Problem 1: Heart Disease Data

1.1 Training and Testing of decision trees (Testing size: 20%)

In this method, the data set was split into two subsets. One subset used for training the decision tree (80% of the original dataset) and the other subset used as the testing set (20% of the original dataset). When conducting experiments on this method, I recorded the training and testing accuracy of the datasets before and after pruning the tree.

For pruning, I utilized sklearns cost_complexity_pruning; it is important to note that when the pruning was done, the decision tree was fitted with training data and training labels with different values for the complexity parameter (alpha). With that being said, the graph (Figure 1 and 2) for the pruned tree was created by using the different values of alpha as the x-axis and the y-axis consisted of the different accuracy scores depending on the alpha used to train the tree.

Furthermore, to be able to analyze the methods involved in decision trees I conducted this experiment twice, one experiment was conducted using 'Entropy' as the split criterion and the other one employed the Gini Index for the split criterion. The table below shows the different values assumed by the training and testing accuracy for the different split criterions and shows the accuracies before and after the pruning. Figure 1 shows the graph of how the accuracies for training and testing change with the variation of alphas when using Entropy and Figure 2 highlights the same information with respect to Gini Index as the split criterion.

<table>
<tr><th rowspan="2"></th><th colspan="4">Decision Tree</th></tr>
<tr><th colspan="2">Before Pruning</th><th colspan="2">After Pruning</th></tr>
<tr><td rowspan="4">Entropy</td><td rowspan="4">Training: 1.0</td><td rowspan="4">Test: 0.7</td><td>Training:</td><td>Testing:</td></tr>
<tr><td>Min: 0.549</td><td>Min: 0.5</td></tr>
<tr><td>Max: 1.0</td><td>Max: 0.816</td></tr>
<tr><td>Mean: 0.880</td><td>Mean: 0.751</td></tr>
<tr><td rowspan="4">Gini Index</td><td rowspan="4">Training: 1.0</td><td rowspan="4">Test: 0.75</td><td>Training:</td><td>Testing:</td></tr>
<tr><td>Min: 0.549</td><td>Min: 0.5</td></tr>
<tr><td>Max: 1.0</td><td>Max: 0.833</td></tr>
<tr><td>Mean: 0.894</td><td>Mean: 0.776</td></tr>
</table>

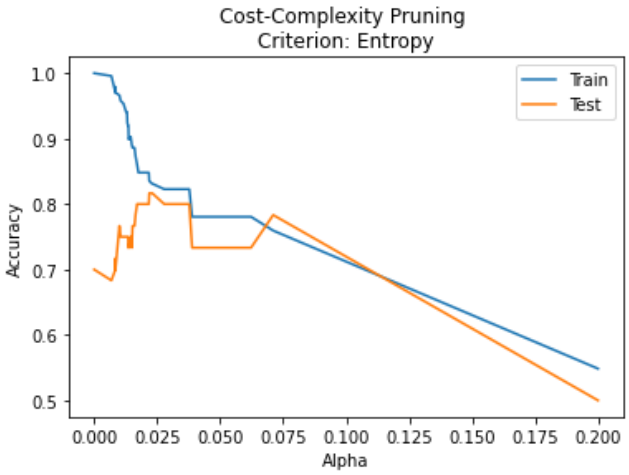


Figure 1: Accuracies for Entropy

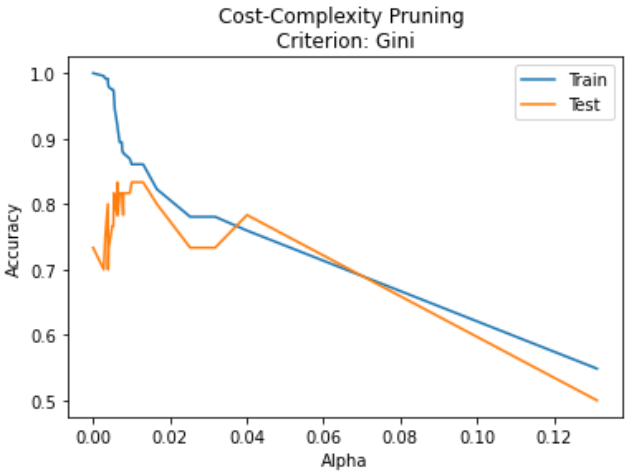


Figure 2: Accuracies for Gini Index

From the table and the figures, we can deduce certain features about the methods used when classifying datasets using decision trees. Just by scanning over the table of recorded accuracies it can be noticed that before pruning the tree, Entropy and Gini Index have the same training accuracy, however, the accuracy of Gini Index on the testing examples is better than that of Entropy (though not by a large factor). This trend can further be noticed when the classification is conducted on the pruned tree; the minimum and maximum accuracy for the training set using different values of alpha within the two split criterions is the same but Gini Index has a more accurate mean. Additionally, when applying the classification on the testing samples, Gini Index has a greater maximum accuracy as well as a mean accuracy.

The graphs aid in justifying the data from the tables, from figure 2 you can see that the training accuracy for the Gini Index has a more gradual descent when the alphas are increased as compared to that of Entropy. With respect to the testing samples, the graphs are almost identical, however, if you focus on alpha between in the range (0,0.05), it is evident that the accuracies for the Gini Index remain higher than that of Entropy despite fluctuating more.

All in all, from the table and the graphs it can be concluded that using Gini Index as the split criterion results in a higher training and testing accuracy regardless of the tree being pruned or not. Additionally, with respect to each split criterion, the accuracy of training and testing is better when the value of alpha is closer to 0. There is also an alpha such that, from that point on, the accuracies gradually decline as the alpha increases.

