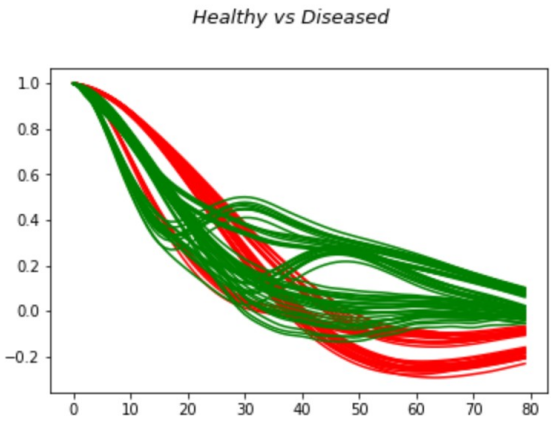
Multivariate Data Analysis Assignment

Date: 12/10/19
Student Number: 5074274
Student Name: Aditya Kunar

Ex 2.1-

Code and Method Used-

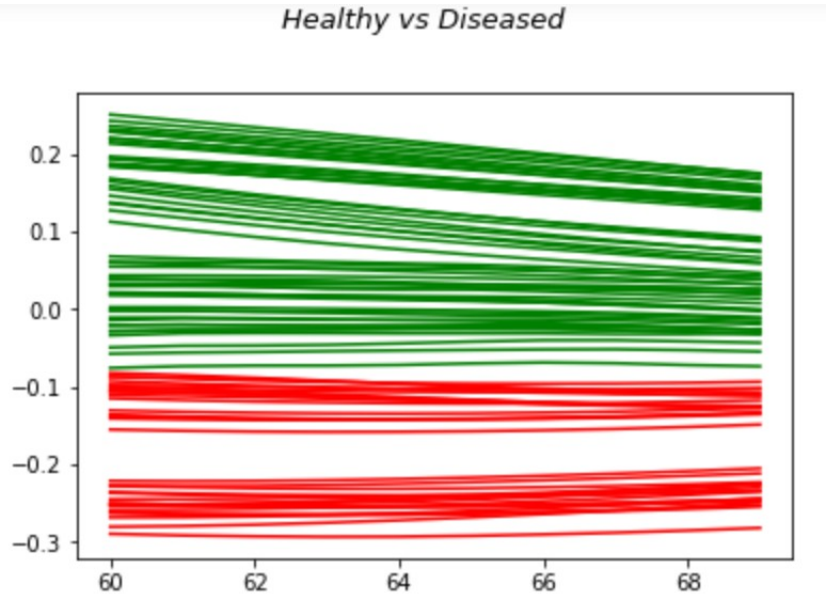
```
//Loading in the data.
Tmat = scipy.io.loadmat('trainecg.mat')
Tsmat= scipy.io.loadmat('testecg.mat')

//Defining the auto-correlation function.
def autocorr(x):
    result = np.correlate(x, x, mode='full')
    //Disregarding the auto-correlation values for k<0.
    res= result[3999:]
    return (pd.Series(res))

//Normalizing by the maximum value of each row as the maximum value is at k=0.
def norm(row):
    result = row/row.max()
    return result

//Applying the auto-correlation function on each row and normalizing.
dfHealthy=pd.DataFrame(Tmat["healthy"])
df_H_autocorr=dfHealthy.apply(autocorr,axis=1)
df_H_autocorrN=df_H_autocorr.apply(norm,axis=1)

dfDiseased=pd.DataFrame(Tmat["diseased"])
df_D_autocorr=dfDiseased.apply(autocorr,axis=1)
df_D_autocorrN=df_D_autocorr.apply(norm,axis=1)

//Plotting the data
for i in range(0,36):
            y=df_D_autocorrN.iloc[i]
            y=np.array(y).astype('float')
            y=y[0:80]
            k=range(0,80)
            plt.plot(k,y,color='red')

for i in range(0,54):
            x=df_H_autocorrN.iloc[i]
            x=np.array(x).astype('float')
            x=x[0:80]
            k=range(0,80)
            plt.plot(k,x,color='green')

plt.suptitle("Healthy vs Diseased", fontsize=13, fontweight=0, color='black',
style='italic', y=1.02)
```



Red denotes diseased.

Green denotes healthy.

