Nathan Dass
CS 4641 – Assignment 1

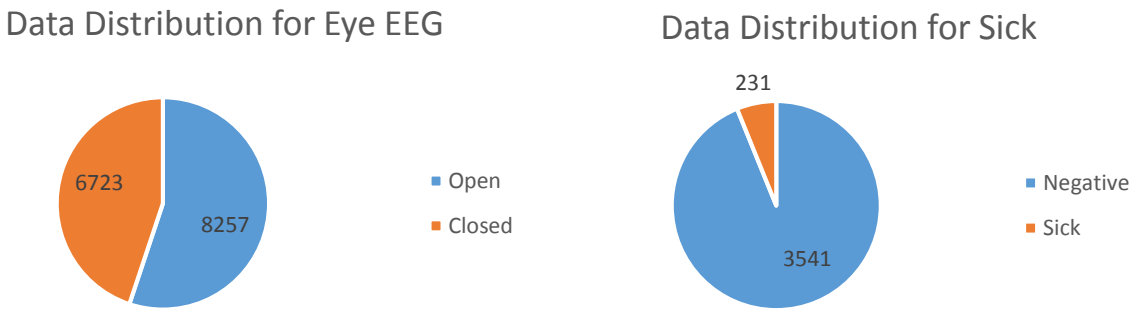
Supervised Learning Report

Datasets

Eye EEG. The first dataset I used tracked the state of an eye through an EEG. 14 EEG values were recorded and whether or not the eye was open or closed at a given time. The state of the eye was recorded through a camera and manually added to the data set by analyzing the video frames. The data was logged over the course of 117 seconds, giving about 128 measurements per second. The dataset had no missing values, contained 14 attributes that were all continuous, and 14980 instances.

Sick. The second dataset I used was a record of thyroid disease supplied by J. Ross Quinlan at the Garavan Institute from New South Wales Institute in Sydney, Australia. This dataset tries to determine whether or not a subject was sick or healthy (negative). This dataset contained a mix of continuous and discrete attributes. The dataset has 6064 missing values, which is 5.4% of all of the values. The dataset contained 30 attributes (7 continuous attributes and 23 discrete attributes) and 3772 instances. One thing to note is the high number of attributes in this dataset, but the lower number of instances. This plays into the Curse of Dimensionality, implying that we would actually need more instances to have accurate results.

Below are pie charts that describe the data distribution of both data sets.

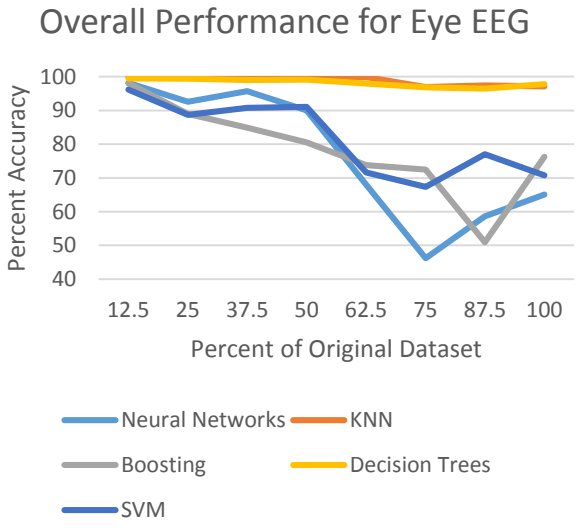


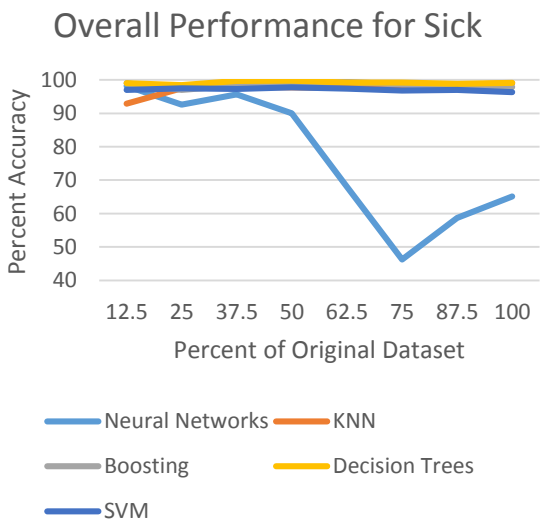
But why these datasets?