1.2 Training and Testing of Random Forests (Testing size: 20%)

Splitting of the data for these methods was done in the same fashion as described in section 1.1. The split criterions employed were Entropy and Gini Index. Within each of these split criterions, the number of trees in the forest was changed and the accuracies were recorded.

To showcase enough data was collected, this method was repeated for a total of 20 times and each time the number of trees was incremented by a factor of 50 (the range of trees was 50-1000). The table below shows the training and testing accuracies with each split criterion.

Furthermore, figures 3 and 4 show the graphs of the accuracies with respect to each split criterion. In these graphs the x-axis is the change in number of trees in the forest and the y-axis consists of the accuracies with respect to the number of trees.

Random Forests			
Entropy		**Gini Index**	
Training:	Testing:	Training:	Testing:
Min: 1.0	**Min:** 0.833	**Min:** 1.0	**Min:** 0.833
Max: 1.0	**Max:** 0.883	**Max:** 1.0	**Max:** 0.883
Mean: 1.0	**Mean:** 0.866	**Mean:** 1.0	**Mean:** 0.859

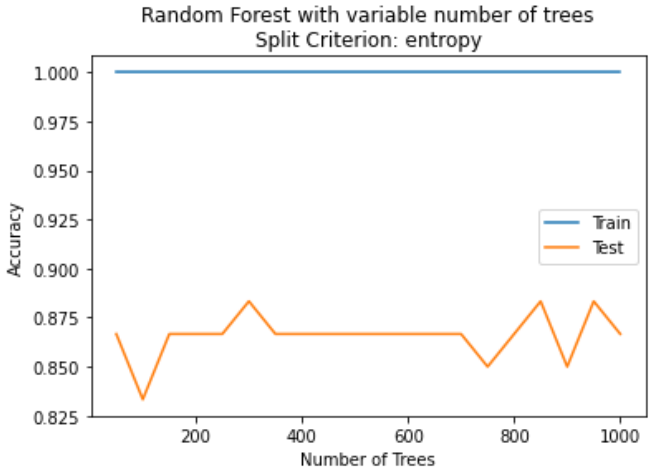


Figure 3: Accuracy for Entropy

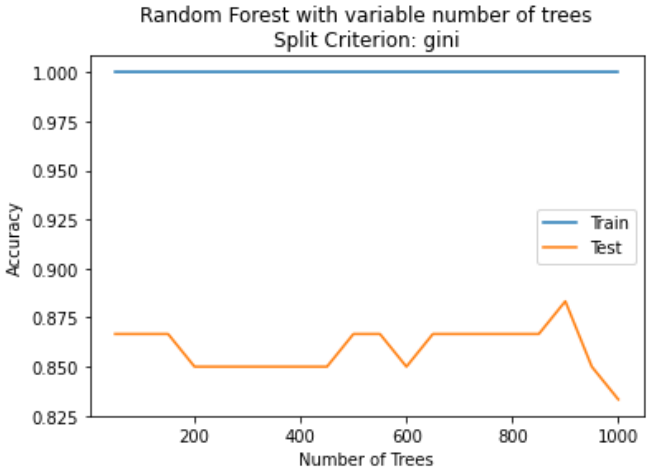


Figure 4: Accuracy for Gini Index

Looking at the table for the different statistics on the accuracies of the testing and training datasets when using random forests for classification it can be deduced that the two split criterions are essentially the same when it comes to reliability of correctness. On the training datasets the two methods perform the same and for the testing dataset the only difference can be seen in the mean accuracy (Entropy performs slightly better than Gini Index).

From the two graphs (figure 3 & 4) we can see that the testing accuracies for both the methods are predominantly between 0.850 and 0.875. However, it is evident that Gini Index is more stable than Entropy for the testing data and does not fluctuate between the values too much. Although, upon closer inspection (figure 5) it is noticed that as the number of trees in the forest increase, Entropy preforms slightly better than Gini Index.

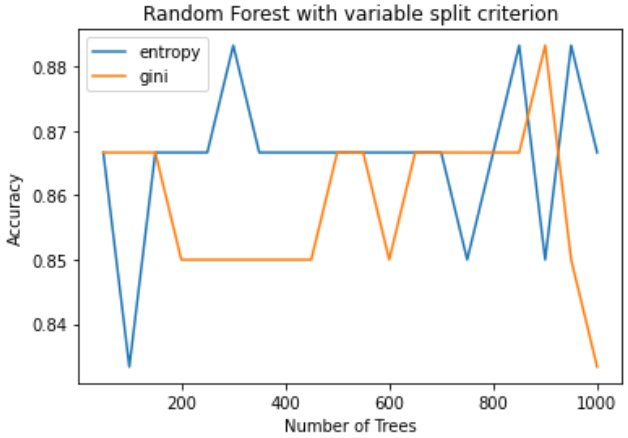


Figure 5: Entropy VS Gini Index

In conclusion, it can be said that Entropy performs slightly better than Gini Index when using a variable number of trees in the forest. However, the difference between the two split criterions is so miniscule that one claim both the methods to perform the same over a range of forest sizes.

1.3 Training and Testing of Neural Networks (Testing size: 20%)

The experiments pertaining neural networks were done in two ways, one way changed the number of neurons present in the hidden layer while keeping the other hyperparameters the same and the other method changed the magnitude of the regularization parameter while keeping the other hyper parameters the same. It is important to note that these experiments were conducted after the data was scaled.

The table below shows the different values for the training and testing accuracies as and when the size of the hidden layer changes when using SGD and Adam as the solvers (figure 6 graphically represents the data for SGD and figure 7 for Adam). With respect to SGD, we can conclude from that that as the number of neurons increase the training accuracy gets stronger and approaches a value of 1, however, the testing accuracy gets smaller and smaller moving away from 1 but eventually flattens and assumes a constant value.

