Classification Of Medical Diagnosis

SENG 474 Assignment 1
Austin Bassett
V00905244

February 2021

1 Introduction

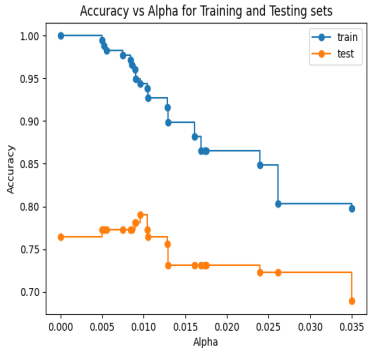
For the second classification problem I went with a dataset that would classify breast cancer as malignant or benign. This problem is interesting because it takes doctors years of schooling and experience to properly diagnose breast cancer. The dataset I used from the UCI machine learning repository uses 30 features to predict the diagnosis[1]. Compared to the 13 features needed in the provided dataset to predict heart disease. The other notable difference between the two sets is the type of data. The breast cancer data contains mostly floating point values while the heart disease data contains mostly integers. I wanted to know how accurately a decision tree(with pruning), a random forest(no pruning) and a neural network could predict a diagnosis.

The programs I developed only deal with classification of numbers, meaning, the breast cancer dataset required adjusting. I removed the ID column entirely because these values only identified breast tissues from the study. For the diagnosis column I assigned the following values, 1 for malignant and 0 for benign. Now for the testing phase, I ran three different training split percentages, 82%, 60%, and 10% with the remainders used for testing. By running three different training splits I produced a better range of results, but also to know how much training data is needed for an acceptable accuracy. The following sections will only look at results from the 60% split because these results were bounded by the other two splits.

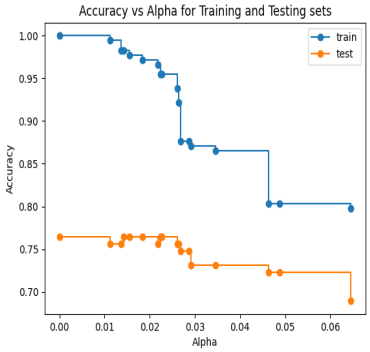
2 Decision Tree

The decision tree was implemented using the DecisionTreeClassifier function from scikit-learn[2]. Post pruning the decision trees was

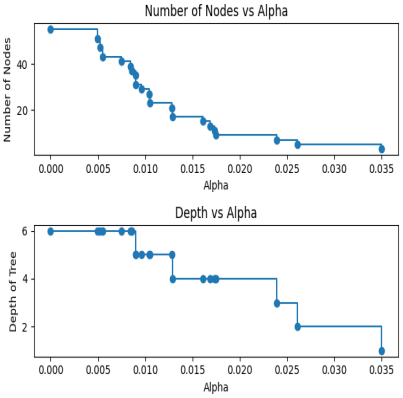
done with scikit-learn's cost complexity pruning path function[3]. The pruning function works by assigning nodes an effective alpha score, and pruning the nodes with the smallest score. Pruning terminates when the minimal effective alpha is greater than alpha[4]. I then created a list of potential decision trees increasing the alpha value for each tree, and graphing the results against these alpha values. Finally, two sets of results were produced for each dataset, one for the gini criterion and one for the entropy criterion.



