Decision Trees and Hyper-parameter Tuning Machine Learning Project

Tommaso Premoli - Nr.34221A

October 18, 2024

Abstract In this analysis, a decision tree was developed for the binary classification of mushrooms, with the aim of determining whether they are poisonous or edible. The decision tree was implemented entirely *from scratch* using the programme 'Python', following an approach that allowed different splitting criteria to be explored in order to optimise the model. In particular, three splitting criteria were considered: entropy, Gini index and the Mean Squared Error, evaluated on the basis of accuracy and zero-one loss. After a comprehensive comparison, the tree that showed the best predictive performance was selected as the reference model. To ensure the robustness of the results, a cross-validation with K-fold was performed in order to verify the consistency of accuracy across multiple data partitions. Finally, the last part of the project was dedicated to hyper-parameter tuning, with the aim of further optimising the model, preventing the risk of underfitting and overfitting, and improving the overall generalisation capability of the classifier.

Objective of the analysis The main objective of this analysis is to build an effective predictive system for the binary classification of mushrooms, focusing on the model's ability to distinguish between poisonous and edible species. The analysis aims to explore and understand decision-making mechanisms through the implementation of a decision tree, evaluating the impact of different subdivision criteria, such as entropy and Gini index, on the model's accuracy and generalization. Finally, it is possible to affirm that the goal is to create a robust and interpretable classifier that balances accuracy with model simplicity.

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.

Contents

1 Introduction

In the following chapter, a descriptive analysis of the available dataset will be carried out, with a focus on the target variable and descriptive features. In addition, the treatment adopted for handling null values in the dataset will be illustrated. Finally, an Exploratory Data Analysis (EDA) will be conducted to better understand the structure of the dataset and the distribution of the variables.

1.1 Dataset Description

The dataset consists of 61.069 hypothetical mushrooms, each modelled on one of 173 species, with 353 mushrooms per species. Each mushroom is classified as edible, poisonous or of unknown edibility (the latter category was combined with that of poisonous mushrooms). The dataset includes 20 descriptive variables, of which 17 are nominal variables and 3 are quantitative variables. The categorical variables include characteristics such as cap shape, colour and surface, while the metric variables include measurements such as cap diameter, height and stem thickness.

1.2 Handling missing values

class	0
cap-diameter	0
cap-shape	0
cap-surface	14120
cap-color	0
does-bruise-or-bleed	0
gill-attachment	9884
gill-spacing	25063
gill-color	0
stem-height	0
stem-width	0
stem-root	51538
stem-surface	38124
stem-color	0
veil-type	57892
veil-color	53656
has-ring	0
ring-type	2471
spore-print-color	54715
habitat	0
season	0

Figure 1: Sum of null values

This paragraph is directed to identify if any features contain missing data for considering a proper handling. Several null values are present in the dataset, in particular, six columns had a high number of missing values, more than 40% of the total number of observations. For this reason, as highlighted in Figure 1, the following columns were removed from the final dataset: 'gill-spacing', 'stem-root', 'stem-surface', 'veil-type', 'veil-colour' and 'spore-print-colour'. This decision was made in order to improve the reliability of the predictive analyses, reducing the negative impact of incomplete information on the quality of the model. Subsequently, the rows of the dataset with null values (called NaN) were removed. At the end of this cleaning process, the final dataset, used for the construction of the tree classifier predictive model, consisted of 37.065 observations and 15 total columns. Below, the final dataset can be observed in the Figure 2.

	class	cap-diameter	cap-shape	cap-surface	cap-color	does-bruise-or-bleed	gill-attachment	gill-color	stem-height	stem-width	stem-color	has-ring	ring-type	habitat	season
0	p	15.26	convex	grooves	orange	f	free	white	16.95	17.09	white	ring	grooved	woods	winter
1	p	16.60	convex	grooves	orange	f	free	white	17.99	18.19	white	ring	grooved	woods	summer
2	p	14.07	convex	grooves	orange	f	free	white	17.80	17.74	white	ring	grooved	woods	winter
3	p	14.17	flat	shiny	red	f	free	white	15.77	15.98	white	ring	pendant	woods	winter
4	p	14.64	convex	shiny	orange	f	free	white	16.53	17.20	white	ring	pendant	woods	winter
5	p	15.34	convex	grooves	orange	f	free	white	17.84	18.79	white	ring	pendant	woods	summer
6	p	14.85	flat	shiny	orange	f	free	white	17.71	16.89	white	ring	grooved	woods	winter

