John Case

Introduction

I chose to analyze the "Adult data set" available in the UC Irvine machine learning repository. The data were gathered via the US census, and the goal is to predict whether a person makes more or less than $50,000 a year based on demographic data. This study has the potential to illuminate the sociological underpinnings of wealth distribution in the US. A more complete understanding of income distribution could help inform future policies which seek to empower disenfranchised subgroups. This data set contains a mixture of nominal and numeric attributes. A total of 13 attributes (including the predicted attribute) were included.

In addition, I chose to analyze the "News data set" available in the UC Irvine machine learning repository. The data summarize online news articles published by Mashable.com over a period of two years. Specially, these feature describe meta-information about the articles, such keyword popularity measured in shares, day of the week it was published on, and the average textual sentiment of the article. The goal is to predict how many times each article was shared given this meta-information. Because the predicted attribute was a numeric variable, it needed to be discretized for classification to work. I binned the variable into a "low" and a "high" class based on whether it was less than or greater than the median number of shares, resulting in roughly 50% of instances in each class.

My approach has two steps. First, I will use a pre-determined cross-validation set in order to find the optimal parameter settings for each algorithm. Second, I will use a different testing set to compare performance between the algorithms in order to account for parameter overfitting in the cross-validation set. This method will help control for overly optimistic error rates in the final analysis and will better estimate how the models will perform on data from the overall population. Using Weka's resample method, I made a 66-17.5-17.5% training-CV-testing set split, resulting in a 19536-6512-6513 split for the Adult data and a 26165-6740-6739 split for the News data.

Thus, aside from the final discussion, "test set" actually refers to the CV set.

Decision Trees

I chose to tune the MinNumObj parameter for Weka's J48 algorithm, an instantiation of the C4.5 method using the information gain splitting method. MinNumbObj dictates the minimum number of instances allowed in each leaf the decision tree. When MinNumObj is small, smaller leaves are allowed, resulting in a more complex tree. Increasing MinNumObj creates a simpler tree. Afterwards, Weka's default post-pruning and subtree raising methods were applied.

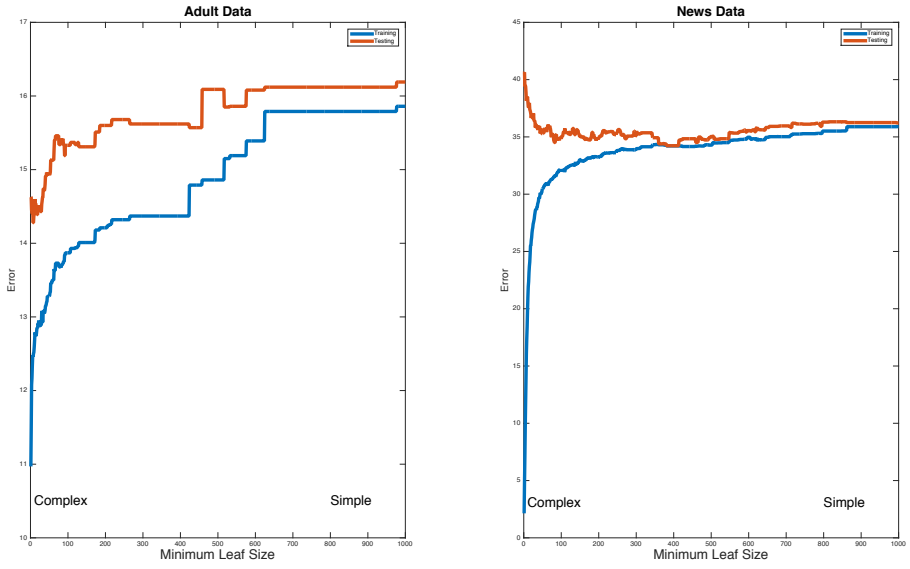


Figure 1. Training and test error as a function of the MinNumObj parameter (1-1000) for both datasets.

The learning curves for the news data (Fig 1, right) follow an expected trend: the complex model overfits the data, causing low training error but high test error. As complexity decreases the learning curves will approximately converge. However, the Adult dataset has a curious pattern (Fig 1): the test set performs the best for the complex model. How can this be so, and why does it not overfit?