To me, these datasets are interesting in terms of the data they represent and the results obtained by running various supervised learning algorithms on them. I am very interested in futuristic user interfaces that move us away from the physical screen that we use right now. It

would be fascinating to explore how we can incorporate eye tracking into user interfaces. Some examples could be an augmented reality interface that moves with your eyes or an interface that can be controlled through blinking. The Eye EEG dataset could be used to notice when a user blinks and have the system react accordingly. The Sick dataset has practical applications in the sense that given various symptoms and information about a patient, we can predict whether or not a patient has a particular thyroid disease. Even if a few patients are helped through the analysis, we would be helping them with their everyday lives and sometimes even detecting thyroid cancer ahead of time.

In terms of the results of the machine learning algorithms themselves, these datasets show how five supervised learning algorithms (neural networks, k-nearest neighbors, boosting, decision trees, and support vector machines) perform with different types of data. Since the Eye EEG dataset only has continuous attributes, the results are different from the results obtained with the Sick dataset since the Sick dataset has both continuous and discrete attributes. Combining the results obtained from both datasets would give a better understanding of how the algorithms perform with different types of attributes. Below are two graphs that show the overall performance of both data sets with the different supervised learning algorithms.





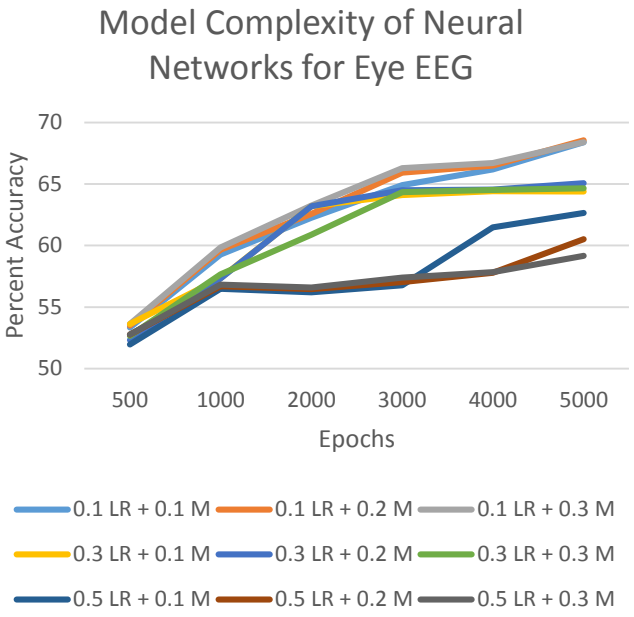
As seen in the graphs, neural networks did not perform well with either datasets. All algorithms start with a high percent accuracy with a small portion of the original dataset, but the accuracy of neural networks for both sets as well as boosting and SVM for the Eye EEG dataset decreases significantly starting at around 50% of the original dataset. Interestingly enough,

