Classifying Emails as Spam or Not Spam using Decision Trees

1. Introduction

The primary objective of this analysis is to classify an email as spam or not spam using decision trees. This dataset was sourced from the UCI Machine Learning Repository. It was originally created by Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt in a study to compare the precision and accuracy of several different models in classifying emails (1999).

By using exploratory data analysis, we seek to identify variables that are most influential in determining whether an email should be classified as spam or not for use in a decision tree.

Problem Domain

Spam emails are a significant issue affecting individuals and businesses worldwide. Spam is unsolicited emails, typically containing advertisements, scams, or phishing schemes designed to compromise personal or financial information (Cranor & LaMacchia, 1998). According to a recent study, spam emails make up over 45% of global email traffic, causing productivity losses and security vulnerabilities (Statista, 2023).

The burden of spam is extensive, affecting internet users by wasting time, creating cybersecurity risks, and increasing the complexity of legitimate email communication management (Hopkins et al., 1999). Effective spam classification models are critical for filtering emails, thereby reducing the negative impacts associated with unsolicited commercial emails.

Method Rationale

Regression analysis is a method used to understand the relationship between an outcome and the observations of the contexts that produced that outcome, i.e. predictors. It quantifies how one or more independent variables are associated with a single dependent variable by fitting a mathematical model to observed data. Such a model can then be used to predict future outcomes or to identify which predictors exert the strongest influence on the response.

Our objective, however, is to **classify** emails as spam (1) or not-spam (0). Although one could force a linear regression framework onto a binary outcome, doing so is problematic for two main reasons:

1. **Unbounded predictions.** Linear regression can produce values below 0 or above 1, making them difficult to interpret as probabilities.

2. **Violated assumptions.** The binary nature of the response invalidates the constant-variance and normally-distributed error assumptions required by ordinary least squares.

Instead, we will use **decision trees**, a classification method that builds a flowchart-like model of decisions and outcomes. Each internal node splits the data on the feature that yields the greatest impurity reduction—measured here by the Gini index—while each leaf assigns a class label. Decision trees are particularly well suited to this problem because they handle both numerical and categorical predictors without extensive preprocessing and produce models that are straightforward to interpret.

To control the tree's complexity and manage the bias–variance trade-off, we tune three key hyperparameters via grid search with 5-fold cross-validation on the training set:

- **max_depth**: Limits the maximum number of splits from root to leaf. The VC dimension of a binary tree with depth d is 2^d; the dataset sample size is 4601. (Blumer et al., 1989). Solving for d would give us an estimate of the max depth (use log function to solve exp equation) 2^d=46012; d≈12.2. We include depths up to 12 to allow sufficient expressiveness.
- **min_samples_split**: The minimum number of observations required to consider splitting a node. Larger values enforce that splits are based on statistically significant sample sizes, reducing overfitting. We test values {2, 5, 10, 20}, corresponding to roughly 0.04%–0.43% of the data.
- **min_samples_leaf**: The minimum number of samples that must appear in a terminal leaf. Requiring more observations per leaf stabilizes class-probability estimates and further guards against noisy splits. We evaluate {1, 2, 4}.
- **criterion='gini'**: Gini impurity is used for its computational efficiency; an alternative is information gain (entropy).

By evaluating these parameter combinations, we identify the tree structure that best generalizes to unseen emails. The chosen model is then retrained on the full 70% training set and its performance assessed on the test set.

