Examining ESD Diagnoses and Wine Classification Using Various Machine Learning Algorithms

Andrew Fang

CS 4641

Background

The first classification problem explored in this report is the diagnosis of erythemato-squamous diseases (ESD). Diseases in the ESD group include psoriasis, seborrheic dermatitis, lichen planus, pityriasis rosea, chronic dermatitis, and pityriasis rubra pilaris. The difficulty in diagnosing these diseases lies in the fact that all the diseases in this group share overlapping symptoms, which makes this interesting. The dataset used has 31 attributes used to classify an instance of a disease into one of 6 options. The data has 366 instances.

The second classification problem explored in this report is the identification of what kind of grape a wine came from. Attributes that may contribute to how this is determined include alcohol content, ash, color intensity, hue, malic acid, etc. In total, the dataset uses 13 attributes to classify a wine into one of 3 types. The data has 178 instances. Wine has a reputation among people for being indistinguishable between varieties, so proving with a machine that these differences (that can be measured by humans) exist is interesting.

Decision Trees

For the decision tree algorithm, I used WEKA's J48 function on the ESD dataset. First, I examined the effect of different percentages of the data used as training on the accuracy of classification. The algorithm seemed to reach a stable accuracy rate relatively quickly, requiring only 30% of the total data to reach a point where accuracy did not increase much further. Maximum accuracy was 92.97%, achieved by using

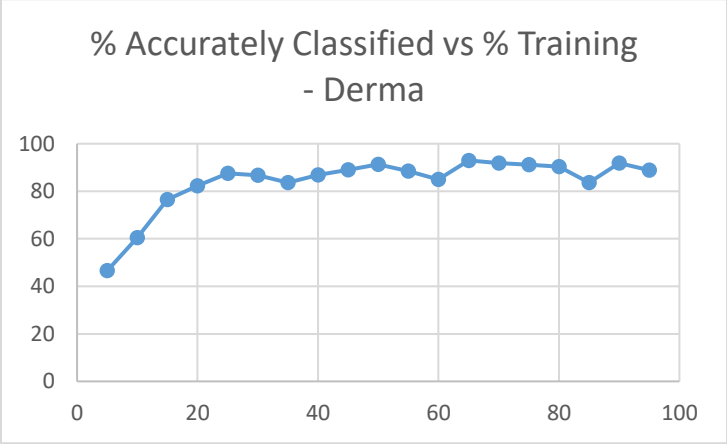


Figure 1

65% of the total data for training. It's possible that this is a result of the decision tree being sufficiently trained after the 110 instances of data (30% of the total number of instances). It is important to note that although the decision tree seems to reach a point where it is accurate after this point (sitting in the 83% to 87% range), it continues to grow more accurate until it reaches the 88% to 92% accuracy.

Examining this phenomenon with a different size data set reveals more information. Using a smaller size data set with the J48 algorithm seems to reveal a growth in accuracy that looks almost linear. The tree is 79.58% accurate at 20% of instances used for training (36 training instances), and eventually reaches accuracy above 95% at 55% of instances (98 training instances).