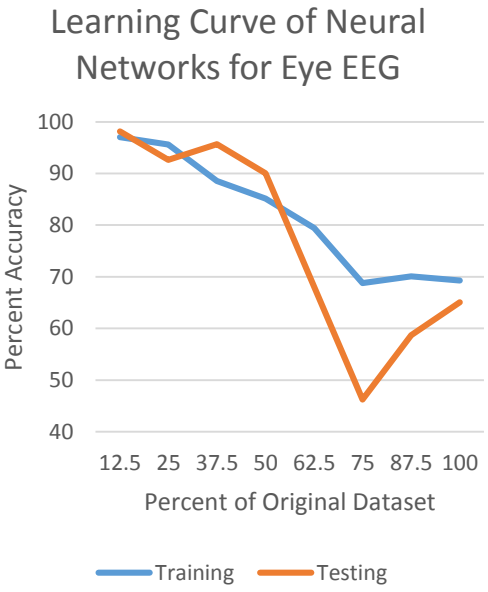
starting at 87.5% of the original dataset, the accuracy starts increasing again. This could be explained by having enough data where the algorithm starts to detect the testing cases that are close to the border, so a better model is built. The high accuracy at the beginning can be due to simply not having many test cases to incorrectly classify. Boosting and SVM did better than neural networks for the Eye EEG, but the accuracy was significantly lower than k-NN and decision trees. It appears that k-NN and decision trees were the algorithms that had the highest accuracy in both datasets. Even in terms of runtime, the only algorithm that has a significantly higher runtime than the rest of the algorithms is the neural networks algorithm. This would make sense since the runtime increases exponentially with the number of epochs. At 5000 epochs, the neural network algorithm ran in 140.89 seconds. Given the runtime and the accuracy of the neural networks algorithm, it doesn't seem like the algorithm is worth it for these two datasets. Boosting ran the fastest in 0.06 seconds, decision trees ran in 0.49 seconds, k-NN ran in 1.72 seconds, and SVM ran in 22.43 seconds. When considering the runtime and the accuracy of the different algorithms, it looks like the decision tree algorithm is the best algorithm for these data sets. Cross validation would help in reducing bias and therefore increasing the accuracy of the algorithms. The values I chose for the model complexity graphs for each algorithm were based on what I thought would be interesting to analyze.

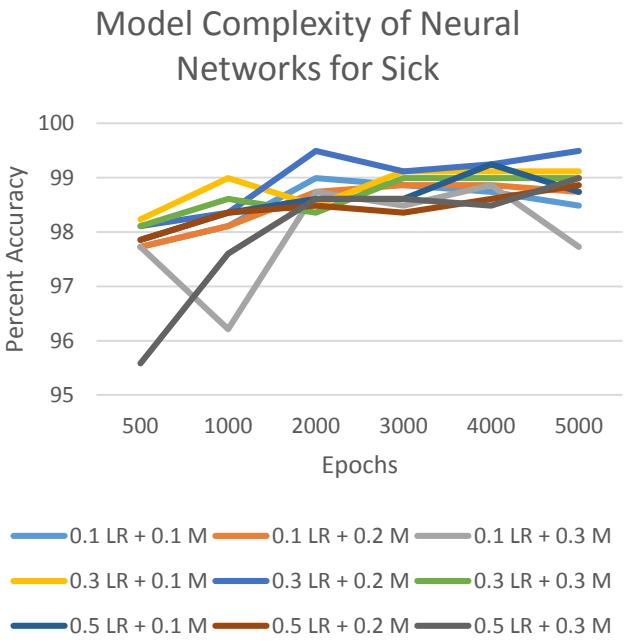
Neural Networks

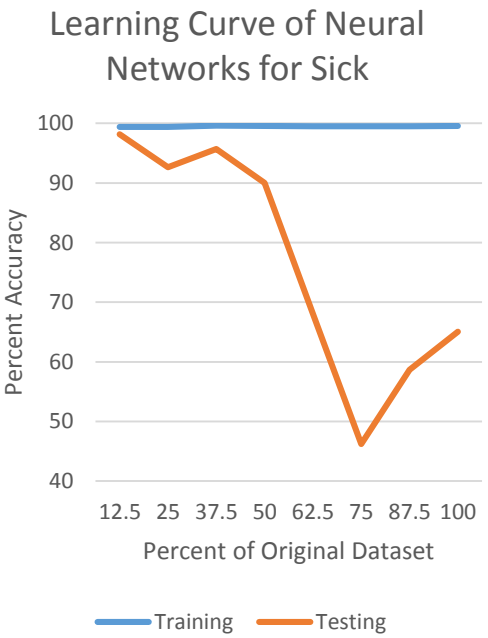
One thing that I found interesting about the model complexity graph for Eye EEG below is the grouping of the lines based on the learning rate that the line represents. While all 9 lines start off around the same accuracy, as the epochs approaches 5000, the lines with the same learning rate are close to each other in terms of accuracy. We can see that the optimal learning rate out of 0.1, 0.3, and 0.5 is 0.1 while the momentum does not seem to have much of an impact in this case. A momentum of 0.3 has a slightly higher accuracy until 5000 epochs when a momentum of 0.2 has the higher accuracy, but this could be due to noise in the data. With the learning curve graph for Eye EEG, we can see that the training accuracy always decreases, showing that the weights are still being adjusted for the different situations. I think that the low accuracy in general can be increased with more data, as we can see by the eventual increase of accuracy after using 75% of the dataset since it looks like the algorithm is overfitting until then.