The adult dataset has several nominal attributes with many different possible values (i.e., there are 15 unique occupations), where each attribute split produces many different sub-branches. For instance, J48 with a high minNumObj value may decide not to split on occupation because the resulting leaves will be small due to how many different occupations there are. This may prematurely stunt the growth of the tree. Having a low minNumObj paradoxically prevents overfitting because it allows the algorithm to explore the nominal attributes adequately before insignificant leaves are pruned away.

On the other hand, the News dataset has no nominal attributes, and thus each split produces only two sub-branches. As a result, J48 with a low minNumObj will overfit the data, even after pruning. A higher minNumObj is required to prevent a large tree before pruning.

I plotted the error as a function of training set error under the most optimal parameters for each dataset (Figure 2). For the News data (Fig 2b), the training and testing errors eventually converge, whereas curves do not converge for Adult data (Fig 2a). However, the curves in Figure 4a do not approach an asymptote; this may indicate that more training data is necessary for the Adult dataset.

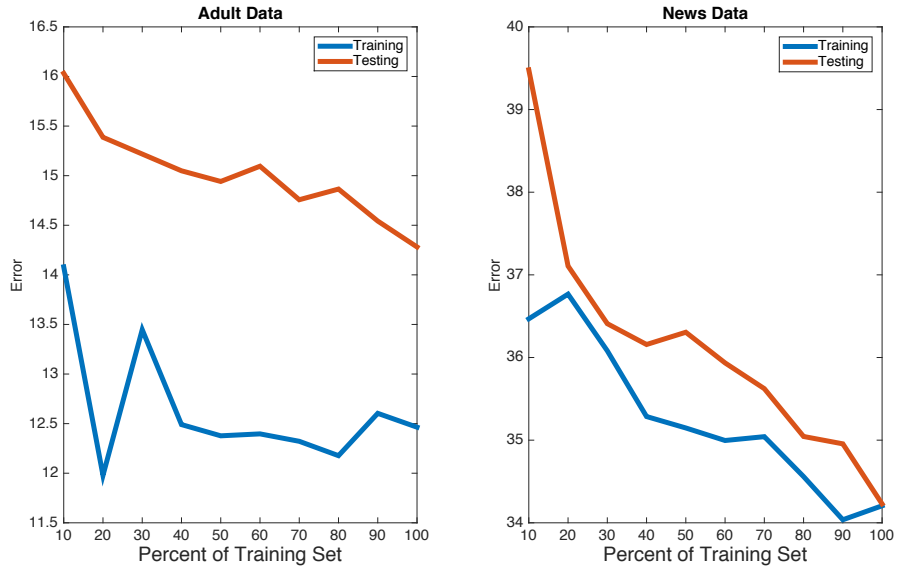


Figure 2. Error as a function of training set size using the optimal parameters (MinNumObj = 7 for Adult; MinNumObj = 381)

Boosting

I used Weka's AdaBoostM1 algorithm with J48 as the weak learner. Using 200 iterations, I tested the same range of MinNumObj parameters to investigate how boosting affects the results presented above.

To my surprise, it reversed pattern identified previously: the testing error for the adult data was highest for small MinNumObj values, whereas the testing error for the news data was lowest for small MinNumObj values (Figure 3). What caused this reversal?

