Poisonous or Edible Mushroom?

Campregher Sofia

May 17, 2025

Abstract

This project presents a custom decision tree which is implemented entirely from scratch, using only single-feature binary tests at each internal node. Firstly, the dataset is processed in order to handle and encode missing values; columns with more than 50

The initial static analysis focuses on building a constructor which only accepts one of the following impurity measures - Gini Index, Entropy, or Scaled Entropy as the metric for evaluating splits, while all the other parameters are fixed. The highest Test Accuracy is reached by the Gini index model. Both Entropy and Scaled Entropy yielded identical test accuracy and zero-one loss scores. The Gini index offers a small advantage in terms of classification accuracy. The computational costs, given the manual implementation, are analyzed by the Training time analysis; where Gini required 65.64 seconds, and Entropy and Scaled Entropy 58.59 and 57.58 seconds, respectively.

The model performed a hyperparameter tuning that is implemented by varying the maximum depth from [3,9] and following a different splitting criterion. The optimal configuration is obtained using the Gini index as a splitting criterion and a maximum depth of 9. This model recorded a training test accuracy of 0.839201 and an overfitting gap of just 0.000079; showing no signs of overfitting. Both Entropy and Scaled Entropy reached a maximum test accuracy of 0.799.

Performance is visualized by a heat map that displays test accuracy as a function of the maximum depth and the splitting criterion used in the decision tree. The test accuracy decreases as the maximum depth of the tree decreases, which means that trees of a lower depth are less effective in capturing the complexity of the data set across all three parameters. Overall, the project demonstrates that despite the high computational costs, a model implemented from scratch can achieve great results in terms of accuracy.

1 Introduction

The decision tree aims at creating a model that estimates the value of a target variable by learning decision rules derived from the dataset features. This technique is widely used because of:

- Its simplicity to understand and interpret
- It requires little data preparation
- It is able to handle both numerical and categorical data
- It performs well on large datasets

This project offers a custom decision tree which is implemented entirely from scratch, using only single-feature binary tests at each internal node. The initial static analysis focuses on building a class structor which contains several parameters which controls the behavior and growth of the model. The constructor accepts one of the following impurity measures - Gini Index, Entropy, or Scaled Entropy as the metric for evaluating splits, while all the other parameters are fixed. The model performs a hyperparameter tuning which is implemented to assess how the adoption of different combinations of splitting criteria and maximum tree depth influence accuracy and generalization. In the last section of the study, training and test set are analyzed using accuracy scores, overfitting gap, and confusion matrices to draw conclusion about its performance.

2 DataSet

2.1 Mushroom Dataset Overview

The Mushroom Dataset is make up by a collection of 61,068 instances, hypothetical mushrooms entities, one class and 20 attributes, which describe the properties of mushrooms; each instance is represented by a set of features. All this features, used to train the model, can be categorical or continuous [17 nominal and 3 metrical] and describe various physical and chemical attributes of mushrooms. Below, a sample of several attributes included in the dataset:

- Cap Shape (e.g., bell, conical, flat)
- Cap Surface (e.g., smooth, scaly)
- Cap Color (e.g., red, green, yellow)
- Bruises (whether the mushroom bruises when handled)
- Odor (e.g., almond, anise, foul, none)
- Gill Attachment (e.g., attached, free)
- Gill Spacing (e.g., close, crowded)
- Gill Size (e.g., narrow, broad)
- Stalk Shape (e.g., enlarging, tapering)
- Stem Height (float value in cm)
- Stem Width (float value in mm)
- Cap Diameter (float value in cm)

Unnamed: 0	cap-diameter	cap-shape	cap-surface	cap-color	does-bruise-or-bleed	gill-attachment	gill-spacing	gill-color	stem-height	...	stem-root	stem-surface	stem-color	veil-type	veil-color	has-ring	ring-type	spore-print-color
0	15.26	x	g	o	f	e	NaN	w	16.95	...	s	y	w	u	w	t	g	NaN
1	16.60	x	g	o	f	e	NaN	w	17.99	...	s	y	w	u	w	t	g	NaN
2	14.07	x	g	o	f	e	NaN	w	17.80	...	s	y	w	u	w	t	g	NaN
3	14.17	f	h	e	f	e	NaN	w	15.77	...	s	y	w	u	w	t	p	NaN
4	14.64	x	h	o	f	e	NaN	w	16.53	...	s	y	w	u	w	t	p	NaN

Figure 1: Dataset Composition

Since and the goal is to predict and to then classify each mushroom as either edible or poisonous based on these features, the target variable for the Mushroom Dataset is a binary classification label:

- Edible (represented as e)
- Poisonous (represented as p)

The data from the UCI Machine Learning Repository is a structured data frame of information; where each column corresponds to a particular feature, and each row represents a mushroom instance.

