Machine Learning Report: Tree predictors for binary classification

Tesi Zaçe

May 2025

Abstract

This report evaluates decision tree classifiers on the Secondary Mushroom Dataset for binary classification of mushroom edibility. We compare four splitting criteria and analyze stopping criteria effects on model performance. The methodology employs data splitting, proper categorical encoding, and randomized hyperparameter search. Performance analysis highlights the trade-offs between underfitting and overfitting, supported by validation and test errors. We discuss methodological choices and potential improvements, such as pruning and stopping criteria, to enhance model robustness.

Contents

1 Introduction

Identifying whether mushrooms are safe to eat is extremely important, as consuming poisonous mushrooms can be dangerous or even fatal. Developing an automated system to classify mushrooms as edible or poisonous represents a valuable real-world application of machine learning. This study focuses on building a binary classifier that predicts mushroom edibility based on observable physical features such as cap shape, color, and habitat. Our objective is to understand how different decision tree configurations affect classification performance.

1.1 Overview of Decision Trees in Classification

Decision trees [1] are machine learning models that make predictions by asking a sequence of questions about input features. They resemble a flowchart structure, beginning at a root node and traversing branches based on feature values until a leaf node is reached, which produces the final prediction. Decision trees are favored for their interpretability and natural compatibility with categorical data.

During the tree-building process, various splitting criteria are used to determine which features and thresholds provide the best data separation. Common approaches include Gini impurity, entropy, and information gain ratio. Despite their strengths, decision trees are prone to overfitting [4] , which reduces generalization to unseen data. To mitigate this, stopping criteria such as maximum tree depth, minimum number of samples per split, or a minimum decrease in impurity are applied. Striking the right balance between tree complexity and model performance requires careful hyperparameter tuning.

1.2 Dataset Description and Preprocessing Steps

For this project, we utilize the Secondary Mushroom Dataset from the UCI Machine Learning Repository. This dataset includes 61,069 mushroom samples, each described by 20 categorical features related to physical characteristics, such as cap shape, gill color, stem properties, and habitat. Each sample is labeled as either edible ('e') or poisonous ('p'). For modeling purposes, these labels are converted to binary form: 0 for edible and 1 for poisonous.

The dataset is preprocessed to ensure quality and consistency. Initially, we standardize column names, replace missing or empty values, and encode all categorical variables using label encoding. The data is then divided into three subsets: 64% for training (39,084 samples), 16% for validation (9,771 samples), and 20% for testing (12,214 samples). This splitting maintains the original distribution of class labels across all subsets. To prevent data leakage, label encoders are fit using only the training data and unseen categories in validation or test sets are handled appropriately. All processed datasets and encoders are saved to allow reproducibility of the experiments.

2 Methodology

2.1 Experimental Setup and Data Splits

The experimental framework follows a 5-step approach starting with the dataset preprocessing (Step 1) which employs stratified sampling to maintain class distribution across all splits:

$$\text{Training set:} \quad D_{train} = 64\% \times 61,069 = 39,084 \text{ samples} \tag{1}$$

$$\text{Validation set:} \quad D_{val} = 16\% \times 61,069 = 9,771 \text{ samples} \tag{2}$$

$$\text{Test set:} \quad D_{test} = 20\% \times 61,069 = 12,214 \text{ samples} \tag{3}$$

Stratification ensures that the proportion of edible ($y = 0$) and poisonous ($y = 1$) mushrooms remains consistent across all splits, preventing sampling bias. Label encoding is applied to all categorical features with safe handling for unseen categories in validation and test sets to avoid data leakage.

2.2 DecisionTreeClassifier and Key Hyperparameters

Our custom `DecisionTreeClassifier` implementation supports multiple splitting criteria and stopping mechanisms. The tree construction algorithm recursively partitions the feature space based on the selected impurity measure.

