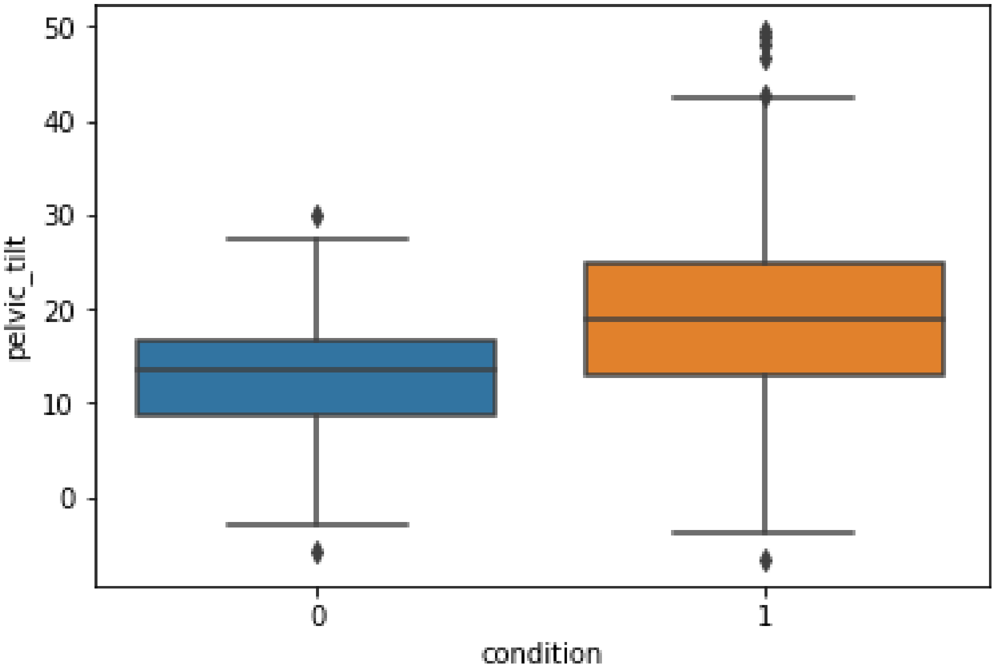
```
Out[352]: <seaborn.axisgrid.PairGrid at 0x1a200c8f28>
```



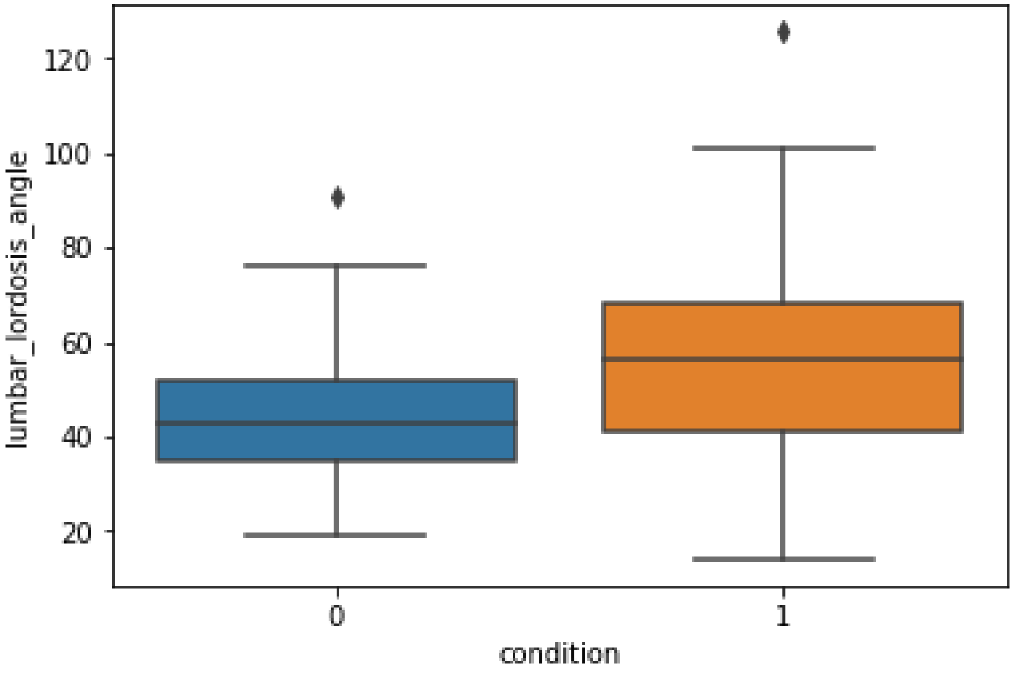
```
In [353]: #import matplotlib.pyplot as plt
          #fig, ax = plt.subplots(2, 3, sharex='col', sharey='row')
          x1 = sns.boxplot(x="condition", y="pelvic_incidence", data=mydata)
```



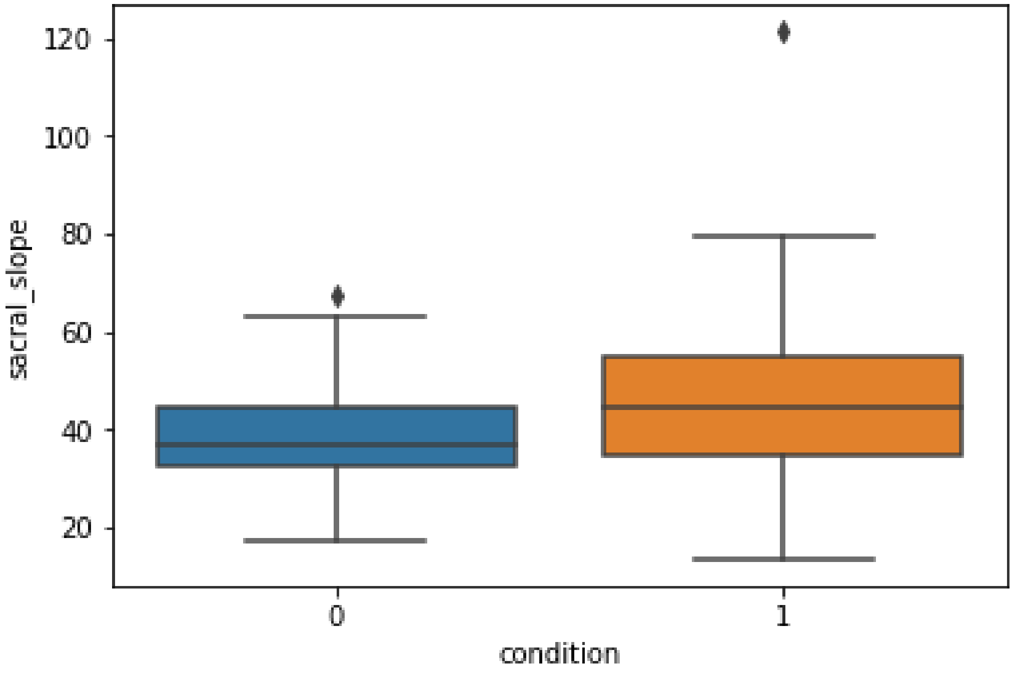
```
In [354]: x2 = sns.boxplot(x="condition", y="pelvic_tilt", data=mydata)
```



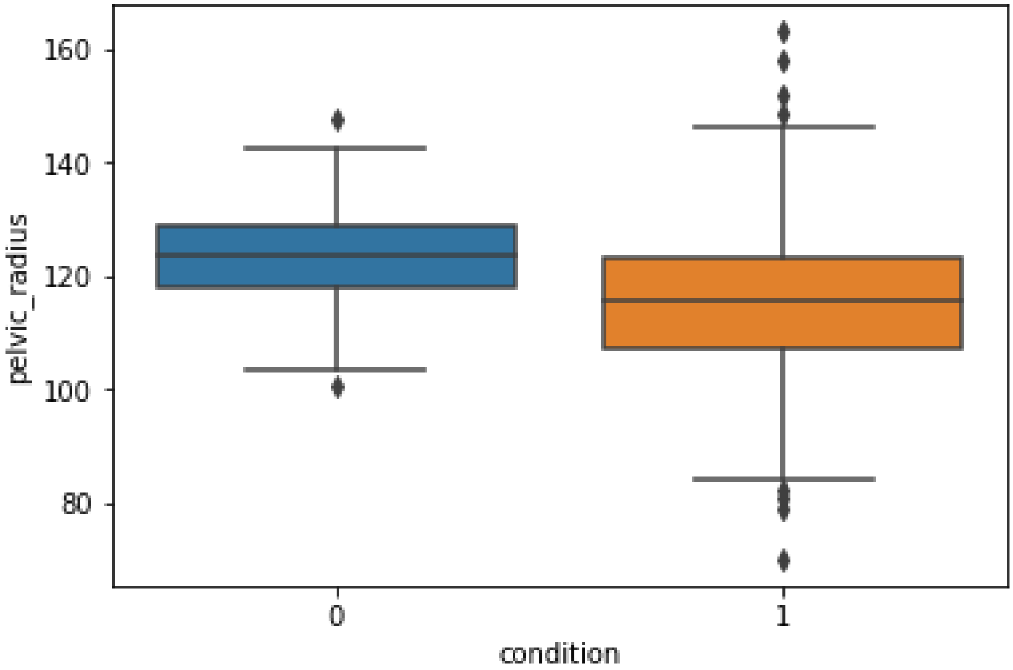
```
In [355]: x3 = sns.boxplot(x="condition", y="lumbar_lordosis_angle", data=mydata)
```



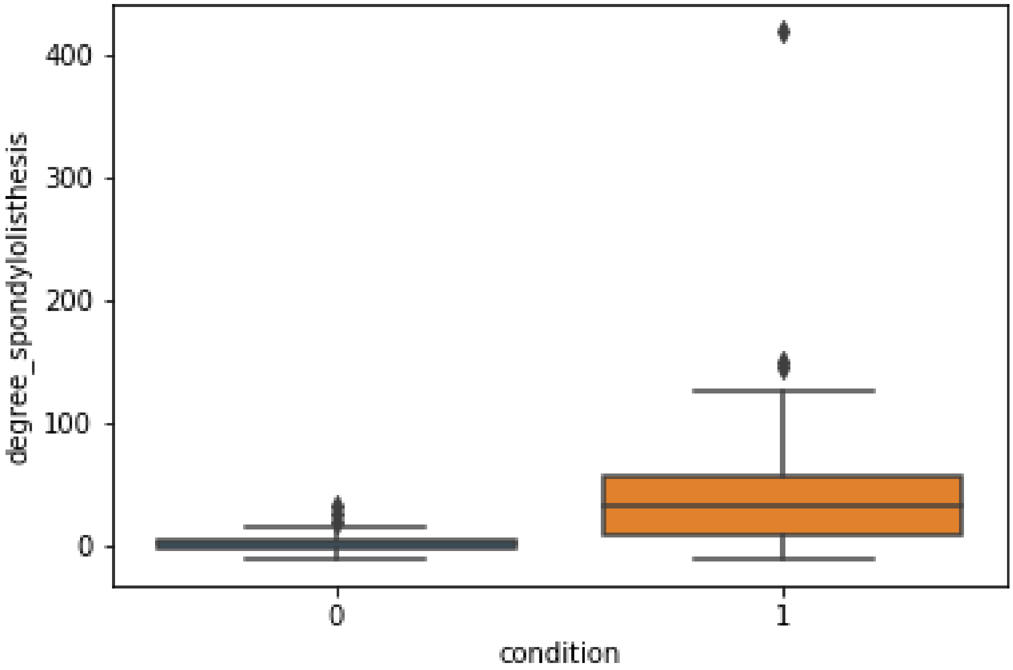
```
In [356]: x4 = sns.boxplot(x="condition", y="sacral_slope", data=mydata)
```



```
In [357]: x5 = sns.boxplot(x="condition", y="pelvic_radius", data=mydata)
```



```
In [358]: x6=sns.boxplot(x="condition", y="degree_spondylolisthesis", data=mydata)
```



```
In [359]: import numpy as np
          from sklearn.model_selection import train_test_split

          abnormal = mydata.loc[mydata.condition==1]
          normal = mydata.loc[mydata.condition ==0]

          nTrain_ab = 140
          nTrain_no = 70

          Xab = abnormal[["pelvic_incidence","pelvic_tilt","lumbar_lordosis_angle","sacral_slop
          Xab = np.array(Xab)
          Yab = abnormal["condition"]
          Xab_train, Xab_test, Yab_train, Yab_test=train_test_split(Xab,Yab, test_size=2/3, ran

          Xno = normal[["pelvic_incidence","pelvic_tilt","lumbar_lordosis_angle","sacral_slope"
          Xno = np.array(Xno)
          Yno = normal["condition"]
          Xno_train, Xno_test, Yno_train, Yno_test=train_test_split(Xno,Yno, test_size=0.7, ran
```

```
In [360]: def distance(instance1, instance2):
              instance1 = np.array(instance1)
              instance2 = np.array(instance2)

              return np.linalg.norm(instance1 - instance2)

          def get_neighbors(training_set,
                            labels,
                            test_instance,
                            k,
                            distance=distance):
              distances = []
              for index in range(len(training_set)):
                  dist = distance(test_instance, training_set[index])
                  distances.append((training_set[index], dist, labels[index]))
              distances.sort(key=lambda x: x[1])
              neighbors = distances[:k]
              return neighbors
```

```
In [361]: Xtrain = np.concatenate((Xab_train,Xno_train))
          Xtest = np.concatenate((Xab_test,Xno_test))

          Ytrain = np.concatenate((Yab_train,Yno_train))
          Ytest = np.concatenate((Yab_test,Yno_test))

          for i in range(5):
              neighbors = get_neighbors(Xtrain, Ytrain, Xtest[i], 3, distance=distance)
              print(i, Xtest[i], Ytest[i], neighbors)

0 [ 56.03  16.3   62.28  39.73 114.02  -2.33] 1 [(array([ 65.61,  23.14,  62.58,  42.47, 124.13
1 [ 85.68  38.65  82.68  47.03 120.84  61.96] 1 [(array([ 92.03,  35.39,  77.42,  56.63, 115.72
2 [ 74.43  41.56  27.7   32.88 107.95   5.  ] 1 [(array([ 69.3 ,  24.65,  44.31,  44.64, 101.87
3 [ 56.54  14.38  44.99  42.16 101.72  25.77] 1 [(array([56.67, 13.46, 43.77, 43.21, 93.69, 21
4 [ 31.48   7.83  24.28  23.66 113.83   4.39] 1 [(array([ 39.06,  10.06,  25.02,  29.  , 114.4
```

```
In [362]: from collections import Counter
          def vote(neighbors):
              class_counter = Counter()
              for neighbor in neighbors:
                  class_counter[neighbor[2]] += 1
              return class_counter.most_common(1)[0][0]

          n_training_samples = nTrain_ab + nTrain_no


          for i in range(n_training_samples):
              neighbors = get_neighbors(Xtrain,
                                        Ytrain,
                                        Xtest[i],
                                        3,
                                        distance=distance)
              print("index: ", i,
                    ", result of vote: ", vote(neighbors),
                    ", label: ", Ytest[i],
                    ", data: ", Xtest[i])

index:  0 , result of vote:  0 , label:  1 , data:  [ 56.03  16.3   62.28  39.73 114.02  -2.33]
index:  1 , result of vote:  1 , label:  1 , data:  [ 85.68  38.65  82.68  47.03 120.84  61.96]
index:  2 , result of vote:  1 , label:  1 , data:  [ 74.43  41.56  27.7   32.88 107.95   5.  ]
index:  3 , result of vote:  1 , label:  1 , data:  [ 56.54  14.38  44.99  42.16 101.72  25.77]
index:  4 , result of vote:  1 , label:  1 , data:  [ 31.48   7.83  24.28  23.66 113.83   4.39]
index:  5 , result of vote:  1 , label:  1 , data:  [ 71.19  23.9   43.7   47.29 119.86  27.28]
index:  6 , result of vote:  0 , label:  1 , data:  [ 38.66  12.99  40.    25.68 124.91   2.7 ]
