Machine Learning And Statistical Learning

Machine Learning Project: Tree Predictors for Binary Mushroom Classification

Prepared by Alina Imansakipova

DSE, second year

Matricola number: 30587A

Table of Contents

Introduction

This project focuses on the implementation of *binary classification models* using *decision trees* and *random forest*, built entirely from scratch. The objective is to develop interpretable, rule-based models that can accurately classify mushrooms as either edible or poisonous, based on their morphological and ecological traits.

To achieve this goal, the following steps were carried out:

1. Exploring and preprocessing two datasets—one real-world (primary data) and one simulated (secondary) – ensuring they were clean, consistent, and numerically encoded.
2. Implementing decision tree classifiers that support multiple splitting criteria, including Gini impurity, entropy, and a custom scaled entropy function. A random forest ensemble was also constructed by combining multiple decision trees trained on bootstrapped samples with feature subsetting.
3. Conducting a series of experiments to tune hyperparameters, evaluated model performance on unseen data, and analyzed the importance of features used during classification. This process was complemented by a detailed exploratory data analysis to gain insight into the data distribution and identify potential predictors of mushroom toxicity

Data Understanding and Preprocessing

Datasets:

1. Primary Data:
   - 173 unique mushroom species
   - 20 features (17 nominal, 3 numerical: cap-diameter, stem-height, stem-width)
   - Class labels: edible (e), poisonous (p), or unknown (merged with poisonous)
2. Secondary Data:
   - 61,069 synthetic mushroom entries
   - Simulated from primary data using randomized trait generation
   - Same features and format as primary data
   - Used for training and evaluation due to larger size and class balance

Preprocessing Steps:

For datasets:

- Converted column names to lowercase with dashes
- Dropped columns with more than 50% missing values
- Filled missing categorical values with "unknown" and numeric ones with the column mean
- Applied label encoding to convert categorical columns into numerical form
- Normalized the numerical features (z-score normalization) for consistent scale

This preprocessing ensures compatibility with our from-scratch implementation and improves model performance.

Exploratory Data Analysis (EDA)

To gain insights into the structure, quality, and patterns within our mushroom datasets, we performed a detailed exploratory data analysis (EDA) on both the real-world dataset (primary_data.csv) and the larger, synthetic dataset (secondary_data.csv). These analyses informed both preprocessing choices and model design decisions.

Primary Dataset Analysis

The primary dataset contains 173 samples with 23 features, describing various physical and ecological characteristics of mushrooms. These include cap shape, diameter, gill attachment, stem size, and more. The target variable, class, indicates whether the mushroom is edible (e) or poisonous (p). The class distribution shows a mild imbalance, with slightly more poisonous entries.

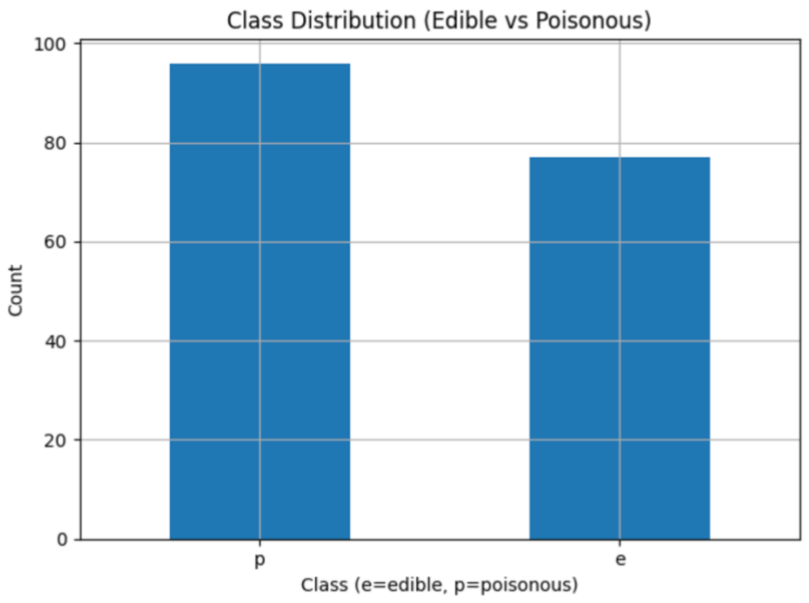
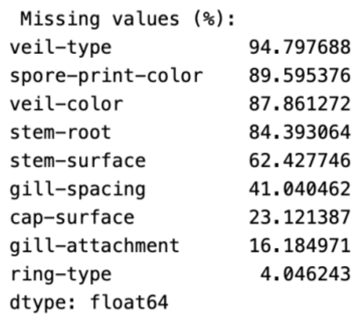


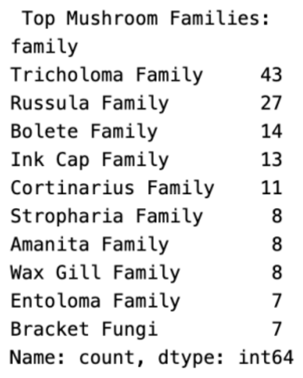
Figure 1. Class distribution in the primary dataset showing a slightly higher number of poisonous mushrooms.

Several columns with a high proportion of missing data were identified. Features like veil-type, veil-color, and spore-print-color were missing in over 80% of rows.



```
 Missing values (%):
veil-type            94.797688
spore-print-color    89.595376
veil-color           87.861272
stem-root            84.393064
stem-surface         62.427746
gill-spacing         41.040462
cap-surface          23.121387
gill-attachment      16.184971
ring-type             4.046243
dtype: float64
```

Figure 2. Percentage of missing values for selected features in the primary dataset.

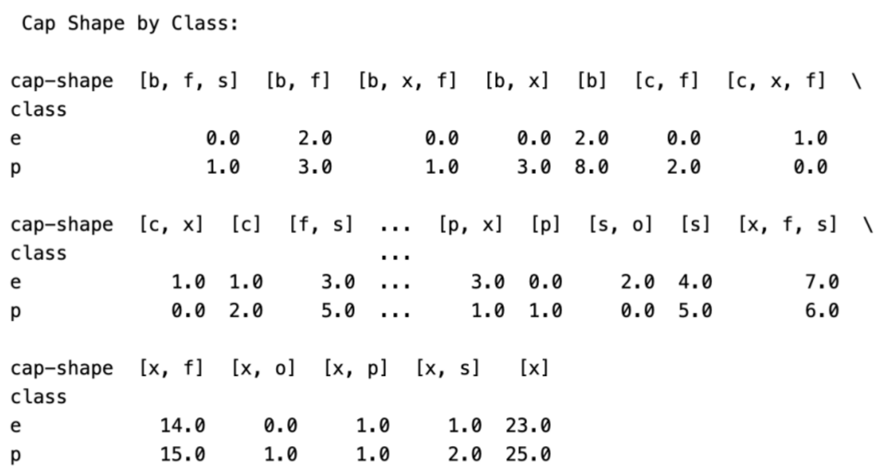


```
 Top Mushroom Families:
family
Tricholoma Family     43
Russula Family        27
Bolete Family         14
Ink Cap Family        13
Cortinarius Family    11
Stropharia Family      8
Amanita Family         8
Wax Gill Family        8
Entoloma Family        7
Bracket Fungi          7
Name: count, dtype: int64
```

Figure 3. Most common mushroom families represented in the primary dataset.

