PROJECT 1

Decision Tree Algorithm Design and Implementation Report

Xuanming Zhang, Xiaoxue Wang

September 20, 2023

1 Introduction

Decision trees are fundamental machine learning models that find applications in various domains, ranging from classification to regression tasks. They are known for their interpretability and ability to handle both categorical and numerical data.

In this report, we delve into the implementation of a decision tree learning algorithm and the construction of decision tree models from scratch. Our goal is to provide insights into the design choices, experimentation, and performance evaluation of our algorithm.

2 Model Construction and Implementation

In this section, we will explore the inner workings of decision tree learning algorithms in depth. Our algorithm aims to make data-driven decisions by recursively splitting the data based on the most informative features, with the goal of partitioning it into homogenous subsets. We will examine the key components of the algorithm, including the splitting criteria, methods for measuring impurity in the data, and techniques for pruning the tree to prevent overfitting, such as reducing error through pruning.

2.1 Algorithm Implementation

We have developed an algorithm from scratch for learning decision trees. The algorithm's key components are as follows:

When all data points share the same label, we will generate a leaf node containing that label as no further splits can reduce impurity.

If all data points have identical feature values, we will create a leaf node holding the most common label among the points.

In other cases, we select a feature that maximizes information gain as our splitting criterion. The data will then be partitioned into distinct subsets according to the values of this feature, with a branch generated for each. A recursive call to the algorithm is made for each branch to further split the data within those subsets, until we reach pure leaf nodes.

A more intuitive presentation of the entire algorithm process is shown in Figure 1.

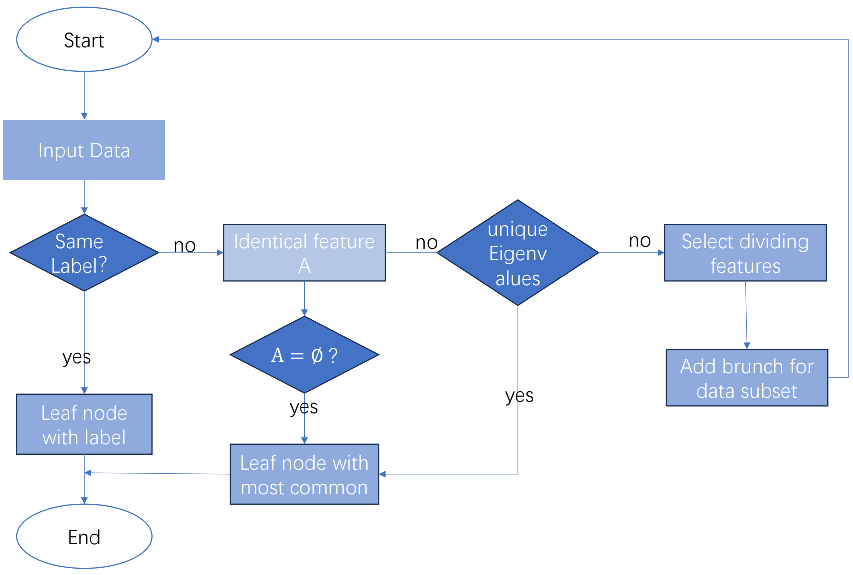


Fig.1 Algorithm Process

Given a dataset $\boldsymbol{X}$, where $\boldsymbol{P}(x)$ represents the proportion of data point x in the dataset, we can compute the information entropy $\boldsymbol{H}(X)$ as follows:

$$H(X) = -\sum_i P(x = x_i)\log_2 P(x = x_i)$$

The information entropy $\boldsymbol{H}(\boldsymbol{X})$can be used to characterize the nature of the dataset$X$. Furthermore, we can compute the information gain $\boldsymbol{IG}(\boldsymbol{X})$ as follows:

$$IG(X) = H(Y) - H(Y|X)$$

Where $\boldsymbol{IG}(\boldsymbol{X})$ represents the information gain of feature $\boldsymbol{X}$, which measures the reduction in information entropy of dataset $\boldsymbol{Y}$ after choosing feature $\boldsymbol{X}$ for splitting. This reflects the importance of feature $\boldsymbol{X}$ in classifying the dataset. $\boldsymbol{H}(\boldsymbol{Y}|\boldsymbol{X})$ represents the conditional entropy of dataset $\boldsymbol{Y}$ given feature $\boldsymbol{X}$, and it can be computed using the respective conditional probabilities.

