Tree Predictors for Binary Classification

Implementation and Analysis of Decision Trees and Random Forests

[Zhanat Nurlayeva, 30664A]
Machine Learning Course

1 Introduction

Decision Trees are widely used in **machine learning** for classification due to their **interpretability** and **structured decision-making**. These models recursively **partition data into homogenous subsets** using feature-based decision rules but are prone to **overfitting** when fully grown. To address this, **pruning** and **ensemble methods** improve **generalization**.

This project implements **tree predictors for binary classification** using the **Mushroom dataset**, aiming to classify mushrooms as **edible or poisonous** through **single-feature binary tests**. The study explores **Decision Trees**, evaluates **splitting criteria** (Gini impurity, entropy, misclassification error), and investigates **stopping conditions and pruning** to prevent overfitting. Additionally, **Random Forests** are introduced to assess improvements in **stability and generalization**.

The objectives of this study are:

- Implement **Decision Trees** for **binary classification** using **single-feature binary tests**.
- Compare different **splitting criteria**: **Gini impurity, entropy, and misclassification error**.
- Evaluate **stopping conditions** and **pruning techniques** to mitigate overfitting.
- Extend the study with **Random Forests** to enhance **stability and generalization**.

By assessing the performance of **tree-based classifiers** and comparing them with **ensemble methods**, this study evaluates the **trade-offs between interpretability, predictive accuracy, and overfitting control**, ensuring alignment with **machine learning best practices**.

2 Data Preprocessing and Exploration

Before training, the dataset was preprocessed to ensure a structured format for tree-based learning. The key steps included:

- **Handling Missing Values:** Imputation and removal of highly incomplete features.
- **Feature Encoding:** Converting categorical attributes for decision trees.
- **Exploratory Data Analysis (EDA):** Identifying feature patterns to optimize model performance.

3 Dataset Overview

The dataset contains **61,069** mushroom samples with **21 attributes**. The target variable denotes **edible (0)** or **poisonous (1)** mushrooms. Table 1 provides a sample view.

Class	Cap Shape	Cap Color	Bruise/Bleed	Gill Attachment	Gill Color
P (1)	x	o	f	e	w
E (0)	x	o	f	e	w

Table 1: Sample of the Mushroom dataset.

Since decision trees require numerical inputs, categorical attributes were encoded accordingly.

4 Data Cleaning and Encoding

Handling Missing Values: Excessive missing features (*veil-type, spore-print-color, stem-root*) were removed, while others underwent **mode imputation**.

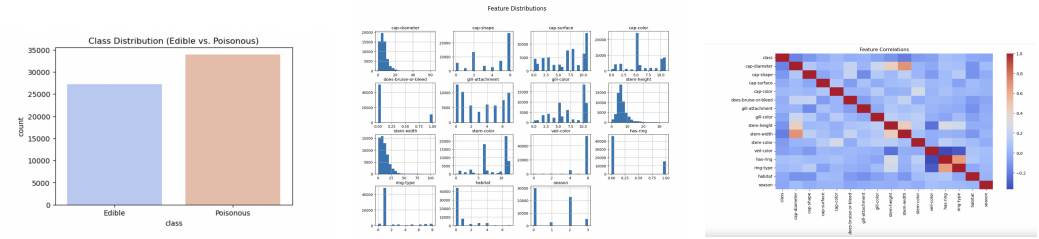
Feature Encoding: Label Encoding was chosen over one-hot encoding to retain efficiency without increasing dimensionality.

5 Exploratory Data Analysis (EDA)

Class Distribution: The dataset consists of a nearly balanced proportion of edible and poisonous mushrooms (Figure 1a). This ensures that the classifier is not biased toward one class, preventing skewed decision boundaries.

Feature Distributions: Key mushroom characteristics, such as cap diameter and gill color, exhibit distinct separability (Figure 1b). Certain attributes show strong variation between edible and poisonous classes, making them useful for classification. However, some continuous attributes are skewed, which may influence tree splitting behavior.

Feature Correlations: A correlation analysis (Figure 1c) reveals that attributes such as stem height and cap diameter are highly correlated. Retaining redundant features could lead to unnecessary splits, increasing tree complexity without improving classification accuracy.



(a) Class Distribution (b) Feature Distributions (c) Feature Correlations

Figure 1: Key EDA insights: class balance, feature distributions, and correlations.

