Cameron Lindsay

V0092778

Undergraduate Student

SENG 474 Assignment 1

1. Cleaned_processed.cleveland.data

This dataset is composed of 297 patients, with 13 features ('age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal') defined by two primary classes: no heart disease, and heart disease-like presentation. Because of the nature of the data, all algorithms applied to this dataset will be classification-based in nature and will look to classify test data into either the 'heart disease' or 'no heart disease' class.

1.1. Decision trees (with pruning):

All decision trees were created using Scikit Learn's 'DecisionTreeClassifier' function utilizing 80% (n=237) of the provided dataset (N=297) for training the decision tree and 20% (n=60) for testing. Reduced error pruning was applied to all decision trees, utilizing a post-order traversal of unpruned trees and testing for a decreased or equivalent root mean squared error (RMSE) following the 'pruning' of a node. Learning curves compared mean accuracy data of trees as sample size increased and are shadowed by the standard deviation (STD) of the tree accuracy data observed. 5-fold cross validation was utilized to compare training scores to test scores. Training sample sizes used in learning curves were 1, 10%, 25%, 50%, 75%, and 100% of the 237 samples.

Equation 1. Gini Index

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2$$

The first comparison was performed using a Gini index-based split criterion when building the decision tree structure. The Gini index (Equation 1) measures the probability of a variable being wrongly classified when randomly chosen. With 0 belonging to a specific outcome, and 1 randomly distributed across outcomes. In the case of the Cleveland heart disease data, whether a patient has heart disease or not.

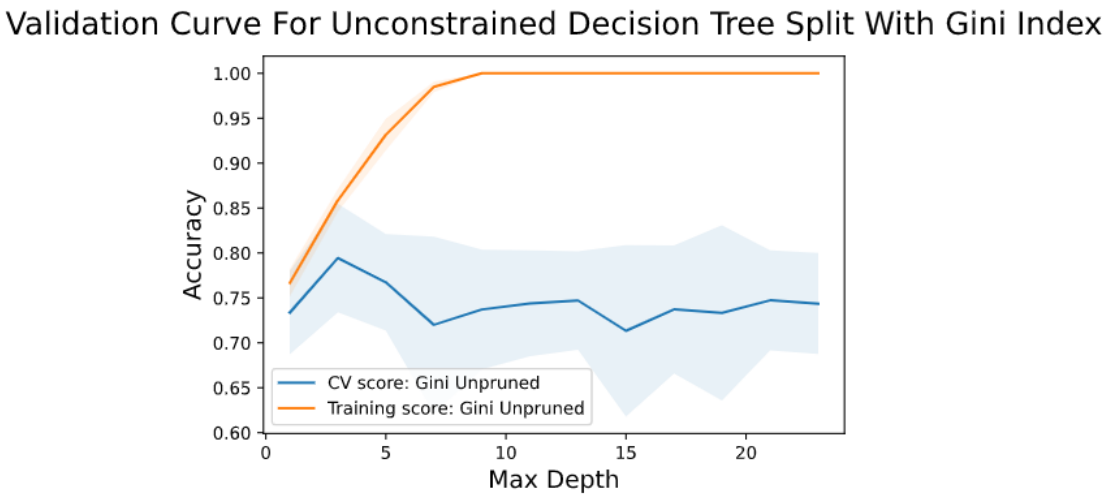

Figure 1. Validation Curve for Unpruned Tree (Gini)

A validation curve was applied to an unconstrained graph using the Gini index as splitting criterion (Figure 1) and clearly demonstrated overfitting, as seen by the level of disparity between the training and

cross validation scores. After trial and error, a max depth of 3 was determined to be the best constraint to apply prior to pruning the decision tree. Before applying the depth constraint, the unpruned decision tree had an RMSE of 0.516 and an accuracy of 0.733. Following the application of the depth constraint, the RMSE of the decision tree had decreased to 0.447 and the accuracy increased to 0.8. The decision tree was then subjected to reduced error pruning whereby the decision tree saw no discernible change in RMSE and accuracy (0.447 and 0.8, respectively). This is further exemplified by the little differences in decision tree structures (Figure 2a, c) and learning curve data (Figure 2b, d).

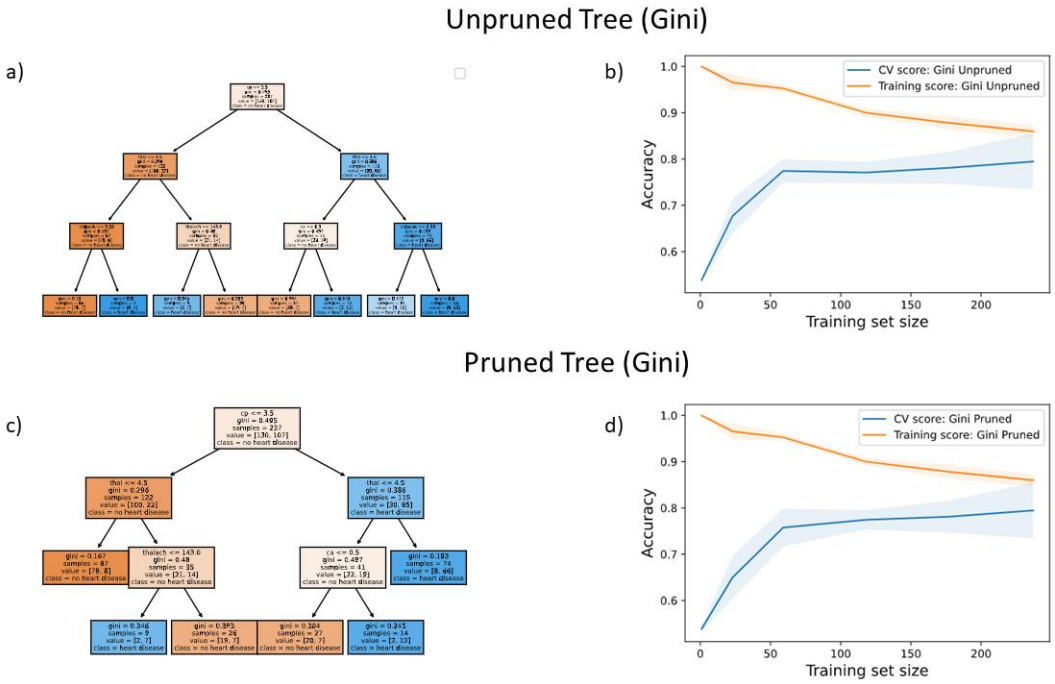

Figure 2. Unpruned Decision Tree and Learning Curve Using Gini Split Criterion

The second comparison was performed using Entropy (Information Gain)-based split criterion when building the decision tree structure. Entropy (Equation 2) measures the disorder of a variable based on the probability of a variable being classified into a target class. If there is a 50:50 chance of the variable being in either target class, the entropy will be high (1). As the likelihood of a variable being put into a target class increases, entropy decreases to the lowest point of 0.

Equation 2. Entropy

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

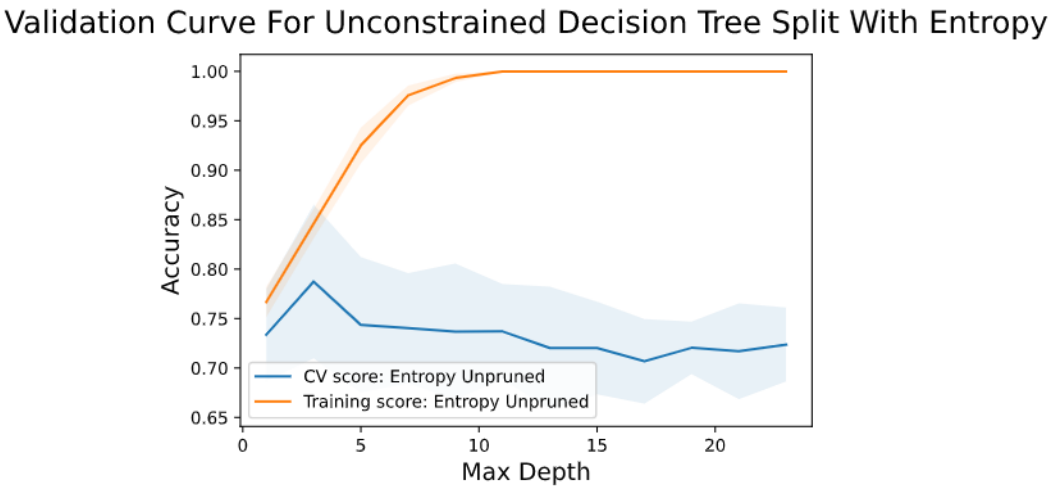

Figure 3. Validation Curve for Unpruned Tree (Entropy)

A validation curve was applied to an unconstrained graph using Entropy as splitting criterion (Figure 3) and clearly demonstrated overfitting as well. Following trial and error, a max depth of 9 was determined to be the best constraint to apply prior to pruning the decision tree. Before applying the depth constraint, the unpruned decision tree had an RMSE of 0.483 and an accuracy of 0.766. The values remained unchanged following the application of the depth constraint (RMSE = 0.483, accuracy = 0.766). However, the noticeable changes occurred following the decision tree's subjection to reduced error pruning. The RMSE decreased to 0.408 and accuracy to 0.833. This is also shown in the large amount of pruned leaves from the original decision tree (Figure 4a, c). The learning curves share a similar pattern, albeit the STD appears slightly smaller in the pruned tree suggesting a higher degree of homogeneity in the data (Figure 4b, d).