An examination of species frequencies revealed that families such as Tricholoma, Russula, and Bolete were most frequently represented in the data.

Cap shape was evaluated by class, revealing differing distributions between edible and poisonous mushrooms. Certain shapes appeared more frequently in one category, suggesting potential predictive value.



```
 Cap Shape by Class:

cap-shape  [b, f, s]  [b, f]  [b, x, f]  [b, x]  [b]  [c, f]  [c, x, f]  \
class
e                0.0     2.0        0.0     0.0  2.0     0.0        1.0
p                1.0     3.0        1.0     3.0  8.0     2.0        0.0

cap-shape  [c, x]  [c]  [f, s]  ...  [p, x]  [p]  [s, o]  [s]  [x, f, s]  \
class                           ...
e             1.0  1.0     3.0  ...     3.0  0.0     2.0  4.0        7.0
p             0.0  2.0     5.0  ...     1.0  1.0     0.0  5.0        6.0

cap-shape  [x, f]  [x, o]  [x, p]  [x, s]   [x]
class
e            14.0     0.0     1.0     1.0  23.0
p            15.0     1.0     1.0     2.0  25.0
```

Figure 4. Frequency of various cap shapes in edible and poisonous mushrooms.

Boxplots of cap diameter stratified by class indicated that edible mushrooms exhibited a broader range of sizes, although considerable overlap remained between the two categories.

Habitat distribution provided additional insights, with specific locations such as [d] or [g, d] showing a distinct class imbalance. Further differences were observed in the distributions of cap color and ring type across classes.

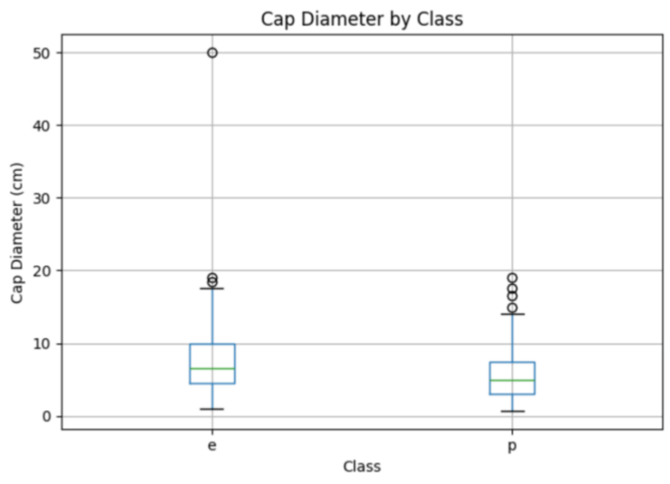


Figure 5. Distribution of cap diameter across classes.

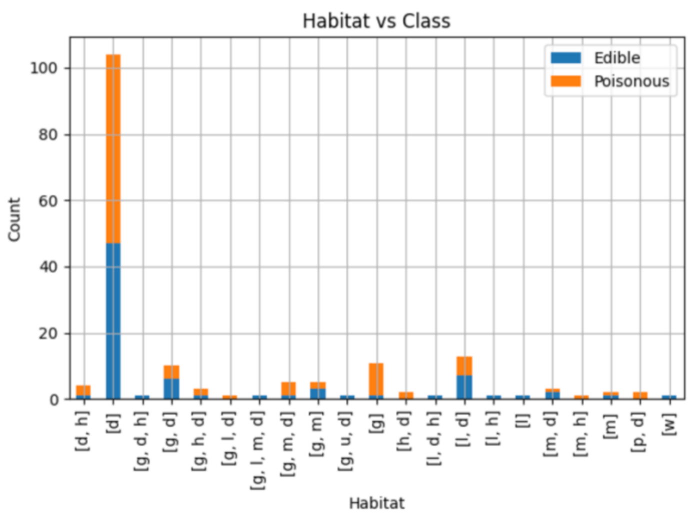
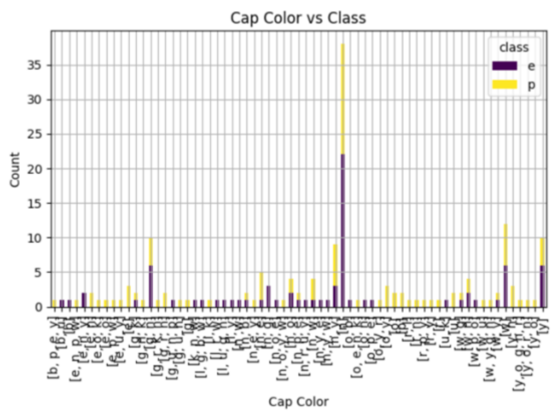


Figure 6. Stacked bar plot showing mushroom class distribution across different habitats.



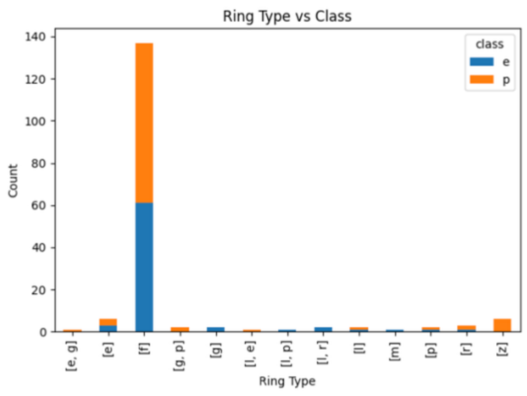


Figure 7. Cap color distribution across edible and poisonous classes.

Figure 8. Ring type distribution across edible and poisonous mushrooms.

A heatmap of numeric features showed moderate correlation between cap diameter and stem width, while stem height displayed weaker associations.