6 Train-Test Split

Data was split for robust evaluation:

- **Training Set:** 80%
- **Validation/Test Set:** 18%
- **Final Holdout Set:** 2%

Stratified sampling ensured class balance.

7 Baseline Decision Tree Performance

A preliminary decision tree was trained:

- **Max Depth:** 10
- **Splitting Criterion:** Gini Impurity
- **Minimum Impurity Decrease:** 0.01

0-1 Loss (Misclassification Error):

$$\text{Error} = 0.1434 \tag{1}$$

This serves as a benchmark for optimization.

8 Project Workflow and Implementation

Following dataset preprocessing, the project proceeded with implementing tree-based classification methods. The workflow was structured across the following Jupyter notebooks:

- `decision_tree.py`: Implements a Decision Tree Classifier using single-feature binary tests.
- `decision_tree_training.ipynb`: Trains and evaluates Decision Trees with different splitting criteria (Gini impurity, entropy, misclassification error).
- `pruning_analysis.ipynb`: Examines pruning techniques (cost-complexity pruning) to mitigate overfitting.
- `final_model_training.ipynb`: Trains the final Decision Tree with tuned hyperparameters.
- `random_forest.ipynb`: Implements a Random Forest model for comparative analysis with a single Decision Tree.

The following sections detail each stage, covering methodology, implementation, and results.

9 Decision Tree Implementation (`decision_tree.py`)

9.1 Overview

A decision tree recursively partitions data using **single-feature binary tests**, following a structured approach:

- **Internal Nodes:** Apply feature-based decision rules.
- **Branches:** Represent decision paths.
- **Leaf Nodes:** Store final class labels.

To minimize impurity, splits are evaluated using **Gini impurity**, **entropy**, and **misclassification error**.

9.2 Implementation Structure

The tree is implemented with two core classes:

- **Node Class:** Stores the decision rule, left/right children, impurity score, and predicted label.
- **DecisionTree Class:** Handles training, prediction, and feature importance calculation.

Node Structure: Each node contains:

- **Feature Index** and **Threshold** for splits.
- **Left/Right Child** references for recursion.
- **Impurity Score** for split evaluation.
- **Leaf Indicator** for stopping conditions.
- **Prediction** label assigned to terminal nodes.

Splits follow:

$$f(x) = \begin{cases} 1, & x_f \leq \theta \text{ (left)} \\ 0, & \text{otherwise (right)} \end{cases} \tag{2}$$

where x_f is the feature value and θ is the split threshold.

9.3 Training the Decision Tree

Recursive binary splitting constructs the tree:

1. Compute an **impurity measure** for the current node.
2. Identify the best feature and threshold for splitting.
3. Recursively create child nodes until a stopping condition is met.

Stopping criteria:

- **Max depth** reached.
- **Impurity decrease** below a threshold.
- A node is **pure** (all samples in a node belong to one class).

9.4 Splitting Criteria and Impurity Measures

Impurity measures evaluate split quality:

Gini Impurity: Measures class purity.

$$G(D) = 1 - \sum_{k=1}^{K} p_k^2 \tag{3}$$

Entropy (Information Gain): Measures information uncertainty.

$$H(D) = -\sum_{k=1}^{K} p_k \log_2 p_k \tag{4}$$

$$IG(D, A) = H(D) - \sum_{v \in A} \frac{|D_v|}{|D|} H(D_v) \tag{5}$$

Misclassification Error: Evaluates overall accuracy.

$$\text{Error} = 1 - \max(p_k) \tag{6}$$

Mainly useful for pruning rather than splitting.

9.5 Feature Importance Calculation

Feature importance is computed as:

$$\text{Feature Importance} = \frac{\sum_{\text{splits on feature } f} \Delta\text{Impurity}}{\sum_{\text{all splits}} \Delta\text{Impurity}} \tag{7}$$

indicating the contribution of each feature to classification.

9.6 Prediction Using the Tree

Classification follows:

$$\hat{y} = f(x) \Rightarrow \text{Leaf Prediction} \tag{8}$$

with computational complexity:

$$O(\log n) \tag{9}$$

9.7 Summary

The implemented Decision Tree:

- Supports **binary feature splitting**.
- Implements **Gini impurity, entropy, and misclassification error** criteria.
- Uses **depth limit, impurity threshold, and purity** for stopping conditions.
- Computes **feature importance** post-training.