	Neural Networks: Using SGD		**Neural Networks: Using Adam**	
Number of neurons	Training	Testing	Training	Testing
3	0.873	0.850	0.911	0.783
7	0.911	0.833	0.987	0.767
13	0.991	0.800	1.0	0.783
21	0.991	0.800	1.0	0.817

A reason why the testing accuracy degrades while the training accuracy increases is that as the number of neurons increase the neural net can better estimate the testing examples and as a result the testing accuracy drops and eventually flattens.

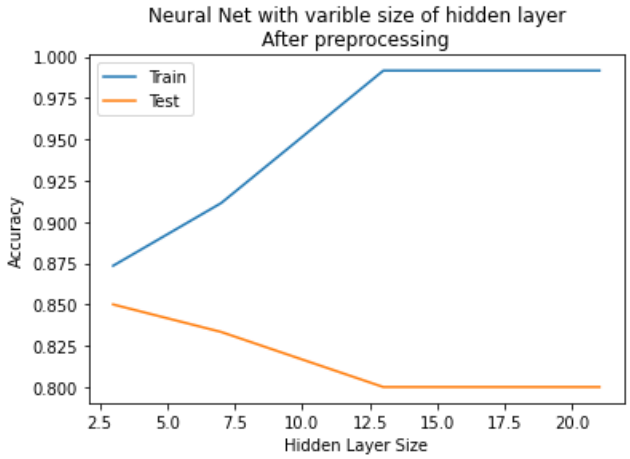


Figure 6: Neural Net Using SGD

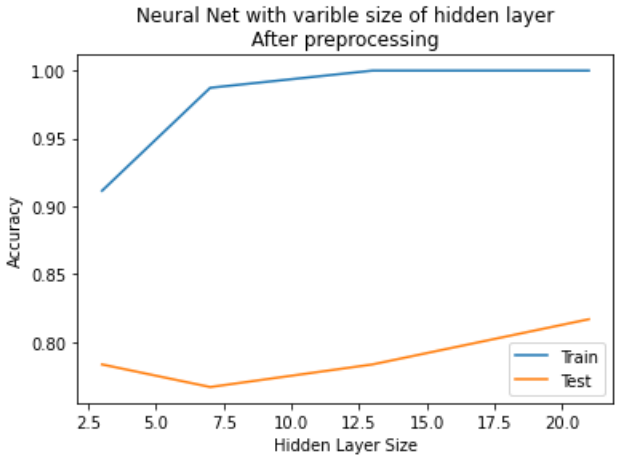


Figure 7: Neural Net Using Adam

When using Adam as the solver, we see the opposite. In this case, as the number of neurons increase, the training and testing accuracy increases (approaching 1). Although, there is a point for which the testing accuracy is an at all-time low and after which it picks up again and starts approaching 1, which is not the case of SGD. From the two graphs and tables it can be said that Adam performs better when increasing the size of the hidden layer.

The table below represents data of the two solvers and how the accuracies change when the value of alpha changes, the accompanying graphs are figures 8 and 9. From the table and the graphs it can be concluded that as the value of alpha gets smaller and smaller the training accuracy increases for both the solvers, however, the testing accuracy degrades.

	Neural Networks: Using SGD		**Neural Networks: Using Adam**	
Alpha	Training	Testing	Training	Testing
0.001	0.987	0.733	0.991	0.750
0.01	0.983	0.767	0.991	0.767
0.1	0.983	0.750	0.987	0.733
1	0.873	0.816	0.881	0.833

The graphs aid in visualizing the trend noticed in the table; we expect the graph for the training accuracies to increase as alpha reduces (right to left on the graph) and this can be confirmed by the blue lines on both plots. With respect to the testing data, our table indicates that the graph will approach 0 as alpha gets smaller. Looking at the orange lines on both plots we can conclude that this indeed happens. For both the solvers there is an alpha for which the testing accuracy is the lowest after which it starts to climb back up. Lastly, the table allows us to claim that the Neural Net performs better when using Adam as the solver as the initial and final testing accuracy is higher than that of SGD. However, if we disregard alpha = 0.001 then the opposite can be said, SGD has a better overall performance than Adam.