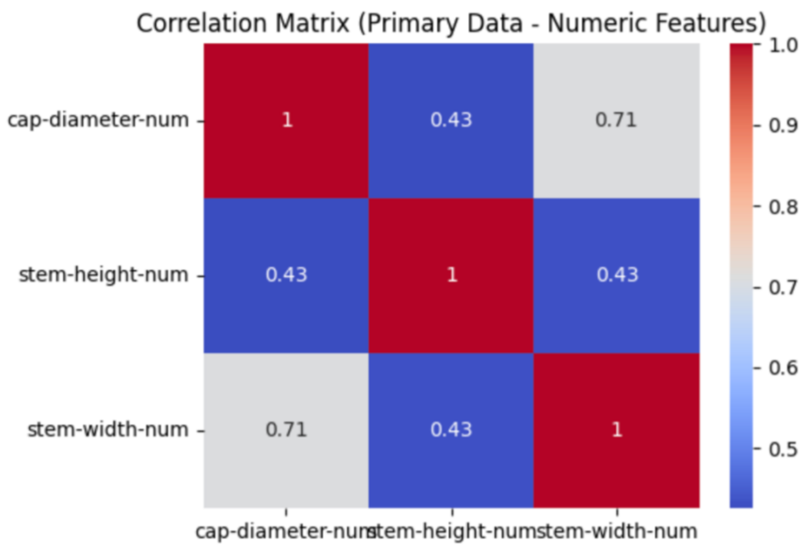


Figure 9. Correlation heatmap for numeric features in the primary dataset.

Secondary Dataset Analysis

The secondary dataset includes 61,069 entries and represents a synthetic but more extensive version of the mushroom dataset. It is numerically encoded and therefore readily usable for training tree-based models. The binary class label uses 0 for edible and 1 for poisonous.

The target distribution remains relatively balanced but with a larger proportion of poisonous samples.

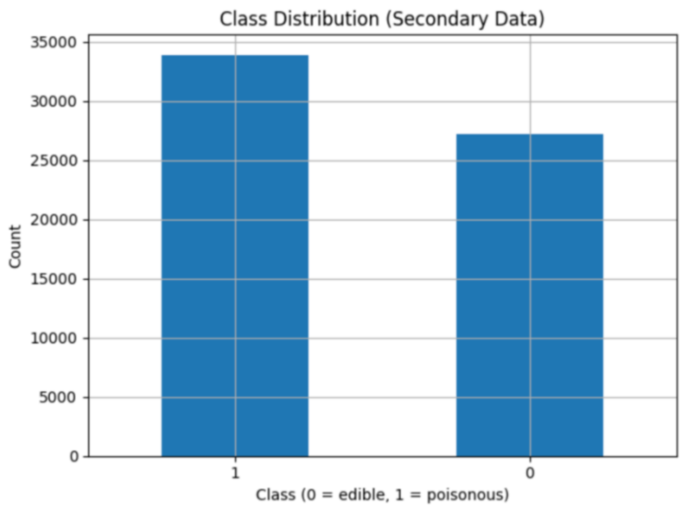


Figure 10: Distribution of edible vs poisonous classes in the secondary dataset.

The distribution of the normalized cap diameter remains skewed, indicating that even after standardization, the underlying data is not perfectly symmetric.

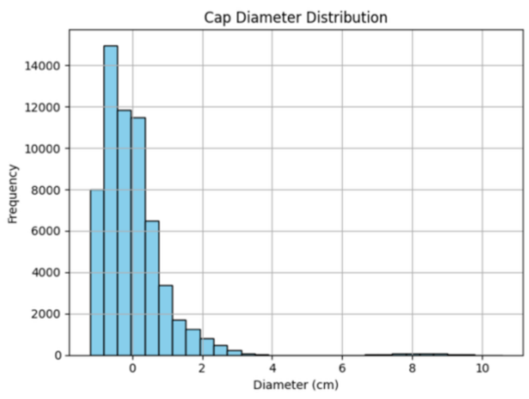


Figure 11: Histogram of normalized cap diameter in the secondary dataset.

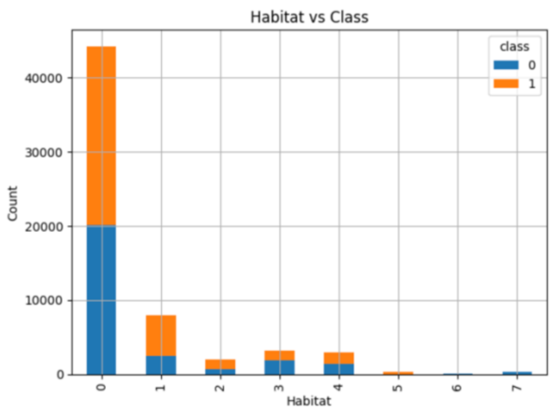


Figure 12: Distribution of habitat types stratified by class.

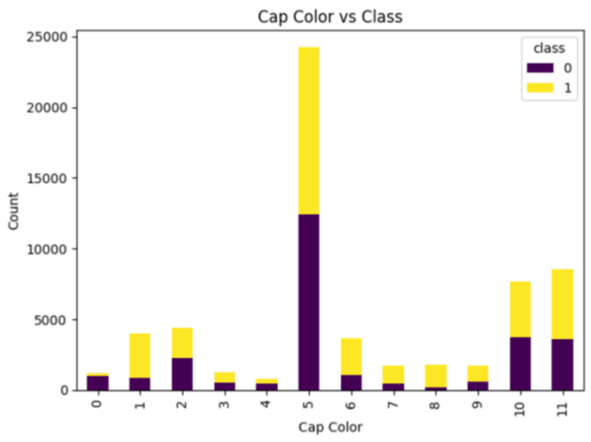


Figure 13: Class breakdown by cap color values.

Class proportions vary significantly by habitat. The most common habitat category is strongly associated with poisonous mushrooms. Cap color is another class-informative attribute. One specific color dominates across both classes but is more strongly associated with edibility.The correlation structure among numeric features largely mirrors the primary dataset. Cap diameter and stem width are moderately correlated, while stem height is more weakly related.

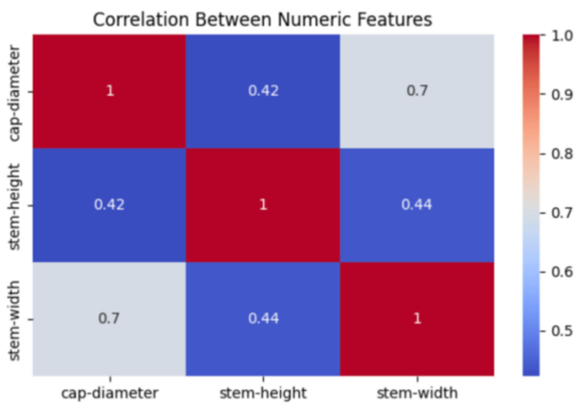


Figure 14: Correlation heatmap between normalized numeric features in the secondary dataset.

