Anthony Tummillo

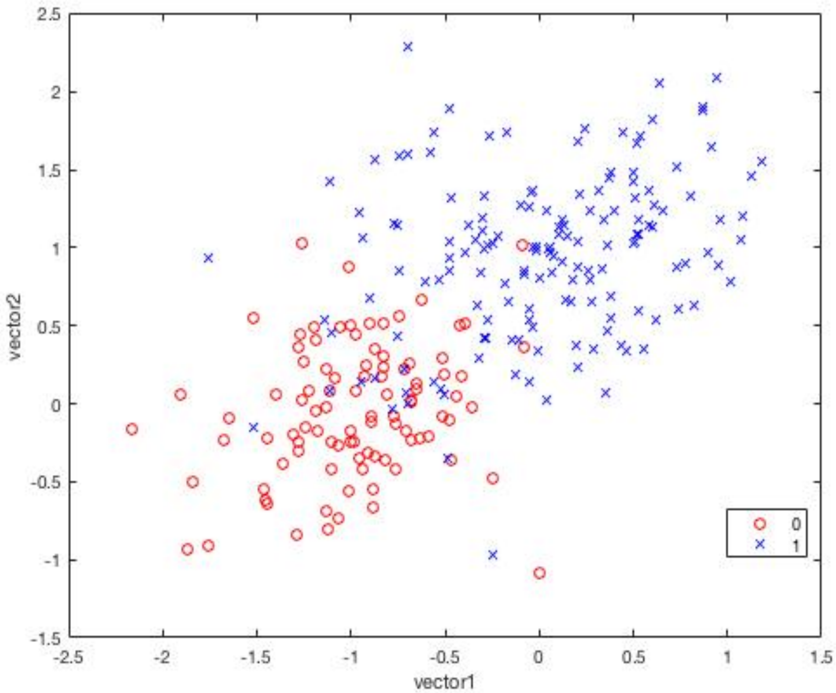
Homework #5 Report

Problem 1.

1.

The program has been written and submitted. It is named make_scatterplot.m

2.



Based on the scatter plot that I created (pictured above) it appears that it is **NOT** possible to separate the two classes perfectly with a linear decision boundary.

Problem 2.

(a)

$$\frac{\partial}{\partial w_j} l(D,w) = \sum_{i=1}^{n} \frac{\partial}{\partial z_i}\left[y_i \log g(z_i) + (1-y_i)\log(1-g(z_i))\right] \frac{\partial z_i}{\partial w_j}$$

$$\boxed{\frac{\partial z_i}{\partial w_j} = x_{i,j}}$$

$$\frac{\partial}{\partial z_i}\left[y_i \log g(z_i) + (1-y_i)\log(1-g(z_i))\right]$$

$$= y_i \frac{1}{g(z_i)} \frac{\partial g(z_i)}{\partial z_i} + (1-y_i)\frac{-1}{1-g(z_i)} \cdot \frac{\partial g(z_i)}{\partial z_i}$$

$$= y_i(1-g(z_i)) + (1-y_i)(-g(z_i)) = y_i - g(z_i)$$

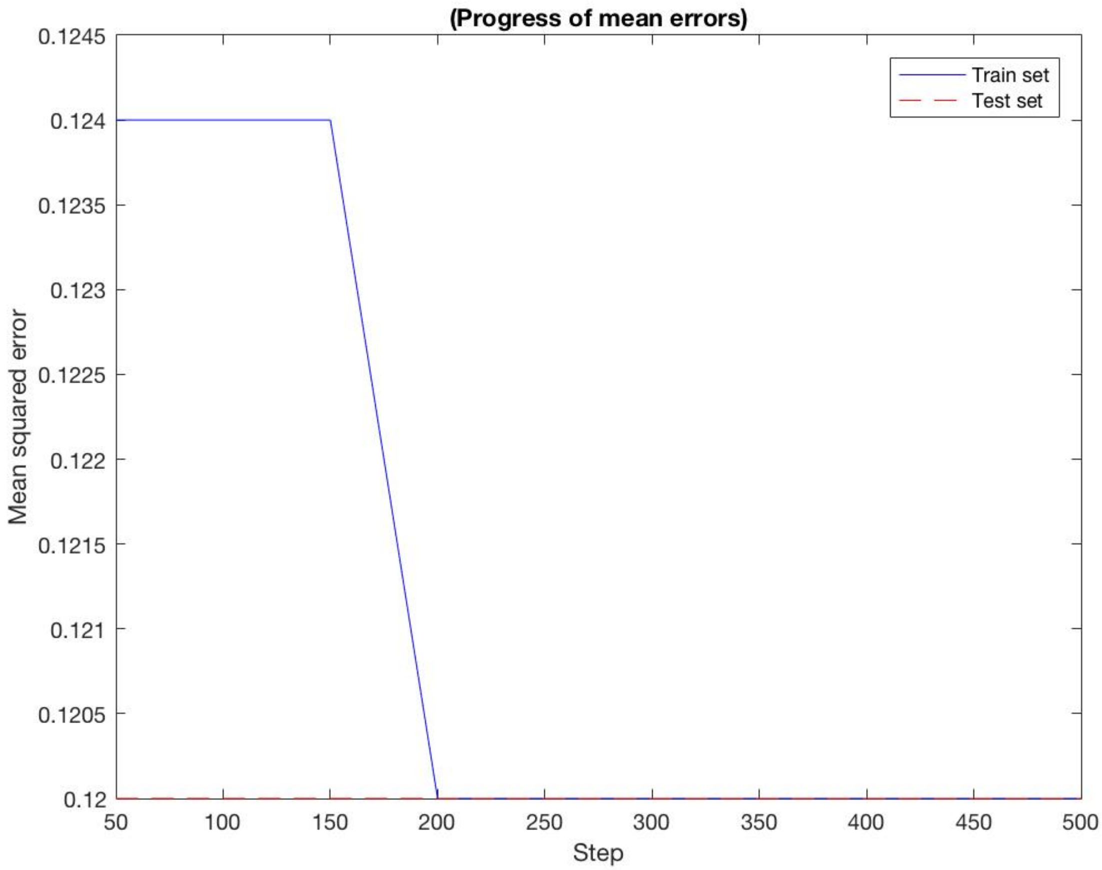
$$\nabla_w l(D,w) = \sum_{i=1}^{n} -X(y_i - g(w^T x_i)) = \sum_{i=1}^{n} -x_i(y_i - f(w, x_i))$$