Figure 2: Cleaned dataset

1.3 EDA

In this paragraph, an in-depth descriptive analysis of the balance of the different variables under study will be conducted. Initially, the target variable ('class') will be examined, followed by the analysis of the distribution of the explanatory variables to better understand their repartition within the dataset. The goal is to understand the dataset's structure and detect any irregularities.

1.3.1 Study of the binary variable

The target variable in this study is called 'class' and comprises two categories: 'e', indicating edible mushrooms, and 'p', representing poisoned and inedible mushrooms. As shown in Figure 3, the distribution of the class variable is evenly balanced, with a proportion between edible and poisonous mushrooms.

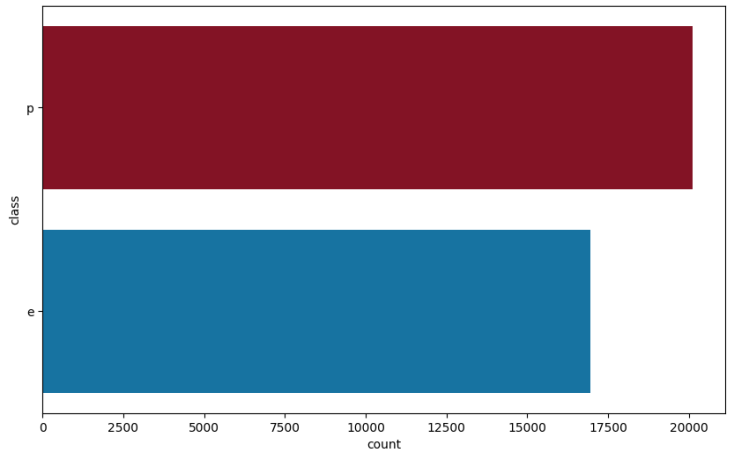


Figure 3: Distribution of the variable 'class'

In the cleaned dataset, there are 16,944 mushrooms labelled 'e', representing 45.71% of the total observations. On the other hand, poisonous mushrooms, labelled 'p', number 20,121, corresponding to 54.29% of the total. This relatively balanced distribution between the two classes ensures that the evaluation of the model is balanced, avoiding potential bias in the predictions. Thanks to this proportion, the model is not biased by a predominance of one of the two classes, which allows a more accurate evaluation of its performance on both categories.

1.3.2 Analysis of the features

There are 14 total explanatory variables in the dataset. As mentioned earlier, three of these are continuous variables, while the remaining eleven are categorical variables. To facilitate a deeper understanding of the content of each column, a description of the meaning of each variable is given below:

1. cap-diameter (m): Diameter of the cap, expressed in centimetres.
2. cap-shape (n): Hat shape with possible values: bell=b, conical=c, convex=x, flat=f, sunken=s, spherical=p, others=o.
3. cap-surface (n): Hat surface with possible values: fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t, wrinkled=w, fleshy=e.
4. cap-colour (n): Hat colour, with variables such as brown=n, buff=b, grey=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k.
5. does-bruise-bleed (n): Indicates whether the fungus is bleeding or showing signs of bruising, with values: bruises-or-bleeding=t, no=f.
6. gill-attachment (n): Mode of gill-attachment, with values such as adnate=a, adnexed=x, decurrent=d, free=e, sinuate=s, pores=p, none=f, unknown=?
7. stem-height (m): Height of the stem, in centimetres.
8. stem-width (m): Width of the stem, in millimetres.
9. stem-color (n): Colour of the stem, with the same values as the variable cap-color.
10. has-ring (n): Indicates the presence of a ring on the stem, with values: ring=t, none=f.
11. ring-type (n): Ring type, with values such as cobwebby=c, evanescent=e, flaring=r, grooved=g, large=l, pendant=p, sheathing=s, zone=z, scaly=y, movable=m, none=f, unknown=?
12. habitat (n): Habitat in which the fungus grows, with values such as grasses=g, leaves=l, meadows=m, paths=p, heaths=h, urban=u, waste=w, woods=d.
13. season (n): Season in which the mushroom is found, with values: spring=s, summer=u, autumn=a, winter=w.