Overall, the exploratory analysis highlights several features—such as habitat, cap shape, and color—that exhibit class-specific patterns and could be valuable for classification. These insights informed feature selection and guided the modeling choices in subsequent stages of the project.

Methodology

This section describes the implementation of custom tree-based classifiers used to address the binary classification problem of distinguishing edible from poisonous mushrooms. Two models were developed from scratch: a Decision Tree and a Random Forest. Both were designed to be flexible with respect to splitting criteria and stopping conditions, enabling comprehensive experimentation.

Decision Tree Implementation

The decision tree classifier was constructed using two core classes: TreeNode and DecisionTree. Each node in the tree is represented by an instance of TreeNode, storing information such as the splitting feature, threshold, and prediction (if it is a leaf node). The evaluate method performs recursive traversal for inference.

The training process is handled by the DecisionTree class, which builds the tree recursively. At each node, the algorithm considers all features and their unique thresholds, computing the gain associated with splitting on each candidate. The best split is chosen based on the highest information gain.

To support flexible experimentation, three splitting criteria were implemented:

- Gini impurity: Measures class impurity and favors more homogeneous splits.
- Entropy: Captures the amount of information or uncertainty in the class labels.
- Scaled Entropy: A variant where the entropy of each child node is scaled by the proportion of samples, enabling more nuanced gain calculations.

Stopping conditions for tree growth include:

- Reaching a specified maximum depth.
- The number of samples falling below a minimum split threshold.
- All samples at a node belonging to a single class.
- No split producing positive information gain.

The recursive function _build_tree handles the training phase, returning leaf nodes when any stopping condition is met. The predict method performs batch predictions by evaluating each sample individually via the root node.

Random Forest Implementation

The Random Forest classifier extends the decision tree model through bagging and random feature selection. It builds an ensemble of n_trees decision trees, each trained on a bootstrap sample of the training data. For every tree, a random subset of features is selected to promote diversity among base learners and reduce overfitting.

Each tree is trained independently using the previously described DecisionTree class. The feature subset size is determined using one of the following strategies:

- A fixed number.
- The square root of the total number of features (sqrt).
- The base-2 logarithm of the total number of features (log2).

During inference, each tree predicts a class label for every input. The final prediction for each sample is obtained through majority voting among the ensemble.

The random forest implementation therefore enhances generalization and robustness, while retaining interpretability through the underlying decision tree structure.

Experiments and Results

This section presents a series of experiments conducted to evaluate the performance of custom decision tree classifiers trained on the secondary mushroom dataset. The primary focus is on understanding how different splitting criteria and hyperparameter settings affect classification accuracy and generalization.

To ensure robust model evaluation, the dataset was manually shuffled and split into three distinct subsets:

- *Training set (60%):* Used to build the model.
- *Validation set (20%):* Used for tuning hyperparameters.
- *Test set (20%):* Used for final evaluation of selected models.

The features and labels were extracted accordingly, and the same split was used across all experiments to maintain consistency.

Hyperparameter Grid

To tune the tree-based models, a grid search was performed over two key hyperparameters:

- max_depth: The maximum depth of the tree. Values [3, 5, 7] were selected to explore a range from shallow (underfitting) to deeper (potential overfitting) models.

- min_samples_split: The minimum number of samples required to split a node. Values [2, 5, 10] allow investigation into how aggressively or conservatively nodes are split.

This grid results in 9 combinations per criterion, each evaluated using training and validation error.

Gini Criterion

The Gini impurity was the first criterion evaluated. The grid search results show a clear pattern in which increasing the maximum depth consistently reduces both training and validation errors.

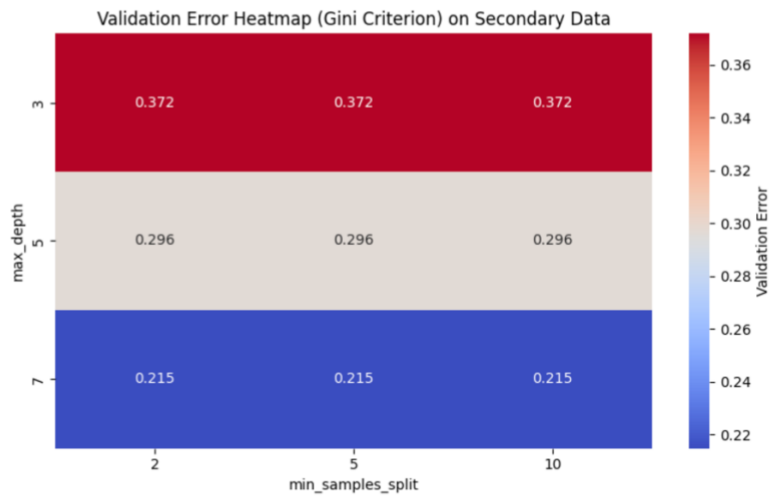
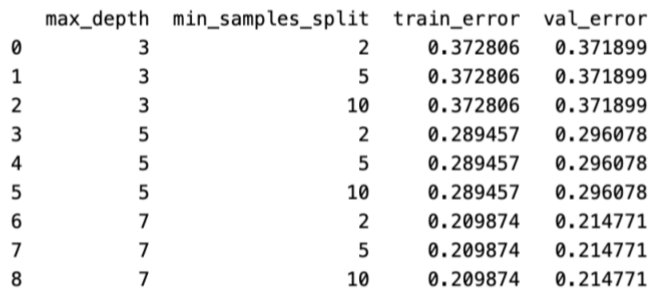


Figure 15: Validation Error Heatmap (Gini Criterion) on Secondary Data



```
   max_depth  min_samples_split  train_error  val_error
0          3                  2     0.372806   0.371899
1          3                  5     0.372806   0.371899
2          3                 10     0.372806   0.371899
3          5                  2     0.289457   0.296078
4          5                  5     0.289457   0.296078
5          5                 10     0.289457   0.296078
6          7                  2     0.209874   0.214771
7          7                  5     0.209874   0.214771
8          7                 10     0.209874   0.214771
```

Figure 16: Training and Validation Error Table (Gini Criterion)

The heatmap reveals that at depth 3, the model performs poorly, with a validation error of 0.372 across all min_samples_split values. Increasing the depth to 5 improves validation performance to 0.296, and depth 7 yields the lowest error of 0.215. The influence of the min_samples_split parameter appears negligible in this range, as its variation does not impact validation performance significantly.