index:  7 , result of vote:  1 , label:  1 , data:  [72.34 16.42 59.87 55.92 70.08 12.07]
index:  8 , result of vote:  1 , label:  1 , data:  [ 80.11  33.94  85.1   46.17 125.59 100.29]
index:  9 , result of vote:  1 , label:  1 , data:  [95.38 24.82 95.16 70.56 89.31 57.66]
index:  10 , result of vote:  1 , label:  1 , data:  [ 57.52  33.65  50.91  23.88 140.98 148.75
index:  11 , result of vote:  0 , label:  1 , data:  [ 49.71   9.65  28.32  40.06 108.17   7.92
index:  12 , result of vote:  1 , label:  1 , data:  [ 41.35  16.58  30.71  24.78 113.27  -4.5
index:  13 , result of vote:  0 , label:  1 , data:  [ 38.7   13.44  31.    25.25 123.16   1.43
index:  14 , result of vote:  1 , label:  1 , data:  [ 48.11  14.93  35.56  33.18 124.06   7.95
index:  15 , result of vote:  1 , label:  1 , data:  [66.8  14.55 72.08 52.25 82.46 41.69]
index:  16 , result of vote:  1 , label:  1 , data:  [ 61.41  25.38  39.1   36.03 103.4   21.84
```

```
index:  17 , result of vote:  1 , label:  1 , data:  [75.65 19.34 64.15 56.31 95.9  69.55]
index:  18 , result of vote:  1 , label:  1 , data:  [ 55.84  28.85  47.69  27.   123.31   2.81
index:  19 , result of vote:  1 , label:  1 , data:  [ 59.6   32.    46.56  27.6  119.33   1.47
index:  20 , result of vote:  0 , label:  1 , data:  [ 43.2   19.66  35.    23.54 124.85  -2.92
index:  21 , result of vote:  1 , label:  1 , data:  [46.39 11.08 32.14 35.31 98.77  6.39]
index:  22 , result of vote:  1 , label:  1 , data:  [ 78.43  33.43  76.28  45.   138.55  77.16
index:  23 , result of vote:  1 , label:  1 , data:  [80.82 19.24 61.64 61.58 89.47 44.17]
index:  24 , result of vote:  1 , label:  1 , data:  [ 35.49  11.7   15.59  23.79 106.94  -3.46
index:  25 , result of vote:  1 , label:  1 , data:  [81.11 20.69 60.69 60.42 94.02 40.51]
index:  26 , result of vote:  1 , label:  1 , data:  [ 31.23  17.72  15.5   13.52 120.06   0.5
index:  27 , result of vote:  1 , label:  1 , data:  [ 85.64  42.69  78.75  42.95 105.14  42.89
index:  28 , result of vote:  0 , label:  1 , data:  [ 66.88  24.89  49.28  41.99 113.48  -2.01
index:  29 , result of vote:  1 , label:  1 , data:  [ 55.51  20.1   44.    35.42 122.65  34.55
index:  30 , result of vote:  1 , label:  1 , data:  [ 65.01  27.6   50.95  37.41 116.58   7.02
index:  31 , result of vote:  1 , label:  1 , data:  [ 70.4   13.47  61.2   56.93 102.34  25.54
index:  32 , result of vote:  1 , label:  1 , data:  [ 80.43  17.    66.54  63.43 116.44  57.78
index:  33 , result of vote:  1 , label:  1 , data:  [ 44.55  21.93  26.79  22.62 111.07   2.65
index:  34 , result of vote:  0 , label:  1 , data:  [ 63.83  20.36  54.55  43.47 112.31  -0.62
index:  35 , result of vote:  1 , label:  1 , data:  [ 58.52  13.92  41.47  44.6  115.51  30.39
index:  36 , result of vote:  1 , label:  1 , data:  [ 57.04   0.35  49.2   56.69 103.05  52.17
index:  37 , result of vote:  1 , label:  1 , data:  [ 56.61  16.8   42.    39.81 127.29  24.02
index:  38 , result of vote:  1 , label:  1 , data:  [ 43.58  16.51  47.    27.07 109.27   8.99
index:  39 , result of vote:  1 , label:  1 , data:  [ 57.29  15.15  64.    42.14 116.74  30.34
index:  40 , result of vote:  1 , label:  1 , data:  [ 44.53   9.43  52.    35.1  134.71  29.11
index:  41 , result of vote:  1 , label:  1 , data:  [48.26 16.42 36.33 31.84 94.88 28.34]
index:  42 , result of vote:  1 , label:  1 , data:  [65.01  9.84 57.74 55.18 94.74 49.7 ]
index:  43 , result of vote:  1 , label:  1 , data:  [ 48.03   3.97  58.34  44.06 125.35  35.
index:  44 , result of vote:  1 , label:  1 , data:  [ 94.17  15.38  67.71  78.79 114.89  53.26
index:  45 , result of vote:  1 , label:  1 , data:  [ 76.33  42.4   57.2   33.93 124.27  50.13
index:  46 , result of vote:  1 , label:  1 , data:  [ 70.25  10.34  76.37  59.91 119.24  32.67
index:  47 , result of vote:  1 , label:  1 , data:  [ 91.47  24.51  84.62  66.96 117.31  52.62
index:  48 , result of vote:  1 , label:  1 , data:  [ 63.9   13.71  62.12  50.19 114.13  41.42
index:  49 , result of vote:  1 , label:  1 , data:  [ 63.36  20.02  67.5   43.34 131.    37.56
index:  50 , result of vote:  1 , label:  1 , data:  [ 65.67  10.54  56.49  55.12 109.16  53.93
index:  51 , result of vote:  1 , label:  1 , data:  [43.72  9.81 52.   33.91 88.43 40.88]
index:  52 , result of vote:  1 , label:  1 , data:  [ 80.07  48.07  52.4   32.01 110.71  67.73
index:  53 , result of vote:  1 , label:  1 , data:  [ 42.02  -6.55  67.9   48.58 111.59  27.34
index:  54 , result of vote:  1 , label:  1 , data:  [ 58.6   -0.26  51.5   58.86 102.04  28.06
index:  55 , result of vote:  1 , label:  1 , data:  [ 57.3   24.19  47.    33.11 116.81   5.77
index:  56 , result of vote:  1 , label:  1 , data:  [ 56.99   6.87  57.01  50.12 109.98  36.81
index:  57 , result of vote:  1 , label:  1 , data:  [64.62 15.23 67.63 49.4  90.3  31.33]