2. Exploratory Analysis

2.1 Data Description and Summary Statistics

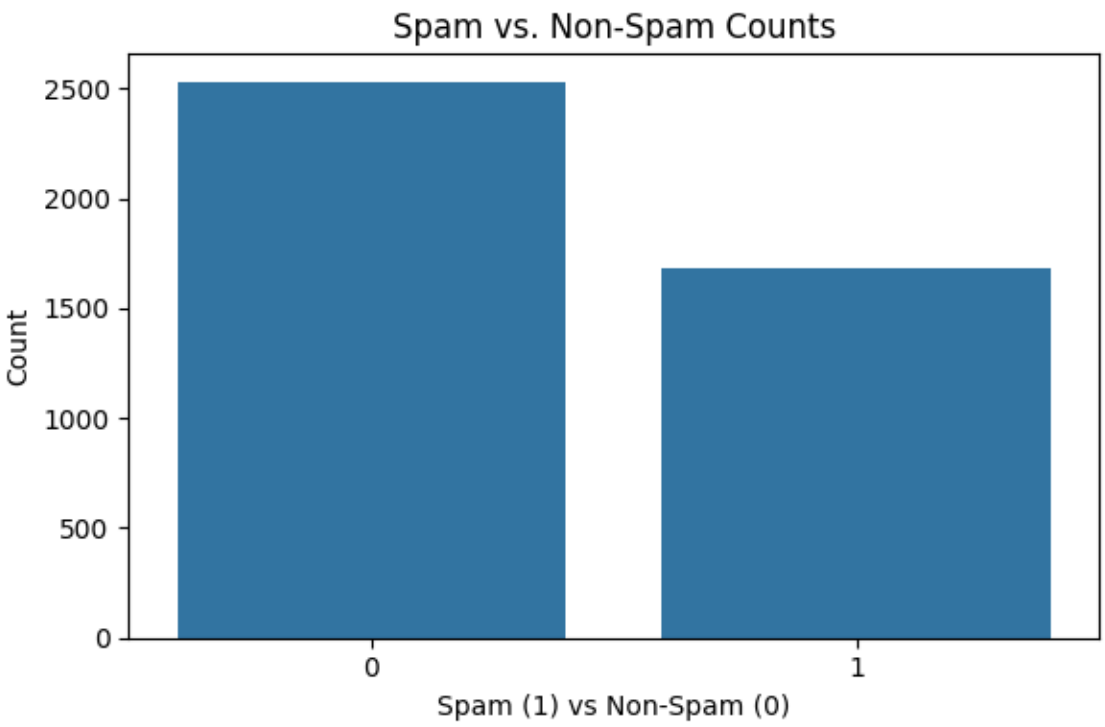
The Spambase data set is comprised of 4,601 records, each representing an email that is considered spam (1) or not spam (0), i.e. unsolicited commercial email (Hopkins et al, 1999). The spam emails were collected from the authors' postmaster and individuals who had filed spam, and the non-spam emails were collected from filed work and personal emails. As such, non spam emails are more likely to contain the name 'george' or the area code '605' (contact information).

The dataset contains 57 continuous independent variables used to predict the target variable, and one binary dependent variable. A brief description of the data attributes are below:

Dataset Attributes:

- 57 Independent Variables:
  - 48 continuous 'word_freq_WORD' variables: The percentage of words in the email that match WORD, i.e.
    100 * (number of times WORD appears in email / total number of words in email)
    A WORD is defined as any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.
  - 6 continuous 'char_freq_CHAR': The percentage of characters in the email that match CHAR, i.e.
    100 * (number of times CHAR appears in email/total characters in email)
  - 1 continuous 'capital_run_length_average;: The average length of uninterrupted sequences of capital letters
  - 1 continuous 'capital_run_length_longest;: The length of longest uninterrupted sequence of capital letters
  - 1 continuous 'capital_run_length_total;: The sum of the length of uninterrupted sequences of capital letters; or, the total number of capital letters in the email
- 1 Dependent Variable:
  - 1 nominal 'Class': denotes whether email is spam (1) or not spam (0)

Summary statistics revealed several insights:



- The target variable 'Class' is moderately imbalanced, with ~60.1% of the emails in the dataset marked as NOT spam,
- For most 'word_freq_WORD' and 'char_freq_CHAR' variables, the median, 25th, and often 75th percentile are all 0; most emails do not contain the vast majority of these words or characters; only a small fraction of messages include them.
- Means are almost always higher than median (0), and max values are over way above the 75th percentile, an indication of heavy right skew and presence of outliers
    - e.g. 'word_freq_internet' has 75th percentile ~ 0, but max ~ 11.1
    - 'char_freq_!' has 75th percentile ~ 0.331, but max ~ 32.48
    - 'capital_run_length_longest' : 75th percentile ~ 44, but max ~ 9,989. indication of heavy right skew and presence of outliers;
- Means and standard deviation values vary greatly between variables, indicating some words appear more frequently than others
- 'capital_run_length' variables show very large spread - average run has mean ~ 5.38, but standard deviation ~ 33.15

2.3 Key variables

All 57 features were used to build the decision tree; as such, key variables were not identified prior to fitting the model. For the sake of the assignment, I have shared some visualization of feature distributions:

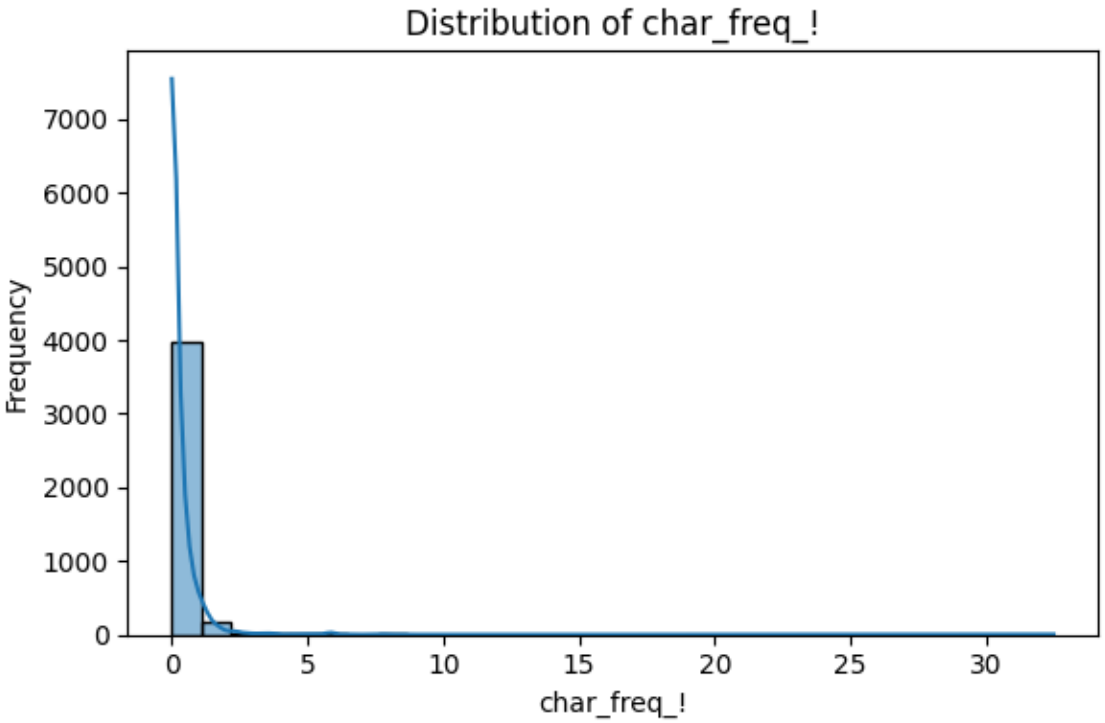


Figure 1 Histogram shows distribution of character "!" – long right tail typical of outliers, heavy positive skew

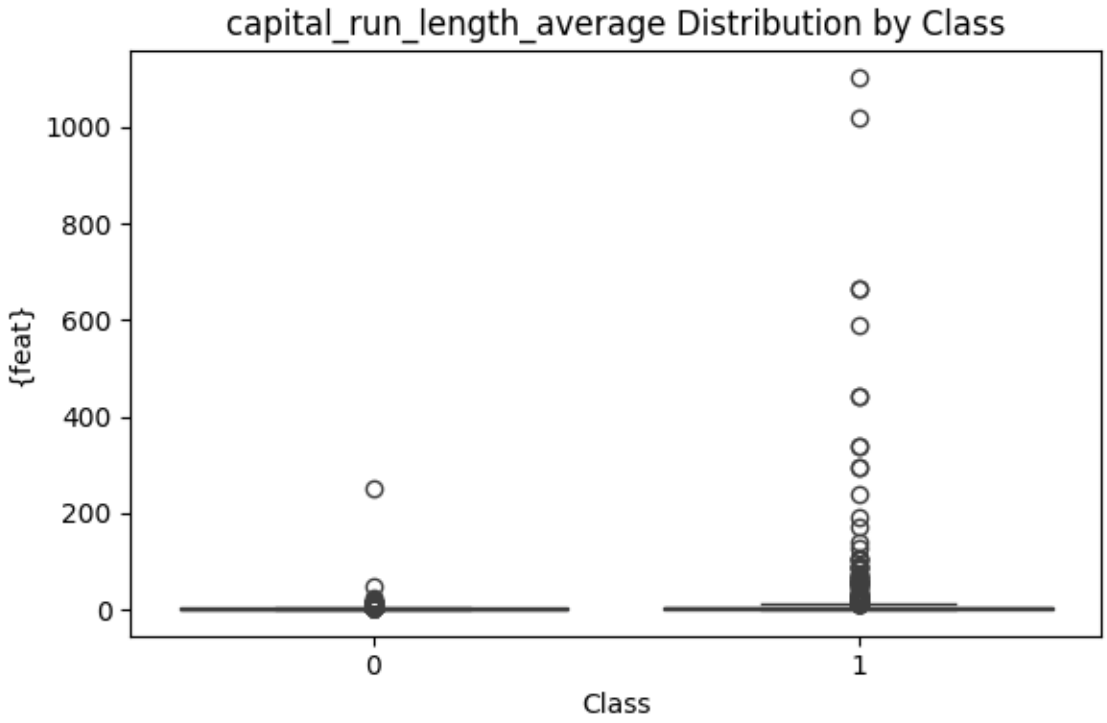


Figure 2 Box plot shows 'capital_run_length_average' distribution, separated by 'Class", i.e. spam =1, non-spam =0; longer sequences of capital letters are more commonly represented in Spam emails than non-spam