The grid table confirms the same trend: deeper trees (up to depth 7) yield both lower training and validation errors, suggesting that the tree is still benefiting from additional splits and has not yet overfit the training data.

The learning curve shows how increasing tree depth reduces both training and validation error in parallel, indicating improved model capacity without immediate signs of overfitting at the maximum depth tested. This behavior supports the selection of depth=7 and min_samples_split=2 as the best-performing configuration under the Gini criterion.

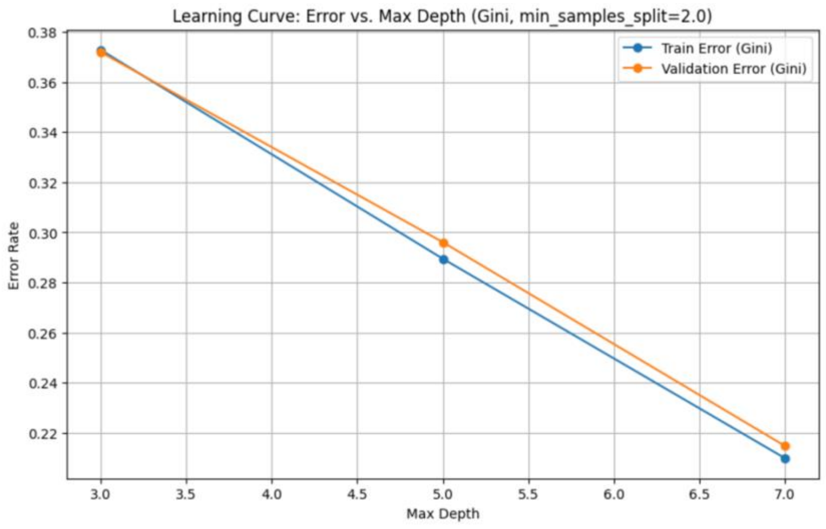


Figure 17: Learning curve for Gini criterion with min_samples_split = 2.

Entropy Criterion

The entropy-based splitting strategy led to a slightly improved performance over the Gini-based approach across most hyperparameter settings. As with Gini, the best results were observed at the largest tested tree depth.

The validation error heatmap (Figure 18) shows a consistent decrease in error with increasing tree depth. For max_depth=7, all values of min_samples_split yielded a validation error of 0.206. At max_depth=5, the error plateaued at 0.307, while shallower trees with max_depth=3 failed to differentiate between splitting strategies, all resulting in a higher validation error of 0.372. The train-validation error table in Figure 19 confirms this consistency across settings.

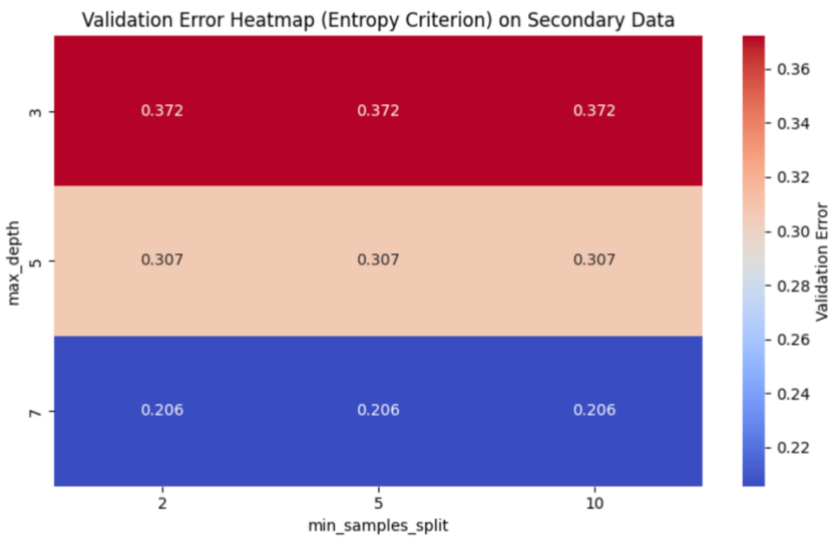
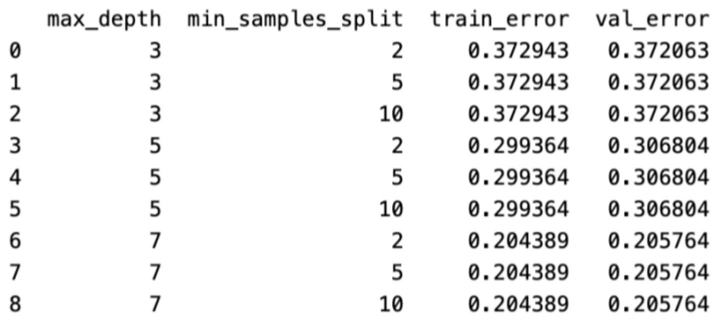


Figure 18: Validation Error Heatmap (Entropy Criterion) on Secondary Data



```
   max_depth  min_samples_split  train_error  val_error
0          3                  2     0.372943   0.372063
1          3                  5     0.372943   0.372063
2          3                 10     0.372943   0.372063
3          5                  2     0.299364   0.306804
4          5                  5     0.299364   0.306804
5          5                 10     0.299364   0.306804
6          7                  2     0.204389   0.205764
7          7                  5     0.204389   0.205764
8          7                 10     0.204389   0.205764
```

Figure 19: Training and Validation Error Table (Entropy Criterion)

To further examine model performance over increasing tree depth, a learning curve for the best-performing min_samples_split=2 was plotted (Figure 20). A steep drop in both training and validation errors between depths 3 and 7 is observed, with a narrowing gap between the two curves. This indicates that increasing model complexity improved performance without overfitting the training set—a desirable behavior for a decision tree.

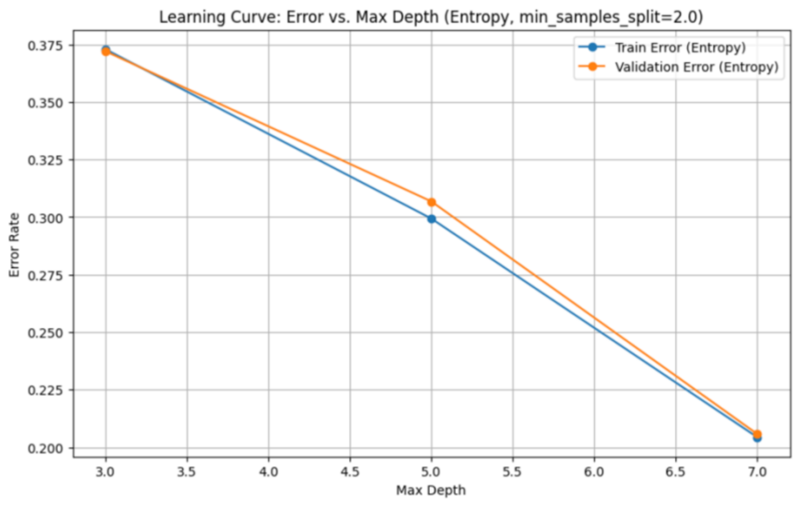


Figure 20: Learning curve for Entropy criterion with min_samples_split = 2.

