Seng474 Assignment 1

Ali Gaineshev
V00979349
October 5 2023

1 Introduction

Accurately identifying spam emails is crucial for keeping inboxes clean and preventing users from falling victim to scams and other malicious activities. In this report, we present an analysis of different machine learning models using email spam detection data. We evaluate the performance of three types of models: Decision Trees, Random Forests, and Boosted Decision Trees. The dataset used for this analysis is called "Spambase", sourced from the UCL Machine Learning Repository [1]. It contains 4601 samples - relatively large dataset with 57 attributes and target variable with 0 as not spam and 1 as spam.

In preparation for our analysis, the data underwent a preprocessing stage, including rescaling of attribute values to ensure they fall within the [0,1] interval. This rescaling procedure was consistently applied to both the training and validation/test sets. Each model's initial dataset was partitioned into an 80% training subset and a 20% validation/test subset.

2 Separate Analysis

Throughout our analysis, we primarily utilize accuracy as our key performance metric for testing both the training and validation/test sets. This metric was chosen due to its easier interpretability on a plot. However, for k-fold cross validation on models we used test error as a performance metric. To explore the effects of various hyperparameters, we conducted experiments aimed at identifying optimal models and then conducted additional measurements with fixed hyperparameters. Every single model was taken from scikit learn using Python.

2.1 Decision Trees

Decision tree is a machine learning model that works by predicting the target variable's value after training on some dataset [2]. However, there is a risk of overfitting, where the model becomes too complex and captures noise and irrelevant details from the training data, resulting in poor performance on unseen data. In our analysis, we trained decision trees using 80% (3680 samples) of the dataset. This extensive training can lead to overly complex trees that rely on specific details in the data to make predictions. To address that issue, we applied cost complexity pruning on multiple trees to find an optimal one. This technique introduces a hyperparameter called alpha (α) which controls the trade-off between the complexity of the tree and its accuracy [3]. Gini and Entropy split criterions were both used to compare the performance.

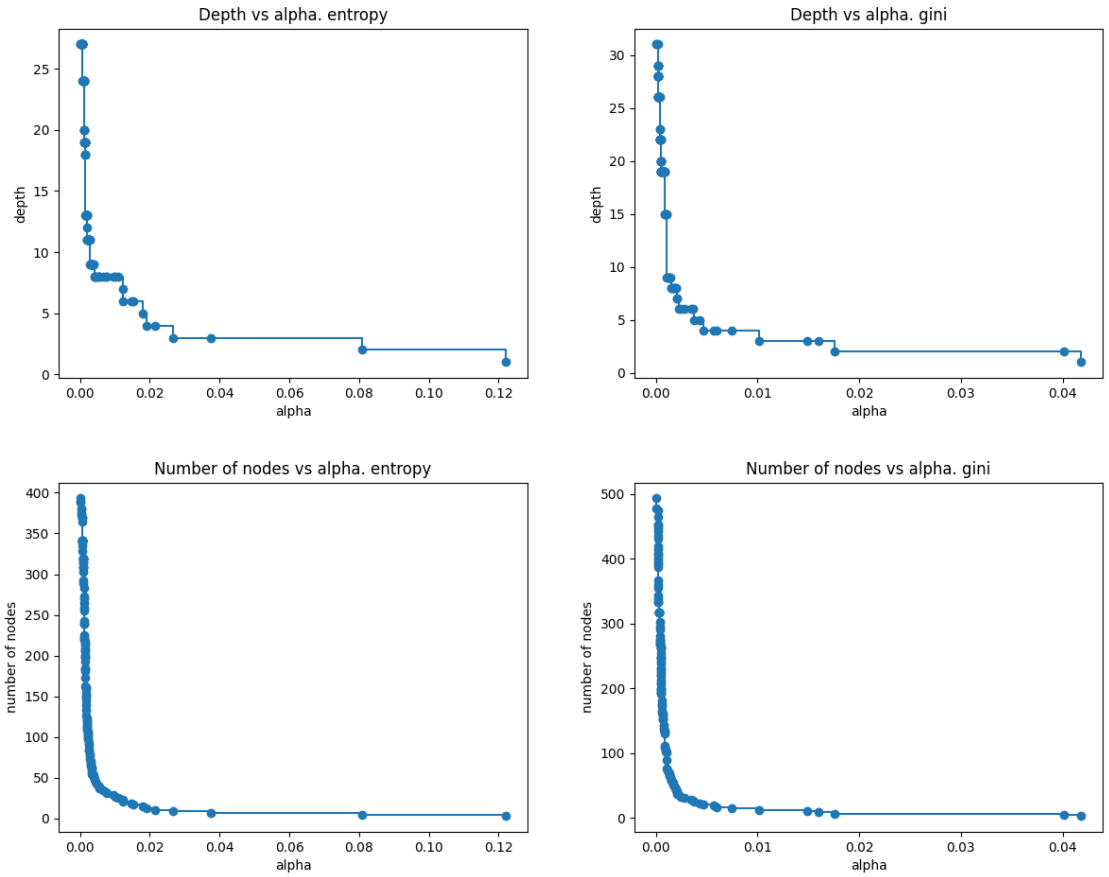


***Figure1**. Tree properties as alpha increases with gini and entropy.*

By using multiple alpha parameters, we were able to make the decision tree simpler and with less depth and node count as alpha increased (see **Figure 1**). The pruning helps the model avoid overfitting, ensuring that it makes predictions based on meaningful patterns in the data rather than outliers. As a result, the pruned decision tree becomes simpler and less complex (see **Figures 1,2)**.

