Decision Tree Implementation

Ashi Veerman 2021MT10241

October 13, 2024

1 Introduction

This report covers the implementation of a Decision Tree classifier, along with the effects of pruning on the size of the tree and the model's performance. We explore how pruning improves generalization and mitigates overfitting by comparing model performance before and after pruning using various evaluation metrics.

2 Upsampling

When one class is significantly underrepresented compared to others, the model may learn to favor the majority class, resulting in poor performance on the minority class. One effective technique to mitigate this issue is upsampling the minority class (in our case 'yes' class).

Upsampling involves duplicating instances from the minority class until its size matches that of the majority class. This helps the model learn better from the minority class, improving overall classification performance.

By applying upsampling, we effectively address class imbalance in our dataset. The resulting dataset is more balanced, allowing the machine learning model to learn effectively from both classes. This can significantly improve the performance of the model, particularly on the minority class, and lead to more reliable predictions.

3 Decision Tree Implementation

The decision tree was implemented from scratch using numpy and pandas library. The structure of the tree is built recursively, splitting data at each node based on the feature and threshold that provide the highest information gain, using either entropy or the gini index as the splitting criterion.

3.1 Key Components of the Implementation

3.1.1 Node Structure

Each node in the tree contains the following attributes:

- **Feature:** The feature (or column index) used to split the data at that node.
- **Threshold:** The value of the feature used for the split.

- **Left and Right Child Nodes:** The subtrees resulting from the split.
- **Gain:** The information gain from splitting the data at this node.
- **Value:** The predicted class for leaf nodes (used when the node cannot be split further).

3.1.2 Splitting Criterion

The decision tree uses information gain to decide which feature and threshold to split on. Two metrics are used to compute the quality of splits:

- **Entropy:** Measures the uncertainty in a dataset. A split is chosen to reduce the entropy, maximizing information gain.
- **Gini Index:** A measure of how often a randomly chosen element from the dataset would be incorrectly classified.

3.1.3 Recursive Tree Building

The tree is built recursively, starting with the full dataset at the root and splitting it at each node based on the feature and threshold that maximize the information gain or minimize the Gini index. Splitting continues until a stopping criterion is met:

- **Max Depth:** The maximum depth of the tree, beyond which no further splits are performed.
- **Min Samples:** The minimum number of samples required to split a node. Nodes with fewer samples become leaf nodes.
- **Pure Nodes:** Nodes where all data points belong to the same class become leaf nodes.

3.2 Tree Pruning

To avoid overfitting, pruning was implemented post-construction. Once the tree is built, it is pruned using a validation set. Branches that do not contribute to improved performance are removed, resulting in a simpler tree.

4 Hyperparameter Tuning

The table below presents the training and testing accuracies for various combinations of `max_depth` and `min_samples` in the Decision Tree model.

max_depth	min_samples	Training Accuracy	Testing Accuracy
2	25	0.88968	0.88708
	50	0.88968	0.88708
	100	0.88968	0.88708
	125	0.88968	0.88708
5	25	0.90398	0.89482
	50	0.90367	0.89460
	100	0.90365	0.89516
	125	0.90365	0.89516
10	25	0.91587	0.89361
	50	0.91241	0.89449
	100	0.91056	0.89571
	125	0.90976	0.89571
12	25	0.92355	0.89648
	50	0.91777	0.89726
	100	0.91413	0.89781
	125	0.91269	0.89803
15	25	0.93204	0.89593
	50	0.92267	0.89869
	100	0.91719	0.89903
	125	0.91554	0.89936
20	25	0.94050	0.89106
	50	0.92684	0.89482
	100	0.91957	0.89604
	125	0.91736	0.89659

Table 1: Training and Testing Accuracies for Different Hyperparameters

5 Effect of Pruning on Tree Size

Pruning significantly reduces the size of the decision tree by removing branches that do not contribute to improved performance on the validation set. This simplification makes the model more interpretable and reduces the risk of overfitting.

Before pruning, the tree had $N_{\text{pre-prune}}$ nodes, and after pruning, the number of nodes was reduced to $N_{\text{post-prune}}$. The size reduction percentage is given by:

$$\text{Size Reduction (\%)} = \frac{N_{\text{pre-prune}} - N_{\text{post-prune}}}{N_{\text{pre-prune}}} \times 100\% \tag{1}$$

Substituting the values, we have:

$$\text{Size Reduction (\%)} = \frac{705 - 385}{705} \times 100\%$$

Calculating this gives:

$$\text{Size Reduction (\%)} = \frac{320}{705} \times 100\% \approx 45.38\%$$

This indicates that pruning led to a size reduction of approximately 45.38%, highlighting the effectiveness of pruning in simplifying the decision tree while maintaining its performance.

6 Performance Metrics Before Pruning

The following metrics were obtained before pruning:

- **Training Data:**
  - Accuracy: 0.87196
  - Precision: 0.85375
  - Recall: 0.89782
  - F1-Score: 0.87523
- **Testing Data:**
  - Accuracy: 0.86235
  - Precision: 0.84292
  - Recall: 0.89009
  - F1-Score: 0.86586

The total number of nodes in the tree before pruning was 705.

7 Performance Metrics After Pruning

After applying pruning, the following metrics were obtained:

- **Training Data:**
  - Accuracy: 0.86851
  - Precision: 0.84826
  - Recall: 0.89772
  - F1-Score: 0.87229
- **Testing Data:**
  - Accuracy: 0.86354
  - Precision: 0.84254
  - Recall: 0.89360
  - F1-Score: 0.86732

The total number of nodes in the tree after pruning was 385.

8 Analysis of Results

Comparing the metrics before and after pruning:

- **Training Data:**
  - Accuracy decreased slightly from 0.87196 to 0.86851.
  - Precision decreased from 0.85375 to 0.84826.
  - Recall remained nearly constant at 0.89782 before pruning and 0.89772 after pruning.
  - F1-Score decreased from 0.87523 to 0.87229.
- **Testing Data:**
  - Accuracy increased from 0.86235 to 0.86354.
  - Precision decreased from 0.84292 to 0.84254.
  - Recall increased from 0.89009 to 0.89360.
  - F1-Score increased from 0.86586 to 0.86732.

The pruning process resulted in a reduction of the tree's complexity from 705 nodes to 385 nodes, indicating a significant simplification of the model. This reduction can help in improving generalization on unseen data while maintaining similar performance metrics.

9 Summary

Implemented the following functionality

- upsampling for unbalanced data
- standard decision tree structure
- cost complexity pruning to avoid overfitting and generalise the data