The separation of lines based on learning rate is still prevalent with the Sick dataset, but a learning rate of 0.3 has the highest accuracy in this case. Additionally, a momentum of 0.2 had the higher accuracy of each learning rate in most cases. Oddly, the training error stays very close to 100% in this dataset and but the testing error follows the same trend as the Eye EEG dataset.

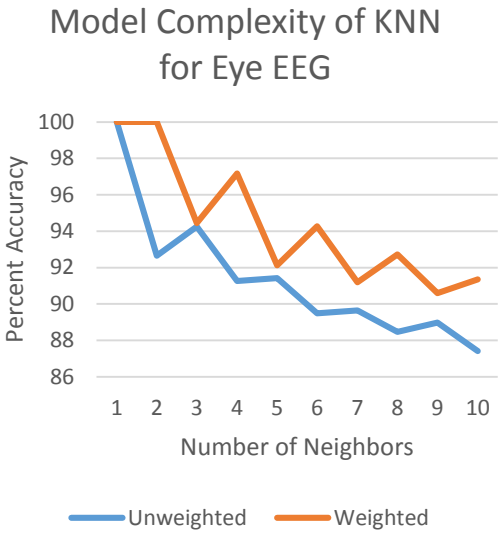


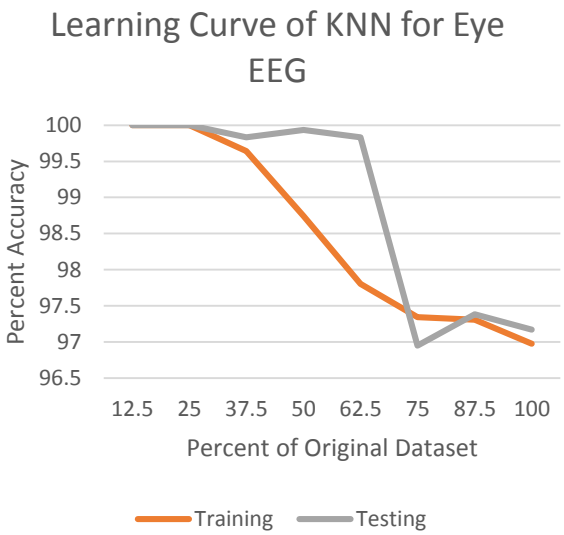


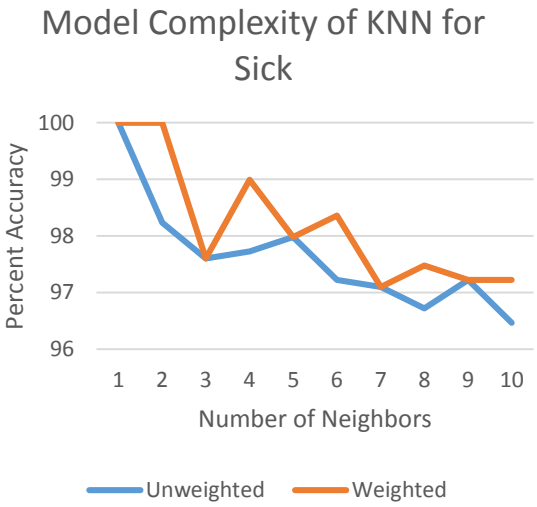


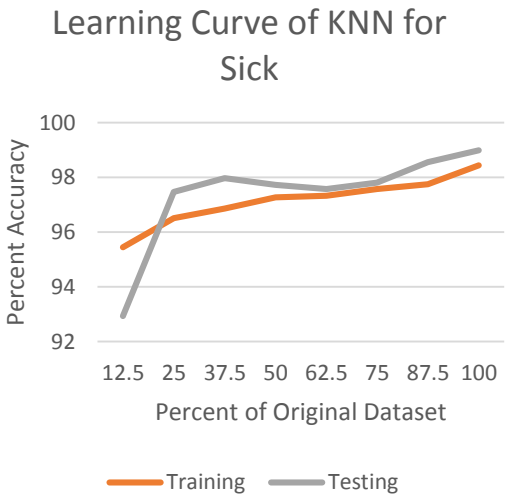


k-NN





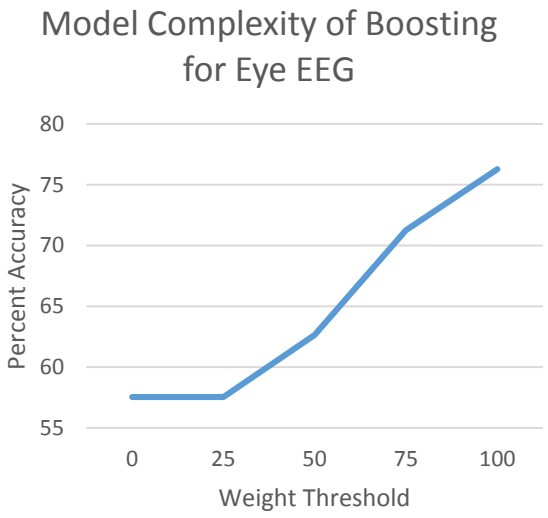


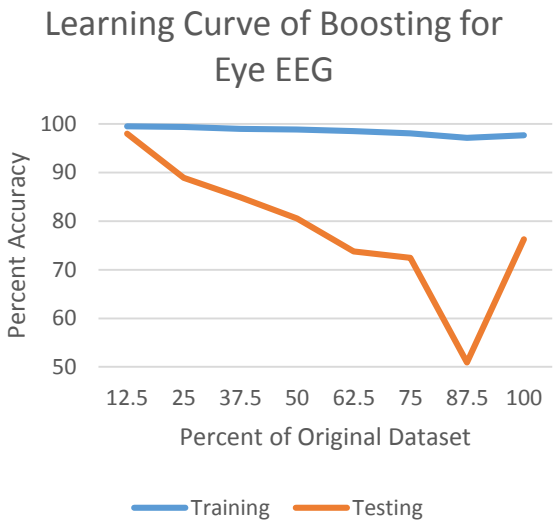


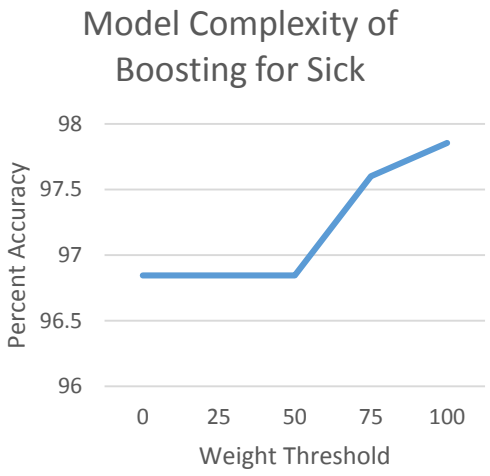
The overall trend of the model complexity graphs show that in general, weighted is more accurate than unweighted, but note the overall decrease in accuracy with an increase in neighbors despite the alternating increase and decrease between successive number of neighbors. This can be attributed to the fact that with a larger number of neighbors being considered, there is more variation between the data and the addition of weighting limits the effect that farther neighbors have on the answer. While the learning curves decreased for the Eye EEG dataset, they increased for the Sick dataset. This would make sense because the Sick dataset also has discrete attributes, so the neighbors are more likely to be representative of the correct classification since there is