The data is split into two parts:

- Features (X): the input data, - categorical or continuous features.
- Targets (y): the classification label - edible or poisonous.

As mentioned before, features can be categorical [eg. cap-shape] or continuous [eg. cap-diameter]. Nominal attributes are encoded, in the row file, as a single letters, instead, metrical attributes are represents as an interval of floats number. The "nan" values indicate missing values and uncoded letter (such ad "d" for Cap-Shape) means that the specific one-digit code is is absent from the metadata file. Due to page restriction the table shows only a sample of three examples and half column [10 out of 20].

The Dataset includes:

- three numerical attributes: cap diameter [avg 6.7 cm], stem height [avg 6.58cm] and stem width[avg 12.5cm];
- Missing data in several features: stem root [84.39

The distribution of categorical variables is displayed by 12 plots. Each of them show the frequency of values for each feature in the dataset, but also detect the dominant classes and potentially underrepresented categories.

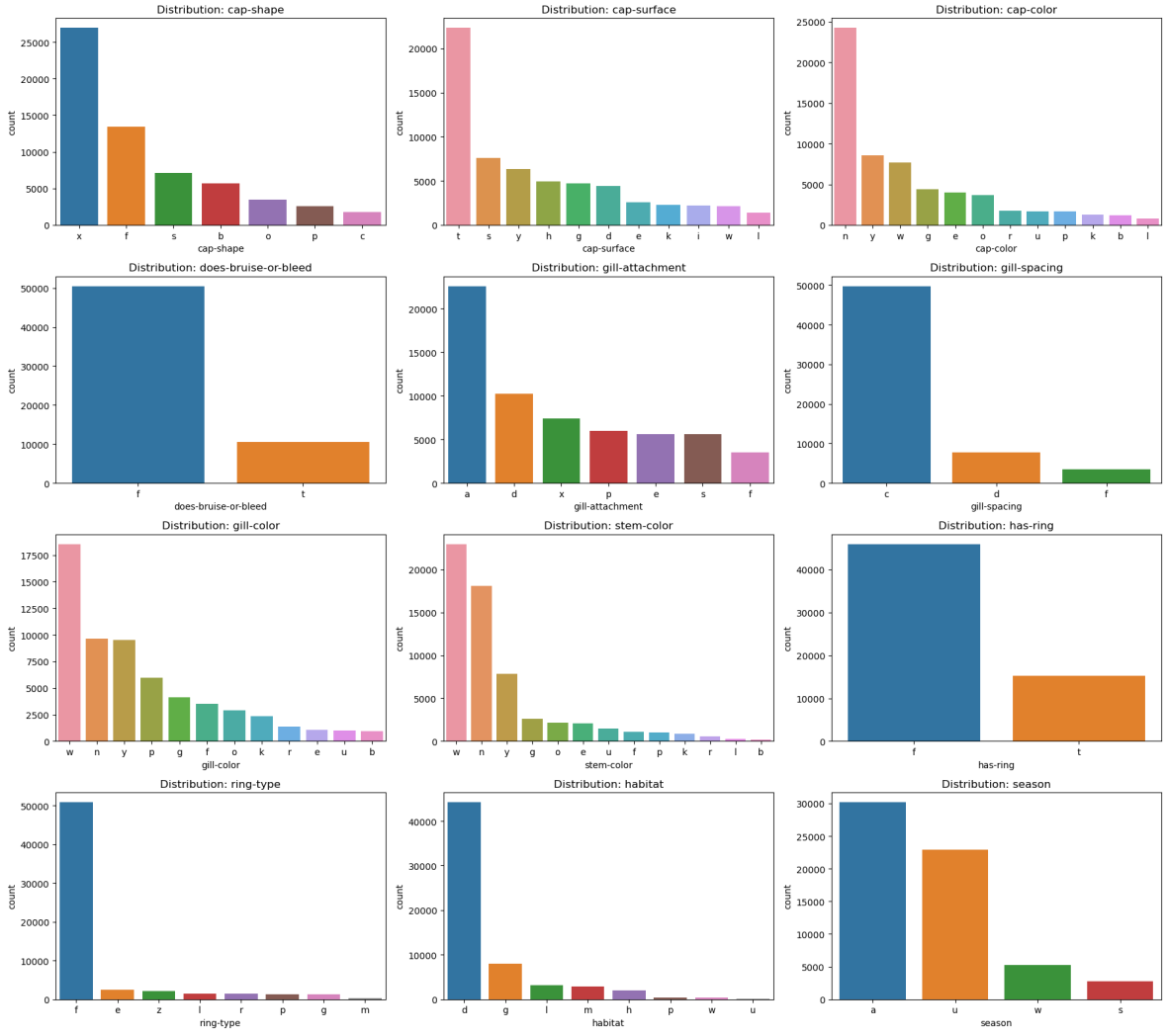

Figure 2: Distribution of categorical features

The distribution of numerical variables is explored through histogram plots to asses the shape, tendency, trend and spread of each features. Histograms are combined with Kernel Density Estimation (KDE) - which provides approximation of the data distribution. All three variables present a right-skewed distribution, where most observations are concentrated in the lower range values.