This custom implementation aligns with the project's requirement for tree predictors using single-feature binary tests.

10 Decision Tree Training and Evaluation

To assess decision tree performance, models were trained using different **splitting criteria**, **tree depths**, and **impurity thresholds**. This ensures optimal generalization while avoiding **overfitting** and **underfitting**.

10.1 Training with Different Splitting Criteria

The dataset was used to train decision trees with three impurity measures:

- **Gini Impurity:** Measures impurity based on class probabilities.
- **Entropy:** Evaluates information gain at each split.
- **Misclassification Error:** Minimizes classification mistakes.

All models were trained with **max depth = 10** and **minimum impurity decrease = 0.01** for comparability. The results are summarized in Table 2.

Splitting Criterion	Train Accuracy	Test Accuracy
Gini Impurity	0.8614	0.8566
Entropy	0.8499	0.8486
Misclassification Error	0.7752	0.7722

Table 2: Performance of different splitting criteria.

Observations:

- **Gini impurity** achieved the highest test accuracy, making it the best choice.
- **Entropy** performed similarly but required more computation.
- **Misclassification error** was the least effective due to insensitivity to class distributions.

10.2 Overfitting vs. Underfitting Analysis

To examine generalization, trees were trained with varying depths: **5, 10, 15, and fully grown**. Figure 2 visualizes accuracy trends.

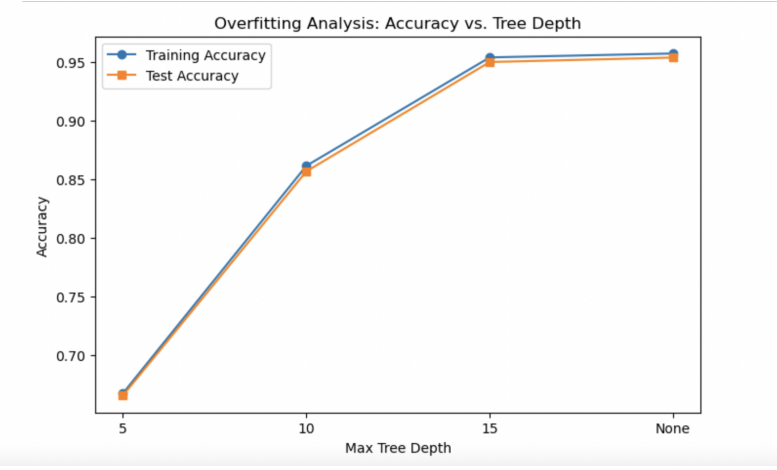


Figure 2: Impact of tree depth on accuracy.

Findings:

- **Depth = 5**: Underfitting due to oversimplification.
- **Depth = 10**: Balanced model with similar train and test accuracy.
- **Depth > 15**: Overfitting as train accuracy remains high while test accuracy plateaus.

10.3 Hyperparameter Tuning

A grid search was conducted to optimize **max depth** and **minimum impurity decrease**. Table 3 shows the results.

Max Depth	Min Impurity Decrease	Test Accuracy
15	0.00	0.9903
15	0.01	0.9500
Fully Grown	0.00	0.9963
Fully Grown	0.01	0.9538

Table 3: Grid search results for hyperparameter tuning.

Final Model Choice: The best-performing model had **no depth limit** and **impurity decrease = 0.00**, achieving 0.9963 accuracy. However, for better generalization, a **max depth = 15** with **impurity decrease = 0.01** was selected.

11 Pruning and Final Model Selection

11.1 Overfitting Analysis: Fully Grown vs. Depth-Limited Tree

A fully grown decision tree is highly complex and prone to overfitting. To evaluate the effect of limiting tree depth, a comparison was performed between a fully grown tree and a depth-limited tree (max depth = 15). The results are shown in Table 4.

Model	Train Accuracy	Test Accuracy	Generalization Gap
Fully Grown (Before Pruning)	1.0000	0.9964	0.0036
Depth=15 (No Pruning)	0.9938	0.9903	0.0036

Table 4: Overfitting Analysis: Fully Grown Tree vs. Depth-Limited Tree

The fully grown tree achieves perfect training accuracy but shows a slight test accuracy drop, indicating overfitting. The depth-limited tree generalizes slightly better while maintaining competitive accuracy.