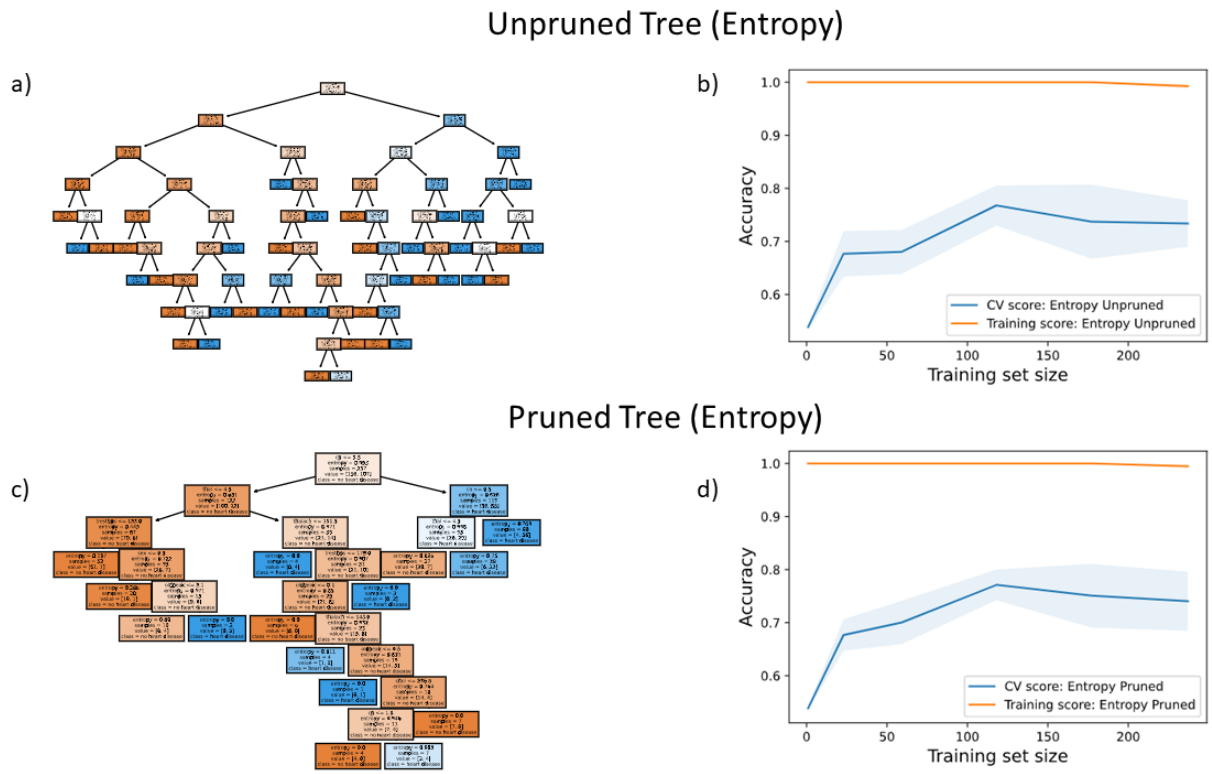

Figure 4. Unpruned Decision Tree and Learning Curve Using Entropy Split Criterion

As the accuracy values of the entropy-based split decision tree are higher and RMSE values lower (Entropy: RMSE = 0.408, accuracy = 0.833, relative to Gini: RMSE = 0.447, accuracy = 0.8), it would be the better model to classify patients with heart disease according to the features outlined within this dataset.

1.2. Random forests (no pruning):

All random forests were created using Scikit Learn's 'RandomForestClassifier' function utilizing 80% (n=237) of the provided dataset (N=297) for training the forest and 20% (n=60) for testing. All sample sizes were sampled with replacement through the bootstrap setting of the function. Learning curves compared mean accuracy data of trees as sample size increased and are shadowed by the standard deviation (STD) of the tree accuracy data observed. 5-fold cross validation was utilized to compare training scores to test scores. Training sample sizes used in learning curves were 1, 10%, 25%, 50%, 75%, and 100% of the 237 samples. A max depth of 3 was determined through trial and error and chosen due to learning curve data training curves best matching their cross-validation scores whilst maintaining a high accuracy. Observations were made comparing 3 different feature sizes used to create the random forests: 2 features, square root (d=4), and all available features (d=13). The number of trees in forests ranged from 5 to 200.

Table 1. Two Features Error and Accuracy

Number of Trees/Forest	RMSE	Accuracy
5	0.5	0.75
10	0.447	0.8
25	0.387	0.85
50	0.387	0.85
100	0.387	0.85
200	0.387	0.85

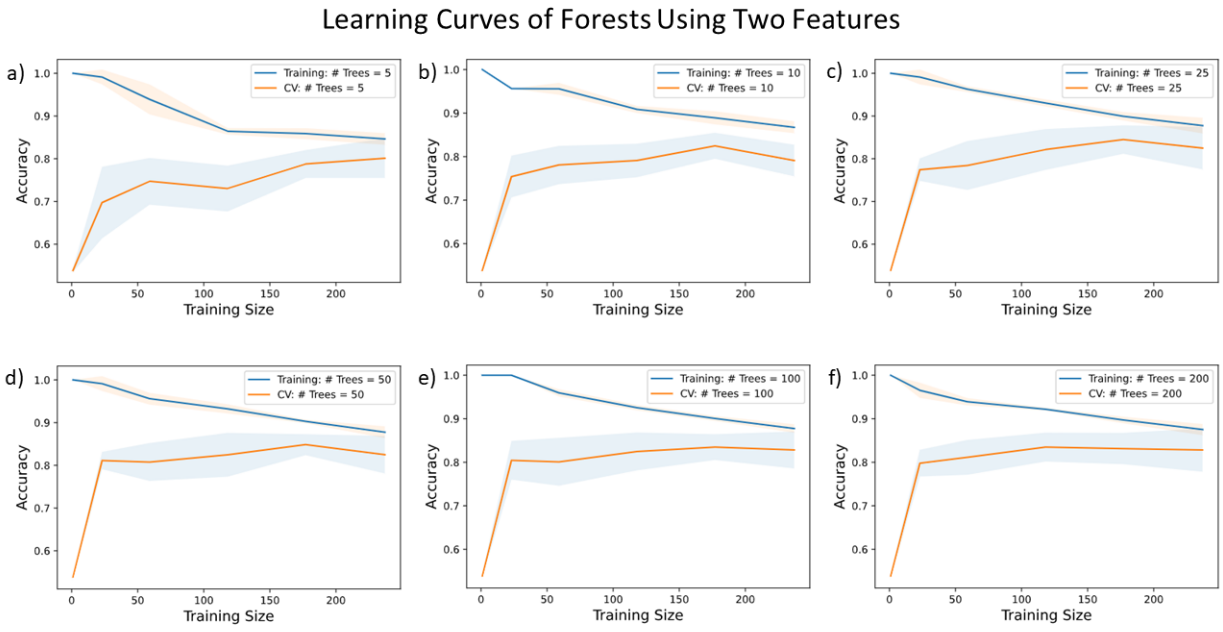

Figure 5. Learning Curve Data Two Features

When utilizing 2 features, comparisons of the learning curves (Figure 6) and the features (Table 2) of the created forests, a general trend of reduced error and increased accuracy can be seen as the tree size is

increased. Cross validation scores also show a similar trend, as their accuracy nears their training scores and linearizes with increased training size.