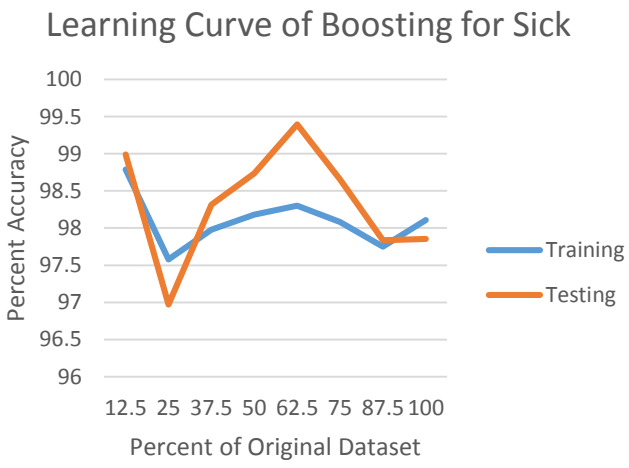
less variation in discrete values and the likelihood of being correct would only increase with more data. These results confirm that k-NN is capable of appropriately modeling datasets with both discrete and continuous attributes. Note the high accuracy of k-NN for very small values of k. This can be explained by the low bias and high variance associated with low values of k, which accounts for the variation in the dataset by only looking at very close neighbors instead of multiple neighbors.

