```
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
print("Switched to:",matplotlib.get_backend())
from scipy.stats import norm
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics


data = pd.read_csv(r'data_banknote_authentication.csv')
##Q1.2
def GR(data):
    if data['class']==0:
        return'green'
    else:
        return'red'
data.loc[:,'Color']=data.apply(GR,axis =1)
print(data)
##Q1.3
print("Class 0")
df_0 =
data.loc[data['class']==0,['variance','skewness','curtosis','entropy']]
print(df_0.describe().round(2))
print("Class 1")
df_1 =
data.loc[data['class']==1,['variance','skewness','curtosis','entropy']]
print(df_0.describe().round(2))
print("Class All")
print(data.describe().round(2))
```

```
##Q2
df_2=data.drop(labels = ['variance', 'skewness', 'curtosis', 'entropy' ],
axis = 1)
df_3 = df_0.drop(labels = ['variance', 'skewness', 'curtosis', 'entropy' ],
axis = 1)
df_4 = df_1.drop(labels = ['variance', 'skewness', 'curtosis', 'entropy' ],
axis = 1)
data_train_0,data_test_0,df_3_train_0,df_3_test_0=
train_test_split(df_0,df_3,test_size=0.5,random_state = 50,shuffle = True)
features = ['variance', 'skewness', 'curtosis', 'entropy']
pair_plot = sns.pairplot(data_train_0[features])
```

```
plot_kws={'color':'green'}
diag_kws={'color':'green'}
#plt.savefig('Good_bills.pdf')
#plt.figure()
data_train_1,data_test_1,df_4_train_1,df_4_test_1=
train_test_split(df_1,df_4,test_size=0.5,random_state = 50,shuffle = True)
features = ['variance', 'skewness', 'curtosis', 'entropy']
pair_plot = sns.pairplot(data_train_1[features])
#plt.savefig('Fake_bills.pdf')
#plt.figure()
data_train,data_test= train_test_split(data,test_size=0.96,random_state =
50,shuffle = True)
print('#################################################################
#############')
print(data_train)
print('#################################################################
#############')
```

```
list0 =
[1,0,0,0,0,1,0,1,0,0,0,1,1,1,1,0,0,1,1,1,0,0,0,1,0,0,1,0,0,1,0,1,1,0,1,
1,0,0,0,1,1,0,0,1,0,1,1,1,1,0,0,0,1,0]
list1 =
[1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,1,1,1,0,1,
1,0,0,0,0,0,0,0,1,0,1,0,1,1,0,0,0,1,0]
list2 = [None] * 50
```

```
for i in range(50):
    if(list0[i] == 0) and (list1[i] == 0):
        list2[i] = 'TN'
    elif(list0[i] == 0) and (list1[i] == 1):
        list2[i] = 'FP'
    elif(list0[i] == 1) and (list1[i] == 1):
        list2[i] = 'TP'
    elif(list0[i] == 1) and (list1[i] == 0):
        list2[i] = 'FN'
    else:
        continue
temp1 = 0
temp2 = 0
temp3 = 0
temp4 = 0
for i in range(50):
    if(list2[i] == 'TN'):
        temp1 += 1
```

```
    elif(list2[i] == 'FP'):
        temp2 += 1
    elif(list2[i] == 'TP'):
        temp3 += 1
    elif(list2[i] == 'FN'):
        temp4 += 1
    else:
        continue
print('TP: %d' % temp3 )
print('TN: %d' % temp1 )
print('FP: %d' % temp2 )
print('FN: %d' % temp4 )
temp5 = (temp1+temp3)/(temp1+temp2+temp3+temp4)
print('Accuracy: %d' % int(temp5 * 100) + '%')
temp6 = temp3/(temp3+temp4)
temp7 = temp1/(temp1+temp2)
print('TPR: %f' % temp6)
print('TNR: %f' % temp7)
print('##########################################################################
############')
##Q3
data2 = [
  [-2.48350,-7.449400,6.896400,-0.644840,1],
  [-2.44730,12.624700,0.735730,-7.661200,0],
  [1.85330,6.145800,1.017600,-2.040100,0],
  [1.72570,-4.469700,8.221900,-1.807300,0],
  [4.25860,11.296200,-4.094300,-4.345700,0],
  [-1.78860,-6.348600,5.615400,0.425840,1],
  [4.06320,3.584000,0.725450,0.394810,0],
  [-0.59587, 2.481100,-2.867300,-0.898280,1],
  [2.42260,-4.575200,5.947000,0.215070,0],
  [5.26200,3.983400,-1.557200,1.010300,0],
  [2.36780,6.839000,8.420700,-0.448290,0],
  [0.68180,4.850400,-5.213300,-6.104300,1],
  [-3.00610,-12.237700,11.955200,-2.160300,1],
  [-1.13060,1.845800,-1.357500,-1.380600,1],
  [-1.98810,0.999450,-0.285620,-0.700440,1],
  [3.62160,8.666100,-2.807300,-0.446990,0],
  [3.26970,-4.341400,3.688400,-0.298290,0],
  [-0.94255,0.039307,-0.241920,0.315930,1],
  [-1.47810,0.142770,-1.162200,-0.485790,1],
  [1.35180,1.059500,-2.343700,0.399980,1],
  [4.15290,-3.935800,2.863300,-0.017686,0],
  [5.50400,10.367100,-4.413000,-4.021100,0],
```

```
  [1.14320,-3.741300,5.577700,-0.635780,0],
  [-0.40951,-0.155210,0.060545,-0.088807,1],
  [1.98180,9.262100,-3.521000,-1.872000,0],
  [1.13170,3.964700,3.397900,0.843510,0],
  [-3.85520,3.521900,-0.384150,-3.860800,1],
  [4.24060,-2.485200,1.608000,0.715500,0],
  [4.60140,5.626400,-2.123500,0.193090,0],
  [-0.66008,-3.226000,3.805800,1.183600,1],
  [0.51950,-3.263300,3.089500,-0.984900,0],
  [-2.90200,-7.656300,11.831800,-0.842680,1],
  [-2.29180,-7.257000,7.959700,0.921100,1],
  [-1.18040,11.509300,0.155650,-6.819400,0],
  [-3.22380,2.793500,0.322740,-0.860780,1],
  [-3.32030,-0.026910,2.961800,-0.449580,1],
  [-0.16735,7.627400,1.206100,-3.624100,0],
  [1.93580,8.165400,-0.023425,-2.258600,0],
  [3.76350,2.781100 ,0.661190,0.341790,0],
  [-4.02180,-8.304000,12.555000,-1.509900,1],
  [-5.03010,7.503200,-0.133960,-7.503400,1],
  [3.22940,7.739100,-0.378160,-2.540500,0],
  [4.13730,0.492480,1.093000,1.827600,0],
  [-2.55260,-7.362500,6.925500,-0.668110,1],
  [-0.64472,-4.606200,8.347000,-2.709900,0],
  [-1.99830,-6.607200,4.825400,-0.419840,1],
  [-5.20490,7.259000,0.070827,-7.300400,1],
  [-2.57240,-0.956020,2.707300,-0.166390,1],
  [-1.38870,-4.877300,6.477400,0.341790,1],
  [5.49440,1.547800,0.041694,1.928400,0]
]
data3=[
    [-1.5252,-6.2534,5.3524,0.59912],
    [-2.0336,-1.4092,1.1582,0.36507],
    [0.57461,10.1105,-1.6917,-4.3922],