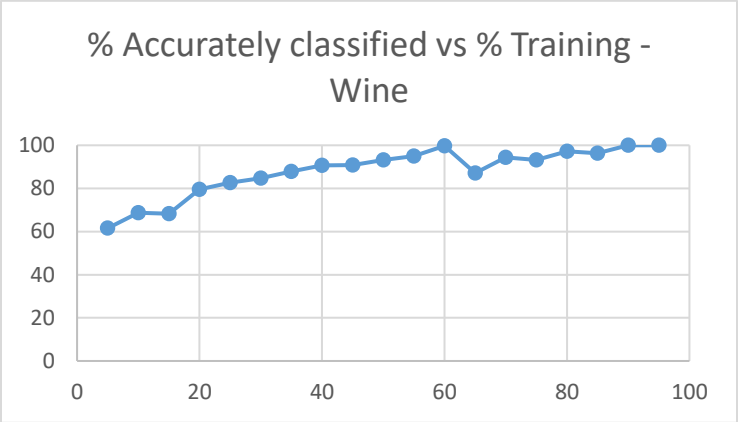


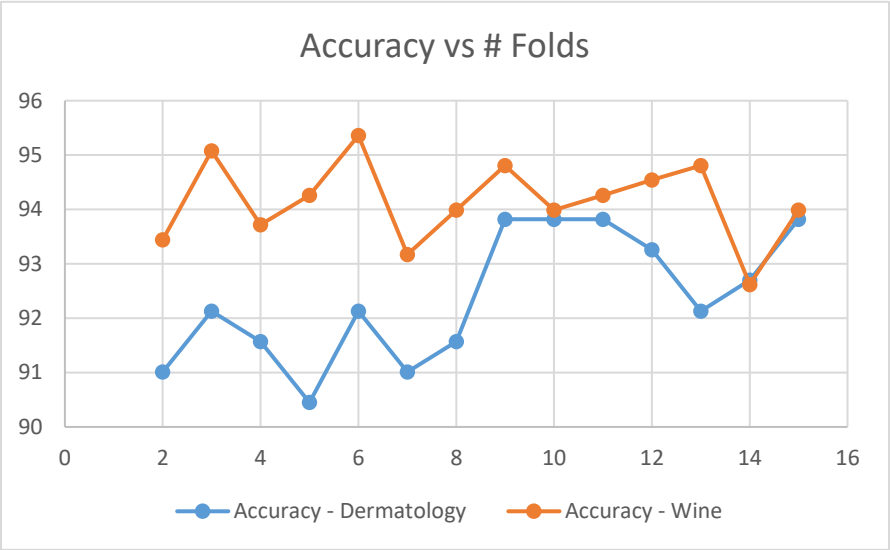
Figure 2

It is important to note with the wine training set, the relatively low number of instances of data makes the 100% accuracy on the later points in figure 2 not entirely reliable, because only 18 instances were in the test set at 90%, and 9 instances in the 95%. It is possible that the high accuracy is derived from the low number of instances in the training set, and not because of the amount of training done in the tree.

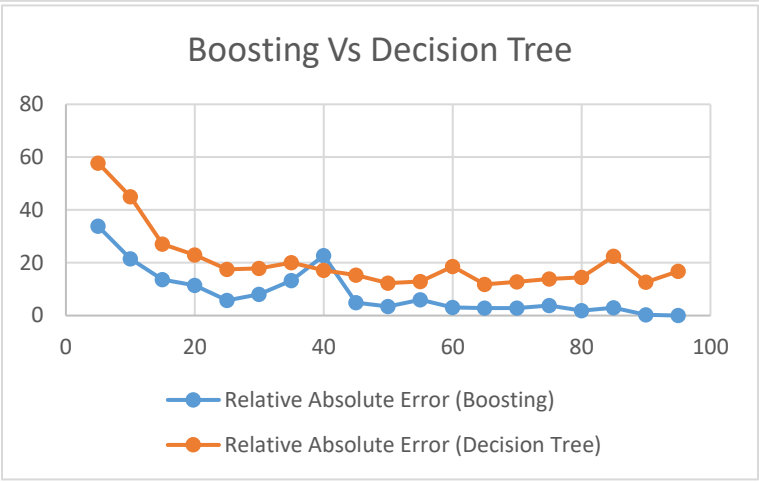
It is possible to conclude, however, that the accuracy of the decision tree algorithm relies not on the percentage of data used in the training set, but rather the quantity of data used to train the tree

Figure 3

instead. It appears that accuracy becomes relatively stable at around 100 instances. In the dermatology data set, this was at 30% of the total data, and 55% of total data in the wine data set.



One additional aspect of decision trees that we can examine is the effect of different amounts of cross validation on the accuracy of the tree. In figure 3, it appears that higher numbers of folds for cross validation does not have as much of an effect on the wine dataset as it does on the dermatology dataset. This appears to be because of the different number of data instances. Cross validation seems to have a more significant effect on the dermatology dataset, which has more data instances than does the wine dataset.



