I. Data sets

Two classification problems were selected for analysis. The first data set dealt with predicting whether or not a person's income exceeded $50,000/year. Many different groups could be interested in this problem. For instance, several of the attributes used in this data set are readily available via social media and other online resources. A retailer could gather this information and determine what advertisements to display to depending on the predicted income of the person.

Determining income could also be important to government officials and city planners. Government officials could take the data and determine what groups are most at risk for having inadequate income and develop programs to assist them. City planners could use the data to design layouts for new suburbs and to price homes appropriately for the area.

The second data set dealt with determining whether a bank note was authentic or counterfeit given several values derived from analysis of the bank note images. Determining whether a bank note is legitimate is important for legal and business reasons.

Both of these data sets were found on http://archive.ics.uci.edu/ml/.

II. Results and Analysis

A. Decision Tree

1. Income Data set

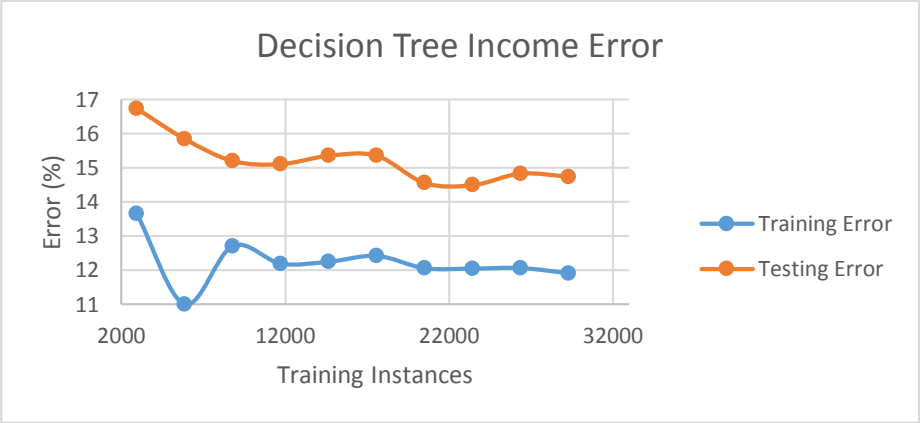

Figure 1: Decision Tree Error

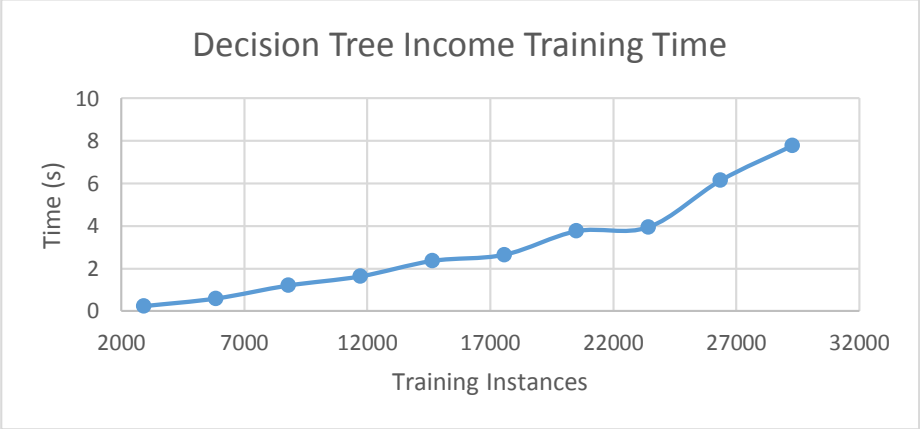

Figure 2: Decision Tree Training Time

Figure 1 shows that there was a general trend of the training and testing error to decrease as the number of training instances increased. This was as expected; as more examples were added, the algorithm should be able to generalize better. The large dip in the training error that occurs around 6500 training instances was unexpected but is explainable. The training examples must have had just the right amount of noise to cancel each other out so that the training error went to zero. Cross-validation would have helped decrease the testing error, but it was not possible to apply the testing set on a cross-validated model using the tool.

Figure 2 shows that the training time increased at an almost linear rate. This indicates that the decision tree did not grow at an exponential rate, but rather in a more linear fashion.