Together, these results highlight entropy as a strong criterion for this dataset and tree design, yielding lower error rates at greater tree depth while maintaining generalization.

Scaled Entropy Criterion

The final splitting criterion evaluated was Scaled Entropy, which modifies the standard entropy calculation by incorporating a scaling factor. As with previous criteria, a grid search was conducted using combinations of max_depth values {3, 5, 7} and min_samples_split values {2, 5, 10}. These parameter values were selected to balance model expressiveness and computational feasibility, ensuring that the models are neither too simple nor prone to overfitting.

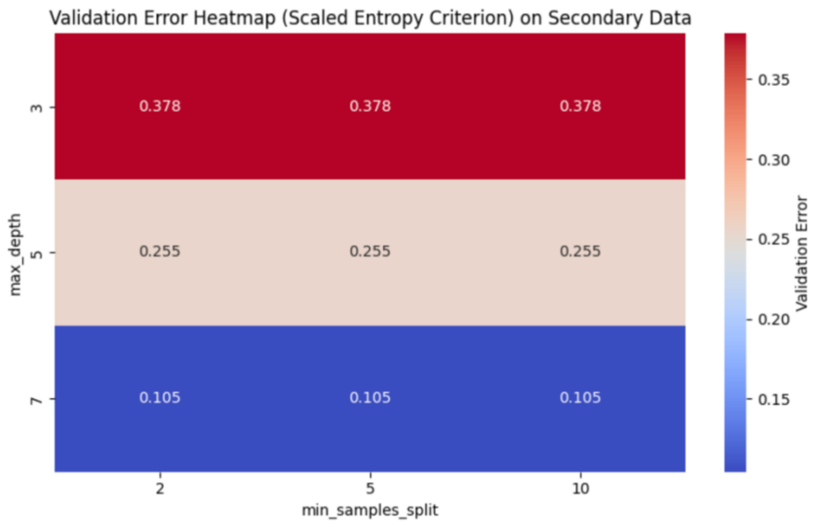
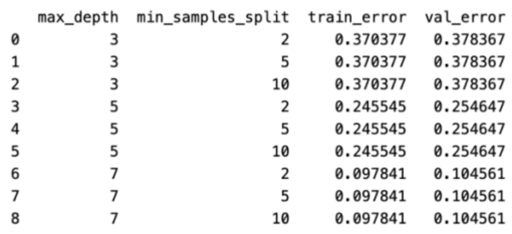


Figure 21: Validation Error Heatmap (Scaled Entropy Criterion) on Secondary Data

Figure 21 shows the validation error heatmap for Scaled Entropy. The pattern of results is consistent with previous criteria: increasing tree depth leads to a notable reduction in validation error. At max_depth=3, the error remains uniformly high (0.378). With max_depth=5, it drops significantly to 0.255, and further to 0.105 at max_depth=7, regardless of the min_samples_split value. This again suggests that min_samples_split has limited effect within this depth range, while depth plays a more dominant role in improving model fit.

The tabular summary in Figure 22 lists the training and validation errors across all parameter combinations. The learning curve in Figure 23 further illustrates the behavior of the model as depth increases. Both training and validation errors decrease almost linearly with increasing depth. The training error drops from 0.370 at depth 3 to 0.098 at depth 7, while the validation error follows a similar trend, reducing from 0.378 to 0.105. This indicates a consistent improvement in model performance without immediate signs of overfitting at these depths.



```
   max_depth  min_samples_split  train_error  val_error
0          3                  2     0.370377   0.378367
1          3                  5     0.370377   0.378367
2          3                 10     0.370377   0.378367
3          5                  2     0.245545   0.254647
4          5                  5     0.245545   0.254647
5          5                 10     0.245545   0.254647
6          7                  2     0.097841   0.104561
7          7                  5     0.097841   0.104561
8          7                 10     0.097841   0.104561
```

Figure 22: Training and Validation Error Table (Scaled Entropy Criterion)

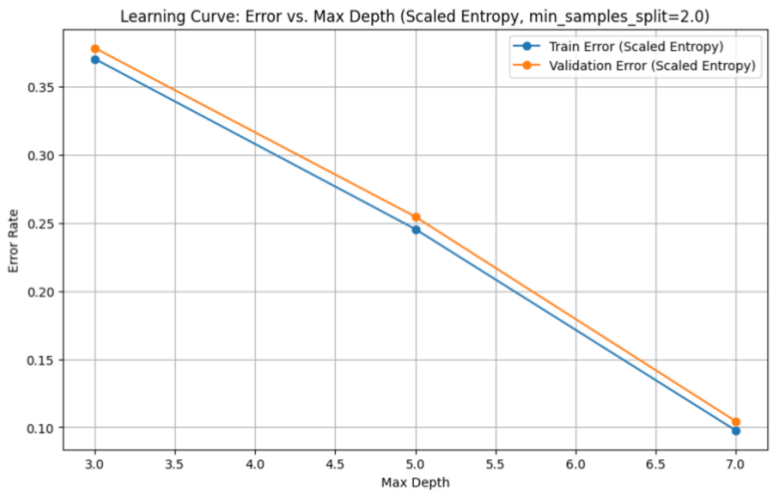
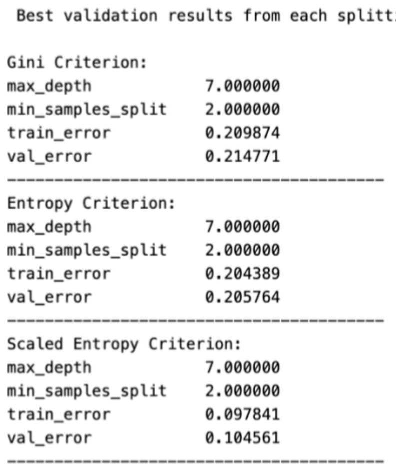


Figure 23: Learning curve for Scaled Entropy criterion with min_samples_split = 2.

These results confirm the robustness of Scaled Entropy at deeper depths, and suggest that this criterion may yield better generalization performance than Gini or standard Entropy under the tested conditions.

