UNIVERSITÀ DEGLI STUDI DI MILANO

MASTER IN DATA SCIENCE FOR ECONOMICS

MACHINE LEARNING AND STATISTICAL LEARNING
TREE PREDICTORS FOR BINARY CLASSIFICATION

Academic year: 2023 – 2024
Student: Inas El kouch
Professor: Nicolò Cesa Bianchi

1. Introduction

The objective of this project is to design and implement a binary decision tree from scratch and evaluate its performance on a mushroom dataset.
The primary goal is to classify mushrooms as either poisonous or edible based on a range of distinguishing attributes.
For operational simplicity the project could be divided into different tasks.
The initial task involves a thorough analysis of the dataset, followed by preprocessing steps such as splitting the data into training and test sets and finally cleaning.
The core task revolves around constructing a decision tree predictor, where single-feature binary tests are applied at each internal node. As part of this process, a node structure will be designed, and a tree predictor class will be implemented to train on the dataset and generate predictions for unseen data through the "fit" and "predict" functions.
The key methods involve three splitting criteria:

- Gini impurity
- Scaled entropy
- Squared impurity

and three stopping criteria:

- Maximum depth
- Minimum sample split
- Minimum impurity decrease

The fundamental idea is that these are essential to avoid overfitting and underfitting.
Another task is the hyperparameter tuning that is able to optimize the parameters that control the learning process consequently improving the model's performance, complexity, and generalization ability.
The metrics implemented to evaluate the model include 0 – 1 loss and confusion matrix.

2. Dataset analysis

The dataset used in this study is the "Secondary Mushroom Dataset", comprising 61069 mushrooms with caps derived from 173 species. Each mushroom is labeled as either "e" (edible) or "p" (poisonous), which serves as the target class for classification.
Additionally, the dataset includes 21 features that describe various physical attributes of the mushrooms. Here is a thorough description:

- classification: edible or poisonous

- cap diameter: Numerical

- cap shape: conical, convex, bell, flat, sunken, spherical, other.
  Categorical.

- cap surface: fibrous, grooves, scaly, smooth, shiny, leathery, silky, sticky, wrinkled, fleshy
  Categorical

- cap color: brown, buff, gray, green, pink, purple, red, white, yellow, blue, orange, black
  Categorical

- bruise or bleed: true, false
  Categorical

- gill attachment: adnate, adnexed, decurrent, free, sinuate, pores, none
  Categorical

- gill spacing: close, distant, none
  Categorical

- gill color: brown, buff, gray, green, pink, purple, red, white, yellow, blue, orange, black, none
  Categorical

- stem height: Numerical

- stem width: Numerical

- stem root: bulbous, swollen, club, cup, equal, rhizomorphs, rooted
  Categorical

- stem surface: fibrous, grooves, scaly, smooth, shiny, leathery, silky, sticky, wrinkled, fleshy, none
  Categorical

- stem color: brown, buff, gray, green, pink, purple, red, white, yellow, blue, orange, black, none
  Categorical

- veil type: partial, universal
  Categorical

- veil color: brown, buff, gray, green, pink, purple, red, white, yellow, blue, orange, black, none
  Categorical

- has ring: true, false
  Categorical

- ring type: cobwebby, evanescent, flaring, grooved, large, pendant, sheathing, zone, scaly, movable, none
  Categorical

- spore print color: brown, buff, gray, green, pink, purple, red, white, yellow, blue, orange, black
  Categorical

- habitat: grasses, leaves, meadows, paths, heaths, urban, waste, woods
  Categorical

- season: spring, summer, autumn, winter
  Categorical

First, the features and target variable were separated, followed by an assessment to determine whether the dataset was balanced. A balanced dataset ensures that all classes are represented proportionally, preventing bias in the model. If a dataset is highly imbalanced, the model may become biased toward the majority class, leading to poor generalization and inaccurate predictions for the minority class.
In this case it was concluded that the dataset was properly balanced.
Then, the data was split into a train-to-test ratio 80:20.