(b)
The function is written and submitted. It uses a function I wrote and included log_function.m

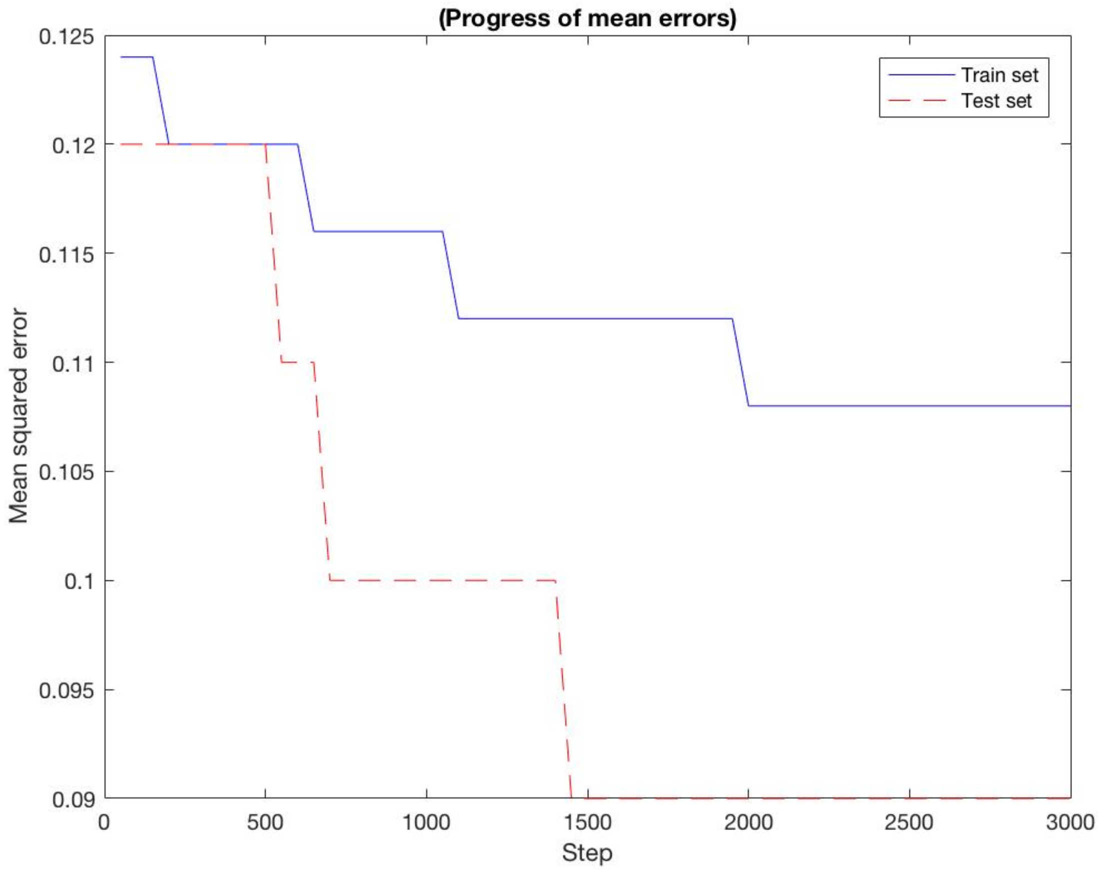
(c)
The script is written and submitted. It utilizes two other functions I wrote and submitted, confusion_matrix.m and mean_misclass.m

Weights: <w0 = 16.0552, w1 = 36.2745, w2 = 34.8291>
Mean misclassification train = 0.1200
Mean misclassification test = 0.1200

