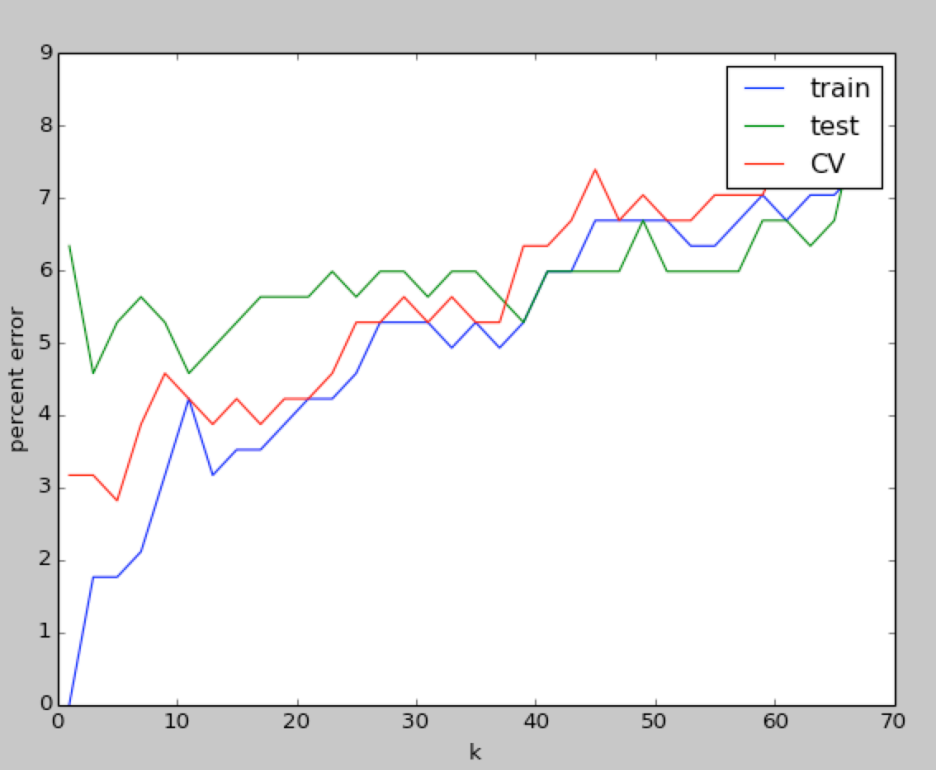
Alex Nguyen – nguyalex
Michael Lee - leemi
CS434

Assignment 2 - Logistic regression with L_2 regularization

1. Part 1 – KNN
    a. Implemented algorithm – done.
    b. Graph of K vs. Error of Training, Test, Cross-validation error



    c. We observe that as K get larger that the error rate rises. Our training data is perfect when compared to itself and with a K of 1. The error rate rises pretty rapidly thereafter. Our test data's error rate stays approximately 6 percent error for the first 70 values of K. Our leave-one-out cross-validation error rate has a somewhat linear climb similar to the training set. Looking at the graph, an acceptable K value would be approximately 37 values of K.
2. Decision Trees
    a. Decision stump:
        i. Feature: 12
        ii. Majority Classifier – Left Branch: 0 Right Branch: 0
        iii. Train Correct Percentage: 64.4366197183
        iv. Test Correct Percentage: 60.9154929577
    b. Decision tree with depth 6:
        i. Tree with info gain per depth level is in tree.txt file
        ii. Train correct percentage: 64.4366197183
        iii. Test correct percentage: 60.9154929577

iv. Compared to the decision stump rates, the decision tree performs the same. These error rates could probably be caused by over fitting to the training data at depth six. From our results, the amount of -1 flags overrun both datasets after the split at depth one. This causes our data to predict very similar to each other. We can see that from the stump we are always predicting zero, ideally we would get further information from splitting on different features, but we still get a prediction value of zero which means that on each feature we've split upon -1 is the majority classifier. This could mean that our data is difficult to split distinctly as opposing classifiers are interspersed with one another, but it seems that we've chosen our splits poorly.

v. Our biggest difficulty was having the data split at certain thresholds. We followed the slide on selecting a threshold for the attribute data and calculated the entropy for each split whenever the classifier changes, choosing the minimum entropy. However, we had some bugs of wrong entropy values from bad data splits that occurred when we entropy was undefined for a certain split.

3. Using decision tree rates to improve KNN
    a. By knowing which feature splits the data, we can use that feature to improve KNN algorithm to increase the division of neighbors in neighborhoods. With this, we can run KNN on the smaller datasets and improve the classification from the decision tree alone. By improving neighborhoods, we can achieve a higher prediction.