The three numerical variables are distributed as shown in the following figures.

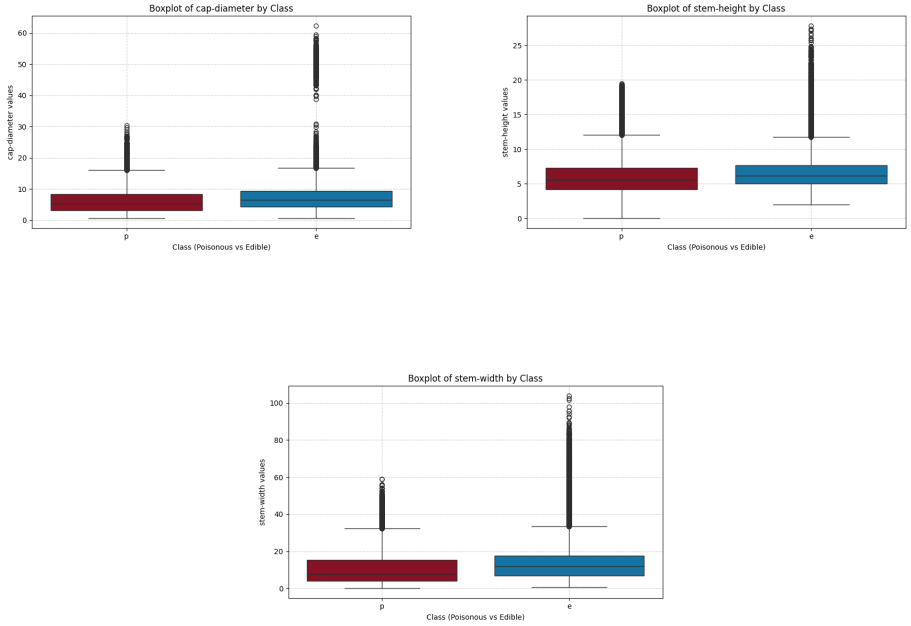
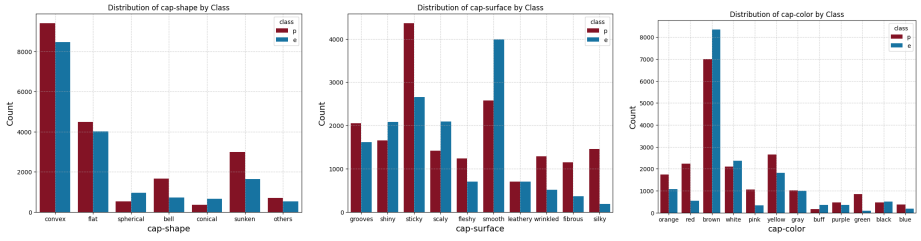
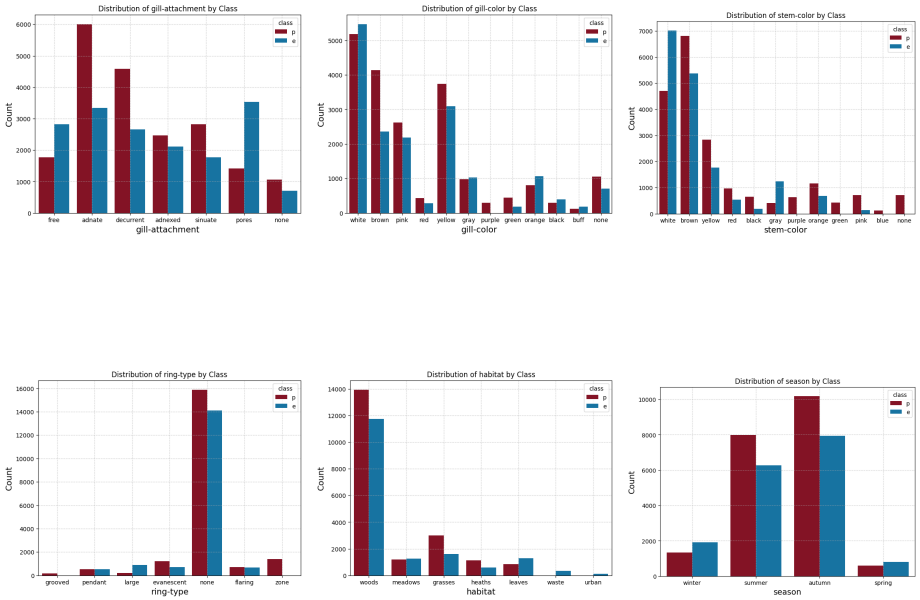