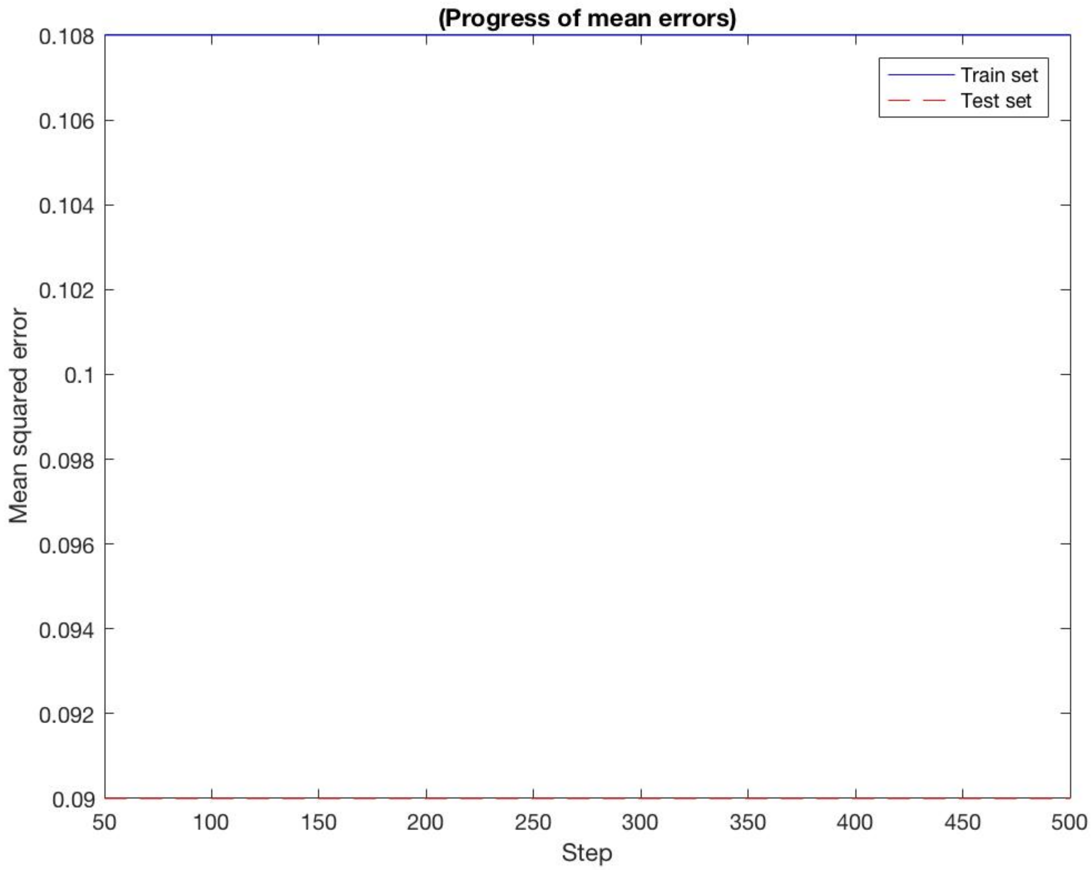
(d)



(e)
(I)
When 3000 steps were used, with a learning rate of 2/k
Weights: <w0 = 11.3281, w1 = 21.8031, w2 = 14.6949>
Mean misclassification train = 0.1080
Mean misclassification test = 0.0900