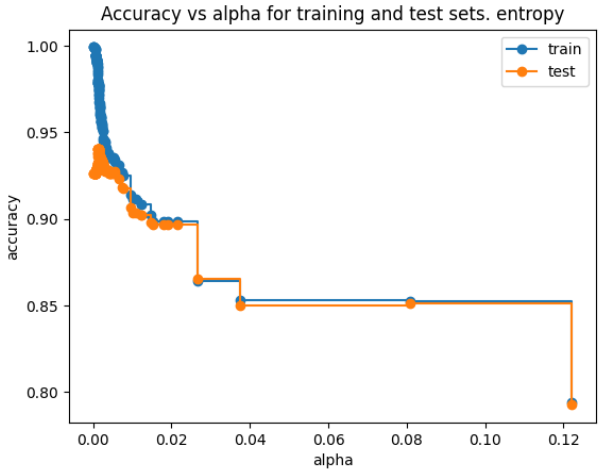


***Figure 2**. Accuracy with entropy for splitting*

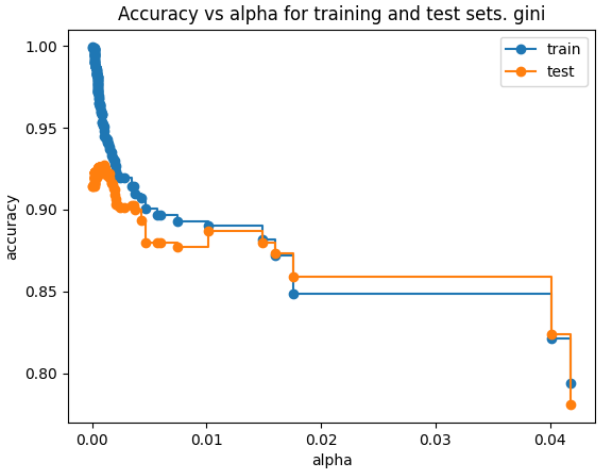


***Figure 3.** Accuracy with gini for splitting*

In figure 2 and 3 we can observe the accuracy as alpha increased and more branches were pruned. As was expected the accuracy was decreased for the validation set, since the tree becomes more generalized as alpha increased, so less attributes were considered. However, we were surprised by accuracy with gini as a split criterion around $\alpha = 0.02$, since the accuracy for the test set was higher than the training set. Decision trees with entropy had the same property around $\alpha = 0.08$ but the difference was less significant. In general we can see that as $\alpha = 0$, without pruning, the difference in accuracy for training and test sets are significant, meaning that the tree is not optimal for unseen data. Therefore, It is important to select optimal alpha, so in our case we decided to select 2 optimal alphas based on these 2 factors:

- Optimal tree 1 : lowest difference between training and test accuracies
- Optimal tree 2 : highest test scores

The same method is applied to random forests and boosted decision trees for searching an optimal model. Selecting the alpha value with the lowest difference between training and test scores is beneficial because it can show a balance between model complexity and generalization, ensuring that the tree is both well-fitted to the training data and capable of performing well on unseen data. Optimal tree 2 seemed like the most intuitive tree to choose since it had the highest accuracy for test data. However, in reality, the tree that was obtained was way too complex. For both split criterions, optimal trees 2 gave us trees with high depth and node count. The alpha value was also low (very close to 0) for these trees meaning that the trees might have been overfit.

	Optimal tree 1	**Optimal tree 2**
Description	Lowest difference between training and test sets scores	Highest score for test set
alpha	0.0265	0.0012
Depth	3	20
Node count	9	225
Accuracy - training set	0.8644	0.9788
Accuracy - test set	0.8654	0.9403

Table 1. *Comparison of optimal trees with entropy as split criterion*

All of the optimal trees with both split criterion can be viewed at appendices A,B,C,D. Optimal trees 2 can be clearly seen as overgrown, therefore making optimal tree 1 the best option. The last test is training a new classifier with only some percent of the training data; the validation/test data was never changed. At the end, measures of accuracy for testing original training and testing sets are shown. This method will be applied to the rest of the models.