Table 2. Square Root Features Error and Accuracy

Number of Trees/Forest	RMSE	Accuracy
5	0.465	0.783
10	0.408	0.833
25	0.387	0.85
50	0.408	0.833
100	0.387	0.85
200	0.387	0.85

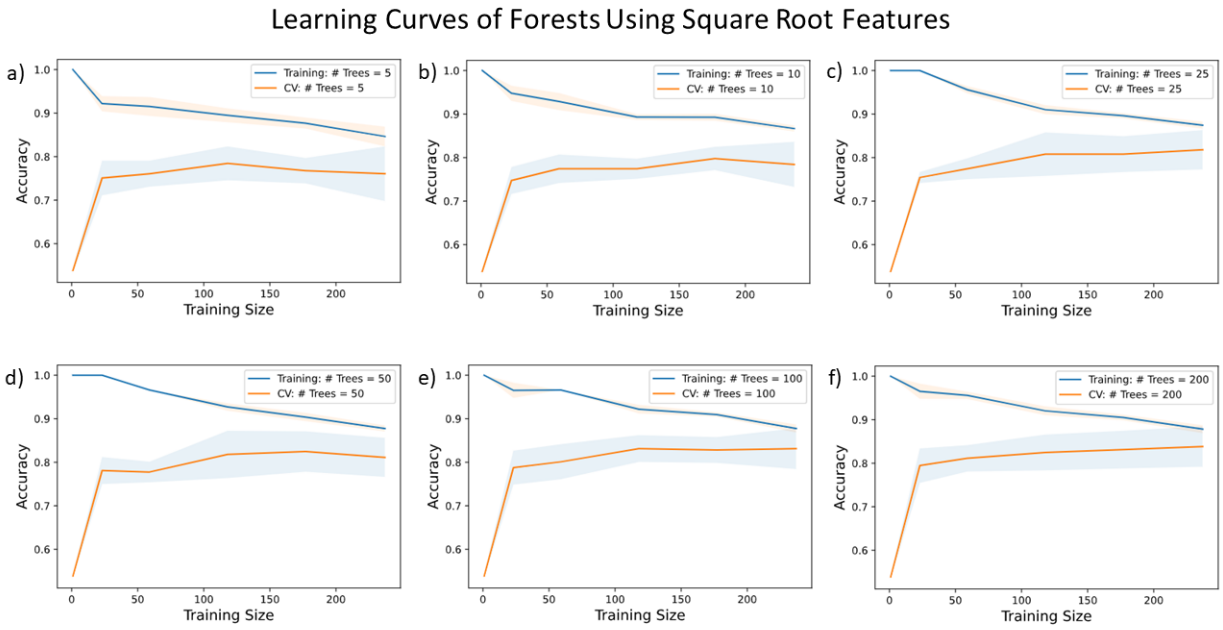

Figure 6. Learning Curve Data Square Root Features

When utilizing square root of the features, comparisons of the learning curves (Figure 6) and the features (Table 2) of the created forests, a general trend of reduced error and increased accuracy is also seen as the tree size is increased. Cross validation scores also show the same trend, as their accuracy nears their training scores and linearizes with increased training size.

Table 3. All Features Error and Accuracy

Number of Trees/Forest	RMSE	Accuracy
5	0.387	0.85
10	0.365	0.866
25	0.387	0.85
50	0.408	0.833
100	0.387	0.85

200	0.387	0.85

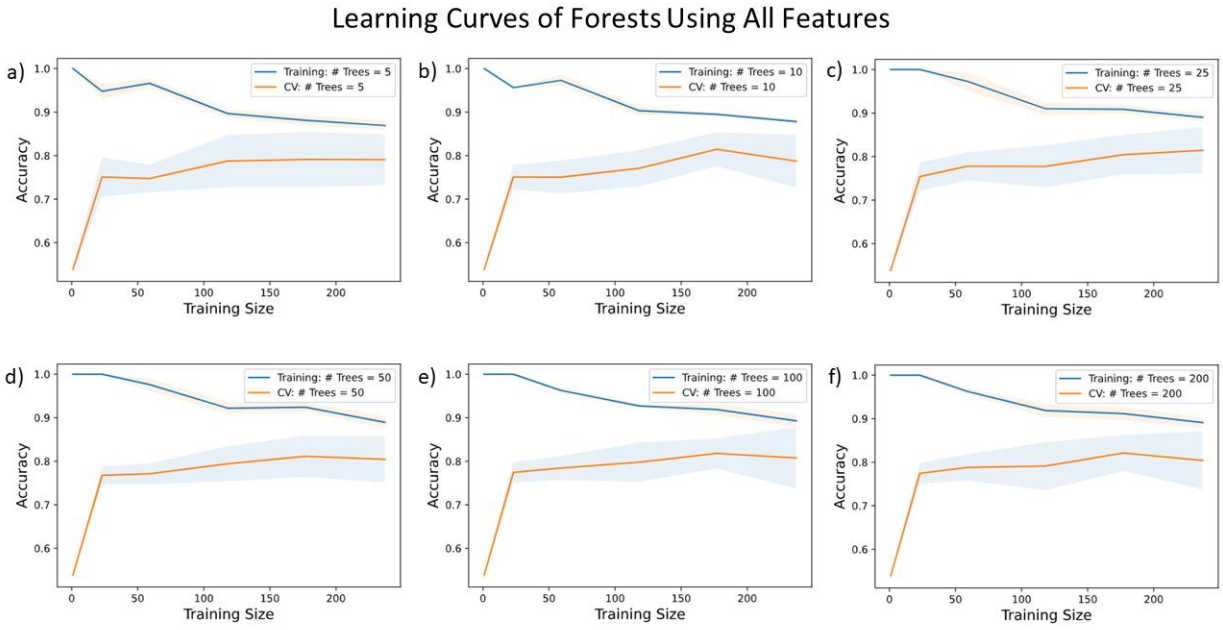

Figure 7. Learning Curve Data All Features

When utilizing all available features, comparisons of the learning curves (Figure 7) and the features (Table 3) of the created forests, a general trend of reduced error and increased accuracy is also seen as the tree size is increased; however, not to the degree of models utilizing fewer features. Although, the model with 10 trees holds the smallest RMSE (0.365) relative to others used. Interestingly, the previous trend of linearization is not present as tree size increases either.

When comparing all forests with varying feature sizes, trends suggest that only a few features and their relationships to one another are beneficial in the classification of patients with heart disease. This is shown by the relatively constant accuracy and RMSE maintained as feature count increases. Tree size appears to have a modest impact on accuracy and RMSE values as well, suggesting combinations of features play a role in classification for this dataset.

1.3. Neural networks:

All neural networks were created using Scikit Learn's 'MLPClassifier' function utilizing 80% (n=237) of the provided dataset (N=297) for training and 20% (n=60) for testing. Stochastic gradient descent was utilized for weight optimization with an initial learning rate of 0.01. All sample sizes were tested using 100, 500, and 1000 iterations of the function containing 1,3,10, or 50 neurons within the hidden layer. Results of individual iteration sizes were plotted on a loss curve and compared according to their mean accuracy when training versus when testing.

Table 4. Accuracy Training Versus Testing 100 Iterations

	Mean Training Accuracy	**Mean Test Accuracy**	**|Difference|**
1 Neuron	0.835	0.816	0.019
3 Neurons	0.873	0.833	0.040
10 Neurons	0.869	0.816	0.053
50 Neurons	0.886	0.816	0.070

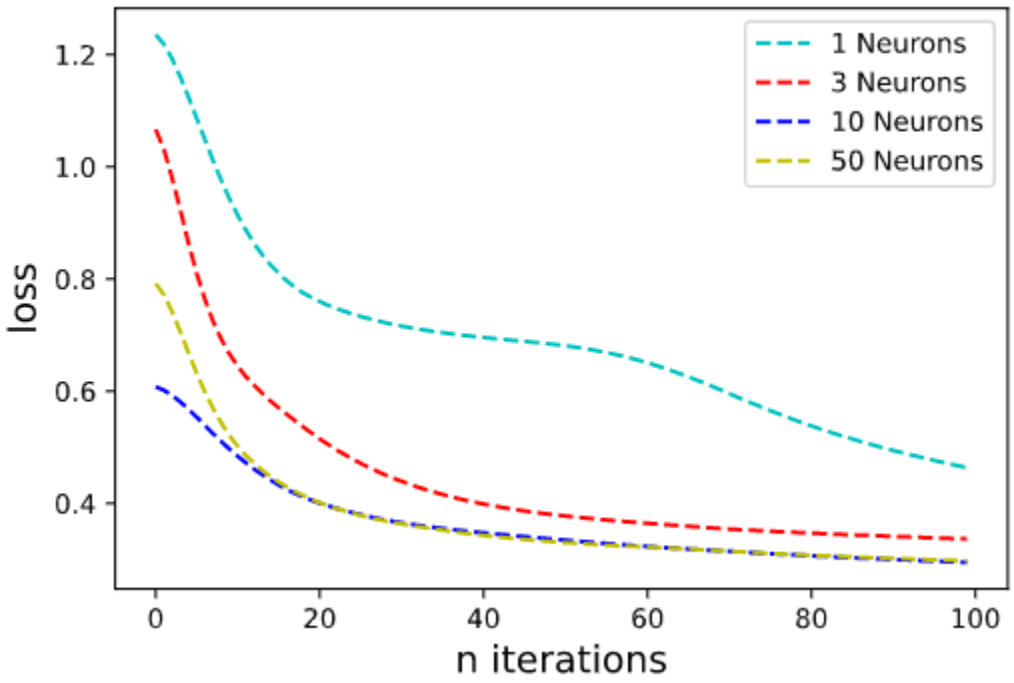

Figure 8. Loss Curve 100 Iterations

When observing the mean accuracies following 100 iterations (Table 4), we see that the training accuracy increases as the number of neurons increases. However, with the increase in training accuracy we see an increase in the difference between test accuracies with little change in test accuracy. This is suggestive of potential overfitting of data with increased neurons. The optimal amount of neurons in the hidden layer appears to be 1 as the difference is 0.019. This is enforced by the learning rate displayed in the loss curve (Figure 8), which recommends 1-3 neurons in the hidden layer.