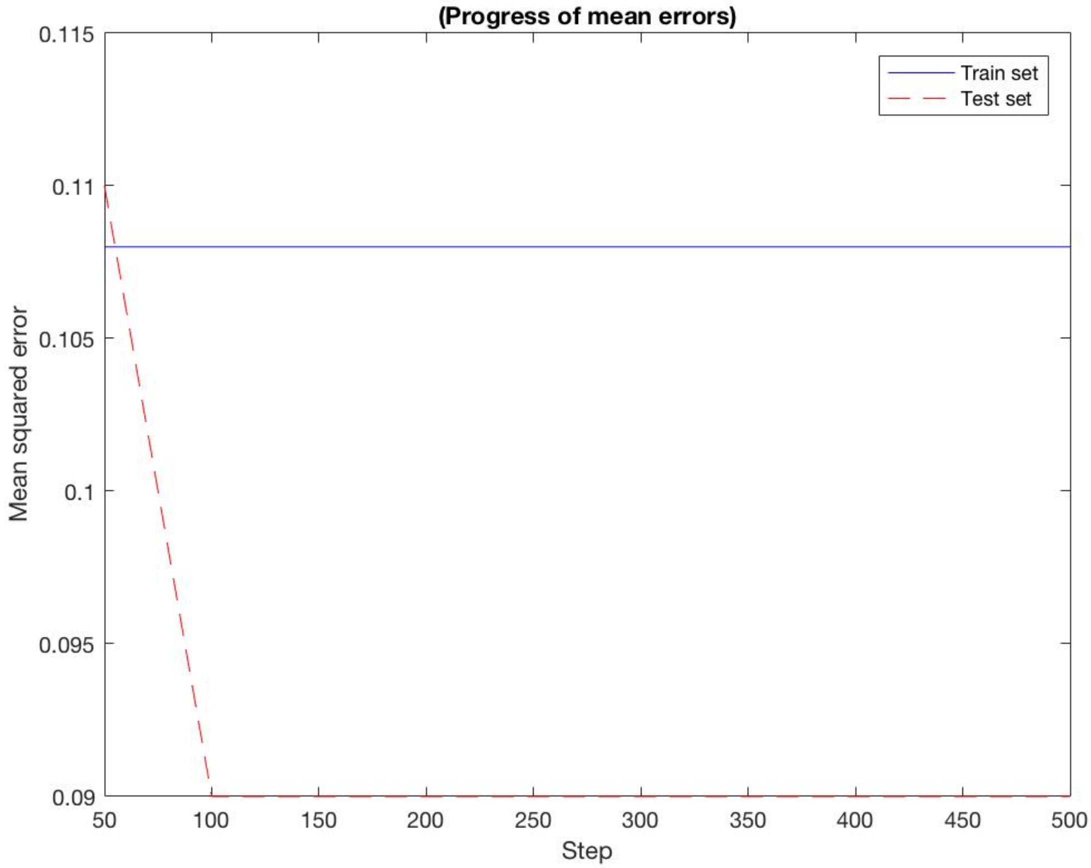


(II)
When 500 steps were used with a learning rate of 0.05
Weights: <w0 = 1.0086, w1 = 3.0950, w2 = 2.8938>
Mean misclassification train = 0.1080
Mean misclassification test = 0.0900



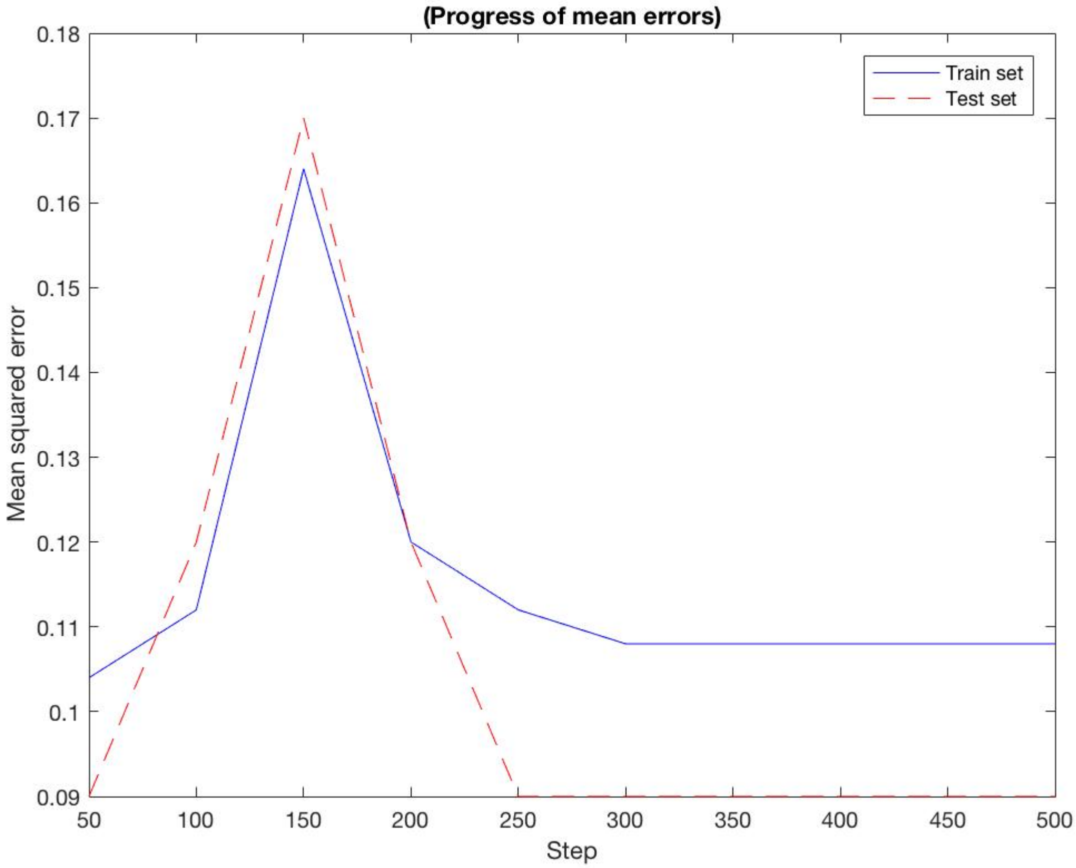
The minimum error found was calculated for both the train and test data within the first 50 steps.

When 500 steps were used with a learning rate of 0.01
Weights: <w0 = 1.0086, w1 = 3.0949, w2 = 2.8938>
Mean misclassification train = 0.1080
Mean misclassification test = 0.0900



The minimum error found for the train data was calculated within the first 50 steps, while the minimum error found for the test data was calculated between the 50^{th} and 100^{th} step.

When 500 steps were used with a learning rate of 1/k^(1/2)
Weights: <w0 = 1.0086, w1 = 3.0950, w2 = 2.8938>
Mean misclassification train = 0.1080
Mean misclassification test = 0.0900



This run was interesting as the error calculated for both data sets had a significant peak at the 150^{th} step, however, the minimum error found was calculated for both data sets by the 300^{th} step.