Figure 5: Distribution of numerical variables through boxplots

The graph shows no significant difference between the classes, as the medians and the distribution itself are very similar. However, one can see the presence of a considerable number of outliers for both classes. In many cases, outliers should be removed to prevent them from negatively influencing the prediction and evaluation of the model. However, models based on decision trees, such as the tree predictor, are considered robust to outliers. This is due to a main factor: during training, the optimal split point is calculated on a data sample that, if sufficiently large, tends to mitigate the effect of outliers. Therefore, in the case of decision trees, it is not strictly necessary to eliminate outliers.

Next, the distributions of the categorical variables are evaluated using barplots, always observing the division between the 'poisonous' and 'edible' classes.





The distribution of the categorical variables confirms what was observed previously, which is a homogeneity indicating the absence of bias in the dataset.

2 Implementation of the Tree Classifier

After completing the analysis of the data structure, the decision tree for binary classification is constructed. The objective of classification is to analyse the training set, identifying specific patterns to determine the class to which a particular observation belongs. Decision trees are constructed by recursively dividing observations into increasingly homogeneous subsets, where combinations of attributes lead to the same class. This process creates a hierarchical tree structure, where the subsets are called *nodes*, and the terminal nodes are the *leaves*. An object is classified by following a path through the nodes, starting with the *root*, which represents the feature with the best subdivision criterion. The choice of how to subdivide the nodes is based on a specific splitting criterion. In this project, three were used (entropy, Gini index and Mean Square Error for regression) to evaluate the efficacy of splitting and which criterion offered the most accurate predictions.

Considering the large size of the dataset, the tree could grow too large, becoming complex and susceptible to overfitting. To avoid this, it is necessary to impose restrictions in the form of rules. In this implementation, the maximum depth of the tree was limited and a minimum number of observations in a node was defined for splitting to take place. These parameters help to control the

complexity of the tree and improve the generalisation of the model. Finally, through model training and prediction on test data, the accuracy and training error will be evaluated to measure the overall performance of the three models.

2.1 Train-Test Splitting

The first crucial step in the construction of the model is the division of the dataset into two parts: training set and test set. These two parts play distinct roles in the training and evaluation of the classification tree algorithm. The training set is used to train the model, allowing the algorithm to learn the relationships between the input variable, X, and the target variable, y. The test set, on the other hand, is used to evaluate the model's performance on unseen data, thus simulating how the model will perform on new data.

The split was performed with an 80:20 proportion, meaning that 80% of the observations were allocated to the training set, while 20% were allocated to the test set. As a result, the training set comprises 29.652 observations, while the test set contains 7.413. This division ensures that the model has sufficient data to learn, without sacrificing the ability to test its accuracy on an independent validation set.

2.2 Creation of the model

2.2.1 Splitting criteria

One of the fundamental aspects in the construction of a classification tree is the selection of the best criterion to perform the splitting of nodes, which is the division of observations into two distinct groups. The objective of this process is to ensure that, as the tree splits, the observations in each node are as homogeneous as possible, so that they belong to the same class. In other words, with each division, an attempt is made to minimise the mixing of different classes within the nodes. For this project, three splitting criteria were adopted to assess the accuracy of the tree predictions. These criteria are selectable using the criterion `parameter` when implementing the `TreePredictor` class. Here is a description of the three criteria used:

Entropy: Measures the level of impurity or uncertainty in the data. The objective during the splitting process is to reduce entropy by creating subgroups of data that are as homogeneous as possible. Entropy reaches its maximum value when the classes are uniformly distributed, meaning no class predominates, and uncertainty is at its highest. Conversely, when entropy is zero, all samples belong to the same class, indicating maximum homogeneity. In the context of binary classification (edible vs. poisonous), entropy specifically measures the impurity between the two classes. It is calculated as:

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

where p represents the probability of an observation belonging to one of the two classes. This formulation ensures that the entropy calculation is adapted

to the binary nature of the problem. The goal during the splitting process is to find splits that minimize entropy, thereby increasing the purity of the resulting nodes. This reduction in entropy is called *information gain*, and it is used to select the optimal feature and threshold for each split.

Gini Index: It is used to measure impurity in the dataset and aims to reduce the mixing of classes in the resulting nodes after splitting. The Gini index represents the probability that two randomly selected samples from a dataset belong to different classes. The value of the Gini index ranges between 0 and 1, where 0 indicates perfect purity (all observations belong to the same class) and 1 indicates maximum impurity (even distribution between classes). Specifically designed for binary classification, the Gini index measures impurity by calculating the probability that two randomly chosen samples belong to different classes.It is computed as:

$$\text{Gini}(S) = 2p(1-p)$$

where p is the probability that a sample belongs to a certain class. This approach ensures that the impurity measure is directly aligned with the binary nature of the dataset. The objective during the splitting process is to choose splits that minimize the Gini index, thereby maximizing the purity of the resulting nodes.

Mean Squared Error (MSE) Reduction: This criterion is based on reducing the mean squared error when splitting. It is most commonly used in regression contexts, but can also be applied in this project. The idea is to minimise the MSE in order to obtain nodes that have predicted values closer to the true values. The formula for the MSE is defined as:

$$\text{MSE}(S) = \sqrt{p(1-p)}$$

where p is the probability that a sample belongs to a certain class. At each split, the algorithm selects the split point that minimises the weighted sum of the MSE of the child nodes.

2.2.2 Handling of Categorical and Continuous Variables

After having explained the criteria used for splitting, it is necessary to elaborate on how these are applied according to the type of variable, whether categorical or continuous. For nominative variables, the division is made by comparing discrete values. When a categorical variable is chosen for splitting within a node, the function checks whether the value of a sample is equal to the specific value of the category or not. In contrast, for continuous numeric variables, there is a numeric threshold for distinction. This threshold separates the observations into two distinct groups: those that have a value less than or equal to the threshold and those that have a value greater than the threshold.

2.2.3 Stopping criteria

The decision tree is built recursively by dividing the data into increasingly homogeneous subsets until certain stopping conditions are met. These stopping

criteria are crucial to prevent the tree from growing too deep, leading to a complex model prone to overfitting. In the project, the following three stopping criteria were implemented:

Maximum tree depth: One of the main stopping criteria is the maximum depth the tree can reach. If the current depth of a node exceeds the value set in `max_depth` (set to 5), the subdivision stops and the node becomes a leaf. This prevents the tree from becoming too deep and complex, reducing the risk of overfitting. Depth controls the number of consecutive splits, and too much depth can lead to the model 'memorising' training data instead of generalising well to new data.

Minimum number of samples per splitting: Another stopping criterion concerns the size of subsets in the nodes. If a node contains less than a certain number of samples (defined by the `min_samples_split` parameter, which is set to 500), no splitting is performed. This ensures that each node contains enough data to produce a meaningful decision, avoiding splits based on a few samples, which could be noise rather than a useful signal.

Node purity: An additional automatic stop criterion is activated when all samples in a node belong to the same class (i.e. when there is no variation in the target variable, y). In this case, there are no more informational advantages in continuing to subdivide the node, and recursiveness stops. The node then becomes a leaf, and the class associated with the leaf is the most common among the samples at that node.

These stopping criteria not only ensure that the decision tree does not become overly complex, but also improve the generalisation of the model. In particular, the combination of maximum depth and minimum number of samples per split ensures that the tree does not grow too large or become too specific to the training data, avoiding the problem of overfitting.