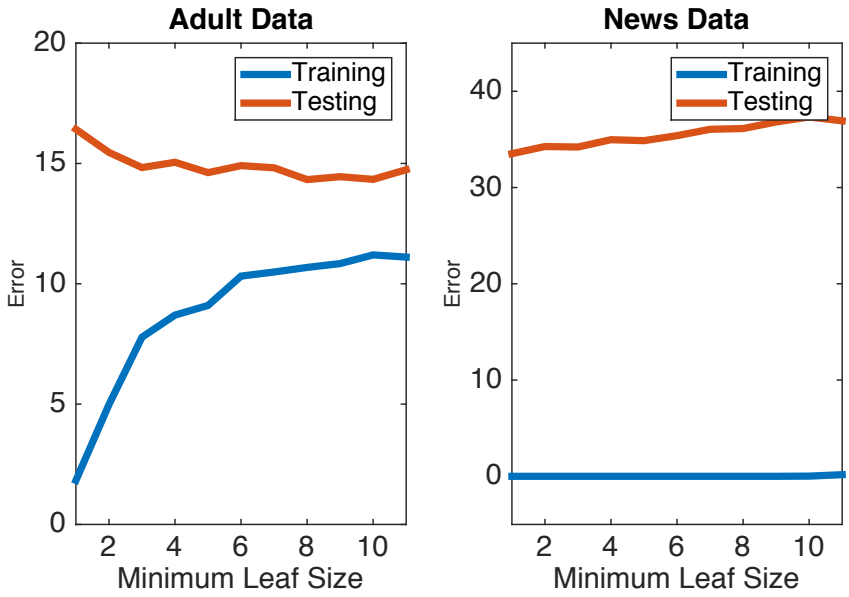


Figure 3

Plotting the hypothesis weights versus the tree sizes may give us a clue. Figure 4a shows that greater weight was given to very complex trees (which presumably overfit the data) in the top 10 most weighted hypothesis when MinNumObj = 1 for the Adult data. However when MinNumObj = 100, less weight was given to large trees. This shows that weighting smaller trees leads to less training error, at least for the New data. Interestingly, the News data shows a similar pattern. However, weighting larger trees more actually improves performance in the News data. Since the weighting patterns are similar between the two data sets, this does not explain the differences in performance.

The answer must lie in how the unbalanced number of instances in each class for the predicted attribute: there are 14833 and 4703 instances for <=50K and >50K, respectively when using Adaboost with MinNumObj=2 and Iterations=10. Consulting the confusion matrix, I found that ~9% of the <=50k were misclassified in the test set, whereas ~38% of the >50K were misclassified. Adaboost may be boosting instances from the less frequent class at the expense of being able to classify instances from the more frequent class. To test this, I used Weka's SpreadSubsample filter to create a subsample of 9406 (48% of the original set) training example in which both classes are equally represented. I ran Adaboost on this subsample, and found that class prediction to be approximately equal: 21% and 18% misclassification for <=50k and >50k, respectively (note that the test set was not subsampled).

Also of note, the Adaboost testing errors are higher than the regular J48 testing errors described above. This result is unexpected, and I can only speculate that the boosting algorithm may be focusing on some noisy or incorrectly labeled data in the training or test set.

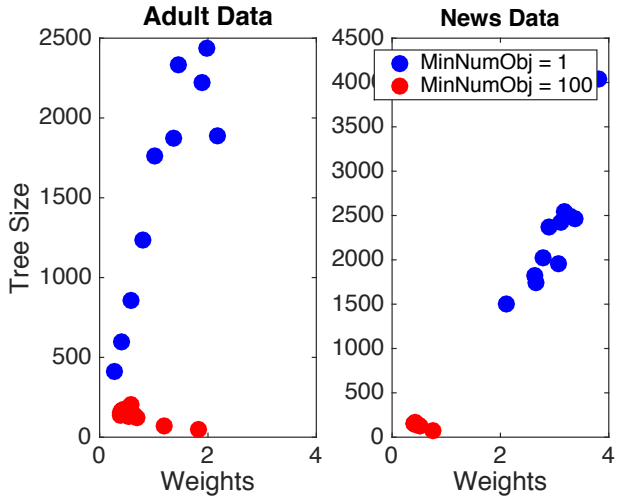


Figure 5

Neural Networks

I was interested in the interaction between learning rates and the number of hidden nodes. Using Weka's MultiLayerPerceptron algorithm, I experimented with a single hidden layer with 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40 nodes, and with learning rates of 0.3 and 0.1. Each hidden layer / learning rate pair was run 10 times with different randomized initial weights. The mean and standard deviations are visualized in Figure 6.

Interestingly, test error does not improve with the Adult data with a learning rate of 0.3 or 0.1, though the training error progressively declines. Somehow the algorithm can find more complex neural networks that reduce training error without overfitting too badly.

One interesting detail to note is that the standard deviation of the 10 runs is much greater when the learning rate is 0.3 compared to 0.1 (Figure 6, top left). This indicates that a minimum can be consistently found when the learning rate is 0.3, even though the weights are initialized differently each run. However, it is very possible that the learning rate may be too big and may be overshooting the minimum. Although the testing curve is similar, the testing curve of the smaller learning rate has a larger standard deviation, which grows with node size (Fig 6, bottom left). Although more iterations would be necessary to tell if a learning rate of 0.1 significantly outperforms a learning rate of 0.3, it is possible that a subset of the runs begin with randomized weights that are closer to the minimum. This subset is able to reach the minimum; however, runs with weights initialized further away cannot reach the minimum due to the small learning rate. Thus, the standard deviation will be higher. Also of note, as node size grows, more minima become available for the algorithm to explore. As a result, the 10 randomly seed runs begin to diverge as they explore these different minima, contributing to increased standard deviation.