In the first decision tree algorithm we use information gain as the splitting criterion. To utilize the IG, one can call the learning function as ` learn(X, y, impurity_measure='entropy') in our code.

2.2 Incorporating the Gini Index

We have implemented the Gini index as an alternative impurity measure. To utilize the Gini index, one can call the learning function as `learn(X, y, impurity_measure='gini')`.Given a dataset $\boldsymbol{X}$, where $\boldsymbol{P}(x_i)$ represents the proportion of data point x in the dataset, we can compute the information entropy $\boldsymbol{Gini}(X)$ as follows

$$Gini(X) = 1 - \sum_{i=1}^{k}(P(x_i))^2$$

The Gini impurity ranges from 0 to 1, where 0 indicates that the dataset is completely pure (all samples belong to the same class), and 1 indicates that the dataset is highly impure (samples from different classes are evenly distributed).

The criterion for decision tree construction is to select the feature with the smallest Gini impurity for splitting. The algorithm attempts each feature, calculates the Gini impurity of the child nodes after splitting with that feature, and then chooses the feature with the smallest Gini impurity as the splitting criterion. This selection maximizes the purity of the child nodes, enabling effective data classification.

2.3 Addressing Overfitting with Reduced-Error Pruning

To mitigate overfitting, we incorporate reduced-error pruning into our decision tree learning algorithm. This involves:

- Partitioning the dataset (X) into training (XT) and pruning (XP) subsets.
- Building a maximal tree T^* induced by XT, where each internal node performs a binary split on a feature that maximizes the splitting criterion (e.g. information gain).
- For each subtree T of T^*:
  - Let lT be the *majority class* label of examples in T projected from XT
  - Estimate the error εT of replacing T with a leaf predicting lT on XP
  - Estimate the error εT^* if T is not replaced on XP
  - If $\boldsymbol{\varepsilon T} \le \boldsymbol{\varepsilon T}^*$, prune T by replacing it with a leaf node predicting lT

By iteratively pruning subtrees where the majority class provides an error rate no worse than the subtree, we are able to counteract overfitting effects on unseen data without compromising accuracy on the training data. This improves the tree's generalization power.

3 Experimental Procedure

In this section, some specifics we will address include the metrics used to determine the "best" splits, such as information gain and Gini impurity. We will see how these metrics are optimized during the tree construction process to obtain partitions with maximal internal homogeneity. We will also discuss validation and test set evaluation to objectively gauge when additional splits begin to hurt generalization.

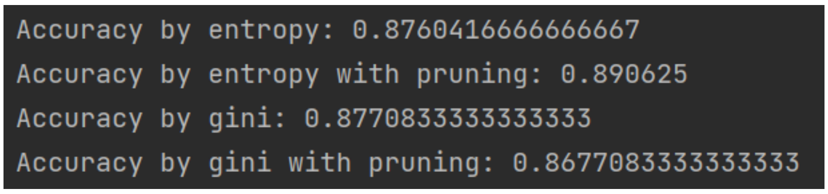
3.1 Dataset

We employed the provided `*wine_dataset.csv*` dataset, which includes 5 continuous (numerical) features: *citric acid, residual sugar, pH, sulphates, and alcohol*. The last column, "type," serves as the binary class label, where 0 denotes white wine and 1 denotes red wine. The dataset comprises 3198 rows, with exactly 1599 rows for each class.

3.2 Algorithm Performance Evaluation

We evaluated the algorithm's performance and selected optimal settings, including impurity measure (entropy or Gini) and the use of pruning. Performance metrics were applied to assess algorithm performance, utilizing training, validation, and test sets for model selection and evaluation.

Generally speaking, the four algorithms performed comparably on this dataset, with accuracies fluctuating around 87%. However, through multiple experiments, decision trees using entropy-based pruning and those using Gini impurity performed better, achieving higher accuracy scores.