2.3 Evaluation of accuracy and zero-one loss

After having implemented the class for the predictor tree and applied the three splitting criteria, the next step is to evaluate the actual performance of the model. To do this, it is crucial to use appropriate metrics that provide a clear measure of the accuracy and quality of the predictions. The main indicators to consider include *accuracy*, which indicates the percentage of correct classifications, and *zero-one loss*, which measures the forecast error rate. These two tools help to quantify the effectiveness of the model in correctly distinguishing between edible and poisonous mushrooms.

Accuracy represents the percentage of observations correctly classified by the model. It is calculated by dividing the number of correct predictions by the total number of predictions made. The formula is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- TP (True Positives): Number of correctly classified positive examples.

- TN (True Negatives): Number of correctly classified negative examples.
- FP (False Positives): Number of negative examples incorrectly classified as positive.
- FN (False Negatives): Number of positive examples wrongly classified as negative.

In our case, accuracy was assessed using the three splitting criteria implemented, and the results obtained are as follows:

- Model accuracy with 'Entropy': 0.7638
- Model accuracy with 'Gini Index': 0.7811
- Model accuracy with 'MSE distance': 0.6185

The results show that the model based on the Gini index achieved the best performance, with an accuracy of 78.11%, followed by the model based on entropy, which achieved an accuracy of 76.38%. The criterion based on the reduction of the MSE, on the other hand, produced inferior results, with an accuracy of 61.85%, indicating a lower effectiveness in correctly separating the classes in the mushroom dataset.

Besides accuracy, another key metric for measuring model quality is the zero-one loss indicator. This measure represents the percentage of incorrect predictions out of the total number of predictions. In other words, zero-one loss counts the number of times the model misclassified a sample, regardless of the severity or type of error. A zero-one loss value of 0 indicates that the model has correctly classified all observations, while a value of 1 indicates that all classifications are incorrect. The formula is:

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

where $l(y, \hat{y})$ is the loss function, y is the true label and $\hat{y}$ is the predicted label. In this case, the test error results for the three splitting criteria are as follows:

- Test error with 'Entropy': 0.2362
- Test error with 'Gini Index': 0.2189
- Test error with 'MSE distance': 0.3815

Again, the results confirm that the model based on the Gini index is the most effective, with the lowest test error (21.89%), followed by the model based on entropy (23.62%). The MSE criterion also shows worse results in this case. In addition, the training error results are shown:

- Training error with 'Entropy': 0.2298
- Training error with 'Gini Index': 0.2098
- Training error with 'MSE distance': 0.3848

The training error does not guarantee that the model will perform equally well on new data. Nevertheless, it is very useful because if the training error were to be very low and the test error very high, it could indicate that the model is overfitting, meaning that that it fails to generalise. Fortunately, in this case, the Gini index and entropy are both low and therefore the model can make accurate predictions on unseen data. The MSE reduction criterion, on the other hand, has a much higher error in both the training (38.48%) and the test (38.15%), suggesting a poor capacity for generalisation.

Metric	Entropy	Gini Index	MSE Distance
Accuracy	0.7638	0.7811	0.6185
Training Error	0.2298	0.2098	0.3848
Test Error	0.2362	0.2189	0.3815

For better understanding and visualisation, a final table with the results obtained is shown.

In conclusion, as we have seen, it can be said that the Gini index gave better results and, therefore, this criterion will be used from now on.

2.4 Insight into the output of trees

To better understand how the prediction tree works, this section presents the output (with the tree structure) obtained using the best performing criterion, the tree with the Gini index.

```
Node: stem-width <= 8.94
---> Left:
|   Node: stem-color == g
|   ---> Left:
|   |   Node: gill-attachment == x
|   |   ---> Left:
|   |   |   Leaf: Predicted class = p
|   |   ---> Right:
|   |   |   Leaf: Predicted class = e
|   ---> Right:
|   |   Node: stem-height <= 3.28
|   |   ---> Left:
|   |   |   Node: stem-color == w
|   |   |   ---> Left:
|   |   |   |   Node: habitat == g
|   |   |   |   ---> Left:
|   |   |   |   |   Leaf: Predicted class = p
|   |   |   |   ---> Right:
|   |   |   |   |   Leaf: Predicted class = e
|   |   |   ---> Right:
|   |   |   |   Node: stem-width <= 7.06
|   |   |   |   ---> Left:
|   |   |   |   |   Leaf: Predicted class = p
|   |   |   |   ---> Right:
|   |   |   |   |   Leaf: Predicted class = e
...
|   |   |   |   ---> Left:
|   |   |   |   |   Leaf: Predicted class = p
|   |   |   |   ---> Right:
|   |   |   |   |   Leaf: Predicted class = e
```