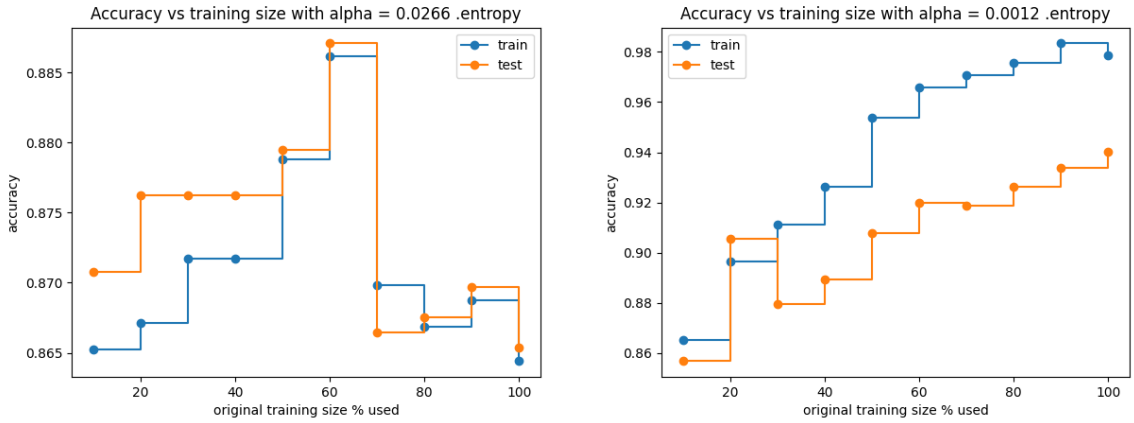


Figure 4. *Optimal tree 1 vs 2 with different training sizes. Entropy*

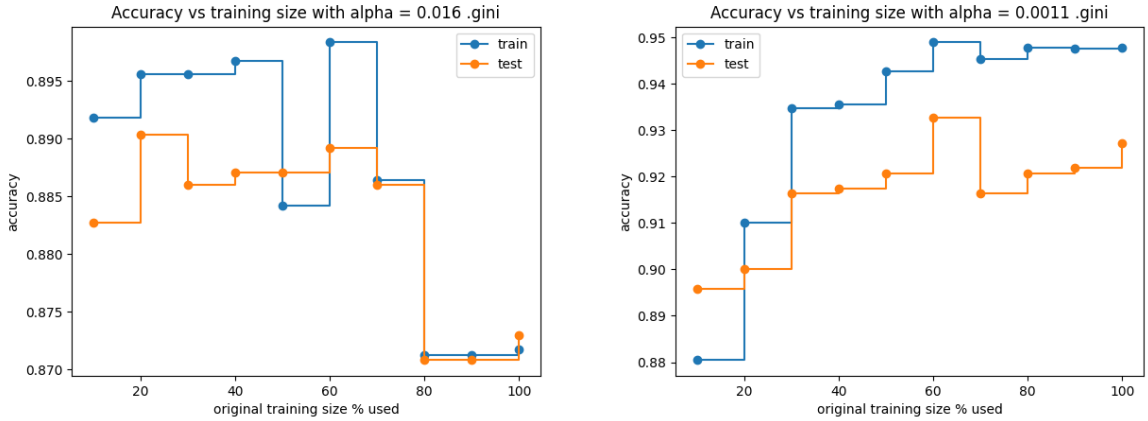


Figure 5. *Optimal tree 1 vs 2 with different training sizes. Gini*

Figures 4 and 5 are hard to interpret, because the accuracy varies a lot from these trees. Especially, the optimal tree 1 with split criterion as entropy achieved better results with both original training and test sets, while it was trained only on 60% (2208 samples) of the original training set (see **Figure 4**). We can also observe that for optimal trees 2, the difference between training and test sets accuracies are significant, while optimal tree 1 has both accuracies very close to each in most of the cases.

At the end, optimal tree with entropy split misclassified 499 samples, while tree with gini split did only 472 (see appendices A, C). The difference in accuracies for optimal tree 1 with gini and entropy split was only around 1% for both sets. No other significant differences were found.

2.2 Random Forests

Random Forest is a machine learning model that consists of multiple decision trees, each trained on a slightly different subset of the same dataset, ensuring low correlation among the trees [4]. These subsets are generated by randomly sampling from the training set with replacement, which may result in some data points being repeated or omitted in different trees. In this analysis, we explore the behavior of Random Forests without pruning on a dataset and investigate the impact of various hyperparameters, such as the number of trees (forest size) and the number of random features, on the model's performance. Initially, we considered both Gini and entropy split criteria to assess the accuracy of the Random Forests with varying forest sizes. However, for later experiments, after obtaining optimal forest size, we focused only on the gini criterion.

Initially, we tested random forests with different sizes of trees and $\sqrt{d}$ max features where d is the number of attributes (57, target variable is excluded). The idea is to get the optimal tree with fixed forest size hyperparameter to test max features hyperparameters. The max depth of a single tree was omitted, since it adds too many variables to control. To find an optimal tree, we used the same method as discussed in "Decision Trees". However, upon testing optimal tree 1 and 2 for both split criterias, we were given exactly the same trees for the same split criteria. Only gini split would give us different forest sizes for both optimal forests; therefore, for later experiments, we chose gini as the only criterion.

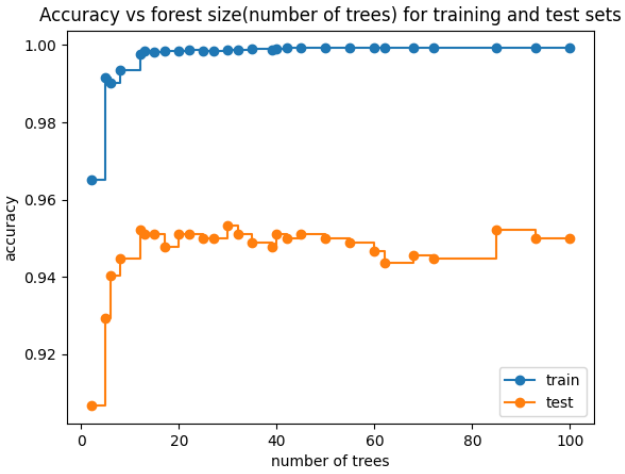


Figure 6. *Accuracy with forest size. Entropy*

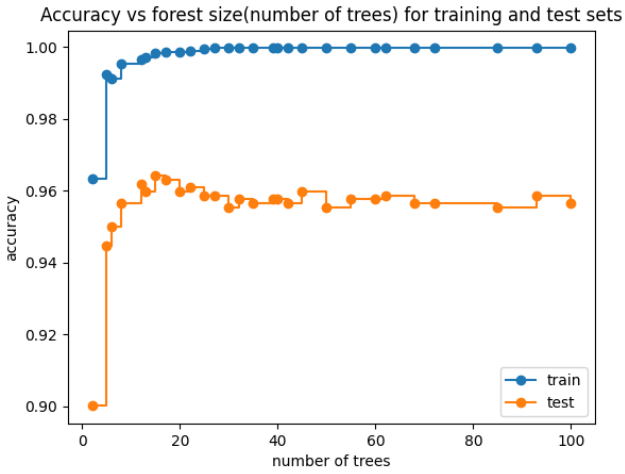


Figure 7. *Accuracy with forest size. Gini*

Retesting the training set gave us an almost perfect score of 1 for both of split criterions. However, the accuracy for validation/test set was a bit lower with entropy achieving better accuracy in general. The difference between optimal trees for both criterions was about 1%. We can clearly see in figures 6 and76 that low forest size (number of trees) doesn't give us good results, but it increases and stays relatively the same after using more than 10 trees. Optimal forest 1 with gini split gave us a forest size of 12 with test set accuracy of 0.9522. While optimal forest 2 with gini split

gave us a forest size of 30 with test set accuracy of 0.9533. The next experiment involved having fixed forest size hyperparameters but different max features values.