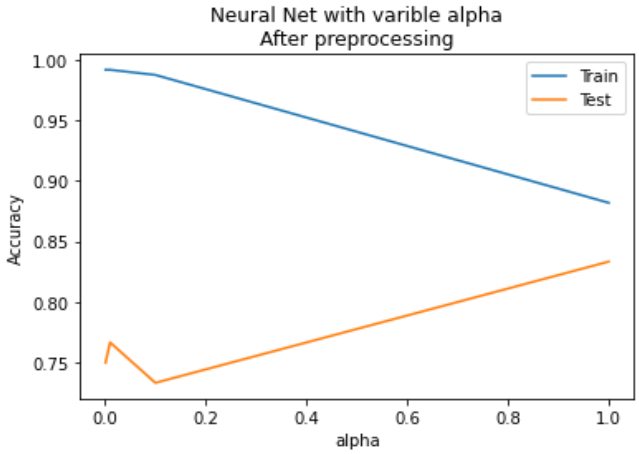


Figure 8: Neural Net Using SGD

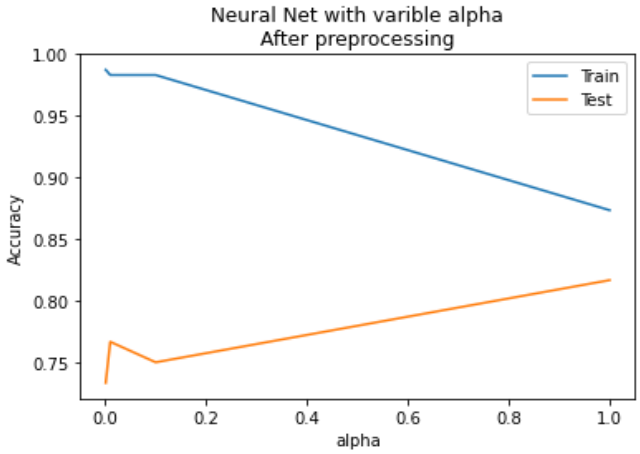
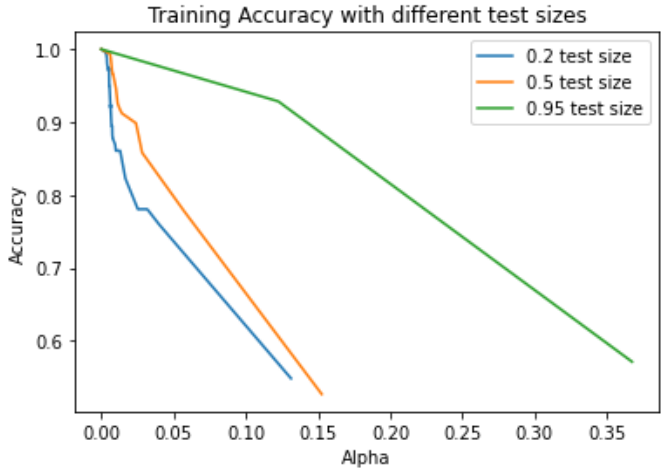


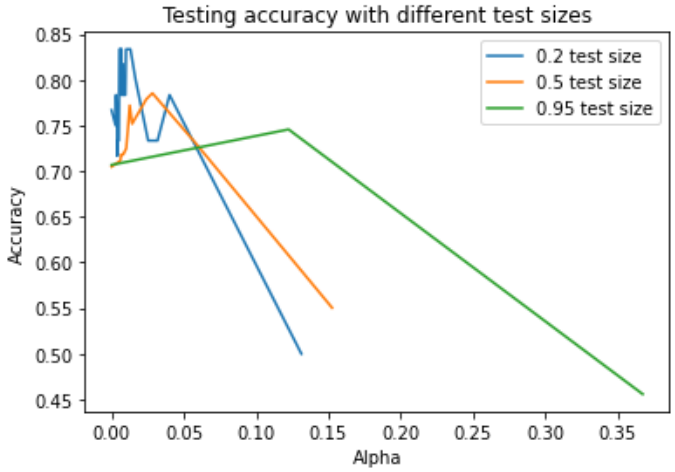
Figure 9: Neural Net Using Adam

1.4 All experiments with different test size

In this section I will present two graphs, one graph will be for the accuracies of the training data with different test sizes and the other one will be for the testing data. This will be done for all the methods. For the experimentation of this method I only used a single split criterion (solver in the case of neural nets), I selected the split criterion/solver based on which one deemed to be more accurate given the analysis performed in the previous sections.

1.4.1 Decision Trees: Testing sizes 20%, 50% and 95% , Split Criterion: Gini Index





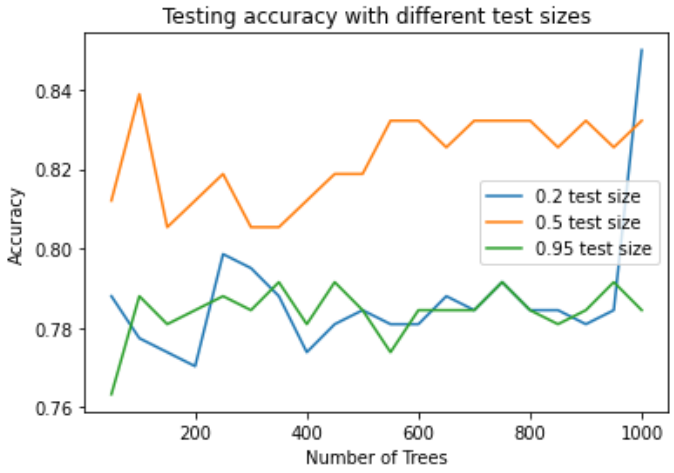
The two graphs above show the trend of the training and testing accuracies with different test sizes. The different test size used in this experiment are 20% of the original dataset, 50% of the original dataset and 95% of the original dataset. We have already seen and analyzed the data when the testing set was 20%. The objective of this experiment is to identify what size of testing data is the most ideal and will return the highest possible accuracy.

The graphs make it very evident that the training accuracy degrades as the value of alpha increase (this is the case for all test sizes), however, looking at the training accuracy closely it is noticed that as the testing size increase the rate at which the training accuracy declines also increases. With the testing accuracy, one trend that is to be noted is that the graphs are more

stable when the test size is larger, and they all have an initial incline after which they begin to degrade. Lastly, as noticed in the training accuracy, the testing accuracy also degrades at a quicker rate when the testing size increases.

Therefore, the most optimal size for testing data is 20% of the original dataset, this is because when the model is given over 80% of the data to train it becomes more efficient and can predict the testing examples with a greater accuracy.