Boosting

Boosting (using the AdaBoost algorithm) decreased the amount of error compared when applied to a decision tree (the same J48 algorithm) at a relatively consistent rate.

Figure 4

There was a spike in the relative absolute error when using 40% of the total dataset for training. Whether this is because the boosting algorithm amplified a mistake in classification is unclear. However, there is evidence that this is the case when the error graph for the 30% and 40% levels are used. In figure 6, the errors that are present in the 30% classifications are seemingly made more severe in the 40% error graph.

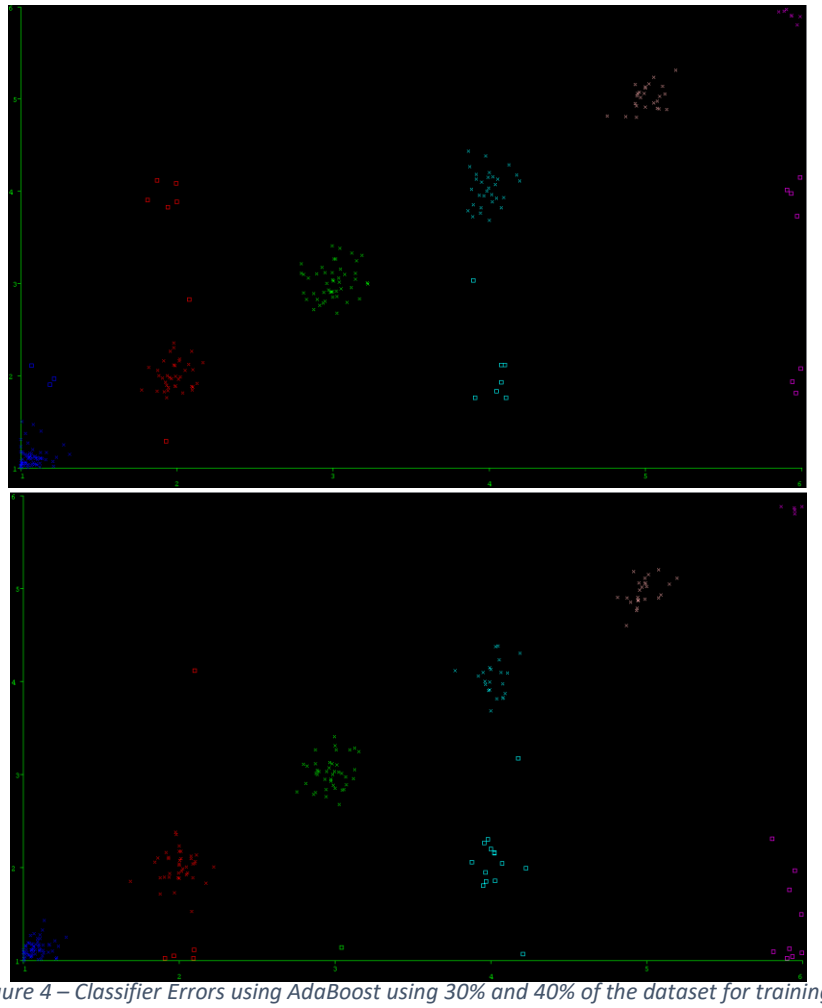


Figure 4 – Classifier Errors using AdaBoost using 30% and 40% of the dataset for training

Changing the number of folds for cross validation did not significantly decrease error past a certain point. The minimum relative absolute error was 3.12%, maximum relative absolute error was 6.82%. However, the relative absolute error value for 3-fold cross-validation (4.32%) is already lower than the average error for all values of K (4.34%). Since it is unclear whether the