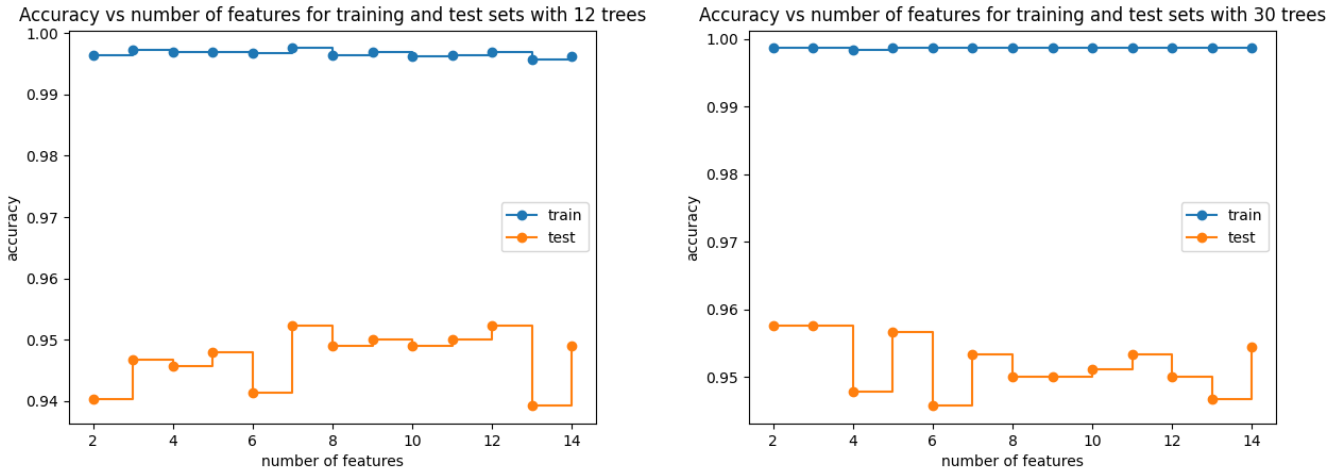


***Figure 8**. Accuracies with different numbers of max features. Gini*

For optimal forest 1 with 12 trees we can see that we achieve the best performance with original max feature value (around 7, $\sqrt{57} \approx 7$). For the optimal forest 2 with 30 trees we achieved better performance with less features (2 and 3). We suspect that with a greater number of trees (optimal forest 2), each tree contributes less to the final prediction i.e trees are less complex, but since the forest is large and diverse, the model will make more accurate predictions. On the other hand, large forest size (optimal forest 2) with more features makes trees more complex and may rely on specific information. In both cases, accuracy for the training set was almost 1 again.

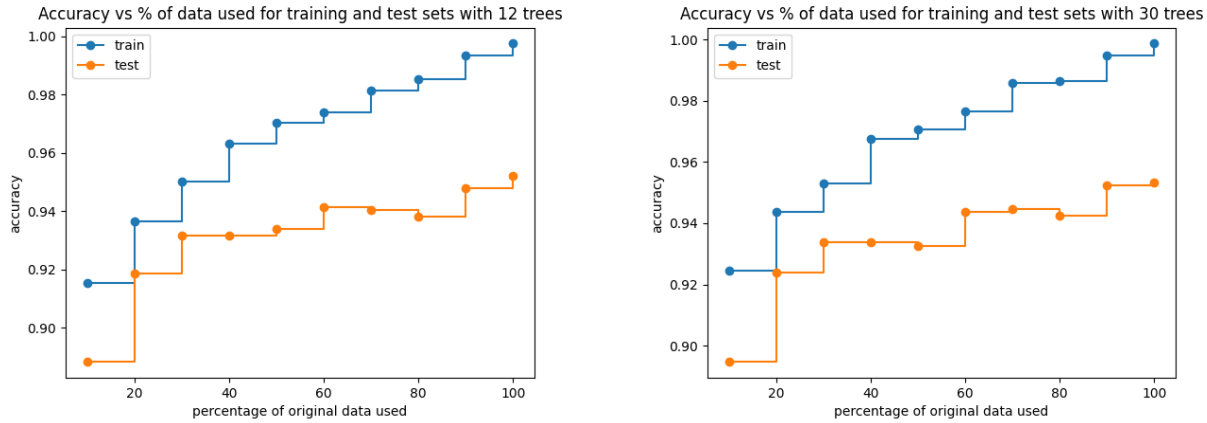


***Figure 9.** Accuracy with different training sizes. Gini*

The last test that was made is cutting % of the training data. There are no observations that can be made, besides that the accuracy increases as more training data becomes available.

2.3 Boosted Decision Trees

Boosted Decision Trees is a learning method that relies on combining the strength of multiple weak learners, in our case decision trees. This method aims to improve predictive accuracy by iteratively training a sequence of weak learners, with each learner focusing on the mistakes made by the previous ones, in order to build a stronger learner. It does that until the training data is predicted perfectly or the maximum number of learning have been used [5].

The 2 main hyperparameters that can be used are max depth of weak learners and number of weak learners/iterations. From experimenting with decision trees to find an optimal one, the depths of those trees usually varied from 1 to 5. Therefore, we chose max depths for weak learners fixed as 3 and 5. Max depth of 3 trees can give us trees that are not too complex and the same can be said for 5 but in the other directions. This was also done for simplicity. The other hyperparameter that was changed and experimented with was the number of iterations which was set to numbers from 2 to 80. Additionally, only gini split criterion was used for the classifier.