Boosting



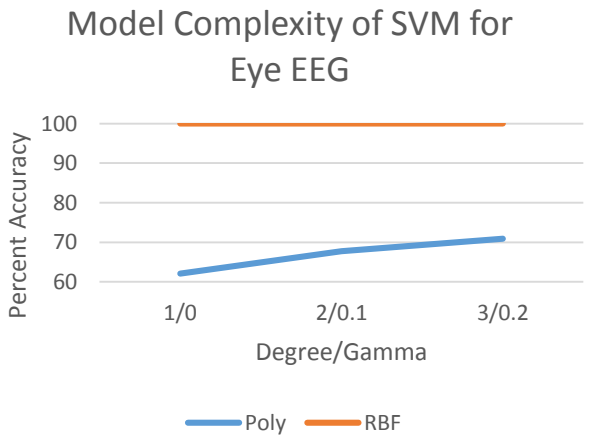


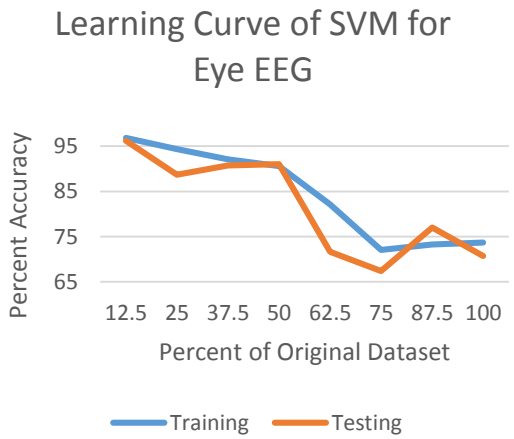


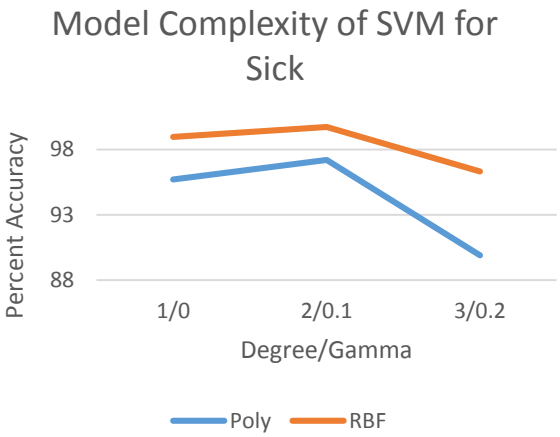


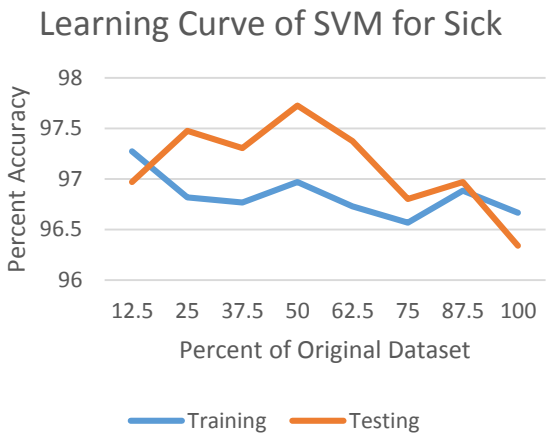
As seen in the first graph, boosting does not work very well for the Eye EEG dataset. Knowing that boosting is sensitive to noisy data and given the circumstances under which the data is collected (through a machine, which can lead to errors in precision and accuracy of the machine), we can guess that there is some noise in the data that is decreasing the accuracy of the boosting algorithm. Seeing that the training accuracy is very high for boosting with the Eye EEG dataset and the testing accuracy is significantly lower comparatively, we can also guess that the data is being overfitted and even if there is some noisy data, if the algorithm overfits to the noisy data, the error obtained when testing on unseen data can be much higher than other algorithms. On the other hand, boosting works very well for the Sick dataset in general. By looking at the learning curve graph for the Sick dataset, we can see that the optimal amount of data is 62.5% of the original dataset. With 62.5% of the original dataset, we get the highest accuracy for the testing, reducing the most bias from our results. After 62.5%, we can see a decrease in the testing accuracy of the algorithm, implying overfitting to the training data. In both data sets, we can see that as the weight threshold increases, the percent accuracy goes up. As the weight threshold increases, less pruning occurs since the threshold to actually prune is higher and therefore harder to meet. Ideally, pruning is good when a data set is simple and it eliminates unnecessary branches while barely affecting the accuracy of the algorithm. However, these data sets are pretty complex, having 14 attributes in the Eye EEG dataset and 30 attributes in the Sick dataset, so it would make sense that the higher the weight threshold, the higher the accuracy of the algorithm.