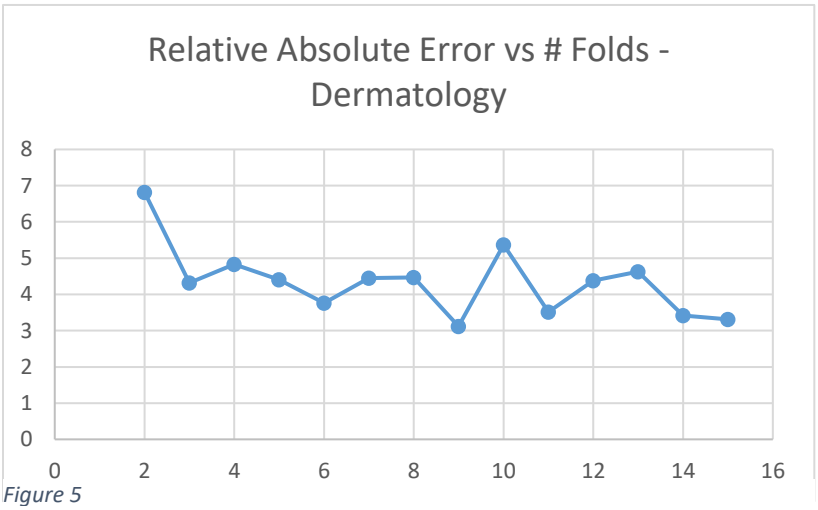


Figure 5

2-fold cross validation error value was a result of the low number of folds or natural error, I ran the same experiment on the wine dataset. Here, we can see that the general trend for increasing number of folds for cross-validation is associated with a decrease in relative absolute error. The mean error for this experiment was 5.52%, with maximum error being 7.82% and minimum error being 3.41%. The combination of these two datasets allows us to conclude that generally, as number of folds increase, error decreases. However, it seems like this is only true up until a certain point. It is likely that error stops decreasing past a certain point based on the learning algorithm used that the cross validation is applied to.

