Assignment 1: Experiments and Analysis

Isaak Wiebe, V01004501

February 2, 2025

Contents

1 Introduction

A binary classification task was performed on the Spambase dataset to distinguish between spam and non-spam emails using various statistical learning models. The models were implemented using SciKit-Learn and NumPy, and visualizations were generated using Matplotlib. The goal of this analysis was to evaluate the performance of different machine learning techniques, including decision trees, random forests, and boosted decision stumps, in classifying emails. The dataset, sourced from the UCI Machine Learning Repository, contains features derived from email content, such as word frequencies and special characters, making it suitable for binary classification tasks.

—

2 Part I: Separate Analysis

This section details the individual performance of each model and the approaches taken to optimize their performance on the dataset. For each model, multiple configurations and pruning techniques were explored to identify the best fit for the data.

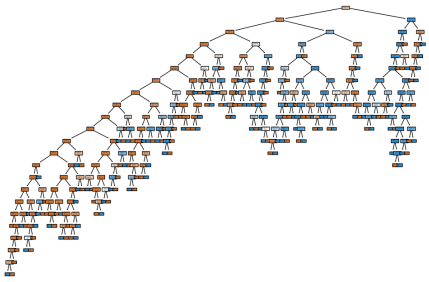
2.1 Decision Trees

Decision trees were implemented using four different pruning techniques to evaluate their impact on model performance and complexity:

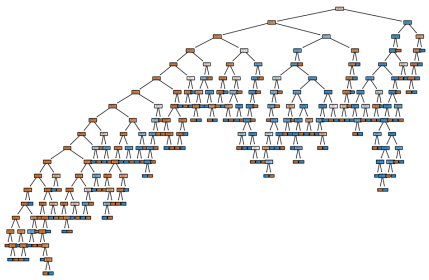
- **No pruning**: A fully grown tree without any constraints, which often leads to overfitting.
- **Minimum Cost-Complexity Pruning**: A method that balances tree complexity and accuracy by introducing a penalty for additional nodes.
- **Reduced Error Pruning (Gini Index)**: Pruning based on the Gini impurity measure, which minimizes misclassification.
- **Reduced Error Pruning (Just Error)**: Pruning based solely on classification error.

Figure 1 illustrates the resulting decision trees for each pruning method. The pruned trees are significantly smaller and less complex than the unpruned tree, demonstrating

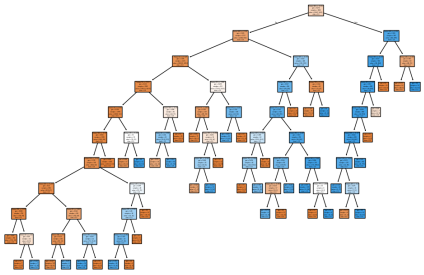
the effectiveness of pruning in reducing overfitting.



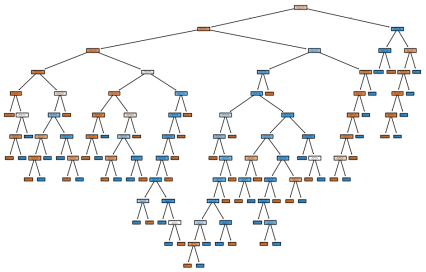
(a) Raw Tree



(b) Cost-Complexity Pruning Tree



(c) Reduced Error Tree



(d) Gini Index Tree

Figure 1: Comparison of Decision Trees with Different Pruning Methods

To determine the optimal pruning parameters, the relationship between the pruning parameter (alpha) and model accuracy was analyzed. Figures 2, 3, and 4 show these relationships for each pruning method.

2.1.1 Minimum Cost-Complexity Pruning

Figure 2 shows the trade-off between alpha and accuracy for minimum cost-complexity pruning. As alpha increases, the tree becomes simpler, but accuracy may decrease if pruning is too aggressive.

2.1.2 Reduced Error Pruning (Gini Index)

Figure 3 illustrates the relationship between alpha and accuracy for reduced error pruning using the Gini index. The Gini index measures the likelihood of misclassification, and pruning based on this metric helps balance accuracy and complexity.