index:  58 , result of vote:  1 , label:  1 , data:  [ 41.73  12.25  30.12  29.48 116.59  -1.24
index:  59 , result of vote:  1 , label:  1 , data:  [ 72.64  18.93  68.    53.71 116.96  25.38
index:  60 , result of vote:  1 , label:  1 , data:  [ 81.66  28.75  58.23  52.91 114.77  30.61
index:  61 , result of vote:  1 , label:  1 , data:  [ 52.42  19.01  35.87  33.41 116.56   1.69
index:  62 , result of vote:  1 , label:  1 , data:  [ 70.48  12.49  62.42  57.99 114.19  56.9
index:  63 , result of vote:  1 , label:  1 , data:  [ 74.85  13.91  62.69  60.95 115.21  33.17
index:  64 , result of vote:  1 , label:  1 , data:  [ 43.35   7.47  28.07  35.88 112.78   5.75
```

```
index:  65 , result of vote:  1 , label:  1 , data:  [ 37.9    4.48  24.71  33.42 157.85  33.6
index:  66 , result of vote:  1 , label:  1 , data:  [47.66 13.28 36.68 34.38 98.25  6.27]
index:  67 , result of vote:  1 , label:  1 , data:  [ 87.68  20.37  93.82  67.31 120.94  76.7
index:  68 , result of vote:  1 , label:  1 , data:  [ 30.15  11.92  34.    18.23 112.68  11.4
index:  69 , result of vote:  1 , label:  1 , data:  [ 77.66  22.43  93.89  55.22 123.06  61.2
index:  70 , result of vote:  1 , label:  1 , data:  [ 55.08  -3.76  56.    58.84 109.92  31.7
index:  71 , result of vote:  1 , label:  1 , data:  [ 67.41  17.44  60.14  49.97 111.12  33.1
index:  72 , result of vote:  1 , label:  1 , data:  [ 86.04  38.75  47.87  47.29 122.09  61.9
index:  73 , result of vote:  0 , label:  1 , data:  [ 53.85  19.23  32.78  34.62 121.67   5.3
index:  74 , result of vote:  0 , label:  1 , data:  [ 31.28   3.14  32.56  28.13 129.01   3.6
index:  75 , result of vote:  0 , label:  1 , data:  [ 40.25  13.92  25.12  26.33 130.33   2.2
index:  76 , result of vote:  1 , label:  1 , data:  [ 41.19   5.79  42.87  35.39 103.35  27.6
index:  77 , result of vote:  0 , label:  1 , data:  [ 78.4   14.04  79.69  64.36 104.73  12.3
index:  78 , result of vote:  0 , label:  1 , data:  [ 40.56  17.98  34.    22.58 121.05  -1.5
index:  79 , result of vote:  1 , label:  1 , data:  [44.25  1.1  38.   43.15 98.27 23.91]
index:  80 , result of vote:  1 , label:  1 , data:  [ 78.49  22.18  60.    56.31 118.53  27.3
index:  81 , result of vote:  1 , label:  1 , data:  [ 60.63  20.6   64.54  40.03 117.23 104.8
index:  82 , result of vote:  1 , label:  1 , data:  [ 41.17  17.32  33.47  23.85 116.38  -9.5
index:  83 , result of vote:  1 , label:  1 , data:  [ 74.72  19.76  82.74  54.96 109.36  33.3
index:  84 , result of vote:  1 , label:  1 , data:  [ 60.75  15.75  43.2   45.   113.05  31.6
index:  85 , result of vote:  1 , label:  1 , data:  [ 82.41  29.28  77.05  53.13 117.04  62.7
index:  86 , result of vote:  1 , label:  1 , data:  [ 58.1   14.84  79.65  43.26 113.59  50.2
index:  87 , result of vote:  1 , label:  1 , data:  [56.56  8.96 52.58 47.6  98.78 50.7 ]
index:  88 , result of vote:  1 , label:  1 , data:  [50.07  9.12 32.17 40.95 99.71 26.77]
index:  89 , result of vote:  1 , label:  1 , data:  [ 48.92  19.96  40.26  28.95 119.32   8.0
index:  90 , result of vote:  1 , label:  1 , data:  [ 72.05  24.7   79.87  47.35 107.17  56.4
index:  91 , result of vote:  1 , label:  1 , data:  [ 63.17   6.33  63.    56.84 110.64  42.6
index:  92 , result of vote:  1 , label:  1 , data:  [ 69.63  21.12  52.77  48.5  116.8   54.8
index:  93 , result of vote:  1 , label:  1 , data:  [115.92  37.52  76.8   78.41 104.7   81.2
index:  94 , result of vote:  1 , label:  1 , data:  [ 65.76   9.83  50.82  55.92 104.39  39.3
index:  95 , result of vote:  1 , label:  1 , data:  [ 74.01  21.12  57.38  52.88 120.21  74.5
index:  96 , result of vote:  1 , label:  1 , data:  [ 76.31  41.93  93.28  34.38 132.27 101.2
index:  97 , result of vote:  1 , label:  1 , data:  [ 71.    37.52  84.54  33.49 125.16  67.7
index:  98 , result of vote:  1 , label:  1 , data:  [ 53.57  20.46  33.1   33.11 110.97   7.0
index:  99 , result of vote:  1 , label:  1 , data:  [81.08 21.26 78.77 59.83 90.07 49.16]
index:  100 , result of vote:  1 , label:  1 , data:  [ 88.62  29.09  47.56  59.53 121.76  51.
index:  101 , result of vote:  1 , label:  1 , data:  [50.83  9.06 56.3  41.76 79.   23.04]
index:  102 , result of vote:  1 , label:  1 , data:  [ 60.42   5.27  59.81  55.15 109.03  30.
index:  103 , result of vote:  1 , label:  1 , data:  [ 50.91  23.02  47.    27.9  117.42  -2.
index:  104 , result of vote:  1 , label:  1 , data:  [ 80.99  36.84  86.96  44.14 141.09  85.
index:  105 , result of vote:  1 , label:  1 , data:  [ 49.78   6.47  53.    43.32 110.86  25.
index:  106 , result of vote:  0 , label:  1 , data:  [ 32.09   6.99  36.    25.1  132.26   6.
index:  107 , result of vote:  1 , label:  1 , data:  [ 64.81  15.17  58.84  49.64 111.68  21.
index:  108 , result of vote:  1 , label:  1 , data:  [ 71.24   5.27  86.    65.97 110.7   38.