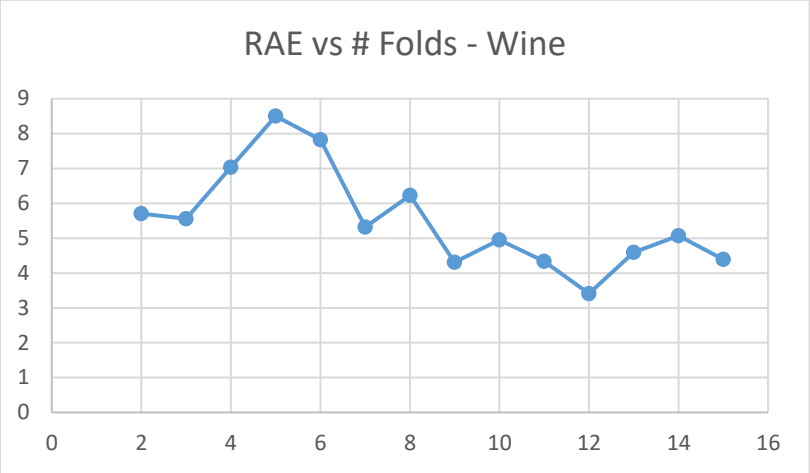


Figure 6

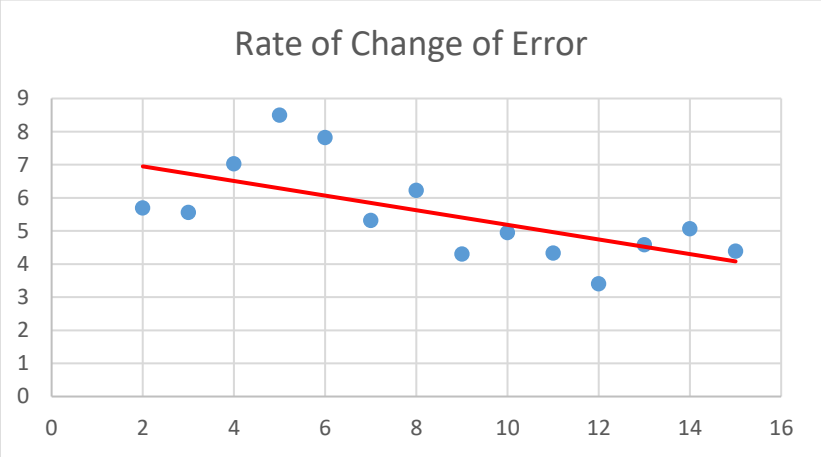


Figure 7

K Nearest-Neighbors

The K Nearest-Neighbor algorithm, run without weights and cross-validation using 70% of the dataset for training produced less accurate results than any other algorithm so far. Error was at its minimum when K = 1. Even then, i

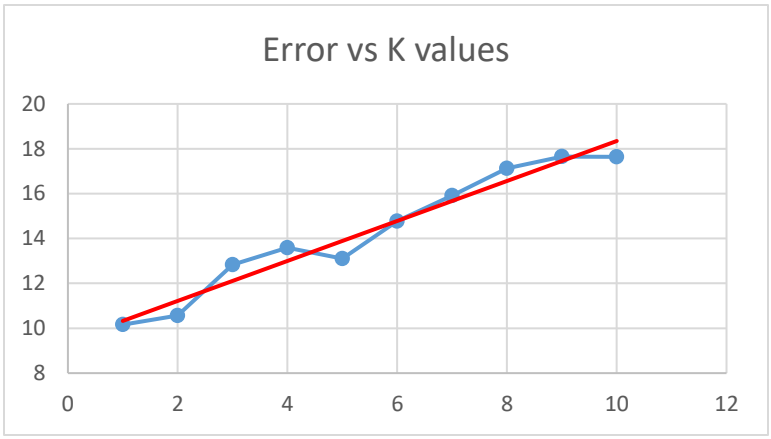


Figure 8

increased, relative absolute error also increased at what appears to be a linear rate. The addition of cross validation and weighting slightly reduced error rates. The two weighting methods used were inverse distance and 1 minus distance weighting. Of the two weighting methods, inverse distance managed to reduce error more, if only slightly. 1 minus distance weighting actually increased error across the total experiment by 0.17%, whereas inverse weighting decreased error by 0.8%. Cross validation exhibited a very unusual behavior where at $3 \leq K \leq 6$ and $7 \leq K \leq 10$, the relative absolute error remained the same.

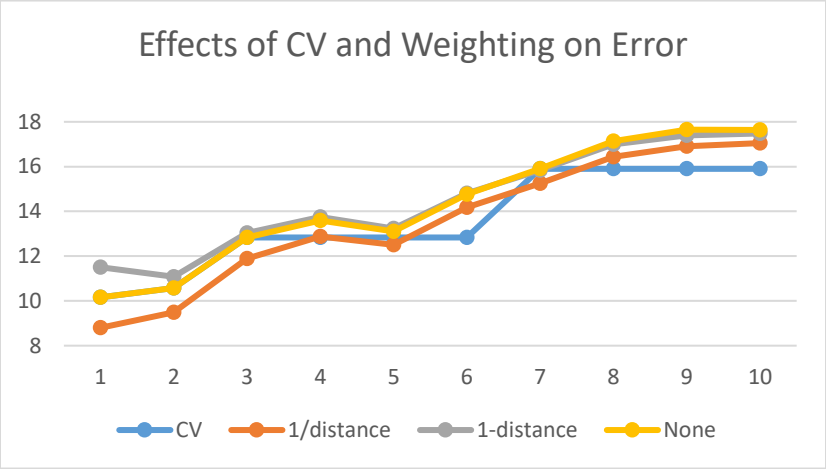


Figure 9

SVM

First, to determine which kernel to use for further experiments, I compared WEKA's 4 kernels with each other on the dermatology dataset with 10-fold cross-validation. These four kernels were Linear, Polynomial, Gaussian, and Sigmoid. Of these, the linear kernel performed