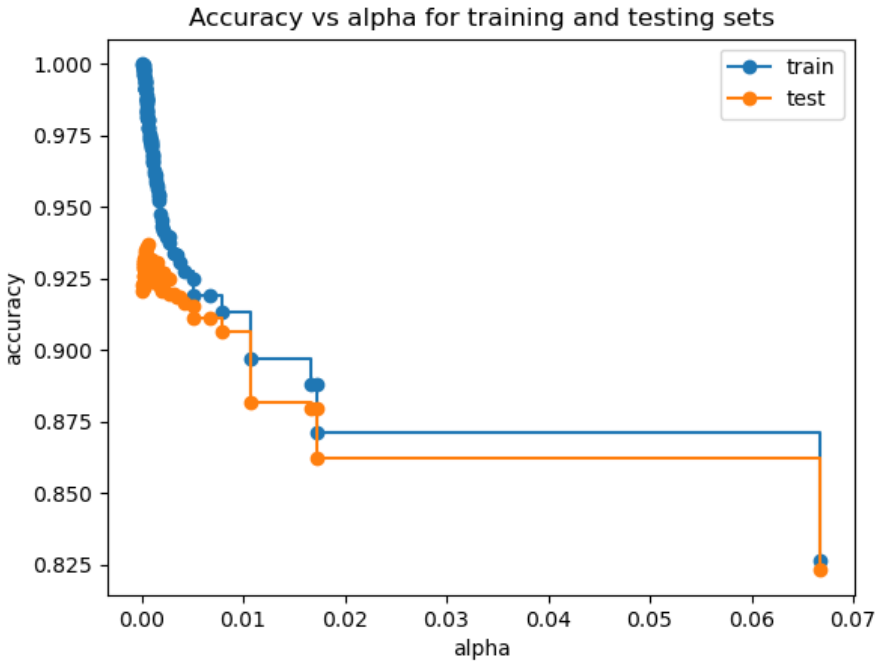


Figure 2: Alpha vs. Accuracy for Minimum Cost-Complexity Pruning

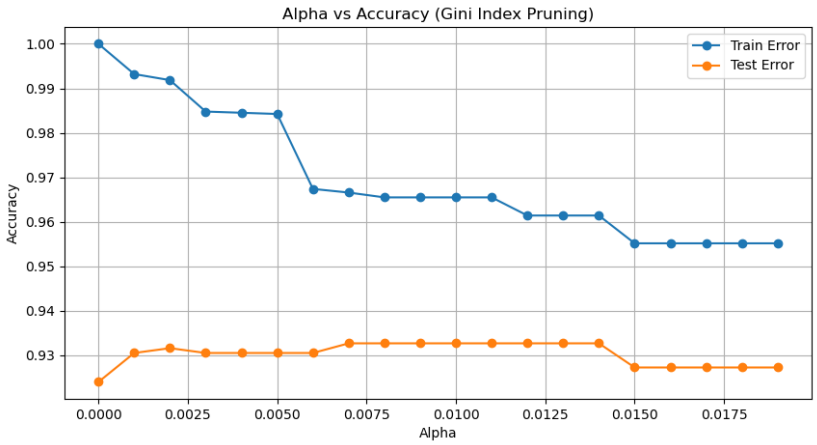


Figure 3: Alpha vs. Accuracy for Reduced Error Pruning (Gini Index)

2.1.3 Reduced Error Pruning (Just Error)

Figure 4 shows the relationship between alpha and accuracy for reduced error pruning based solely on classification error. This method is simpler but may not always yield the best balance between accuracy and complexity.

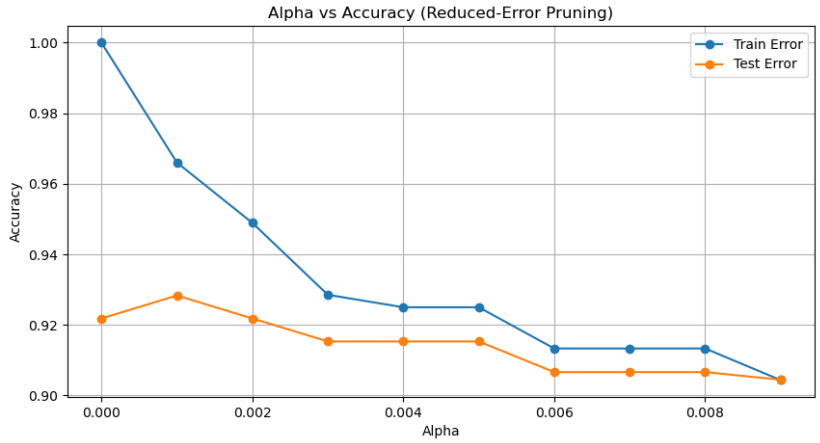


Figure 4: Alpha vs. Accuracy for Reduced Error Pruning (Just Error)

2.2 Random Forests

Random forests were analyzed by varying ensemble size (number of trees) and maximum tree depth. The importance of each feature in the dataset was also evaluated, as shown in Figure 5. Figures 6 and 7 display the max depth, number of trees, and the corresponding error rates, respectively.