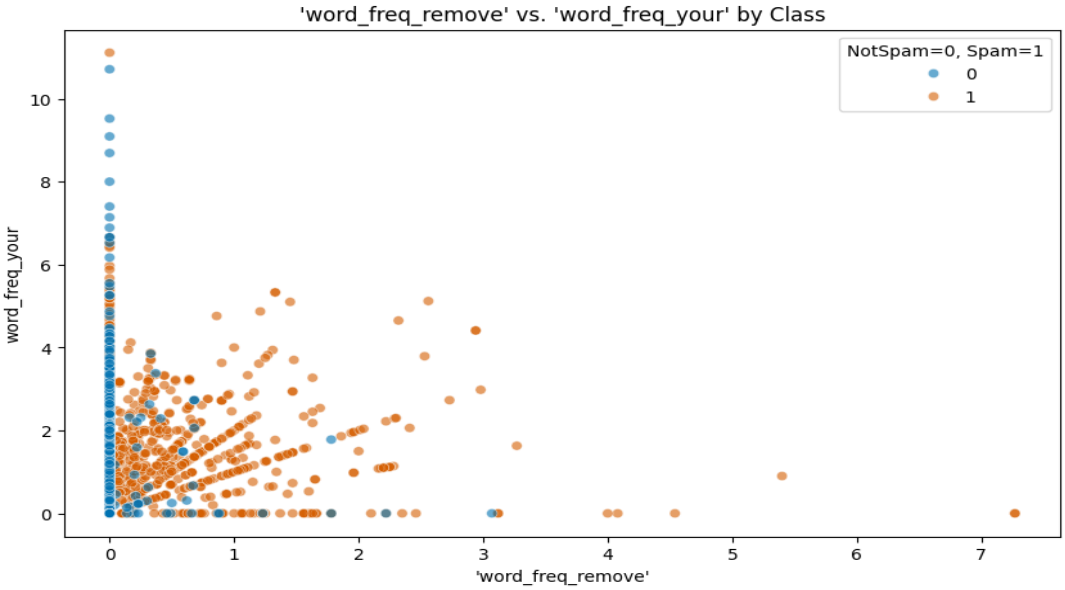


Figure 3 Scatterplot of 'word_freq_REMOVE' and 'word_freq_YOUR' by Spam/Not Spam. Higher counts of the word 'remove' associated with spam emails; word 'YOUR' associated with non-spam emails. See Technical Appendix line 30 for pair plots of top 10 features, and line 35 for comp

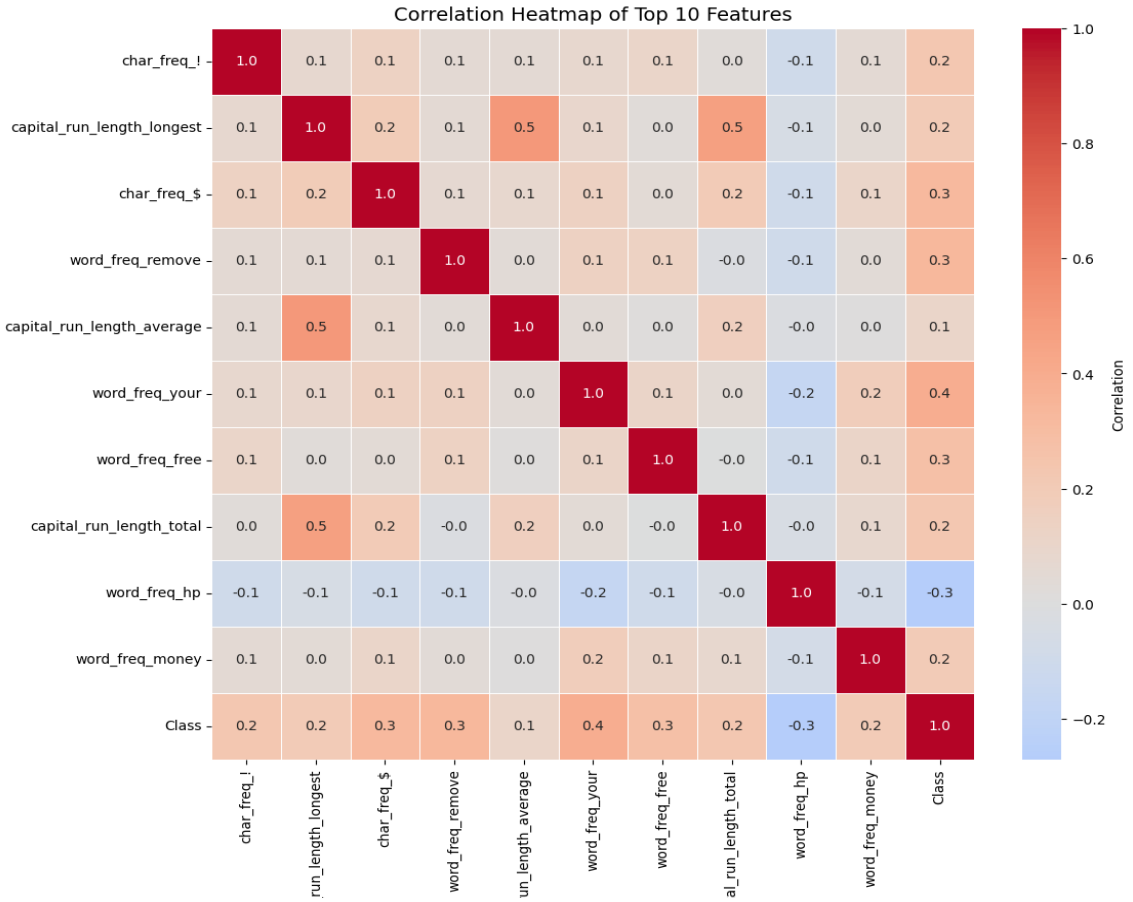


Figure 2 Correlation heatmap of top 10 features. No pairs with correlation values over 0.6, indicating multicollinearity no present in this subset.