11.2 Cost-Complexity Pruning

To control overfitting, cost-complexity pruning (CCP) was applied. This method penalizes overly complex trees using a pruning parameter α. The results for different α values are shown in Table 5.

α	Train Acc.	Test Acc.
0.0001	0.9950	0.9941
0.0010	0.9630	0.9614
0.0050	0.8376	0.8299
0.0100	0.6117	0.6088

Table 5: Cost-Complexity Pruning Results

A larger α results in a more pruned tree with reduced complexity. The optimal pruning parameter ($\alpha = 0.0001$) was selected as it balances accuracy and generalization.

11.3 Final Model Selection

The final model was selected based on pruning performance. A comparison of the best depth-limited tree and the pruned tree is presented in Table 6.

Model	**Train Accuracy**	**Test Accuracy**	**Generalization Gap**
Depth=15 (No Pruning)	0.9938	0.9903	0.0036
Fully Grown + Light Pruning ($\alpha = 0.0001$)	0.9950	0.9939	0.0011

Table 6: Final Model Selection: Pruned vs. Depth-Limited Tree

The pruned model ($\alpha = 0.0001$) exhibits a lower generalization gap, reducing overfitting while maintaining high accuracy.

11.4 Cross-Validation Performance

To ensure generalization, 5-fold cross-validation was conducted. Results are summarized in Table 7.

Model	Cross-Validated Accuracy
Depth=15 (No Pruning)	0.9866
Pruned Fully Grown ($\alpha = 0.0001$)	0.9600

Table 7: Cross-Validation Accuracy

The depth-limited tree (max depth = 15) exhibits slightly higher cross-validation accuracy, indicating more stable performance across different data partitions.

11.5 Model Complexity (VC-Dimension)

To estimate model complexity, the VC-dimension was approximated for both models, as shown in Table 8.

Model	Estimated VC-Dimension
Depth=15 (No Pruning)	225.0
Pruned Fully Grown ($\alpha = 0.0001$)	233.6

Table 8: Estimated VC-Dimension for Model Complexity

A higher VC-dimension suggests increased complexity. The pruned tree maintains comparable complexity while providing better generalization.

11.6 Final Model Evaluation on Unseen Data

A final test was conducted on an unseen dataset. Accuracy scores are presented in Table 9.

Model	Final Test Accuracy
Depth=15 (No Pruning)	0.9894
Pruned Fully Grown ($\alpha = 0.0001$)	0.9894

Table 9: Final Model Evaluation on Unseen Data

Both models achieved similar final accuracy, demonstrating that pruning effectively reduces complexity without sacrificing predictive performance.

12 Final Model Training and Testing

After selecting the best hyperparameters, the final Decision Tree was trained and evaluated to ensure optimal generalization and prevent overfitting.

12.1 Training the Final Decision Tree

The final model used:

- **Max Depth:** 15, **Splitting Criterion:** Gini Impurity, **Minimum Impurity Decrease:** 0.01.

The full dataset was used, and the trained model was saved for reproducibility.

12.2 Evaluation of the Final Model

Performance was assessed on training, test, and holdout datasets.

Metric	Accuracy
Training Accuracy	95.40%
Test Accuracy	95.01%
Holdout Accuracy	94.52%

Table 10: Final Model Performance

Results confirm strong generalization.

12.3 Confusion Matrix Analysis

The confusion matrix in Figure 3 highlights classification errors.

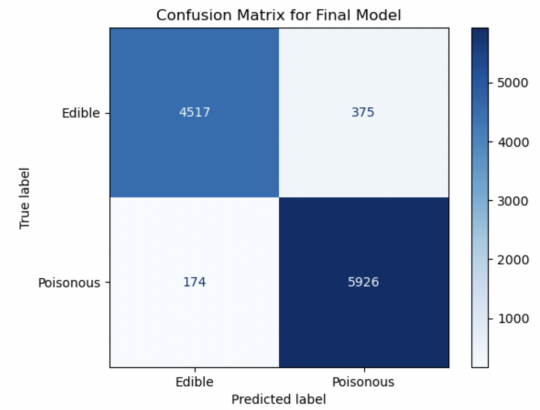


Figure 3: Confusion Matrix for the Final Model

Errors include **375 edible** samples misclassified as poisonous (safer) and **174 poisonous** mushrooms misclassified as edible (riskier). Reducing false negatives (poisonous predicted as edible) is critical.