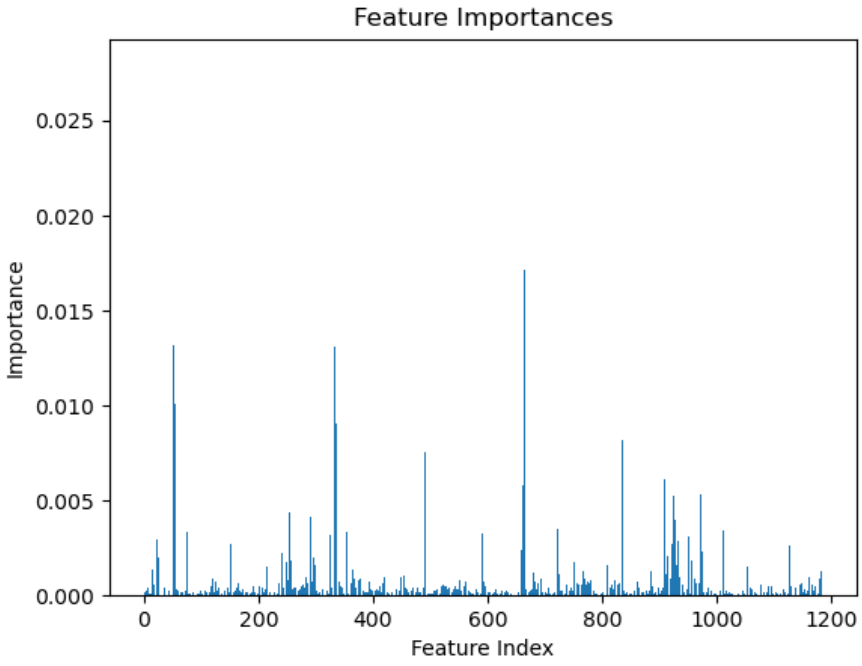


Figure 5: Importance of Each Feature

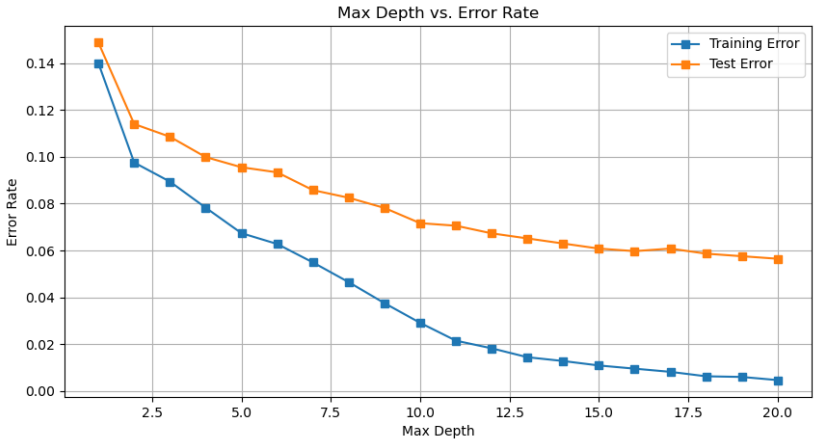


Figure 6: Max depth vs training and test error rate

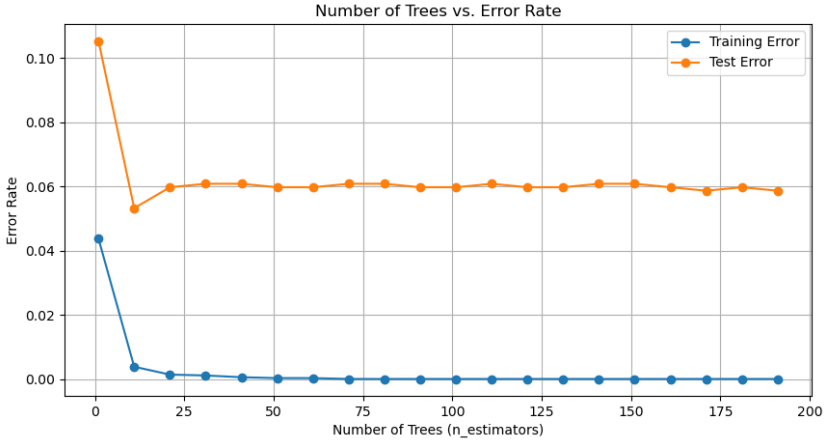


Figure 7: Number of trees vs training and test error rate

2.3 Boosted Decision Trees

AdaBoost, a boosting algorithm, was used to analyze boosted decision stumps. The ensemble size and stump weights were varied to evaluate their impact on error rates. This method combines multiple weak learners to create a strong classifier, often achieving higher