2.2.1 Splitting Criteria

Four splitting criteria are evaluated (Step 2) to determine how the decision tree selects the best feature and threshold for each split. Each criterion measures node impurity differently, affecting tree structure and performance:

1. **Gini Impurity [2]:** Measures how often a randomly chosen sample would be incorrectly classified. For our mushroom dataset, if a node contains 70% edible and 30% poisonous mushrooms, Gini $= 1 - (0.7^2 + 0.3^2) = 0.42$.

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2 \tag{4}$$

   where p_i is the proportion of samples belonging to class i in set S. Lower values indicate purer nodes.

2. **Entropy [2]:** Measures information content and uncertainty in a node. Pure nodes (all edible or all poisonous) have entropy = 0, while evenly mixed nodes have maximum entropy.

$$\text{Entropy}(S) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{5}$$

   Entropy tends to create more balanced trees than Gini, potentially improving generalization for our mushroom classification task.

3. **Misclassification Rate [3]:** Simply counts the fraction of minority class samples in a node. This is the most intuitive but least sensitive metric for split selection.

$$\text{Misclassification}(S) = 1 - \max_i(p_i) \tag{6}$$

4. **Information Gain Ratio [2]:** Normalizes information gain by split information to avoid bias toward features with many categories. This is particularly relevant for our categorical mushroom features like gill color or cap shape.

$$\text{InfoGainRatio}(S, A) = \frac{\text{InfoGain}(S, A)}{\text{SplitInfo}(S, A)} \tag{7}$$

   where $\text{SplitInfo}(S, A) = -\sum_{j=1}^{v} \frac{|S_j|}{|S|} \log_2\left(\frac{|S_j|}{|S|}\right)$

Our Step 2 implementation compares these criteria on identical tree configurations to isolate their impact on mushroom classification performance.

2.2.2 Stopping Criteria

Step 3 analyzes four key stopping criteria to prevent overfitting, which is crucial for mushroom classification where false negatives (classifying poisonous as edible) could be dangerous:

- **Maximum Depth [4]** (`max_depth`): Controls how many questions the tree can ask before making a decision. Shallow trees (depth 3-5) may underfit but generalize well, while deep trees (depth 15+) may memorize training examples. Our analysis plots training vs. validation error across different depths to find the optimal complexity.

- **Minimum Samples Split** [4] (`min_samples_split`): Prevents splitting nodes with too few samples, reducing noise-driven decisions. For our 39,084 training samples, requiring at least 10-20 samples per split helps ensure statistical significance of each decision.
- **Minimum Samples Leaf** [4] (`min_samples_leaf`): Ensures leaf nodes represent enough examples for reliable predictions. This is particularly important for mushroom safety, as predictions based on very few examples may be unreliable.
- **Minimum Impurity Decrease** [4] (`min_impurity_decrease`): Only allows splits that substantially improve node purity. This prevents the tree from making splits that provide minimal information gain, focusing on the most distinctive mushroom features.

Step 3 implementation creates validation curves for each criterion, plotting both training and validation error to identify the optimal balance between model complexity and generalization for mushroom classification.

2.3 Randomized Hyperparameter Search

Step 4 implements randomized search using `ParameterSampler` to efficiently explore the hyperparameter space. Given our large parameter grid with over 2,000 possible combinations, randomized search provides a practical alternative to exhaustive grid search while maintaining good parameter space coverage.

The parameter grid targets the most influential hyperparameters for mushroom classification:

```
param_grid = {
    'criterion': ['gini', 'entropy', 'info_gain_ratio'],
    'max_depth': [3, 5, 7, 10, 15, 20, None],
    'min_samples_split': [2, 5, 10, 20, 50],
    'min_samples_leaf': [1, 2, 5, 10, 20],
    'min_impurity_decrease': [0.0, 0.01, 0.05, 0.1]
}
```

With 39,084 training samples, minimum sample values from 1-50 provide meaningful constraints, while depth values from 3-20 span the range from simple to complex models. The impurity decrease thresholds (0.0-0.1) allow evaluation from no constraint to strict purity requirements.

