Machine Learning And Statistical Learning

Machine Learning Project: Tree Predictors for Binary Mushroom Classification

Prepared by Adiya Sapargali

DSE, second year

Matricola number: 33486A

Table of Contents

Introduction

This project implements *decision tree* and *random forest classifiers* from scratch for the binary classification task of identifying whether mushrooms are edible or poisonous using the Mushroom dataset. The main goal is to understand the internal mechanics of tree-based models, including their training, splitting criteria, and stopping conditions.

The implementation includes reading and preprocessing the Mushroom dataset, analyzing feature distributions, and building a custom decision tree classifier. Three splitting criteria *(Gini impurity, Entropy, and Classification Error)* were implemented to assess their influence on tree structure and model performance. Stopping criteria such as maximum tree depth and minimum samples per leaf were used to prevent overfitting. Additionally, a random forest classifier was developed by aggregating multiple decision trees trained on bootstrapped samples with random feature subsets to enhance model robustness.

Model performance was evaluated using accuracy, confusion matrices, and classification reports. The results highlighted the trade-offs between different splitting criteria, the impact of stopping rules, and the benefits of ensemble methods in improving generalization.

Data Loading and Preprocessing

The project utilized two datasets: the primary dataset and the secondary dataset, both in CSV format. They were loaded into the analysis environment using appropriate delimiters and encodings to make sure all data were read properly. All column names were cleaned from extra spaces to keep the format consistent.

In the primary dataset, some numerical columns, such as *cap diameter*, *stem height*, and *stem width*, were stored as text ranges like "[4, 8]". To convert them into usable numbers, a helper function called parse_measured_value() was created. This function checks whether a cell contains a range, splits the two values, turns them into numbers, and replaces the range with their average. If the cell already contains a single number, it simply converts it to a float; if the value cannot be converted, it is set as missing (NaN). This step made all numerical features consistent and ready for analysis.

The secondary dataset was already clean, so only basic checks were needed to confirm the data types and structure. During the data exploration stage, missing values were also reviewed. Since only a few records were incomplete, those rows were removed to make sure the models were trained on clean and reliable data. This preparation ensured that the datasets were consistent and ready for modeling.

Exploratory Data Analysis (EDA)

The Exploratory Data Analysis (EDA) stage aimed to understand the structure and characteristics of the primary and secondary datasets, identify potential issues, and inform subsequent modeling decisions.

Primary Dataset

The primary dataset was analyzed to determine the distribution of the target variable (class) and to explore the behavior of both numerical and categorical features.

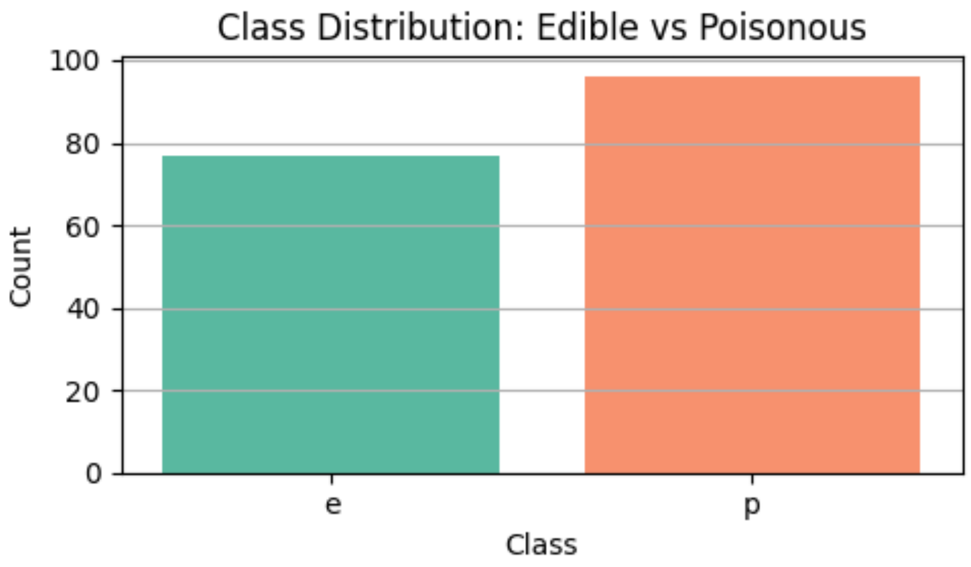
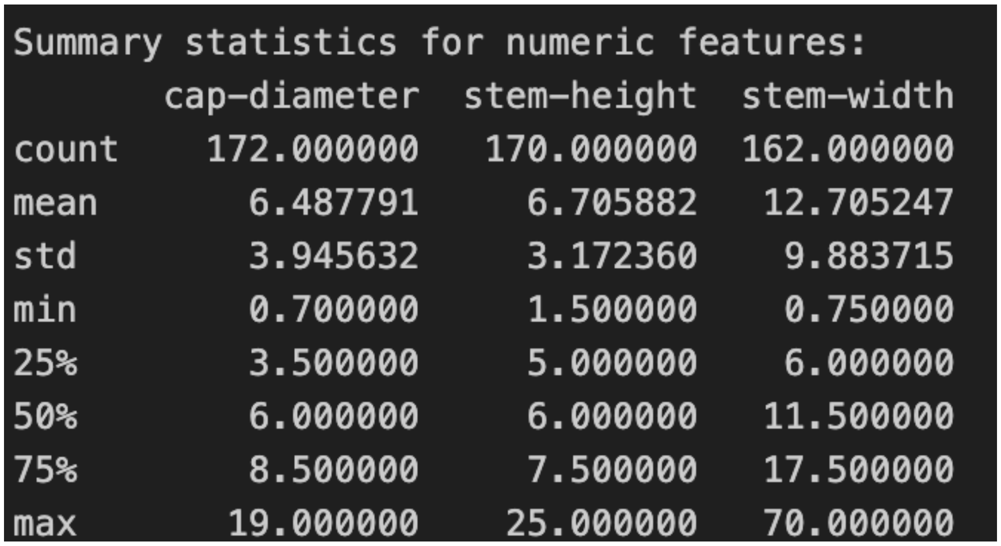


Figure 1. Class Distribution (Edible vs. Poisonous)

The dataset contains two target classes: edible (e) and poisonous (p). There are 96 poisonous samples and 77 edible ones, meaning approximately 55.5% of the mushrooms are poisonous and 44.5% are edible. This indicates a slight class imbalance, with poisonous mushrooms being the majority class. This distribution highlighted the importance of stratified sampling when splitting data for training and testing to ensure both classes were proportionally represented.



```
Summary statistics for numeric features:
       cap-diameter  stem-height  stem-width
count    172.000000   170.000000  162.000000
mean       6.487791     6.705882   12.705247
std        3.945632     3.172360    9.883715
min        0.700000     1.500000    0.750000
25%        3.500000     5.000000    6.000000
50%        6.000000     6.000000   11.500000
75%        8.500000     7.500000   17.500000
max       19.000000    25.000000   70.000000
```

Table 1. Summary Statistics for Numeric Features

The table shows summary statistics for three numeric features in the dataset: cap diameter, stem height, and stem width. On average, mushroom caps are about 6.49 cm in diameter, stems are about 6.71 cm tall, and 12.71 cm wide. The values vary, with cap diameter ranging from 0.7 to 19.0 cm, stem height from 1.5 to 25.0 cm, and stem width from 0.75 to 70.0 cm. The stem width has the highest variation, as shown by its large standard deviation and wide range, suggesting possible outliers. Most values fall within the interquartile range, which helps understand the typical sizes of these features in the dataset.

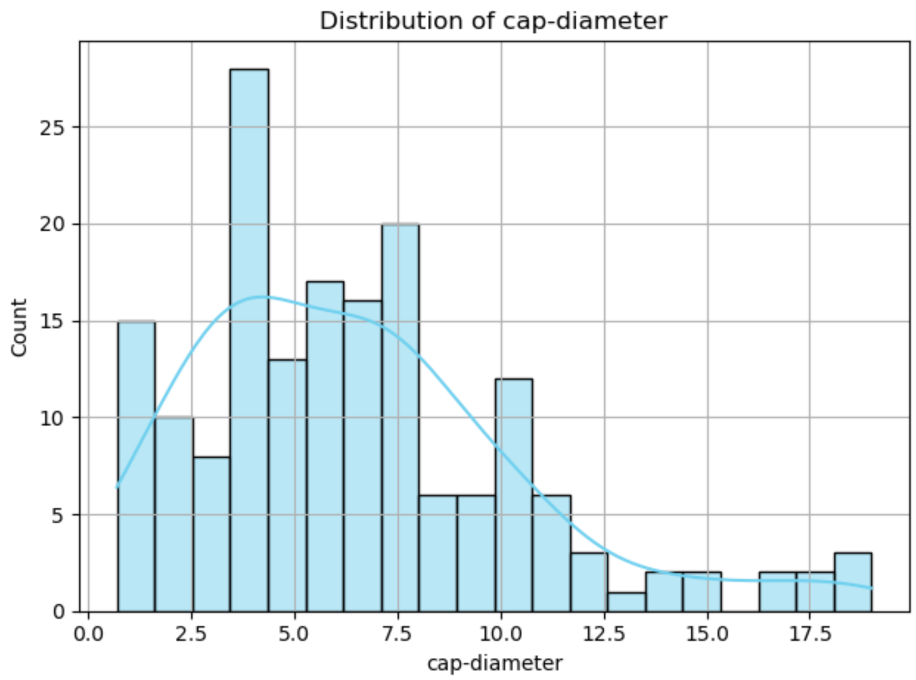


Figure 2. Distribution of cap-diameter

This plot shows the distribution of the cap-diameter feature in the dataset. Most mushrooms have a cap diameter between 2 and 8 centimeters, with a clear peak around 4 cm. The distribution is right-skewed, meaning there are a few mushrooms with much larger cap diameters, up to 18–19 cm, but they are relatively rare. The shape of the curve suggests that smaller caps are more common, and the number of samples decreases as the diameter increases. This information is useful to understand the variability of this feature and check whether it contains extreme values or outliers that could affect model performance.