2. Banknote Authentication Data Set

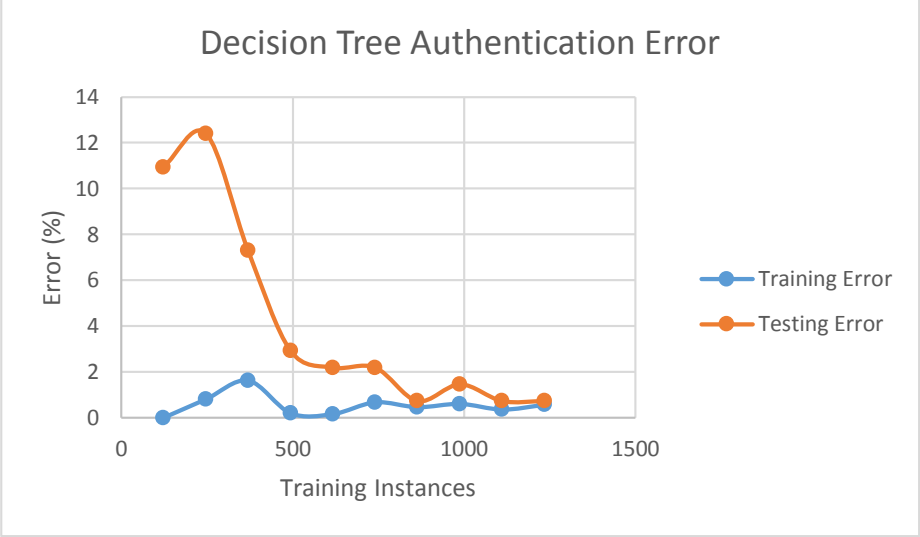

Figure 3: Decision Tree Error

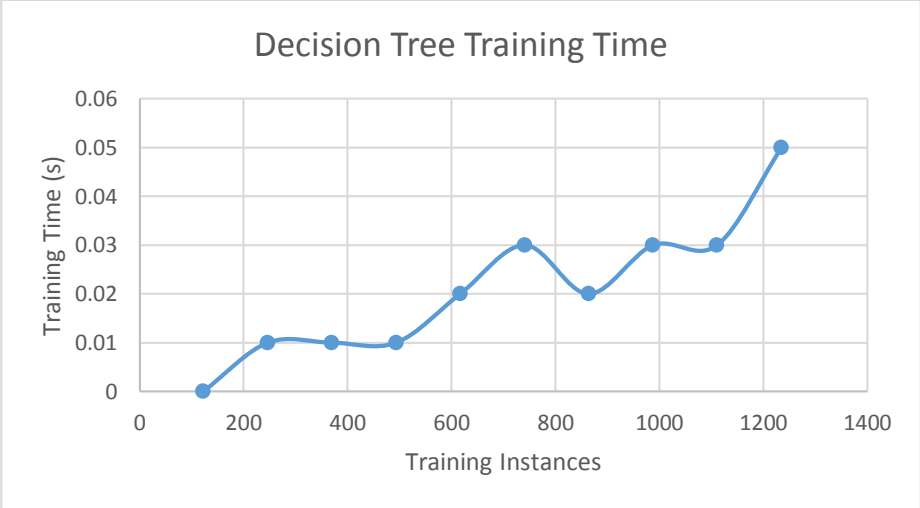

Figure 4: Decision Tree Training Time

For low numbers of training instances, there was a very high testing error. See Figure 3. Generalizing was difficult when there were few examples. However, as the number of training instances increased, the testing error became very low. This graph is important because it shows the power of adding more training instances. More data can, though not always, lead to better results. Since the error was so low, cross-validation would not have significantly improved performance.

The training time was again roughly linear. Since the data set was relatively small, the training time was more susceptible to system processes skewing results. As such, the training time was not monotonic as anticipated. Nevertheless, Figure 4 does show that the tree size was not growing exponentially.