index:  109 , result of vote:  1 , label:  1 , data:  [118.14  38.45  50.84  79.7   81.02  74.
index:  110 , result of vote:  0 , label:  1 , data:  [ 65.54  24.16  45.78  41.38 136.44  16.
index:  111 , result of vote:  1 , label:  1 , data:  [86.47 40.3  61.14 46.17 97.4  55.75]
index:  112 , result of vote:  1 , label:  1 , data:  [ 74.09  18.82  76.03  55.27 128.41  73.
```

```
index:  113 , result of vote:  1 , label:  1 , data:  [ 74.72  14.32  32.5   60.4  107.18  37.0
index:  114 , result of vote:  1 , label:  1 , data:  [ 75.44  31.54  89.6   43.9  106.83  54.9
index:  115 , result of vote:  1 , label:  1 , data:  [ 70.22  39.82  68.12  30.4  148.53 145.3
index:  116 , result of vote:  1 , label:  1 , data:  [ 54.92  21.06  42.2   33.86 125.21   2.4
index:  117 , result of vote:  1 , label:  1 , data:  [ 70.95  20.16  62.86  50.79 116.18  32.5
index:  118 , result of vote:  1 , label:  1 , data:  [67.26  7.19 51.7  60.07 97.8  42.14]
index:  119 , result of vote:  0 , label:  1 , data:  [ 68.83  22.22  50.09  46.61 105.99  -3.5
index:  120 , result of vote:  1 , label:  1 , data:  [ 72.56  17.39  52.    55.18 119.19  32.1
index:  121 , result of vote:  0 , label:  1 , data:  [ 45.44   9.91  45.    35.54 163.07  20.3
index:  122 , result of vote:  1 , label:  1 , data:  [64.27 12.51 68.7  51.77 95.25 39.41]
index:  123 , result of vote:  0 , label:  1 , data:  [ 53.43  15.86  37.17  37.57 120.57   5.9
index:  124 , result of vote:  1 , label:  1 , data:  [ 67.03  13.28  66.15  53.75 100.72  33.9
index:  125 , result of vote:  1 , label:  1 , data:  [ 54.74  12.1   41.    42.65 117.64  40.3
index:  126 , result of vote:  0 , label:  1 , data:  [63.03 22.55 39.61 40.48 98.67 -0.25]
index:  127 , result of vote:  1 , label:  1 , data:  [ 53.94   9.31  43.1   44.64 124.4   25.0
index:  128 , result of vote:  1 , label:  1 , data:  [58.78  7.67 53.34 51.12 98.5  51.58]
index:  129 , result of vote:  1 , label:  1 , data:  [69.76 19.28 48.5  50.48 96.49 51.17]
index:  130 , result of vote:  1 , label:  1 , data:  [ 83.93  41.29  62.    42.65 115.01  26.5
index:  131 , result of vote:  1 , label:  1 , data:  [ 48.33  22.23  36.18  26.1  117.38   6.4
index:  132 , result of vote:  1 , label:  1 , data:  [ 72.22  23.08  91.    49.14 137.74  56.8
index:  133 , result of vote:  1 , label:  1 , data:  [ 85.35  15.84  71.67  69.51 124.42  76.0
index:  134 , result of vote:  1 , label:  1 , data:  [ 63.4   14.12  48.14  49.29 111.92  31.7
index:  135 , result of vote:  1 , label:  1 , data:  [ 44.91  10.22  44.63  34.7  130.08  37.3
index:  136 , result of vote:  1 , label:  1 , data:  [ 67.51  33.28  96.28  34.24 145.6   88.3
index:  137 , result of vote:  1 , label:  1 , data:  [ 49.71  13.04  31.33  36.67 108.65  -7.8
index:  138 , result of vote:  1 , label:  1 , data:  [ 44.94  17.44  27.78  27.49 117.98   5.5
index:  139 , result of vote:  1 , label:  1 , data:  [ 88.02  39.84  81.77  48.18 116.6   56.7
index:  140 , result of vote:  1 , label:  0 , data:  [ 36.42  13.88  20.24  22.54 126.08   0.1
index:  141 , result of vote:  0 , label:  0 , data:  [ 46.24  10.06  37.    36.17 128.06  -5.1
index:  142 , result of vote:  0 , label:  0 , data:  [ 34.65   7.51  43.    27.14 123.99  -4.0
index:  143 , result of vote:  1 , label:  0 , data:  [ 51.08  14.21  35.95  36.87 115.8    6.9
index:  144 , result of vote:  0 , label:  0 , data:  [ 34.38   2.06  32.39  32.32 128.3   -3.3
index:  145 , result of vote:  0 , label:  0 , data:  [ 61.54  19.68  52.89  41.86 118.69   4.8
index:  146 , result of vote:  1 , label:  0 , data:  [ 46.64  15.85  40.    30.78 119.38   9.0
index:  147 , result of vote:  0 , label:  0 , data:  [ 34.76   2.63  29.5   32.12 127.14  -0.4
index:  148 , result of vote:  0 , label:  0 , data:  [ 48.8   18.02  52.    30.78 139.15  10.4
index:  149 , result of vote:  0 , label:  0 , data:  [ 67.29  16.72  51.    50.57 137.59   4.9
index:  150 , result of vote:  0 , label:  0 , data:  [ 54.6   21.49  29.36  33.11 118.34  -1.4
index:  151 , result of vote:  1 , label:  0 , data:  [ 50.75  20.24  37.    30.52 122.34   2.2
index:  152 , result of vote:  1 , label:  0 , data:  [ 46.37  10.22  42.7   36.16 121.25  -0.5
index:  153 , result of vote:  0 , label:  0 , data:  [ 42.52  14.38  25.32  28.14 128.91   0.7
index:  154 , result of vote:  0 , label:  0 , data:  [ 74.98  14.92  53.73  60.05 105.65   1.5
index:  155 , result of vote:  0 , label:  0 , data:  [ 39.09   5.54  26.93  33.55 131.58  -0.7
index:  156 , result of vote:  1 , label:  0 , data:  [ 51.53  13.52  35.    38.01 126.72  13.9
index:  157 , result of vote:  0 , label:  0 , data:  [ 72.96  19.58  61.01  53.38 111.23   0.8
index:  158 , result of vote:  1 , label:  0 , data:  [ 74.57  15.72  58.62  58.84 105.42   0.6
index:  159 , result of vote:  1 , label:  0 , data:  [ 44.49  21.79  31.47  22.7  113.78  -0.2