SVM



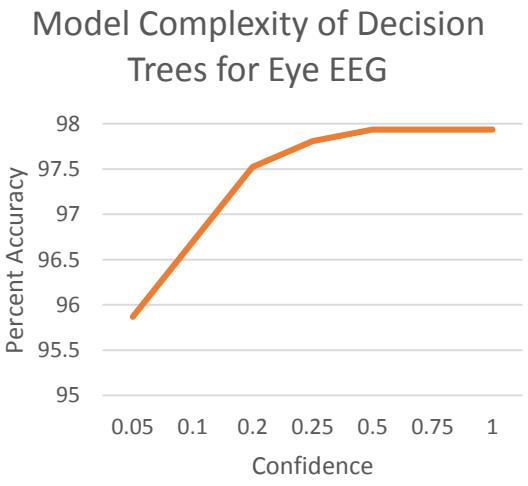


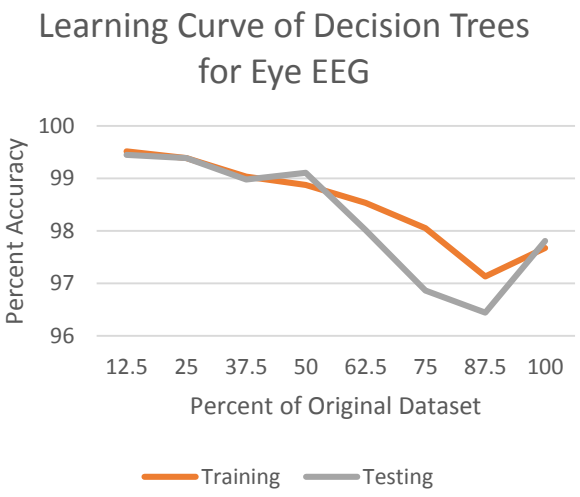


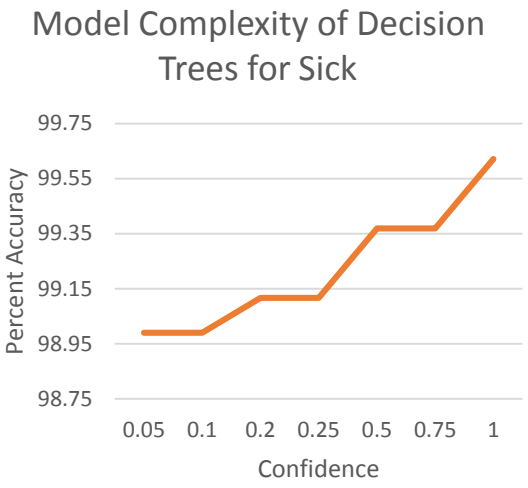


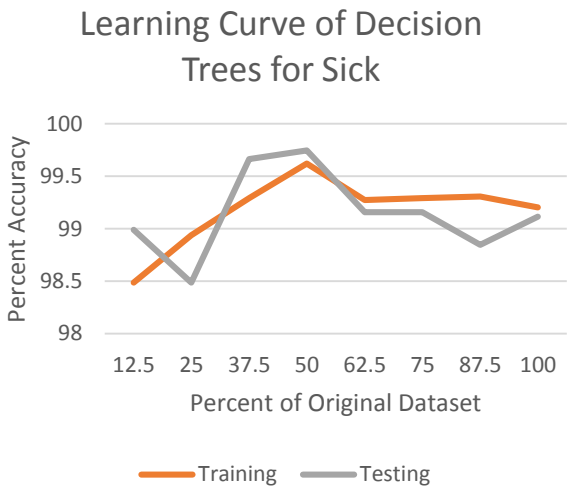
In general, we can see that the RBF kernel has a significantly accuracy than the Polynomial kernel for SVM. Oddly, the RBF kernel perfectly fits the data regardless of what the gamma value is which implies that the data can be perfectly split by a hyperplane when using the RBF kernel. An increase in the degree of the polynomial typically led to an increase in the percent accuracy, which would make sense because the hyperplan has a higher degree of freedom and flexibility when optimizing the margin. We can also see an increase in the accuracy of the RBF kernel in the Sick dataset with a slight increase of gamma which can be explained by the decrease in variance associated with a higher gamma.We can also see overfitting occuring at 50% of the original dataset in both datasets, but significantly higher in the Eye EEG dataset.

Decision Trees









The decision tree algorithm worked very well for the Sick dataset regardless of what the confidence was, having a minimum accuracy of a little under 99%. Even the training and testing accuracy was very high for the Sick dataset. I think this is because 23 of the 30 attributes in this dataset are discrete, which is what decision trees excel at. We can also see overfitting occuring in the area when using 50% of the original dataset. At this point, the highest testing accuracy and training accuracy is achieved at above 99.5% accuracy. When using more than 50% of the dataset, we can see that the algorithm overfits the training set and performs poorly on the testing set. On the other hand, we can see the model complexity graph for the Eye EEG dataset hit a saturation point of about 98% at 0.5 confidence. The confidence parameter measures how confident you are in your dataset. Therefore a low confidence leads to more pruning and a high confidence leads to less pruning. Through this, we can infer that the Eye EEG dataset is complex enough that too much pruning will significantly reduce the accuracy of the decision trees, but there is a point where pruning fewer trees does not have an effect on the accuracy of the algorithm. Regardless of the saturation that occurs in the Eye EEG dataset, the decision tree algorithm still performs very well which implies that the dataset can be split pretty evenly with the different attributes.