accuracy than individual decision trees. The optimization methods are shown in figures 8, 9 and 10.

—

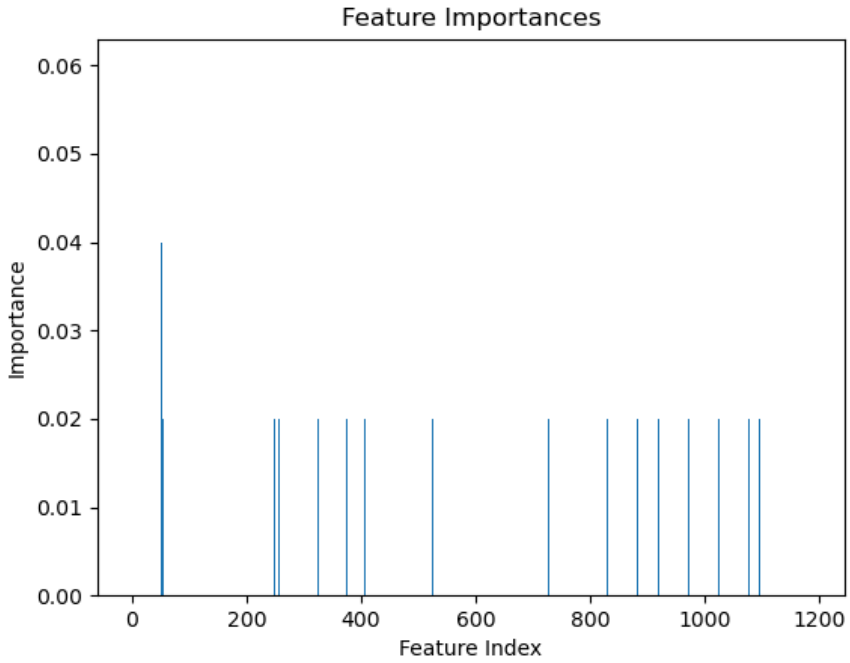


Figure 8: AdaBoost Feature Importance

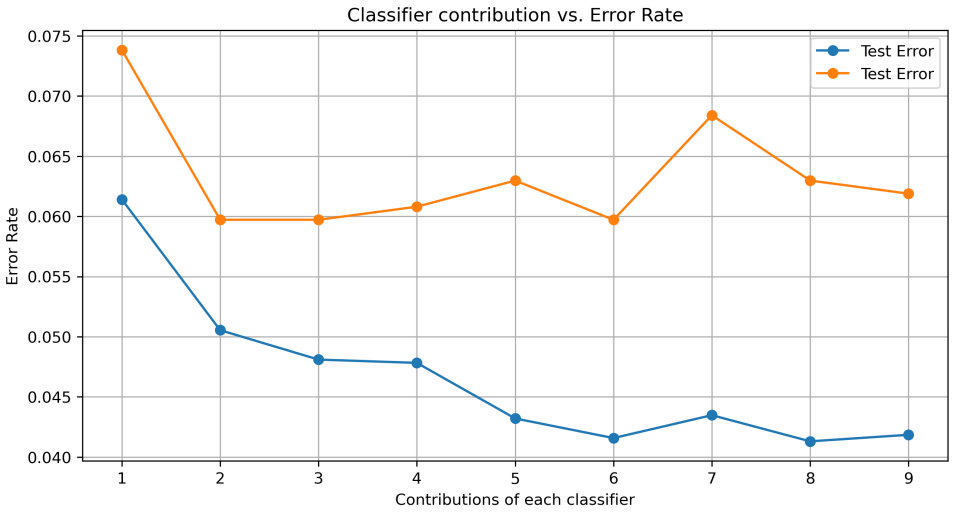


Figure 9: Max stump size vs training and test error rate

3 Part II: Comparative Analysis

This section compares the performance of random forests and boosted decision stumps. Key steps included:

- Using k-fold cross-validation to tune the ensemble size for each method.
- Comparing test errors for the tuned versions of each method.