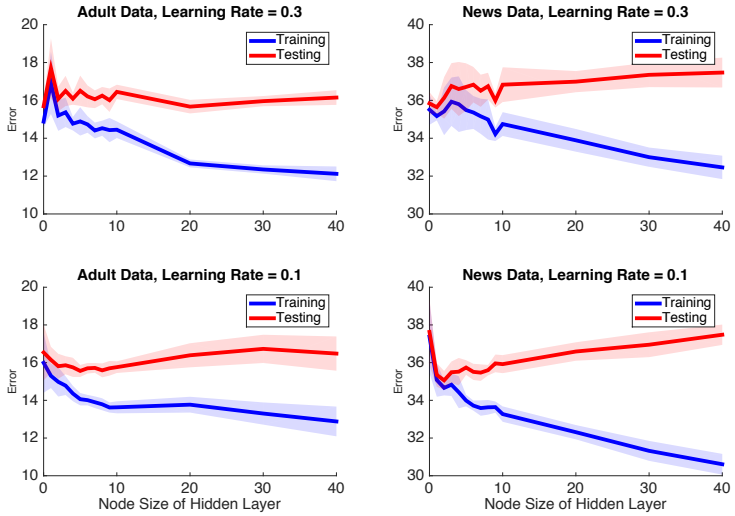


Figure 6

Overall, a learning rate of 0.03 outperforms a learning rate of 0.01 when nodes size is less than 10 for the News data (Figure 7, left) because less minima exist. As a result, a larger learning rate is more likely to converge on these minima faster than a learning rate of 0.01, which might be too slow to reach the minima. However, when node size exceeds 10, a learning rate of 0.01 is marginally better. More nuanced minima exist in this complex representation and a larger learning rate may overshoot the minima (Figure 8, left).

On the other hand, the larger learning rate always outperforms the smaller learning rate for the News data (Figure 7, right). When the learning rate is small, the algorithm may be getting trapped in local minima that the larger learning rate was able to overshoot (Figure 8, right). Perhaps a larger momentum parameter would be able to fix this.

From these results, I conclude that the News data has more local minima than the Adult data, possibly due to the fact that the News data has about 3 times as many attributes as the Adult data.

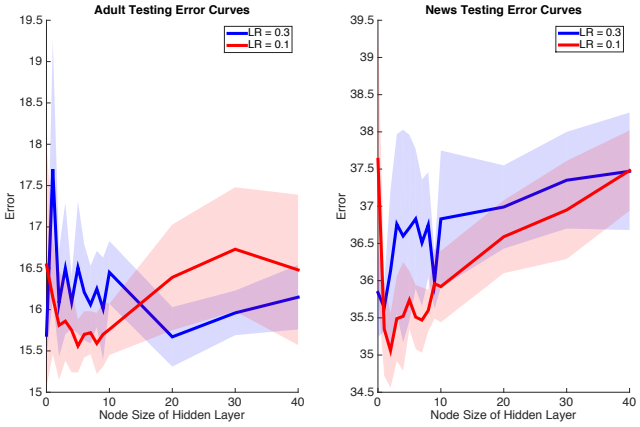
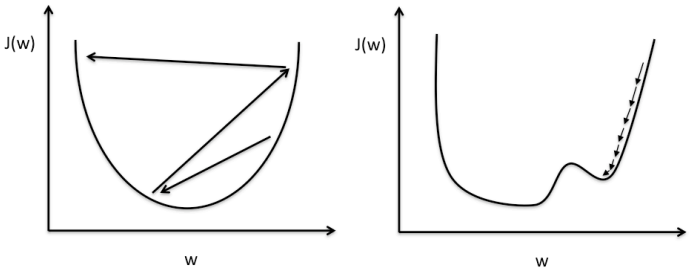


Figure 7



Large learning rate: Overshooting.

Small learning rate: Many iterations until convergence and trapping in local minima.

Figure 8

Nearest Neighbors

Using Weka's iBK implementation of nearest neighbors, I computed the testing error for all k from 1 to 100. I used Euclidean distance as my distance metric.