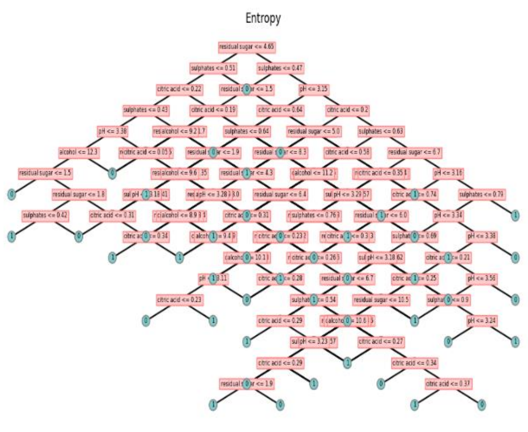


```
Accuracy by entropy: 0.8760416666666667
Accuracy by entropy with pruning: 0.890625
Accuracy by gini: 0.8770833333333333
Accuracy by gini with pruning: 0.8677083333333333
```

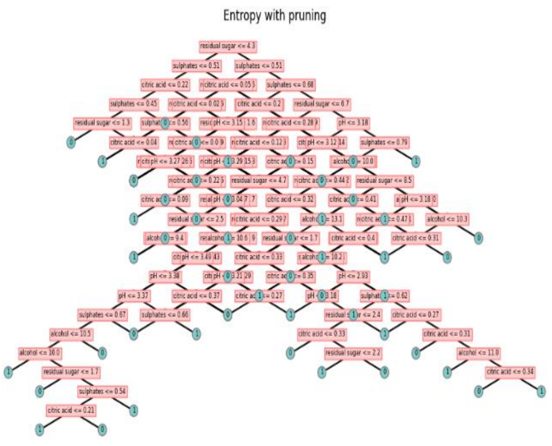
Fig.2 Output in one experiment

Simply printing out the decision tree structure is not intuitive enough. To better demonstrate the effects of different decision tree algorithms, we not only print out the trees but also visualize them (Fig.5). Through visualization, we can more clearly see the shapes of decision trees from different algorithms.

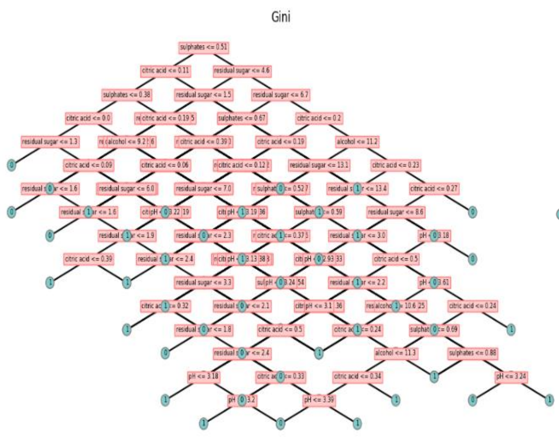
From the visualization results, it is evident that the shape of the pruned decision tree differs greatly from the pre-pruned one. The pre-pruned decision tree (Fig. 3 a&c) has an overall complex structure with many branches; while the pruned decision tree (Fig. 3 b&d) has a logically clearer overall shape with relatively simpler branches. This also verifies that pruning can remove irrelevant or low-information branches in the training data, thus obtaining a cleaner and more efficient model.



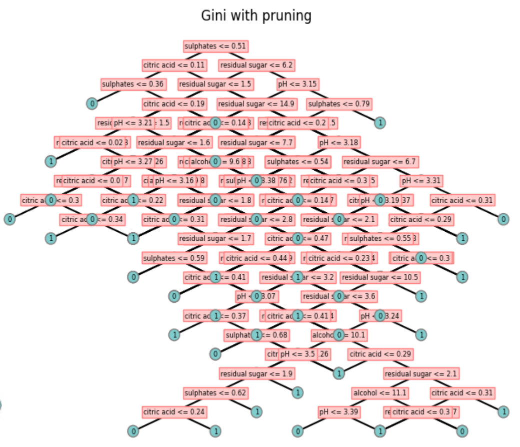
a. Entropy Decision Tree



b. Entropy Decision Tree with pruning



c. Gini Decision Tree



d. Gini Decision Tree with pruning

Fig.3 Decision Tree Visualization

In order to quantitatively assess the classification performance of different decision tree algorithms, we utilized confusion matrices and classification reports.