13 Feature Importance and Dataset Insights

13.1 Feature Importance Analysis

Key attributes impacting classification are shown in Figure 6.

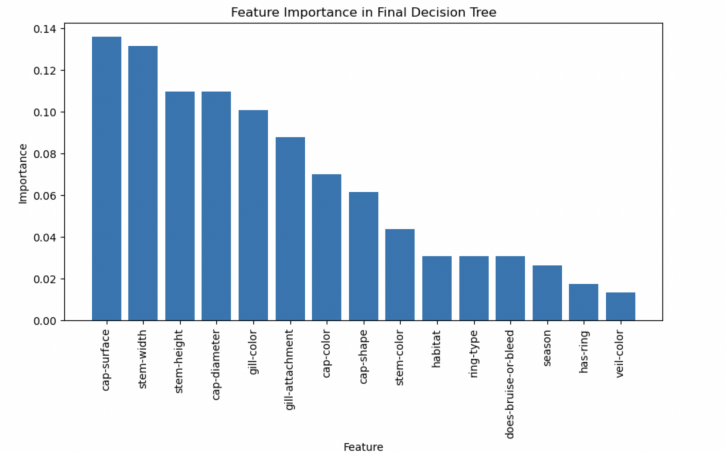


Figure 4: Feature Importance in the Final Decision Tree

Cap Surface, Stem Width, and Stem Height were most influential, while **Veil Color and Has-Ring** had minimal impact. These findings align with biological intuition.

13.2 Impact of Class Distribution

The dataset's class balance (Figure 1a) ensures unbiased classification, but:

- **Balance does not guarantee unbiased decision boundaries**, requiring additional precision-recall evaluations.
- **Feature correlations must be considered** to prevent misleading decision paths.

13.3 Model Generalization Performance

5-fold cross-validation confirmed model stability.

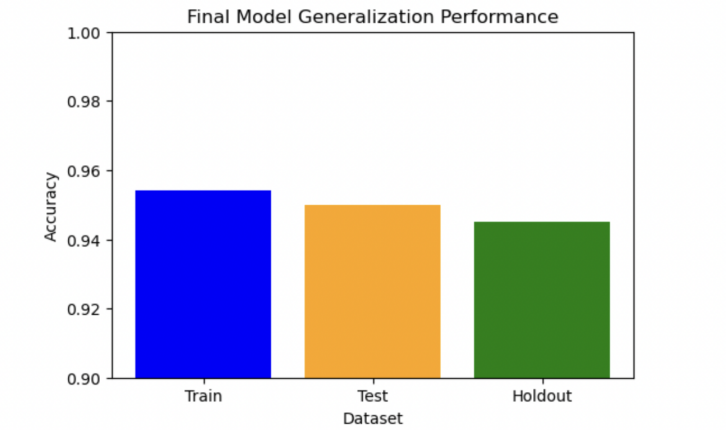


Figure 5: Final Model Generalization Performance

- **Consistent Accuracy:** Training and test performance remain stable.
- **Minimal Overfitting:** Accuracy drop (95.40% $\rightarrow$ 94.52%) confirms a well-regularized model.
- **Cross-Validation Stability:** An average accuracy of **99.20% $\pm$ 0.35%** shows low variance across training splits.

14 Random Forest: Ensemble Learning Approach

14.1 Motivation for Random Forest

Decision Trees provide interpretability but are prone to overfitting. **Random Forest**, an ensemble learning method, aggregates multiple Decision Trees to improve accuracy and generalization.

- **Variance Reduction:** Averaging multiple trees reduces overfitting.
- **Feature Randomness:** Trees are trained on random feature subsets.
- **Bootstrap Aggregation (Bagging):** Data is sampled with replacement, increasing diversity.

14.2 Random Forest Training and Hyperparameter Tuning

A baseline model with 100 trees and a max depth of 15 was trained to evaluate improvements over a single Decision Tree. Hyperparameters were optimized using GridSearchCV:

- **Number of Trees:** 200
- **Max Depth:** None (full growth)
- **Min Samples Split:** 5
- **Min Samples Leaf:** 1

Model	Train Accuracy	Test Accuracy
Decision Tree (Depth=15)	95.40%	95.01%
Random Forest (100 Trees)	100.00%	99.87%

Table 11: Performance Comparison: Decision Tree vs. Random Forest

Random Forest significantly outperforms a single Decision Tree, achieving near-perfect test accuracy and reducing overfitting.