In addition to experimenting on the full dataset, I chose to also run this experiment on only a subset of best attributes. To choose this subset, I used Weka's CfsSubsetEval attribute selection method with 10-fold cross validation. Only attributes that were selected in all 10 folds were used for this analysis. In total, 5 out of 12 attributes were selected for the Adult data, and 8 out of 58 attributes were selected for the News data. Excluding redundant attributes and attributes with little predictive power should improve kNN performance because kNN has a bias to treat every attribute equally.

In addition to these two data sets, I chose to normalize all numeric attributes so that they range from 0 to 1. Because numeric attributes with larger ranges will tend to influence the distance metric more than attributes on smaller ranges, normalizing will give each attribute a more equal contribution to the distance metric.
As expected, a low k caused overfitting for most datasets (Figure 9). Interestingly, the News data performs the best when all attributes are included and not normalized. The News data also prefers larger k compared to the Adult data. I believe that this is due to the fact that there are fewer homogenous clusters in the News data. That is, entropy will be high for any small subspace in the data. However, taking larger subspaces in the data improves entropy. On the other hand, the Adult data seems to be more homogenously clustered because it has a smaller optimal k.

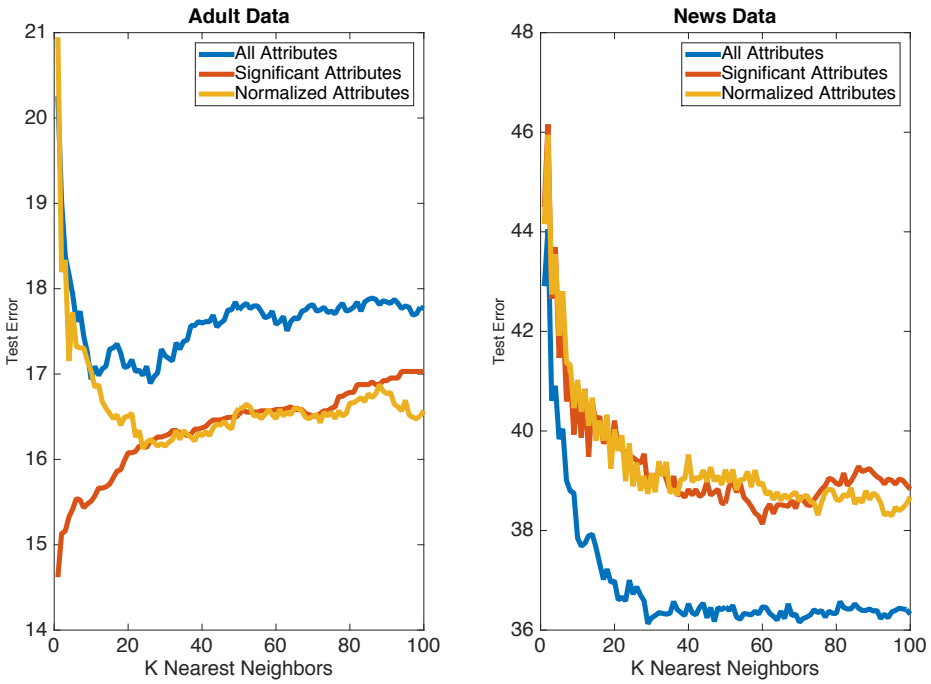


Figure 9

Surprisingly, 1-NN performed the best for the Adult data when only the significant attributes were used (Fig 9, left, red line). I believe that this is due to that for the Adult significant dataset has 3 nominal attributes and 2 numeric attributes on very large scale (i.e., capital gain and capital loss). iBK computes distance between nominal attributes by simply returning 1 if the attributes differ and 0 if they are equal. On the other hand, distances for the capital gain and loss attributes will be many orders of magnitude greater. Thus, the numeric attributes will contribute much more than the numeric attributes. The reason why this improves performance is that the numeric attributes are actually very good predictors of income. For instance, the vast majority of people with income <50K have capital gains less than 10,000 (Figure 10). In this situation where predictive clusters exist in the attribute data, having a small k actually improves performance.