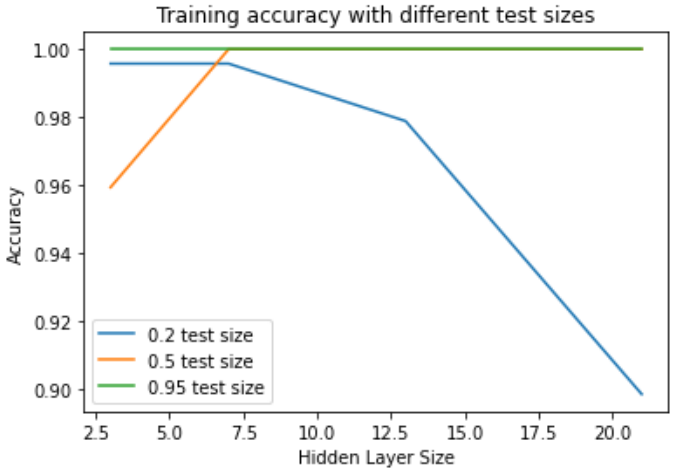
1.4.2 Random Forests: Testing sizes 20%, 50% and 95%, Split Criterion: Entropy

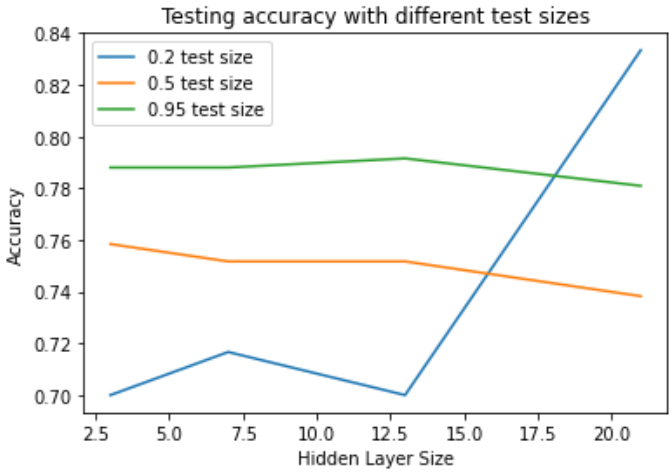


The training accuracies for the different testing sizes were all at a constant value of 1, hence the missing graph for the training accuracies. The graph above shows how the testing accuracy of different test sizes changes when the number of trees in the forest increase. As expected by the blue line (20% testing size) the accuracy fluctuates until it begins to rise for a certain forest size. Surprisingly, the line for 95% testing size is not too far away from the 20% line and the accuracy for this also fluctuates between 0.76 and 0.79. On the other hand, the accuracy for testing size of 50% is the highest of all three despite the fluctuation. This is very unlikely as mentioned earlier, the larger the training size the better the testing accuracy.

Based on the graph shown above it would be safe to say the ideal testing size for random forests would be 50%, however, this is not to be trusted blindly as more experiments could lead to other results.

1.4.3 Neural Network: Testing sizes 20%, 50% and 95%, Solver: Adam





The following graphs show the training and testing accuracy of a neural net using Adam as a solver over a range of testing size in which the number of neurons in the hidden layer increase. The expected trend would be the training accuracy would increase for testing size as the hidden layer size increased. Furthermore, as the testing size increased, we would expect the overall accuracy to decrease. Although, this is not seen by the graphs. The training accuracy for testing size 20% gradually declines as the number of neurons increase but for the other two testing sizes the accuracy increases. A reason for this could be, with a smaller training size the data is not split into equal positive and negative examples which could lead to some false negatives or false positives.

On the other hand, the testing accuracy graph satisfies our expectations. The graphs for size 50% and 95% still accurate, with 95% testing size being more accurate which is unexpected as we would have hoped it be the other way around. However, accuracy for 20% test size is the best after a certain hidden layer size and then it begins to increase.

Based on the graphs above, the most ideal testing size for neural nets using Adam would be 20% of the original dataset, this is because the training accuracy is more accurate and refined which results in a better prediction/classification of testing examples.

Problem 2: Tic-Tac-Toe Endgame Data Set

For this problem the way the experiments were conducted are the same way it was done for problem 1. Therefore, this section will only be used to present the data and preform a short analysis on it.

2.1 Problem Description

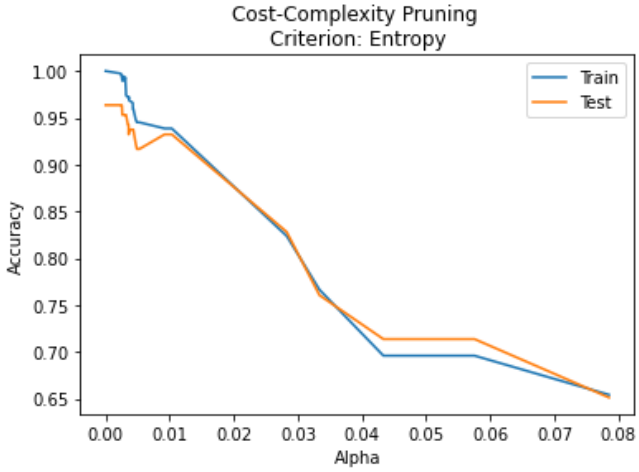
This dataset is pertaining to a binary classification task on possible winning configurations of tic-tac-toe game. It contains a complete set of possible configurations at the end of the game where "x" is assumed to have played first. The goal in this dataset is "win for x" (this is essentially one of eight possible ways to creating a winning configuration for "x").