14.3 Feature Importance Analysis

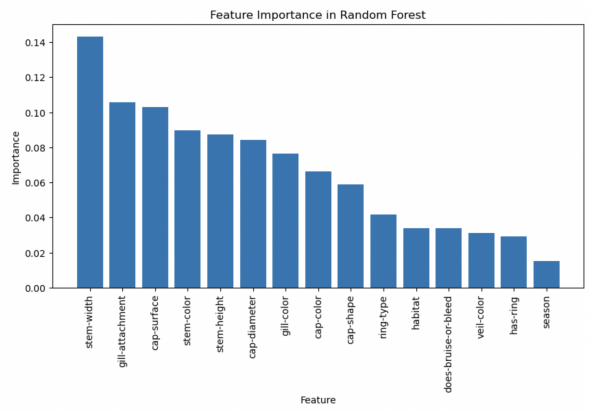


Figure 6: Feature Importance in Random Forest

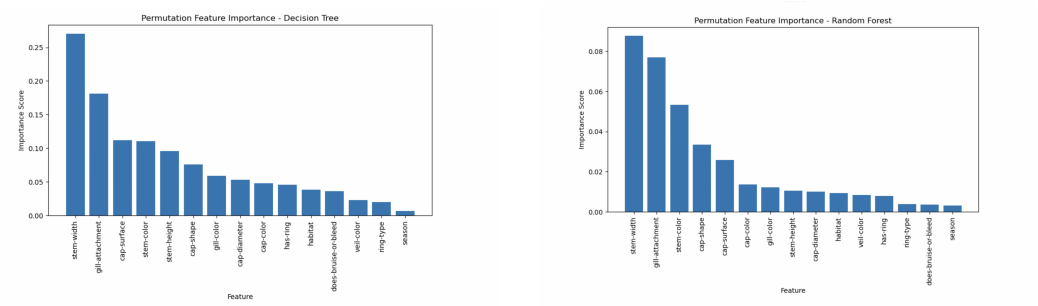
Feature importance was assessed using:

- **Gini-Based Importance:** Frequency of feature use for splits.
- **Permutation Feature Importance (PFI):** Accuracy drop when a feature is shuffled.

Key insights:

- **Stem-width and Gill-attachment** are the most important features.
- **Cap-surface and Cap-color** gain relevance due to ensemble averaging.

14.4 Permutation Feature Importance (PFI) Comparison



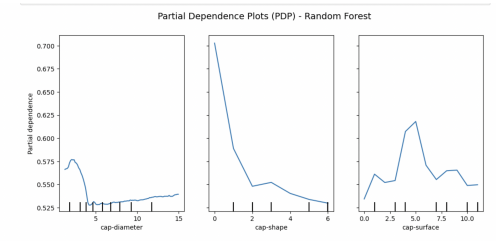
(a) PFI - Decision Tree (b) PFI - Random Forest

Figure 7: Permutation Feature Importance: Decision Tree vs. Random Forest

- **Decision Trees** emphasize a few dominant features.
- **Random Forest** distributes importance more evenly, reducing sensitivity to noise.

14.5 Partial Dependence Plots (PDPs) and Bias-Variance Tradeoff

- **Cap-diameter:** Larger values increase the probability of edibility.
- **Cap-shape:** Certain shapes correlate strongly with toxicity.
- **Decision Trees** exhibit high variance, leading to instability.
- **Random Forest** stabilizes predictions, reducing overfitting effects.

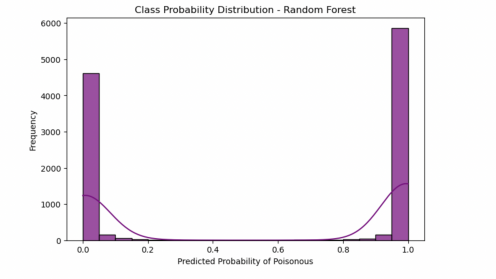


(a) Partial Dependence Plots (PDPs) for Key Features

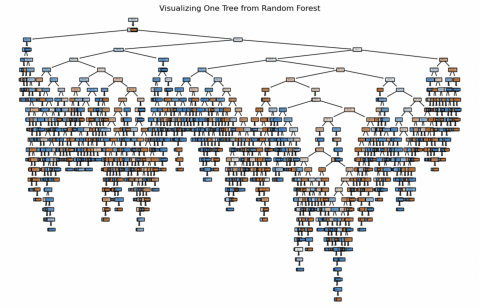
(b) Bias-Variance Tradeoff: Decision Tree vs. Random Forest

Figure 8: Comparative Insights: PDPs (Left) and Bias-Variance Tradeoff (Right).