index:  160 , result of vote:  0 , label:  0 , data:  [ 44.36   8.95  46.9   35.42 129.22   4.9
```

```
index:  161 , result of vote:  0 , label:  0 , data:  [ 49.    13.11  51.87  35.88 126.4    0.5
index:  162 , result of vote:  0 , label:  0 , data:  [ 61.45  22.69  46.17  38.75 125.67  -2.7
index:  163 , result of vote:  1 , label:  0 , data:  [ 39.36   7.01  37.    32.35 117.82   1.9
index:  164 , result of vote:  0 , label:  0 , data:  [ 43.44  10.1   36.03  33.34 137.44  -3.1
index:  165 , result of vote:  0 , label:  0 , data:  [ 47.81  10.69  54.    37.12 125.39  -0.4
index:  166 , result of vote:  0 , label:  0 , data:  [ 45.08  12.31  44.58  32.77 147.89  -8.9
index:  167 , result of vote:  0 , label:  0 , data:  [ 62.14  13.96  58.    48.18 133.28   4.9
index:  168 , result of vote:  0 , label:  0 , data:  [ 54.75   9.75  48.    45.   123.04   8.2
index:  169 , result of vote:  1 , label:  0 , data:  [ 47.32   8.57  35.56  38.75 120.58   1.6
index:  170 , result of vote:  0 , label:  0 , data:  [ 65.76  13.21  44.    52.55 129.39  -1.9
index:  171 , result of vote:  0 , label:  0 , data:  [ 33.84   5.07  36.64  28.77 123.95  -0.2
index:  172 , result of vote:  0 , label:  0 , data:  [ 54.5    6.82  47.    47.68 111.79  -4.4
index:  173 , result of vote:  1 , label:  0 , data:  [ 47.9   13.62  36.    34.29 117.45  -4.2
index:  174 , result of vote:  0 , label:  0 , data:  [ 66.51  20.9   31.73  45.61 128.9    1.5
index:  175 , result of vote:  0 , label:  0 , data:  [ 36.16  -0.81  33.63  36.97 135.94  -2.0
index:  176 , result of vote:  0 , label:  0 , data:  [ 40.35  10.19  37.97  30.15 128.01   0.4
index:  177 , result of vote:  1 , label:  0 , data:  [ 37.14  16.48  24.    20.66 125.01   7.3
index:  178 , result of vote:  0 , label:  0 , data:  [ 40.68   9.15  31.02  31.53 139.12  -2.5
index:  179 , result of vote:  1 , label:  0 , data:  [ 63.03  27.34  51.61  35.69 114.51   7.4
index:  180 , result of vote:  0 , label:  0 , data:  [ 39.66  16.21  36.67  23.45 131.92  -4.9
index:  181 , result of vote:  0 , label:  0 , data:  [ 33.79   3.68  25.5   30.11 128.33  -1.7
index:  182 , result of vote:  0 , label:  0 , data:  [ 33.04  -0.32  19.07  33.37 120.39   9.3
index:  183 , result of vote:  0 , label:  0 , data:  [ 59.73   7.72  55.34  52.   125.17   3.2
index:  184 , result of vote:  0 , label:  0 , data:  [ 48.9    5.59  55.5   43.32 137.11  19.8
index:  185 , result of vote:  0 , label:  0 , data:  [ 40.75   1.84  50.    38.91 139.25   0.6
index:  186 , result of vote:  1 , label:  0 , data:  [ 67.54  14.66  58.    52.88 123.63  25.9
index:  187 , result of vote:  1 , label:  0 , data:  [ 89.01  26.08  69.02  62.94 111.48   6.0
index:  188 , result of vote:  0 , label:  0 , data:  [ 61.82  13.6   64.    48.22 121.78   1.3
index:  189 , result of vote:  1 , label:  0 , data:  [ 45.25   8.69  41.58  36.56 118.55   0.2
index:  190 , result of vote:  1 , label:  0 , data:  [ 63.79  21.35  66.    42.45 119.55  12.3
index:  191 , result of vote:  0 , label:  0 , data:  [ 48.17   9.59  39.71  38.58 135.62   5.3
index:  192 , result of vote:  1 , label:  0 , data:  [ 37.73   9.39  42.    28.35 135.74  13.6
index:  193 , result of vote:  0 , label:  0 , data:  [ 45.7   10.66  42.58  35.04 130.18  -3.3
index:  194 , result of vote:  0 , label:  0 , data:  [ 57.15  16.49  42.84  40.66 113.81   5.0
index:  195 , result of vote:  0 , label:  0 , data:  [ 63.62  16.93  49.35  46.68 117.09  -0.3
index:  196 , result of vote:  0 , label:  0 , data:  [ 53.91  12.94  39.    40.97 118.19   5.0
index:  197 , result of vote:  0 , label:  0 , data:  [ 45.58  18.76  33.77  26.82 116.8    3.1
index:  198 , result of vote:  0 , label:  0 , data:  [ 48.32  17.45  48.    30.87 128.98  -0.9
index:  199 , result of vote:  0 , label:  0 , data:  [ 59.17  14.56  43.2   44.6  121.04   2.8
index:  200 , result of vote:  1 , label:  0 , data:  [ 41.65   8.84  36.03  32.81 116.56  -6.0
index:  201 , result of vote:  0 , label:  0 , data:  [ 51.33  13.63  33.26  37.69 131.31   1.7
index:  202 , result of vote:  1 , label:  0 , data:  [ 50.09  13.43  34.46  36.66 119.13   3.0
index:  203 , result of vote:  0 , label:  0 , data:  [ 42.92  -5.85  58.    48.76 121.61  -3.3
index:  204 , result of vote:  0 , label:  0 , data:  [ 50.16  -2.97  42.    53.13 131.8   -8.2
index:  205 , result of vote:  1 , label:  0 , data:  [ 82.91  29.89  58.25  53.01 110.71   6.0
index:  206 , result of vote:  0 , label:  0 , data:  [ 51.62  15.97  35.    35.66 129.39   1.0
index:  207 , result of vote:  0 , label:  0 , data:  [ 51.31   8.88  57.    42.44 126.47  -2.1
index:  208 , result of vote:  0 , label:  0 , data:  [ 38.51  16.96  35.11  21.54 127.63   7.9
```

```
index:  209 , result of vote:  0 , label:  0 , data:  [ 50.68   6.46  35.     44.22 116.59  -0.