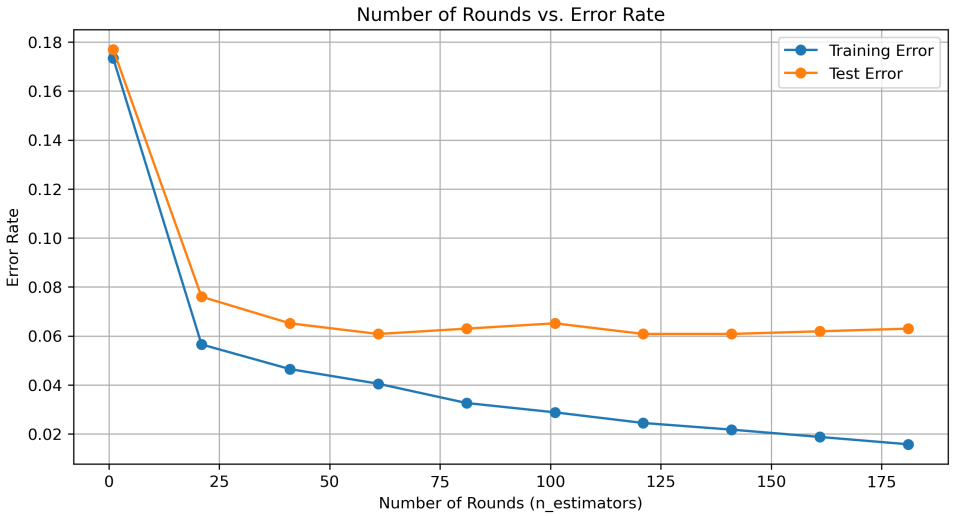


Figure 10: Number of stumps vs training and test error rate

The graphs look the same. I'm pretty sure they're unique. Maybe. Probably.

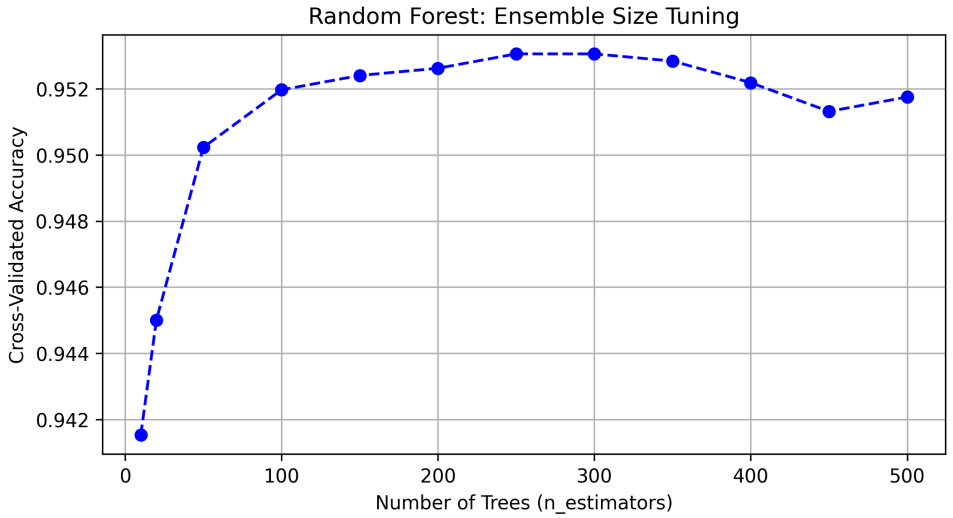


Figure 11: Random Forest Ensemble Size Tuning

—

4 Out-of-Bag Error Estimate

The out-of-bag (OOB) error estimate provides an unbiased measure of the generalization error for random forests. This section details the implementation and results of the OOB error analysis.