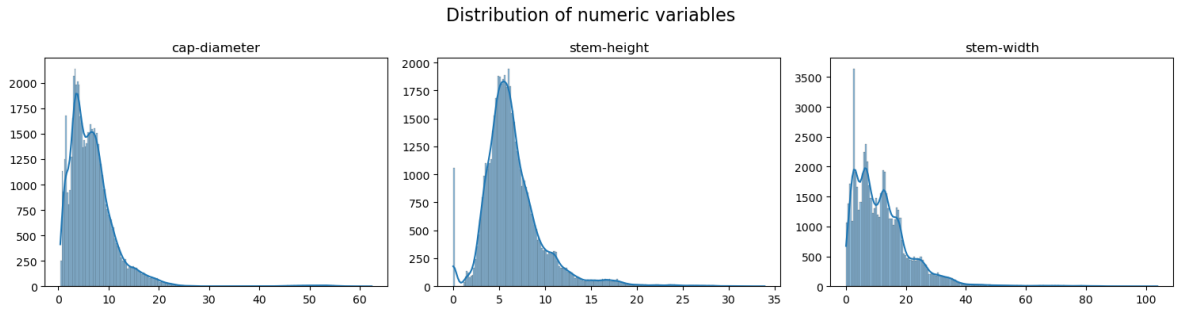

Figure 3: Distribution of numerical features

The analysis focuses exclusively on the Secondary Mushroom Dataset, as it is more comprehensive and it offered an enriched collection of features. The secondary dataset includes both numerical and categorical features which are not present in the primary. Moreover, the secondary dataset is a more recent release (2022).

2.2 Statistical Property

The DataSet is largely balanced, indeed, considering the class distribution the percentage of mushrooms being poisonous is 55.49

The following heatmap illustrates the correlation matrix for all variables in the dataset, including the target variable - class. All categorical features were encoded into numerical format using integer number. Correlation values goes form [-1, 1], where -1 means perfect negative correlation and 1 perfect positive correlation.

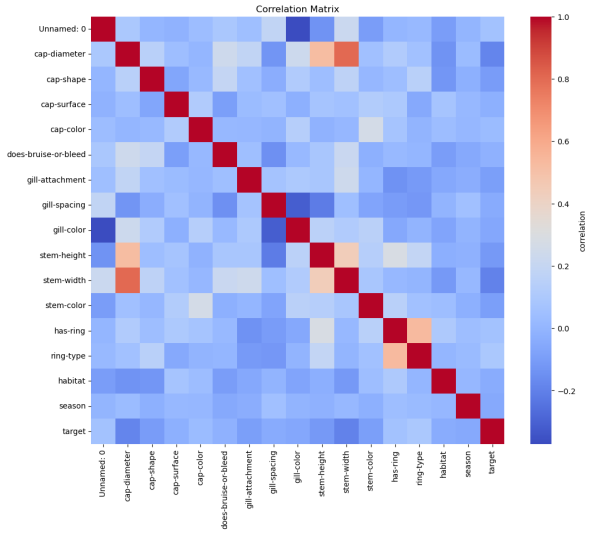

Figure 4: Correlation

The diagonal displays a perfect positive correlation for each variable with itself, and off diagonal elements exhibit weak or no correlation. The only couples that exhibits some positive structural dependency are cap-diameter - stem-height, and cap-diameter - stem-width. The target attributes, class - shows very weak correlation with most predictors.

A specific insight into the numerical variable correlation is exhibits by the following heatmap:

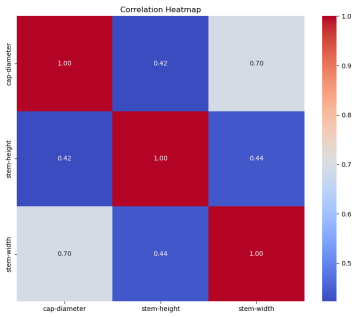

Figure 5: Correlation of numerical features

The annoyed value represents the strength and direction of their linear relationships. The highest is observed by the couple cap-diameter - stem-width, suggesting that mushrooms with larger caps also tend to have thicker stems. A moderate positive and negative correlation is reached by stem-height - cap-diameter and stem-height - stem-width respectively.