B. K-Nearest Neighbors

1. Income Data set

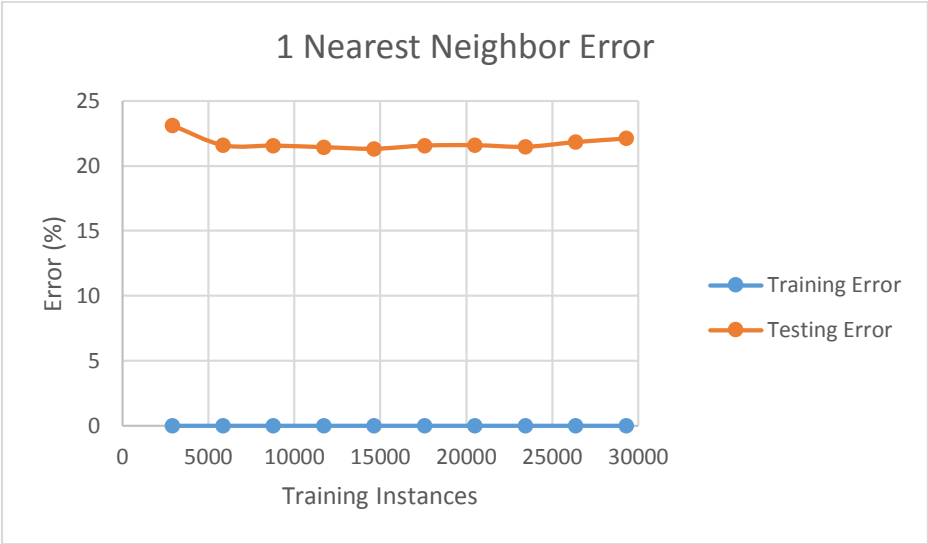

Figure 5: 1 Nearest Neighbor Error

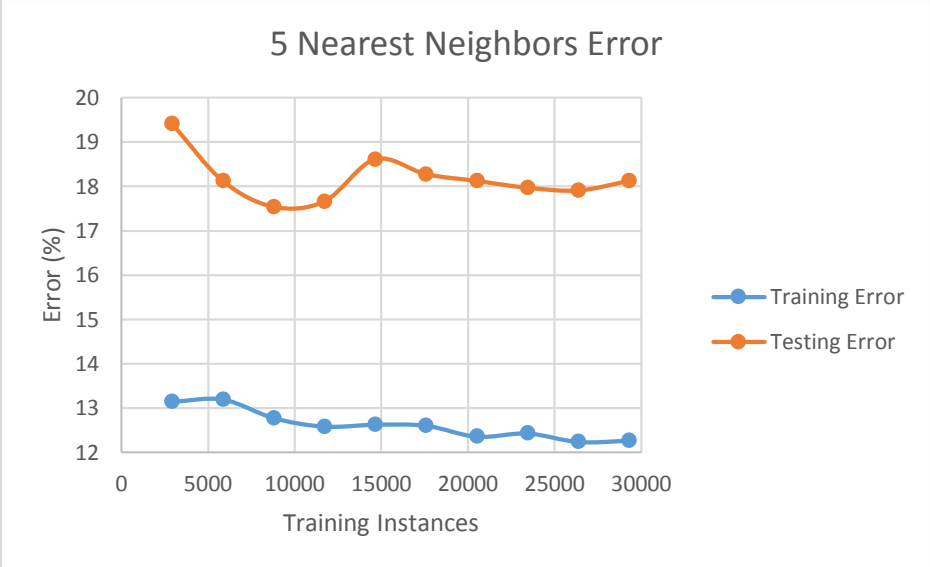

Figure 6: 5 Nearest Neighbors Error

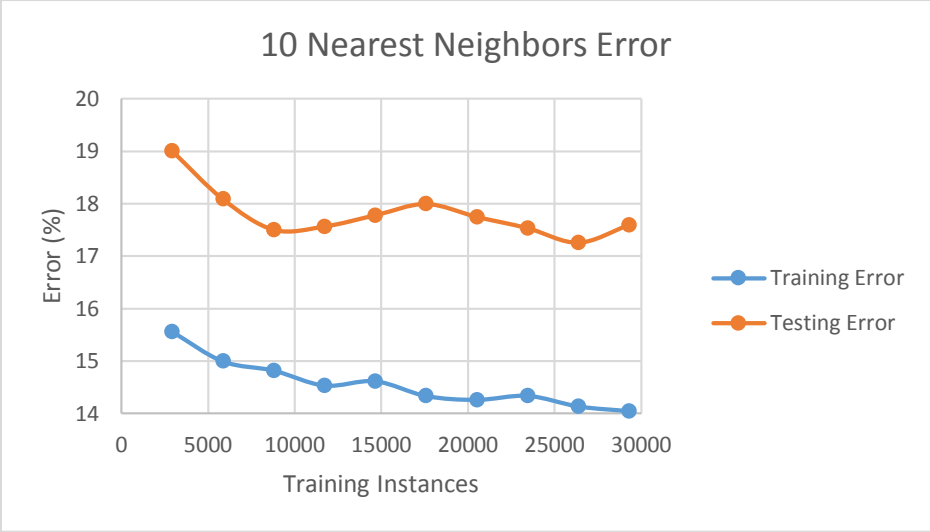

Figure 7: 10 Nearest Neighbors Error

Figure 5 through Figure 7 not only show the importance of choosing a value of k for kNN but also some of its limitations. The first graph shows that 1NN was not a very good choice. While the training error was 0, the testing error was around 22% no matter the size of the training set. Choosing values of 5 or 10 for k significantly reduced the testing error, but still left it in the range of 17-19%. Therefore either there were not many training instances close to the testing instances or nearby instances were not labeled similarly. This data set had a large number of attributes with many being continuously valued. Thus, even the closest training instances could have been relatively far away in a dimension or two. On the other hand, applying kNN could just show a lack of domain knowledge in the choice of the distance metric. Performance could have been improved by weighting the distances (none was used to generate these graphs). Experimenting with various distance metrics could have led to different results, but it would have required extensive effort to determine if the results were better.