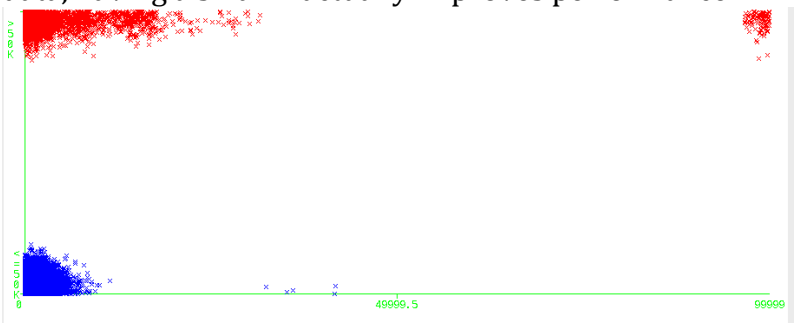


Figure 10. Capital gains (x-axis) separates income (y-axis) fairly well. Notice that very few 50k< instances (blue) have more than 10,000 in capital gains.

SVM

I used Weka's SMO implementation of SVM with two different kernels: Polykernel (which essentially was a linear model) and RBFkernel (a non-linear kernel). I chose to tune the complexity parameter, which controls how many instances are used to construct the set of hyperplanes used by SVM to separate the data. Larger complexity will have softer class margins.

For the Adult data, the linear kernel outperformed the non-linear kernel, suggesting that the dataset may be linearly separable to some degree. Interestingly, the RBF and linear kernels had comparable performance for the News data. Also, both datasets performed the best with highly complex support vectors.

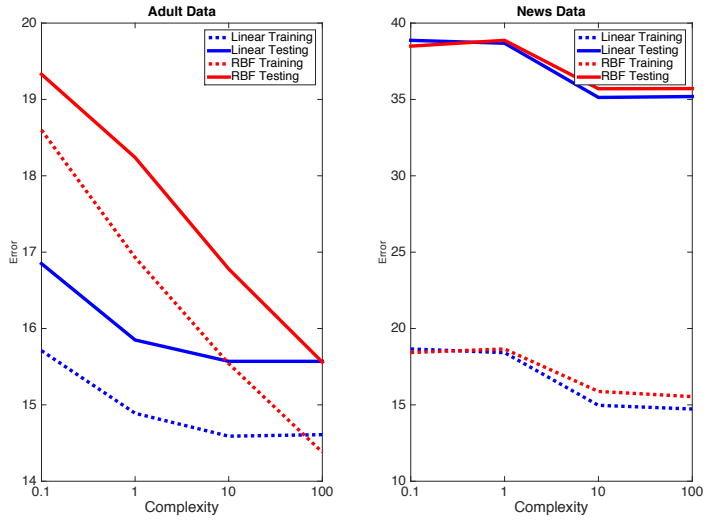


Figure 11

General Discussion

I will now to turn comparing the performance of each algorithm (tuned for the best parameters) on a previously unseen test set.

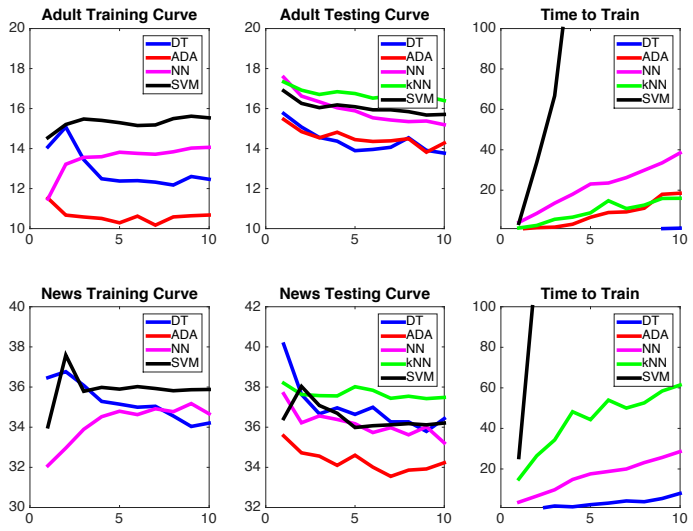


Figure 12

Interestingly, for the Adult data, the decision trees had the best performance with boosted trees a close second. I am surpised that boosting did not significantly improve performance as it did in the News data. This could relate to the unbalanced number of instances in each class, as discussed previously in the Boosting section. Judging by the gap between the training and testing error curves for each model, I believe that the models for the Adult data would have benefitted from less

complexity and data. On the other hand, the training and testing curves for the News data look fairly close to each other. If anything, these models may suffer from high bias.