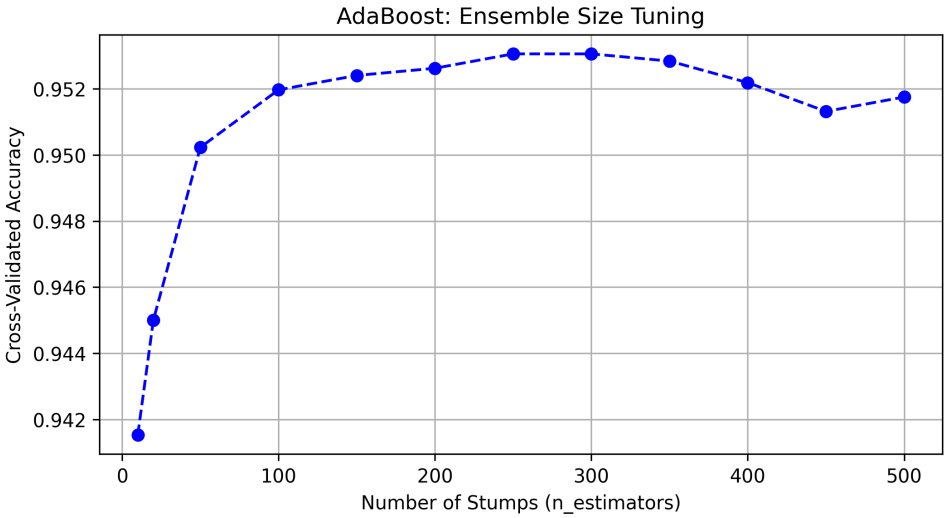


Figure 12: AdaBoost Ensemble Size Tuning

Implementation Details

The OOB error was computed using a Random Forest classifier with `n_estimators` ranging from 10 to 200. Hyperparameters such as `max_depth` and `max_features` were set to default values, and the OOB error was enabled via `oob_score=True`. For each tree count, the OOB error was calculated as $1 - \text{oob\_score}$, where `oob_score` represents the classification accuracy on out-of-bag samples.

Results

Figure 13 shows the OOB error as a function of the number of trees. The error stabilizes beyond approximately 100 trees, indicating that further increases in tree count yield minimal improvement.

Discussion

The OOB error stabilizes at around 3%, closely matching the test error. This confirms the utility of the OOB error as a reliable estimate of generalization performance without the need for a separate validation set.

—

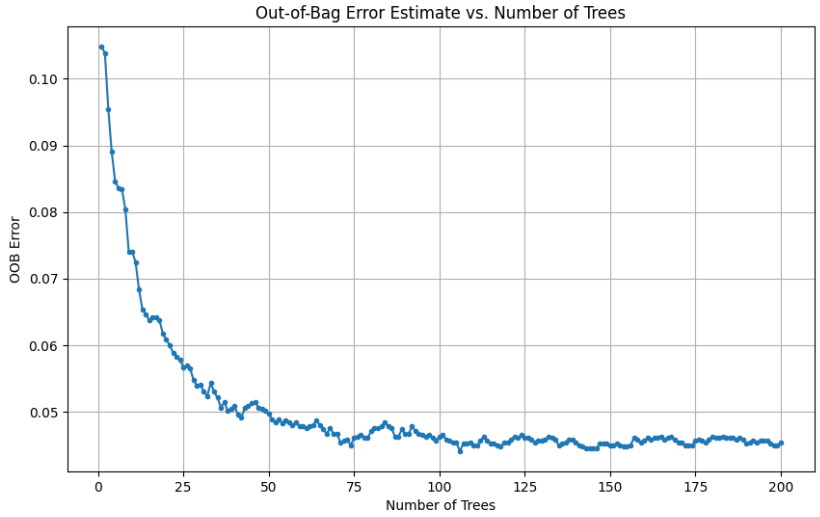


Figure 13: OOB Error vs. Number of Trees in the Random Forest. The error stabilizes as the number of trees increases, indicating convergence.

5 Conclusion

This analysis evaluated the performance of decision trees, random forests, and boosted decision stumps on the Spambase dataset. Key findings include:

- Pruning techniques significantly reduce overfitting in decision trees.
- Random forests achieve robust performance, with feature importance providing insights into the dataset.
- Boosted decision stumps, while computationally intensive, often yield higher accuracy.

Potential improvements include exploring additional hyperparameters and ensemble methods, as well as using larger datasets for more robust evaluation.

—

6 References

- Spambase Dataset: UCI Machine Learning Repository
- Scikit-learn Documentation: scikit-learn