2. Banknote Authentication Data Set

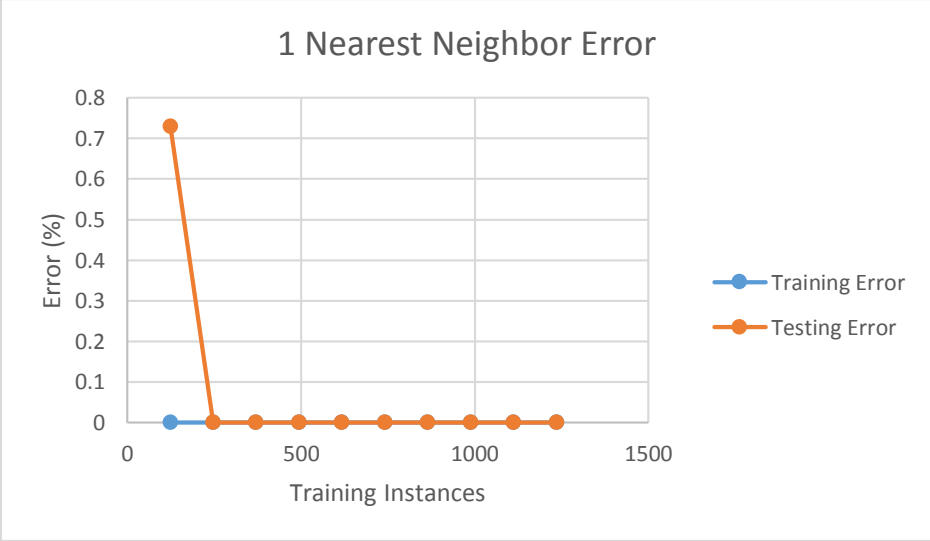

Figure 8: 1 Nearest Neighbor Error

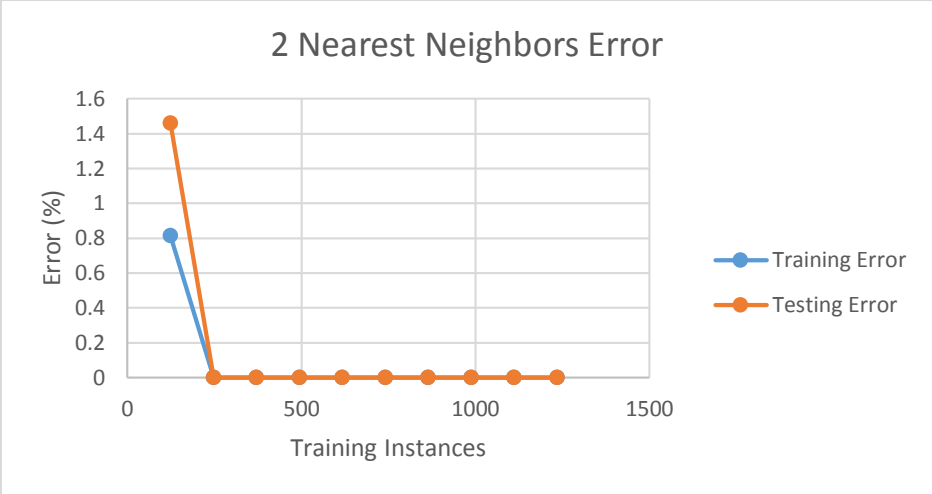

Figure 9: 2 Nearest Neighbor Error

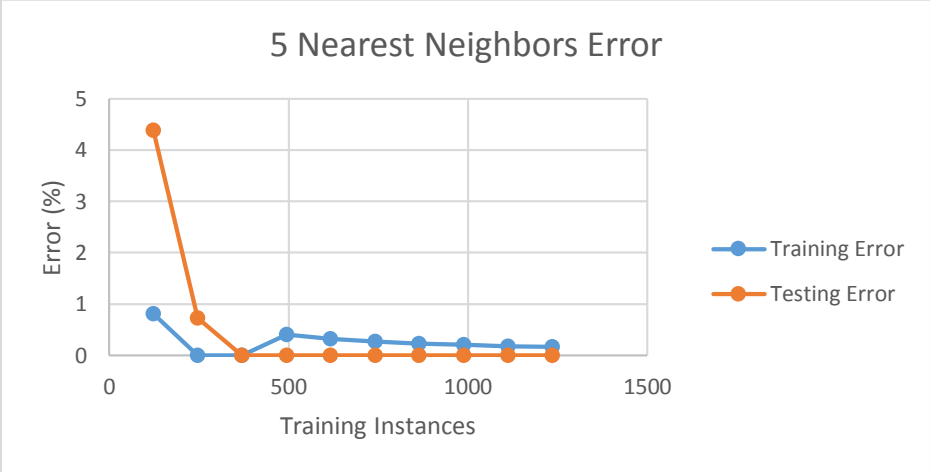

Figure 10: 5 Nearest Neighbor Error

Figure 8 through Figure 10 also show the importance of selecting a k value for kNN. It was important to only include the proper number of training instances when calculating the output for new values and no more or less. For k=1 and k=2 there was no error except for the lowest number of training instances. This shows excellent domain knowledge. Linearly nearby data points were very good indicators of the classification of new data points. However, examining too many training instances generalized too much. In other words, it was not only possible to overfit (trust data too much), but also to underfit (not trust data enough or generalize more than necessary).

C. Neural Networks

1. Income Data set

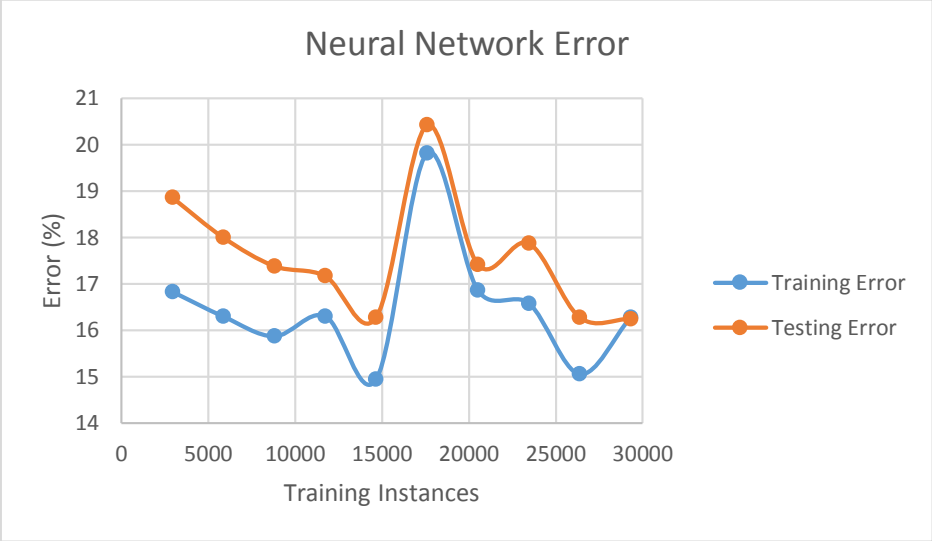

Figure 11: Neural Network Error

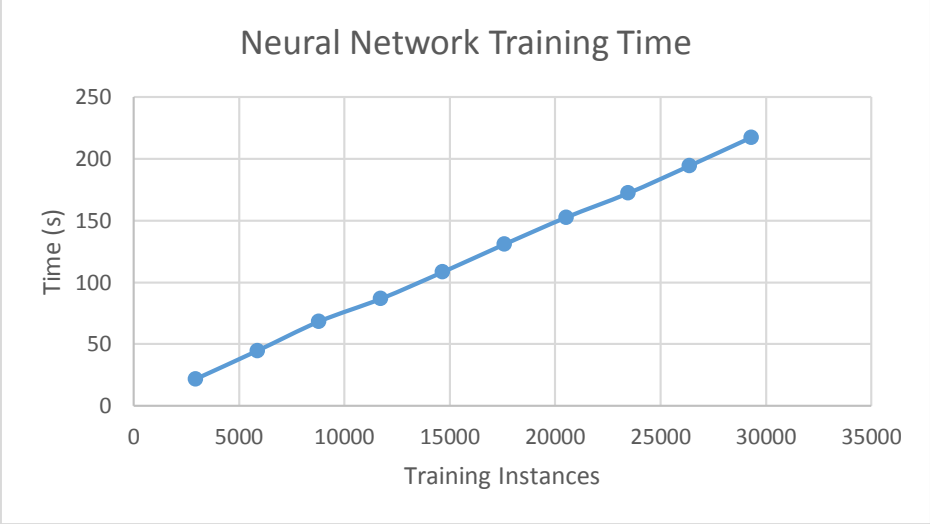

Figure 12: Neural Network Training Time

Figure 11 is interesting because error decreased as the number of training instances increased, then suddenly spiked and decreased again. This could be the result of the neural network hitting a local minimum that was significantly higher than surrounding minima. Performance could have been improved via cross-validation and by adjusting the momentum value to roll through local minima. Increasing the number of hidden layers may have improved performance, but with 2 hidden layers, it should have been able to simulate any arbitrary function.

The training time in Figure 12 was almost perfectly linear as a function of the training instances. Since the number of hidden layers was restricted to two, this was a reasonable result. The neural network training time could not grow much faster unless the number of hidden layers was allowed to change as a function of the input space size.