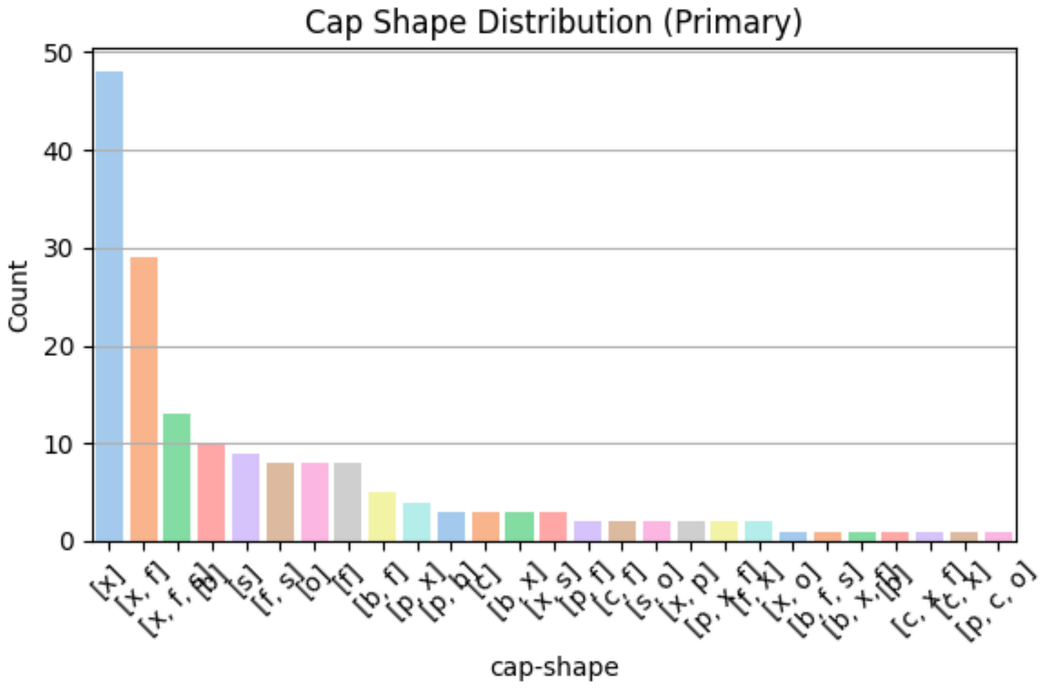


Figure 3. Cap shape distribution

This plot shows the distribution of different cap-shape combinations in the primary mushroom dataset. The most common cap shape is represented by the value [x], which appears nearly 50 times. The next most frequent shapes include [x, f] and [f], but their counts drop off quickly compared to the most common one. The distribution is highly skewed, with a small number of shape combinations making up the majority of samples, while many others occur only a few times. This suggests that certain cap shapes are much more typical in the dataset, and the rare combinations may have limited impact on the overall model but could be important for specific identification cases.

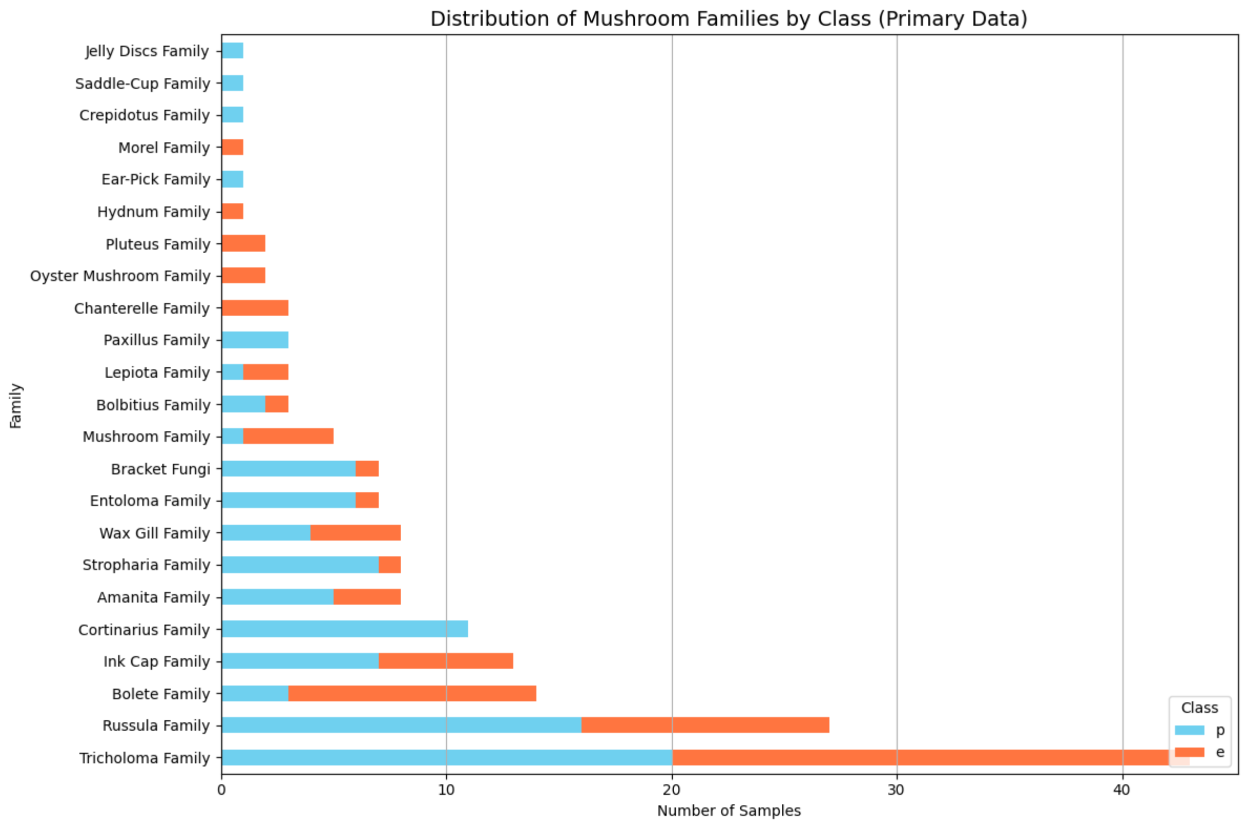


Figure 4. Distribution of Mushroom Families by Class

This plot shows the distribution of mushroom families by class (edible vs. poisonous) in the primary dataset. It helps reveal how different families are associated with each class. For example, the Tricholoma and Russula families contain many edible species, while the Cortinarius and Amanita families include more poisonous ones. Some families, like the Boletus and Ink Cap, contain both edible and poisonous species in nearly equal proportions. It helps identify which families might be strong indicators of class, uncover potential patterns in species distribution, and guide feature importance or selection decisions in model building. It also highlights class imbalance across categories, which can impact classification performance.

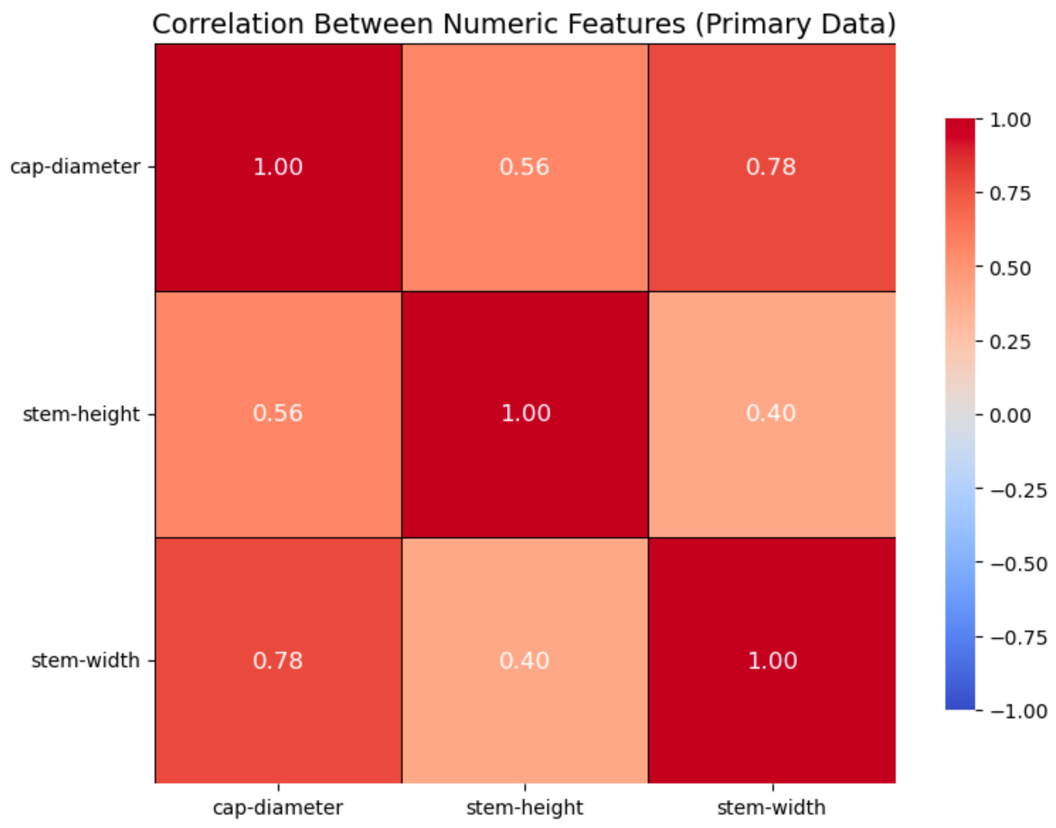


Figure 5. Correlation between Numeric Features

This heatmap shows the correlation between the three numeric features in the primary dataset: cap-diameter, stem-height, and stem-width. The values range from 0.40 to 0.78, indicating moderate to strong positive correlations. The strongest correlation is between cap-diameter and stem-width (0.78), meaning mushrooms with larger caps tend to also have thicker stems. The correlation between cap-diameter and stem-height is also moderate (0.56), while stem-height and stem-width are less correlated (0.40). It helps identify relationships between features, which can inform feature selection, model interpretation, or the need for dimensionality reduction. It also shows that none of the numeric features are redundant (correlation close to 1), so all of them may provide unique information for classification.

