Implementation of Tree Predictors for Mushroom Classification

Peri Sara

October 18, 2024

Abstract

This report presents the development and evaluation of decision tree classifiers and a Random Forest algorithm for classifying mushrooms as edible or poisonous using data from the UCI Machine Learning Repository. After meticulous data preprocessing—including handling missing values and removing duplicates—we obtained a clean dataset comprising 36,948 observations and 15 features. We explored various impurity measures such as the Gini index, entropy, and a square root function to determine the most effective splitting criteria for the decision trees.

Through comprehensive hyperparameter tuning using cross-validation, we identified that the optimal decision tree configuration utilized the Gini impurity measure with no maximum depth and a minimum of five samples required to split. This model achieved a mean cross-validation accuracy of 99.77%. Building upon this, we developed a Random Forest model by integrating multiple decision trees with the selected hyperparameters, employing bootstrap aggregation and random feature selection to enhance performance. The Random Forest outperformed the individual decision tree classifier, achieving an accuracy of 99.95

The results demonstrate that both decision trees and Random Forests are highly effective for this classification task, exhibiting strong predictive capabilities and excellent generalization to unseen data.

Contents

1 Introduction

The main objective of this project is to develop decision tree classifiers from scratch for the binary classification of mushrooms using the Mushroom Dataset. We aim to create models capable of effectively distinguishing between edible and poisonous mushroom samples based on their inherent characteristics.

Our methodology involves implementing decision trees with various splitting and stopping criteria, performing hyperparameter tuning through cross-validation, and analyzing the models' performance. Additionally, we explore the implementation of a Random Forest, reusing the previously developed decision tree structure to enhance predictive capabilities further.

The report is organized as follows: Section 2 presents the dataset description and data preprocessing steps; Section 3 outlines the implementation of the decision tree classifier; Section 4 discusses the training of the decision trees, including the adopted splitting and stopping criteria; Section 5 is dedicated to hyperparameter tuning and analysis of the obtained results; Section 6 provides an evaluation of the models' performance and discusses potential overfitting or underfitting; Section 7 describes the implementation of the Random Forest and evaluates its performance; finally, Section 8 concludes the report with final considerations and suggestions for future work.

2 Dataset Description and Data Preprocessing

The Mushroom Dataset, sourced from the UCI Machine Learning Repository [1], contains records of various mushroom species, each characterized by several attributes. Initially, the dataset consists of 8,124 observations and 23 features. These features include categorical and ordinal attributes such as cap shape, cap color, gill size, and habitat. The target variable classifies mushrooms as either *edible* or *poisonous*, resulting in a binary classification task.

Table 1: Description of Variables

No.	**Variable Name**	**Type**	**Possible Values (Codes)**
1	cap-diameter	numerical	Float number in cm
2	cap-shape	categorical	bell (b), conical (c), convex (x), flat (f), sunken (s), spherical (p), others (o)
3	cap-surface	categorical	fibrous (i), grooves (g), scaly (y), smooth (s), shiny (h), leathery (l), silky (k), sticky (t), wrinkled (w), fleshy (e)
4	cap-color	categorical	brown (n), buff (b), gray (g), green (r), pink (p), purple (u), red (e), white (w), yellow (y), blue (l), orange (o), black (k)
5	does-bruise-bleed	categorical	bruises-or-bleeding (t), no (f)
6	gill-attachment	categorical	adnate (a), adnexed (x), decurrent (d), free (e), sinuate (s), pores (p), none (f), unknown (?)
7	gill-spacing	categorical	close (c), distant (d), none (f)
8	gill-color	categorical	Same as cap-color plus none (f)
9	stem-height	numerical	Float number in cm
10	stem-width	numerical	Float number in mm
11	stem-root	categorical	bulbous (b), swollen (s), club (c), cup (u), equal (e), rhizomorphs (z), rooted (r)
12	stem-surface	categorical	Same as cap-surface plus none (f)
13	stem-color	categorical	Same as cap-color plus none (f)
14	veil-type	categorical	partial (p), universal (u)
15	veil-color	categorical	Same as cap-color plus none (f)
16	has-ring	categorical	ring (t), none (f)
17	ring-type	categorical	cobwebby (c), evanescent (e), flaring (r), grooved (g), large (l), pendant (p), sheathing (s), zone (z), scaly (y), movable (m), none (f), unknown (?)
18	spore-print-color	categorical	Same as cap-color
19	habitat	categorical	grasses (g), leaves (l), meadows (m), paths (p), heaths (h), urban (u), waste (w), woods (d)
20	season	categorical	spring (s), summer (u), autumn (a), winter (w)

Data Exploration A preliminary examination of the dataset revealed the presence of missing values in certain features. Specifically, some features exhibited a significant proportion of missing data, which could adversely affect model performance if not addressed properly. Additionally, the dataset was inspected for duplicate entries to prevent redundancy and potential bias in the analysis.