Selection of Best Model(s)

Based on validation performance, models using a maximum depth of 7 and a minimum samples split of 2 consistently yielded the lowest error across all three criteria. A comparison of the best-performing configurations is shown below:



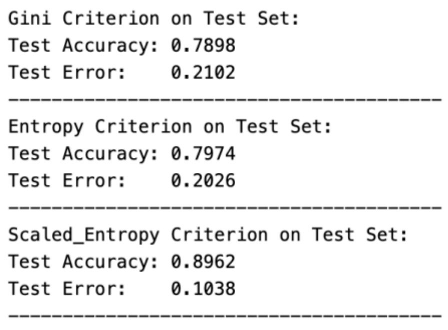
```
 Best validation results from each splitting criterion:

Gini Criterion:
max_depth            7.000000
min_samples_split    2.000000
train_error          0.209874
val_error            0.214771
----------------------------------------
Entropy Criterion:
max_depth            7.000000
min_samples_split    2.000000
train_error          0.204389
val_error            0.205764
----------------------------------------
Scaled Entropy Criterion:
max_depth            7.000000
min_samples_split    2.000000
train_error          0.097841
val_error            0.104561
----------------------------------------
```

Figure 24: Summary of best-performing hyperparameter settings for each splitting criterion, including corresponding training and validation error rates.

Among the three criteria, the Scaled Entropy model achieved the lowest validation error (0.1046), followed by Entropy (0.2058), and Gini (0.2148). This suggests that the scaling applied to entropy may improve the model's discriminative capacity, potentially by better balancing the class distributions or stabilizing split evaluations during training.

Performance on Test Set

To assess generalizability, the three best models were evaluated on the test set. The Scaled Entropy-based decision tree maintained its lead with the lowest test error (0.1038) and highest test accuracy (0.8962), clearly outperforming its Gini and Entropy counterparts:



```
Gini Criterion on Test Set:
Test Accuracy: 0.7898
Test Error:    0.2102
----------------------------------------
Entropy Criterion on Test Set:
Test Accuracy: 0.7974
Test Error:    0.2026
----------------------------------------
Scaled_Entropy Criterion on Test Set:
Test Accuracy: 0.8962
Test Error:    0.1038
----------------------------------------
```

Figure 25: Comparison of test accuracy and error for the best-performing model from each splitting criterion.

This reinforces the earlier observation from the validation phase and supports the selection of the Scaled Entropy-based decision tree as the most effective model under the evaluated configurations. Furthermore, a confusion matrix for the Scaled Entropy model reveals balanced performance across both edible and poisonous classes, though with a slightly higher false negative count. This is consistent with the model's conservative leaning in predicting poisonous mushrooms.

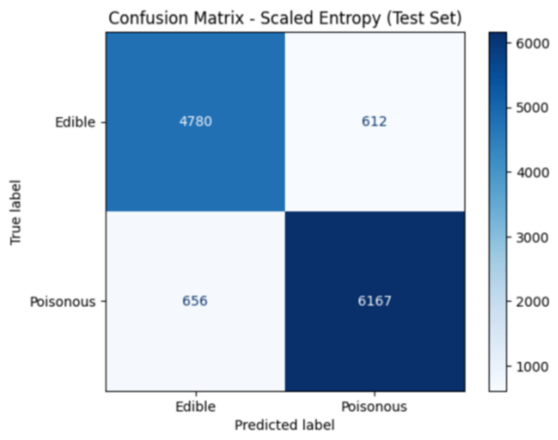


Figure 26: Confusion matrix for the Scaled Entropy model on the test set.

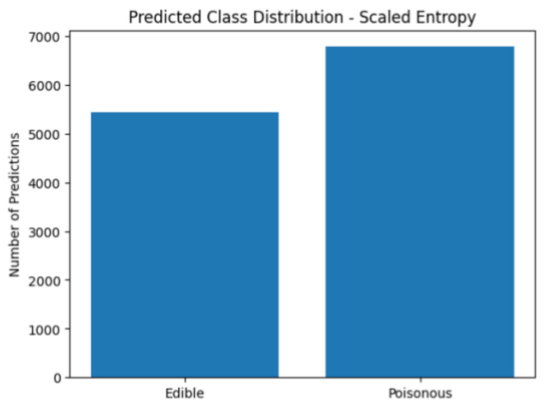


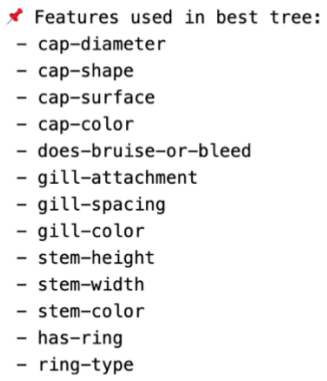
Figure 27: Distribution of predicted classes for the Scaled Entropy model.

Additionally, the distribution of predicted classes shows a modest preference toward the poisonous class. This aligns with the model's conservative bias toward classifying mushrooms as poisonous — a favorable tendency in this context, where the cost of misclassifying a poisonous mushroom as edible is significantly higher. Additionally, the predicted class distribution reflects the underlying class proportions observed in both the primary and secondary datasets, further supporting the model's robustness in capturing real-world patterns.

Feature Importance and Usage

To better understand the decision-making process of the final model, the features used in the best-performing tree—based on the Scaled Entropy criterion—were extracted and analyzed. The model selected 15 distinct features, encompassing a mix of morphological, structural, and physiological traits of mushrooms.

Several of these features align with domain knowledge in mycology, such as cap characteristics (diameter, shape, surface, and color), gill structure, and the presence of bruising or bleeding. Their inclusion supports the biological relevance of the learned patterns. Moreover, the diversity of feature types—both numeric and categorical—demonstrates the model's ability to integrate multi-modal data to distinguish between edible and poisonous specimens.



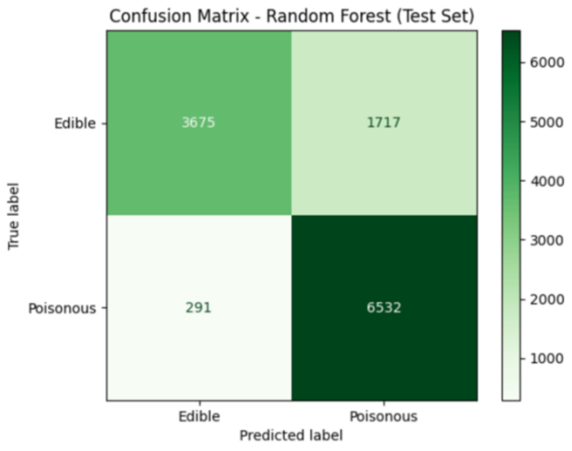
```
📌 Features used in best tree:
 - cap-diameter
 - cap-shape
 - cap-surface
 - cap-color
 - does-bruise-or-bleed
 - gill-attachment
 - gill-spacing
 - gill-color
 - stem-height
 - stem-width
 - stem-color
 - has-ring
 - ring-type
```

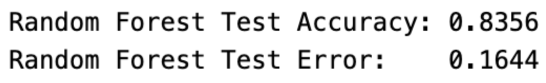
Figure 28: Features utilized in the best decision tree model based on Scaled Entropy, reflecting key predictors in the classification of mushrooms.