Table 5. Accuracy Training Versus Testing 500 Iterations

	Mean Training Accuracy	Mean Test Accuracy	|Difference|
1 Neuron	0.852	0.85	0.002
3 Neurons	0.869	0.816	0.053
10 Neurons	0.936	0.733	0.203
50 Neurons	0.983	0.75	0.233

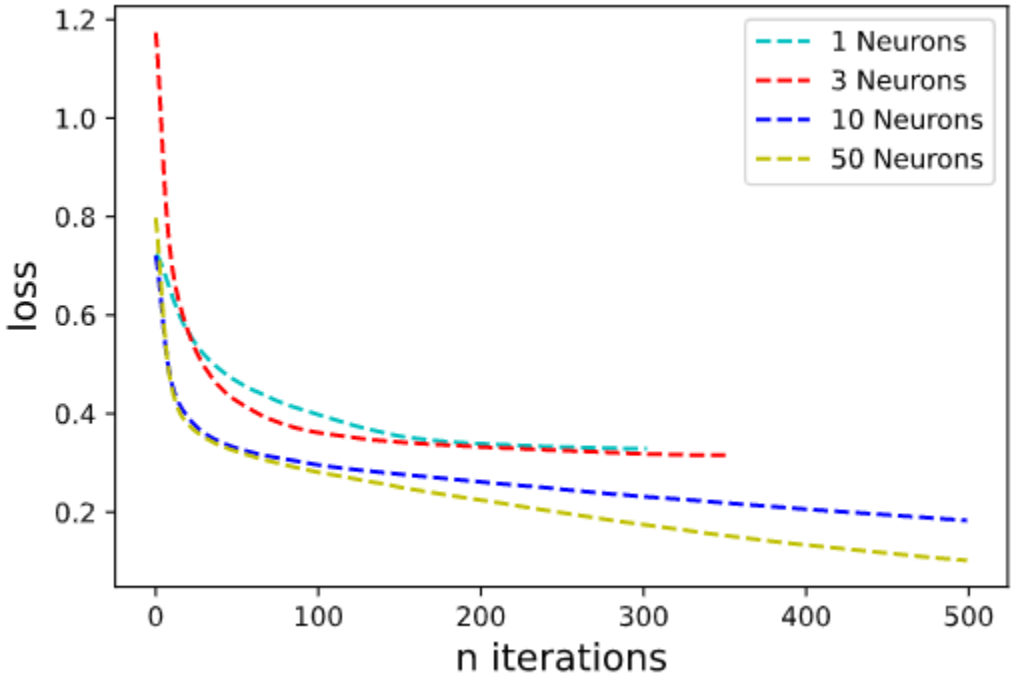

Figure 9. Loss Curve 500 Iterations

When observing the mean accuracies following 500 iterations (Table 5), we see the same trend of increasing training accuracy with increasing neurons. Apart from one neuron in the hidden layer, the increase in neurons is associated with a decrease in test accuracies. However, the mean test accuracy with this model is higher than its counterpart utilizing 100 iterations. Once again, this decrease in test accuracies is suggestive of potential overfitting of data with increased neurons. The optimal amount of neurons in the hidden layer appears to be 1 as the difference is 0.002, lower than its counterpart at 100 iterations. This is enforced by the learning rate displayed in the loss curve (Figure 9), which recommends 1-3 neurons in the hidden layer.

Table 6. Accuracy Training Versus Testing 1000 Iterations

	Mean Training Accuracy	Mean Test Accuracy	|Difference|
1 Neuron	0.860	0.833	0.027
3 Neurons	0.873	0.816	0.057
10 Neurons	0.966	0.85	0.116
50 Neurons	0.995	0.783	0.221

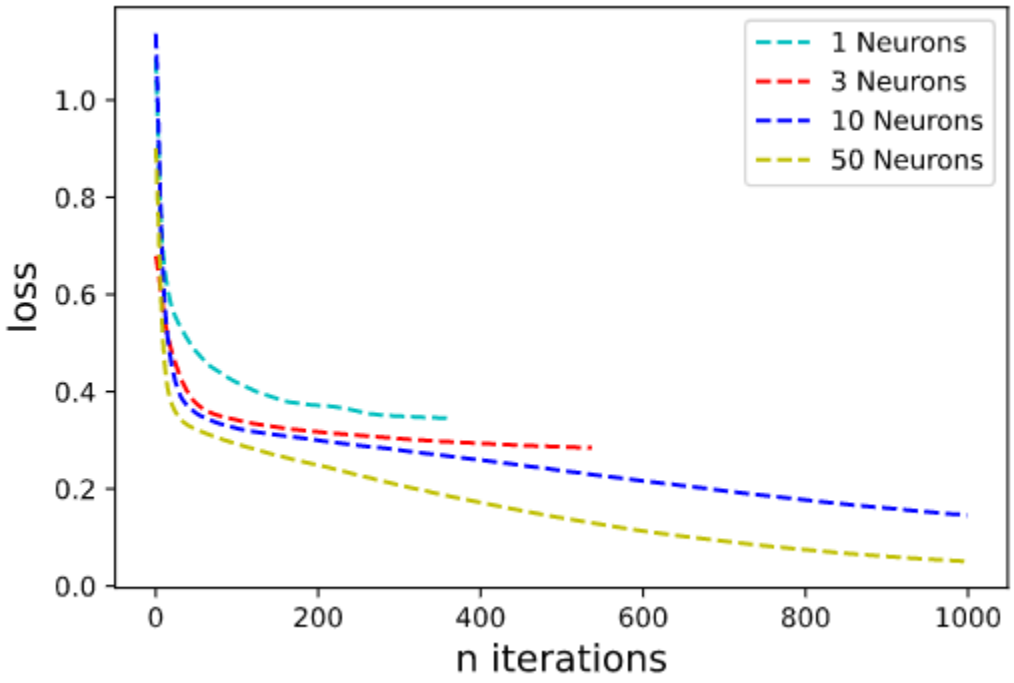

Figure 10. Loss Curve 1000 Iterations

When observing the mean accuracies following 1000 iterations (Table 6), we once again see the same general trend of increasing training accuracy and decreasing test accuracy with increasing neurons. Overall, the accuracy scores are overall better for both the training and test data. Unfortunately, the differences between the two have increased. Surprisingly, the optimal amount of neurons in the hidden layer appears to be 10 as the accuracy of both the training accuracy and test accuracy are high (0.966 and 0.85, respectively). However, their difference (0.116) is still higher than the one neuron model (0.027). This is contrary to the learning rate displayed in the loss curve (Figure 10), which recommends 1-3 neurons in the hidden layer.

As the dataset size appears to be quite low, it benefits from having fewer neurons within the hidden layer and fewer iterations. Should a starting point to fine-tune the model require choosing, between 100 to 500 iterations using 1-3 neurons in the hidden layer would likely be the optimal model to work from. This is due to the differences in training and testing data being low (0.02 at 500 iteration and 1 neuron) and the accuracies remaining high (0.852 training, 0.85 accuracy).

2. sklearn.datasets.load_breast_cancer

The load_breast_cancer dataset is a built-in dataset from the Scikit Learn library containing patient breast cancer data from the UCI ML Breast Cancer Wisconsin (Diagnostic) dataset [1]. This data has a total of 569 patients observing 30 different features ('mean radius', 'mean texture', 'mean perimeter', 'mean area', 'mean smoothness', 'mean compactness', 'mean concavity', 'mean concave points', 'mean symmetry', 'mean fractal dimension', 'radius error', 'texture error', 'perimeter error', 'area error', 'smoothness error', 'compactness error', 'concavity error', 'concave points error', 'symmetry error', 'fractal dimension error', 'worst radius', 'worst texture', 'worst perimeter', 'worst area', 'worst smoothness', 'worst compactness', 'worst concavity', 'worst concave points', 'worst symmetry', and 'worst fractal dimension') and classifying them according to whether their cancer is benign or malignant. This dataset, while classification based as well, is nearly double the size of Cleveland heart disease in both sample size and features. This will help determine the limitations of the models tested and aid in determining which models are better suited to lower sample sizes versus lower feature sizes. The dataset will look to classify test data into either the 'malignant' or 'benign' class, using real-world cancer features.