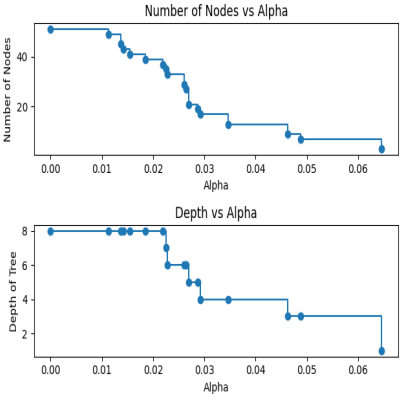
(a) Gini Accuracy as Alpha Increases



(b) Entropy Accuracy as Alpha Increases



(c) Gini Depth Vs Accuracy



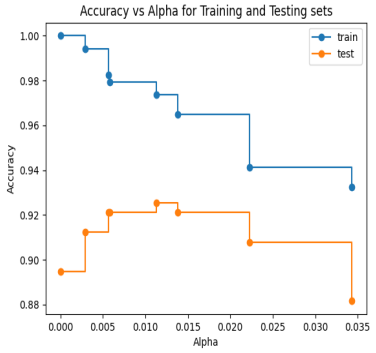
(d) Entropy Depth Vs Accuracy

Figure 1: Heart Disease Decision Tree Accuracy Results

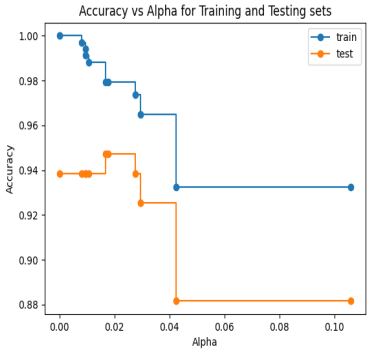
The heart disease was able to achieve 79% accuracy with a tree depth of 5 using gini, and 76% accuracy with a tree depth of 8 using entropy. Looking at the results (see Figure 1a,b) we can see that in both cases the training and testing results are marginally different. The gini results (see Figure 1a) show it is an overfitting model. The

entropy results (see Figure 1b) also show it is an overfitting model.

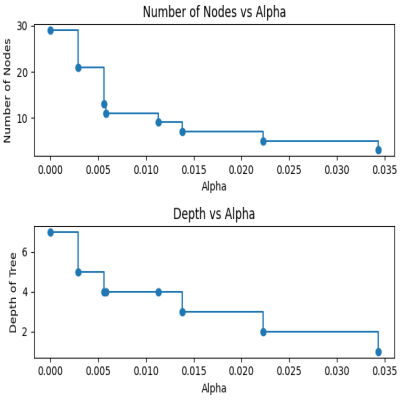
The breast cancer achieved 95% accuracy with a tree depth of 4 using entropy, and 93% accuracy with a tree depth of 4 using gini. Looking at the results (see Figure 2a,b) we see that, again, the training and testing results are marginally different. The gini method (see Figure 2a) is an overfitting model. Similarly, the entropy (see Figure 2b) method is also an overfitting model.



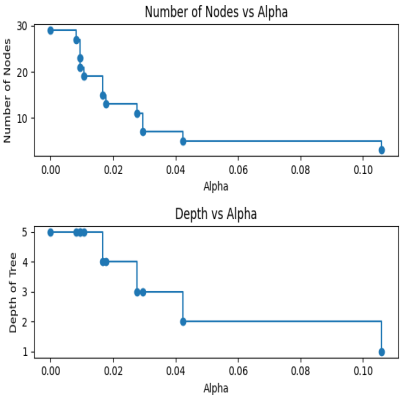
(a) Gini Accuracy as Alpha Increases



(b) Entropy Accuracy as Alpha Increases



(c) Gini Depth Vs Accuracy



(d) Entropy Depth Vs Accuracy

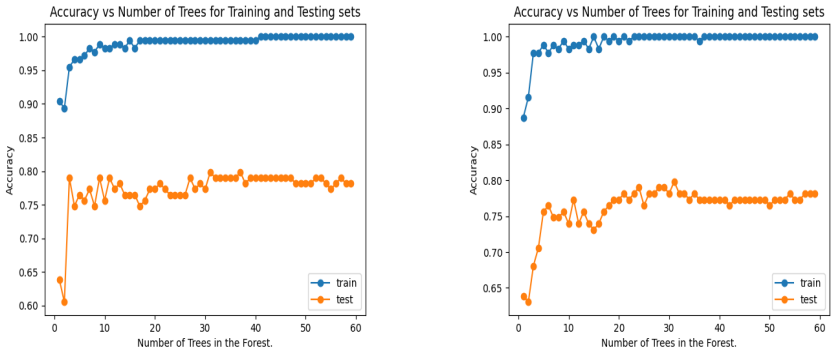
Figure 2: Breast Cancer Decision Tree Accuracy Results