3. Preprocessing

3.1 Data Cleaning, Transformation

The preprocessing phase began with the removal of 571 duplicate rows, leaving us with 4,210 unique emails: 1,679 spam and 2,531 non-spam. The UCI dataset has no missing values and consists entirely of numeric features. Since decision trees do not require feature scaling or normalization, we can proceed to split the data. Instead, we will stratify – or segment - the train-test split to preserve the class distribution. (James et al., 2023) Later, we will tune max_depth and min_samples_split using grid search with cross-validation to find the best decision tree model.

3.2 Evaluation of Models (Interpreting confusion matrix)

To assess classifier performance beyond overall accuracy, we examine the confusion matrix, which tabulates predicted versus actual class labels:

- **True Negative (TN):** Predicted *Not Spam* and actual *Not Spam*.
  Correctly identifies legitimate emails.
- **False Negative (FN) (Type I Error):** Predicted *Not Spam* but actual *Spam*.
  Missed spam messages that slip through the filter.
- **False Positive (FP) (Type II Error):** Predicted *Spam* but actual *Not Spam*.
  Legitimate emails wrongly flagged as spam.
- **True Positive (TP):** Predicted *Spam* and actual *Spam*.
  Correctly identifies spam emails.

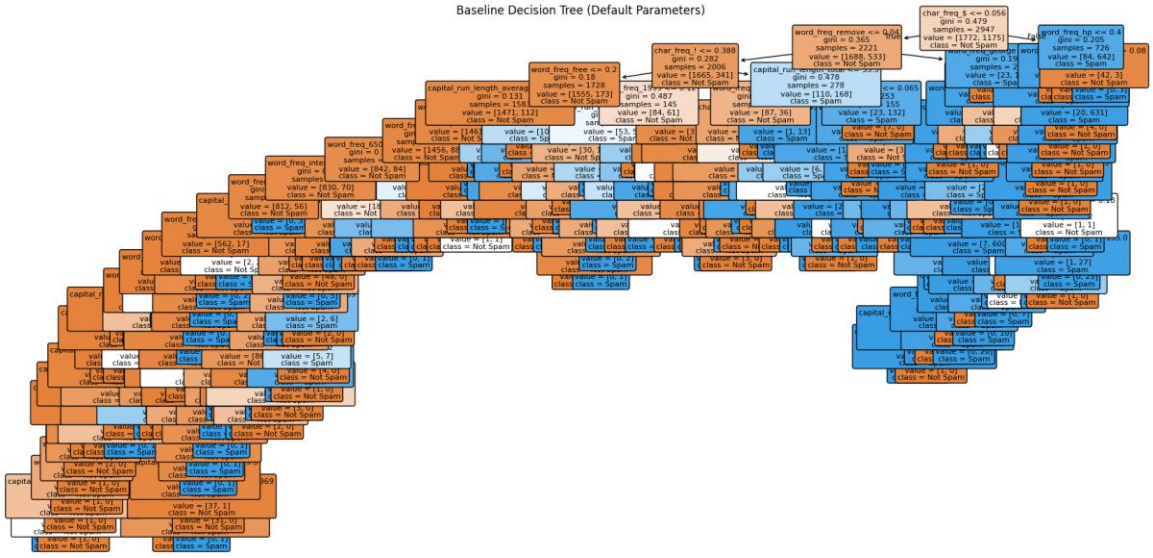
By counting TN, FN, FP, and TP, we compute key metrics:

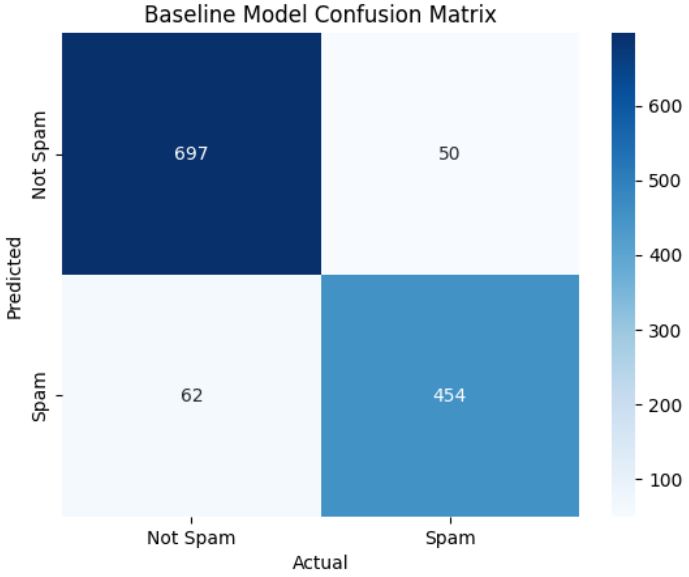
- **Precision:** TP/(TP+FP)
  Of all messages flagged as spam, the fraction that truly are spam.
- **Recall (sensitivity):** TP/(TP+FN)
  Of all actual spam messages, the fraction correctly detected.
- **Specificity:** TN/(TN+FP)
  Of all legitimate emails, the fraction correctly left unfiltered.
- **Accuracy:** (TP+TN)/total
  Overall fraction of correctly classified messages.
- **F1-Score** = 2 * ((Precision x Recall)/(Precision + Recall)
  Overall balanced score

Interpreting these values helps us understand the trade-off between catching spam and avoiding false alarms.

4. The Models

4.1 'Baseline' Decision Tree Model Performance (All 57 features, default parameters)





Precision = TP/(TP+FP) = 454/504 ≈ 0.9000

- When the tree model classifies an email as spam, it's correct about 90% of the time

Accuracy = (TP+TN)/total = 1151/1263 ≈ 0.911

- The model correctly predicts about 91.1% of emails.

Recall = TP/(TP+FN) = 454/516 ≈ 0.879

- The model detects about 87.9% of all actual spam emails.

Specificity = TN/(TN+FP) = 697/747 ≈ 0.933

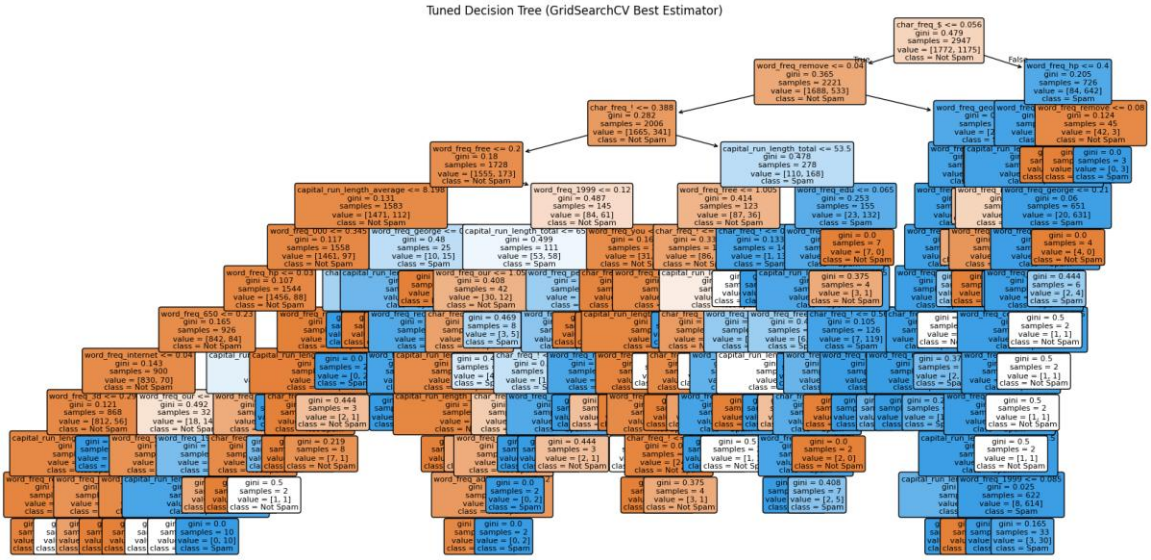
- The model correctly identifies 93.3% of non-spam emails.

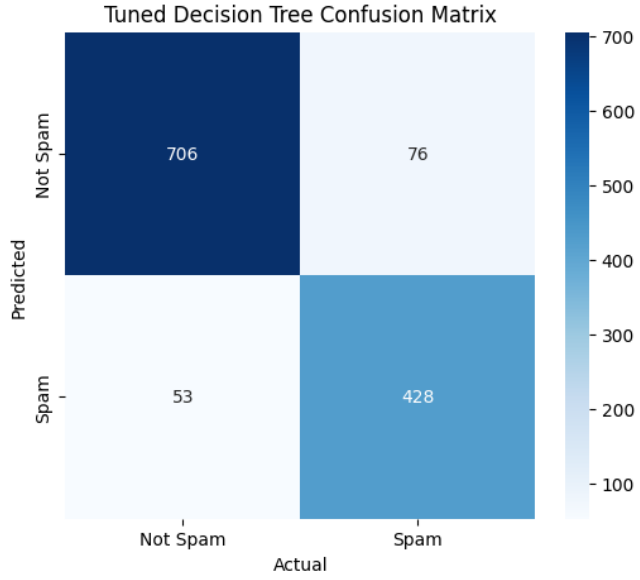
F1-Score = 2*((0.90*0.879)/(0.90+0.879)) ≈ 0.889

- This score balances precision and recall, indicating the model has good overall effectiveness in classifying spam emails.

