Chris Tsuei

cxt240

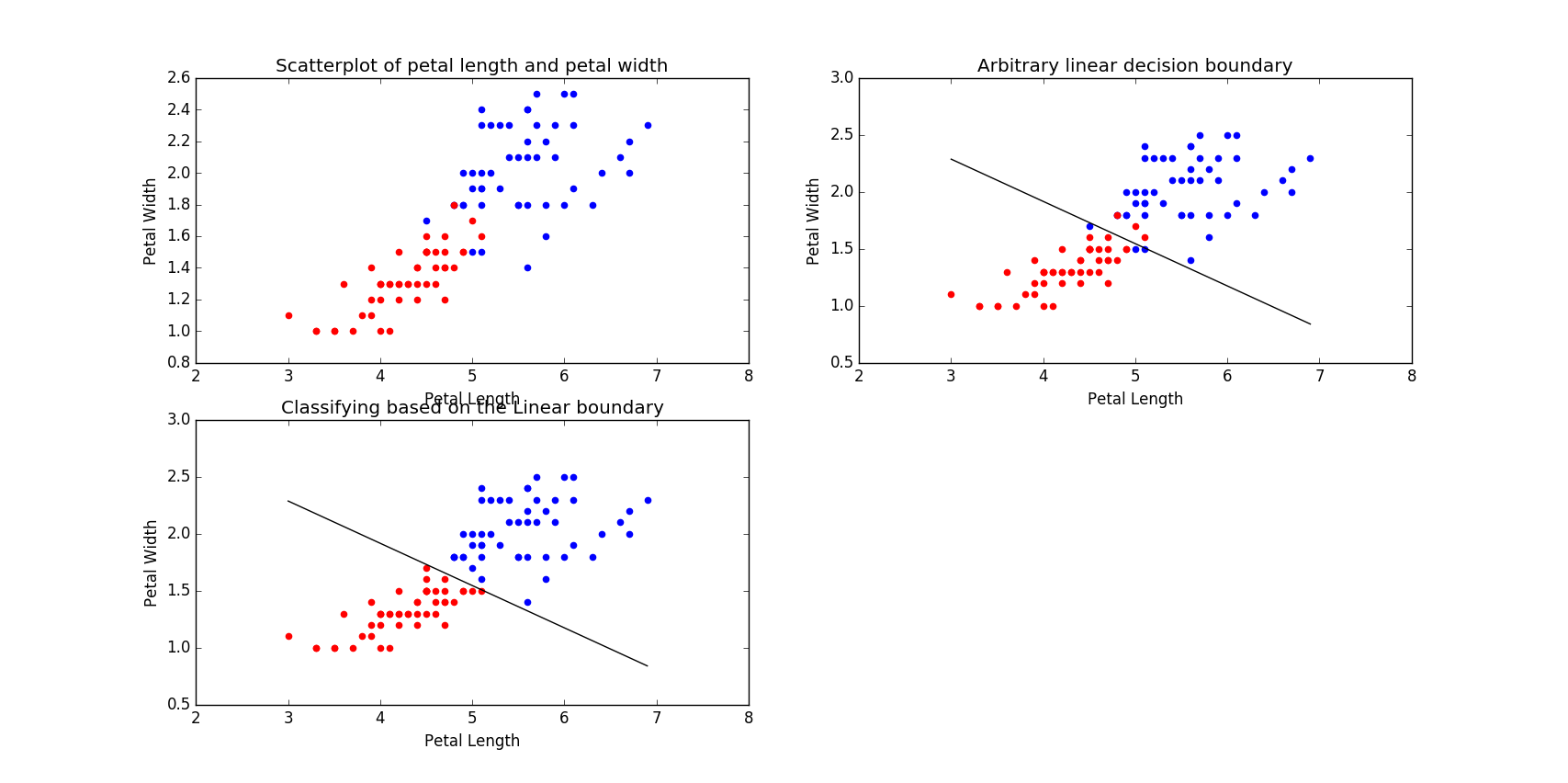
EECS391

12-4-16

Project 2 Write-Up

All of this code and plotting were done with python 3.4 using the matlibplot API on a linux mint computer.