Figure 9: Gini output

In this output the starting point, or root node, evaluates the stem-width of the mushroom. If the `stem-width` is less than or equal to 8.94 mm, the tree branches to the left, otherwise to the right. On the left side, the model assesses the `stem-colour`. If the stem is grey (g), the model then examines the `gill-attachment`. If this attachment is adnexed (x), the tree is further divided: mushrooms with these characteristics are classified as poisonous on the left, and as edible on the right. If, on the other hand, the stem colour is not grey, the tree considers the `stem-height`. If this is less than or equal to 3.28 cm, the model checks whether the stem is white (`stem-colour` == w). Then you can evaluate the whole tree in this way (also for the other two splitting criteria).

2.5 Confusion matrix

The confusion matrix below presents the results of the predictions based on the Gini index as splitting criteria. The model correctly classified 2.799 edible mushrooms and 2.991 poisonous mushrooms, i.e. 37.76% and 40.35% of the total observations in the test set, respectively. In contrast, the model made 1.042 (14.06%) errors classifying

poisonous mushrooms as edible (potentially dangerous), and classified 581 (7.84%) edible mushrooms as poisonous (a less serious but undesirable error). Thus, it can be said that there is a significant number of false positives.

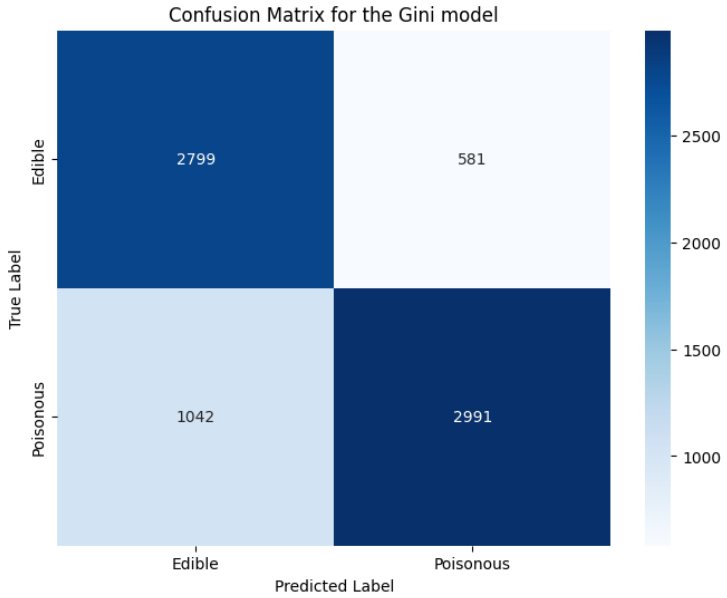


Figure 10: Confusion matrix

3 Hyper-parameter Tuning

Hyperparameter tuning optimises the parameters of a model to improve its performance. In decision trees, the main hyperparameters are `max_depth`, which adjusts the maximum depth of the tree and balances complexity with generalisation, and `min_samples_split`, which defines the minimum number of samples required to split a node. In this project, the objective was to test different thresholds for `min_samples_split`, from 500 to 3000 in increments of 500, to find the combination that offered the best balance between accuracy and model complexity.