2. Banknote Authentication Data Set

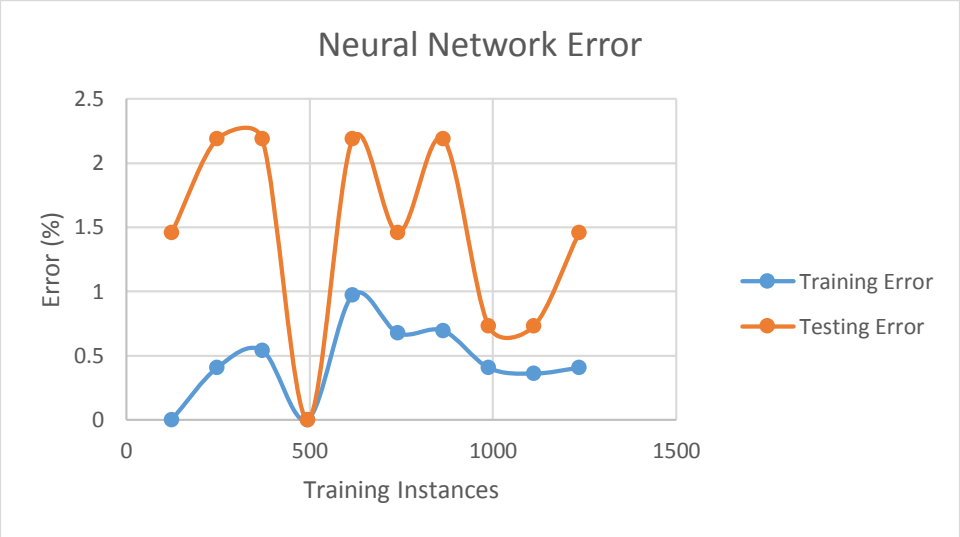

Figure 13: Neural Network Error

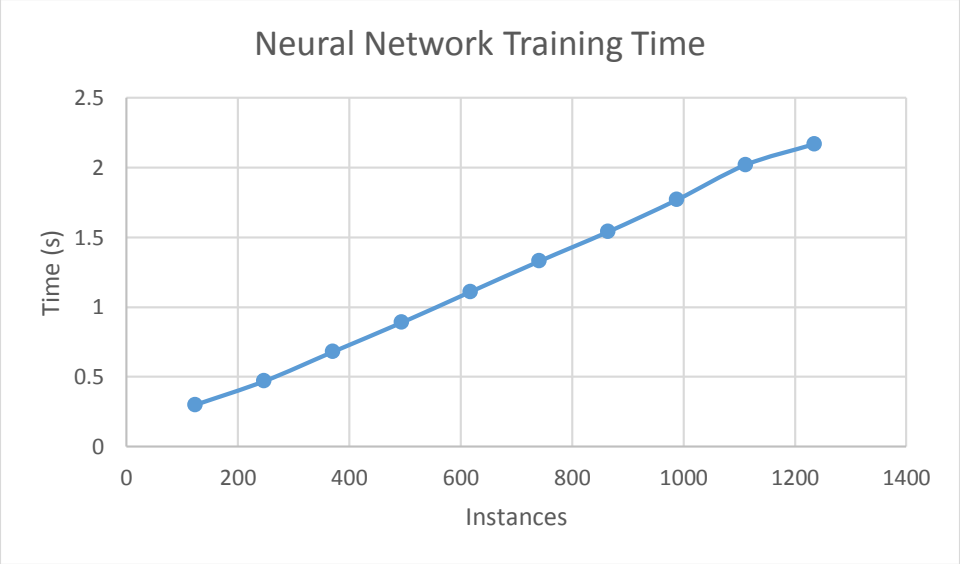

Figure 14: Neural Network Training Time

Figure 13 shows that the error for this data set fluctuated drastically, just as the adult income data set in Figure 11. The neural network was finding local minima rather than global minima. However, error was significantly less than with the adult income data set. This further demonstrates that the banknote authentication data set was much more consistent and contained significantly less noise than the adult income data set.

The training time increased almost perfectly linearly as a function of training instances. This was reasonable since the number of hidden layers was restricted to two. If the number of hidden layers was allowed to change as the number of training instances increased, the training time would not have been linear.