Data Preprocessing **Handling Missing Values:**

Features with more than 25% missing values were dropped from the dataset to mitigate the impact of incomplete data on model accuracy. This threshold was selected to balance the retention of valuable information with the necessity of data quality. Specifically, six features exceeding this threshold were removed. Subsequently, any remaining rows containing missing values were eliminated to ensure a complete dataset for analysis.

Removing Duplicates:

Duplicate records were identified and removed to prevent skewing the training process. A total of 117 duplicate observations were eliminated from the dataset.

Final Dataset After preprocessing, the final dataset consists of 36,948 observations and 15 features. The dataset is now free of missing values and duplicates, providing a clean foundation for model development. No additional transformations or normalizations were applied, as the features are primarily categorical or ordinal in nature.

Class Distribution The class distribution in the final dataset is relatively balanced, with 27,181 edible (55.49%) and 33,888 poisonous (44.50%) mushrooms. This balance is crucial for training unbiased classifiers and ensuring reliable performance across classes.

3 Implementation of the Decision Tree Classifier

Decision tree classifiers are powerful tools in supervised learning, particularly for classification tasks. They recursively partition the dataset into subsets based on feature values, aiming to increase the purity of the subsets concerning the target class.

The core idea behind decision trees is to select splits that best separate the data into homogeneous subsets. This selection is guided by impurity measures, which quantify how well a feature at a node splits the data into distinct classes. Lower impurity indicates a better split.

3.1 Impurity Measures

In this implementation, three impurity measures were considered to evaluate potential splits at each node:

Gini Function The Gini function measures the impurity of a split and is used to determine which feature should be used to partition the data at a node. It is calculated as:

$$\psi(p) = 2p(1-p)$$

where:

- p represents the probability that a randomly selected sample belongs to the positive class (e.g., *poisonous* mushrooms).
- $1-p$ is the probability that the sample belongs to the negative class (e.g., *edible* mushrooms).

Entropy Entropy is another measure of uncertainty or impurity in the data after a split. It is based on a concept from information theory that measures the degree of disorder or randomness in a variable. It is calculated as:

$$\psi(p) = -p\log_2(p) - (1-p)\log_2(1-p)$$

Square Root Function ψ_4 An additional function considered is:

$$\psi_4(p) = \sqrt{p(1-p)}$$

This function offers an alternative to the Gini function and entropy for measuring the impurity or uncertainty of a split.

3.2 Selecting the Best Split

At each node, the decision tree algorithm evaluates all possible splits across all features to determine the one that results in the greatest reduction in impurity. This is calculated by the impurity reduction, also known as information gain:

$$\text{Information Gain} = \phi_{\text{parent}} - \left(\frac{N_{\text{left}}}{N}\phi_{\text{left}} + \frac{N_{\text{right}}}{N}\phi_{\text{right}}\right)$$

where:

- ϕ_{parent} is the impurity of the parent node.
- ϕ_{left} and ϕ_{right} are the impurities of the left and right child nodes, respectively.
- N is the total number of samples at the parent node.
- N_{left} and N_{right} are the numbers of samples in the left and right child nodes.

3.3 Implementation Approach

The decision tree was implemented recursively, starting from the root node and expanding by selecting the best splits according to the impurity measure and splitting criterion. At each step, the following actions were performed:

1. **Calculate Node Impurity**: Compute the impurity of the current node using the chosen impurity measure.
2. **Evaluate Possible Splits**: For each feature, consider all possible splits and calculate the resulting impurity.
3. **Select Best Split**: Choose the feature and threshold that result in the highest impurity reduction.
4. **Apply Stopping Criteria**: Check if any stopping criteria are met (e.g., maximum depth reached, minimum samples per leaf).
5. **Split the Data**: Partition the data into left and right subsets based on the best split.
6. **Create Child Nodes**: Recursively repeat the process for each child node.