3 Random Forest

The random forest was implemented using the RandomForestClassifier function from scikit-learn[5]. There was no pruning of the trees with this method, instead, I varied the maximum number of trees

a forest could contain. I decided to use the square root method for determining the maximum features to consider when splitting. Then I created a list of potential forests where each forest could contain more trees than the last. Using this list I graphed the accuracy against the number of trees, as I wanted to see if a relationship existed between the two. Finally, two sets of results were produced for each dataset, one for the gini criterion and one for the entropy criterion.

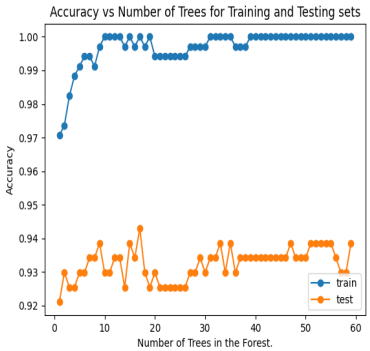
The heart disease dataset achieved a remarkable 80% accuracy with 31 trees for both the gini and entropy criterion. Looking at the results (see Figure 3a,b), we can see both methods are sporadic at the start, they both start converging to 77% just over 30 trees in the forest. However, looking at the gini results (see Figure 3a) we see it is an overfitting model. This is because the training accuracy is 100%, while the testing accuracy was 77%. The entropy method is also an overfitting model for the same reasons (see Figure 3b).



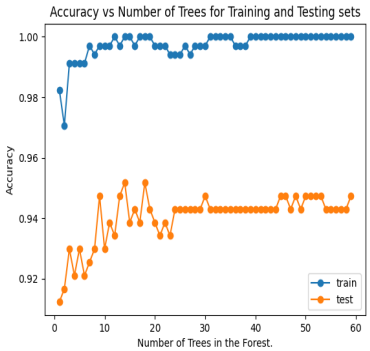
(a) Gini Accuracy as Trees Increase (b) Entropy Accuracy as Trees Increases

Figure 3: Heart Disease Random Forest Accuracy Results

The breast cancer dataset achieved 95% accuracy with 14 trees using entropy, and 94% accuracy with 17 using gini. Looking at the results (see Figure 4a,b), we can see entropy starts converging to 94% at about 25 trees, while gini bounced between 91-94% accuracy. With the gini method (see Figure 4a) we can see it is a best-fit model. This is because the training accuracy is 100%, while the testing accuracy was around 92%. Similarly, the entropy method is also a best-fit model for the same reasons (see Figure 4b).



(a) Gini Accuracy as Trees Increase

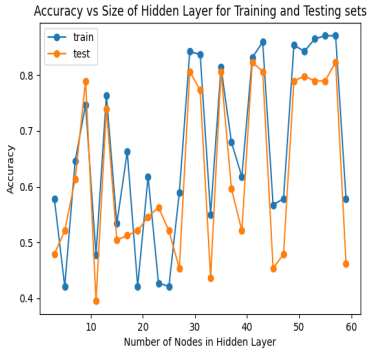


(b) Entropy Accuracy as Trees Increase

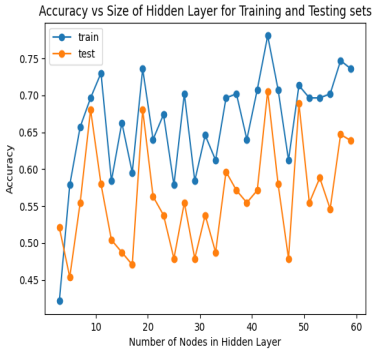
Figure 4: Breast Cancer Random Forest Accuracy Results

4 Neural Network

The neural network was implemented using the MLPClassifier function from scikit-learn[6]. I varied the maximum number of nodes available in the hidden layer, with the minimum being 3. I then created a list of neural networks with each network containing more hidden nodes than the last. I graphed the accuracy results against the number of nodes in the hidden layer, as I wanted to see if a relationship existed between them. Finally, two sets of results were produced for each dataset, one for the stochastic gradient descent(sgd) and one for the stochastic gradient-base optimizer(adam)[5].