D. Boosting

1. Income Data Set

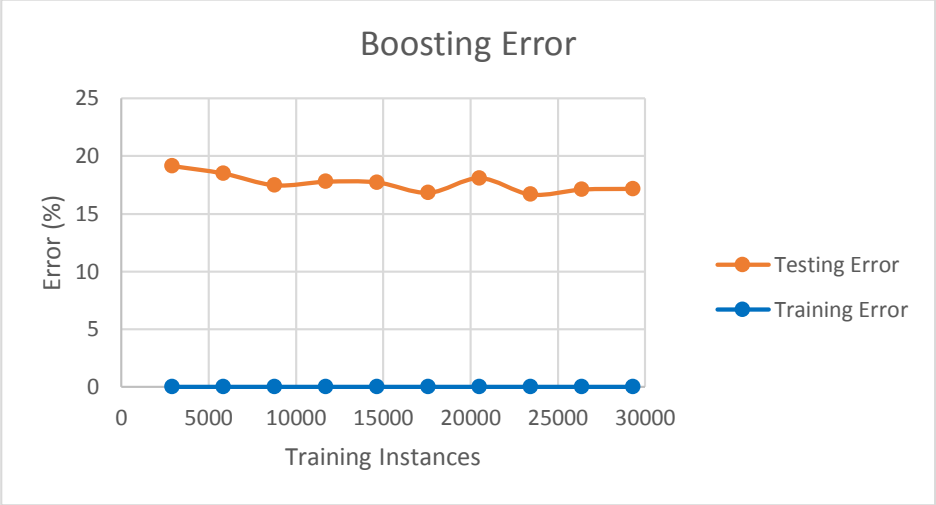

Figure 15: Boosting Error

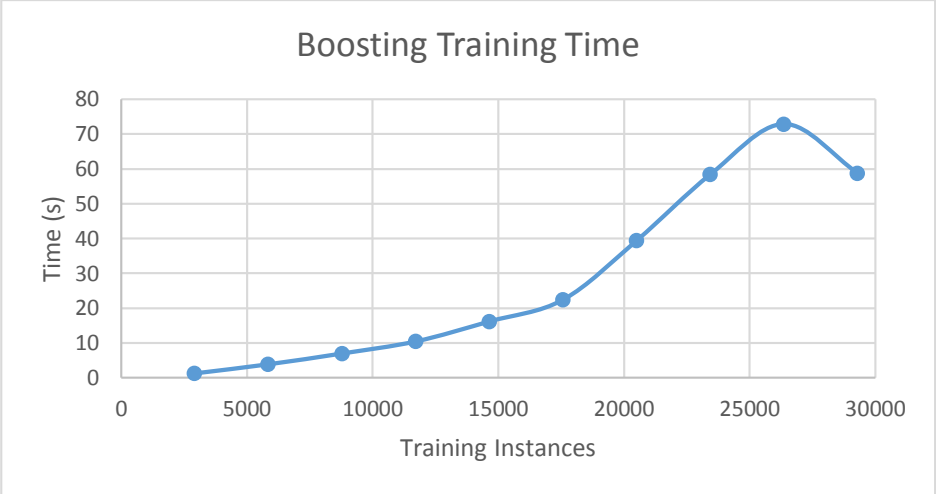

Figure 16: Boosting Training Time

The boosted version of J48 did not perform relatively well on the adult income data set, especially as compared to the original decision tree algorithm. Training time (Figure 16) appeared to increase polynomially with the number of training instances while the testing error did not greatly improve as compared to other learning algorithms. Given the error plot in Figure 15, this looks like a classic overfitting situation. The training error was zero, indicating that the learner fit the training instances perfectly. However, it failed to generalize to other data sets well. Since the boosting algorithm used weak classifiers to build a stronger model, the most likely cause of this behavior was that uniform noise was present in the data set, making it difficult to classify new instances.

Since the training error was zero, it is unlikely that cross-validation would have improved testing performance. The difficulty this algorithm had was generalizing to the testing data set, not in classifying the training data. A trial run with the number of iterations an order of magnitude greater (100 instead of 10) was tested and the testing error was unchanged. Therefore, increasing the number of iterations would not have improved the testing error. The most important decision for this learning algorithm would have

been the selection of the classifier that was used. Perhaps a neural network or kNN would have performed better. This seems unlikely, however, because the decision tree outperformed the other learning algorithms, and the boosted performance was worse than the original decision tree.

2. Banknote Authentication Data Set

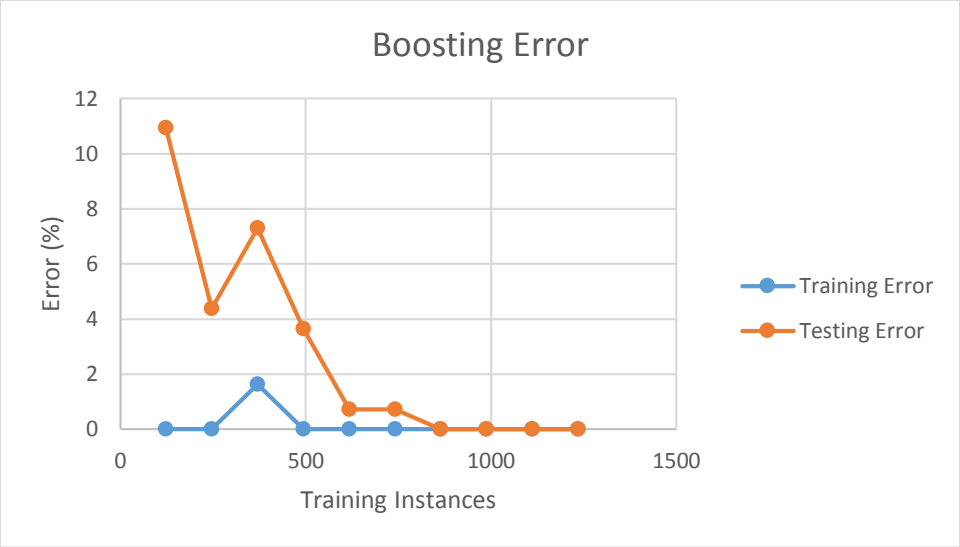

Figure 17: Boosting Error

Figure 18: Boosting Training Time

Boosting performed moderately well on the bank note authentication data set. Testing error was better than both the decision tree and the neural network and consistently improved with the number of training instances (Figure 17). Training time was better than the neural network but worse than the decision tree while still being fairly linear (see Figure 18). With the minimal error observed, it is easy to presume that the data set had very low uniform error.