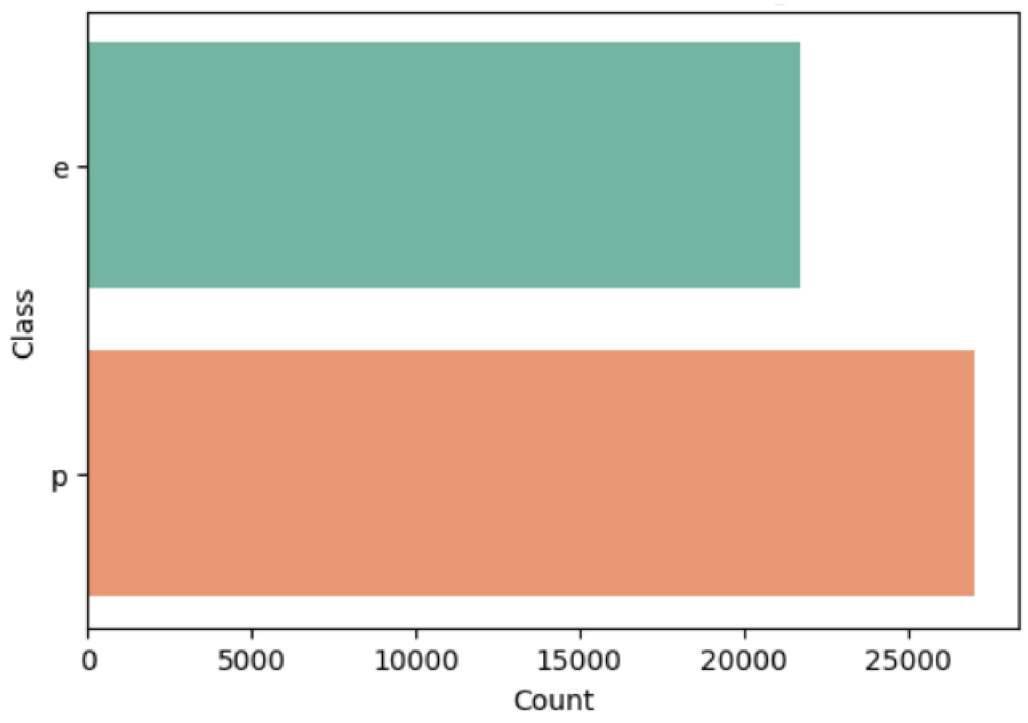
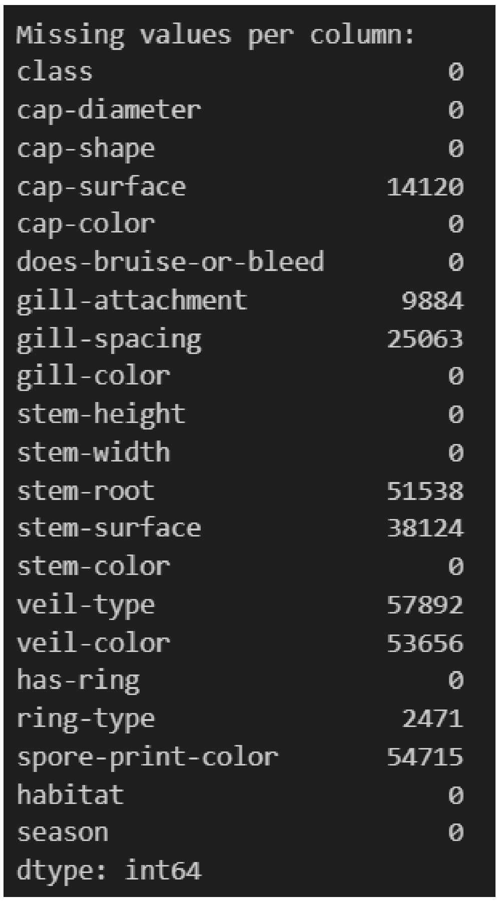