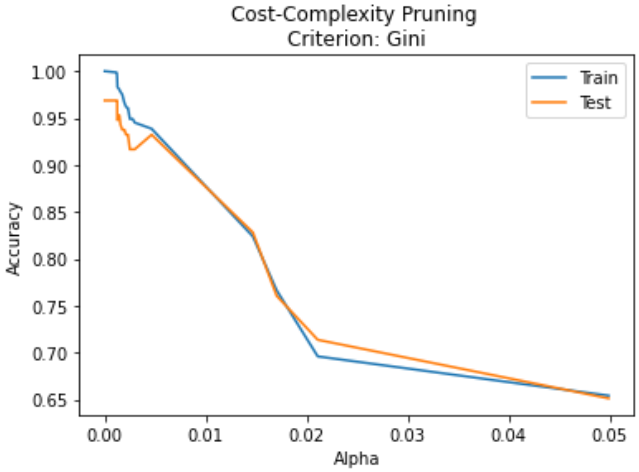
The structure of the dataset is the game board (i.e., it has 9 squares labeled top left, top middle, top right and so on). Before using the dataset for the algorithms, it needed to be cleaned, all the "x's" were converted to a 1, all "o's" to a 0 and all "b's" to a 3 ("b" signifies a blank square). This dataset contains 958 examples with each having nine attributes and one class label (positive meaning win for x, negative meaning win for o).

This problem is interesting because some may believe that tic-tac-toe is a game of chance and the player who starts first always wins. However, some believe that there is skill required to win the game. To test the first claim this dataset can be used and models can be trained to predict whether a given configuration of the board would lead to a winning condition for a certain player. This problem is not trivial as the possibility of winning conditions are quite large and hence using different methods to classify this problem allows us to compare these methods and their performance on the classification.

2.2 Training and Testing of decision trees (Testing size: 20%)

	Decision Tree			
	Before Pruning		**After Pruning**	
Entropy	Training: 1.0	Test: 0.96	Training:	Testing:
			Min: 0.654	**Min:** 0.651
			Max: 1.0	**Max:** 0.964
			Mean: 0.929	**Mean:** 0.906
Gini Index	Training: 1.0	Test: 0.97	Training:	Testing:
			Min: 0.654	**Min:** 0.651
			Max: 1.0	**Max:** 0.969
			Mean: 0.937	**Mean:** 0.912

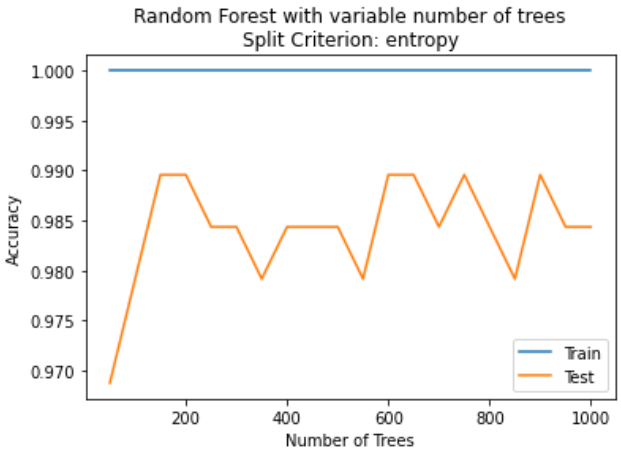


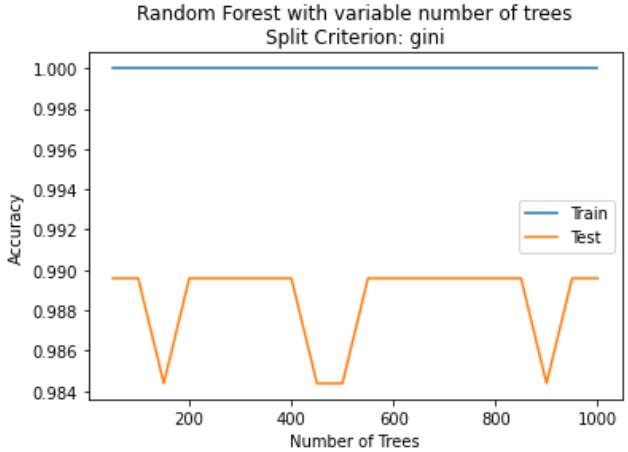


The results for this dataset are essentially the same as the one seen in problem one. As the value of alpha increases the accuracies decrease. From the table and the graphs, it is evident that the Gini Index performs better on the unpruned and pruned decision tree (although the difference between the two is not much). One aspect that is noteworthy is that in this dataset the training and testing accuracies are a lot closer to each other for each split criterion and intersect in multiple places. This allows for a better analysis, and it can be concluded that this dataset is more accurately predicted when using methods employed by sklearn.

2.3 Training and Testing of Random Forests (Testing size: 20%)

Random Forests			
Entropy		**Gini Index**	
Training:	Testing:	Training:	Testing:
Min: 1.0	**Min:** 0.968	**Min:** 1.0	**Min:** 0.984
Max: 1.0	**Max:** 0.989	**Max:** 1.0	**Max:** 0.989
Mean: 1.0	**Mean:** 0.984	**Mean:** 1.0	**Mean:** 0.988