Secondary Dataset

The secondary dataset is used for model training. The analysis focuses on class balance, numeric feature behavior, categorical variable distributions, and potential feature relationships with the target variable.

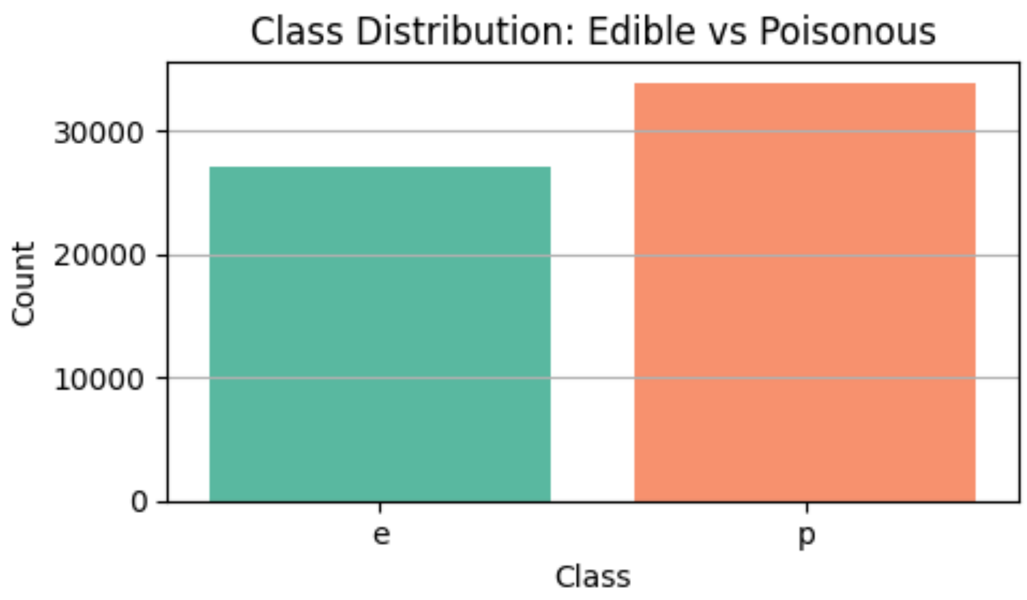


Figure 6. Class Distribution: Edible vs Poisonous

The class distribution plot shows the number of edible (e) and poisonous (p) mushrooms in the dataset. The dataset exhibits a slight class imbalance, with approximately 33,888 poisonous mushrooms and 27,181 edible mushrooms. Poisonous mushrooms make up about 55.5% of the dataset, while edible mushrooms comprise about 44.5%. This distribution underscores the importance of using stratified sampling during data splitting to ensure that both classes are proportionally represented in the training and testing sets.

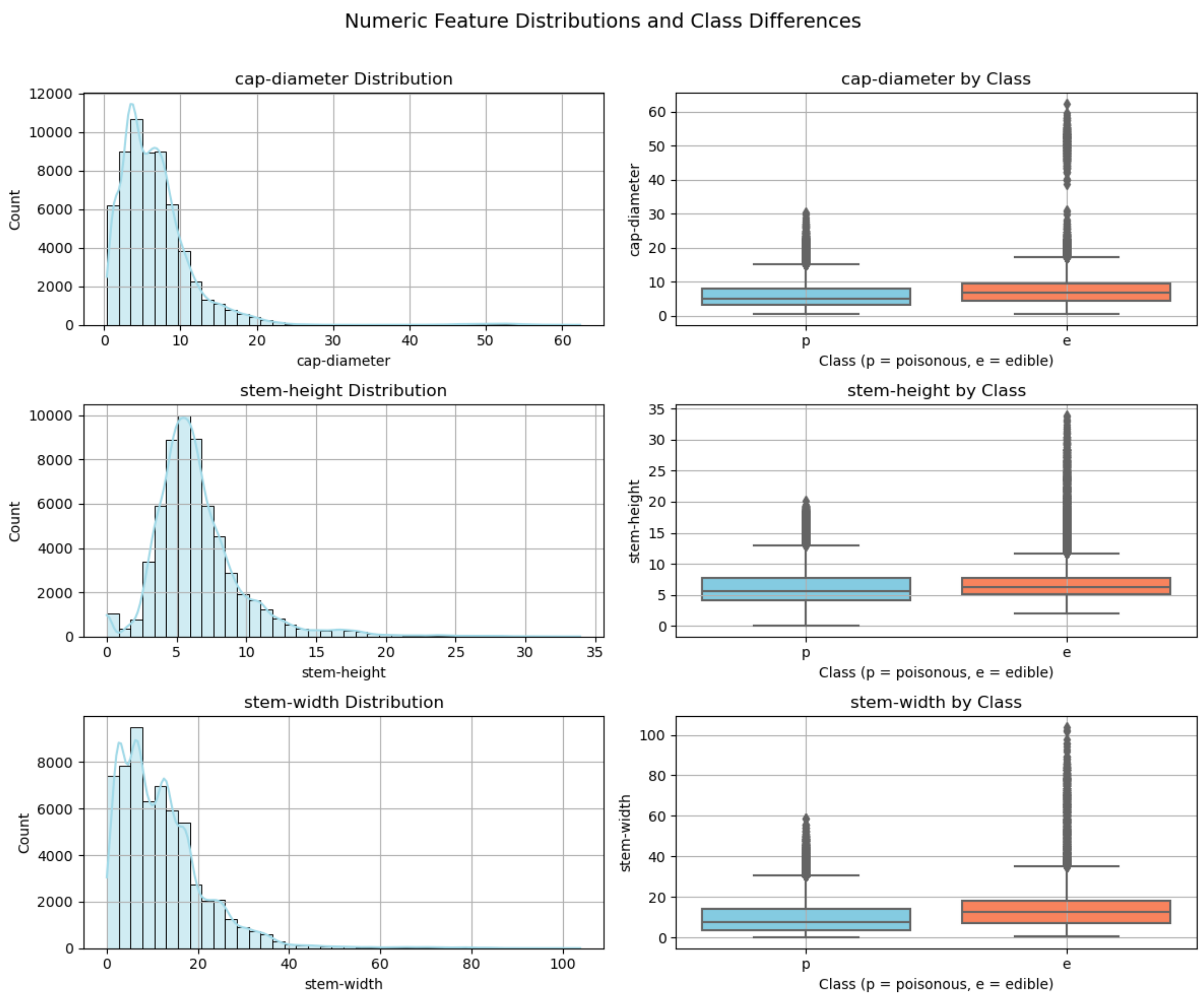


Figure 7. Numeric Feature Distributions and Class Differences

The distributions of the three numeric features (cap diameter, stem height, and stem width) were analyzed to understand their overall behavior and differences between edible and poisonous mushrooms.

- *Cap Diameter:* The distribution is right-skewed, with most mushrooms having caps between 2 cm and 10 cm. Both classes show similar patterns, although edible mushrooms tend to have slightly larger cap diameters on average.
- *Stem Height:* The stem height feature also shows a right-skewed distribution, with most values between 4 cm and 10 cm. There is a slight overlap between the classes, with edible mushrooms showing slightly higher median stem heights.

- *Stem Width:* Stem width displays the largest variability, with values ranging from less than 1 cm to over 70 cm. Edible mushrooms tend to have higher stem widths on average, but there is significant overlap between classes.

Boxplots for each feature by class highlight that none of the features alone perfectly separates the classes. While there are small differences in the medians and ranges, substantial overlap exists, suggesting that tree-based models will need to consider combinations of features to achieve accurate classification.

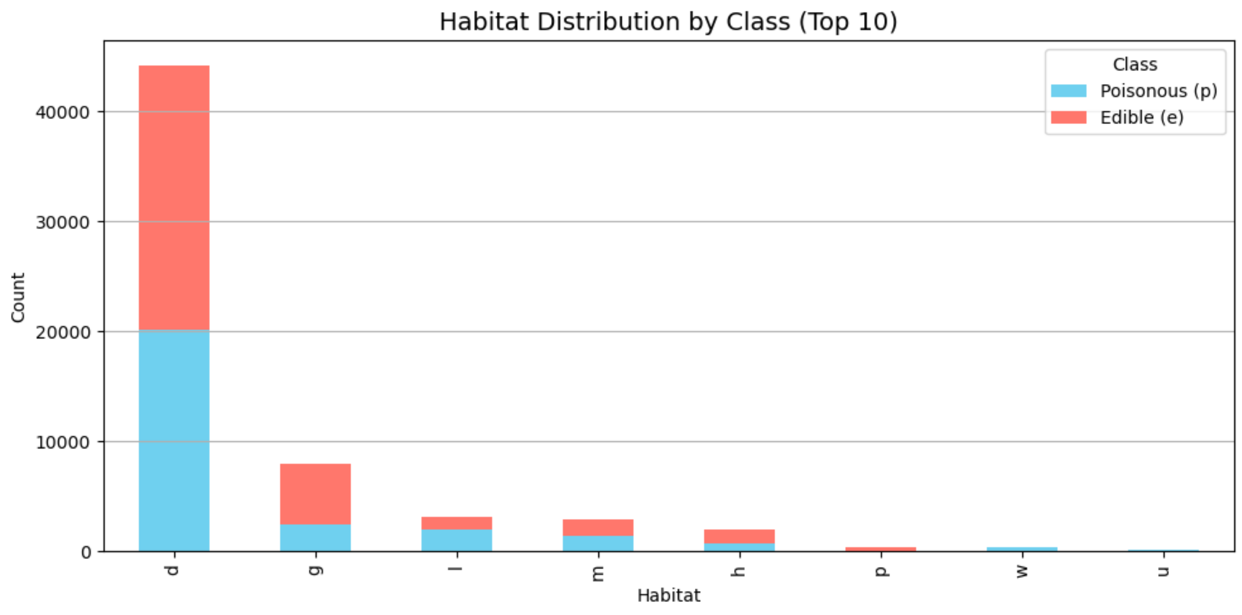


Figure 9. Habitat Distribution by Class

The habitat distribution plot illustrates the frequency of mushrooms in various habitats, split by class. The most common habitat overall is habitat "d," which contains a large proportion of both edible and poisonous mushrooms. Poisonous mushrooms are more frequent in every habitat, with particularly high counts in habitat "d." Other habitats such as "g," "l," "m," and "h" show lower counts but still include both classes. Some habitats like "w" and "u" are very rare and have minimal representation in the dataset. This distribution suggests that habitat alone does not fully separate edible from poisonous mushrooms, but may still contribute useful information when combined with other features in classification models.