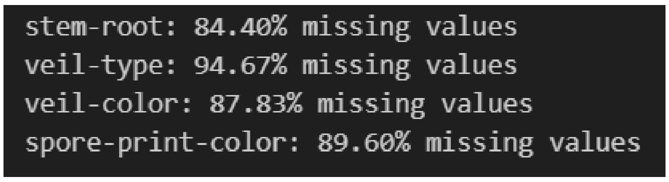
Fig. 1: Distribution of classes E and P

Finally, the dataset had a significant number of missing values, which were managed by setting an elimination threshold. Variables with more than 80% missing values were removed to maintain data quality and reliability. It was argued whether to choose 70% as an elimination threshold but the highest percentage ensured that only the most incomplete and unreliable variables were removed while maintaining a robust and informative dataset.



```
Missing values per column:
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
dtype: int64
```

Fig. 2: Total missing values



```
stem-root: 84.40% missing values
veil-type: 94.67% missing values
veil-color: 87.83% missing values
spore-print-color: 89.60% missing values
```

Fig. 3: Variables with more than 80% missing values

Ultimately, to avoid the risk of overfitting duplicate columns were dropped.
The preprocessing procedure terminates at this step as for the implementation of a decision tree there is no need to assess and remove outliers.

3. Decision tree

A decision tree is a supervised learning model that recursively partitions the input space into distinct, non-overlapping regions based on feature values, forming a hierarchical structure of decision rules. Nodes are the elementary units of a decision tree, from root node to leaves. Each node has the following attributes:

- Feature: the index of the feature used to split the data.
- Threshold: threshold value for the feature to decide the split.
- Left: reference to the left child node (where data values are less than or equal to the threshold).
- Right: reference to the right child node (where data values are greater than the threshold).
- Value: the predicted class or target value if it's a leaf node (i.e., no further splits).

The process begins with a root node that considers all data points and selects the optimal feature and threshold for splitting, typically based on a criterion such as, Gini impurity for classification, scaled entropy or squared impurity.

- Scaled entropy
  It is used to measure how “mixed” or “impure” the class labels are in a given node. A node with many different labels in roughly equal proportions has *high entropy*, whereas a node with mostly one label has *low entropy*.

  $$\psi(p) = -\frac{p}{2}\log_2(p) - \frac{1-p}{2}\log_2(1-p)$$

- Gini impurity
  It assesses the impurity of a node by estimating the probability of an incorrect classification based on the distribution of classes within the node. It indicates the chance that a randomly selected element would be misclassified if assigned a class label according to the observed class proportions in the node.

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

- Squared impurity
  It’s a specialized adaptation of Gini impurity for binary classification. It is computed by taking the square root of the product between each class probability and its complement, effectively capturing how “mixed” the node is for that class label.

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

Once the best split is determined, two child nodes are created: the left node contains data points that satisfy the split condition, while the right node contains the remaining points. This process continues recursively until a stopping condition is met. The stopping criteria implemented in the model are:

- Maximum depth the decision tree can grow.

- Maximum number of leaf nodes allowed.

- Minimum number of samples required to allow a node to be split.

- Entropy, as when a split does not reduce it significantly (i.e., information gain is low), the tree may stop splitting to avoid unnecessary complexity.

Last, if a node cannot be further split, it becomes a leaf node, storing a final predicted value or class label. The last procedure to better the model is the hyperparameter tuning. Hyperparameters are predefined parameters that must be set before training a model, as they cannot be directly learned from the data during the training process. Optimizing these hyperparameters is essential for enhancing the performance of the decision tree. This optimization process involves testing various hyperparameter configurations and selecting the one that produces the most effective results.
An additional operation was introduced in order to speed up the whole process: the parallelization provided by “Joblib” that enables efficient execution of independent tasks across multiple CPU cores, significantly reducing computational time. Instead of processing tasks sequentially, Joblib's Parallel function distributes them across the available processors, allowing simultaneous execution. By using n_jobs=-1, all CPU cores are utilized, maximizing resource efficiency. The delayed function wraps the target function, deferring its execution until it is called within Parallel, which then schedules multiple instances of the function to run concurrently.