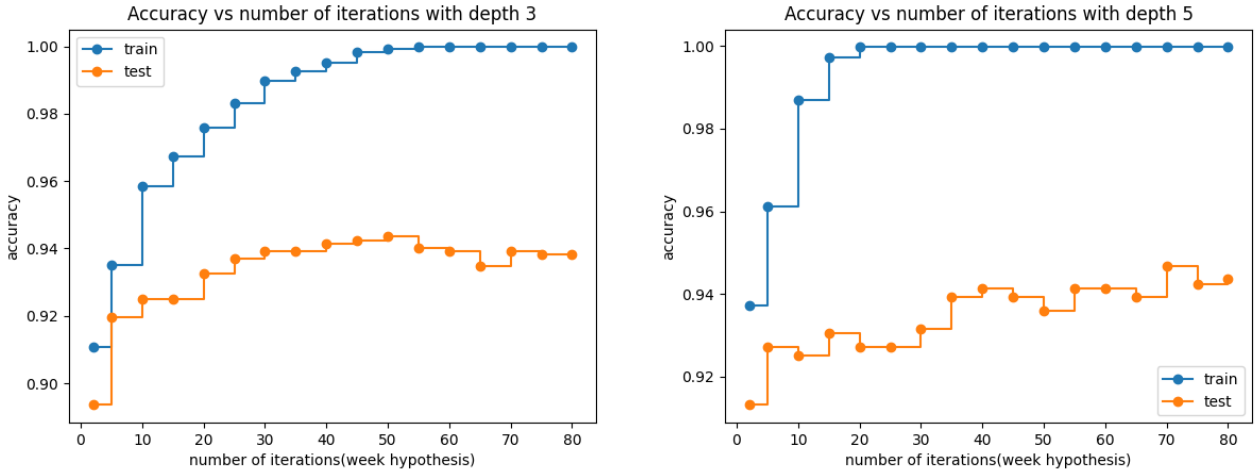


Figure 10. *Accuracy vs number of iterations with different depths. Gini*

From figure 10 we can see that the accuracy for a training set is around the same for both depths of weak learners. Although weak learners with depths 3 seem to have higher accuracy in general, the highest accuracy from a validation/test set was given by weak learner with depth 5 and number of iterations as 750 but with only less than 0.05% difference. Additionally, trees with depths 3 struggled with the training set until it reached around 50 number of iterations before achieving a perfect 1.0 score. We suspect that trees with depth 3 struggled to find meaningful patterns, because the trees were simple.

We used the same method as before to select an optimal model and we got these results:

	Optimal model 1	Optimal model 1	Optimal model 2	Optimal model 2
Description	Lowest difference between training and test set accuracies		Highest test set scores	
Depth	3	5	3	5
Number of iterations	5	2	20	70
Training score	0.9350	0.9372	0.9992	0.9997
Test score	0.9196	0.9131	0.9435	0.9468
Difference in scores	0.0154	0.0241	0.557	0.0529

***Table 2**. Analysis of optimal models with adaboost. Gini*

To conduct the last test with feeding only some % of the training data to the classifier we had to choose an optimal model. The decision was made for model 2, since it has highest test scores and in general we observed that as the number of iterations increases the accuracy increases as well. Optimal model 1 had a low number of iterations/weak learners.

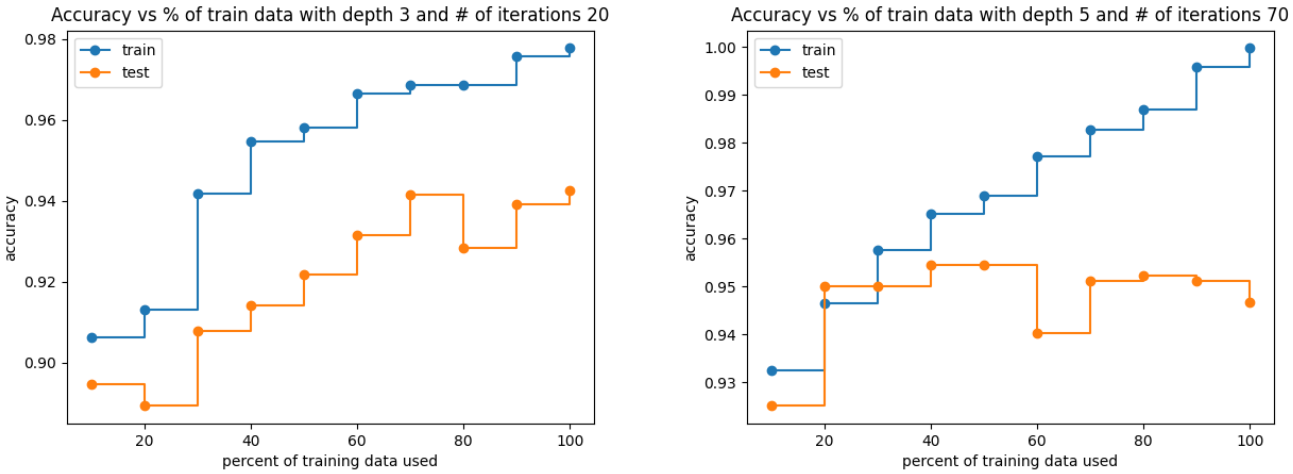


***Figure 11**. Accuracy vs cutting training set for adaboost. Gini*

From figure 11, we can see that accuracy for the training set increased as we used more samples from that set. However, we can't say the same thing for weak learners with depth 5 for the test set, since it stayed the same or decreased. Though, it needs to be said that weak learners with depths of 5 achieved better performance in general for the test set; we suspect the reason for that is higher number of iterations.

3 Comparative Analysis

K-fold cross validation is a technique to assess the performance of a model based on partitioning data into k subsets [6]. For k iterations, k-1 folds are used for training, while the rest is used as a validation set. The performance is based on average cross validation error. For this report, we used K-fold cross-validation to evaluate and tune the ensemble size for the best performance of Random Forests and Boosted Decision Trees. Figure 12 shows how we split the data, trained on each fold and used test data as our final verification. The list of ensemble sizes for both models were the same for both models.