2.1. Decision trees (with pruning):

Methodology regarding the creation of decision trees remains unchanged from Section 1.1. Training the decision tree utilized 80% (n=455) of the provided dataset (N=569) and 20% (n=114) for testing. Reduced error pruning was applied to all decision trees, utilizing methodology described in Section 1.1. Learning curves compared mean accuracy data of trees as sample size increased and are shadowed by the standard deviation (STD) of the tree accuracy data observed. 5-fold cross validation was utilized to compare training scores to test scores. Training sample sizes used in learning curves were 1, 10%, 25%, 50%, 75%, and 100% of the 455 samples.

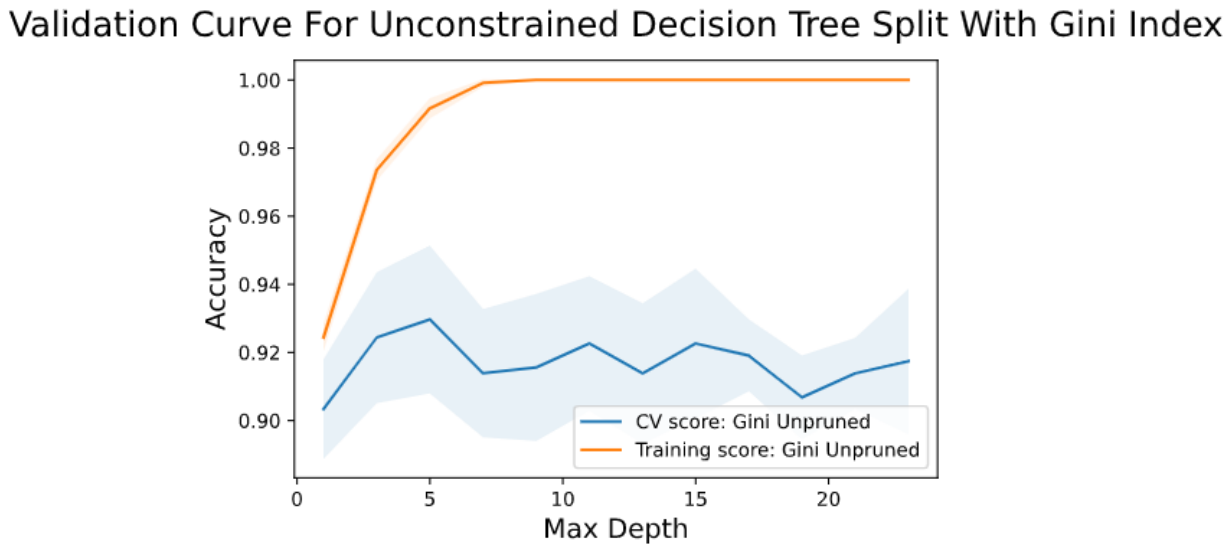

Figure 11. Validation Curve for Unconstrained Decision Tree (Gini)

The first comparison was performed using a Gini index-based split criterion (Equation 1) when building the decision tree structure, determining whether a patient had malignant disease or benign. A validation curve was applied to an unconstrained graph (Figure 11) and demonstrated some minor overfitting. After

trial and error, a max depth of 3 was determined to be the best constraint to apply prior to pruning the decision tree. Before applying the depth constraint, the decision tree had an RMSE of 0.296 and an accuracy of 0.912. Following the application of the depth constraint, the RMSE of the unpruned decision tree had decreased to 0.187 and the accuracy increased to 0.964. The decision tree was then subjected to reduced error pruning whereby the decision tree saw a further decrease in RMSE (0.162) and accompanying increase in accuracy (0.973). As shown in their decision tree structures (Figure 12a, c) a high amount of pruning had occurred, but is visually their learning curve data suggests little change. As represented by STD, only a minor increase in the heterogeneity of the data in the pruned tree (Figure 12b, d).

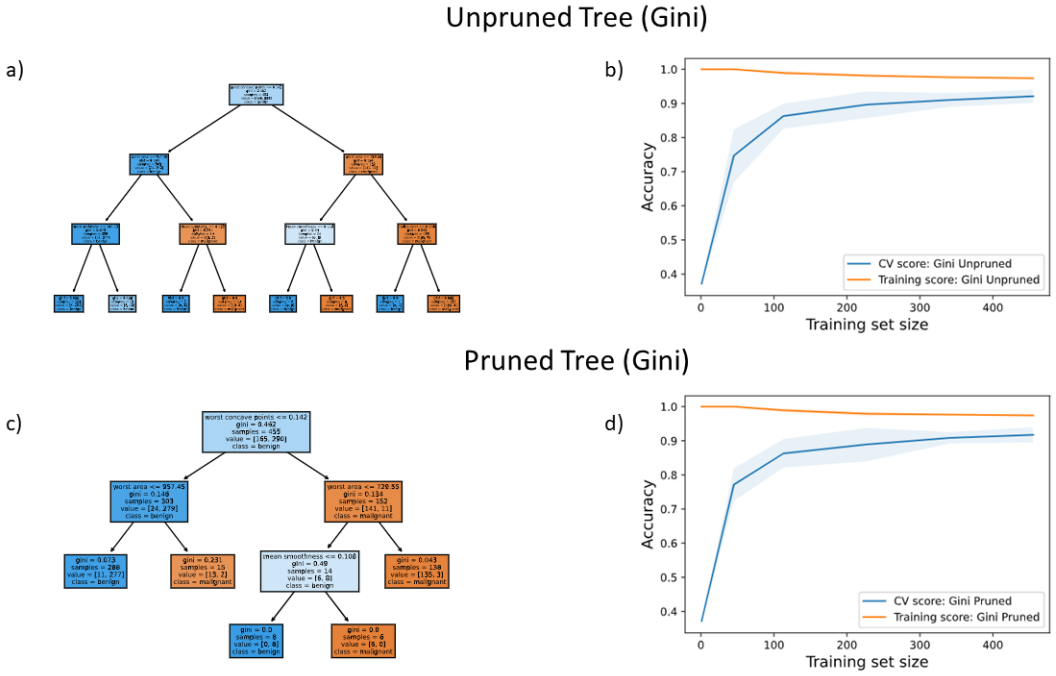

Figure 12. Unpruned Decision Tree and Learning Curve Using Gini Split Criterion

The second comparison was performed using Entropy-based split criterion (Equation 2. Entropy). A validation curve was applied to an unconstrained graph (Figure 13) and demonstrated some minor overfitting. After trial and error, a max depth of 3 was determined to be the best constraint to apply prior to pruning the decision tree, sharing the same depth restriction as the Gini index split criterion.

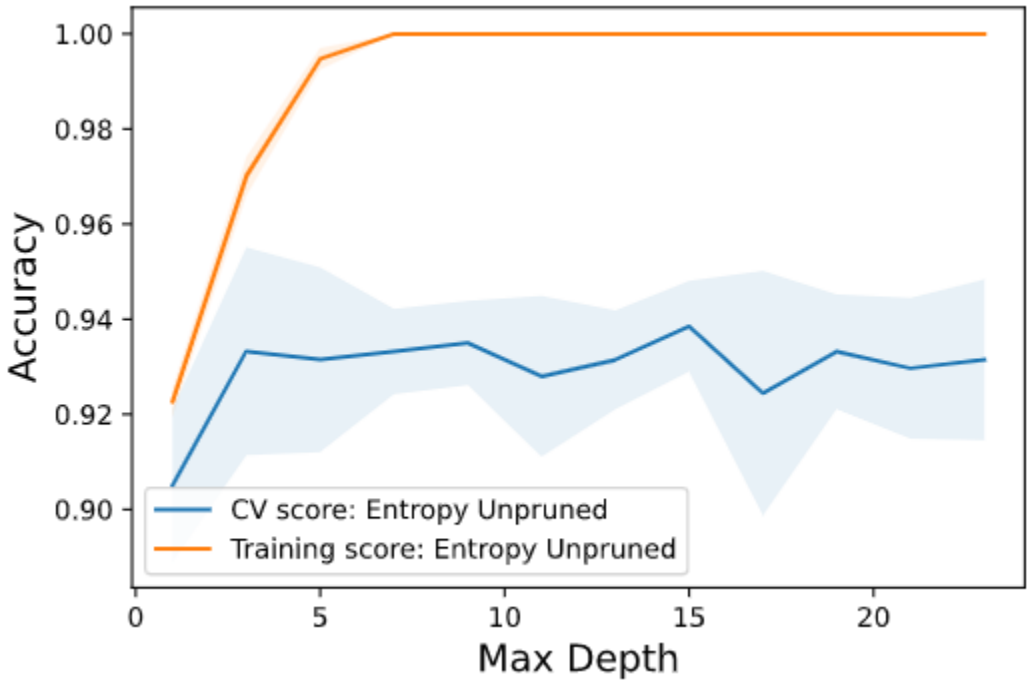

Figure 13. Validation Curve for Unconstrained Decision Tree (Entropy)

Before applying the depth constraint, the unpruned decision tree had an RMSE of 0.264 and an accuracy of 0.929. Following the application of the depth constraint, the RMSE of the decision tree had decreased to 0.229 and the accuracy increased to 0.947. The decision tree was then subjected to reduced error pruning whereby the decision tree saw no discernible change in RMSE (0.229) and accuracy (0.947). As shown in their decision tree structures (Figure 14a, c) little pruning had occurred, this is also represented in their learning curve data (Figure 14b, d).