4. Evaluation

In order to evaluate the model, these are the metrics that were utilized:

- 0 – 1 Loss where the predicted label matches the true label, the loss for that sample is 0; otherwise, it is 1 then these values are averaged over all samples to get an overall measure of the model's error rate.

$$L_{0-1} = \frac{1}{n}\sum_{i=1}^{n} \ell_i$$

- Accuracy is the proportion of correct predictions among all predictions made. Formally, it is the number of correctly classified instances divided by the total number of instances.

$$= \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(y_i = \hat{y}_i)$$

- Confusion matrix is a tabular layout that shows how predictions from a classification model compare to the true labels. It has rows representing the actual classes and columns representing the predicted classes or vice versa. Each cell in the matrix indicates how many samples belonging to a particular actual class were predicted as a particular predicted class.

/	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

- Precision is the ratio of correctly predicted positives over the total predicted positives.

$$= TP / (TP + FP)$$

- Recall is the ratio of correctly predicted positives over the total actual positives

$$TP / (TP + FN)$$

- F1 score is the harmonic mean of precision and recall.

$$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The results obtained by the initial model were unsatisfactory although it achieves a 73.5% accuracy and a high recall of 90.6%, indicating it successfully identifies most poisonous mushrooms where only 512 were missed. However, its precision of 64.5% reflects a tendency to label edible mushrooms as poisonous, as shown by the 2724 false positives. While the F1 score of 0.75 suggests a reasonable balance between precision and recall, the high risk posed by false negatives in the scenario of poisonous mushrooms remains a critical factor that could be improved with hyperparameter tuning, hopefully obtaining a lower 0 – 1 loss than the shown 26.5%.

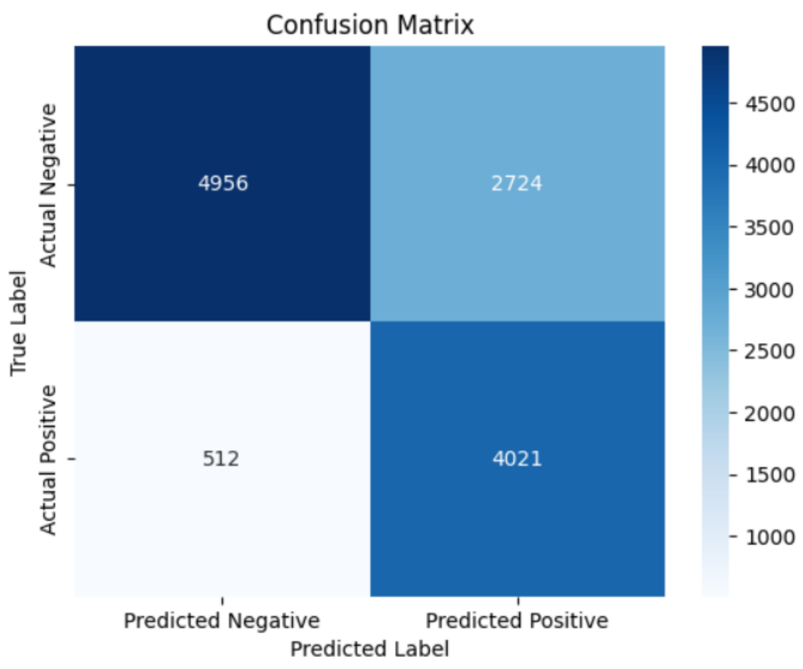
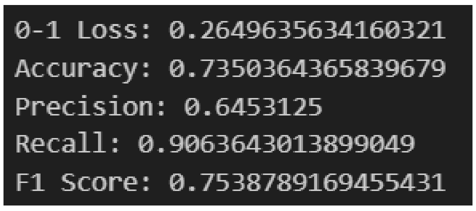


Fig. 4: Confusion matrix model with no hyperparameter tuning



```
0-1 Loss: 0.2649635634160321
Accuracy: 0.7350364365839679
Precision: 0.6453125
Recall: 0.9063643013899049
F1 Score: 0.7538789169455431
```

Fig. 5: Evaluation 1

Finally, the model improved consistently but it was still not able to reach a perfect prediction of a stake of poisonous mushrooms. This model shows a very high level of accuracy of 99.7% and very low 0–1 loss of just 0.0023. With a precision of 99.87% and a recall of 99.62%, it demonstrates a strong ability to identify both poisonous and edible mushrooms correctly. The confusion matrix confirms this performance: only 7 edible mushrooms are mistakenly classified as poisonous (false positives), while 21 poisonous mushrooms are misclassified as edible (false negatives).