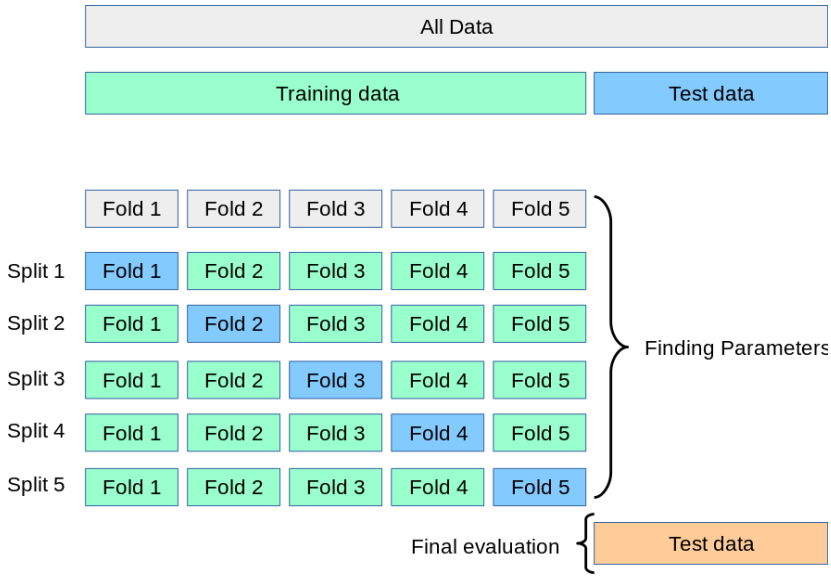


***Figure 12**. K fold cross validation [7]*

By varying ensemble sizes, we aimed to identify its optimal hyperparameter for each model. Finally, after selecting the optimal ensemble size, we obtained test error from the original, untouched test set. We performed 2 tests with k being 5 and 10 i.e 5 and 10 folds for tuning hyperparameters.

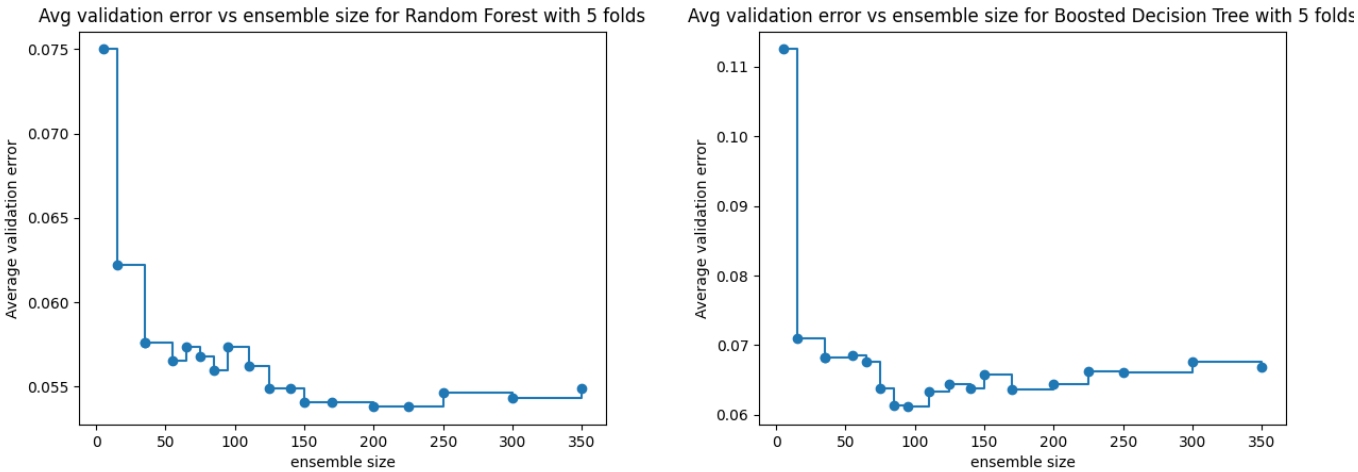


***Figure 13.** Average validation error for ensemble size with 5 folds.*

Both models show the same pattern of decrease in validation error as ensemble size increased. However, boosted decision trees preferred a lower value of ensemble size (around 90) and random forest preferred higher value(around 200).

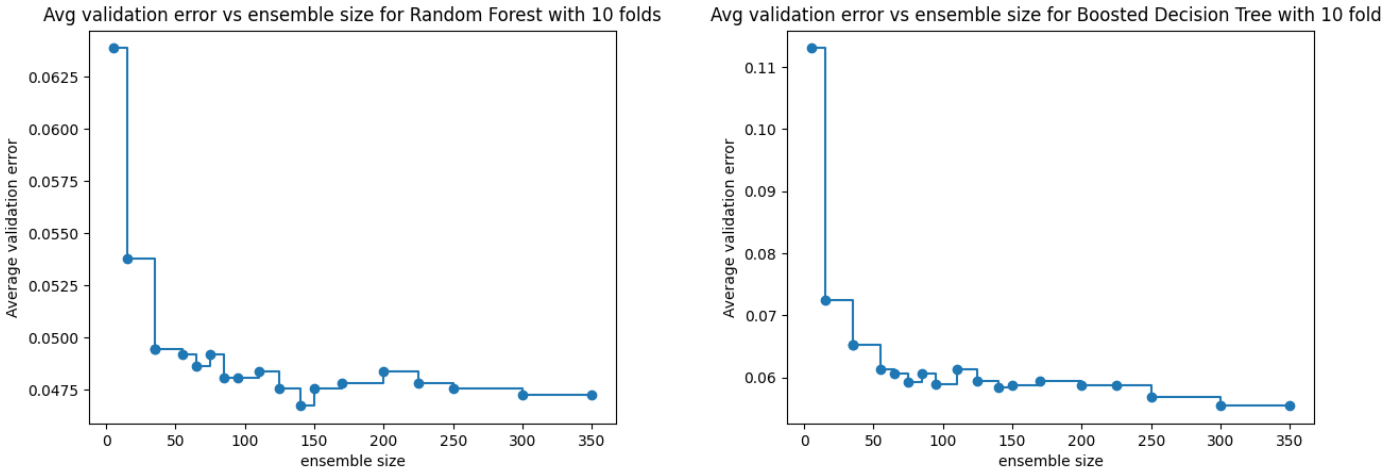


Figure 14. *Average validation error for ensemble size with 10 folds*

With k = 10, we got completely different results with a boosted decision tree preferring a higher value (300), which was the maximum ensemble size we had. Random forest preferred ensemble size of 140.