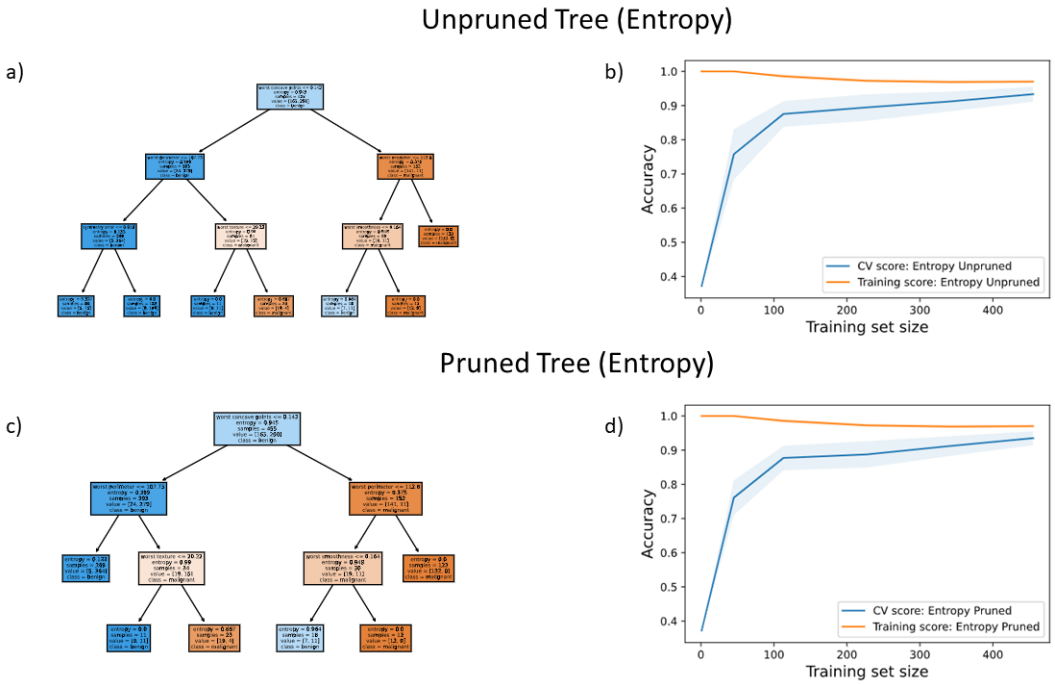

Figure 14. Unpruned Decision Tree and Learning Curve Using Entropy Split Criterion

Unlike the Cleveland heart disease dataset, the accuracy and RMSE of the Gini index-based split decision tree are superior to the entropy-based model (Entropy: RMSE = 0. 0.229, accuracy = 0.947, relative to Gini: RMSE = 0.162, accuracy = 0.973). Therefore, I would recommend the Gini split decision tree model for classifying patients with malignancy (or benign) disease according to the features outlined within this dataset.

2.2. Random forests (no pruning):

Methodology regarding the creation of random forests remains unchanged from Section 1.2. Training the models utilized 80% (n=455) of the provided dataset (N=569) and 20% (n=114) for testing. All sample sizes were sampled with replacement through the bootstrap setting of the function. Learning curves and training sample sizes used in learning curves remain the same as Section 1.2. A max depth of 3 was determined through trial and error and chosen due to learning curve data training curves best matching their cross-validation scores whilst maintaining a high accuracy. Observations were made comparing 3 different feature sizes used to create the random forests: 2 features, square root (d=5), and all available features (d=30). The number of trees in forests ranged from 5 to 200.

Table 7. Two Features Error and Accuracy

Number of Trees/Forest	RMSE	Accuracy
5	0.264	0.929
10	0.187	0.964
25	0.187	0.964
50	0.162	0.973
100	0.162	0.973
200	0.162	0.973

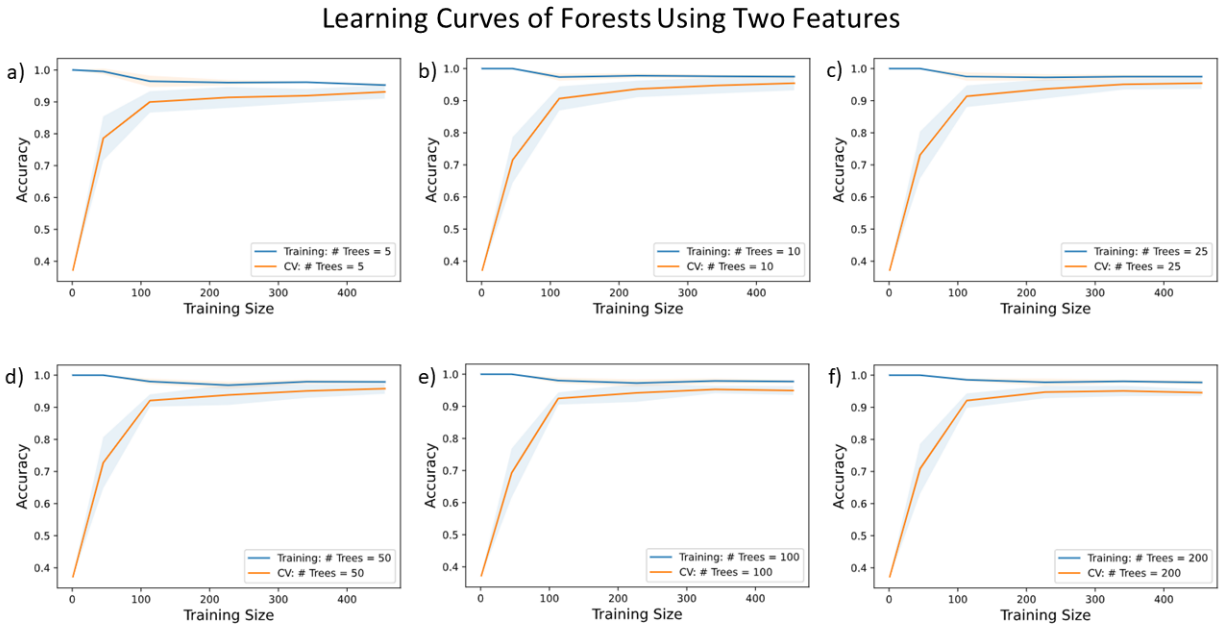

Figure 15. Learning Curve Data Two Features

When utilizing 2 features, comparisons of the learning curves (Figure 15) and the features (Table 7) of the created forests, a trend of reduced error and increased accuracy can be seen as tree size is increased. However, cross validation scores show a trend of closely approaching training scores when the number of trees is between 25 and 50 and remaining relatively constant in accuracy for the higher tree sizes.