```

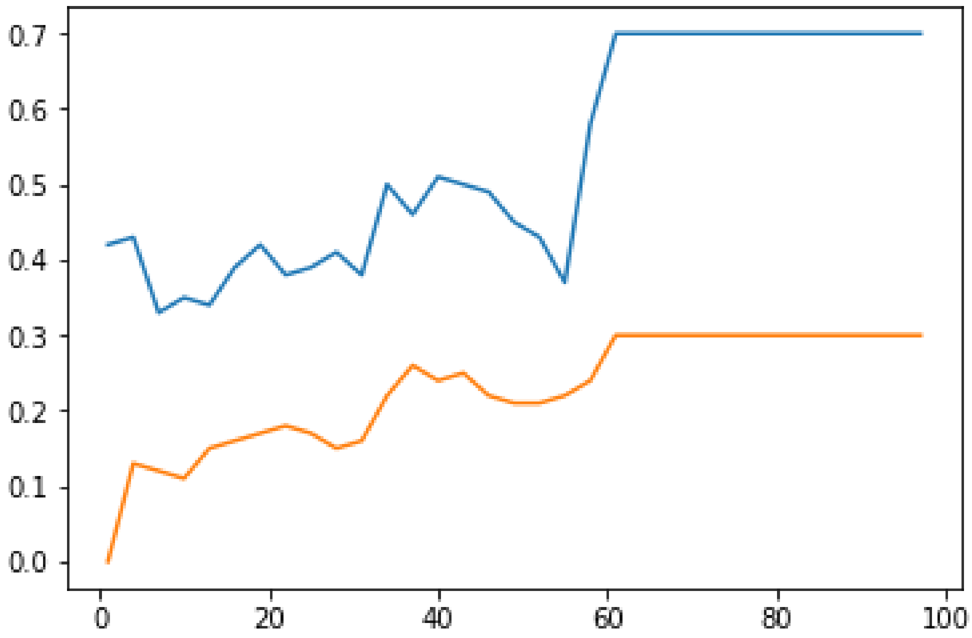
```
In [363]: #k_range = M[::-1]
          #accuracy=[]
          #for j in range(70):
          #    for i in range(n_training_samples):
          #        neighbors = get_neighbors(Xtrain,
          #                                  Ytrain,
          #                                  Xtest[i],
          #                                  k_range[j],
          #                                  distance=distance)
          #        if vote(neighbors) == Ytest[i]:
          #
          #
          #        print("k value", k_range[j],
          #              ", index: ", i,
          #              ", result of vote: ", vote(neighbors),
          #              ", label: ", Ytest[i],
          #              ", data: ", Xtest[i])
```

```
In [364]: from sklearn import preprocessing,neighbors
          from sklearn.neighbors import KNeighborsClassifier
          from sklearn.metrics import confusion_matrix
          from sklearn.metrics import f1_score
```

```
In [365]: range_k=[]
          test_error=[]
          train_error=[]

          for i in range(1,len(Xtrain),3):
              range_k.append(i)
              clf = neighbors.KNeighborsClassifier(n_neighbors = i)
              clf.fit(Xtrain,Ytrain)
              y_pred_train=clf.predict(Xtrain)
              y_pred_test=clf.predict(Xtest)
              cm_train=confusion_matrix(Ytrain,y_pred_train)
              cm_test=confusion_matrix(Ytest,y_pred_test)
              trainerror=((cm_train[0,1]+cm_train[1,0])/len(Xtrain))
              testerror=((cm_test[0,1]+cm_test[1,0])/len(Xtrain))
              train_error.append(trainerror)
              test_error.append(testerror)
```

```
In [366]: plt.figure
          plt.plot(range_k,test_error)
          plt.plot(range_k,train_error)
```

```
Out[366]: [<matplotlib.lines.Line2D at 0x1a282c66d8>]
```

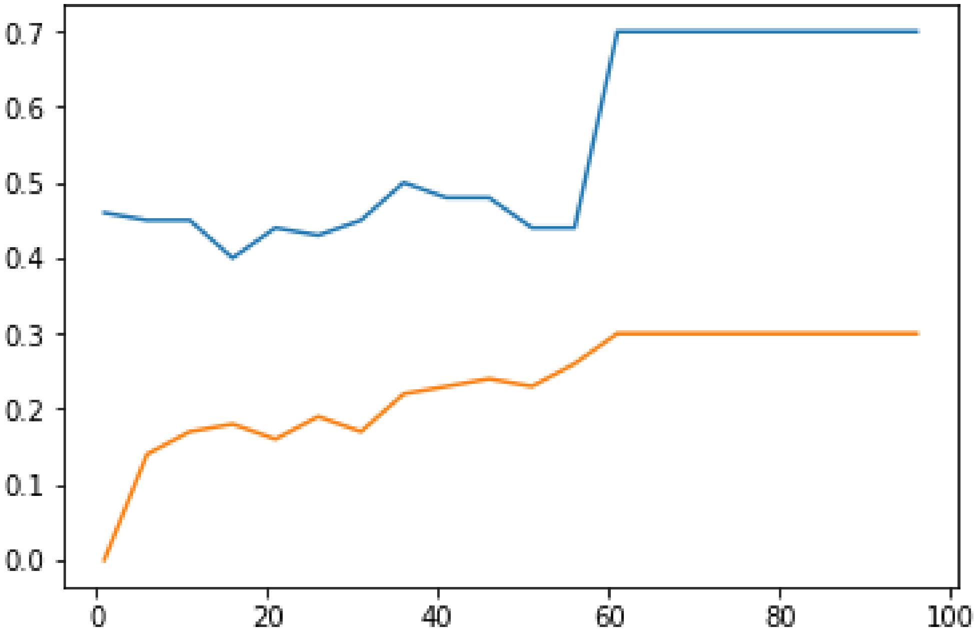


```
In [367]: k_order=test_error.index(min(test_error))
          k_star=range_k6[k_order]
          print("The best k is", k_star)
          lowest_training = min(train_error)
          lowest_testing=[]
          lowest_testing.append(min(test_error))
          print("Since training error cannot be positive, the lowest_training error is:",lowes
```

```
The best k is 7
Since training error cannot be positive, the lowest_training error is: 0.0
```

```
In [368]: range_k2=[]
          test_error2=[]
          train_error2=[]

          for i in range(1,len(Xtrain),5):
              range_k2.append(i)
              clf = KNeighborsClassifier(n_neighbors = i,
                                 algorithm='auto',
                                 leaf_size=30,
                                 metric='minkowski',
                                 metric_params=None,
                                 n_jobs=1,
                                 p=1,
                                 weights='uniform')
              clf.fit(Xtrain,Ytrain)
              y_pred_train=clf.predict(Xtrain)
              y_pred_test=clf.predict(Xtest)
              cm_train=confusion_matrix(Ytrain,y_pred_train)
              cm_test=confusion_matrix(Ytest,y_pred_test)
              trainerror=((cm_train[0,1]+cm_train[1,0])/len(Xtrain))
              testerror=((cm_test[0,1]+cm_test[1,0])/len(Xtrain))
              train_error2.append(trainerror)
              test_error2.append(testerror)

              plt.figure
          plt.plot(range_k2,test_error2)
          plt.plot(range_k2,train_error2)
```

```
Out[368]: [<matplotlib.lines.Line2D at 0x1a2014f8d0>]
```

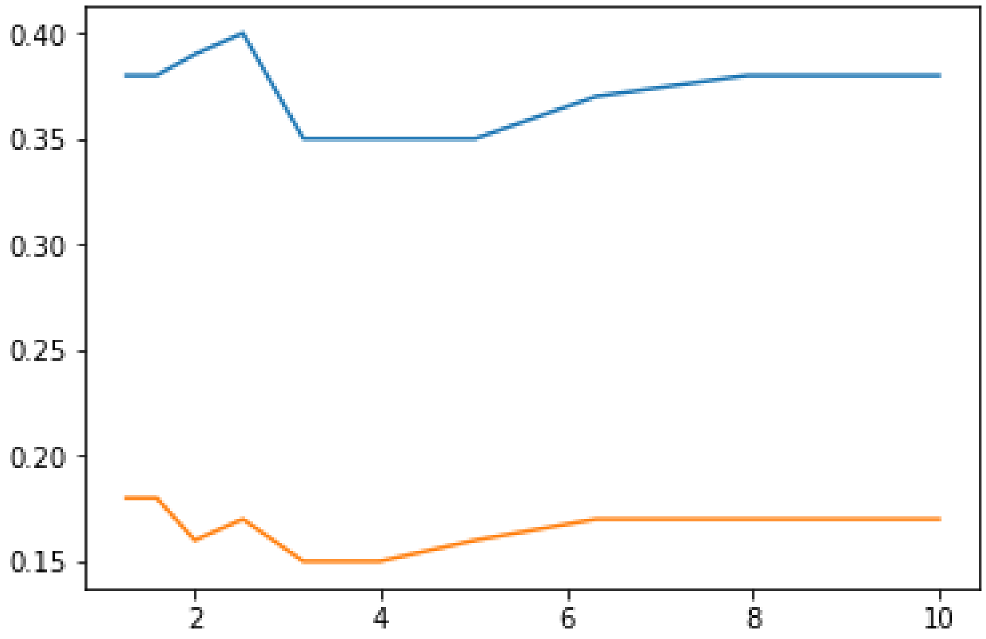


```
In [369]: best_ks=[]
          k_order2=test_error2.index(min(test_error2))
          k_star2=range_k2[k_order2]
          print("The best k is:", k_star2)
          lowest_testing.append(min(test_error2))
```

```
The best k is: 16
```

```
In [370]: import math
          p_range =[]
          train_error3=[]
          test_error3=[]

          for i in range(1,11,1):
              j=i/10
              num = math.pow(10, j)