Ex 2.2-

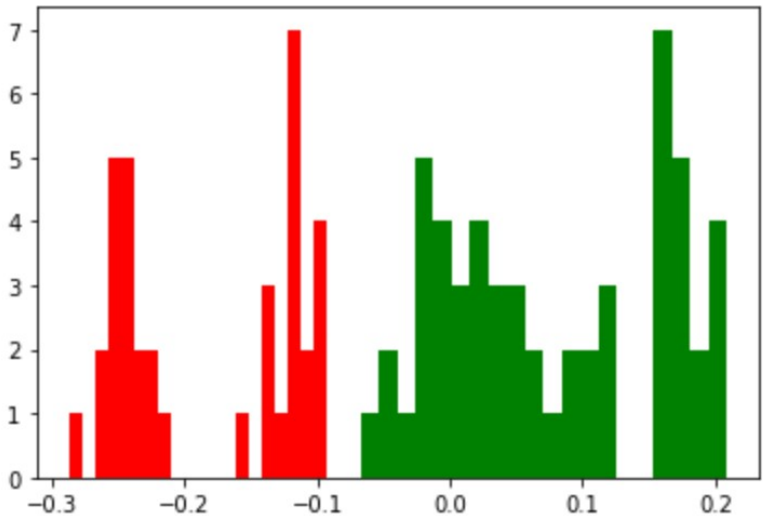


Magnifying the plot created earlier and performing the visual inspection, a good value for k^* which is able to best separate the two classes is 66.

Ex 3.1-

Code and Method Used-

```
//Plotting the histogram for k=66.
k=66
x = df_H_autocorrN[k]
y = df_D_autocorrN[k]
num_bins = 20
n, bins, patches = plt.hist(x, num_bins, facecolor='green', alpha=1)
n, bins, patches = plt.hist(y, num_bins, facecolor='red', alpha=1)
plt.show()
```



Red denotes diseased.

Green denotes healthy.

We can visually see a separation between the two classes for k=66.

Ex 3.2-

Code and Method Used-

```
Prob_H=54/90 #prior for healthy.
Prob_D=36/90 #prior for diseased.
x = df_H_autocorrN[k] #k=66
y = df_D_autocorrN[k] #k=66
muH, sigmaH = x.mean(), x.std() #mean and std.dev for healthy at k=66
muD, sigmaD = y.mean(), y.std() #mean and std.dev for diseased at k=66

nH=0 #initial count for healthy predictions.
//iterating through all healthy samples.
for i in range(0,54):

    //Generating a likelihood probability from the gaussian pdfs for healthy and
diseased.
    D_likelihood = scipy.stats.norm(muD, sigmaD).pdf(x[i])
    H_likelihood = scipy.stats.norm(muH, sigmaH).pdf(x[i])

    //Normalizing Constant
    norm_const= Prob_H*H_likelihood+Prob_D*D_likelihood

    //Using bayes' rule to obtain posterior probability.
    post_H=Prob_H*H_likelihood/(norm_const)
    post_D=Prob_D*D_likelihood/(norm_const)

    //Applying the Maximum A Posteriori decision rule and updating nH.
    if(post_H>post_D):
        nH+=1

//Printing the accuracy.
print("Accuracy : "+str(nH*100/54))
Accuracy : 100.0
//Here I am dividing the total number of healthy predictions(nH) with the number
of healthy patients in the sample ie 54 and converting it to a percentage.

//iterating through all diseased samples.
nD=0 #initial count for diseased predictions.
for i in range(0,36):

//Generating a likelihood probability from the gaussian pdfs for healthy and
diseased.
    D_likelihood = scipy.stats.norm(muD, sigmaD).pdf(y[i])
    H_likelihood = scipy.stats.norm(muH, sigmaH).pdf(y[i])

    //Normalizing Constant
    norm_const= Prob_H*H_likelihood+Prob_D*D_likelihood

    //Using bayes' rule to obtain posterior probability.
    post_H=Prob_H*H_likelihood/(norm_const)
    post_D=Prob_D*D_likelihood/(norm_const)

    //Applying the Maximum A Posteriori decision rule and updating nD.
    if(post_D>post_H):
        nD+=1

//Printing the accuracy.
print("Accuracy : " + str(nD*100/36))
Accuracy : 100.0
//Here I am dividing the total number of diseased predictions(nD) with the
number of diseased patients in the sample ie 36 and converting it to a
percentage.
```