E. Support Vector Machines

1. Income Data Set

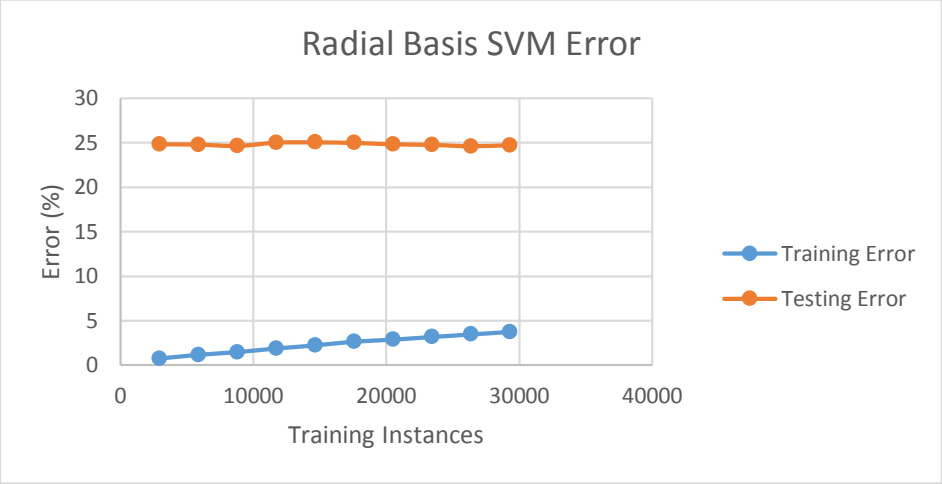

Figure 19: Radial Basis SVM Error

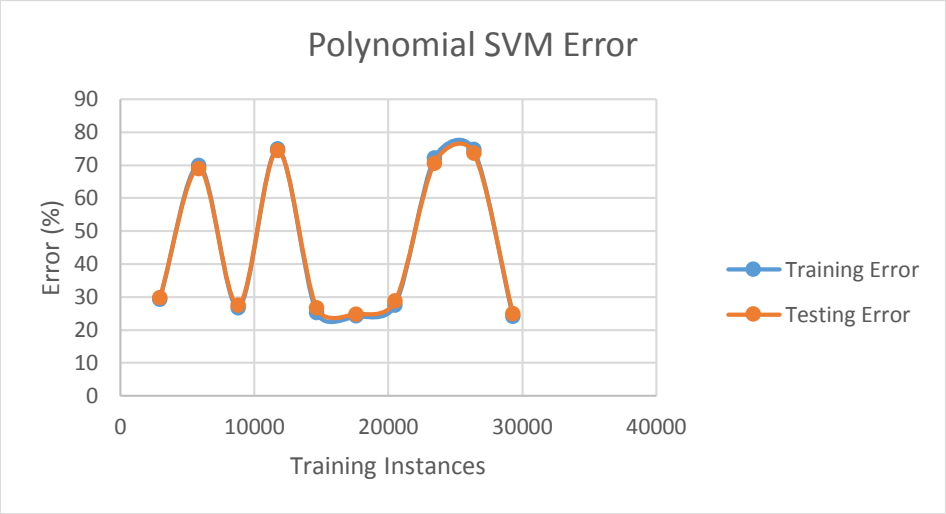

Figure 20: Polynomial SVM Error

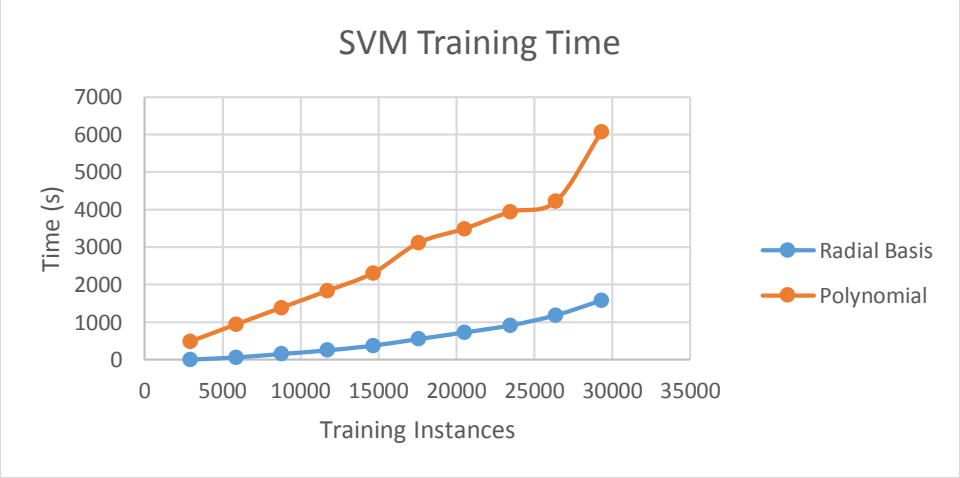

Figure 21: SVM Training Time

The most important choice when working with support vector machines is the selection of the kernel function because therein lie the assumptions about the data. These tests included two different kernel functions: radial basis and polynomial. Neither of them made valid assumptions about the data set because the performance of both was the worst of all the learning algorithms tested (Figure 19-Figure 20). Not only was the training time enormously large, more than an hour for some tests, the testing error was the worst among all the tests run on this data set. See Figure 21.

It seemed that the algorithm had trouble separating the data into distinct groups. Since SVMs work by maximizing the line of separation between the various instances based on the distance metric as specified by the kernel function, it must be the case that data values of both types (above and below 50k) interspersed throughout the data set. Thus, SVMs were ill suited for this data set.