              p_range.append(num)
              clf = KNeighborsClassifier(n_neighbors = k_star2,
                                 algorithm='auto',
                                 leaf_size=30,
                                 metric='minkowski',
                                 metric_params=None,
                                 n_jobs=1,
                                 p=num,
                                 weights='uniform')
              clf.fit(Xtrain,Ytrain)
              y_pred_train=clf.predict(Xtrain)
              y_pred_test=clf.predict(Xtest)
              cm_train=confusion_matrix(Ytrain,y_pred_train)
              cm_test=confusion_matrix(Ytest,y_pred_test)
              trainerror=((cm_train[0,1]+cm_train[1,0])/len(Xtrain))
              testerror=((cm_test[0,1]+cm_test[1,0])/len(Xtrain))
              train_error3.append(trainerror)
              test_error3.append(testerror)

          plt.figure
          plt.plot(p_range,test_error3)
          plt.plot(p_range,train_error3)
```

```
Out[370]: [<matplotlib.lines.Line2D at 0x1a296411d0>]
```



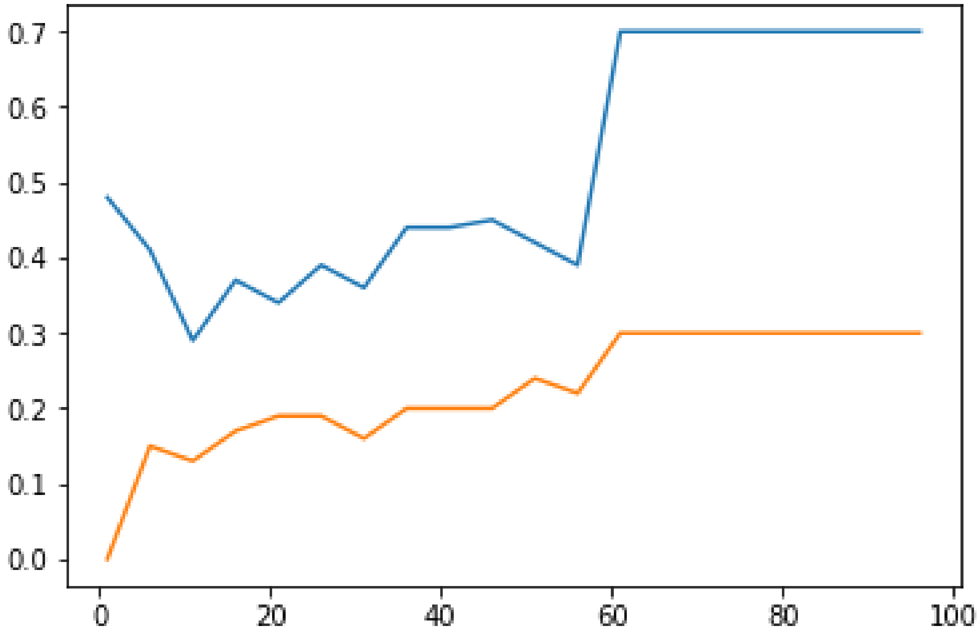
```
In [371]: p_order=test_error3.index(min(test_error3))
          p_star=p_range[p_order]
          print("The best 10log(p) is:",p_order)
          lowest_testing.append(min(test_error3))
```

```
The best 10log(p) is: 4
```

```
In [372]: inf_p = math.inf
          range_k4=[]
          test_error4=[]
          train_error4=[]

          for i in range(1,len(Xtrain),5):
              range_k4.append(i)
              clf = KNeighborsClassifier(n_neighbors = i,
                                algorithm='auto',
                                leaf_size=30,
                                metric='minkowski',
                                metric_params=None,
                                n_jobs=1,
                                p=inf_p,
                                weights='uniform')
              clf.fit(Xtrain,Ytrain)
              y_pred_train=clf.predict(Xtrain)
              y_pred_test=clf.predict(Xtest)
```

```
        cm_train=confusion_matrix(Ytrain,y_pred_train)
        cm_test=confusion_matrix(Ytest,y_pred_test)
        trainerror=((cm_train[0,1]+cm_train[1,0])/len(Xtrain))
        testerror=((cm_test[0,1]+cm_test[1,0])/len(Xtrain))
        train_error4.append(trainerror)
        test_error4.append(testerror)

    plt.figure
    plt.plot(range_k4,test_error4)
    plt.plot(range_k4,train_error4)
```

```
Out[372]: [<matplotlib.lines.Line2D at 0x1a296e6710>]
```



```
In [373]: k_order4=test_error4.index(min(test_error4))
          k_star4=range_k4[k_order4]
          print("The best k is", k_star4)
          lowest_testing.append(min(test_error4))
```

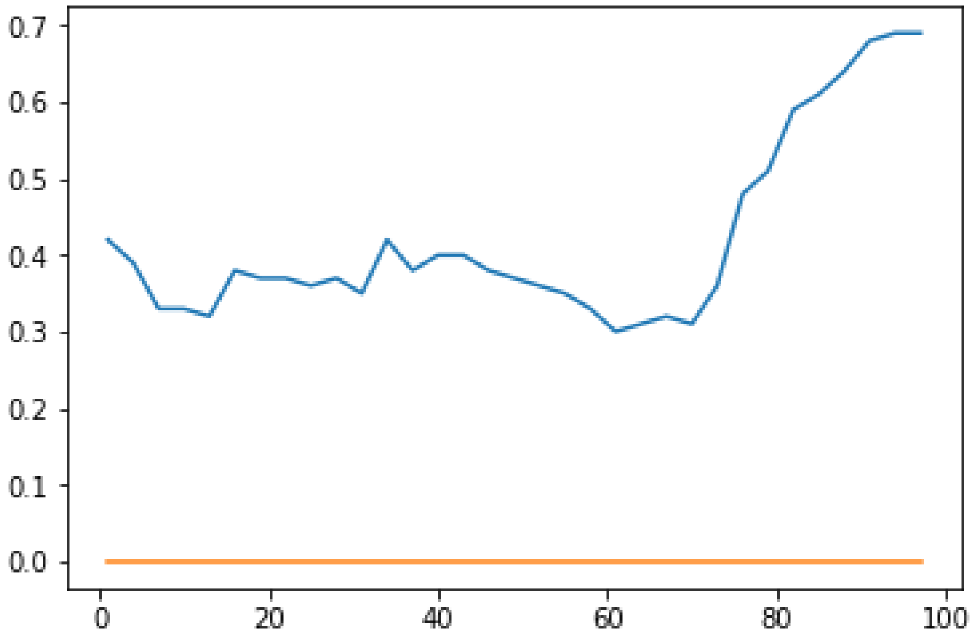
```
The best k is 11
```

```
In [374]: #range_k5=[]
          #test_error5=[]
          #train_error5=[]

          #for i in range(1,len(Xtrain),5):
```

```
#    range_k5.append(i)
#    clf = KNeighborsClassifier(n_neighbors = i,
#                     algorithm='auto',
#                     leaf_size=30,
#                     metric='mahalanobis',
#                     metric_params={'V': np.cov(Xtrain,Ytrain)},
#                     n_jobs=1,
#                     p=inf_p,
#                     weights='uniform')
#    clf.fit(Xtrain,Ytrain)
#    y_pred_train=clf.predict(Xtrain)
#    y_pred_test=clf.predict(Xtest)
#    cm_train=confusion_matrix(Ytrain,y_pred_train)
#    cm_test=confusion_matrix(Ytest,y_pred_test)
#    trainerror=((cm_train[0,1]+cm_train[1,0])/len(Xtrain))
#    testerror=((cm_test[0,1]+cm_test[1,0])/len(Xtrain))
#    train_error5.append(trainerror)
#    test_error5.append(testerror)

#plt.figure
#plt.plot(range_k5,test_error5)
#plt.plot(range_k5,train_error5)
```

```
In [375]: #Inverse Weight
          range_k6=[]
          test_error6=[]
          train_error6=[]

          for i in range(1,len(Xtrain),3):
              range_k6.append(i)
              clf = neighbors.KNeighborsClassifier(n_neighbors = i,weights='distance')