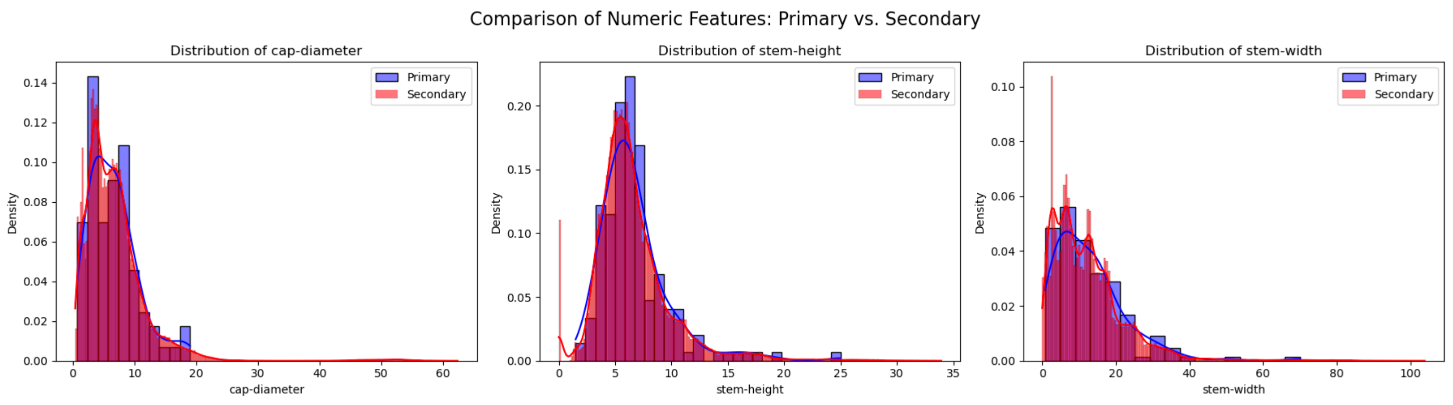


Figure 12. Comparison of Numeric Features: Primary vs Secondary

The density plots compare the distributions of cap diameter, stem height, and stem width between the primary and secondary datasets. Overall, the distributions are similar across datasets, with slight variations:

- *Cap Diameter:* The secondary dataset shows a slightly broader distribution with more samples at higher diameters, but both datasets are similarly right-skewed.
- *Stem Height:* Both datasets have consistent distributions, peaking around 5 to 7 cm, with no significant differences.
- *Stem Width:* The secondary dataset has a slightly longer tail, indicating the presence of a few outliers with larger stem widths compared to the primary dataset.

These similarities support using the secondary dataset for model training and evaluation, while acknowledging some differences that may influence model generalization.

Methodology

This section describes the step-by-step approach used to develop, train, and evaluate decision tree and random forest classifiers for predicting mushroom edibility. The methodology integrates data preparation, model construction, splitting and stopping criteria, hyperparameter tuning, and evaluation metrics. Each component is described in detail to provide a comprehensive overview of how the project was implemented from both a theoretical and practical standpoint.

The target variable was transformed into a binary format, assigning the value 1 to poisonous mushrooms and 0 to edible ones. All categorical features were converted into numerical codes using label encoding to support threshold-based splitting. The dataset was then partitioned into training, validation, and test sets through stratified sampling to preserve the original class distribution across all splits.

Decision Tree Implementation

The decision tree classifier was designed using two main classes: Node and TreePredictor. The fundamental building block of the classifier is the *Node class*. Each node contains all information required to perform a decision or return a prediction. A node may function either as:

- a *leaf node*, storing the predicted class label, or
- a *decision node*, storing:
    - the index of the feature used for splitting,
    - the threshold value for the binary test,
    - references to the left and right child nodes.

A decision node applies a simple rule:

$$x[\text{feature\_index}] \leq \text{threshold}$$

Samples satisfying the condition are directed to the left child and all others follow the right child. This structure creates a binary branching process based on a single feature at each step. During prediction, the **evaluate()** function follows these branches until a leaf is reached, and then returns the stored class label.

The *TreePredictor class* is responsible for building and evaluating the decision tree. The tree is constructed recursively, starting from the root node. At each internal node, all possible splits across all features are examined. For every feature, the algorithm evaluates every unique threshold and computes the impurity reduction produced by that split. The split that yields the highest improvement in purity is selected, and the dataset is divided into left and right subsets according to the rule:

$$x[\text{feature\_index}] \leq \text{threshold}$$

This recursive splitting process continues until one of the stopping conditions is met, resulting in a hierarchical structure where each node becomes increasingly homogeneous. This approach follows the standard top-down decision tree procedure in which each split aims to reduce uncertainty and improve classification accuracy.

Splitting Criteria

Three impurity measures were implemented to determine how informative each potential split is:

- *Gini Impurity.* Measures the probability of incorrect classification based on label distribution. Mathematically defined as

$$Gini(p) = 1 - \sum p_i^2 \text{ ,}$$

where p_i is the proportion of class i in the node. This criterion is smooth, efficient, and commonly used in practice.

- *Entropy.* Measures uncertainty in the node and is used to compute information gain. Mathematically defined as

$$Entropy(p) = -\sum p_i \, log_2 \, (p_i),$$

where p_i is the proportion of class i in the node. Entropy tends to favor balanced and informative splits.

- *Classification Error.* Reflects the fraction of incorrectly classified samples. Mathematically defined as

$$Error(p) = 1 - \max(pi),$$

where p_i is the proportion of class i in the node. This criterion is less sensitive to subtle changes but is useful for comparison.

During training, each candidate split is evaluated using these measures, and the split producing the highest impurity reduction (information gain) is selected.

Stopping Criteria

To prevent excessive growth of the tree and limit overfitting, two constraints were implemented:

- *Maximum Tree Depth.* Restricts how deep the tree can grow, reducing model complexity and limiting variance.
- *Minimum Samples per Split.* Ensures that a node must contain a sufficient number of samples before attempting a split, preventing unreliable decisions based on very small subsets.

These stopping rules are standard mechanisms in statistical learning to balance accuracy and generalization.

Hyperparameter Tuning and Model Selection

A grid search was conducted over combinations of maximum depth and minimum samples per split for each splitting criterion. For every configuration, the model was trained on the training set and evaluated on the validation set. Training and validation errors were recorded and visualized using heatmaps to identify the presence of underfitting or overfitting.

The configuration with the lowest validation error was selected as the best-performing model for each impurity criterion.

Final Evaluation

After determining the optimal hyperparameters, the final model was retrained on the combined training and validation data. Performance on the held-out test set was measured using accuracy, classification reports, and confusion matrices. This evaluation provided insight into the model's predictive ability and highlighted any class-specific strengths or limitations.

Random Forest Implementation

The Random Forest classifier was implemented to improve performance by combining multiple decision trees into an ensemble. This approach reduces variance and increases generalization compared to relying on a single decision tree. Each tree is trained on a bootstrap sample of the training data, meaning samples are selected randomly with replacement to form a unique subset for each tree. This creates diversity among the trees and helps prevent the ensemble from overfitting to specific training instances.

In addition to bootstrapping, each split within a tree considers only a random subset of features rather than all available features. This method, often referred to as the Random Subspace Method, ensures that individual trees focus on different features and patterns in the data.

Randomizing both data samples and feature subsets increases diversity across the ensemble and stabilizes the final predictions.

Each decision tree within the forest uses the same splitting criteria (Gini impurity, Entropy, or Classification Error) and the same stopping rules (maximum depth and minimum samples per split) as used in the stand-alone decision tree implementation. This makes the Random Forest directly comparable to the single-tree models while introducing additional robustness through ensembling.

After all trees are trained, predictions are combined using majority voting, where each tree casts a prediction, and the class receiving the most votes becomes the final output of the ensemble. This voting mechanism helps balance out the errors made by individual trees and produces a more reliable classifier.

Overall, the Random Forest implementation in this project integrates bootstrap sampling and random feature selection to create a diverse and stable ensemble model, offering improved predictive performance for the mushroom classification task.

Experiments and Results

This section presents a series of experiments conducted to evaluate the performance of the custom decision tree classifiers implemented from scratch using the Mushroom dataset. The primary focus was to investigate how different splitting criteria (Gini impurity, Entropy, and Classification Error) and various hyperparameter settings affect model performance and generalization.

The experiments were carried out using the preprocessed secondary Mushroom dataset described in the Methodology section, where categorical features were label-encoded and the target variable was binarized. The data was partitioned into training, validation, and test sets using stratified sampling to maintain class balance.

For each splitting criterion, models were trained using a grid search approach over various hyperparameter configurations. As detailed in the Methodology section, the grid included maximum tree depths of 3, 5, 7, and 9, and minimum samples per split values of 2, 5, and 10. These values were selected to provide a systematic exploration of both shallow and deeper trees, allowing assessment of underfitting and overfitting tendencies.

Each model was trained on the training set, and performance was evaluated on the validation set using classification accuracy and zero-one loss. The configuration yielding the lowest validation error was chosen as the best model for each splitting criterion.

Following hyperparameter tuning, the best-performing models were evaluated on the held-out test set. Performance metrics such as test accuracy, classification reports, and confusion matrices were used to assess the models' generalization capabilities.