Ex 3.3-

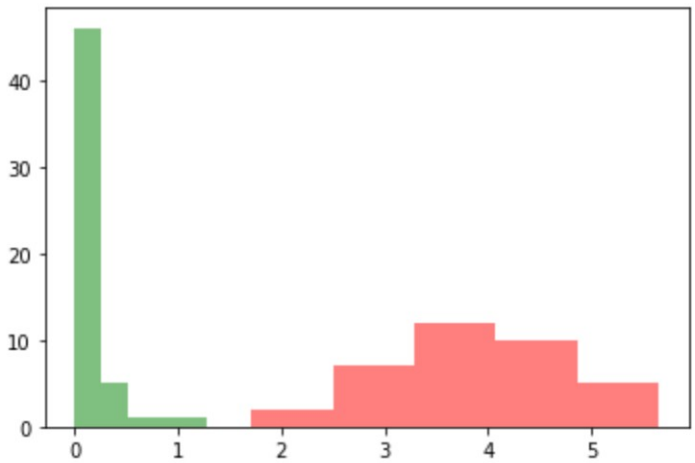
Code and Method Used-

```
//Generating the Gaussian PDFs for healthy and diseased. And calculating the
expected value and variance as shown below.
x = df_H_autocorrN[k] #k=66
y = df_D_autocorrN[k] #k=66
PDF_D = scipy.stats.norm(muD, sigmaD).pdf(y)
PDF_H = scipy.stats.norm(muD, sigmaD).pdf(x)
exp_D,var_D=PDF_D.mean(),PDF_D.var()
exp_H,var_H=PDF_H.mean(),PDF_H.var()
```

Results-

```
Expected Value for the Gaussian PDF of Healthy: 0.1078770717629081
Expected Value for the Gaussian PDF of Diseased: 3.8791699478632387
Variance for the Gaussian PDF of Diseased: 0.7392923516822234
Variance for the Gaussian PDF of Healthy: 0.051511887409997646
```

```
// Plotting the histogram of the PDFs for healthy and diseased.

x = df_H_autocorrN[k] #k=66
y = df_D_autocorrN[k] #k=66
muH, sigmaH = x.mean(), x.std() #mean and std.dev for healthy at k=66
muD, sigmaD = y.mean(), y.std() #mean and std.dev for diseased at k=66
PDF_D = scipy.stats.norm(muD, sigmaD).pdf(y)
PDF_H = scipy.stats.norm(muD, sigmaD).pdf(x)
num_bins = 10
n, bins, patches = plt.hist(PDF_H, num_bins, facecolor='green', alpha=0.5)
n, bins, patches = plt.hist(PDF_D, num_bins, facecolor='red', alpha=0.5)
plt.show()
```



Red denotes diseased.

Green denotes healthy.

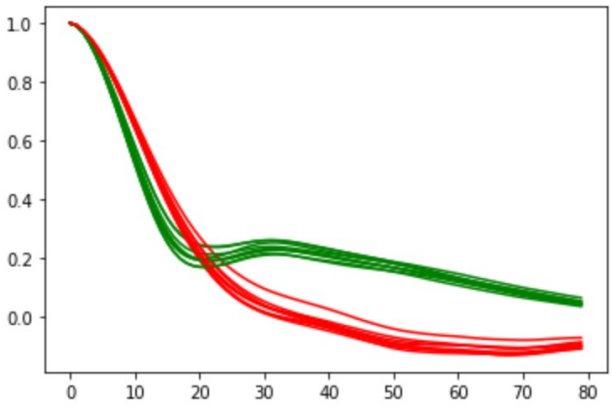
Similar to the previous histogram, we can visually see a separation here too.

Ex 3.4-

With k^*=66, I was able to achieve 100 percent accuracy on the training set. The results have been provided in my answer to Ex 3.2 as well. Essentially, I calculate the accuracy by measuring the number of correct “healthy” predictions on the healthy samples and the number of correct “diseased” predictions on the diseased samples. As shown, since there are no incorrect predictions, the accuracy is 100 percent on the overall training set.

Ex 3.5- Predictions on the test-set along with the auto-correlation plot.

```
user0: Healthy
user1: Healthy
user2: Healthy
user3: Diseased
user4: Diseased
user5: Diseased
user6: Healthy
user7: Healthy
user8: Diseased
user9: Diseased
user10: Healthy
user11: Healthy
user12: Healthy
user13: Diseased
user14: Diseased
user15: Healthy
user16: Diseased
user17: Diseased
```



Red denotes diseased and green denotes healthy.