              clf.fit(Xtrain,Ytrain)
              y_pred_train=clf.predict(Xtrain)
              y_pred_test=clf.predict(Xtest)
              cm_train=confusion_matrix(Ytrain,y_pred_train)
              cm_test=confusion_matrix(Ytest,y_pred_test)
              trainerror=((cm_train[0,1]+cm_train[1,0])/len(Xtrain))
              testerror=((cm_test[0,1]+cm_test[1,0])/len(Xtrain))
              train_error6.append(trainerror)
              test_error6.append(testerror)

          plt.figure
          plt.plot(range_k6,test_error6)
          plt.plot(range_k6,train_error6)
```

```
Out[375]: [<matplotlib.lines.Line2D at 0x1a297a7518>]
```



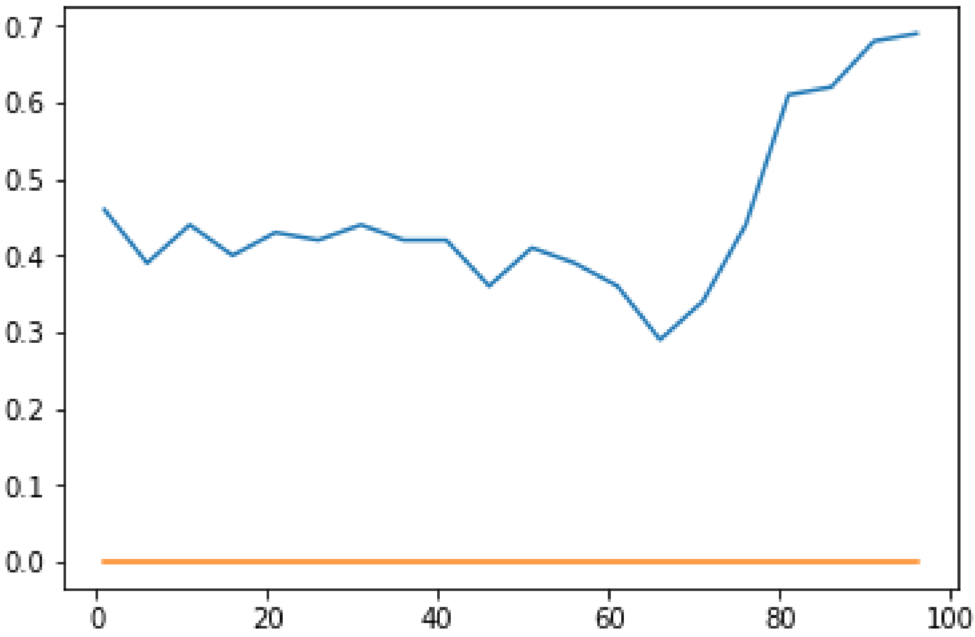
```
In [376]: k_order6=test_error6.index(min(test_error6))
          k_star6=range_k6[k_order6]
          print("The best k is", k_star6)
          lowest_testing.append(min(test_error6))
```

```
The best k is 61
```

```
In [377]: range_k7=[]
          test_error7=[]
          train_error7=[]

          for i in range(1,len(Xtrain),5):
              range_k7.append(i)
              clf = KNeighborsClassifier(n_neighbors = i,
                                 algorithm='auto',
                                 leaf_size=30,
                                 metric='minkowski',
                                 metric_params=None,
                                 n_jobs=1,
                                 p=1,
                                 weights='distance')
              clf.fit(Xtrain,Ytrain)
              y_pred_train=clf.predict(Xtrain)
              y_pred_test=clf.predict(Xtest)
              cm_train=confusion_matrix(Ytrain,y_pred_train)
```

```
            cm_test=confusion_matrix(Ytest,y_pred_test)
            trainerror=((cm_train[0,1]+cm_train[1,0])/len(Xtrain))
            testerror=((cm_test[0,1]+cm_test[1,0])/len(Xtrain))
            train_error7.append(trainerror)
            test_error7.append(testerror)

            plt.figure
        plt.plot(range_k7,test_error7)
        plt.plot(range_k7,train_error7)
```

```
Out[377]: [<matplotlib.lines.Line2D at 0x1a2986bc50>]
```



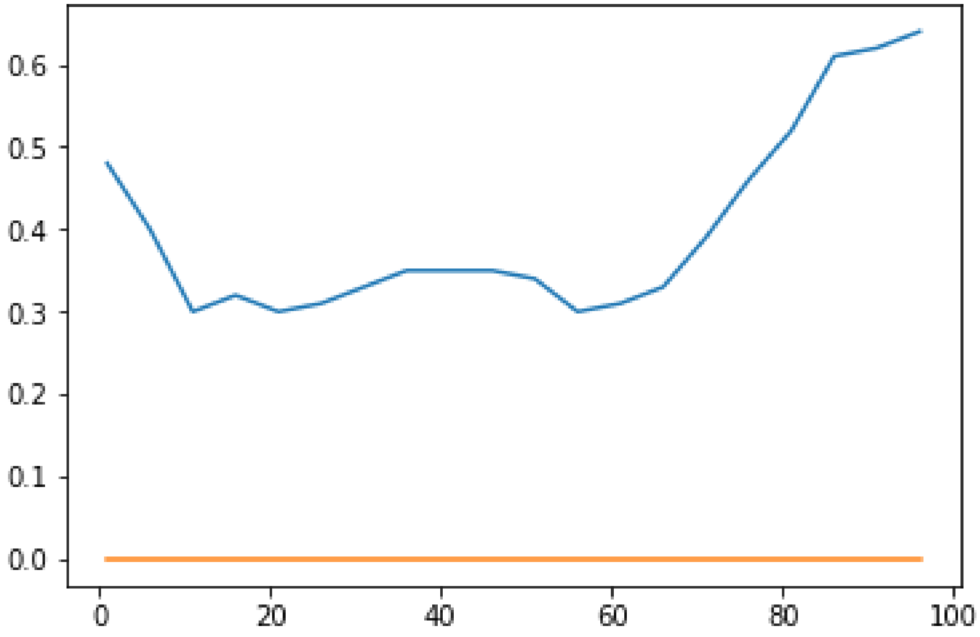
```
In [378]: k_order7=test_error7.index(min(test_error7))
          k_star7=range_k7[k_order7]
          print("The best k is", k_star7)
          lowest_testing.append(min(test_error7))
```

```
The best k is 66
```

```
In [379]: inf_p = math.inf
          range_k8=[]
          test_error8=[]
          train_error8=[]

          for i in range(1,len(Xtrain),5):
```

```
    range_k8.append(i)
    clf = KNeighborsClassifier(n_neighbors = i,
                     algorithm='auto',
                     leaf_size=30,
                     metric='minkowski',
                     metric_params=None,
                     n_jobs=1,
                     p=inf_p,
                     weights='distance')
    clf.fit(Xtrain,Ytrain)
    y_pred_train=clf.predict(Xtrain)
    y_pred_test=clf.predict(Xtest)
    cm_train=confusion_matrix(Ytrain,y_pred_train)
    cm_test=confusion_matrix(Ytest,y_pred_test)
    trainerror=((cm_train[0,1]+cm_train[1,0])/len(Xtrain))
    testerror=((cm_test[0,1]+cm_test[1,0])/len(Xtrain))
    train_error8.append(trainerror)
    test_error8.append(testerror)

plt.figure
plt.plot(range_k8,test_error8)
plt.plot(range_k8,train_error8)
```

```
Out[379]: [<matplotlib.lines.Line2D at 0x1a2993a9b0>]
```



```
In [380]: k_order8=test_error8.index(min(test_error8))
          k_star8=range_k8[k_order8]
          print("The best k is", k_star8)
          lowest_testing.append(min(test_error8))
```

```
The best k is 11
```

```
In [381]: lowest_training = min(train_error)
          print("Since training error cannot be negative, the lowest training error is",lowest_
          lowest_testing_val = min(lowest_testing)
          print("The lowest testing error is",lowest_testing_val,",under the Chebyshev Distanc
```

```
Since training error cannot be negative, the lowest training error is 0.0
The lowest testing error is 0.29 ,under the Chebyshev Distance evaluation
```