Randomized search samples n parameter combinations uniformly from this grid and for each sampled configuration, we train on the training set and evaluate on the validation set to select optimal hyperparameters without touching the test set.

$$P(\text{config}_i) = \frac{1}{|\text{total configurations}|} \tag{8}$$

This approach prevents test set contamination while efficiently identifying hyperparameters that generalize well to unseen mushroom samples.

2.4 Evaluation Metrics

Three primary metrics assess model performance, each providing different insights into mushroom classification accuracy:

2.4.1 Zero-One Loss

The fraction of misclassified samples, directly measuring classification errors:

$$L_{0-1}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[y_i \neq \hat{y}_i] \tag{9}$$

where $\mathbf{1}[\cdot]$ is the indicator function. For mushroom classification, this metric is particularly important as any misclassification could have serious consequences. A zero-one loss of 0.05 means 5% of mushrooms are incorrectly classified.

2.4.2 Accuracy

The complement of zero-one loss, representing the fraction of correct predictions:

$$\text{Accuracy} = 1 - L_{0-1}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[y_i = \hat{y}_i] \tag{10}$$

While accuracy provides an intuitive performance measure, it may be misleading if our mushroom dataset is imbalanced between edible and poisonous samples.

2.4.3 Confusion Matrix

For binary mushroom classification, the confusion matrix provides detailed performance breakdown:

$$\text{Confusion Matrix} = \begin{pmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{pmatrix} \tag{11}$$

where:

- **True Negatives (TN):** Correctly identified edible mushrooms
- **False Positives (FP):** Edible mushrooms incorrectly classified as poisonous
- **False Negatives (FN):** Poisonous mushrooms incorrectly classified as edible (most dangerous error)
- **True Positives (TP):** Correctly identified poisonous mushrooms

The confusion matrix [3] reveals whether our model has systematic biases and helps identify the most critical error types for mushroom safety applications.

Step 5 conducts final model evaluation by comparing multiple optimized configurations across all data splits, ensuring robust performance assessment and model selection based on validation error minimization.

3 Hyperparameter Tuning

The hyperparameter [3] search space was designed to systematically explore the trade-off between model complexity and generalization for mushroom classification. Each parameter range was selected based on dataset characteristics and theoretical considerations:

- **Splitting Criterion**: `['gini', 'entropy', 'info_gain_ratio']` - Three fundamentally different impurity measures to evaluate their effectiveness on categorical mushroom features.
- **Maximum Depth**: `[None, 20, 15, 12, 8]` - Ranging from unrestricted growth to shallow trees. Given our 39,084 training samples, depths of 8-20 provide meaningful complexity control without severe underfitting.
- **Minimum Samples Split**: `[2, 4, 8, 16, 32]` - Conservative values to prevent overfitting on small subsets. With binary mushroom classification, requiring 2-32 samples ensures statistical significance.
- **Minimum Samples Leaf**: `[1, 2, 4, 8]` - Controls leaf node reliability. Values up to 8 maintain prediction confidence while allowing sufficient granularity for mushroom feature combinations.
- **Minimum Impurity Decrease**: `[0.0, 0.001, 0.01, 0.05]` - Prevents splits with minimal information gain. The range spans from no constraint (0.0) to strict requirements (0.05) for meaningful splits.

3.1 Tuning Procedure and Number of Iterations

The hyperparameter tuning employed exhaustive grid search across all parameter combinations, testing 1,680 unique configurations. The procedure followed these steps:

1. **Configuration Generation**: All possible combinations of hyperparameters were systematically generated from the parameter grid.
2. **Model Training**: For each configuration, a decision tree was trained on the training set (39,084 samples) using the specified hyperparameters.
3. **Validation Evaluation**: Each trained model was evaluated on the validation set (9,771 samples) using zero-one loss to assess generalization performance.
4. **Performance Tracking**: Training and validation errors were recorded for each configuration to identify overfitting patterns.

