Tree Predictors for Binary Classification

Project 2: Machine Learning

February 28, 2025

1 Introduction

The goal of this project is to implement tree-based predictors from scratch to classify mushrooms as either poisonous or edible based on their features. The dataset consists of categorical and ordinal attributes, requiring a tree-based method for effective classification.

2 Data Source and Description

The dataset is derived from the following sources:

- Patrick Hardin. *Mushrooms; Toadstools.* Zondervan, 1999.
- Jeff Schlimmer. *Mushroom Data Set.* April 1987.
- Mushroom Repository by Dennis Wagner (05 September 2020).

The dataset contains 173 species of mushrooms classified as:

- Edible (e)
- Poisonous (p)
- Of unknown edibility (combined with poisonous class)

It includes 20 variables (17 nominal, 3 metrical).

3 Theoretical Background

Tree predictors are defined as ordered, rooted trees where each internal node represents a decision rule based on a single feature. Each leaf contains a label from the set $Y = \{-1, 1\}$. Given a feature vector $x \in \mathbb{R}^d$, the classifier assigns a label by traversing the tree based on decision criteria at each node.

3.1 Decision Criteria

A decision function at node v is defined as:

$$f_v(x) = \begin{cases} 1, & \text{if } x_j \leq \theta \text{ (for numerical features)} \\ 1, & \text{if } x_j \in S \text{ (for categorical features)} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where θ is a threshold and S is a subset of categorical values.

3.2 Splitting Criteria

The optimal split at node v is determined using impurity measures such as:

- **Gini Index**: $G(p) = 2p(1-p)$
- **Entropy**: $H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$
- **Misclassification Error**: $E(p) = \min(p, 1-p)$

The chosen split minimizes the weighted sum of impurity in child nodes:

$$\Delta I = I(S) - \left(\frac{|S_L|}{|S|}I(S_L) + \frac{|S_R|}{|S|}I(S_R)\right) \tag{2}$$

where S is the set of samples at node v, and S_L, S_R are the left and right subsets after splitting.

3.3 Stopping Criteria

A tree expansion stops when:

- Maximum depth $D_{\max}$ is reached.
- The number of samples in a leaf is below a threshold $N_{\min}$.
- The impurity decrease ΔI is below a threshold ϵ.

4 Implementation Details

The decision tree is implemented as two main classes: `TreeNode` and `DecisionTree`.

4.1 TreeNode Class

Represents an internal node or a leaf in the decision tree.

- **Attributes:**
  - left: Pointer to left child.
  - right: Pointer to right child.
  - is_leaf: Boolean indicating whether node is a leaf.

  - split_feature: Index of feature used for splitting.
  - split_value: Threshold or subset used for splitting.
- **Methods:**
  - `evaluate(x)`: Routes sample x to the appropriate child node.

4.2 DecisionTree Class

Manages the training and inference process of the tree.

- **Attributes:**
  - root: Root node of the tree.
  - criterion: Function for calculating impurity.
- **Methods:**
  - `train(S)`: Builds the tree by recursively splitting data.
  - `predict(X)`: Routes a dataset X through the tree and returns predictions.
  - `prune()` (optional): Post-pruning to reduce overfitting.

5 Class Structure

The implementation consists of the following main classes:

5.1 `TreeNode`

Defines a single node in the decision tree.

- **Attributes:**
  - Left and right child nodes
  - Decision criterion
  - Leaf status
- **Methods:**
  - Constructor to initialize node attributes
  - Method to check if the node is a leaf

5.2 `DecisionTree`

Implements the full binary decision tree.

- **Attributes:**
  - Root node
  - Splitting criterion
  - Stopping conditions
- **Methods:**
  - `train(X, y)`: Builds the tree from training data
  - `predict(x)`: Classifies a new sample
  - `evaluate(X, y)`: Computes prediction accuracy

6 Training Procedure

The tree is trained using recursive splitting:

Algorithm 1 Train Decision Tree

```
1: Initialize tree with root node
2: Select best split based on impurity reduction
3: Recursively split until stopping conditions are met
4: Assign labels to leaves =0
```

7 Hyperparameter Tuning

Optimal hyperparameters are selected by performing grid search over:

- Maximum tree depth $D_{\text{max}} \in \{3, 5, 10, 20\}$.
- Minimum samples per leaf $N_{\text{min}} \in \{1, 5, 10\}$.
- Impurity threshold $\epsilon \in \{0.01, 0.05, 0.1\}$.

Performance is measured using cross-validation with 0-1 loss:

$$L_{01}(h, X) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}(h(x_i) \neq y_i) \tag{3}$$

8 Results and Discussion

After training and evaluation, we compare different trees based on:

- Training error vs. Validation error.
- Overfitting indicators (depth vs. accuracy).
- The effect of pruning on performance.

9 Conclusion

This project implemented a decision tree from scratch, demonstrating the impact of different splitting and stopping criteria. Further improvements include pruning and ensemble methods such as random forests.