**Question 1:**



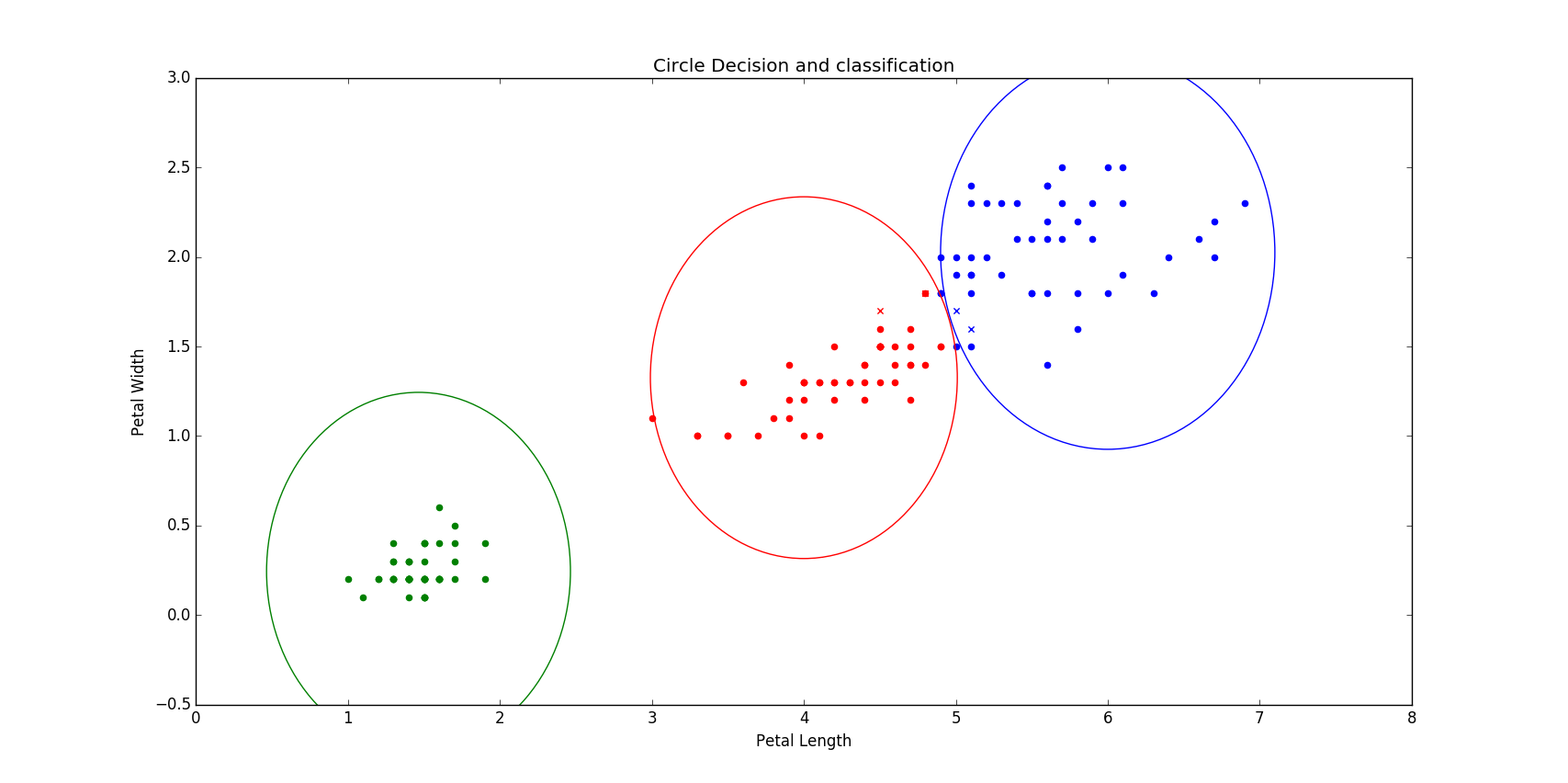
*Figure 1*

**a.)** Upper left plot in the above picture.

**b.)** Upper right plot in the above picture

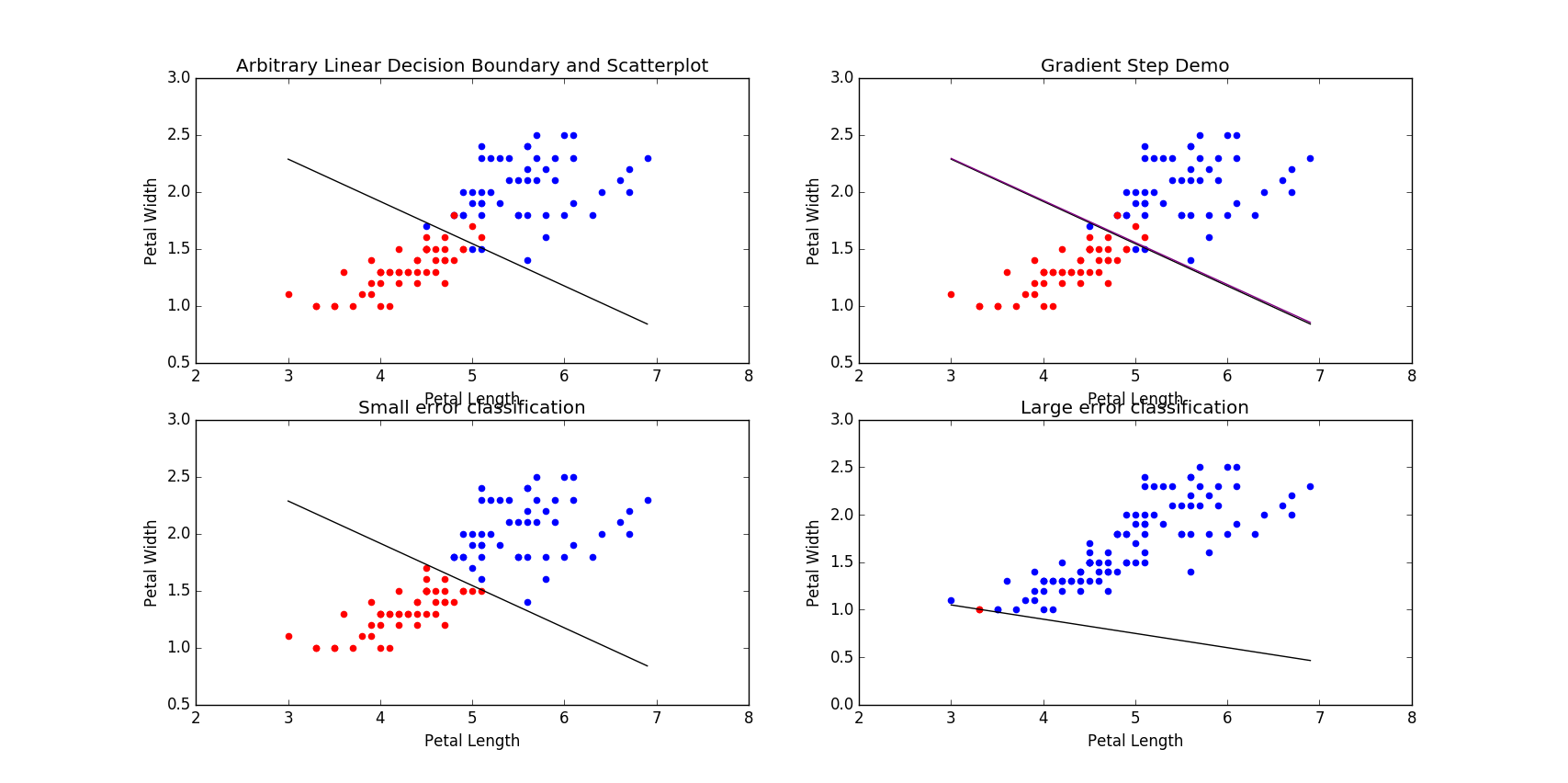
**c.)** Lower left plot in the above picture. The manually predicted decision line was   
y = -0.371x + 3.4. Classification was done by the decision function, which compared expected and observed values and placed each data point accordingly (if the point was above the line, it was blue, otherwise red)