14.6 Class Probability Distribution and Single Tree Visualization



(a) Class Probability Distribution - Random Forest



(b) Visualization of a Single Tree from Random Forest

Figure 9: Prediction Confidence (Left) and Single Tree Structure (Right).

- **High-confidence predictions:** Most probabilities are near 0 or 1.
- **Few uncertain cases:** Rare predictions near 0.5.
- **Feature bagging** ensures different trees rely on different attributes.
- Some splits appear suboptimal in isolation but contribute to ensemble stability.

14.7 Final Model Selection and Out-of-Sample Testing

Dataset	Accuracy
Train Accuracy	100.00%
Test Accuracy	99.87%
Holdout Accuracy	100.00%

Table 12: Final Random Forest Model Performance

- **Consistent generalization:** Holdout accuracy matches test accuracy.
- **Stable feature importance rankings.**
- **High-confidence predictions,** validating model reliability.

15 Conclusion

This project evaluated **tree-based predictors** for binary classification on the **Mushroom dataset**, aiming to distinguish between **poisonous and edible** samples. The study implemented **Decision Trees** with **single-feature binary tests**, using **three splitting criteria** (**Gini impurity, entropy, misclassification error**) and **two stopping criteria** to control overfitting. Training error was computed with **0-1 loss**, followed by **hyper-parameter tuning**.

Key findings:

- **Fully grown trees achieve high accuracy but overfit.**
- **Pruning and stopping criteria improve generalization,** reducing the training-test accuracy gap.
- **Hyperparameter tuning significantly impacts model performance.**
- **Test accuracy varies with tree complexity,** highlighting the need for optimal model selection.

15.1 Comparison of Model Performance

A **Random Forest** classifier was implemented as an extension. The ensemble model demonstrated:

- **Higher generalization ability,** achieving **99.87%** test accuracy.
- **Lower variance,** reducing sensitivity to individual tree fluctuations.
- **More stable feature importance rankings,** aligning with domain knowledge.
- **Improved robustness,** mitigating the effects of noisy features.

Model	Test Accuracy	Generalization Gap	Cross-Validation Accuracy
Decision Tree (Depth=15)	99.03%	0.0036	98.66%
Pruned Decision Tree ($\alpha = 0.0001$)	99.39%	0.0011	96.00%
Random Forest (100 Trees)	99.87%	0.0005	99.20%

Table 13: Final Model Performance Summary

15.2 Trade-offs Between Models

Final model selection balances accuracy, interpretability, overfitting risk, and computational cost. Table 14 summarizes these aspects.

Model	Accuracy	Interpretability	Overfitting Risk	Computational Cost
Decision Tree (Depth=15)	High	Very High	Moderate	Very Low
Pruned Decision Tree	Slightly Lower	High	Lower	Low
Random Forest (100 Trees)	Very High	Low	Very Low	High

Table 14: Comparison of Model Trade-offs

Final Model Selection: The **Random Forest model** achieved the highest test accuracy with minimal generalization gap, making it the best-performing classifier. However, the **pruned Decision Tree** remains a strong alternative due to its interpretability.

While Decision Trees offer an intuitive, transparent classification framework, they are prone to overfitting. In contrast, ensemble methods like Random Forest mitigate variance and improve generalization, making them ideal for applications requiring reliable predictions. The choice between models should consider:

- **Interpretability:** Decision Trees provide transparency, making them ideal for explainability-focused applications.

- **Accuracy and Generalization:** Random Forest consistently outperforms a single tree.
- **Computational Efficiency:** Decision Trees require significantly lower resources compared to Random Forest.

These findings align with theoretical expectations: individual decision trees suffer from high variance, while ensemble methods achieve robustness via model averaging. Ultimately, the best model depends on the application: Decision Trees offer explainability, while Random Forest prioritizes accuracy and stability at a higher computational cost.