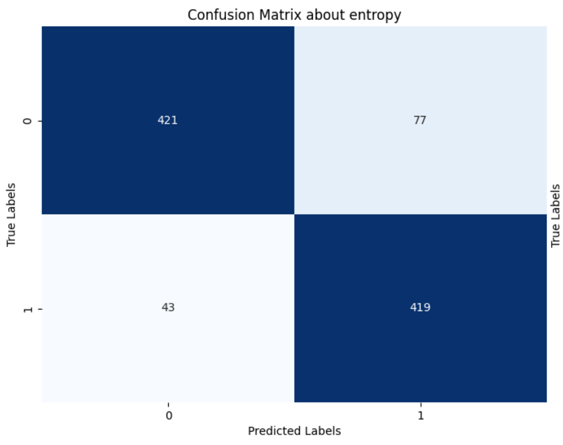
A confusion matrix reflects the situations of correct and incorrect classifications and allows us to compute metrics such as accuracy, precision, recall, and F1 score. Accuracy calculates the proportion of correctly classified instances and reflects the overall classification effectiveness of the model. Recall computes the proportion of instances in each class that were correctly classified, measuring the model's ability to identify different classes. The F1 score takes into account both precision and recall, providing an evaluation of the model's classification balance.

The classification report provides precision and recall information for each category, as well as macro and weighted averages. By analyzing the metrics for different categories, we can identify subtle differences in the model's recognition performance for specific categories. The metrics in the

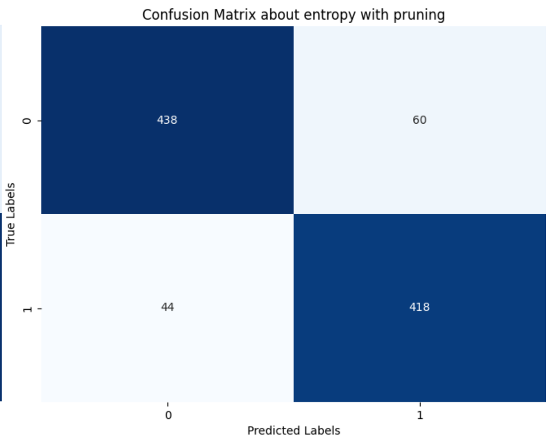
Because the coordinate settings are consistent on the left and right, leaves and branches may overlap.

"avg/total" row offer an overview of the model's classification performance on the entire dataset.

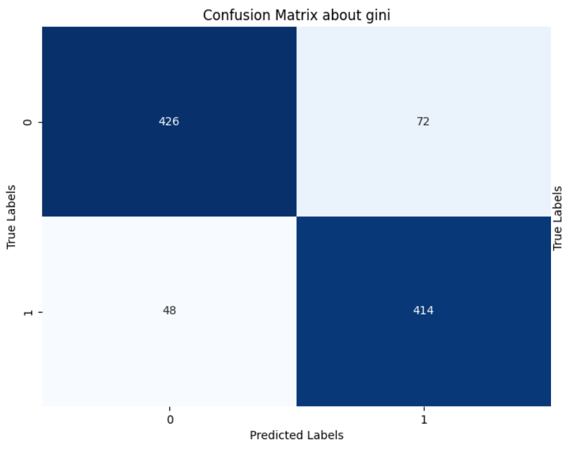
Here are the confusion matrices for four models from a recent experiment. It can be observed that the pruned entropy (Fig.4 b) method exhibits a significant improvement in classification accuracy for items with a type of 0 on the test dataset, reducing the number of misclassifications by 17.