	max_depth	min_samples	accuracy_train	accuracy_test
15	6	500	0.820754	0.819102
16	6	1000	0.811817	0.809659
17	6	1500	0.796574	0.795629
10	5	500	0.790166	0.781060
18	6	2000	0.780925	0.779981
19	6	2500	0.780925	0.779981
11	5	1000	0.783556	0.773911
5	4	500	0.768616	0.762714
12	5	1500	0.768312	0.759881
13	5	2000	0.767908	0.759611
14	5	2500	0.767908	0.759611
6	4	1000	0.764299	0.757723
7	4	1500	0.749056	0.743694
8	4	2000	0.748820	0.743424
9	4	2500	0.748820	0.743424
0	3	500	0.728787	0.726831
1	3	1000	0.724471	0.721840
4	3	2500	0.724471	0.721840
3	3	2000	0.724471	0.721840
2	3	1500	0.724471	0.721840

As can be seen from the figure, the hyperparameter tuning established that the best hyperparameters are a maximum depth level of 6 and a minimum number of samples to split at a node of 500 with an accuracy of 81.91%. Thus, this procedure improved the overall performance compared to the default parameters. Furthermore, it can be seen that increasing the number of samples per split too much associated with a lower depth number reduces the model's ability to split the data, decreasing accuracy.

Finally, a graph is displayed showing the relationship between accuracy and maximum depth while keeping the minimum number of samples parameter equal to 500. Train accuracy and test accuracy go hand in hand, which means that there are absolutely no overfitting problems (because the test error as the depth increases should have increased).

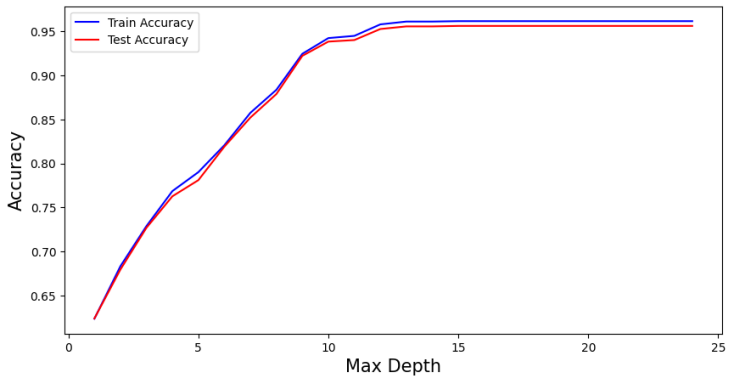


Figure 11: Train and test accuracy visualization

4 K-fold cross-validation

K-fold cross-validation is a validation technique in which the randomised dataset is divided into K subgroups (or 'folds') of equal size and with the same distribution. In each of the K iterations, K-1 folds are used as a training set to train the model, while the remaining fold is used as a test set to evaluate its performance. This process is repeated K times, ensuring that each fold is used exactly once as a test set. In the end, the accuracy of the model is calculated as the average of the accuracies obtained in each iteration, providing a more reliable estimate of its performance than simply splitting it into training and test sets once. In this case, K equal to 5 was chosen, meaning that the dataset was divided into 5 equal parts. In each iteration, 4 parts were used for training and 1 for validation, repeating the process for a total of 5 iterations. Each fold contained 29,652 observations in the training set and 7,413 observations in the test set. Cross-validation is a commonly used technique to assess the performance of a machine learning model, as it provides an estimate of its ability to generalise to unseen data. By using this method, the risk of depending too much on a specific subdivision of the dataset is minimised. Proceeding with the 5 iterations, the following accuracies were obtained on each fold:

	Accuracy
Fold 1	0.7875
Fold 2	0.7726
Fold 3	0.7789
Fold 4	0.7904
Fold 5	0.7883
Mean Accuracy	0.7835

To conclude, it can be observed that the mean accuracy obtained with the K-fold cross-validation is 0.7835, very similar to the accuracy of the model with the Gini index (0.7811). This confirms the consistency of the model's performance across different subdivisions of the dataset, indicating a good ability to generalise.

5 Conclusion

In this analysis, a decision tree was built from scratch to classify mushrooms as poisonous or edible. Three splitting criteria—entropy, Gini index, and Mean Squared Error—were implemented, with the Gini index achieving the best accuracy (78.11%) and zero-one loss. K-fold cross-validation confirmed the model's robustness with a consistent accuracy of 78.35%. Hyperparameter tuning further improved performance, raising accuracy to 81.91%. Overall, the model demonstrated strong predictive ability and reliability, making it suitable for future classification tasks.