Gini Splitting Criterion

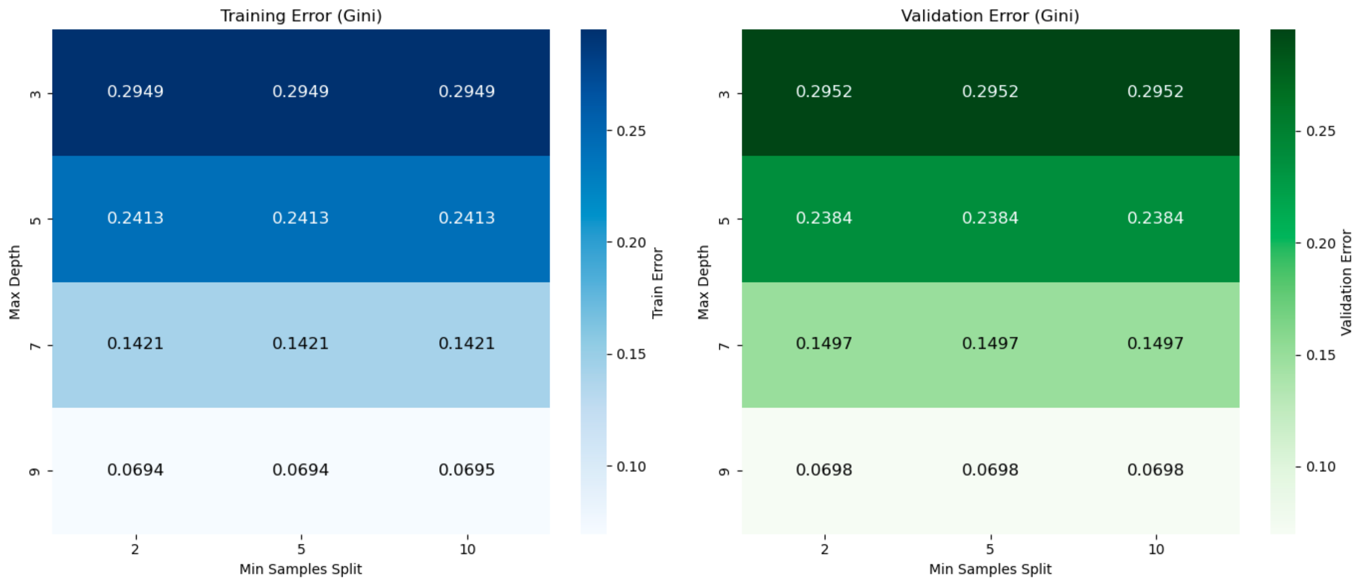
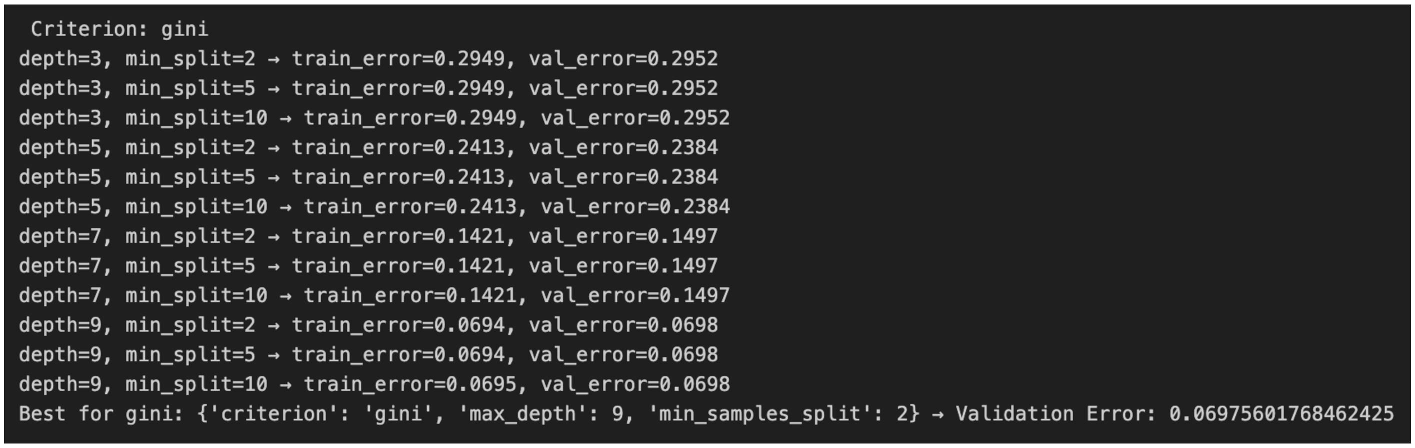


Figure 13. Training and Validation Error Heatmaps for Gini Splitting Criterion



```
 Criterion: gini
depth=3, min_split=2 → train_error=0.2949, val_error=0.2952
depth=3, min_split=5 → train_error=0.2949, val_error=0.2952
depth=3, min_split=10 → train_error=0.2949, val_error=0.2952
depth=5, min_split=2 → train_error=0.2413, val_error=0.2384
depth=5, min_split=5 → train_error=0.2413, val_error=0.2384
depth=5, min_split=10 → train_error=0.2413, val_error=0.2384
depth=7, min_split=2 → train_error=0.1421, val_error=0.1497
depth=7, min_split=5 → train_error=0.1421, val_error=0.1497
depth=7, min_split=10 → train_error=0.1421, val_error=0.1497
depth=9, min_split=2 → train_error=0.0694, val_error=0.0698
depth=9, min_split=5 → train_error=0.0694, val_error=0.0698
depth=9, min_split=10 → train_error=0.0695, val_error=0.0698
Best for gini: {'criterion': 'gini', 'max_depth': 9, 'min_samples_split': 2} → Validation Error: 0.06975601768462425
```

Table 2. Numerical Results of Training and Validation Errors for Gini Splitting Criterion

The plots summarize the training and validation errors for decision trees trained using the Gini impurity criterion. Both heatmaps show that increasing the maximum tree depth consistently reduced training error, from approximately 0.295 at depth 3 to around 0.069 at depth 9. Validation error followed a similar decreasing trend, reaching its lowest value of roughly 0.070 at depth 9.

Interestingly, the minimum samples per split parameter had little impact on either training or validation errors, as the values remained almost constant across different settings. This suggests that, for this dataset and criterion, tree depth was the dominant factor influencing performance.

Overall, these results indicate that deeper trees improved both training and validation accuracy without clear signs of overfitting at the evaluated parameter ranges, supporting the robustness of the Gini impurity criterion for this classification task. Notably, the best model was achieved with a maximum tree depth of 9 and a minimum samples per split of 2, yielding the lowest validation error of approximately 0.070.

Entropy Splitting Criterion

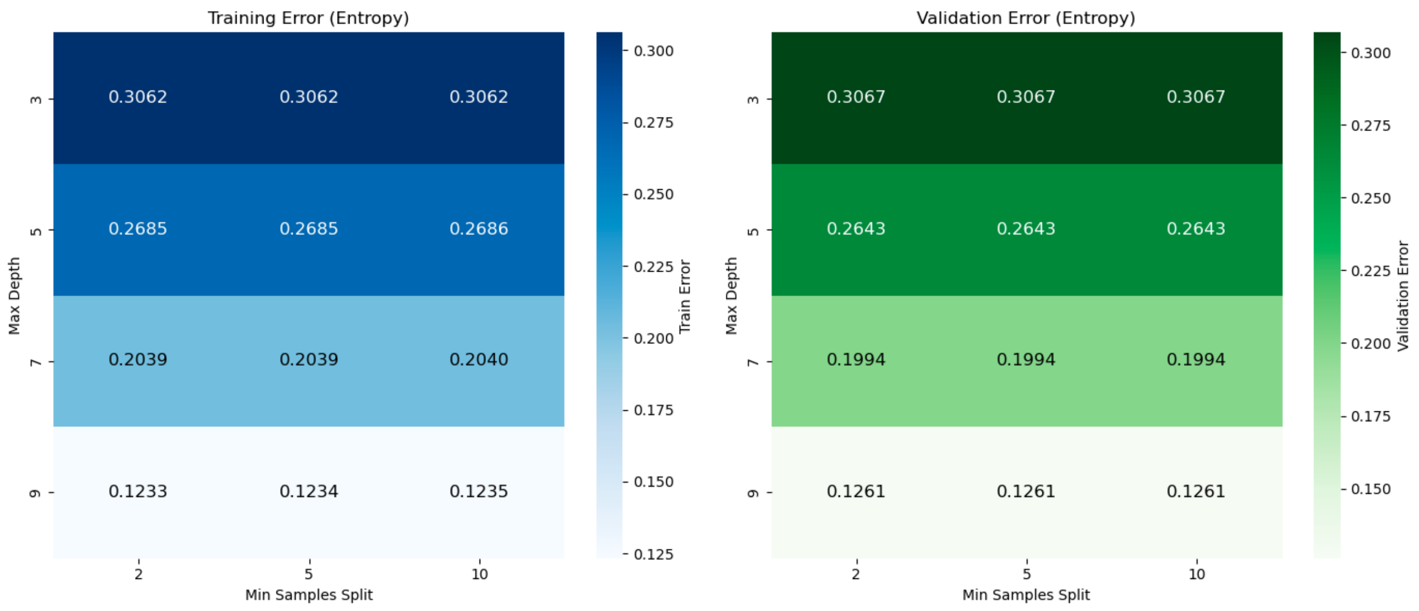
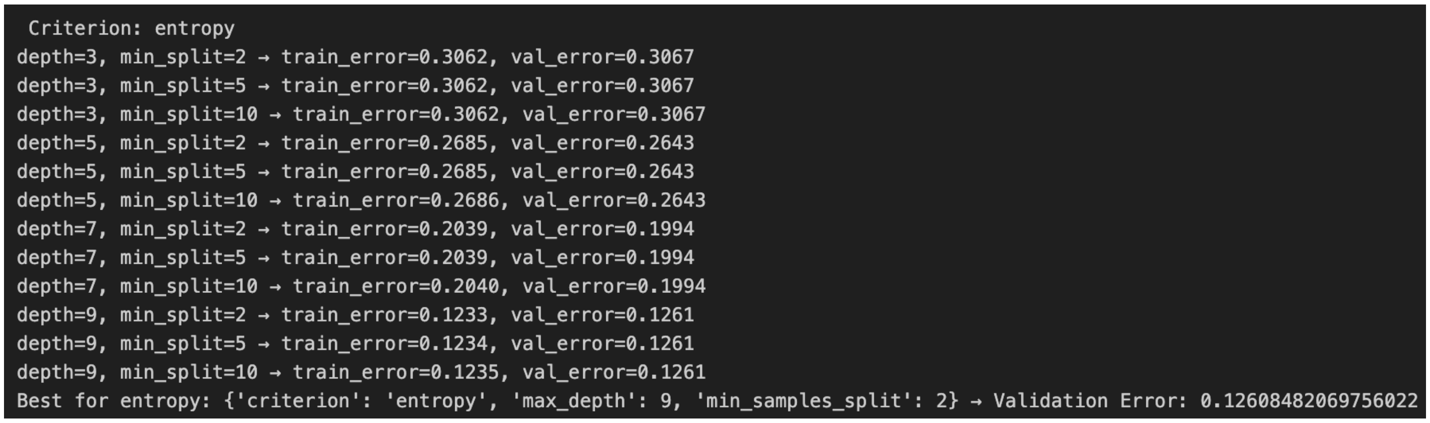