    [-0.3489,3.1929,-3.4054,-3.1832],
    [-3.9933,2.6218,0.62863,-1.1595],
    [0.6818,4.8504,-5.2133,-6.1043],
    [-1.9966,-9.5001,9.682,-0.12889],
    [-2.9672,-13.2869,13.4727,-2.6271],
    [-4.3667,6.0692,0.57208,-5.4668],
    [-3.8952,3.8157,-0.31304,-3.8194],
    [-4.1429,2.7749,0.68261,-0.71984],
    [4.3239,-4.8835,3.4356,-0.5776],
    [0.77445,9.0552,-2.4089,-1.3884],
    [-2.5912,-0.10554,1.2798,1.0414],
```

```
    [-1.7063,2.7956,-2.378,-2.3491],
    [5.0185,8.5978,-2.9375,-1.281],
    [2.6104,8.0081,-0.23592,-1.7608],
    [-6.5773,6.8017,0.85483,-7.5344],
    [5.086,3.2798,-1.2701,1.1189],
    [3.4776,8.811,-3.1886,-0.92285],
    [3.966,3.9213,0.70574,0.33662],
    [-3.2238,2.7935,0.32274,-0.86078],
    [3.245,6.63,-0.63435,0.86937],
    [1.5077,1.9596,-3.0584,-0.12243],
    [-1.8554,-9.6035,7.7764,-0.97716],
    [4.2969,7.617,-2.3874,-0.96164],
    [-2.3797,-1.4402,1.1273,0.16076],
    [-3.1366,0.42212,2.6225,-0.064238],
    [-1.5222,10.8409,2.7827,-4.0974],
    [-1.2528,10.2036,2.1787,-5.6038],
    [0.2346,-4.5152,2.1195,1.4448],
    [-3.8894,-7.8322,9.8208,0.47498],
    [-3.3924,3.3564,-0.72004,-3.5233],
    [-0.36038,4.1158,3.1143,-0.37199],
    [2.7296,2.8701,0.51124,0.5099],
    [-2.5373,-6.959,8.8054,1.5289],
    [-2.456,-0.24418,1.4041,-0.45863],
    [-3,-9.1566,9.5766,-0.73018],
    [-1.0833,-0.31247,1.2815,0.41291],
    [-0.72068,-6.7583,5.8408,0.62369],
    [3.82,10.9279,-4.0112,-5.0284],
    [-0.49081,2.8452,-3.6436,-3.1004],
    [2.8561,6.9176,-0.79372,0.48403],
    [-1.803,11.8818,2.0458,-5.2728],
    [-3.8483,-12.8047,15.6824,-1.281],
    [-2.6479,10.1374,-1.331,-5.4707],
    [3.5499,8.6165,-3.2794,-1.2009],
    [1.3754,8.8793,-1.9136,-0.53751],
    [4.0713,10.4023,-4.1722,-4.7582],
    [4.7181,10.0153,-3.9486,-3.8582]
]
datamat = np.array(data2)
X = datamat[:,0:4]
Y = datamat[:,4]
knn = KNeighborsClassifier(n_neighbors=3,weights='distance')
knn.fit(X,Y)
print(knn.predict([data3[0]]))
print(knn.predict([data3[1]]))
```

```
print(knn.predict([data3[2]]))
print(knn.predict([data3[3]]))
print(knn.predict([data3[4]]))
print(knn.predict([data3[5]]))
print(knn.predict([data3[6]]))
print(knn.predict([data3[7]]))
print(knn.predict([data3[8]]))
print(knn.predict([data3[9]]))
print(knn.predict([data3[10]]))
print(knn.predict([data3[11]]))
print(knn.predict([data3[12]]))
print(knn.predict([data3[13]]))
print(knn.predict([data3[14]]))
print(knn.predict([data3[15]]))
print(knn.predict([data3[16]]))
print(knn.predict([data3[17]]))
print(knn.predict([data3[18]]))
print(knn.predict([data3[19]]))
print(knn.predict([data3[20]]))
print(knn.predict([data3[21]]))
print(knn.predict([data3[22]]))
print(knn.predict([data3[23]]))
print(knn.predict([data3[24]]))
print(knn.predict([data3[25]]))