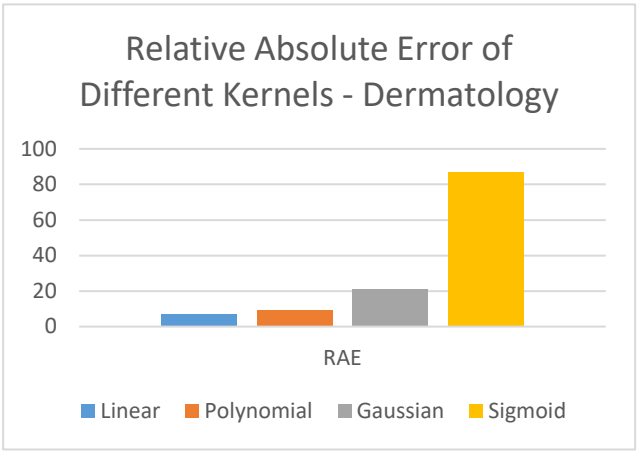


Figure 10

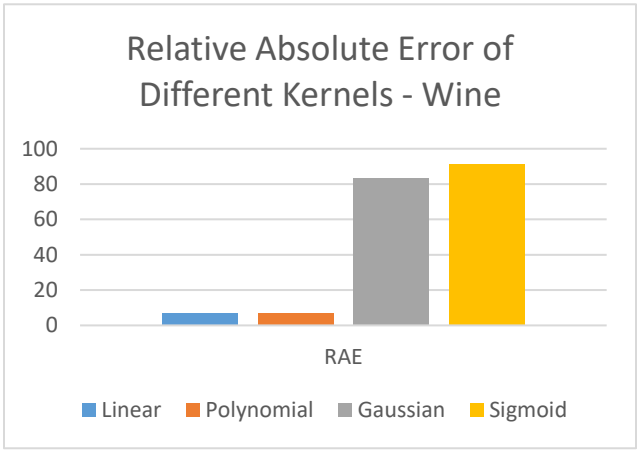


Figure 11

best with 6.84% relative absolute error, followed by 9.57% using a polynomial kernel, then the Gaussian kernel with a 21.19% relative absolute error, and then followed by the sigmoid kernel with a dismal 87.15% error. The kernels performed similarly on the wine dataset, with the Gaussian and Sigmoid kernels performing significantly worse than the linear and polynomial kernels.

Neural Nets

I first determined that the number of folds of cross validation did not have a significant impact on the relative absolute error of the neural net. I then examined the effect of different number of layers on the error value. The learning rate was set at 0.3 and the momentum set at

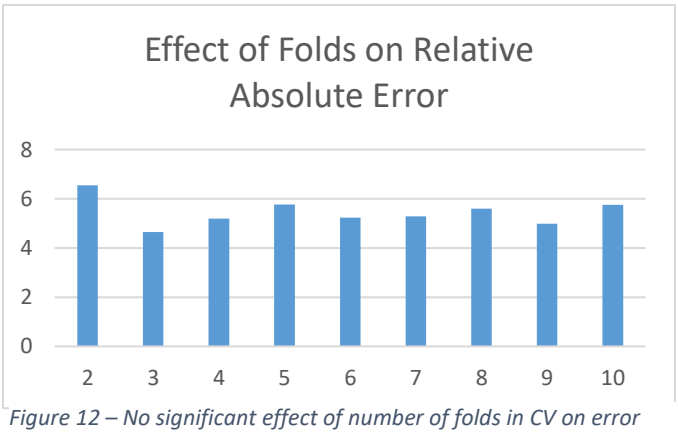


Figure 12 – No significant effect of number of folds in CV on error

0.2 for this experiment. I varied the number and "shape" of the layers by manually adjusting the number of nodes in each layer in either a doubling pattern or a diamond pattern. The doubling pattern involves the next layer having double the number of nodes as the previous layer. The diamond pattern involves doubling the number of nodes in the layers until the middle layers, then halving the number of nodes in each layer after the middle point. It turned out that the diamond shape is extremely detrimental to the accuracy of the neural net. The error values of