**d.)**



Initially the circle and radius of each category was determined by the center and findRad methods. I used those to base the circles and radius boundaries above. Classification was done by taking each point and seeing if the distance to the center of each circle was less than that circle’s radius. Misclassified points were placed on the figure as x’s instead of dots.

**Question 2:**

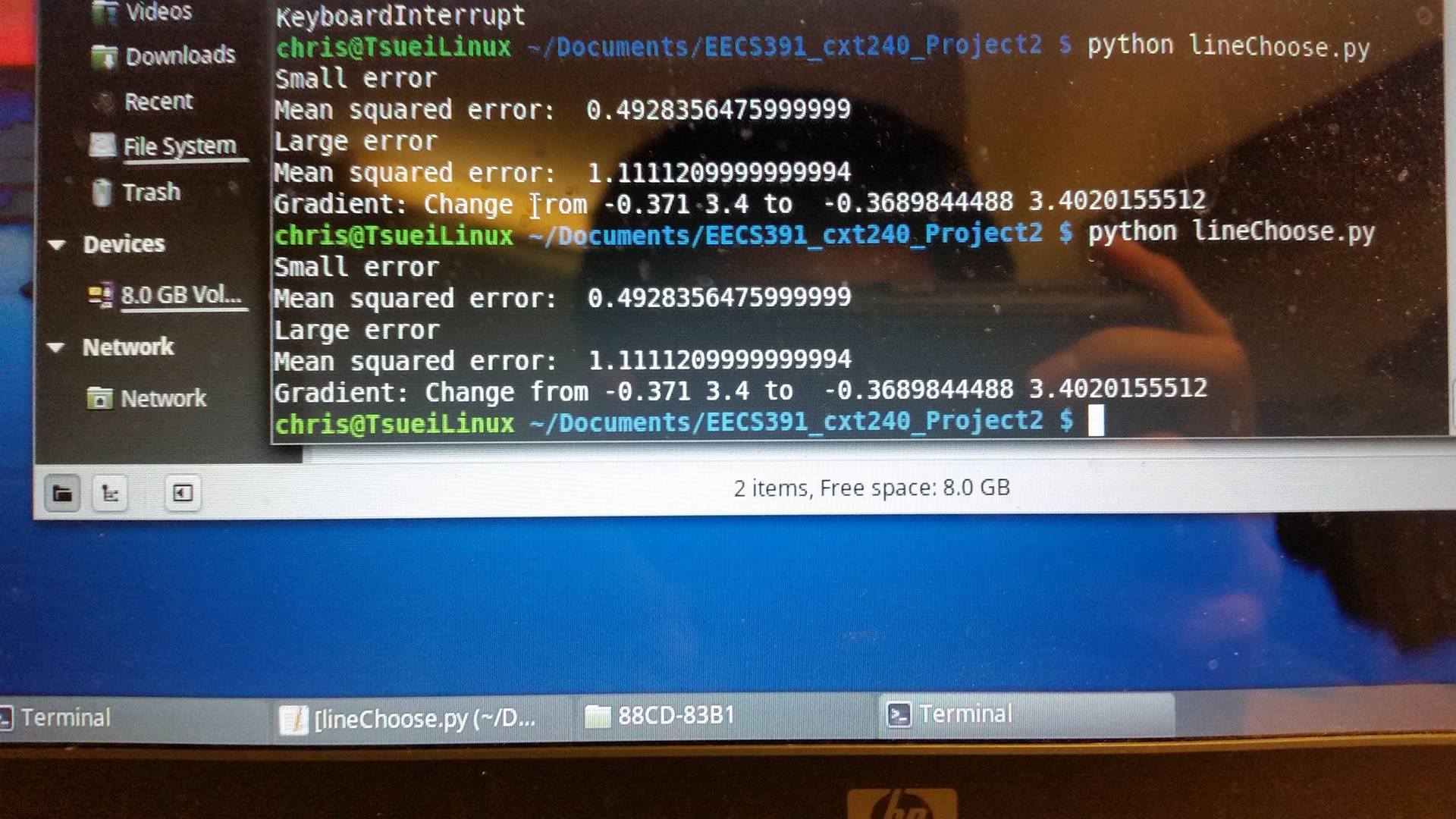
****

*Figure 2*

**a.)** The function to calculate the mean squared error was error(data, boundary, pattern), which took in the data (the read irisdata.csv file), the boundary (in a struct of the form (m, b), where m was the slope and b was the y-intercept), and the pattern (the two flower types that need to be in the irisdata.csv 5th column). The function calculated first the error for one flower type then the other with respect to the boundary and then added the errors and divided by the number of total points, following the following equation:

where N is the number of points, n is the nth point in the summation, Yi is the observed value and the other Y is the predicted value according to the decision boundary.

**b.)**



*Figure 3*

The small error decision boundary is shown in the bottom left plot of figure 2. The large error decision boundary is shown in the bottom right plot of figure 2. The equation for the small error linear decision boundary was y = -0.371x + 3.4 and the large error decision boundary was   
y = -0.15x + 1.5. The calculated errors are shown twice figure 3 above.

**c.)** To compute the gradient, we need to take the derivative of the objective function with respect to the individual weights. f(x, w) is the matrix multiplication of the 2x1 matrix of the weight function and bias by the 1x2 matrix of the current point and 1, which can be written as f(x,w) = wix+ w0.

f(x, w) is represented by wix+ w0, where the bias is represented by w0. The derivative with respect to w0 would be 1, so the derivative of the objective function with respect to the bias would be:

The derivative of f(x,w) with respect to wi is x. The derivative with respect to wi would be represented by:

**d.)** In vector form, the derivatives for the bias can be represented by:

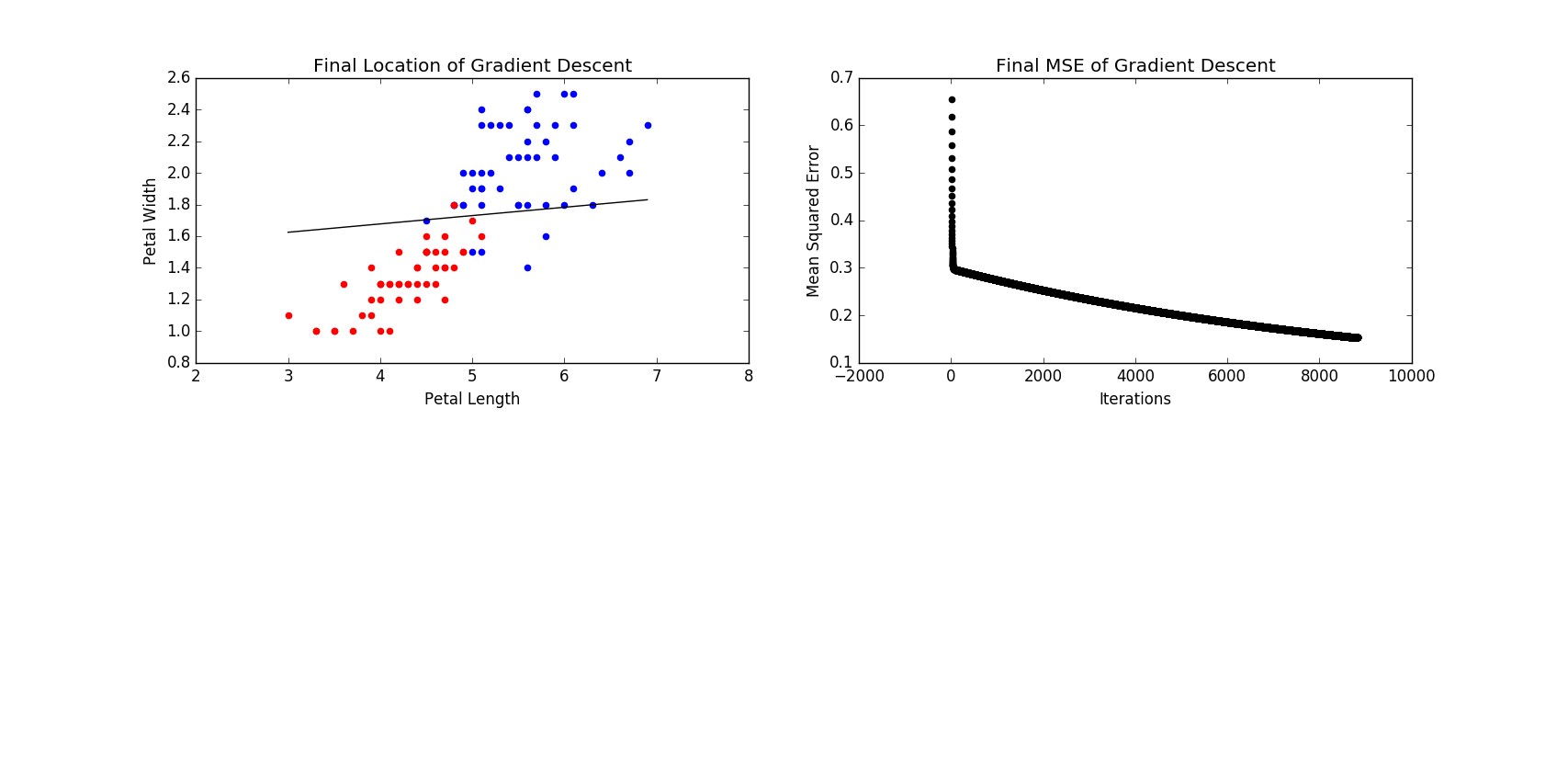
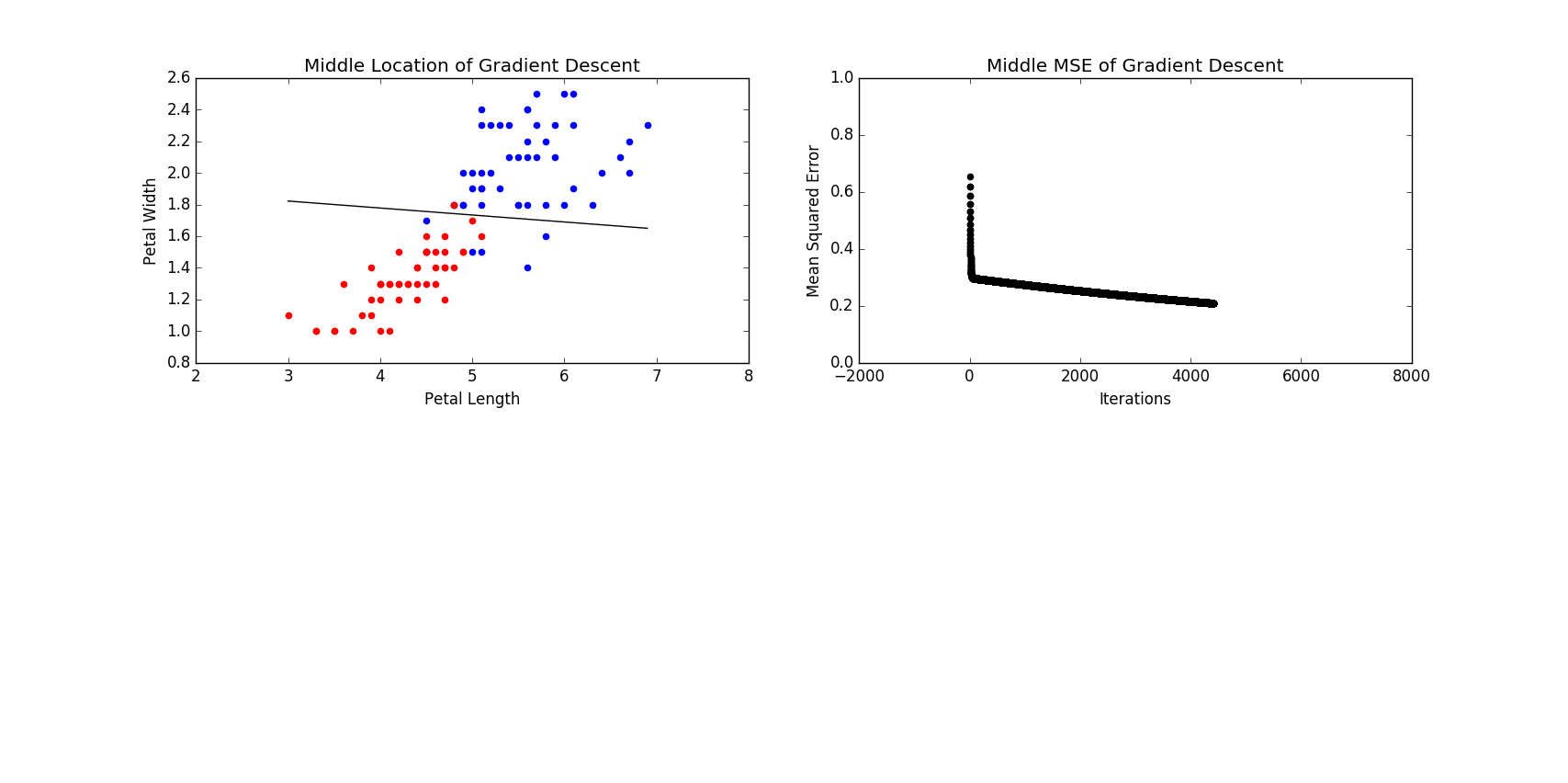
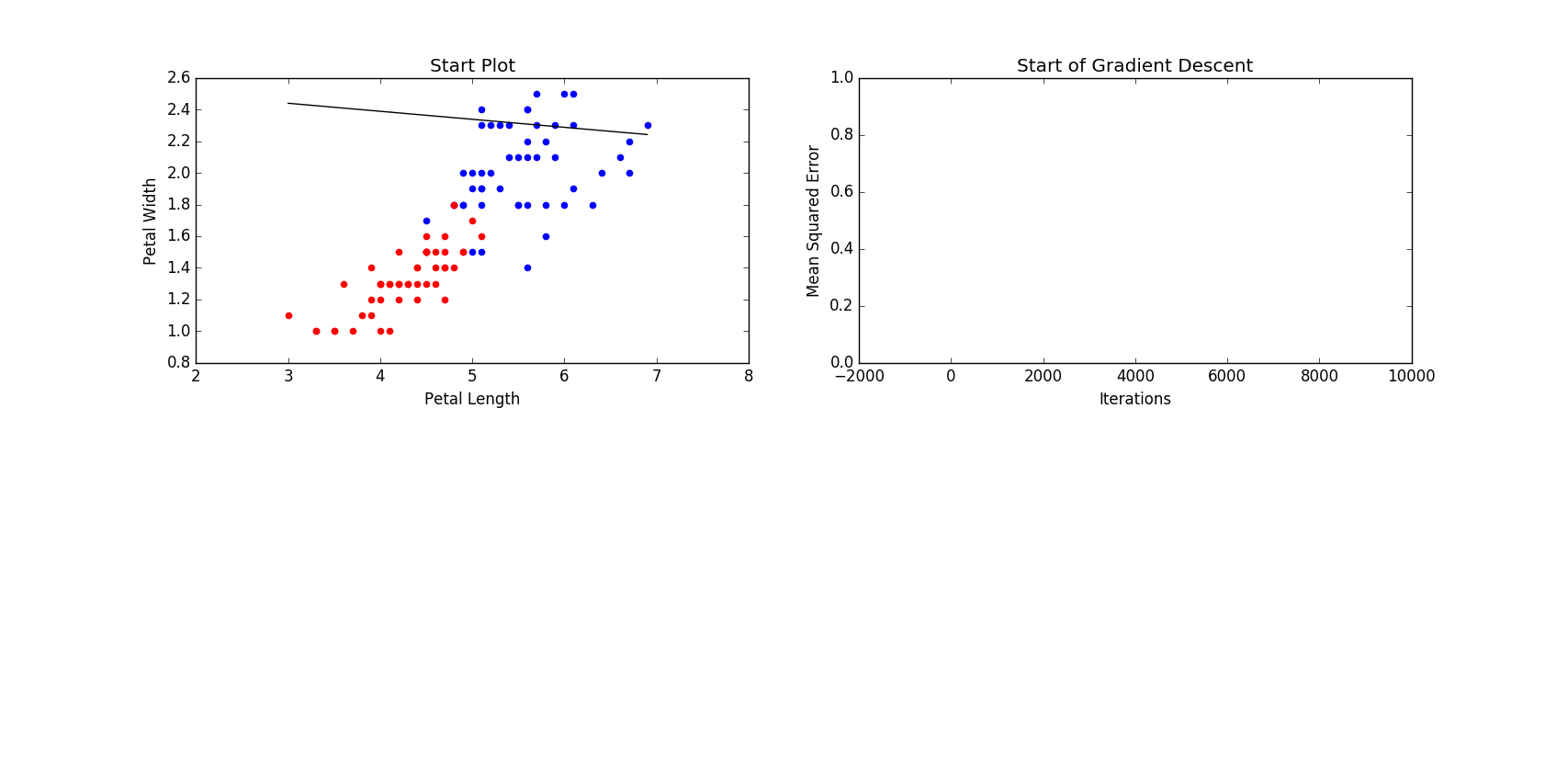
In scalar form, this would be:

**e.)** In figure 2, the step function is demonstrated in the upper right hand plot. I have also included the figure in its original picture with my deliverables. Zooming in, you can see that there is a purple line, which shows the decision boundary changing for a small step. I used the derivations above (specifically the scalar derivation) for computing the gradient (in the code, this is in the function called gradient). The gradient function also outputs the updated slope and y intercept by subtracting the product of epsilon and the derivative with respect to the points from the current slope and intercept. My epsilon value is explained later in this document.

**Question 3:**

**a.)** The function that implements gradient descent is called descent. It takes the data points, a randomized weighted boundary, and the two types of flowers compared and computes a decision boundary using the gradient function.

**b.) and c.)**

****

**d.)** I chose an epsilon value of 0.1/N for my step function, where N is the number of points being classified. To decide this, I primarily tested values in the numerator until I had a reasonable-looking output and number of iterations for the gradient descent function.

**e.)** Stopping criterion was chosen when the mean squared error change was less than 0.00001. Since the error for gradient descent converges, I chose a relatively small value since if the Mean Squared error for the linear decision boundary only changed a tiny bit, then the boundary would be close to the actual solution.