Table 8. Square Root Features Error and Accuracy

Number of Trees/Forest	RMSE	Accuracy
5	0.162	0.973
10	0.162	0.973
25	0.132	0.982
50	0.132	0.982
100	0.132	0.982
200	0.132	0.982

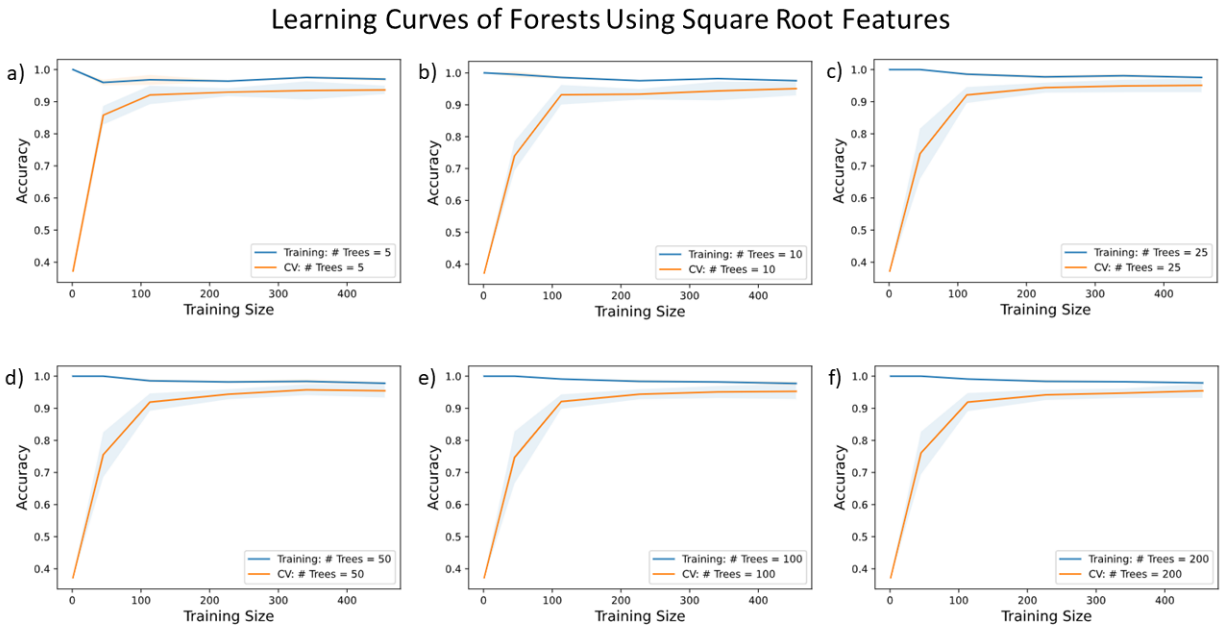

Figure 16. Learning Curve Data Square Root Features

When utilizing square root of the features, comparisons of the learning curves (Figure 16) and the features (Table 8) of the created forests, a general trend of reduced error and increased accuracy is also seen as the tree size is increased until a tree size between 10 and 25. Following this, tree sizes appear to have no impact on RMSE and accuracy. Cross validation scores also show the same trend, as their overall shape remains constant from a tree size of 25 and over.

Table 9. All Features Error and Accuracy

Number of Trees/Forest	RMSE	Accuracy
5	0.132	0.982
10	0.132	0.982
25	0.132	0.982
50	0.132	0.982
100	0.132	0.982
200	0.093	0.991

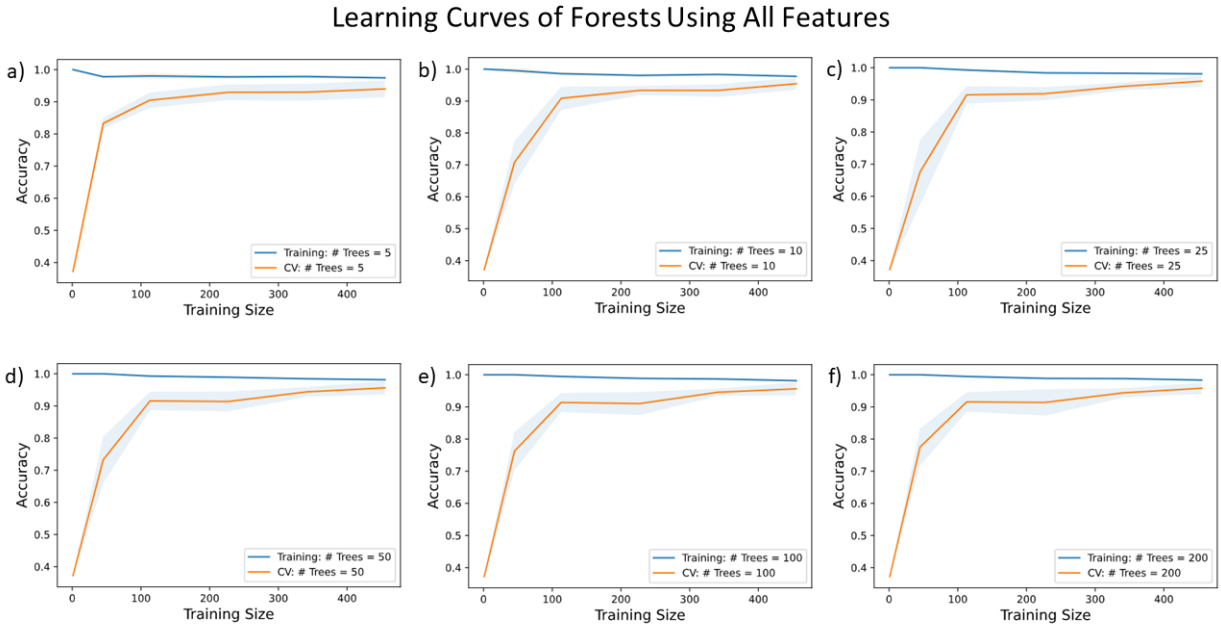

Figure 17. Learning Curve Data All Features

When utilizing all available features, comparisons of the learning curves (Figure 17) and the features (Table 9) of the created forests, RMSE and accuracy remains constant until a tree size between 100 and 200. Following this a small increase in accuracy (0.991 from 0.982) and a reduction in RMSE (0.093 and 0.991) is observed. The previous trend of linearization is not as clearly present as trees with fewer features; however, the cross validation and training accuracies appear to be far closer to one another.

When comparing all forests with varying feature sizes, trends suggest that most, if not all features are beneficial in the classification of patients with breast cancer. This is shown by the increasing accuracy and decreasing RMSE as feature count increases. Tree size appears to have a modest impact on accuracy and RMSE values as well, suggesting combinations of features play a role in classification for this dataset.

2.3. Neural networks:

Methodology regarding the creation of neural networks remains unchanged from Section 1.3. Training the models utilized 80% (n=455) of the provided dataset (N=569) and 20% (n=114) for testing. Stochastic gradient descent was utilized for weight optimization with an initial learning rate of 0.01. All sample sizes were tested using 100, 500, and 1000 iterations of the function containing 1,3,10, or 50 neurons within the hidden layer. Results of individual iteration sizes were plotted on a loss curve and compared according to their mean accuracy when training versus when testing.

Table 10. Accuracy Training Versus Testing 100 Iterations

	Mean Training Accuracy	**Mean Test Accuracy**	**|Difference|**
1 Neuron	0.969	0.929	0.040
3 Neurons	0.982	0.973	0.009
10 Neurons	0.986	0.947	0.039
50 Neurons	0.984	0.938	0.046

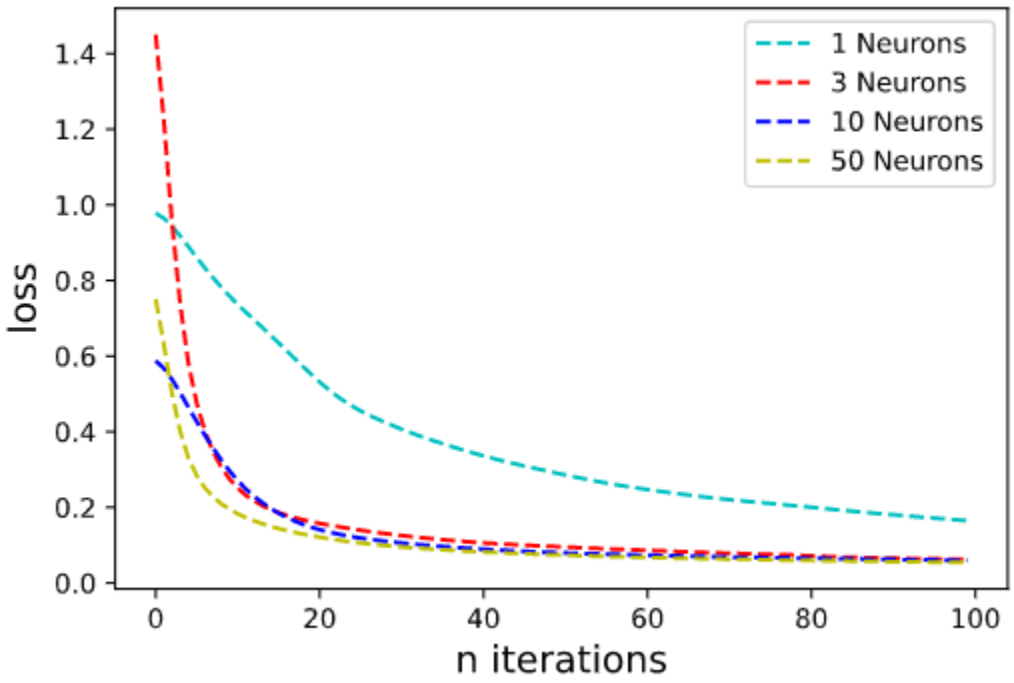

Figure 18. Loss Curve 100 Iterations

When observing the mean accuracies following 100 iterations (Table 10), we see that the training accuracy increases as the number of neurons increases. However, with the increase in training accuracy we see curve-like effect in test accuracies with the peak at 3 neurons. The differences in both training and test accuracies is minimal as well (0.009). This is suggestive of potential underfitting at 1 neuron, followed by overfitting at greater than 3 neurons. The optimal amount of neurons in the hidden layer appears to be 3 but given the choice of jumping from 3 to 10 neurons, further investigation is required.