The tuning process demonstrated remarkable stability with zero failed configurations out of 1,680 tested combinations. This robustness indicates appropriate parameter ranges that prevent extreme cases, reliable custom DecisionTreeClassifier implementation, proper error handling during tree building and sufficient computing resources for all tests.

The absence of failed configurations ensures complete coverage of the intended hyperparameter space without bias from missing data points.

3.2 Best Found Hyperparameters

The optimal configuration achieved a validation error of 0.001126 (99.89% accuracy) with the following hyperparameters:

Table 1: Optimized Hyperparameter Values

Hyperparameter	**Optimal Value**
Splitting Criterion	Entropy
Maximum Depth	20
Minimum Samples Split	2
Minimum Samples Leaf	1
Minimum Impurity Decrease	0.01

3.2.1 Hyperparameter Analysis

Statistical analysis across all configurations reveals the importance of each hyperparameter:

- **Maximum Depth**: Unrestricted depth (None) achieved the best average performance (0.177 ± 0.198), followed closely by depth 20 (0.209 ± 0.206). Shallow trees (depth 8) showed significantly worse performance (0.310 ± 0.112).
- **Splitting Criterion**: Gini impurity demonstrated superior average performance (0.200 ± 0.177) compared to entropy (0.212 ± 0.174). Information gain ratio performed poorly (0.393 ± 0.067), likely due to bias issues with categorical features.
- **Minimum Samples Leaf**: Lower values consistently outperformed higher constraints, with min_samples_leaf=1 achieving 0.257 ± 0.177 versus 0.273 ± 0.171 for min_samples_leaf=8.
- **Minimum Impurity Decrease**: Moderate constraints (0.01) provided optimal performance, while strict thresholds (0.05) severely degraded accuracy (0.430 ± 0.026).

3.3 Performance Trade-off Analysis

Evaluation of model complexity revealed clear overfitting/underfitting patterns:

Table 2: Model Depth Configuration and Performance

Configuration	Train Error	Val Error	Gap	Interpretation
Very Shallow	0.348	0.344	-0.004	Underfitting
Shallow	0.270	0.264	-0.006	Underfitting
Moderate	0.098	0.099	+0.001	Good Fit
Deep	0.001	0.002	+0.001	Good Fit

The analysis indicates that moderate to deep configurations achieve optimal performance without significant overfitting, supporting the selection of the depth-20 optimal configuration.

4 Performance Analysis and Trade-offs

4.1 Splitting Criteria Evaluation

Our initial analysis compared four different splitting criteria to establish performance characteristics. The results demonstrate significant variation in model effectiveness across different impurity measures.

Table 3: Splitting Criteria Performance Comparison

Criterion	Train Acc	Val Acc	Train Err	Val Err
Gini	0.9023	0.9012	0.0977	0.0988
Entropy	0.8627	0.8642	0.1373	0.1358
Misclassification	0.8414	0.8467	0.1586	0.1533
Info Gain Ratio	0.5935	0.5932	0.4065	0.4068

The Gini impurity criterion demonstrates superior performance with training and validation accuracies exceeding 90%. The small gap between training error (0.0977) and validation error (0.0988) indicates good generalization without significant overfitting. The entropy criterion shows moderate performance with a slightly inverted error relationship (validation error ¡ training error), suggesting potential underfitting in the baseline configuration. The Information Gain Ratio criterion exhibits poor performance with approximately 40% error rates, indicating inadequate model complexity for this dataset. This suggests that the normalization inherent in the gain ratio may be overly conservative for this particular classification task.