print(knn.predict([data3[26]]))
print(knn.predict([data3[27]]))
print(knn.predict([data3[28]]))
print(knn.predict([data3[29]]))
print(knn.predict([data3[30]]))
print(knn.predict([data3[31]]))
print(knn.predict([data3[32]]))
print(knn.predict([data3[33]]))
print(knn.predict([data3[34]]))
print(knn.predict([data3[35]]))
print(knn.predict([data3[36]]))
print(knn.predict([data3[37]]))
print(knn.predict([data3[38]]))
print(knn.predict([data3[39]]))
print(knn.predict([data3[40]]))
print(knn.predict([data3[41]]))
print(knn.predict([data3[42]]))
print(knn.predict([data3[43]]))
print(knn.predict([data3[44]]))
print(knn.predict([data3[45]]))
```

```
print(knn.predict([data3[46]]))
print(knn.predict([data3[47]]))
print(knn.predict([data3[48]]))
print(knn.predict([data3[49]]))
print('##################################################################
############')
```

```
list3=[1,1,0,1,1,1,1,1,1,1,1,0,0,1,1,0,0,1,0,0,0,1,0,1,1,0,1,1,0,0,1,1,
1,0,0,1,1,1,1,1,0,1,0,0,1,0,0,0,0,0]
list4 =
[1,1,0,1,1,1,1,1,1,1,1,0,0,1,1,0,0,1,0,0,0,1,0,1,1,0,1,1,0,0,0,1,1,0,0,
1,1,1,1,1,0,1,0,0,1,0,0,0,0,0]
```

```
list5 = [None] * 50
```

```
for i in range(50):
    if(list3[i] == 0) and (list4[i] == 0):
        list5[i] = 'TN'
    elif(list3[i] == 0) and (list4[i] == 1):
        list5[i] = 'FP'
    elif(list3[i] == 1) and (list4[i] == 1):
        list5[i] = 'TP'
    elif(list3[i] == 1) and (list4[i] == 0):
        list5[i] = 'FN'
    else:
        continue
temp1 = 0
temp2 = 0
temp3 = 0
temp4 = 0
for i in range(50):
    if(list5[i] == 'TN'):
        temp1 += 1
    elif(list5[i] == 'FP'):
        temp2 += 1
    elif(list5[i] == 'TP'):
        temp3 += 1
    elif(list5[i] == 'FN'):
        temp4 += 1
    else:
        continue
print('TP: %d' % temp3 )
print('TN: %d' % temp1 )
print('FP: %d' % temp2 )
```

```
print('FN: %d' % temp4 )
temp5 = (temp1+temp3)/(temp1+temp2+temp3+temp4)
print('Accuracy: %d' % int(temp5 * 100) + '%')
temp6 = temp3/(temp3+temp4)
temp7 = temp1/(temp1+temp2)
print('TPR: %f' % temp6)
print('TNR: %f' % temp7)
print('##################################################################
############')
data3 = [
    [7,3,0,8],
]
knn = KNeighborsClassifier(n_neighbors=3, weights='distance')
knn.fit(X,Y)
print("When k = 3",knn.predict([data3[0]]))
knn = KNeighborsClassifier(n_neighbors=5, weights='distance')
knn.fit(X,Y)
print("When k = 5",knn.predict([data3[0]]))
knn = KNeighborsClassifier(n_neighbors=7, weights='distance')
knn.fit(X,Y)
print("When  k = 7",knn.predict([data3[0]]))
knn = KNeighborsClassifier(n_neighbors=9, weights='distance')
knn.fit(X,Y)
print("When k = 9",knn.predict([data3[0]]))
knn = KNeighborsClassifier(n_neighbors=11, weights='distance')
knn.fit(X,Y)
print("When k = 11",knn.predict([data3[0]]))
```