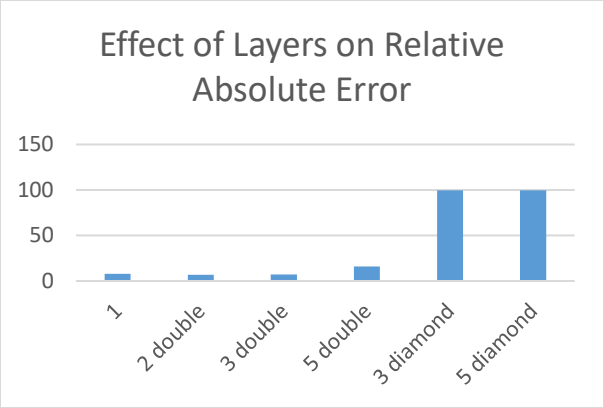


Figure 13

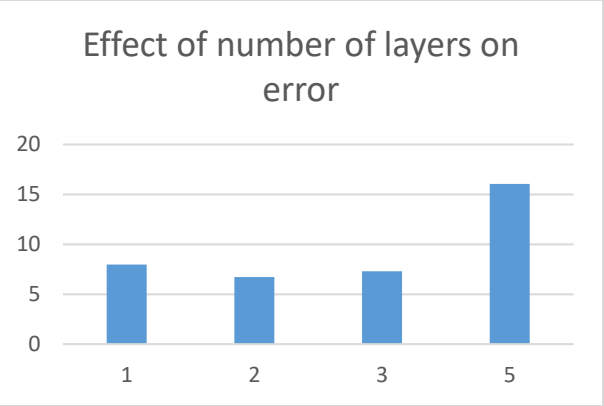


Figure 14

both diamond shaped layers were more than 99%. It also seems that the effect of the number of layers when doubling nodes has no impact other than increasing the error rate at higher numbers of layers.

Next, I examined the effect of altering the learning rate of the neural net on the relative absolute error of the algorithm. It turned out that as learning rate increased, error decreased at a polynomial rate. However, as the learning rate approached 1, the neural net reached a point where

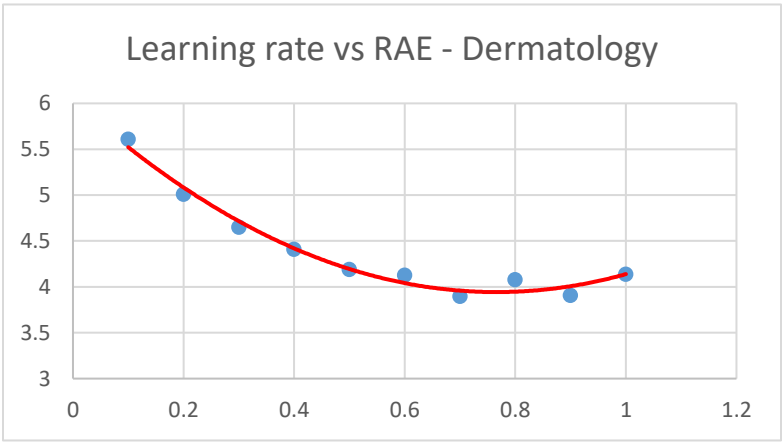


Figure 15

the error started to increase. This is likely because of the algorithm started overfitting to the data instead of generalizing it. Similar results came from the wine dataset, although the change of relative absolute error was never positive. It is possible that since this dataset is smaller, it is more difficult for the neural net to overfit the data, given that there is less data to both train and test on. Similar results occurred while changing momentum. In the dermatology dataset, as momentum increased, error decreased. However, at Momentum = 0.9, the error value skyrocketed to 40.16%, and to 91.17% at Momentum = 1. This is most likely a case of overfitting In the wine dataset, increasing amounts of momentum was associated with decreasing error values. However, like in the dermatology dataset, at Momentum = 1, the error value skyrocketed to 89.73%.