3 Methodology

In this section, we describe the methodology behind the decision tree predictor implemented in Python. The model is based on a binary tree structure, where each node represents a decision point. The tree construction follows a recursive splitting approach, using impurity measures such as Gini Impurity, Entropy and Scaled Entropy to evaluate the quality of a split.

3.1 Data Preparation

The DataSet, which is downloaded and stored locally to avoid redundant downloads, is returned as a dictionary.

The first step of the process involves handling and encoding missing values. Since some categorical and numerical attributes contain missing values, columns with more than 50

The number and percentage of missing values were calculated for each column:

	Missing Values	Missing Percentage
cap-surface	14.120	23%
gill-attachment	9.884	16%
gill-spacing	25.063	41%
stem-root	51.538	84%
stem-surface	38.124	62%
veil-type	57.892	94%
veil-color	53.656	87%
ring-type	2.471	4%
spore-print-color	54.715	89%

Figure 6: Missing Value and Missing Percentage

And visually represented by an horizontal bar plot to display the extent of missingness across features.

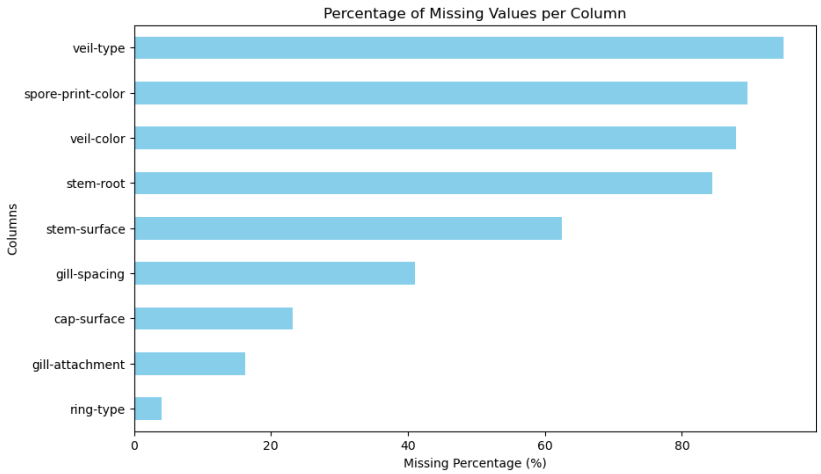

Figure 7: Missing Value

Columns with more than 50Stem-root, Stem-surface, Veil-type, Veil-color, Spore-print-color are considered unreliable. For the remaining attributes, missing entries are replaced using the mode to preserve categorical consistency. Which are:

- t for cap-surface
- a for gill-attachment
- c for gill-spacing
- f for ring-type

The mode approach avoids bias might result from mean or median imputation or by dropping the entire column.

3.2 Tree Predictors (Theory)

A tree predictor follows the structure of an ordered, rooted tree where each node could be a leaf or an internal node. The difference between these two is that the first - leaf - does not have any children; on the contrary, the second - internal or test node - has at least two children. The children of each internal node are arranged in a specific sequence, meaning they are indexed consecutively.

An example:

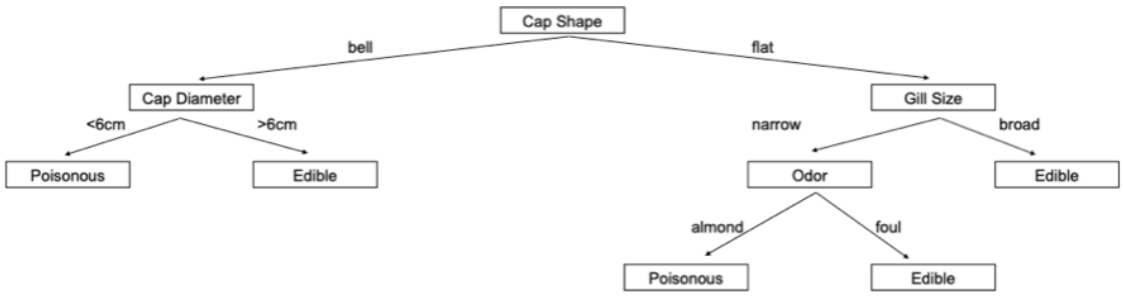

Figure 8: Tree Predictor

The previous decision tree predicts mushrooms editability based on three different attributes: cap shape, cap diameter, gill size and odor. Cap shape and gill size are the primary decision factors; cap diameter and gill size are not directly linked in the examples, indeed they are treated as independent

risk factors for heart attack. Gill size and odor are linked, indeed when the mushroom present a flat cap shape and a narrow gill size, then the odor could be foul or almond; each of which bring to a different final result. The three attributes are internal node, where cap shape has three internal nodes and cap diameter and gill size only two. Poisonous and edible are leaf, because, as the image shows, they do not have any children.