Random Forest Performance

To evaluate whether an ensemble method could outperform a single decision tree, a Random Forest classifier was trained using the same training data. This model combines multiple decision trees, each trained on different bootstrap samples of the data and a random subset of features, to improve generalization and reduce overfitting.

The Random Forest achieved a test accuracy of 0.8356, with a corresponding error rate of 0.1644, placing its performance between the best decision tree with entropy-based criteria and the scaled entropy variant. While not as accurate as the scaled entropy tree (0.8962), the Random Forest displayed a noticeable improvement over the standard Gini-based and Entropy-based trees.





```
Random Forest Test Accuracy: 0.8356
Random Forest Test Error:    0.1644
```

Figure 29: Confusion matrix for Random Forest on the test set.

Figure 30: Random Forest test accuracy and error rate.

A confusion matrix reveals the breakdown of correct and incorrect predictions. The Random Forest classified poisonous mushrooms with high accuracy but made more errors with edible specimens compared to the scaled entropy tree. Specifically, 6532 poisonous mushrooms were correctly classified, while 291 were misclassified as edible. On the other hand, 3675 edible mushrooms were correctly identified, but 1717 were wrongly labeled as poisonous.

These results suggest that the Random Forest provides a solid balance of performance and generalization. While slightly less accurate than the scaled entropy tree, it exhibits a lower variance and is generally more robust to noise, making it a competitive alternative.

Discussion

This project developed and compared several decision tree variants and a random forest classifier for the binary classification of mushroom edibility. Among the models evaluated, the decision tree employing scaled entropy as its splitting criterion achieved the best overall performance, attaining a validation error of 0.1046 and a test accuracy of 0.8962. This model consistently outperformed trees based on standard Gini impurity and Entropy, which exhibited higher error rates. The Random Forest model, while slightly less accurate than the top-performing scaled entropy tree (test accuracy of 0.8356), demonstrated strong generalization, offering robust predictions by averaging across multiple trees and thereby reducing variance.

Overfitting and Underfitting Behavior

The learning curves generated during hyperparameter tuning provided valuable insights into model complexity. For all three decision tree criteria, an increase in tree depth led to a concurrent decrease in both training and validation errors. Crucially, there was no sharp divergence observed between these error curves, suggesting that the models were well-fitted and did not suffer from severe overfitting or underfitting within the tested hyperparameter range. For instance, the scaled entropy model exhibited smooth, closely aligned reductions in both training and validation errors with increasing depth, indicating an appropriate model capacity and good generalization for this classification task.

Comparison of Splitting Criteria

Across all tested depths and min_samples_split values, the scaled entropy criterion consistently resulted in lower training and validation errors compared to its counterparts. While Entropy and Gini impurity yielded similar performance patterns—with Entropy generally performing marginally better in validation accuracy—scaled entropy provided demonstrably more accurate and stable predictions. This superior performance might be attributed to its more nuanced handling of class impurity, potentially by better balancing class distributions or stabilizing split evaluations during the tree construction process.

Strengths of the Implementation

A significant strength of this project lies in the flexibility and interpretability afforded by the from-scratch implementation of the classifiers. All components, including support for multiple impurity criteria and comprehensive hyperparameter control for the decision trees, were custom-built. This approach facilitated a deeper understanding of how individual algorithmic components influence model performance and allows for high adaptability to different datasets or classification problems.

Limitations

Several limitations warrant consideration. Firstly, the computational cost associated with a full grid search for hyperparameters, particularly within a from-scratch Python implementation, is substantial and may not be scalable to larger datasets or more complex tree architectures without significant computational resources or optimized search strategies. Secondly, the hyperparameter space explored was confined to three values for max_depth and min_samples_split respectively; it remains possible that superior configurations exist outside this specific grid. A critical constraint is the heavy reliance on the synthetic secondary dataset for model training. While this dataset enabled extensive experimentation, its synthetic nature raises questions about the models' generalizability to new, real-world mushroom species not represented in the smaller primary dataset. Although the primary data informed EDA and was implicitly part of the synthetic data generation, its limited size restricts the depth of validation against truly novel, real-world examples.

Future Work

To build upon this work and address the identified limitations, several future directions are proposed:

- Expanded Hyperparameter Search: Conduct a more extensive grid search or employ more advanced hyperparameter optimization techniques to explore a broader range of tree configurations.
- Decision Tree Pruning: Implement and evaluate pruning mechanisms (e.g., cost-complexity pruning) to potentially enhance decision tree generalization and reduce model complexity without sacrificing performance.
- Advanced Feature Importance for Random Forest: Utilize more sophisticated feature importance techniques for the Random Forest model, such as permutation importance or SHAP (SHapley Additive exPlanations) values, to gain deeper insights into feature contributions.
- Validation on Diverse Real-World Data: Test the developed models on additional, diverse real-world mushroom datasets, particularly those containing species not encountered in the current datasets, to more rigorously assess their real-world applicability and generalizability beyond synthetic examples.

Conclusion

This project demonstrated the implementation and evaluation of decision tree and random forest classifiers for the task of binary mushroom classification. By building models entirely from scratch, it provided an in-depth understanding of key machine learning concepts, including splitting criteria, overfitting control, and ensemble learning.

Among the tested approaches, the decision tree using the custom scaled entropy criterion consistently outperformed alternatives, achieving the highest validation and test accuracy. The random forest model, while slightly less accurate, proved to be a robust and generalizable alternative, benefiting from ensemble diversity.

The experiments confirmed the importance of thoughtful hyperparameter tuning, informed data preprocessing, and thorough exploratory analysis. Additionally, the project highlighted the value of flexibility in model design, interpretability through feature usage tracking, and the challenges posed by synthetic data when aiming for real-world applicability.

Overall, this work lays a solid foundation for further improvements—both in model sophistication and evaluation depth—and emphasizes the practicality and interpretability of rule-based classifiers in sensitive domains such as mushroom toxicity detection.