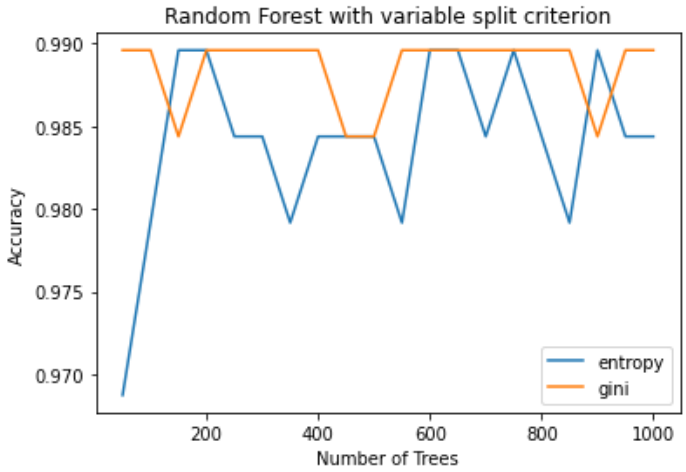




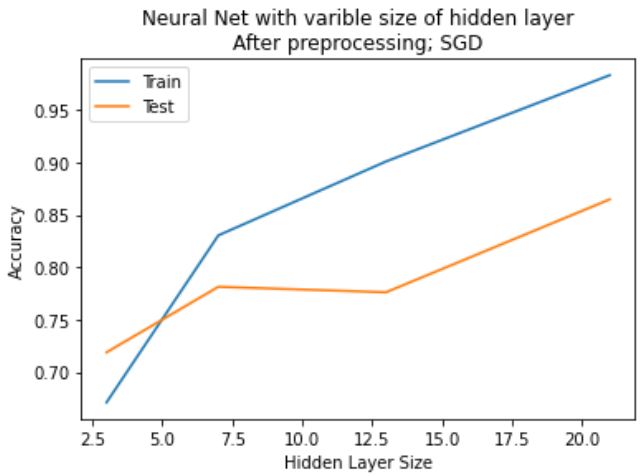
From the table and the graphs on this dataset, the random forest classifier performs almost the same for the two split criterions. The training accuracy is at a constant, but the testing accuracies fluctuate as the number of trees in the forest increase. The results and intuition of these statistics is the same as the ones seen in problem one. However, in this case Gini Index performs slightly better and continues to maintain an accuracy that is higher than that of entropy. This is not seen in problem one where entropy shows this characteristic.

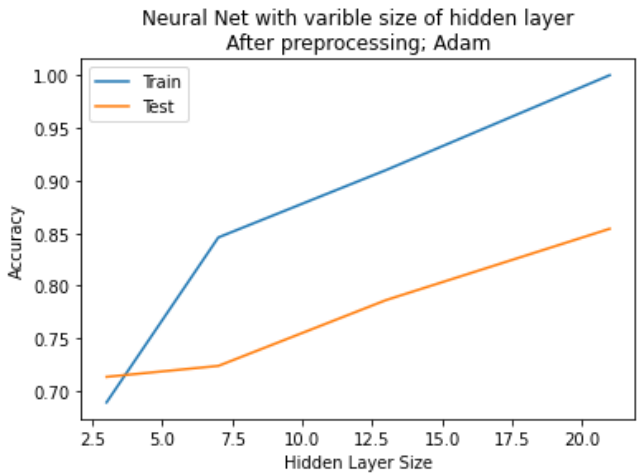
2.4 Training and Testing of Neural Networks (Testing size: 20%)

The table below shows the different values for the training and testing accuracies as and when the size of the hidden layer changes when using SGD and Adam as the solvers. With respect to SGD, we can conclude from that that as the number of neurons increase the training accuracy gets stronger and approaches a value of 1, the testing accuracy also increases, flattens for a certain hidden layer size, and then approaches a value of 1. The same trend can be noticed from the statistics for the solver being Adam, although with using Adam there is no value for which the curve flattens and then begins to climb again. Finally, the two graphs below represent the data from the table to make it easier to visualize.



	Neural Networks: Using SGD		**Neural Networks: Using Adam**	
Number of neurons	Training	Testing	Training	Testing
3	0.671	0.718	0.689	0.713
7	0.830	0.781	0.846	0.724
13	0.900	0.776	0.909	0.786
21	0.983	0.864	1.0	0.854



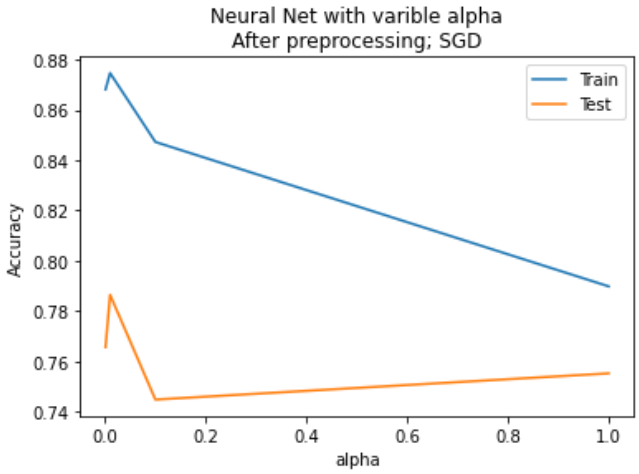


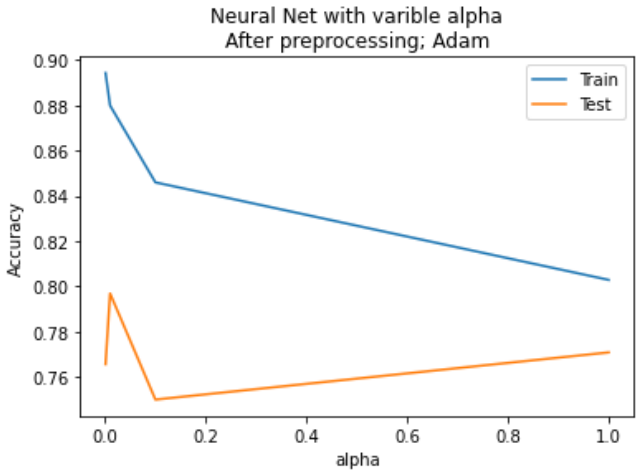
All in all, it can be concluded that the accuracies are higher when using Adam and it can be noticed that the rate at which the accuracy increases is a higher than that of SGD. In conclusion, using Adam as the solver is better when implanting neural nets with a variable size of the hidden layer.