Our particular case consider a binary classification where all internal nodes have exactly two children. The process starts with a single node tree, a leaf, which classifies all data point by assigning the most frequent label. The leaf node it is transformed into an internal node from which two new leaves are added. The two new leaves can remain so or can be transformed into internal nodes and restart recursively the model until a stopping condition is met.

3.3 Impurity Measures (Theory)

The algorithm used in a decision tree operates aims of minimizing the impurity of the split; in other word, ensuring that the chosen node-question pair (nt,qt) leads to the greatest reduction in impurity according to a given impurity function F. This approach helps prevent overfitting, that is when the number of tree nodes grow to much compared to the cardinality of the training set. Impurity function must satisfy certain condition, that capture the notion of "impurity". (Kearns and Mansour, 1999)

The approach used in this paper includes the widely used functions of Entropy, Scaled Entropy and Gini-index.

The following impurity measures are used to evaluate the quality of a split in the decision tree:

- **Standard Entropy:**

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

  Entropy reaches its maximum when $p = 0.5$, indicating maximum uncertainty when both classes are equally likely.

- **Scaled Entropy:**

$$H_{\text{scaled}}(p) = -\frac{p}{2}\log_2(p) - \frac{(1-p)}{2}\log_2(1-p)$$

  This is a scaled-down version of the standard entropy. It also reaches its maximum value of 0.5 at $p = 0.5$.

- **Gini Index:**

$$G(p) = 2p(1-p)$$

  Gini Index also peaks at $p = 0.5$, but increases less sharply near the extremes ($p = 0$ or $p = 1$), making it less sensitive to changes in class probability.

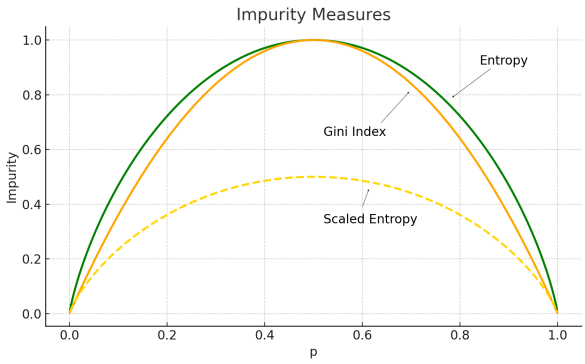

Figure 9: Impurity Measures in comparison

Both Gini Index, Scaled Entropy and Entropy are strictly concave, ensuring a monotonic decrease in impurity after each split, and differentiable, which facilitates numerical optimization during tree construction.

In particular:

- Gini index measures how often a randomly chosen element of a set would be incorrectly labeled if it were labeled randomly and independently according to the distribution of labels in the set. In other word, how mixed the labels are in a node. A lower Gini index indicates a purer node, meaning that all samples in the node are associated with a single target category.
- Entropy measures the level of disorder in the labels. Higher entropy means a more diverse set of label, instead lower entropy means a greater homogeneity.
- Scaled Entropy is a variation of standard entropy; it presents the same shape but compresses by half. It measures the level of disorder in the label, and it reaches 0.5 - its maximum - when classes are equally represented. Scaled entropy is less sensitive to disorder compared to standard entropy.

They follow similar trend but Gini index remains consistently lower than entropy, while scaled entropy follows the same curve as entropy but at a lower scale.

3.4 Tree Structure (Code)

The binary decision tree is implemented entirely from scratch; using only single-feature binary tests at each internal node. A custom TreeNode class represents each node in the tree, storing its splitting condition, child nodes and predicted classes while the node is a leaf. The model includes a section used to determine whether a node represents a leaf or not.

The DecisionTree class contains several parameters which controls the behavior and growth of the model, making it the main class of the prediction model. The class supports the entire training, prediction and visualization metrics of the project. It adopts a flexible configuration through several stopping criteria maximum depth, minimum samples per node, maximum number of leaves, and an impurity threshold. The constructor accepts one of the following impurity measures - Gini Index, Entropy, or Scaled Entropy as the metric for evaluating splits. The training algorithm builds the tree recursively. For each node, it first checks for stopping condition. If not, it evaluates all possible binary splits across all features `best_split`.The best split is determined by the maximum information gain computed. In other words, tree is developed by repeatedly selecting the best split that maximizes information gain `information_gain`, accordingly to the chosen impurity measure. Single-feature test x[features ¡= threshold are used to make decision at internal node. Categorical fears are converted into numeric using one-hot encoding, according to the model requirement. The prediction [predict] traverses the decisione tree for each data sample and continuous until a leaf node is reached. Zero-one loss and accuracy are used to evaluate the model performance. The evaluation metrics are performed both on training and test set in order to assess overfitting, undercutting and how well the model generalizes.