Implementing multiple impurity measures allows for flexibility in optimizing the decision tree model. By comparing different measures, one can select the impurity function that best fits the dataset and improves the model's predictive performance.

4 Training the Decision Trees

This section details the training process of the decision trees, focusing on the splitting and stopping criteria applied, as well as the calculation of the training error using the 0-1 loss function.

4.1 Splitting Criteria

The splitting criteria used during the training of the decision trees are based on the impurity measures discussed in Section 3. At each node, the algorithm evaluates all possible splits across all features to determine the one that results in the greatest reduction in impurity.

4.2 Stopping Criteria

To prevent the decision trees from overfitting the training data, several stopping criteria were implemented:

- **Maximum Tree Depth**: A maximum depth d_{max} was set to limit the number of levels in the tree.
- **Minimum Number of Samples Required to Split**: A threshold $n_{\text{min\_samples\_split}}$ was established.
- **Pure Nodes**: If a node became pure, containing samples of only one class, no further splitting was performed.

The specific values for d_{max} and $n_{\text{min\_samples\_split}}$ were determined empirically through hyperparameter tuning, as discussed in Section 5. These stopping criteria balance the trade-off between model complexity and generalization ability.

4.3 Calculation of Training Error (0-1 Loss)

The training error was evaluated using the 0-1 loss function, which measures the proportion of misclassified samples. The 0-1 loss is defined as:

$$l(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{if } y \neq \hat{y} \end{cases}$$

The training error is defined as:

$$l(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} l(y_i, \hat{y}_i)$$

where:

- N is the total number of training samples.
- y_i is the true class label of sample i.
- $\hat{y}_i$ is the predicted class label of sample i.

5 Hyperparameter Tuning

This section describes the methodology used for hyperparameter tuning of the decision tree classifiers and presents the results and analysis of the tuning process.

5.1 Methodology

The performance of decision tree classifiers is highly dependent on the choice of hyperparameters. To optimize the models, we conducted comprehensive hyperparameter tuning using cross-validation on the training set. The following hyperparameters were considered:

- **Impurity Measure**: Gini function, entropy, and the alternative function ψ_4.
- **Maximum Tree Depth**: Values tested were 3, 5, 9, and None (no limit).
- **Minimum Samples Required to Split**: Values tested were 2, 5, and 10.

Cross-Validation Procedure To evaluate the performance of different hyperparameter combinations, 5-fold cross-validation was employed. The training set was partitioned into five equal subsets. In each iteration, one subset was used as the validation set, and the remaining four subsets were used for training. This process was repeated five times, ensuring that each subset served as the validation set once.

5.2 Results and Analysis

The mean cross-validation accuracies for all tested hyperparameter combinations were recorded. Due to the extensive number of results, only the best-performing findings for each impurity measure are presented here. The complete results are available in Appendix A.

Table 2: Cross-Validation Results for Decision Trees with Different Hyperparameters

Impurity Measure	Max Depth	Min Samples Split	Mean CV Accuracy (%)
Gini	None	10	99.77
Entropy	None	5	99.74
ψ_4	None	10	99.66

Best Hyperparameter Combination The hyperparameter combination that achieved the highest mean cross-validation accuracy was:

- **Impurity Measure**: Gini
- **Maximum Tree Depth**: None (no limit)
- **Minimum Samples Required to Split**: 10
- **Mean Cross-Validation Accuracy**: 99.77%

Impact of Hyperparameters

Impurity Measure: The Gini function achieved the highest mean cross-validation accuracy, closely followed by Entropy and the alternative function ψ_4. This suggests that all three impurity measures are effective for this dataset, with Gini having a slight edge.

Maximum Tree Depth: Allowing the tree to grow without a maximum depth constraint resulted in higher accuracies, indicating that deeper trees can capture more complex patterns in the data.

Minimum Samples Required to Split: A minimum samples split of 10 provided the best performance, possibly due to a balance between preventing overfitting and allowing sufficient splits to capture data nuances.

Results For the selected model, the training error was calculated using the 0-1 loss function, which measures the proportion of misclassified samples.

The model achieved a training accuracy of 99.97% and a test accuracy of 99.82%, with a training error of 0.03%. The close alignment between training and test accuracies suggests that the model generalizes well to unseen data, with minimal signs of overfitting despite the high training performance.