Figure 14. Training and Validation Error Heatmaps for Entropy Splitting Criterion



```
 Criterion: entropy
depth=3, min_split=2 → train_error=0.3062, val_error=0.3067
depth=3, min_split=5 → train_error=0.3062, val_error=0.3067
depth=3, min_split=10 → train_error=0.3062, val_error=0.3067
depth=5, min_split=2 → train_error=0.2685, val_error=0.2643
depth=5, min_split=5 → train_error=0.2685, val_error=0.2643
depth=5, min_split=10 → train_error=0.2686, val_error=0.2643
depth=7, min_split=2 → train_error=0.2039, val_error=0.1994
depth=7, min_split=5 → train_error=0.2039, val_error=0.1994
depth=7, min_split=10 → train_error=0.2040, val_error=0.1994
depth=9, min_split=2 → train_error=0.1233, val_error=0.1261
depth=9, min_split=5 → train_error=0.1234, val_error=0.1261
depth=9, min_split=10 → train_error=0.1235, val_error=0.1261
Best for entropy: {'criterion': 'entropy', 'max_depth': 9, 'min_samples_split': 2} → Validation Error: 0.12608482069756022
```

Table 3. Numerical Results of Training and Validation Errors for Entropy Splitting Criterion

The training and validation error heatmaps illustrate the performance of the decision tree classifier using the Entropy splitting criterion. As seen in the training error heatmap, increasing maximum tree depth reduced the training error from approximately 0.306 at depth 3 to around 0.123 at depth 9, indicating improved fit to the training data.

Validation error showed a similar decreasing trend, dropping from around 0.307 at depth 3 to about 0.126 at depth 9. This suggests that deeper trees enhanced the model’s ability to generalize, at least within the evaluated parameter range.

The minimum samples per split parameter had minimal impact on either training or validation errors, as results were consistent across all tested values. Overall, the Entropy splitting criterion performed reliably across depths, with the best configuration identified at a maximum depth of 9 and minimum samples split of 2, yielding a validation error of approximately 0.126.

Classification Error Splitting Criterion

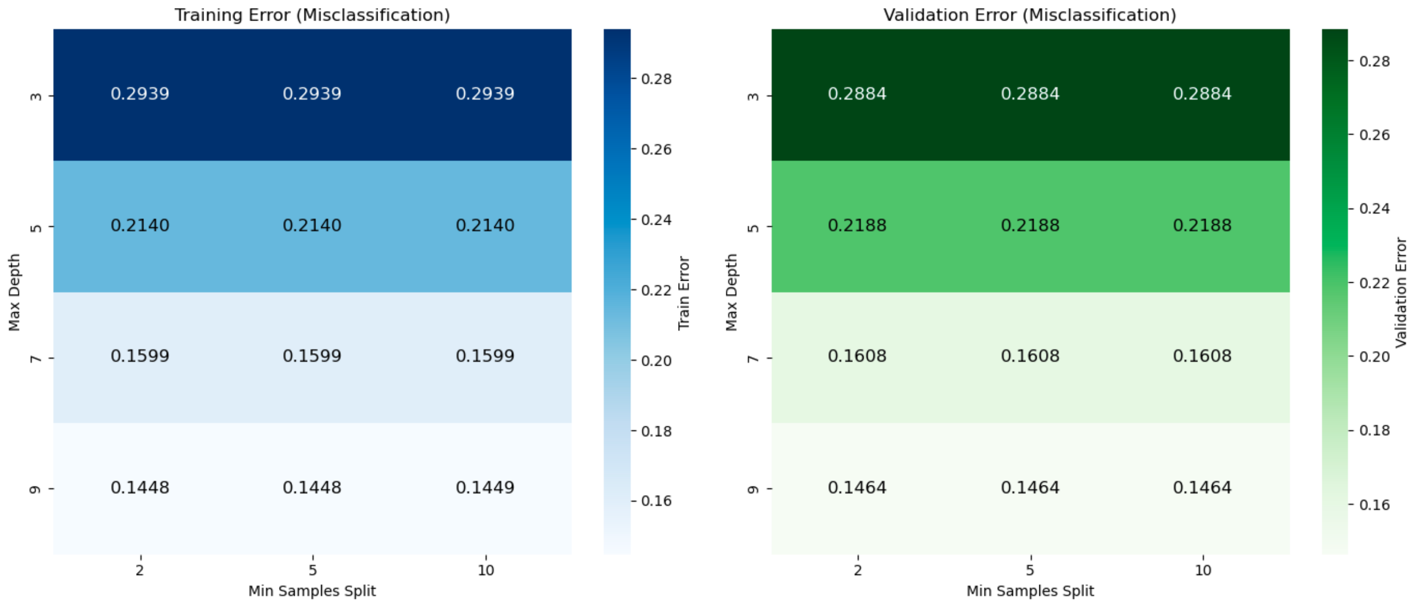
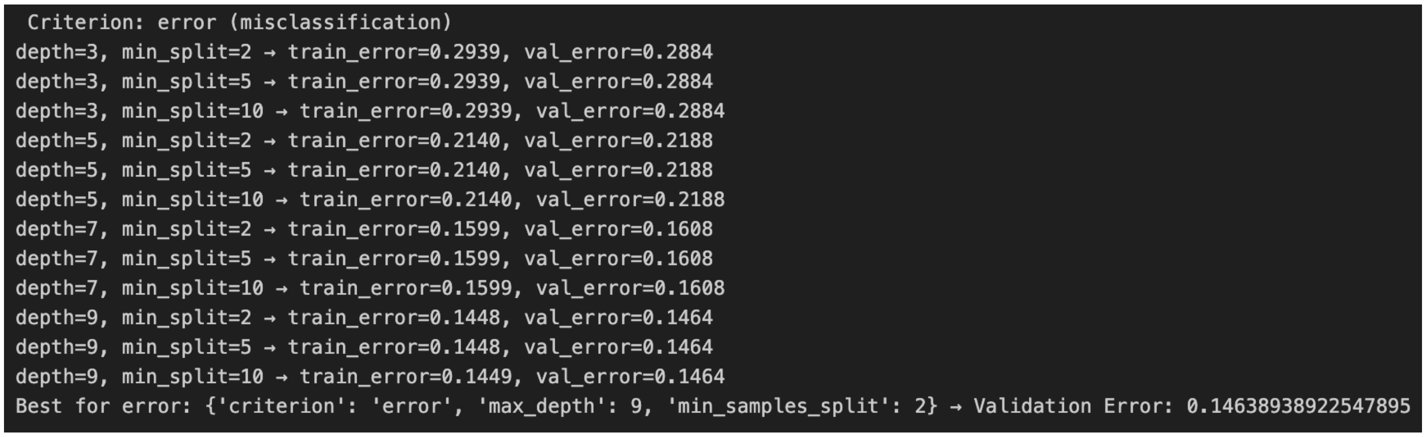


Figure 15. Training and Validation Error Heatmaps for Classification Error Splitting Criterion



```
 Criterion: error (misclassification)
depth=3, min_split=2 → train_error=0.2939, val_error=0.2884
depth=3, min_split=5 → train_error=0.2939, val_error=0.2884
depth=3, min_split=10 → train_error=0.2939, val_error=0.2884
depth=5, min_split=2 → train_error=0.2140, val_error=0.2188
depth=5, min_split=5 → train_error=0.2140, val_error=0.2188
depth=5, min_split=10 → train_error=0.2140, val_error=0.2188
depth=7, min_split=2 → train_error=0.1599, val_error=0.1608
depth=7, min_split=5 → train_error=0.1599, val_error=0.1608
depth=7, min_split=10 → train_error=0.1599, val_error=0.1608
depth=9, min_split=2 → train_error=0.1448, val_error=0.1464
depth=9, min_split=5 → train_error=0.1448, val_error=0.1464
depth=9, min_split=10 → train_error=0.1449, val_error=0.1464
Best for error: {'criterion': 'error', 'max_depth': 9, 'min_samples_split': 2} → Validation Error: 0.14638938922547895
```

Table 4. Numerical Results of Training and Validation Errors for Classification Error Splitting Criterion

The training and validation error heatmaps illustrate the performance of the decision tree classifier using the Classification Error (misclassification) splitting criterion. As observed in the training error heatmap, increasing the maximum tree depth consistently decreased the training error—from approximately 0.294 at depth 3 to around 0.145 at depth 9—indicating that deeper trees improved the model's fit to the training data.

A similar trend appeared in the validation error heatmap, with the error decreasing from around 0.288 at depth 3 to approximately 0.146 at depth 9. This pattern suggests that deeper trees enhanced the model's ability to generalize without significant overfitting within the tested parameter range.

The minimum samples per split parameter again had minimal impact on both training and validation errors, as indicated by the uniformity of values across different settings. Overall, the Classification Error criterion demonstrated reliable performance, with the best configuration achieved at a maximum depth of 9 and a minimum samples split of 2, yielding a validation error of approximately 0.146.