The second section performed a stratified train test and allocates 80

After data are recalled by main function, tree is initialized through controlled parameters in a static situation. The decisional tree is implemented with a maximum depth of 5, a minimum sample split of 5, an entropy threshold of 0.1, iterating for the three splitting criterion. The training process is timed in order to define performance. The training time are relatively high, due to the dataset size and the fact that decision tree is implemented from scratch. Indeed, results highlight the computational cost which are associated with a manual implementation. The training durations for each splitting criterion is:

Splitting Criterion	**Training Time (sec)**
Gini Index (GI)	**59.35**
Entropy (E)	**62.05**
Scaled Entropy (SE)	**56.78**

Table 1: Training time comparison for different impurity criteria

Moreover, accuracy and zero-one loss are computed to quantify the quality of the prediction. The model correctly classifies around 3 out of 4 data points, indicating a solid performance at generalizing without overfitting. Nearly 1 out of 4 data points are incorrectly classified. The highest Test Accuracy

is reached by the Gini Index model. Both Entropy and Scaled Entropy yielded identical test accuracy and zero-one loss scores. Gini Index offers a small advantages in terms of classification accuracy.

Splitting Criterion	Test Accuracy	Zero-One Loss
Gini Index (GI)	**0.7337**	**0.2663**
Entropy / Scaled Entropy	**0.7256**	**0.2744**

Table 2: Test accuracy and zero-one loss for different splitting criteria

Confusion matrix are displayed to provide a visual representation of actual and predicted values. The model is more accurate when predicting 1 compared to class 0. More specifically, considering the Gini Index model, out of the total actual class 0 samples, 4976 were correctly classified (TN), while 460 were incorrectly predicted as class 1 (FP). For actual class 1, 3986 data points were correctly predicted (TP), while 2792 were misclassified as class 0 (FN). Similar pattern are recorded for Entropy and Scaled entropy, which yielded identical results.

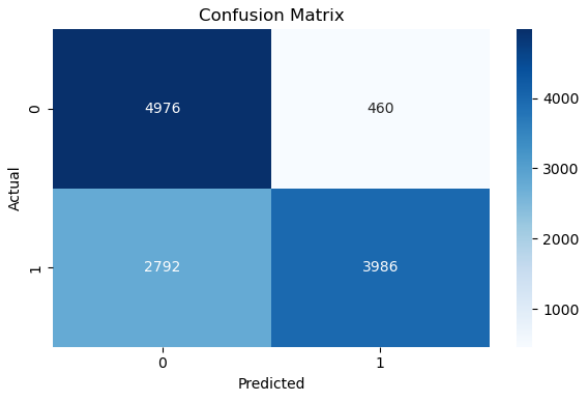

Figure 10: Confusion Matrix - Gini Index

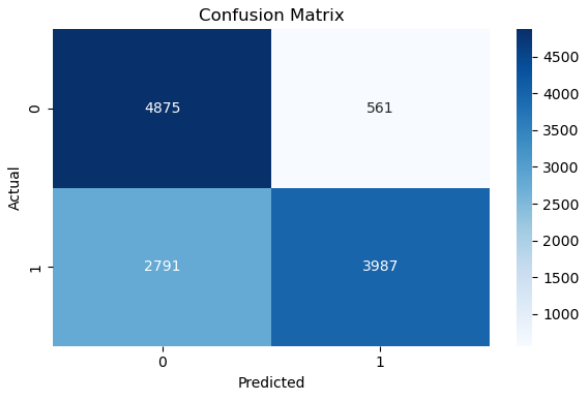

Figure 11: Confusion Matrix - Entropy and Scaled Entropy

Finally, the model produced the decision tree visualization. The binary decision paths, thresholds, and class predictions at leaf nodes are clearly illustrated.