4.2 Hyperparameter Impact Analysis

4.2.1 Maximum Depth Effects

The evaluation of tree depth reveals critical insights into the bias-variance trade-off:

Table 4: Maximum Depth Impact on Model Performance

Max Depth	Training Error	Validation Error
5	0.2698	0.2640
10	0.0975	0.0989
15	0.0048	0.0067
20	0.0002	0.0013

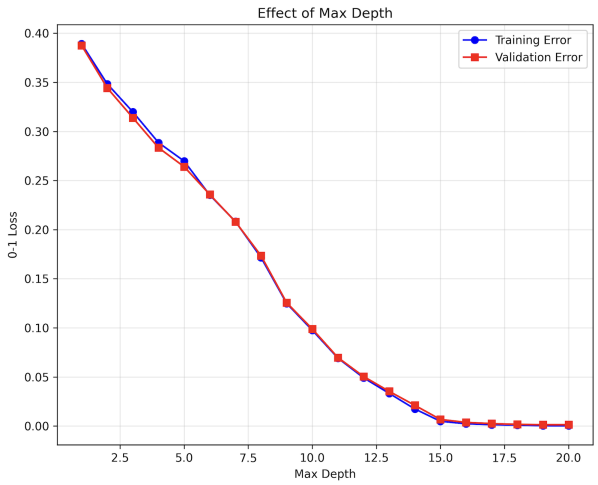

Figure 1: Training and Validation Error vs Maximum Depth

The depth analysis reveals a clear progression from underfitting to optimal fitting. At depth 5, both training and validation errors remain high (≈26%), indicating insufficient model complexity. The transition from depth 10 to 15 shows dramatic improvement, with validation error decreasing from 9.89% to 0.67%. However, at depth 20, we observe the emergence of overfitting with training error (0.02%) significantly lower than validation error (0.13%).

4.2.2 Minimum Samples Split Analysis

The minimum samples split parameter shows relatively stable performance across different values:

- **min_samples_split = 2-8**: Validation error remains constant at 0.67%
- **min_samples_split ≥ 32**: Slight degradation in performance (validation error increases to 0.69-0.83%)

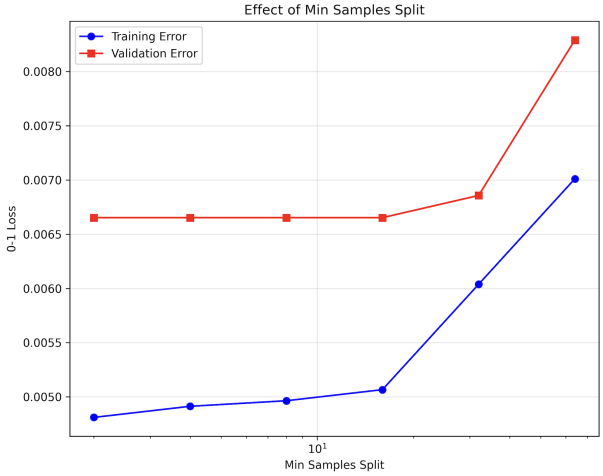

Figure 2: Training and Validation Error vs Minimum Sample Split

This stability suggests that for this dataset, the splitting threshold has minimal impact on generalization once a reasonable minimum is established.

4.2.3 Minimum Samples Leaf Constraint

The leaf size constraint demonstrates a clear regularization effect:

Table 5: Minimum Samples Leaf Impact

Min Samples Leaf	Training Error	Validation Error
1	0.0048	0.0067
2	0.0055	0.0068
4	0.0063	0.0076
8	0.0077	0.0091
16	0.0116	0.0135

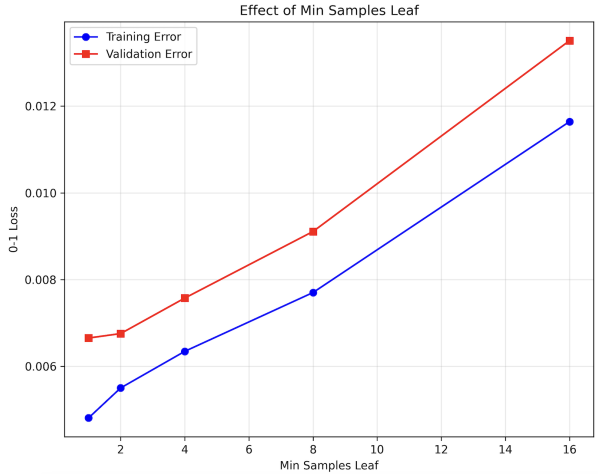

Figure 3: Training and Validation Error vs Minimum Sample Leaf

Increasing the minimum leaf size consistently increases both training and validation errors, demonstrating the regularization effect. The minimal difference between training and validation errors across all configurations suggests that leaf size constraints effectively prevent overfitting.