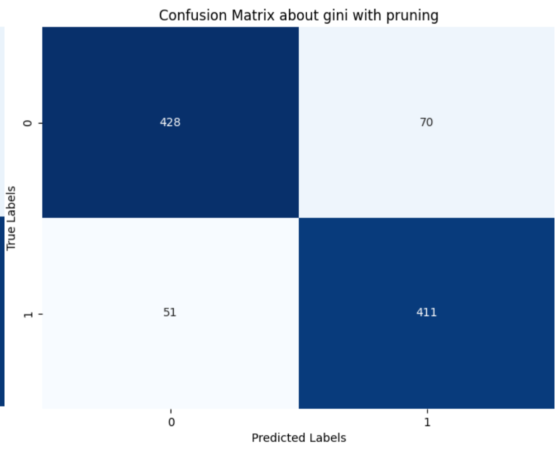
a. Entropy Decision Tree



b. Entropy Decision Tree with pruning



c. Gini Decision Tree



d. Gini Decision Tree with pruning

Fig.4 Confusion Matrix

These four classification reports provide an analysis of the performance of different models, specifically comparing models with and without pruning, and models using entropy and gini as their criteria.

Table 1 presents the performance of the model using entropy as the criterion without pruning. For class 0, it achieves a precision of 0.91, recall of 0.85, and an F1 score of 0.88, with an overall accuracy of 0.88. Class 1 also performs reasonably well with a precision of 0.84, recall of 0.91, and an F1 score of 0.87.

Table.1 Classification report about entropy

	Precision	**Recall**	**F1-score**	**Support**
0	0.91	0.85	0.88	498
1	0.84	0.91	0.87	462
Accuracy			0.88	960
Macro avg	0.88	0.88	0.87	960

Weighted avg	0.88	0.88	0.88	960

Table 2 indicates an improvement in performance for the entropy-based model after pruning. Both class 0 and class 1 exhibit increased precision, recall, and F1 scores, resulting in an accuracy of 0.89.

Table.2 Classification report about entropy with pruning

	Precision	**Recall**	**F1-score**	**Support**
0	0.91	0.88	0.89	498
1	0.87	0.90	0.89	462
Accuracy			0.89	960
Macro avg	0.89	0.89	0.89	960
Weighted avg	0.89	0.89	0.89	960

Table 3 and Table 4 describe models using the Gini criterion, with Table 3 representing the non-pruned Gini model and Table 4 representing the pruned Gini model. The non-pruned Gini model shows relatively high precision and recall for both class 0 and class 1, resulting in an overall accuracy of 0.88. However, after pruning, the Gini model's performance slightly improves, with increased precision, recall, and F1 scores, although the accuracy decreases slightly to 0.87.

Table.3 Classification report about gini

	Precision	**Recall**	**F1-score**	**Support**
0	0.90	0.86	0.88	498
1	0.85	0.89	0.87	462
Accuracy			0.88	960
Macro avg	0.88	0.88	0.87	960
Weighted avg	0.88	0.88	0.88	960

Table.4 Classification report about gini with pruning

	Precision	**Recall**	**F1-score**	**Support**
0	0.89	0.86	0.88	498
1	0.85	0.89	0.87	462
Accuracy			0.87	960
Macro avg	0.87	0.87	0.87	960
Weighted avg	0.87	0.87	0.87	960

Pruning decision trees appears to have a positive impact on both entropy and Gini models, with the pruned entropy model performing the best in this experiment, achieving an accuracy of 0.89.

3.3 Comparison with Existing Implementation

We conducted a comparative analysis of our implementation with existing decision tree implementations, such as the DecisionTreeClassifier class in scikit-learn. Similarly, we visualized the tree diagrams and obtained corresponding confusion matrices and classification reports.