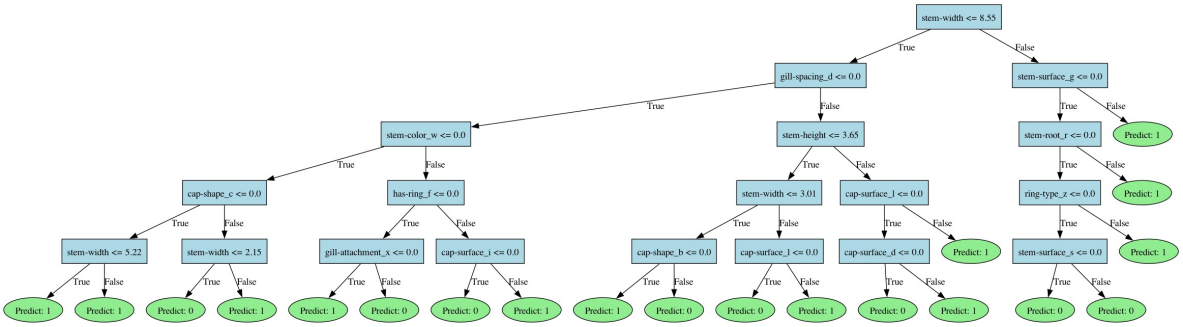

Figure 12: Tree Visualization - Gini Index

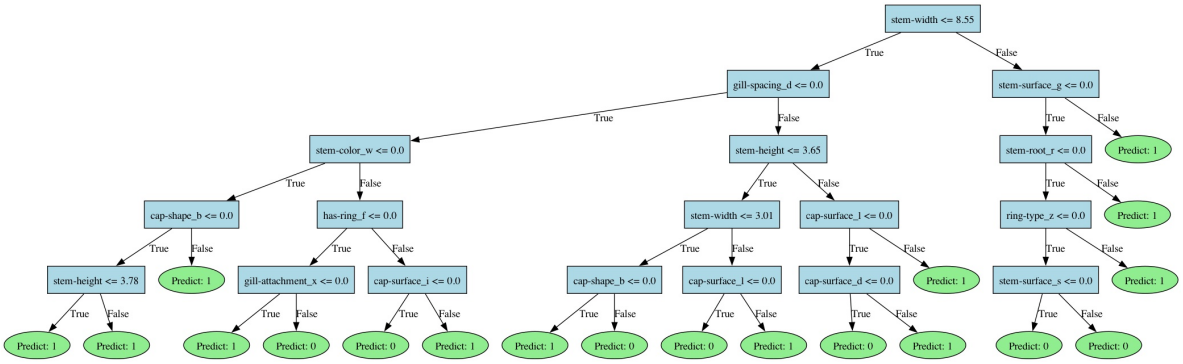

Figure 13: Tree Visualization - Entropy

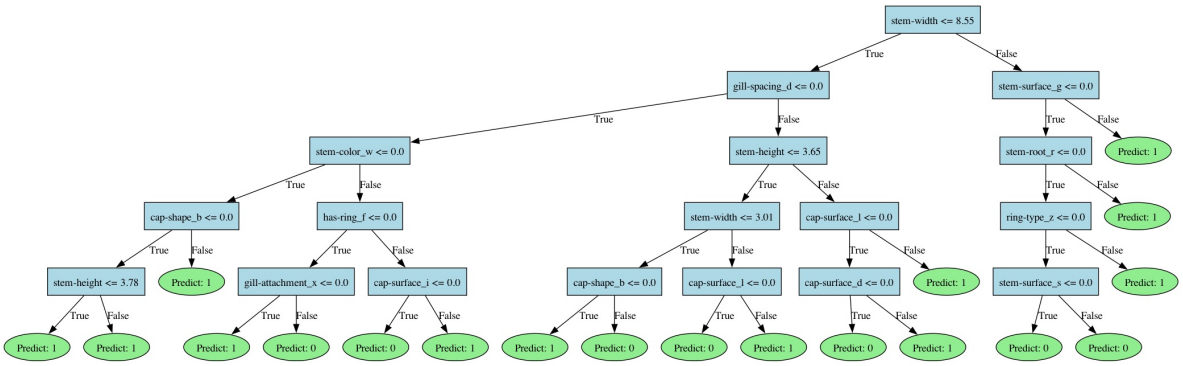

Figure 14: Tree Visualization - Scaled Entropy

The Hyperparameter Tuning procedure is used to evaluate the impact of choosing different splitting criteria and tree depths. In particular, the performance of each model is evaluated across a range of maximum tree depths from 3 to 9 and three different splitting strategies. Each pair of configuration is trained on the dataset and evaluated using accuracy on both the training and test sets. The overfitting gap is used to assess the quality of generalization, which is define as the difference between training and test accuracy.