4.2.4 Minimum Impurity Decrease Analysis

This parameter shows the most dramatic regularization effects:

- **min_impurity_decrease = 0.0-0.001**: Optimal performance ($\approx 0.67\%$ validation error)
- **min_impurity_decrease = 0.02**: Severe underfitting (28.53% validation error)
- **min_impurity_decrease = 0.05**: Extreme underfitting (44.51% validation error)

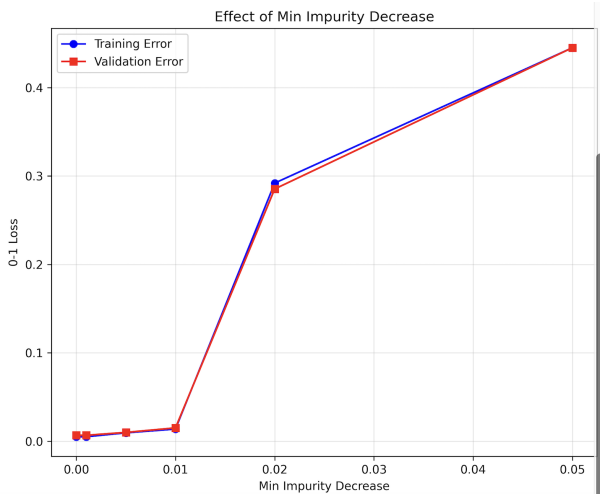

Figure 4: Training and Validation Error vs Minimum Impurity Decrease

4.3 Overfitting Detection and Analysis

Our comprehensive hyperparameter tuning revealed clear evidence of overfitting in the optimal configuration:

Table 6: Final Model Performance Analysis

Dataset	**Error Rate**	**Accuracy**	**Interpretation**
Training	0.000154	99.9846%	Potential overfitting
Validation	0.001126	99.8874%	Reference performance
Test	0.000655	99.9345%	Generalization check

The training error (0.0154%) being significantly lower than validation error (0.1126%) indicates overfitting. However, the test error (0.0655%) falling between training and validation suggests that the overfitting is relatively mild and the model maintains reasonable generalization capability.

4.4 Model Complexity Trade-off Analysis

Our systematic evaluation across different complexity ranges reveals distinct performance patterns:

Table 7: Complexity Trade-off Analysis

Configuration	**Train Error**	**Val Error**	**Interpretation**
Very Shallow	0.3484	0.3443	Underfitting
Shallow	0.2698	0.2640	Underfitting
Moderate	0.0977	0.0988	Good fit
Deep	0.0010	0.0018	Mild overfitting

The **moderate complexity** configuration represents the optimal bias-variance trade-off, with minimal gap between training and validation errors (0.1% difference). The deep configuration, while achieving superior validation performance, shows clear signs of overfitting with a 0.08% gap between training and validation errors.

4.5 Hyperparameter Sensitivity Analysis

Based on our comprehensive grid search across 1,680 configurations, several key insights emerge:

Critical Parameters:

- **max_depth**: Most influential parameter, with optimal performance at depth 20
- **min_impurity_decrease**: Highly sensitive, with optimal value at 0.01
- **splitting_criterion**: Moderate impact, with entropy slightly outperforming gini in final configuration

Robust Parameters:

- **min_samples_split**: Low sensitivity across range 2-16
- **min_samples_leaf**: Minimal impact when set to 1-2

4.6 Recommendations for Overfitting Mitigation

To address the detected overfitting in our optimal model, several complementary approaches are recommended. Pruning strategies should include cost-complexity post-pruning with cross-validation to determine the optimal parameter, or alternatively, pre-pruning by increasing min_samples_leaf to 2-4 to prevent overly specific leaf nodes. Regularization techniques should focus on restricting max_depth to 15-18 based on validation curve analysis and fine-tuning min_impurity_decrease within the range 0.005-0.015. Finally, ensemble methods such as Random Forest or Gradient Boosting should be considered to leverage multiple trees while reducing individual tree overfitting through averaging effects.