The main difference between having k as 10 and 5 is that the validation set (as k = 10) is smaller and the training data is bigger. This means that on average 10 fold cross validation will achieve better performance (lower average validation error), simply because it has less data to test on a fold. In table 3 all of the relevant data shown for models with lowest average validation error.

	k = 5	**k = 5**	**k = 10**	**k = 10**
Model	Random Forest	Boosted Decision Tree	Random Forest	Boosted Decision Tree
Ensemble size	200	95	140	300
Average validation error	0.0538	0.0611	0.0467	0.0553
Test error	0.0336	0.0510	0.0434	0.0554
Result	**Random Forest with lowest test error with difference = 0.0174**		**Random Forest with lowest test error with difference = 0.0245**	

Table 3. *Final Results with k = 5 and k = 10*

From Table 3, we can see that with less folds (k = 5) we achieved better results of test error. In both cases of folds, Random Forest had lower error with difference of about 0.02 and 0.01. In short, Random Forest outperformed Boosted Decision Tree by a small margin.

4 Conclusion

This report presented an analysis of machine learning models for email spam detection using the "Spambase" dataset. Decision Trees, Random Forests, and Boosted Decision Trees were evaluated, with careful consideration of hyperparameters and cross-validation techniques.

In the case of Decision Trees, we demonstrated the importance of cost complexity pruning using alpha values to balance model complexity and simplicity. We identified two optimal trees, one with the lowest difference between training and test set scores and another with the highest test accuracy. This method was used for the rest of the models. Pruning was crucial in preventing overfitting.

For Random Forests, we explored various hyperparameters, including ensemble size and the number of random features. We found that a larger ensemble size improved model performance but eventually stopped growing; Additionally, max feature values varied for different forest sizes.

Boosted Decision Trees, using AdaBoost with decision trees as weak learners with 2 fixed values of depth, were assessed with varying number of iterations. We observed that different depths of weak learners exhibited diverse learning behavior, with depth 5 and higher iterations yielding the best results.

Additionally, k-fold cross-validation was used to tune ensemble sizes for both Random Forests and Boosted Decision Trees. We compared the results obtained with different values of k (5 and 10) and found that smaller folds (k = 5) produced better test error results for both models. Random Forests consistently outperformed Boosted Decision Trees, with lower test error for both k’s.

5 References

[1] “Spambase,” UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/94/spambase (accessed Oct. 1, 2023).

[2] “1.10. decision trees,” scikit, https://scikit-learn.org/stable/modules/tree.html (accessed Oct. 3, 2023).

[3] Lesson 8 - Tree-Based Methods · RN Financial Research Centre, http://www.rnfc.org/courses/isl/Lesson%208/Summary/ (accessed Oct. 2, 2023).

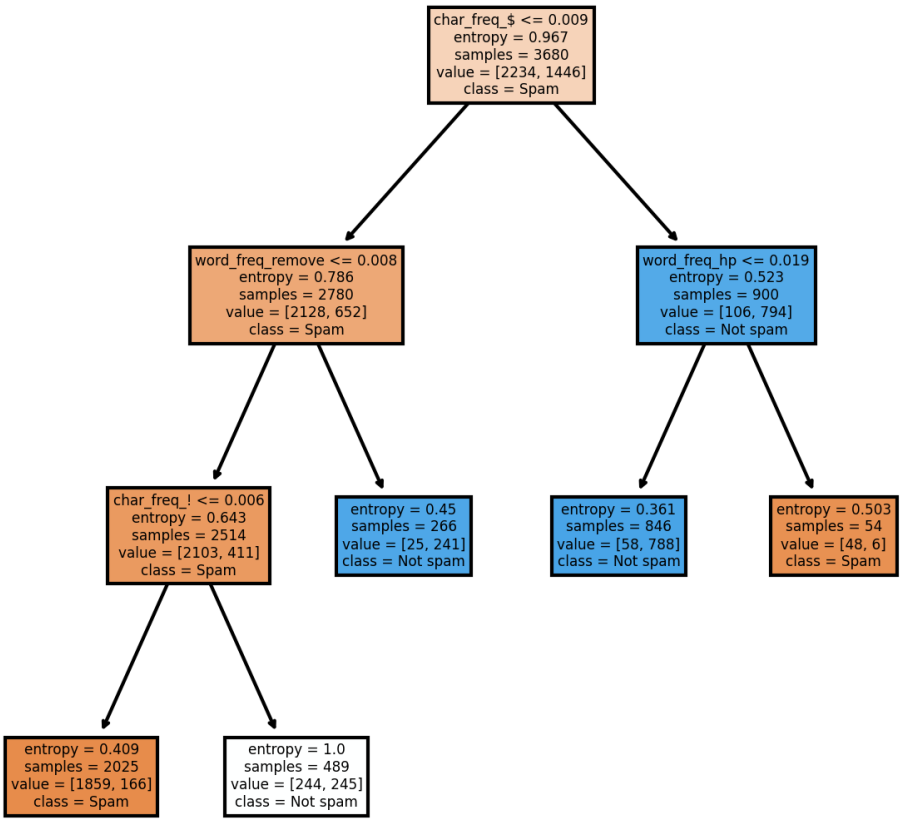
[4] “What is Random Forest?,” IBM, https://www.ibm.com/topics/random-forest (accessed Oct. 2, 2023).

[5] V. Kurama, “A guide to understanding AdaBoost,” Paperspace Blog, https://blog.paperspace.com/adaboost-optimizer/ (accessed Oct. 3, 2023).