This is partially enforced by the learning rate displayed in the loss curve (Figure 18); however, 1 neuron appears to be optimal.

Table 11. Accuracy Training Versus Testing 500 Iterations

	Mean Training Accuracy	Mean Test Accuracy	|Difference|
1 Neuron	0.984	0.982	0.002
3 Neurons	0.991	0.982	0.009
10 Neurons	0.991	0.973	0.018
50 Neurons	0.991	0.982	0.009

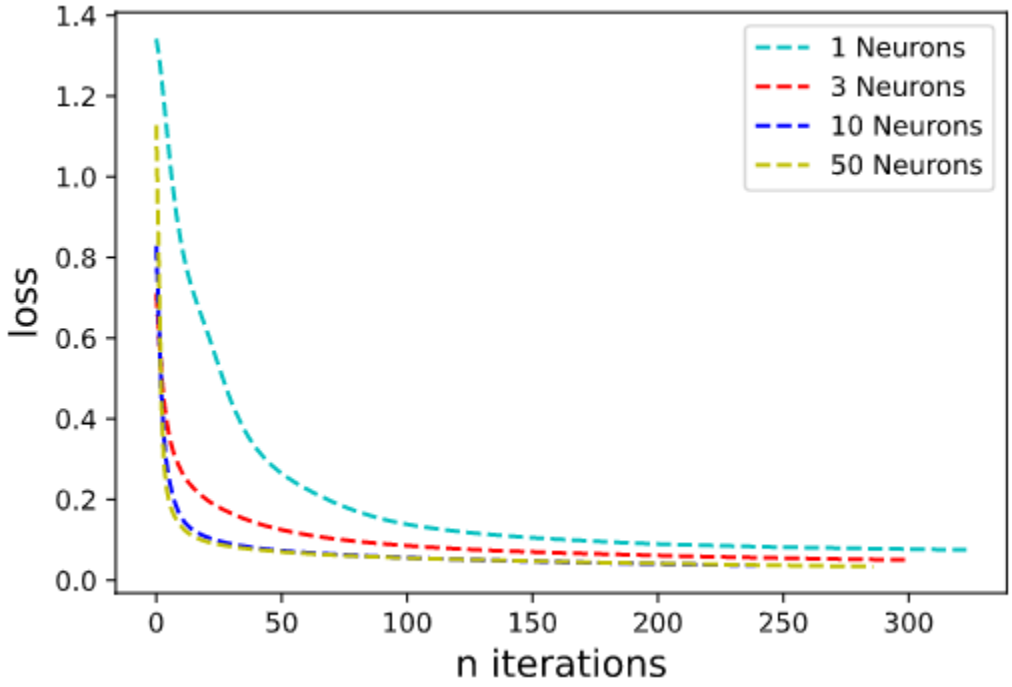

Figure 19. Loss Curve 500 Iterations

When observing the mean accuracies following 500 iterations (Table 11), we see the same trend of increasing training accuracy with increasing neurons. The overall trend of decreasing test accuracy with increasing neurons is also present. However, overall accuracy for both test and training data is much higher than after 100 iterations and differences are generally lower. The optimal amount of neurons in the hidden layer appears to be 1 as the difference is 0.002, lower than the 3 neurons suggested after 100 iterations. This is further enforced by the learning rate displayed in the loss curve (Figure 19).

Table 12. Accuracy Training Versus Testing 1000 Iterations

	Mean Training Accuracy	Mean Test Accuracy	|Difference|
1 Neuron	0.982	0.991	0.009
3 Neurons	0.982	0.973	0.009
10 Neurons	0.984	0.991	0.007

50 Neurons	0.991	0.982	0.009

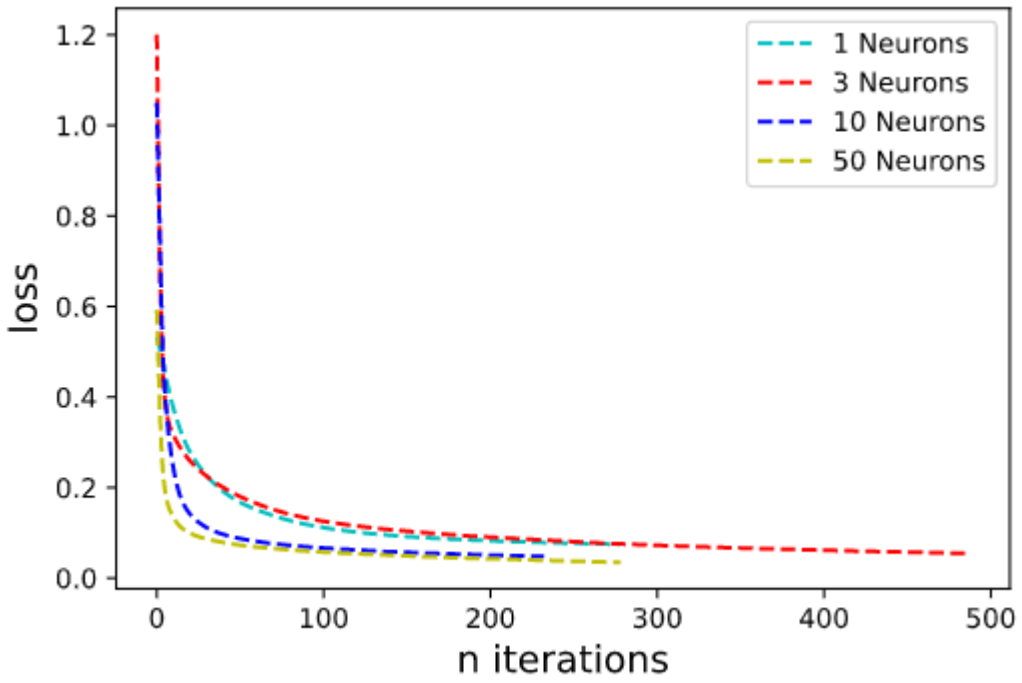

Figure 20. Loss Curve 1000 Iterations

When observing the mean accuracies following 1000 iterations (Table 12), we once again see the same general trend of increasing training accuracy with increasing neurons. However, training accuracy appears to not display the previously decreasing or bell-like distributions seen with fewer iterations. Overall, the accuracy scores are better for both the training and test data and the differences between the two lower. Surprisingly, the optimal amount of neurons in the hidden layer appears to be 10 as the accuracy of both the training accuracy and test accuracy are high (0.984 and 0.991, respectively). This is contrary to the learning rate displayed in the loss curve (Figure 20), which recommends 1-3 neurons in the hidden layer.

This dataset was difficult to interpret regarding an optimal model to use. Overall, this dataset benefitted from a model with higher iterations and 1-3 hidden neurons in the hidden layer. The increased dataset size relative to the Cleveland heart disease dataset utilized in section 1 is the most likely reason for the increased power with increased iterations and neurons in the hidden layer. However, the dataset is still quite small, and a different weighing optimization strategy may be called for in future experiments.

3. Closing Remarks and Conclusions Regarding Models

As the size and overall homogeneity of the dataset increases, the Gini Index splitting criteria appears to be the better method of categorization relative to Entropy. However, for success the Gini index relies on a much lower depth and, in turn, fewer decisions to be made. This would suggest that Gini would benefit from fewer features that more clearly define the of classification problem. Entropy may be better at handling superfluous or widely distributed features.

Regarding random forests, they appear to be much more efficient at eliminating superfluous or unnecessary features of a dataset relative to decision trees. This was shown in the Cleveland heart disease dataset. However, their power appears to be greatly increased should features be necessary in the classification of variables. This was easily exemplified in the breast cancer dataset, as a random forest utilizing all variables and 200 of trees had an accuracy of 0.991 and RMSE or 0.093.

These level of predicting power appears to be shared with neural networks with higher sample sizes. As shown by the breast cancer dataset, the accuracy of test data was comparable to the random forest at 0.991 when using 10 neurons at 1000 iterations. Future experimentation with neural networks is warranted as the stochastic gradient descent was utilized for weight optimization. Strategies like Scikit Learn's 'lbfgs' Quasi Newton solver, is recommended by Scikit Learn for smaller datasets and may be better suited for these datasets [2].

References:

[1] “Sklearn.datasets.load_breast_cancer.” Scikit, scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html.

[2] “Sklearn.neural_network.MLPClassifier.” Scikit, scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html