6 Model Evaluation and Discussion

This section presents the evaluation of the best-performing decision tree model on both the training and test sets. It discusses the performance metrics, confusion matrices, and provides an analysis of the model's behavior in terms of overfitting and generalization.

6.1 Evaluation on the Test Set

The selected decision tree model, utilizing the Gini impurity measure with no maximum depth and a minimum of 5 samples required to split, was evaluated on the test set to assess its generalization performance.

Performance Metrics

The model's performance on the test set is summarized in Table 3.

Table 3: Performance Metrics on the Test Set

Metric	**Value (%)**
Accuracy	99.82
Precision	99.80
Recall	99.83
F1-Score	99.81

Confusion Matrix

The confusion matrix for the test set is presented in Table 4.

Table 4: Confusion Matrix on the Test Set

	Predicted Edible	**Predicted Poisonous**
Actual Edible	3945	7
Actual Poisonous	6	3432

6.2 Evaluation on the Training Set

Performance Metrics

The model's performance on the training set is summarized in Table 5.

Table 5: Performance Metrics on the Training Set

Metric	**Value (%)**
Accuracy	99.97
Precision	99.96
Recall	99.97
F1-Score	99.97

Confusion Matrix

The confusion matrix for the training set is presented in Table 6.

Table 6: Confusion Matrix on the Training Set

	Predicted Edible	**Predicted Poisonous**
Actual Edible	16047	5
Actual Poisonous	4	13502

6.3 Overfitting and Underfitting Analysis

To evaluate whether the model is overfitting or underfitting, we compare the training and test performance metrics.

- **Training Accuracy**: 99.97%
- **Test Accuracy**: 99.82%

The minimal difference between them (approximately 0.15%) suggests that the model generalizes well to unseen data and is not significantly overfitting.

7 Implementation of the Random Forest

This section details the implementation of the Random Forest algorithm, focusing on its integration with the existing decision tree classifier, the theoretical underpinnings of the method, and the evaluation of its performance compared to individual decision trees.

7.1 Integration with the Decision Tree Class

Random Forests are ensemble learning methods that operate by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) of the individual trees. The primary motivation behind Random Forests is to improve the predictive performance and stability of decision trees by reducing overfitting and variance.

The Random Forest algorithm introduces two main concepts to enhance the performance of decision trees:

- **Bootstrap Aggregation (Bagging)**: This involves training each decision tree on a different random subset of the training data, selected with replacement (bootstrap samples). Bagging reduces variance by averaging the models, which helps in mitigating the overfitting that individual trees might exhibit.
- **Random Feature Selection (Feature Subspacing)**: At each split in a decision tree, a random subset of features is considered instead of the full set. This decorrelates the trees in the ensemble, further reducing variance. For classification tasks, it is common to use $\sqrt{p}$ features at each split, where p is the total number of features.

  To implement feature subspacing, each decision tree considers only a random subset of features when determining the best split at each node. Specifically, the number of features m to consider at each split was set to $\sqrt{p}$:

$$m = \sqrt{p}$$

Each tree in the forest was initialized with the same hyperparameters (e.g., impurity measure, maximum depth, minimum samples to split) as the optimal decision tree identified previously. The `DecisionTree` class was used to ensure consistency in the tree construction process.

For classification, the Random Forest aggregates the predictions of all individual trees by majority voting. This means that for each sample, the predicted class is the one that receives the most votes from the ensemble.

Hyperparameters Used The Random Forest was configured with the following hyperparameters:

- **Number of Estimators ($n_{\textbf{estimators}}$)**: Set to 10, representing the number of decision trees in the ensemble.
- **Maximum Depth (max_depth)**: Set to None, allowing trees to grow until all leaves are pure or contain fewer samples than the minimum required for splitting.
- **Minimum Samples Split (min_samples_split)**: Set to 2, the minimum number of samples required to split an internal node.
- **Impurity Measure**: The Gini impurity measure was used, consistent with the best-performing decision tree model.
- **Maximum Features (max_features)**: Set to 'sqrt', meaning each tree considers $\sqrt{p}$ features when looking for the best split.

7.2 Performance Evaluation

The Random Forest model was evaluated using the same training and test sets as the individual decision tree to ensure a fair comparison. The evaluation focused on key performance metrics and confusion matrices to assess the model's effectiveness.