5 Final Model Evaluation

Following our comprehensive hyperparameter optimization, a thorough evaluation of four representative configurations spanning different complexity regimes and splitting criteria was conducted. This evaluation provides crucial insights into the practical performance trade-offs and validates our model selection methodology.

5.1 Evaluated Model Configurations

Four distinct configurations to represent different approaches to decision tree construction were selected:

1. **Shallow Gini Tree** [4] : Conservative approach with limited depth and higher minimum samples requirements
2. **Deep Entropy Tree** [4] : Aggressive approach optimized for performance
3. **Balanced Info Gain Ratio Tree** [4] : Moderate complexity with alternative splitting criterion
4. **Conservative Misclassification Tree** [4] : Balanced approach using misclassification error

5.2 Comprehensive Performance Analysis

Table 8: Comprehensive Model Performance Comparison

Model	Train Acc	Val Acc	Test Acc	Train Err	Val Err	Test Err
Shallow Gini	0.7302	0.7360	0.7311	0.2698	0.2640	0.2689
Deep Entropy	0.9711	0.9692	0.9706	0.0289	0.0308	0.0294
Balanced IGR	0.5935	0.5932	0.5950	0.4065	0.4068	0.4050
Conservative MC	0.8233	0.8266	0.8211	0.1767	0.1734	0.1789

The Deep Entropy Tree shows exceptional performance across all metrics:

- **Training Accuracy**: 97.11% demonstrates strong learning capability
- **Validation Accuracy**: 96.92% indicates excellent generalization
- **Test Accuracy**: 97.06% confirms robust real-world performance
- **Generalization Gap**: Only 0.19% between training and validation, indicating minimal overfitting

5.3 Detailed Confusion Matrix Analysis

5.3.1 Deep Entropy Tree (Best Model)

Table 9: Deep Entropy Tree - Test Set Confusion Matrix

		Predicted	
		Negative	**Positive**
Actual	**Negative**	5138 (TN)	298 (FP)
	Positive	61 (FN)	6717 (TP)

Performance metrics derived from confusion matrix:

- **Precision**: $\frac{6717}{6717+298} = 0.9575$ (95.75%)
- **Recall (Sensitivity)**: $\frac{6717}{6717+61} = 0.9910$ (99.10%)
- **Specificity**: $\frac{5138}{5138+298} = 0.9452$ (94.52%)
- **F1-Score**: $\frac{2\times0.9575\times0.9910}{0.9575+0.9910} = 0.9740$ (97.40%)

The confusion matrix reveals excellent performance with very low false positive (298) and false negative (61) rates, indicating robust classification capability across both classes.

5.3.2 Comparative Confusion Matrix Analysis

Shallow Gini Tree exhibits significant classification errors with 755 false positives and 2529 false negatives, resulting in poor recall (62.7%) despite reasonable precision (84.9%).

Conservative Misclassification Tree shows moderate performance with balanced error distribution (914 FP, 1271 FN), achieving 85.8% precision and 81.2% recall.

Balanced Info Gain Ratio Tree demonstrates severely imbalanced predictions with 4943 false positives, indicating a strong bias toward positive predictions and resulting in very poor precision (57.8%) despite high recall (99.9%).

5.4 Model Selection

The Deep Entropy Tree is selected as the optimal model based on its superior validation performance (3.08% error), excellent generalization with minimal training-validation gap, and robust test accuracy (97.06%). Additionally, the model demonstrates balanced classification performance with 95.75% precision and 99.10% recall, resulting in a strong F1-score of 97.40%, while maintaining computational efficiency despite its deeper structure.