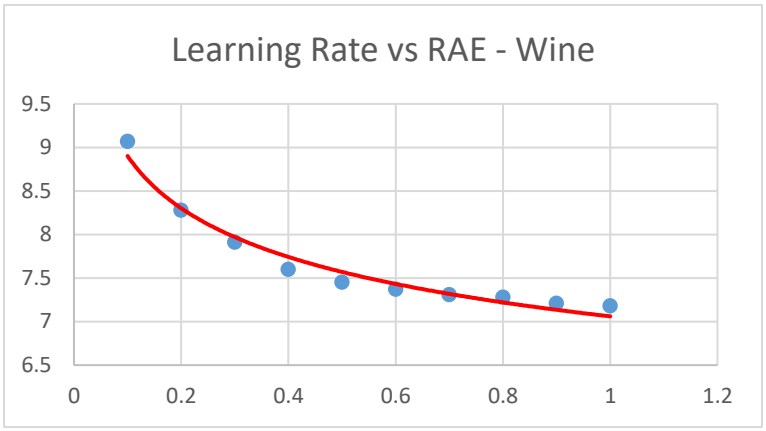


Figure 16

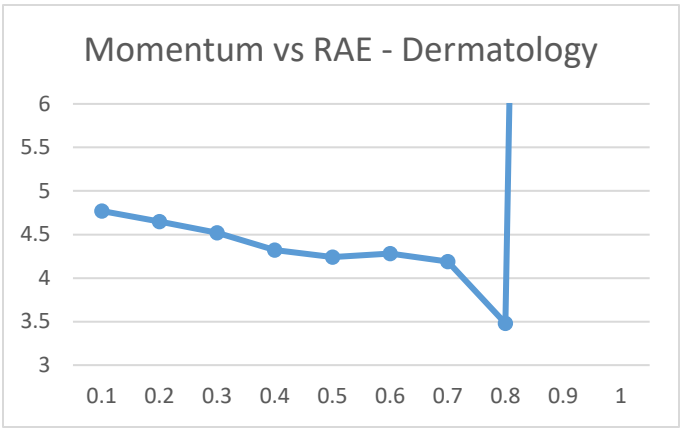


Figure 17

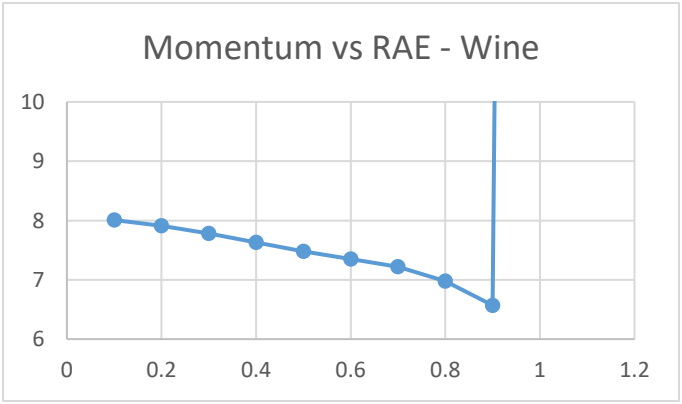


Figure 18

Method

Unless otherwise stated, all experiments were done using 70% of the data set as training data and the remaining 30% as the test set. For runtime purposes, all experiments were done using 3-fold cross validation.

Conclusions

For both the dermatology dataset and the wine dataset, the process that produced the most accurate results was the Adaptive Boosting algorithm applied to the J48 decision tree algorithm. In a close second was the neural network with higher learning rates and momentum values. Some things of note: the Decision Tree, Boosting, and K Nearest-Neighbors algorithms completed almost instantly whereas the SVM and Neural Network algorithms took longer, significantly so in the case of Neural Networks. For classification problems with smaller amounts of data, these algorithms are suitable (obviously with Neural Networks and Boosting being most optimal in this case) because of the reasonable runtimes and accurate results. I would speculate that for certain classification problems where results are time sensitive and the dataset is large, the decision tree or boosting algorithms would be more effective.