7.2.1 Performance Metrics

The performance of the Random Forest on both the training and test sets is summarized in Tables 7 and 8 respectively.

Table 7: Random Forest Performance Metrics on the Test Set

Metric	**Value (%)**
Accuracy	99.95
Precision	100.00
Recall	99.88
F1-Score	99.94

Table 8: Random Forest Performance Metrics on the Training Set

Metric	**Value (%)**
Accuracy	100.00
Precision	100.00
Recall	99.99
F1-Score	100.00

7.2.2 Confusion Matrix

The confusion matrices for both the test and training sets provide a detailed breakdown of the model's predictions.

Table 9: Random Forest Confusion Matrix on the Test Set

	Predicted Edible	**Predicted Poisonous**
Actual Edible	3952	0
Actual Poisonous	4	3434

Table 10: Random Forest Confusion Matrix on the Training Set

	Predicted Edible	**Predicted Poisonous**
Actual Edible	16052	0
Actual Poisonous	1	13505

7.3 Interpretation of Results

The Random Forest model exhibits outstanding performance metrics, achieving an accuracy of 99.95% on the test set and 100% on the training set. The precision, recall, and F1-score values are near-perfect, indicating that the model excels in correctly classifying both edible and poisonous mushrooms with minimal errors.

The implementation of the Random Forest model offers several benefits over individual decision trees:

- **Improved Accuracy**: By aggregating the predictions of multiple trees, Random Forests can achieve higher accuracy and robustness.
- **Reduced Overfitting**: The combination of bagging and random feature selection minimizes the risk of overfitting, ensuring better generalization to unseen data.
- **Feature Importance Insights**: The ability to compute feature importances aids in understanding the underlying data and identifying key predictors.

- **Stability**: Random Forests are less sensitive to variations in the training data compared to single decision trees.

8 Conclusion

In this project, we successfully implemented decision tree classifiers and a Random Forest from scratch to classify mushrooms as edible or poisonous based on their characteristics. Through careful data preprocessing and handling of missing values and duplicates, we prepared a clean dataset suitable for model training.

We experimented with different impurity measures, including Gini, Entropy, and the alternative function ψ_4, and conducted extensive hyperparameter tuning using cross-validation. The best-performing decision tree used the Gini impurity measure with no maximum depth and a minimum of 5 samples required to split, achieving a mean cross-validation accuracy of 99.77%.

Evaluation on the test set demonstrated that the decision tree model generalized well to unseen data, with an accuracy of 99.82% and minimal overfitting. The confusion matrices and performance metrics indicated that the model effectively distinguished between edible and poisonous mushrooms.

The Random Forest implementation further improved the model's performance, achieving an accuracy of 99.95% on the test set. The ensemble method reduced variance and enhanced generalization, confirming the benefits of using multiple decision trees and random feature selection.

Overall, the project demonstrated the effectiveness of decision trees and Random Forests in classification tasks with categorical data.

References

[1] Dua, D. and Graff, C. (2019). *UCI Machine Learning Repository* [`http://archive.ics.uci.edu/ml`]. Irvine, CA: University of California, School of Information and Computer Science.

A Cross-Validation Results

Table 11: Cross-Validation Results for All Hyperparameter Combinations

Impurity Measure	**Max Depth**	**Min Samples Split**	**Mean Accuracy (%)**
gini	3	2	72.68
gini	3	5	72.71
gini	3	10	72.71
gini	5	2	79.02
gini	5	5	79.41
gini	5	10	79.22
gini	9	2	92.69
gini	9	5	92.49
gini	9	10	92.49
gini	None	2	99.74
gini	None	5	99.77
gini	None	10	99.73
entropy	3	2	72.69
entropy	3	5	72.72
entropy	3	10	72.70
entropy	5	2	77.43
entropy	5	5	77.34
entropy	5	10	77.78
entropy	9	2	88.67
entropy	9	5	89.29
entropy	9	10	87.47
entropy	None	2	99.70
entropy	None	5	99.74
entropy	None	10	99.69
sqrt	3	2	58.60
sqrt	3	5	59.53
sqrt	3	10	60.00
sqrt	5	2	65.51
sqrt	5	5	65.29
sqrt	5	10	65.63
sqrt	9	2	79.13
sqrt	9	5	78.91
sqrt	9	10	78.62
sqrt	None	2	99.65
sqrt	None	5	99.66
sqrt	None	10	99.66