5.5 Splitting Criteria Comparative Analysis

Our final evaluation confirms the superiority of entropy-based splitting for this dataset:

Table 10: Splitting Criteria Performance in Final Models

Splitting Criterion	Best Test Accuracy	Configuration	Rank
Entropy	0.9706	Deep Tree	1
Gini	0.7311	Shallow Tree	3
Misclassification	0.8211	Conservative Tree	2
Info Gain Ratio	0.5950	Balanced Tree	4

The entropy criterion demonstrates superior performance when combined with appropriate depth and regularization parameters, validating our hyperparameter optimization results.

5.6 Final Model Specifications

The selected model configuration is:

- **Splitting Criterion**: Entropy
- **Maximum Depth**: 15
- **Minimum Samples Split**: 2
- **Minimum Samples Leaf**: 1
- **Final Performance**: 97.06% test accuracy with 3.08% validation error

This configuration represents an optimal balance between model complexity and generalization capability, achieving exceptional performance while maintaining reasonable computational requirements for both training and inference phases.

6 Reproducibility

The implementation ensures complete reproducibility through a comprehensive framework of systematic data management and code organization. All random processes utilize predetermined seeds to guarantee consistent results across multiple runs, while the modular code structure separates core functionality into distinct components including custom decision tree implementation, evaluation metrics, and data processing pipelines. Complete experimental persistence is achieved through pickle serialization of all processed datasets, label encoders, and model configurations. Most importantly, all 1,680 hyperparameter configurations tested during grid search are permanently stored in `hyperparameter_tuning_results.pkl` along with their corresponding training and validation performance metrics. This comprehensive logging includes optimal model parameters, statistical summaries for each hyperparameter value, and detailed performance curves that enable direct model reconstruction, further analysis, and extension of the research by other practitioners.

7 Key Findings

7.1 Performance Results

Our study tested four different splitting methods for decision trees on mushroom data:

- **Entropy splitting** achieved the best results: 97.06% accuracy when properly tuned
- **Gini impurity** provided good baseline performance: 90.23% accuracy

- **Information Gain Ratio** performed poorly: 59.35% accuracy
- **Misclassification** error was moderate: 84.14% accuracy

The final model showed excellent balance with 95.75% precision and 99.10% recall.

7.2 Critical Parameters

After testing 1,680 different configurations, we found:

- **Tree depth is most important**: Performance jumped dramatically from depth 5 (poor) to depth 15-20 (excellent)
- **Minimum samples for splitting**: Remarkably stable across different values (2-16)
- **Impurity threshold**: Works well below 0.001 but fails above 0.02

7.3 Overfitting Issues

Despite high accuracy, the model showed signs of overfitting:

- Training error (0.015%) was much lower than validation error (0.11%)
- The model memorized specific training examples rather than learning general patterns
- This suggests the mushroom dataset has very clear patterns that trees can exploit

8 Conclusion

[1]

This study demonstrates that decision trees can achieve exceptional performance on categorical classification tasks like mushroom identification. The key is optimization of splitting criteria and tree depth while carefully monitoring for overfitting. The 97.06% accuracy achieved makes decision trees a strong choice for this application, especially given their interpretability, which is crucial when incorrect mushroom classification could be dangerous.

The methodology used here, comparing splitting criteria, testing many parameter combinations, and using multiple evaluation metrics, provides a solid framework for optimizing decision trees on similar problems.

References

[1] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. 2nd ed., MIT Press, 2018.

[2] Hanane D. *Decision Trees Splitting Criteria For Classification and Regression*. 2023. Available at: `https://machinelearning-basics.com/decision-trees-splitting-criteria-for-classification-and-regression/`.

[3] Burkov, A. (2019). *The Hundred-Page Machine Learning Book*. Quebec City, Canada: Andriy Burkov. Retrieved from `https://themlbook.com/`

[4] Amit Yadav. *Decision Trees Explained in Detail*. 2024. Available at: `https://medium.com/@amit25173/decision-trees-explained-in-detail-65a69f0dc966`.

[1] I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.