**Pedro Sanchez Munoz**

**COEN 140 HW #2**

1) I split the data up into 80-20 chunks and imported them into lists of matrices of size 40 and 10 for further processing respectively.

2) LDA:

Testing error:

0.0

Training error:

0.025

Please find code in LDA.py

3) QDA:

Testing error:

0.0

Training error:

0.0166666666667

Please find code in QDA.py

I find it interesting that we are not over-fitting here but rather the opposite—the testing error remains the same at the cost of higher training error. In real world applications, this is definitely favorable over having a higher testing error than a training error.

4) The definition given in class was: “If LDA is applied to the dataset and the outputs for both training and testing error is 0, then the dataset is linearly separable.” Given that my training error for LDA was != 0 then we can say this dataset is NOT linearly separable.

However, upon further investigation online I was led to the idea that if ANY category produced no errors in prediction for both training and testing, then that category is linearly separable. In calculating my LDA’s error, I included print statements to determine which category’s predictions caused an error to see if I could isolate such a category. The results were three errors: two were caused by my model not predicting versicolor and one by it not predicting virginica when it was supposed to. Thus, the setosa category produced no errors and if my research is indeed correct, it is linearly separable from the other two categories.

5) If I remove the 1st feature, testing error went from 0.0 -> 0.0 and training error went from 0.016… -> 0.041666…

If I additionally remove the (both 1st and) 2nd feature, testing error remains at 0.0 and training error only goes up to 0.0333…

In contrast, if I remove the 4th feature, testing error went from 0 -> 0.0333 and training error went from 0.016… -> .05

This leads me to believe that the 1st and 2nd features are not very important in getting a correct overall prediction, since the testing error was unaffected and the training error was only affected by a factor of ~2.5. In predicting outcomes where the outcomes are close, this coefficient may seem large, but the outcomes from the Gaussian distribution I was getting were either huge (~240) or tiny (~1e-64) and so I think it is justified to say that a factor of 2.5 is small enough to be considered noise.

Please find the first two features removed in feature\_removal.py

6) LDA:

Testing error:

0.0

Training error:

0.05

QDA:

Testing error:

0.0

Training error:

0.05

Note: I generated the covariance matrix by taking the average of the three covariance matrices respective to each category.

Please find code in LDA\_indep.py and QDA\_indep.py