I found it interesting that the minimum error found for both data sets was consistent in all runs in **(e)** regardless of alterations, however, the run from part (I) had a different weight vector than the one shared by all three runs in part (II).

Problem 3.

(a)

Written and submitted as GLR_online.m

(b)

Written and submitted.

This first run used 500 steps and a learning rate of 2/k.

confusion matrix train:

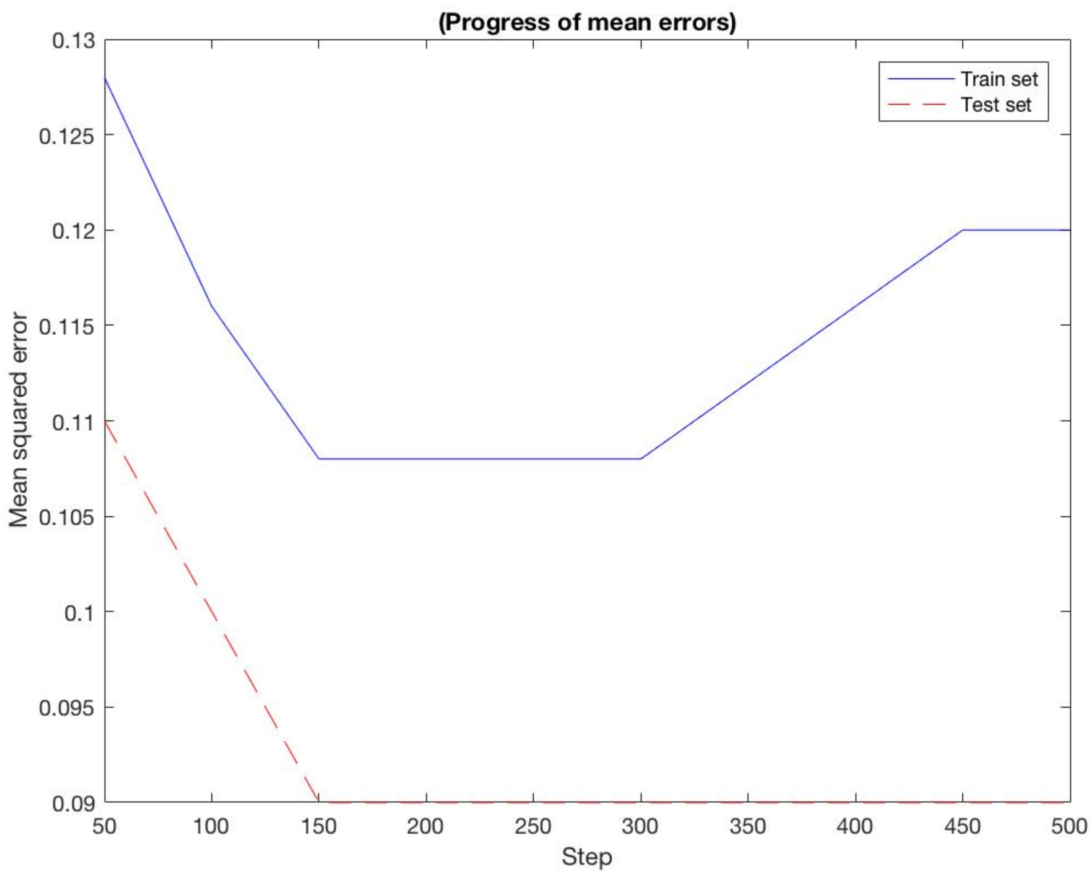
$$\begin{matrix} 139 & 14 \\ 16 & 81 \end{matrix}$$

confusion matrix test:

59	6
3	32

mean misclassification error train = 0.1200

mean misclassification error test = 0.0900



(c)
When 3000 steps were used, with a learning rate of 2/k
Weights: <w0 = 0.8142, w1 = 2.0299, w2 = 1.6487>