The baseline model produced 62 false negatives (or missed spam) and 50 false positives (or legitimate emails labeled as spam). For the purpose of classifying emails as spam, we want to keep the number of false positives low, even if it means a few spam emails slip past the threshold.

4.2 'Model 2' – Decision Tree with Hyperparameter Tuning (GridSearchCV)





Precision ≈ 0.890

- When the tree model classifies an email as spam, it's correct about 89.0% of the time

Accuracy ≈ 0.898

- The model correctly predicts about 89.8% of emails.

Recall ≈ 0.849

- The model detects about 84.9% of all actual spam emails.

Specificity ≈ 0.930

- The model correctly identifies 93.0% of non-spam emails.

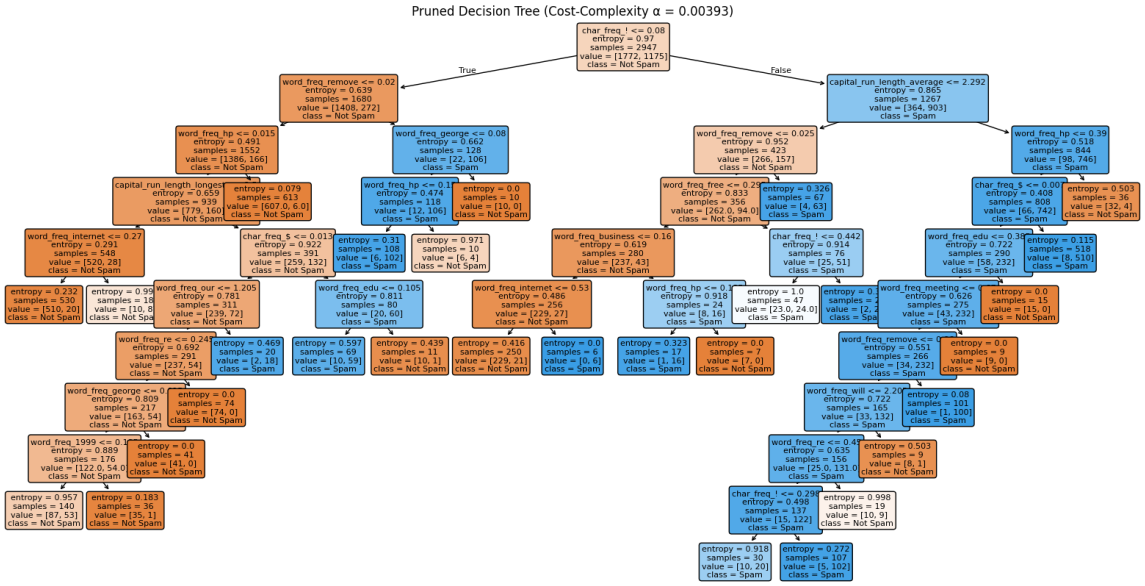
F1-Score ≈ 0.869

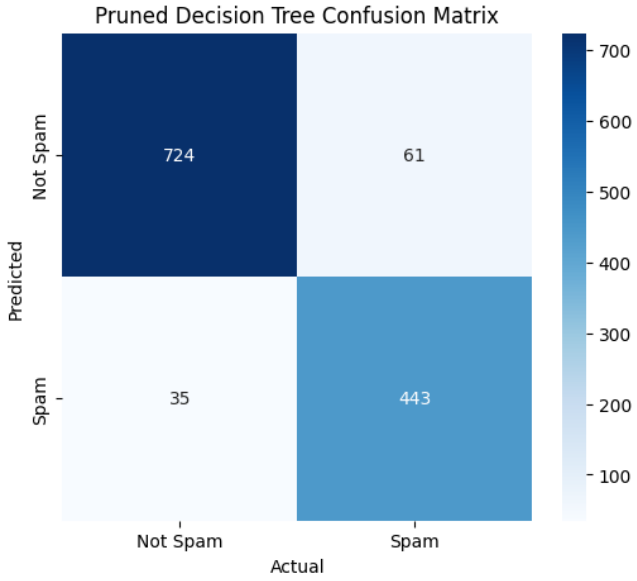
- This score balances precision and recall, indicating the model has good overall effectiveness in classifying spam emails, although not as good as the untuned model.

This model produced 76 false negatives and 53 false positives.

Overall, the tuned model performs worse than the baseline model.

4.3 'Model 3' – Decision Tree with Cost-Complexity Pruning (Cross-Validation)





Precision ≈ 0.927

- When the tree model classifies an email as spam, it's correct about 92.7% of the time

Accuracy ≈ 0.924

- The model correctly predicts about 92.4% of emails.