Criterion	Max Depth	Train Accuracy	Test Accuracy	Overfitting Gap
entropy	2	0.652973	0.650565	0.002408
entropy	3	0.681957	0.677092	0.004865
entropy	4	0.709631	0.710666	-0.001035
entropy	5	0.723836	0.725561	-0.001725
entropy	6	0.740477	0.742836	-0.002359
entropy	7	0.773432	0.775831	-0.002399
entropy	8	0.781783	0.783363	-0.001580
entropy	9	0.797012	0.799001	-0.001990
gini	2	0.652973	0.650565	0.002408
gini	3	0.681957	0.677092	0.004865
gini	4	0.713172	0.714508	-0.001336
gini	5	0.731471	0.733748	-0.002277
gini	6	0.748787	0.750287	-0.001499
gini	7	0.784689	0.786393	-0.001703
gini	8	0.817173	0.816604	0.000569
gini	9	0.839280	0.839201	0.000079
scaled_entropy	2	0.652973	0.650565	0.002408
scaled_entropy	3	0.681957	0.677092	0.004865
scaled_entropy	4	0.709631	0.710666	-0.001035
scaled_entropy	5	0.723836	0.725561	-0.001725
scaled_entropy	6	0.740477	0.742836	-0.002359
scaled_entropy	7	0.773370	0.775831	-0.002461
scaled_entropy	8	0.781701	0.783363	-0.001662
scaled_entropy	9	0.796930	0.799001	-0.002071

Table 3: Comparison of train/test accuracy and overfitting gap for different criteria and tree depths.

Accuracy is visualized by a matrix where rows represent the splitting criterion, columns represent the maximum tree depth, and cell color intensity encodes test accuracy. The visualization highlights the trade-offs between performance and complexity.

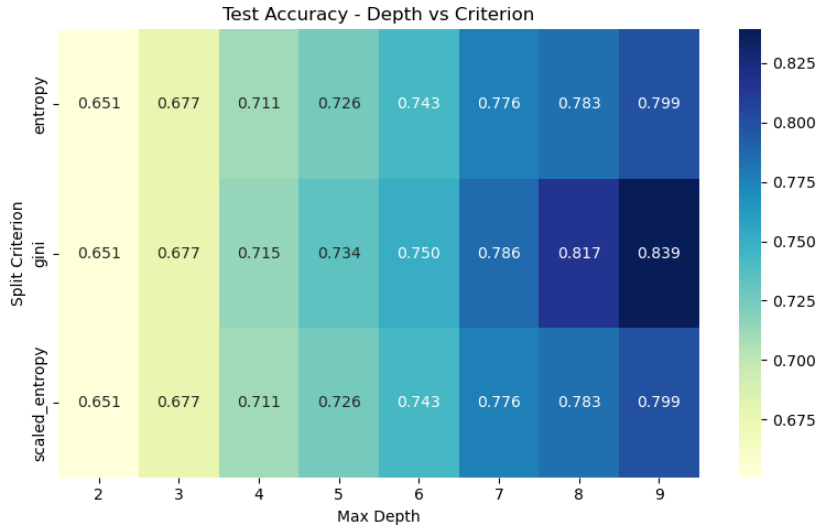

Figure 15: Hyperparameter Tuning

The figure displays test accuracy as a function of the maximum depth and splitting criterion used in the decision tree. Test accuracy decrease as the tree's maximum depth decrease, meaning that a a lower-depth trees is less effective at capture the complexity of dataset across all three parameters. Gini - considering all three criterion - outperforms the other two at each level of depth. The best performance is recorded at depth 9 with a test accuracy of 0.839. Both Scaled Entropy and Entropy display identical results; the highest accuracy is reached at 0.799.

The last section of the project aim at selecting the best model configuration considering key criteria: both accuracy and overfitting gap. The best-performing model is the model with the highest test accuracy, which reflects the model's ability to generalize to unseen data. The most-balanced model, with the lowest absolute overfitting gap, indicates a small distance between training and test performance. The optimal configuration correspond to the decision of using Gini Index as a splitting criterion and a maximum depth of 9. This model recorded a training test accuracy of 0.839201 and an overfitting gap of just 0.000079; showing no signs of overfitting. The same configuration is selected for both analysis, indicating that deeper trees do not compromise the ability to perform well on unseen data.

4 Conclusion

The model illustrates that a decision tree manually implemented is able to achieve competitive accuracy and generalization performance.

Despite the high computational costs incurred by the model, results demonstrate the quality and effectiveness of implementing a decision tree from scratch. Results highlight that Gini Index is the most accurate parameter while Entropy and Scaled Entropy performed similarly and but less effectively. Through systematic hyperparameter tuning, the optimal configuration correspond to the decision of using Gini Index as a splitting criterion and a maximum depth of 9. This model recorded a training test accuracy of 0.839201 and an overfitting gap of just 0.000079; showing no signs of overfitting and strong generalization capabilities.

References