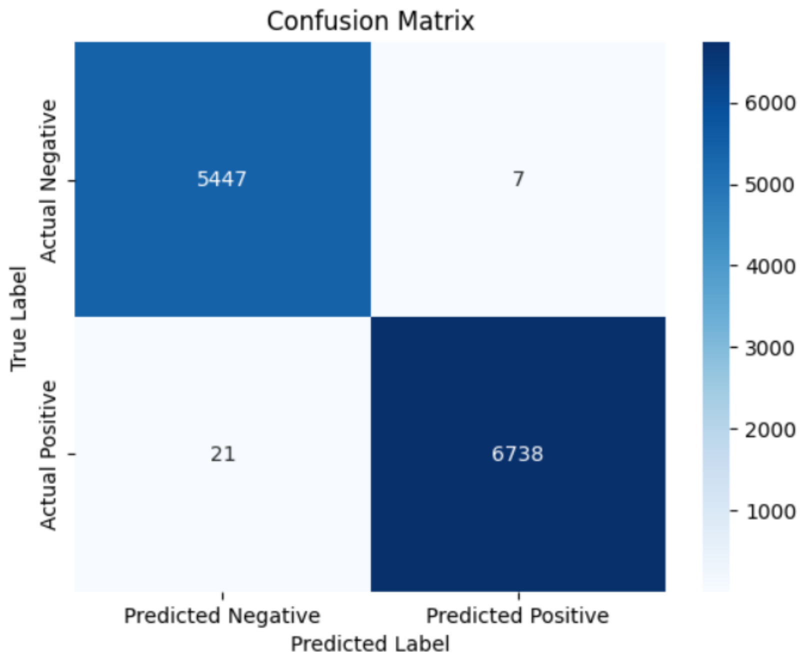
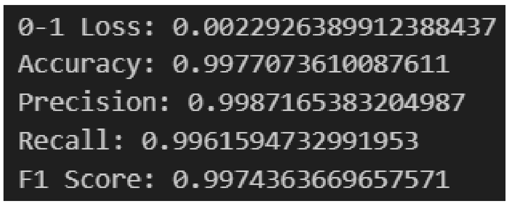
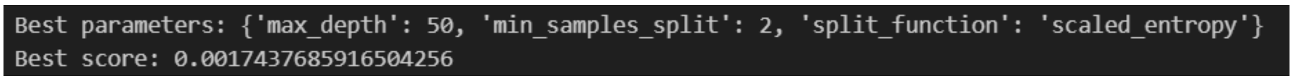


Fig. 6: Confusion matrix model after hyperparameter tuning



```
0-1 Loss: 0.0022926389912388437
Accuracy: 0.9977073610087611
Precision: 0.9987165383204987
Recall: 0.9961594732991953
F1 Score: 0.9974363669657571
```

Fig. 7: Evaluation 2



```
Best parameters: {'max_depth': 50, 'min_samples_split': 2, 'split_function': 'scaled_entropy'}
Best score: 0.0017437685916504256
```

Fig. 8: Best parameter

5. Conclusions

The model obtained the expected improvement but still, in the field of mushrooms toxicity it is essential to obtain even better results. With regard to the process, the model is not faulty for overfitting and as the tree does not grow excessively, no pruning is necessary. The model is able to generalize unseen data well given a small difference between training and test errors.
Future improvements could require an even larger dataset and exploring additional techniques as K – fold cross validation or random forest that leverages ensemble learning to reduce variance and improve overall predictive stability.

6. List of figures