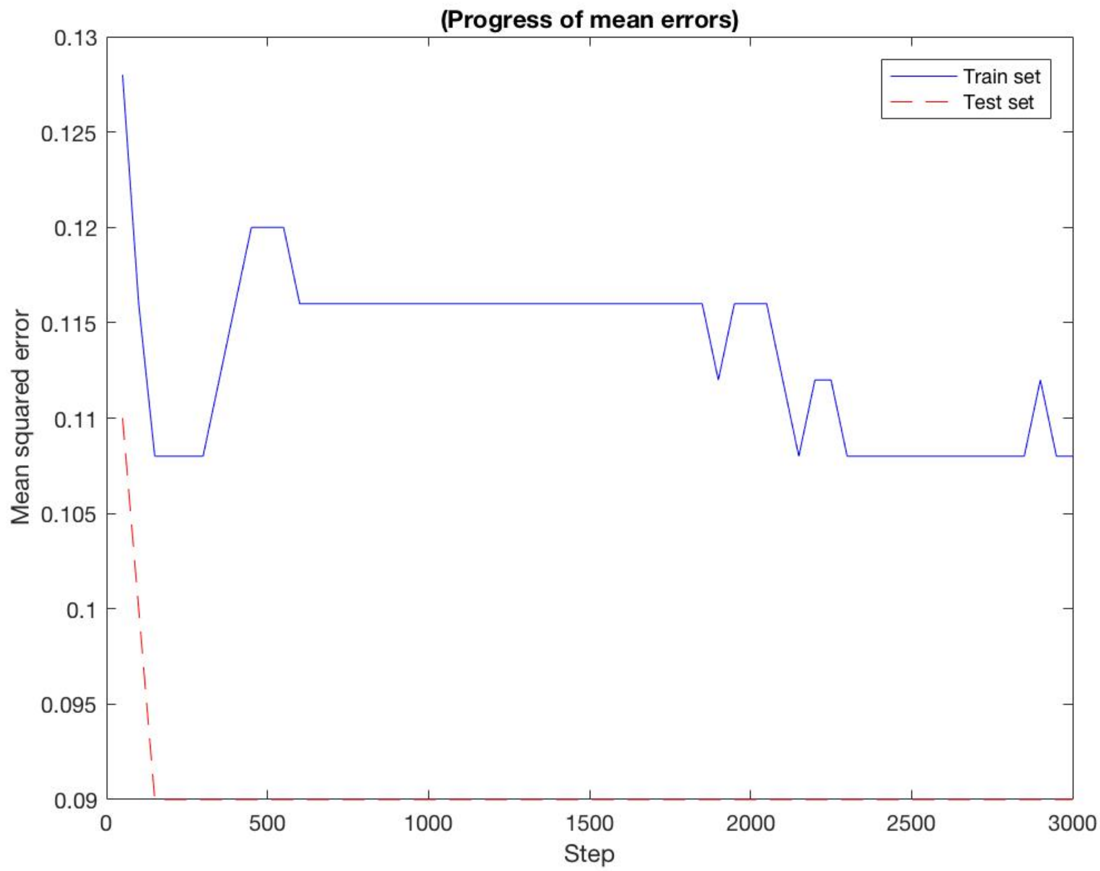
confusion matrix train:

139	11
16	84

confusion matrix test:

59	6
3	32

Mean misclassification train = 0.1080
Mean misclassification test = 0.0900



When 500 steps were used, with a learning rate of 0.05
Weights: <w0 = 0.7351, w1 = 2.0662, w2 = 1.9576>

confusion matrix train:

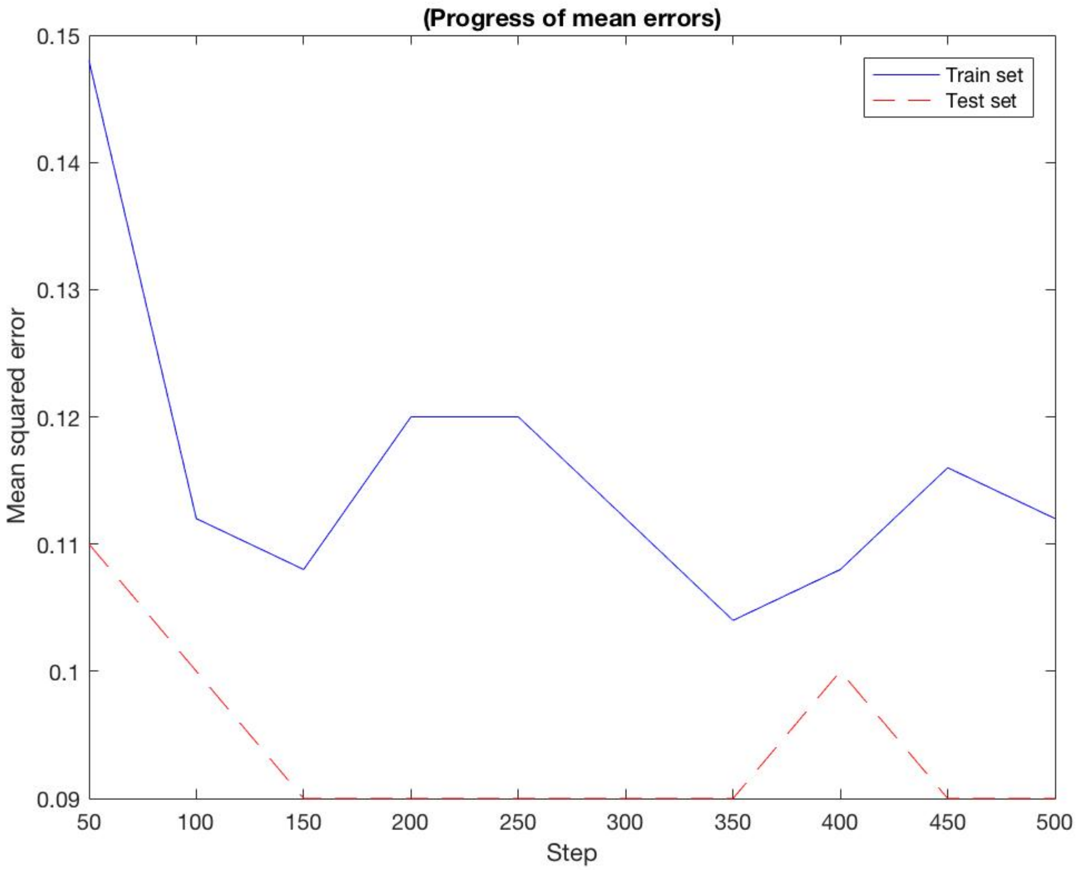
139	12
16	83

confusion matrix test:

59	6
3	32

Mean misclassification train = 0.1120

Mean misclassification test = 0.0900



When 500 steps were used, with a learning rate of 0.01
Weights: <w0 = 0.7566, w1 = 1.4571, w2 = 1.2532>

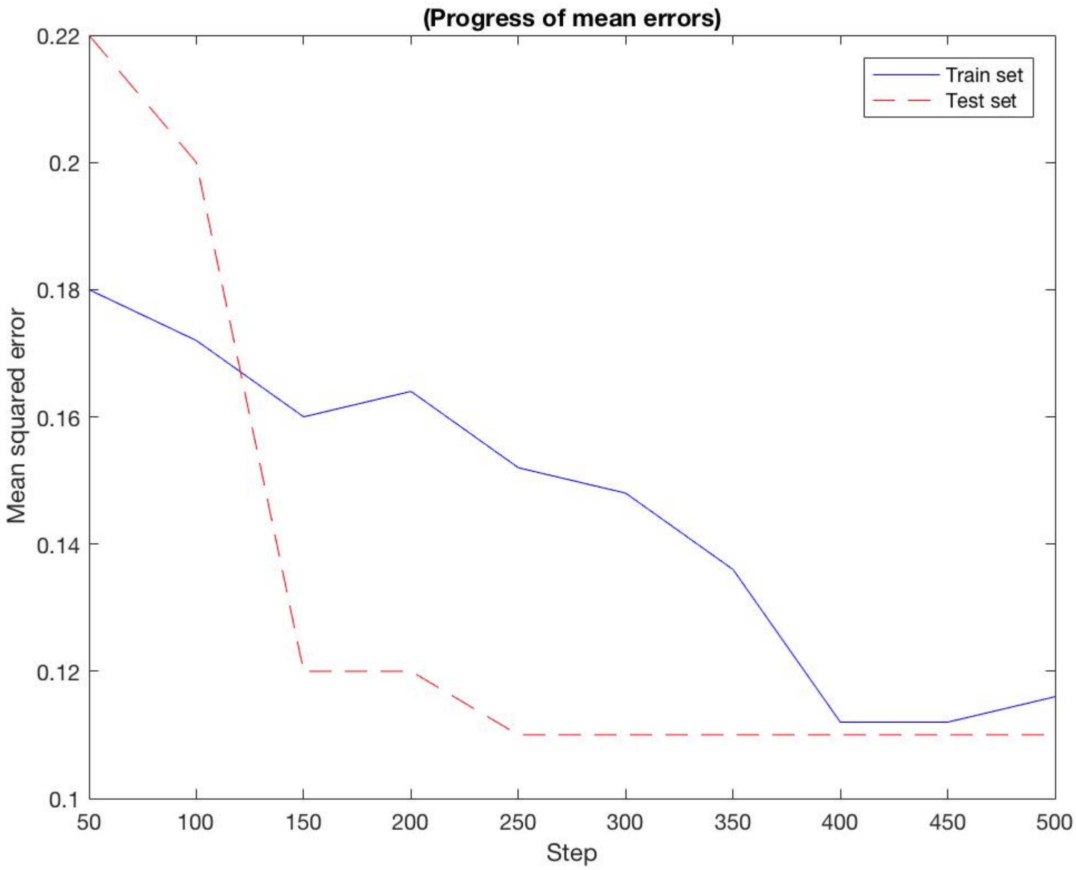
confusion matrix train:

142	16
13	79

confusion matrix test:

59	8
3	30

Mean misclassification train = 0.1160
Mean misclassification test = 0.1100



When 500 steps were used, with a learning rate of 1/k^(1/2)
Weights: <w0 = 0.7905, w1 = 2.3549, w2 = 2.2735>