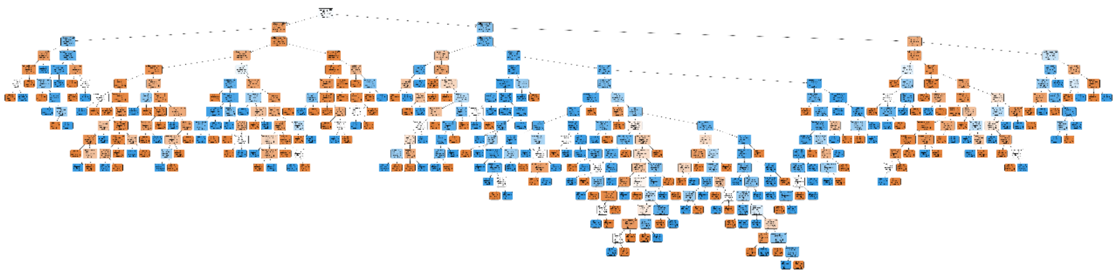


Fig.5 Decision Tree Visualization

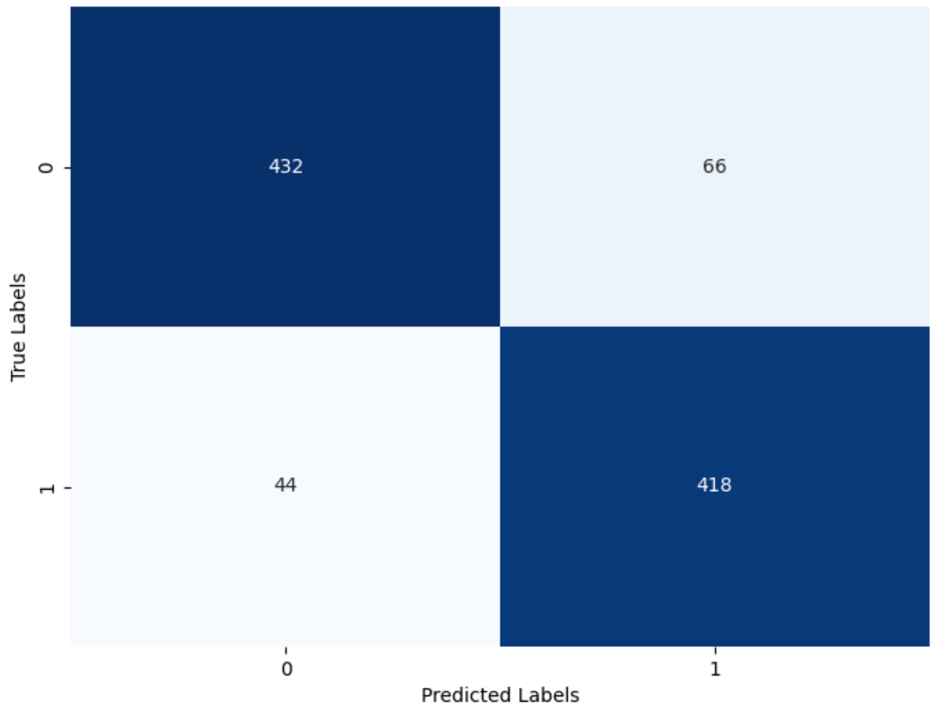


Fig.6 Confusion Matrix

Table.5 Classification report about DecisionTreeClassifier class from scikit-learn

	Precision	**Recall**	**F1-score**	**Support**
0	0.91	0.87	0.89	498
1	0.86	0.90	0.88	462
Accuracy			0.89	960
Macro avg	0.89	0.89	0.89	960
Weighted avg	0.89	0.89	0.89	960

When using the template, the accuracy consistently stabilized at around 0.885, and multiple repeated experiments yielded consistent results. However, in our custom decision tree implementation, different experimental runs resulted in varying accuracy levels.

After further exploration and inspecting the original code of the decision tree, we initially believe that the inconsistency in the results is due to the fact that our decision tree doesn't constrain the maximum depth of the tree, and the data splitting points of the model are unstable. This instability leads to inconsistent results across different experimental runs.

4 Conclusion

PROJECT 1

This report presents a decision tree learning algorithm that uses information gain and Gini impurity for feature selection. Pruning is employed to prevent overfitting, enhancing classification accuracy on the wine dataset. The algorithm's performance, especially with information gain and pruning, achieves up to 89% accuracy. Visual comparisons and quantitative assessments confirm its effectiveness, while a sklearn comparison validates its competitiveness. However, some variability exists, suggesting potential improvements through depth constraints. Overall, this report offers insights and a reference for decision tree algorithm design.

Contributions
Xuanming Zhang: Decision tree algorithm, Code, Error Analysis

Xiaoxue Wang: post-pruning algorithm, visualization, code, write the report