Final Model Evaluation

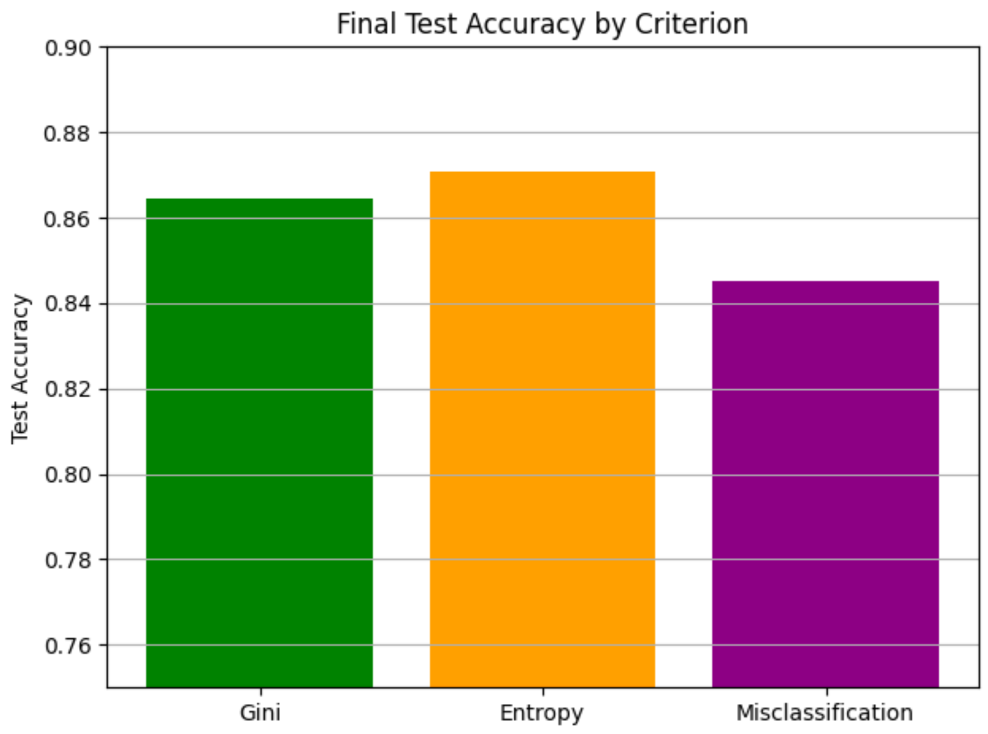


Figure 16. Comparison of Test Accuracy Across Splitting Criteria

The test results show that the Entropy criterion performs best (0.8708), slightly outperforming Gini (0.8646). This indicates that entropy-based splits capture class differences more effectively in this dataset.

The Misclassification Error criterion achieves the lowest accuracy (0.8451), reflecting its lower sensitivity to changes in class distribution.

Overall, entropy and Gini produce similar and stronger results, while misclassification error is less effective for building informative splits.

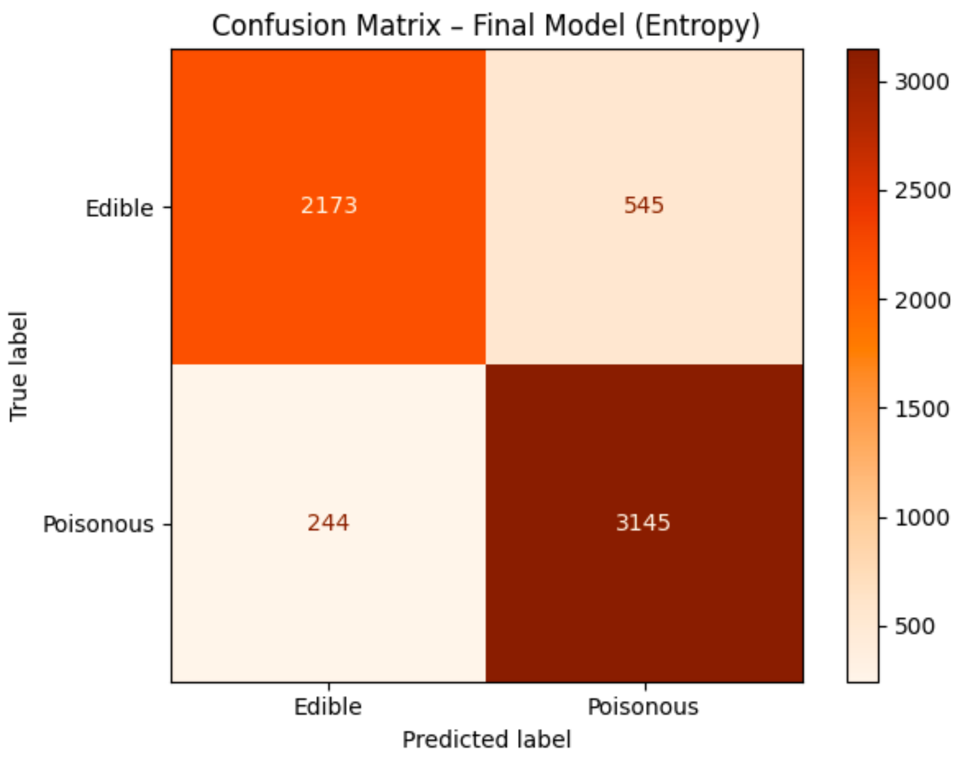


Figure 17. Confusion Matrix for Decision Tree Classifier (Entropy)

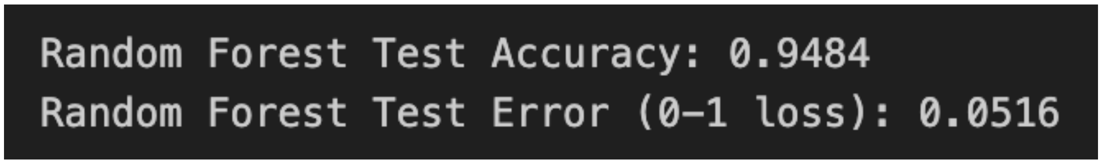
The confusion matrix for the final Decision Tree model trained with the Entropy criterion illustrates the model's performance in classifying edible and poisonous mushrooms on the test set.

- The model correctly identified 2173 edible mushrooms (true negatives) and 3145 poisonous mushrooms (true positives).
- 545 edible mushrooms were incorrectly classified as poisonous (false positives), while 244 poisonous mushroomswere misclassified as edible (false negatives).

Since incorrectly predicting poisonous mushrooms as edible is the more serious error, the relatively small number of such cases indicates that the model is generally reliable in identifying dangerous mushrooms. Overall, the results reflect strong and balanced classification performance.

Random Forest Performance

The Random Forest model, trained on the secondary Mushroom dataset using the Gini splitting criterion, was evaluated on the test set to assess its classification performance. The Gini criterion was chosen because it is computationally efficient and sensitive to class impurity, making it particularly effective for evaluating splits in categorical data such as the mushroom dataset. The results presented below highlight the model's test accuracy, error rate (0–1 loss), and a detailed classification report summarizing precision, recall, and F1-scores across both classes.



```
Random Forest Test Accuracy: 0.9484
Random Forest Test Error (0-1 loss): 0.0516
```

Table 8. Random Forest (Gini) — Test Accuracy and Classification Report

The Random Forest achieved an impressive test accuracy of 0.9484 with a test error (0–1 loss) of 0.0516, indicating a highly accurate classifier with minimal misclassification.

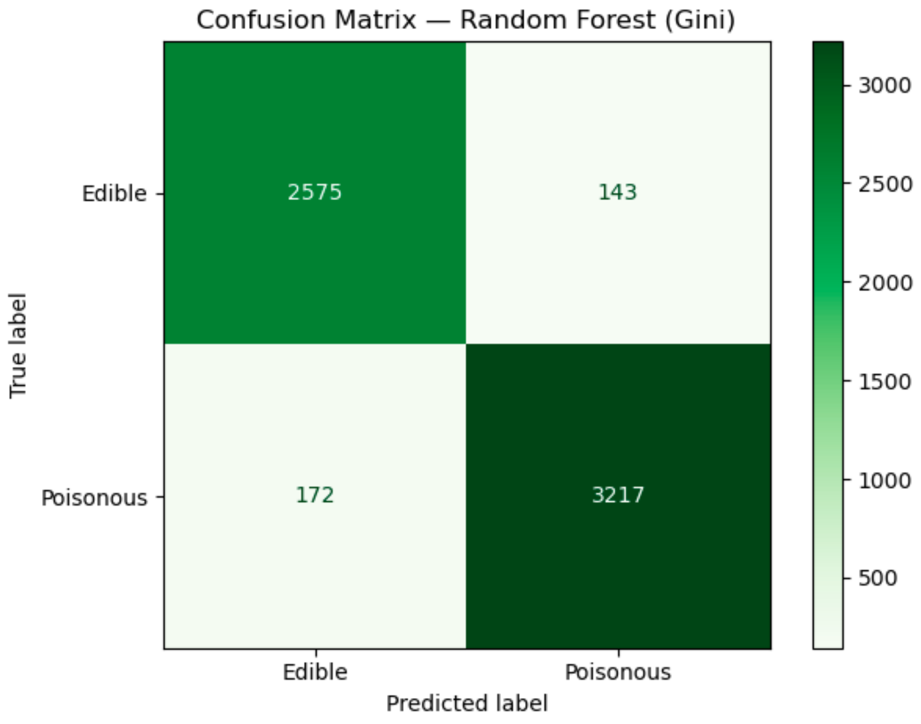


Figure 18. Confusion Matrix for Random Forest (Gini)

The confusion matrix demonstrates that the Random Forest model using the Gini splitting criterion performs well at distinguishing between edible and poisonous mushrooms. The model correctly classified 2575 edible mushrooms and 3217 poisonous mushrooms. However, there were 143 edible mushrooms incorrectly classified as poisonous (false positives) and 172 poisonous mushrooms incorrectly classified as edible (false negatives). These misclassifications highlight areas where the model could potentially be improved, such as fine-tuning hyperparameters or using additional features.

Overall, the high number of correct predictions indicates the model's strong ability to classify mushrooms accurately, consistent with the high accuracy reported earlier.