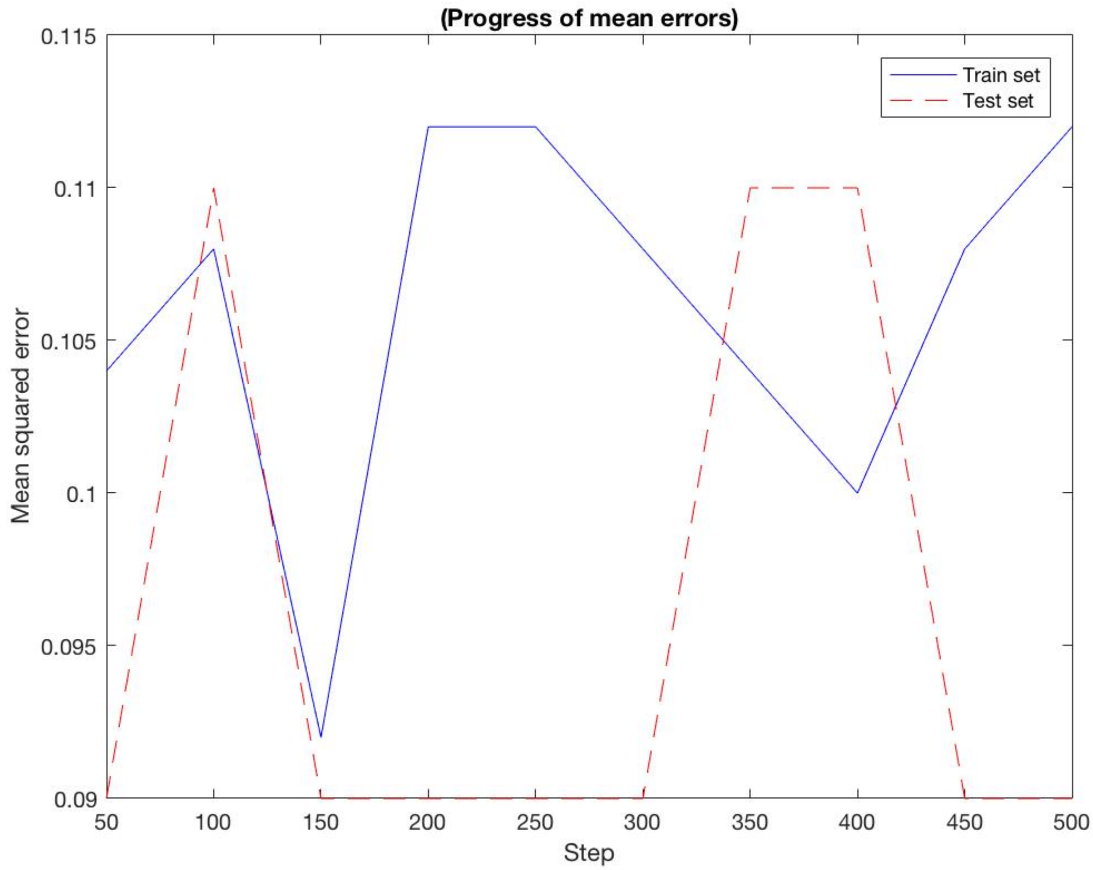
confusion matrix train:

139	12
16	83

confusion matrix test:

59	6
3	32

Mean misclassification train = 0.1120
Mean misclassification test = 0.0900



It appears that both gradient methods can find the same respective minimum of error for both datasets. The batch gradient method is more computationally expensive; however, it appears that the error it finds only ever improves or stays the same over time; it never gets worse/higher. The online gradient method on the other hand is less computationally expensive and therefore runs faster, however, it is prone to multiple peaks and valleys. The results generated from a model using the online gradient method can vary far more greatly depending on number of steps and gradient method. Based on my results it appears possible to get a worse result from the online gradient method by adding more steps, where as it is not when using the batch gradient method.

Problem 4.

(a)

(a) split data set into two based on class c

c = 0

$$\hat{\mu}_0 = \frac{1}{n_0} \sum_{i=1}^{n_0} X_i$$

c = 1

$$\hat{\mu}_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} X_i$$

(b)
First I would remove the column of class data from the matrix of data with which I am working **without** splitting the data into two datasets based on class. Once I have the single matrix of only attribute values I would input it into Matlab's cov() function. In this case the cov() function would give me the estimate of the covariance matrix that combines class 0 and class 1 examples.

(c)
In order to calculate the prior of class 1 I would divide (# of observations in class 1)/(Total number of observations in data). This would get me the maximum likelihood estimate for $\text{theta}_{c=1}$

(d)
Function was implemented and submitted.

(e)
Function was implemented and submitted.

(f)
Function was implemented and submitted.

(g)
$\theta_{c=1} = 0.6200$

$\mu_0 = [-0.9766, -0.0436]$

$\mu_1 = [0.0156, 0.9416]$

$$\Sigma = \begin{matrix} 0.5006 & 0.3039 \\ 0.3039 & 0.4756 \end{matrix}$$

Train error = 0.1280

Test error = 0.1200

This model performed worse than the model from problem 2. Every run of the model in problem 2 calculated a train error of 0.1080 and test error of 0.0900. Both errors are better than the two found for the train and test datasets respectively by this generative classification model.

Problem 5.

(a)
Function was implemented and submitted.

(b)
Function was implemented and submitted.

(c)
Function was implemented and submitted.

(d)

mu_0_1: -0.9766	sigma_0_1: 0.4065
mu_0_2: -0.0436	sigma_0_2: 0.4292
mu_1_1: 0.0156	sigma_1_1: 0.5762
mu_1_2: 0.9416	sigma_1_2: 0.5342

p(y=1): 0.6200

Train error = 0.1000

Test error = 0.1100

This model performed the best on the train data, and it performed better than the generative classification model from problem 4 on the test data. Both logistic regressions (models from problem 2 and problem 3) performed better on the test data.