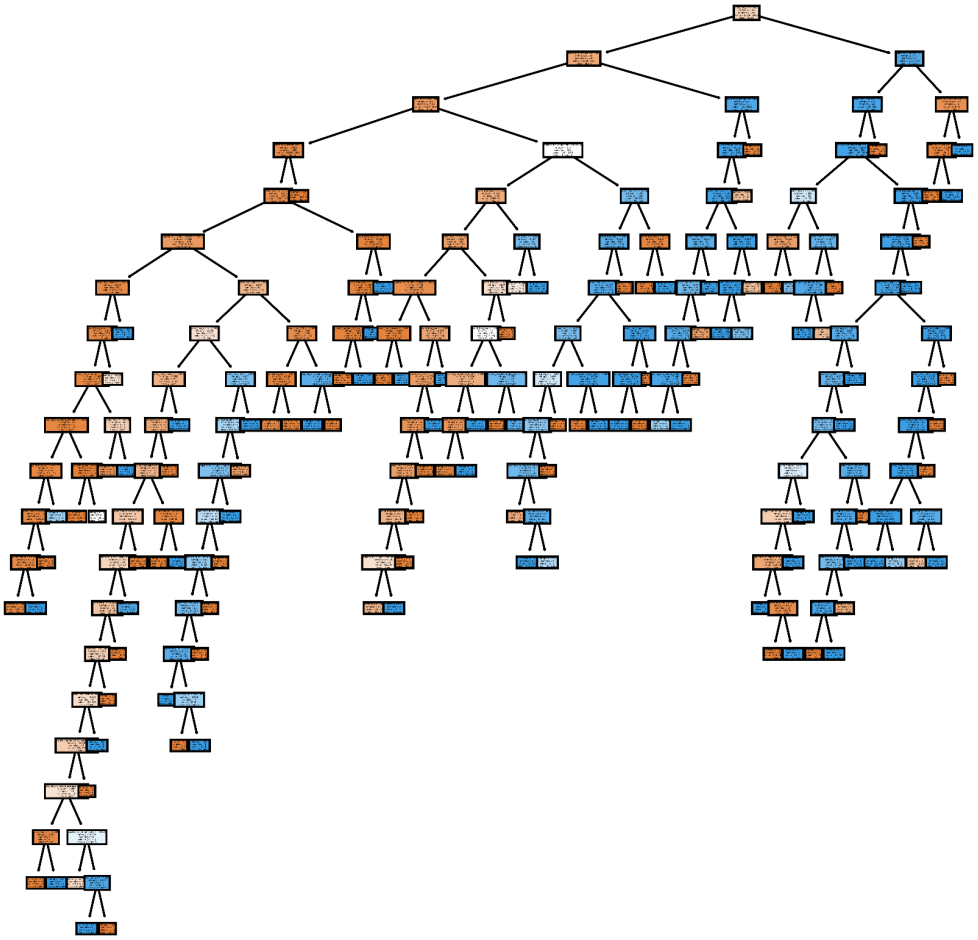
[6] P. bhardwaj, “Cross-validation and hyperparameter tuning,” Medium, https://medium.com/almabetter/cross-validation-and-hyperparameter-tuning-91626c757428 (accessed Oct. 4, 2023).

[7] “3.1.Cross-validation: Evaluating estimator performance,” scikit, https://scikit-learn.org/stable/modules/cross_validation.html (accessed Oct. 5, 2023).

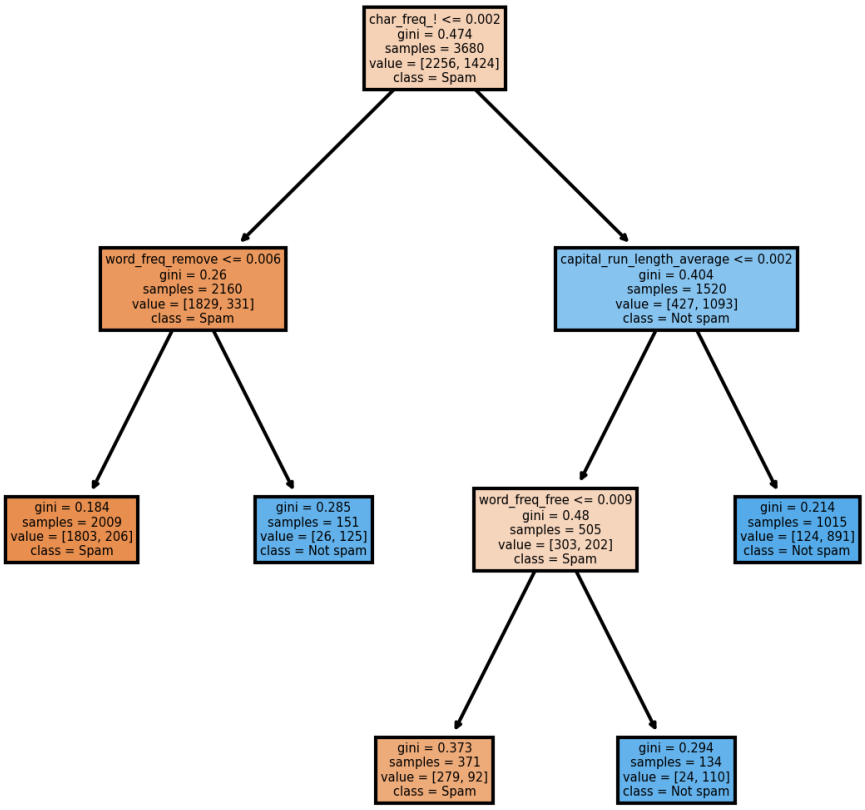
Appendix A - Optimal tree 1 Entropy split



Appendix B - Optimal tree 2 Entropy split



Appendix B - Optimal tree 1 Gini split



Appendix D - Optimal tree 2 Gini split