The table below represents data of the two solvers and how the accuracies change when the value of alpha changes, there are graphs accompanying the data from the table. From the table and the graphs, it can be concluded that as the value of alpha gets smaller and smaller the training accuracy increases for both the solvers. For the testing accuracy the values degrade until it hits a certain value of alpha for which the accuracy is at its peak and then it starts to degrade again. This can be better seen through the graphs below.

All in all, it can be concluded that Adam performs better than SGD when changing the regularization parameter for the neural net even though they eventually have the same testing accuracies for alpha = 0.001. The reason why this conclusion can be made is that the accuracy when using Adam is always higher than that of SGD.

	Neural Networks: Using SGD		**Neural Networks: Using Adam**	
Alpha	Training	Testing	Training	Testing
0.001	0.868	0.765	0.894	0.765
0.01	0.874	0.786	0.879	0.796
0.1	0.847	0.744	0.845	0.750
1	0.789	0.755	0.802	0.770





2.5 All experiments with different test size

2.5.1 Decision Trees: Testing sizes 20%, 50% and 95%, Split Criterion: Gini Index

The graphs make it very evident that the training accuracy degrades as the value of alpha increase (this is the case for all test sizes), however, looking at the training accuracy closely it is noticed that as the testing size increases, the rate at which the training accuracy declines also increases. With the testing accuracy, one trend that is to be noted is that the graphs are more stable when the test size is larger. All the graphs have a gradual decline except the graph for 95% test size which has a small incline and then begins to decline.

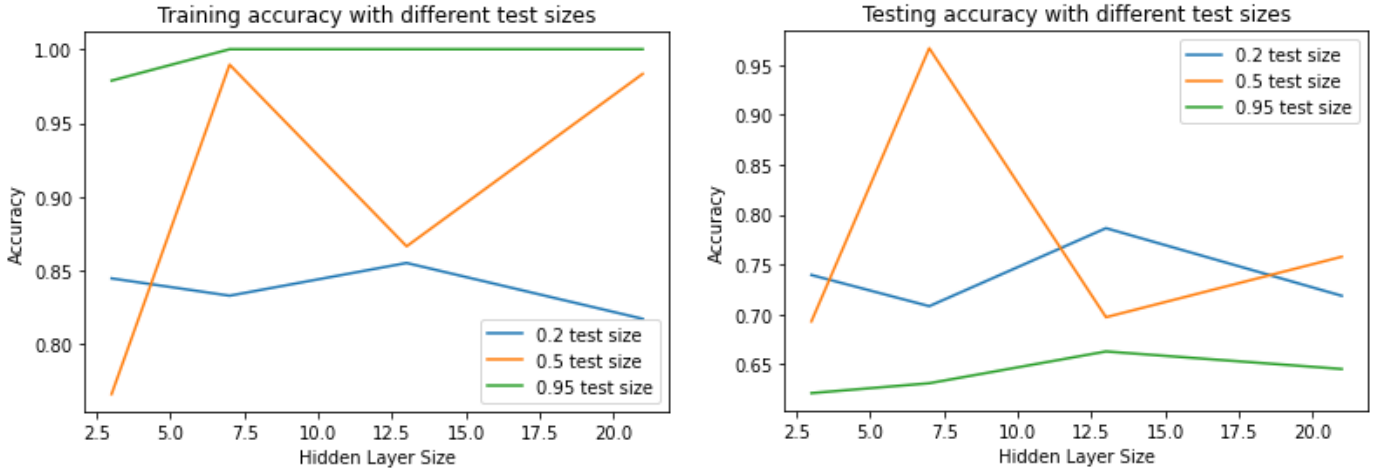
Therefore, the most optimal size for testing data is 20% of the original dataset, this is because when the model is given over 80% of the data to train it becomes more efficient and can predict the testing examples with a greater accuracy. As seen from the graphs the overall training and testing accuracy for this test size is the largest and the rate at which it drops is also slower than the other testing sizes.

2.5.2 Random Forests: Testing sizes 20%, 50% and 95%, Split Criterion: Gini Index

The training accuracies for the different testing sizes were all at a constant value of 1, hence the missing graph for the training accuracies. The graph above shows how the testing accuracy of different test sizes changes when the number of trees in the forest increase. As expected by the blue line (20% testing size) the accuracy fluctuates between 0.97 and 1.0. Furthermore, as expected the line for testing size 50% is not too far away from the 20% line and fluctuates between 0.95 and 0.97. The 95% testing size behaves as expected with a fluctuation and the lowest accuracy of the tree.

Based on the graph shown above it would be safe to say the ideal testing size for random forests would be 20% as it has the highest overall testing accuracy and is able to predict and classify testing examples will as little error as possible.

2.5.3 Neural Networks: Testing sizes 20%, 50% and 95%, Solver: Adam



The following graphs show the training and testing accuracy of a neural net using Adam as a solver over a range of testing size in which the number of neurons in the hidden layer increase. The expected trend would be the training accuracy would increase for each testing

size as the hidden layer size increased. Furthermore, as the testing size increased, we would expect the overall accuracy to decrease. Although, this is not seen by the graphs. The training accuracy for testing size 20% changes and then eventually declines as the number of neurons increase but for the other two testing sizes the accuracy increases overall. A reason for this could be, with a smaller training size the data is not split into equal positive and negative examples which could lead to some false negatives or false positives.

On the other hand, the testing accuracy graph satisfies our expectations. The graph with 50% testing size is more accurate than the one for 95% testing size which is as what we expect with a smaller testing size. The testing accuracy for 20% test size is overall the best even though for some hidden layer size it drops below the 50% testing size (it eventually performs better than 50%).

Based on the graphs above, the most ideal testing size for neural nets using Adam would be 20% of the original dataset, this is because the training accuracy is more accurate and refined which results in a better prediction/classification of testing examples.