**Introduction**

For this homework assignment we were tasked with predicting which class a set of points belonged to, given multivariant Gaussian distribution. In order to train the classifier, I implemented the mvnpdf function, which gave us the discriminant. The mvnpdf function requires the mean and covariance of each class, I used the built-in mean and cov functions to obtain the corresponding values. I then compared the discriminant for each class given the corresponding estimated parameters and assigned each point to the class of whichever pdf resulted in a larger value.

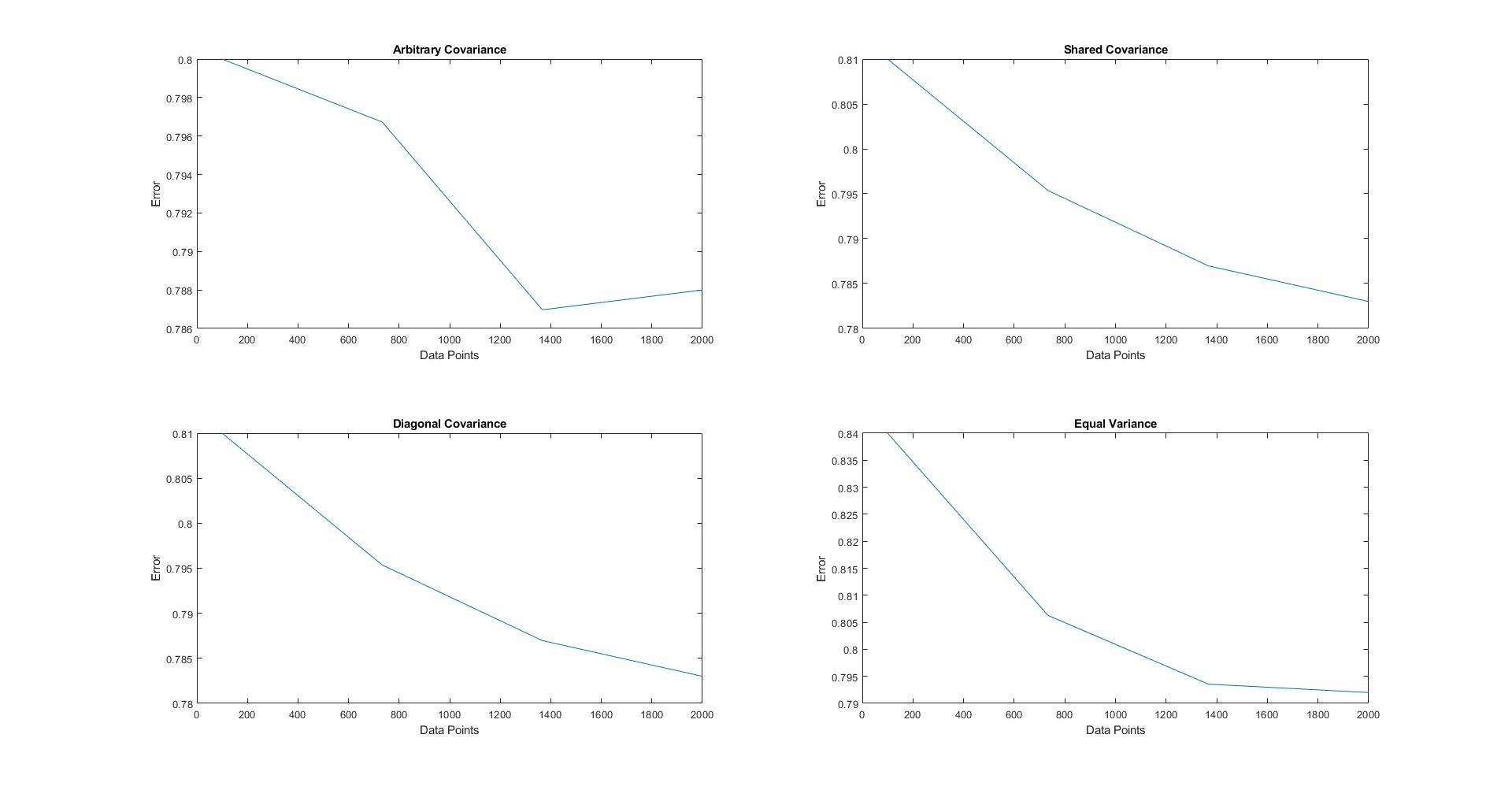
**Methods**

Below are algorithms that are critical to the implementation of the algorithm:

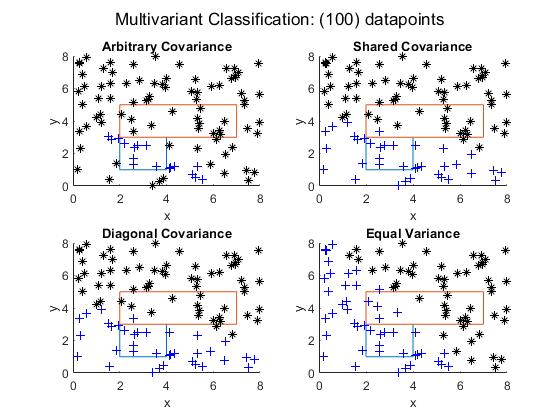
1. function [pts1,pts2] = getPts(ls,ds)
   1. Stores each of the points in the data set in the correct class, either class 1 or 2. This is later used to calculate the means and covariance for each class
2. function [mean1,mean2,cov1,cov2,sharedCov] = getParams(pts1,pts2)
   1. Calculates the shared covariance and each class’s mean and covariance. The mean and covariance are used to estimate the density function.
3. function [actual] = actualClass(x,y)
   1. Returns a vector that contains the correct label for each datapoint in the training set. This vector is used to determine whether the current point was labeled correctly.
4. function [prediction] = getPrediction(x,y,mean1,mean2,cov1,cov2,sharedCov,prediction,i)
   1. Classifies the point given the mean and covariances using the multivariant normal probability density function. It classifies the point for the following complexities: Arbitrary Covariance, Shared Covariance, Diagonal Covariance, and Equal Variance.
5. function [fn,fp] = getErr(pred,act,fn,fp)
   1. Calculates the false negative and false positive for each point. This is later used to calculate the error rate.
6. function plotErr(totalError,models,q)
   1. Plots the datapoint count on the x-axis and the error on the y-axis.

**Results**

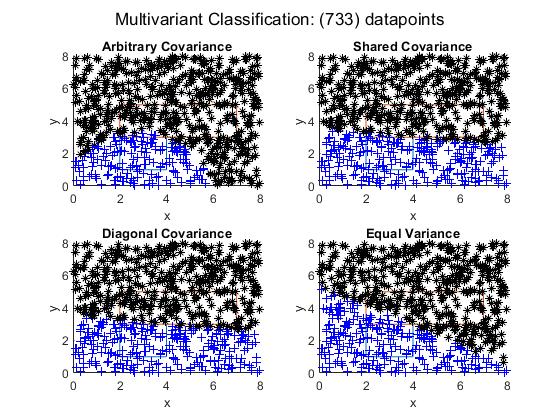
I ran the classification algorithm given 100, 733, 1366, and 2000 datapoints. For each iteration I classified each point for each of the previously mentioned model complexities. Below are the figures obtained.



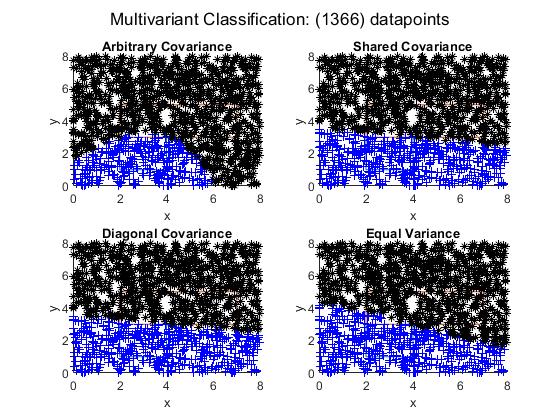
**Figure 1**. Plot of datapoint count versus error for each complexity



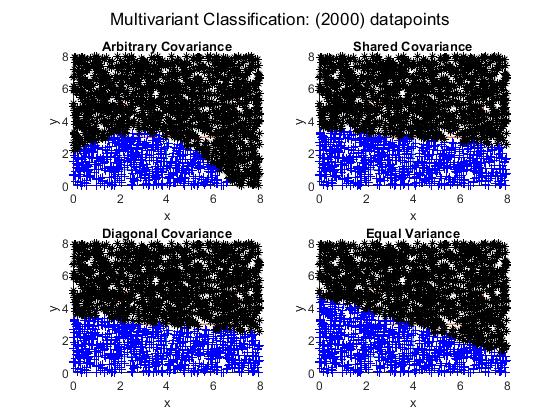
**Figure 2**. Classification Results (100 datapoints)



**Figure 3**. Classification Results (733 datapoints)



**Figure 4**. Classification Results (1366 datapoints)



**Figure 5**. Classification Results (2000 datapoints)

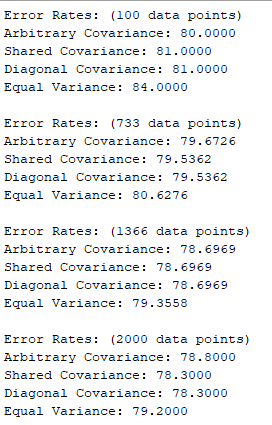


Figure 6. Error Rates

**Discussion**

The classification results for each complexity can be seen in figures 2 through 5. As you can see, each classifier does well differentiating class 1 from class 2. Unfortunately, the classifiers fail when we consider the points that do not belong to either class. All these points are incorrectly assigned to class 2. This leads to a high error rate, which can be seen in figure 6.

The error rate increases when we increase the number of datapoints. This occurs because most of the points are not in either class. Therefore, as the number of datapoints increases, the number of points which belong to class 0 but are assigned to class 2, increases.

The Arbitrary Covariance model complexity performed the best in classifying the data. This is expected since here we used the greatest number of parameters when estimating the distribution.

This classification method would have performed much better if we used rejection. This would have allowed us to reject the points that do not belong to either class. One downside to this would be that we would have to find the rejection threshold which would reject the correct classes, which would increase the complexity of the algorithm.

**Software listing and executable software**

To run the software, simply run the program.