Recall ≈ 0.879

- The model detects about 87.9% of all actual spam emails.

Specificity ≈ 0.954

- The model correctly identifies 95.4% of non-spam emails.

F1-Score ≈ 0.9022

- This score balances precision and recall, indicating the model has the best overall effectiveness in classifying spam emails among the models.

This model produced 61 false negatives and 35 false positives.

Overall, the pruned model performs better than both the hyperparameter tuned model and the baseline model.

Final Evaluation & Comparison

We compare three decision trees evaluated on the same test set:

Model	**Accuracy**	**Precision**	**Recall**	**F1**
Baseline	0.9113	0.88	**0.90**	0.89
Tuned	0.9000	0.89	0.85	0.87
Pruned	**0.9200**	**0.93**	0.88	**0.90**

Accuracy:

- The pruned tree achieves the highest overall accuracy (0.92), outperforming both the baseline (0.911) and tuned (0.90) models.

Precision:
- The pruned model has the highest precision (0.93), meaning it makes the fewest false-positive errors (legitimate emails marked as spam).
- The tuned model (0.89) slightly improves over the baseline (0.88), but not as much as pruning.

Recall:
- The baseline model has the highest recall (0.90), catching the most actual spam, followed by the pruned model (0.88).
- The tuned model trades off recall for precision, dropping to 0.85.

F1-Score:
- Balancing precision and recall, the pruned tree leads with an F1 of 0.90, slightly above the baseline's 0.89.
- The tuned model scores 0.87, the lowest of the three.

Since minimizing false positives (legitimate mail flagged as spam) is our priority, **the pruned decision tree is the best choice**: it achieves the highest precision (0.93) and overall accuracy (0.92).

5. Conclusion

5.1 Summary

The UCI dataset was already properly formatted for classification tasks. 571 duplicates were dropped prior to training the models. Decision trees don't require normalization, so no further transformation steps were taken to find the best fit.

We built three decision-tree models: a baseline (untuned), a hyperparameter-tuned, and a cost-complexity pruned tree, all evaluated on the same 70/30 test split. We stratified our training data to mimic the class imbalance in our sample dataset.

The pruned decision tree achieved the best overall performance with 92.0% accuracy, 93% precision, 88% recall, and an F1-score of 0.90.

Given our priority of minimizing false positives (legitimate emails marked as spam), the pruned tree is the top choice—it reduces false alarms most effectively while maintaining strong spam detection.

5.2 Limitations

Although this was a relatively clean and straightforward dataset, there were quite a few glaring limitations.

Dataset Limitations

The spambase data was collected in the late 1990s – modern spam is a lot more varied in language and content (images, multi-media, etc). While highly accurate in our analysis, our model would not likely generalize well to the spam email data of today.

All the features were simple word or character frequency counts. There was no information captured about word order, context, or attachments – all factors which would be pretty critical in moderm day spam filtering.

We only explored single decision trees; they can be sensitive to small changes in data and prone to high variance.

No advanced text-based feature engineering (e.g., n-grams, TF-IDF) was applied beyond the UCI-provided attributes.

Model evaluation focused on accuracy, precision, and recall; we did not examine ROC curves or cost-sensitive metrics.

5.3 Improvement Areas

Future versions of this dataset should include a broader time range and additional variables, such as word interactions, presence of multimedia or attachments, etc.

Including metadata features, like sender reputation, email timing, or IP addresses, could improve the model's predictive power and performance.

Implementing random forests, gradient boosting machines, or other ensemble methods could reduce variance and potentially improve both precision and recall.

I would also plot ROC curves, calculate AUC, and perform more extensive cross-validation to further validate model robustness.

5. References

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). Learnability and the Vapnik–Chervonenkis dimension. *Journal of the ACM, 36*(4), 929–965. https://doi.org/10.1145/18565.18578

Cranor, L. F., & LaMacchia, B. A. (1998). Spam! *Communications of the ACM, 41*(8), 74–83. https://doi.org/10.1145/280324.280336

Han, J., Kamber, M., & Pei, J. (2011). Data mining: Concepts and techniques (3rd ed.). Elsevier Science.

Hopkins, M., Reeber, E., Forman, G., & Suermondt, J. (1999). *Spambase Data Set*. UCI Machine Learning Repository. https://doi.org/10.24432/C53G6X

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An introduction to statistical learning with applications in Python* (1st printing July 5, 2023). Retrieved from https://www.statlearning.com/

Statista. (2023). Percentage of spam in email traffic worldwide from 2014 to 2023. Retrieved from https://www.statista.com/statistics/420391/spam-email-traffic-share/

Taskesen, E. (n.d.). Algorithm — PCA PCA documentation. Retrieved from https://erdogant.github.io/pca/pages/html/Algorithm.html

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37–45.

University of Maryland Global Campus. (2023). *Decision Trees* [PowerPoint slides]. Brightspace.