Model Comparison

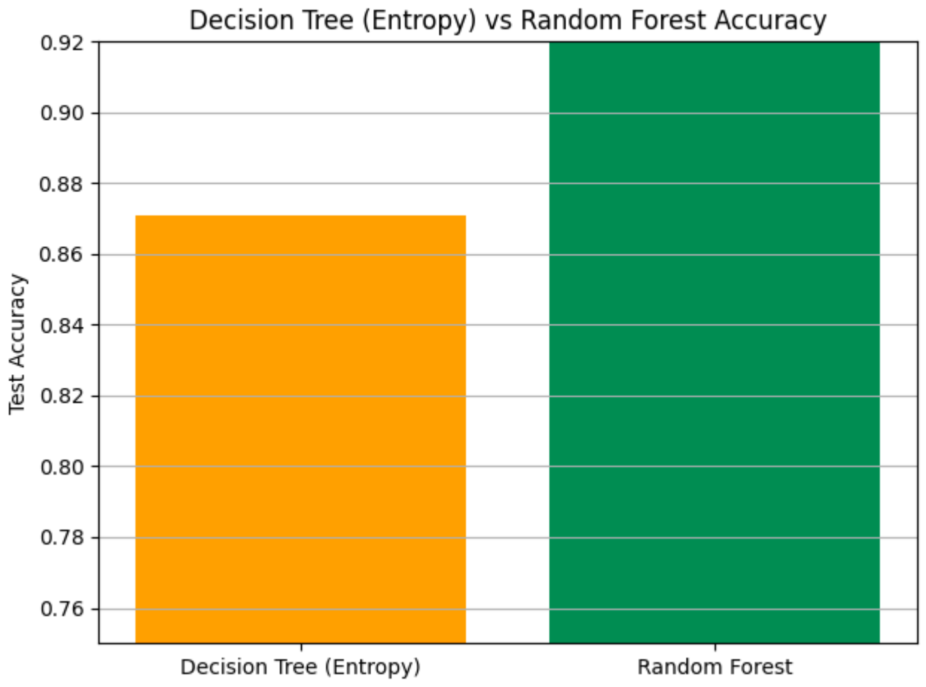
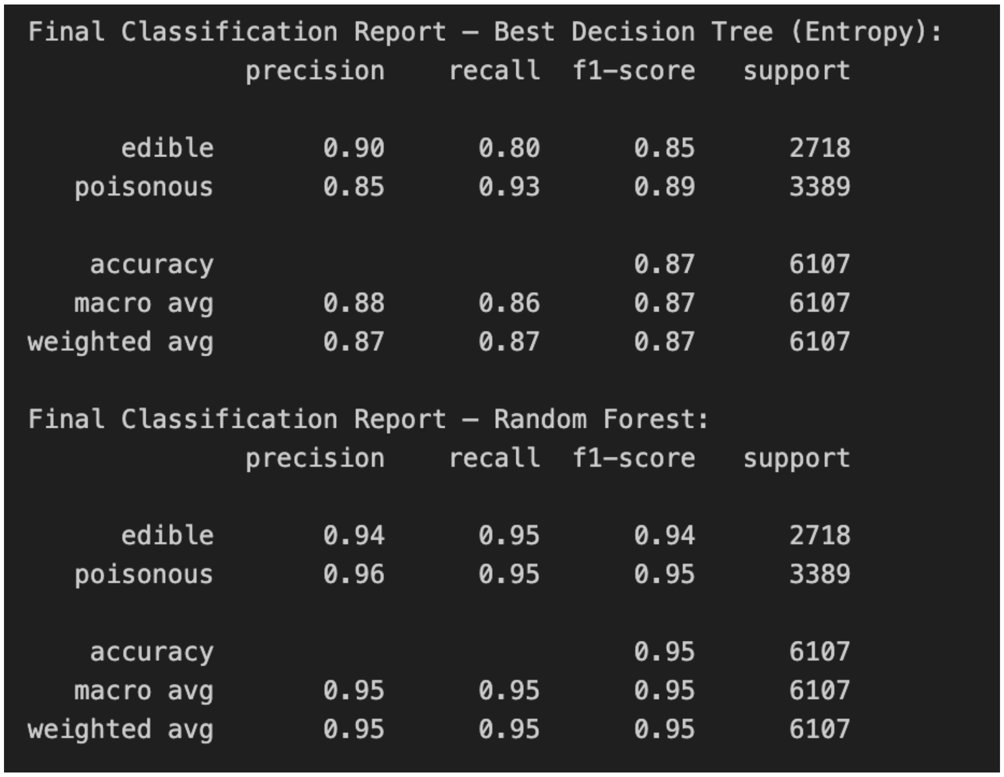


Figure 19. Model Comparison

The comparison shows that the Random Forest model achieves a noticeably higher test accuracy than the entropy-based decision tree. While the decision tree reaches about **0.87 accuracy**, the Random Forest exceeds **0.92**, demonstrating stronger overall performance. This gap indicates that combining multiple trees into an ensemble helps reduce variance and improves generalization, making the Random Forest more reliable for this classification task.



```
Final Classification Report — Best Decision Tree (Entropy):
              precision    recall  f1-score   support

      edible       0.90      0.80      0.85      2718
   poisonous       0.85      0.93      0.89      3389

    accuracy                           0.87      6107
   macro avg       0.88      0.86      0.87      6107
weighted avg       0.87      0.87      0.87      6107

Final Classification Report — Random Forest:
              precision    recall  f1-score   support

      edible       0.94      0.95      0.94      2718
   poisonous       0.96      0.95      0.95      3389

    accuracy                           0.95      6107
   macro avg       0.95      0.95      0.95      6107
weighted avg       0.95      0.95      0.95      6107
```

Table 9. Final Classification Reports

The entropy-based decision tree achieves an overall accuracy of **0.87**, with stronger performance on predicting poisonous mushrooms than edible ones. Its recall for the poisonous class (**0.93**) shows that it successfully identifies most dangerous mushrooms, but the lower recall for the edible class (**0.80**) indicates more edible mushrooms are misclassified as poisonous. This pattern suggests a conservative model that prioritizes safety but sacrifices some precision for edible predictions.

The Random Forest model substantially improves performance, reaching an accuracy of **0.95** and showing balanced, high values across all metrics. Both classes achieve **precision and recall around 0.94–0.96**, indicating consistent and reliable predictions. The improvement comes from the ensemble's ability to reduce variance and capture more complex patterns than a single decision tree.

Overall, the Random Forest demonstrates superior generalization and class balance, making it the more effective model for mushroom classification.

Discussion

This project evaluated custom-built decision tree and random forest classifiers for predicting mushroom edibility. The best decision tree model, using entropy as the splitting criterion, achieved a test accuracy of approximately 87%. The random forest demonstrated substantially stronger performance, reaching an accuracy of 95%, confirming the advantage of ensemble learning for this classification task. By combining predictions from multiple trees trained on different bootstrap samples, the random forest reduced variance and produced more stable, reliable predictions than any single decision tree.

Observations on Overfitting and Underfitting

Hyperparameter tuning revealed clear differences in how tree depth and minimum samples per split affect model behavior.

Shallow trees (e.g., max depth = 3) showed high validation error, reflecting underfitting, as they lacked the capacity to capture important feature interactions.

Deeper trees (e.g., max depth = 9) achieved much lower training error and only modest increases in validation error, indicating that, within the tested range, the deeper trees did not overfit severely.

Stopping criteria played an important role in preventing excessive tree growth and improving generalization. These controls balanced model complexity and helped avoid overfitting while preserving predictive accuracy.

Comparison of Different Splitting Criteria

Three splitting criteria were implemented: Gini impurity, entropy, and classification error. Their behavior was consistent with theoretical expectations:

- *Entropy* produced the best final accuracy on the test set in the decision tree model, making it slightly more effective for this dataset.
- *Gini impurity* showed stable performance across depths and was computationally more efficient, but achieved marginally lower accuracy.
- *Classification error* performed the worst, as expected, due to its insensitivity to class distribution details and smaller gradient during split optimization.

Overall, entropy achieved the strongest predictive performance, while Gini remained a robust and efficient alternative.

Effectiveness of Random Forest Compared to Decision Tree

The random forest clearly outperformed the standalone decision tree, with accuracy improving from 87% (entropy decision tree) to 95%.

This improvement is due to two core mechanisms, such as bootstrap sampling and random feature selection. Bootstrap sampling trains each tree on a slightly different subset of data. Random feature selection reduces correlation between trees and improves generalization.

These two elements reduce variance and help the model avoid overfitting to specific patterns in the training data. As a result, the random forest produces more stable and consistent predictions, making it a stronger classifier for this task.

Potential Future Work

Even though both models performed well, the project has several limitations. The results are based on a single train-validation-test split rather than k-fold cross-validation. No pruning was applied to the decision tree, which could improve generalization. No feature importance analysis was performed, limiting interpretability.

Conclusion

This project implemented and evaluated custom decision tree and random forest classifiers to predict mushroom edibility. After preprocessing the data, encoding categorical features, and tuning hyperparameters, the experiments showed that the best decision tree, using the entropy splitting criterion, achieved a test accuracy of about 87%. Although the decision tree performed well, the random forest model delivered a substantial improvement, reaching an accuracy of 95%. This result highlights the effectiveness of ensemble learning in reducing variance and improving overall predictive performance.

The experiments also showed that tree depth and stopping criteria strongly influence model behavior. Shallow trees tended to underfit the data, while deeper trees improved accuracy without significant overfitting, provided that constraints such as maximum depth and minimum samples per split were applied. These findings emphasize the importance of managing tree complexity to balance bias and variance.

Comparing different splitting criteria revealed that entropy produced the highest test accuracy for the decision tree in this dataset. Gini impurity offered stable and efficient performance, while classification error was less effective due to its lower sensitivity to class distribution differences.

Despite strong results, the project has several limitations. The evaluation relied on a single train–validation–test split rather than cross-validation. No pruning was applied to the decision tree, and feature importance analysis was not conducted, limiting interpretability. Future work should incorporate pruning methods, k-fold cross-validation, and interpretability tools such as feature importance.

Overall, the project successfully demonstrated the value of decision trees and random forests for a real-world classification task and provided insight into how hyperparameters, splitting criteria, and ensemble strategies affect model performance and reliability.