(a) Adam Accuracy as Hidden Nodes Increase

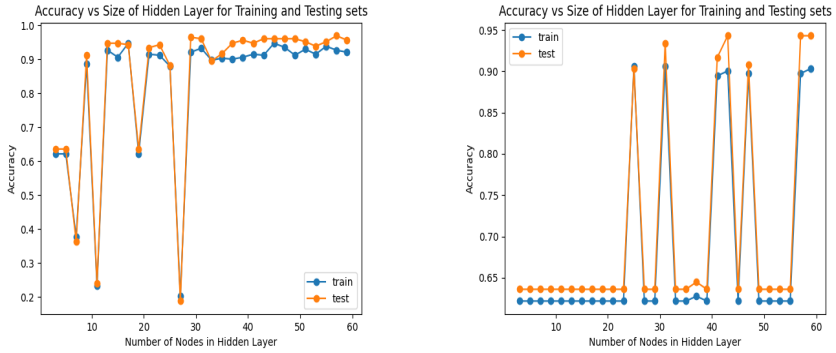


(b) SGD Accuracy as Hidden Nodes Increase

Figure 5: Heart Disease Neural Network Accuracy Results

The heart disease achieved an accuracy of 71% with 43 nodes in the hidden layer using sgd, and 82% accuracy with 41 nodes using adam. Looking at the results (see Figure 5a,b), we see that with both solvers the accuracy was sporadic and was not affected by the number of nodes in the hidden layer. What is interesting is how the test results closely follow the training results for both solvers. Starting with the sgd graph (see Figure 5b), we can see it is an overfitting model. Compared to the adam graph (see Figure 5a), which shows it is a best-fit model, where the training and testing results were relatively close with high accuracy.

The breast cancer dataset achieved 97% accuracy using 57 nodes in the hidden layer with adam, and 94% accuracy using 43 nodes in the hidden layer with sgd. Looking at the results (see Figure 6a,b), we can see the methods are almost inverse to each other with adam maintaining high accuracy and sgd maintaining low accuracy. The adam solver (see Figure 6a) is a best-fit model for this dataset because the training and testing are relatively close with high accuracy. Compared to sgd (see Figure 6b) which is and underfitting model because, while training and testing are relatively close, their accuracy is also low.



(a) Adam Accuracy as Hidden Nodes Increase

(b) SGD Accuracy as Hidden Nodes Increase

Figure 6: Breast Cancer Neural Network Accuracy Results

5 Conclusion

From these results we see that the number of features used, coupled with the type of data can impact results significantly. For the heart disease dataset the best method was the neural networks. While

this method didn't yield the highest accuracy it did generate a best-fit model using the adam solver. This best-fit model could then be generalized for new data. Compared to the breast cancer dataset, which saw the best results from the neural networks using adam as well. This yielded the highest accuracy for this dataset at 97%. In conclusion, the best models for both datasets were neural networks because they produced high accuracy best-fit models.

6 References

[1] *"Breast Cancer Wisconsin (Diagostic) Data Set"*, UCI, Accessed on: Feb 5, 2021.[Online]. Available: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`

[2] *"sklearn.tree.DecisionTreeClassifier"*, Scikit-learn, Accessed on: Feb 5, 2021.[Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`

[3] *"Post pruning decision trees with cost complexity pruning"*, Scikit-learn, Accessed on: Feb 5, 2021.[Online]. Available: `https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html`

[4] *"Minimal Cost-Complexity Pruning"*, Scikit-learn, Accessed on: Feb 5, 2021.[Online]. Available: `https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning`

[5] *"sklearn.ensemble.RandomForestClassifier"*, Scikit-learn, Accessed on: Feb 5, 2021.[Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

[6] *"sklearn.neural_network.MLPClassifier"*, Scikit-learn, Accessed on: Feb 5, 2021.[Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`