2. Banknote Authentication Data Set

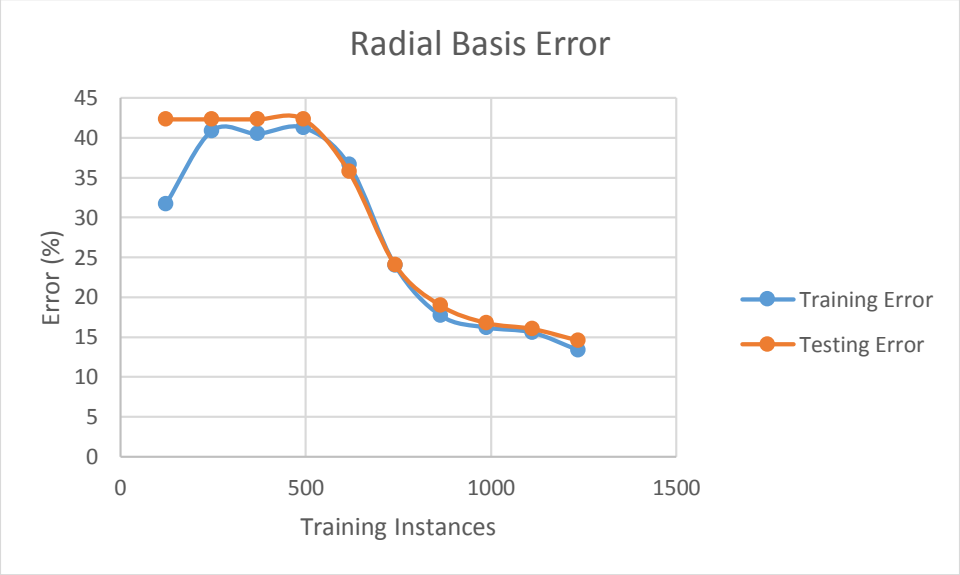

Figure 22: Radial Basis SVM Error

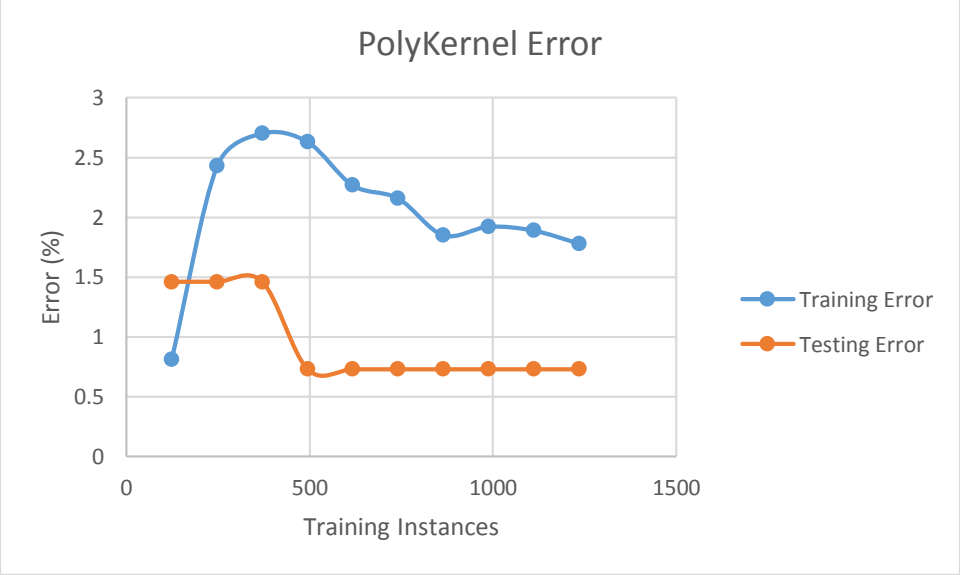

Figure 23: PolyKernel SVM Error

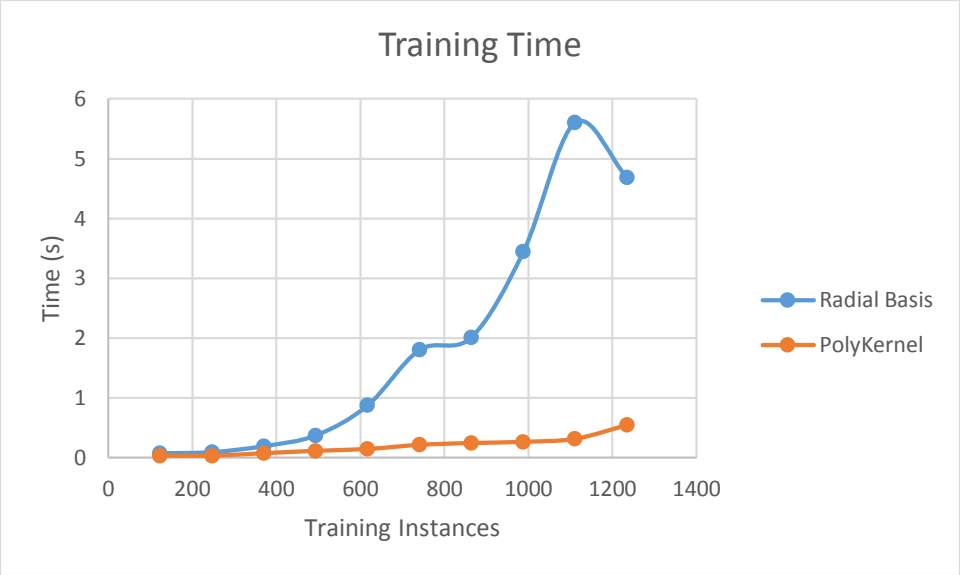

Figure 24: SVM Training Time

The performance of the SVM learning algorithm depended significantly on the kernel function selection. SVM was both the absolute worst and one of the best learning algorithms for this data set.

For the radial basis kernel, the error began extremely high, then decreased drastically as the number of training instances increased (Figure 22). Even though the error decreased, this was still the worst performing algorithm tested on this data set. Even with all the training instances, the testing error still stood at approximately 14%, and the training time appeared to be growing exponentially (Figure 24). The data did not fit the assumptions made that were made in choosing the radial basis kernel.

The polynomial kernel SVM was one of the better performing algorithms tested (Figure 23). It was able to generalize from the training set to the testing set remarkably well; the testing error was below the training error on average. This was a result of how the SVM worked. Since it tried to find the widest margin separating instances of different classifications, it was able to correctly label new data points well. This is further evidence that the data in this set was very consistent because the algorithm was able to find a margin wide enough to identify new values correctly.

III. Conclusions

The best supervised learning algorithm for the adult income data set was a decision tree. In general, decision trees handle large numbers of attributes well while still having reasonable training times, and the same held true here. The decision tree yielded the lowest testing error while having the shortest training time. Testing error could have been further reduced by applying cross-validation to the learning model.

The best supervised learning algorithm for the banknote authentication data set was k-Nearest Neighbors. Not only did the data set score very well in terms of training and testing error, the algorithm is ideally suited for this application. Being a lazy learner, none of the training was done up front; all the training was done at the time of the query. When applied at an ATM or financial institution, there are typically not many banknotes that need authenticated at once. Additionally, there can be long periods of time when no banknotes need authenticated. As such, it was a better tradeoff to only do the training when necessary, especially since only a couple values needed retrieved for each query to obtain the most accurate result.