Universidad Autónoma de Querétaro
Facultad de Ingeniería

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

November 9, 2023

Author:*Lic. Ijtsi Dzaya Ramos Morales*

Machine Learning, Dr. Marco Antonio Aceves Fernández

Report 4: Decision Trees

Abstract

A custom decision tree algorithm was assessed for drug classification. Preprocessing steps, including numerical value transformation and controlled Synthetic Minority Over-sampling Technique (SMOTE) application, enhance dataset quality. Comparative analysis with a scikit-learn classifier reveals competitive results, though the custom algorithm displays limitations potentially related to class imbalance and hyperparameter adjustments. Future work involves refining the custom algorithm and exploring optimization strategies.

Introduction

The dual objective of this report is to develop a decision tree algorithm from the ground up and to conduct a comparative analysis against an established decision tree library. Decision trees play a central role in data analysis, and comprehending their inner workings while constructing a custom solution is a valuable endeavor. Constructing a decision tree algorithm from scratch offers an opportunity to gain deep insights into the intricacies of decision tree construction. Subsequently, the algorithm's accuracy, efficiency, and scalability will be evaluated in contrast to a well-established decision tree library, shedding light on the practical implications of this undertaking in real-world data analysis and predictive modeling.

Theoretical Foundation

Decision trees are versatile tools for data analysis and decision-making. They are commonly used for classification and regression tasks in machine learning. A decision tree is a tree-like model that maps features to outcomes. It comprises nodes, edges, and leaves, where nodes represent feature tests, edges denote the decision outcomes, and leaves contain the predicted values.

Decision Tree Structure

A decision tree is composed of nodes, with each node corresponding to a feature or attribute of the data. At the root of the tree is the initial node, which represents the entire dataset. Branches emanate from each node, leading to child nodes that represent subsets of the data. Internal nodes contain decision rules based on feature values, and terminal nodes, or leaves, provide the final predictions.

The decision at each node is typically based on a threshold or rule that divides the data into two or more subsets. In classification tasks, these subsets correspond to different classes, while in regression tasks, they represent different values or ranges.

$$\text{Impurity}(D) = 1 - \sum_{i=1}^{c} p_i^2 \tag{1}$$

Equation (1) demonstrates how impurity (often measured using Gini impurity or entropy) is calculated for a node. Here, p_i represents the proportion of instances in class i, and c is the total number of classes.

Decision Tree Construction

The construction of a decision tree involves selecting the best attribute to split the data. Several algorithms, such as ID3, C4.5, and CART, use different criteria like information gain or Gini index to make this decision. Information gain measures the reduction in entropy or impurity after a split, aiming to maximize the purity of the child nodes.

$$\begin{aligned} \text{f}(D, A) = \text{Entropy}(D) \\ - \sum_{v \in \text{values}(A)} \frac{|D_v|}{|D|} \cdot \text{Entropy}(D_v) \end{aligned} \tag{2}$$

Equation (2) represents the formula for information gain. Here, D is the dataset, A is the attribute to be split, values(A) are the possible values of attribute A, and D_v represents the subset of data with attribute A equal to v.

Pruning and Overfitting

Decision trees are prone to overfitting, which occurs when the tree is excessively complex and fits the training data noise. Pruning is a technique used to prevent overfitting by removing branches that do not significantly improve the model's predictive power.

Materials and Methods

1. **Data Set:** The dataset used in this study was sourced from Kaggle. It comprises information relevant to the classification of hypothetical drugs, contributing to a comprehensive understanding of classification using decision trees. The dataset consists of 200 observations, encompassing 5 attributes, including laboratory values and demographic information from both blood donors and hepatitis C patients. The attributes are predominantly categorical, with exceptions in the "Age" and "Na-to-K" attributes.
2. **Software:** The analysis was conducted using Python, a versatile programming language. The following libraries were employed for numerical computations, data manipulation, and graphical analysis: `numpy`, `pandas`, `matplotlib`, `sklearn`, `seaborn`, and `random`.
3. **Methods:**
    - **Decision Tree Algorithm Construction:** The decision tree is a machine learning algorithm that recursively partitions the dataset based on the features to create a tree-like structure.
        (a) **Tree Node Representation:** The decision tree is represented as a collection of nodes. Each node contains information about a specific feature and a threshold for splitting the data. The nodes are organized in a hierarchical manner, with parent and child relationships.
        (b) **Entropy and Information Gain:** Entropy is a measure of impurity in a dataset. The decision tree aims to minimize entropy by selecting features that result in the most homogeneous subsets. Information gain is a metric used to evaluate the effectiveness of a feature in reducing entropy. Features with higher information gain are preferred for splitting the data.
        (c) **Recursive Splitting:** The algorithm recursively splits the dataset based on the feature with the highest information gain at each node. This process continues until a stopping criterion is met, such as reaching a maximum depth or achieving a minimum number of samples in a node.
        (d) **Leaf Node Prediction:** Once a stopping criterion is met, the algorithm assigns a prediction to each leaf node. In classification tasks, the majority class of the samples in the leaf node is used as the prediction.
        (e) **Evaluation:** The performance of the custom decision tree algorithm is assessed in comparison to the `DecisionTreeClassifier` from `sklearn`. Metrics such as accuracy, precision, recall, and F1 score are used to evaluate the algorithm's effectiveness in making accurate predictions.
    - **External Libraries:** Additionally, the `DecisionTreeClassifier` from `sklearn` was utilized for comparative analysis. This established library provides a baseline for evaluating the performance of the custom decision tree algorithm.
    - **Data Preprocessing:** Prior to algorithm implementation, the dataset underwent preprocessing using Python. Techniques such as data scaling, splitting into training and testing sets, and handling imbalanced data were applied.

Results

Figure 1 illustrates the results of various preprocessing steps undertaken to refine the dataset before implementing the decision tree algorithm. Three tables are presented, each representing a distinct stage of preprocessing. The final table at the bottom of the figure encapsulates the refined dataset utilized in the subsequent stages of the algorithm.

- **Table 1:** This table displays the dataset after the initial import, showcasing the raw information obtained from the source. It includes all attributes and instances without any modifications.
- **Table 2:** After the initial import, Table 2 demonstrates the dataset following basic cleaning procedures. This involves transforming the continuous or numerical values into intervals in order to reduce the observations.
- **Table 3:** The third table represents the final dataset after feature engineering and additional preprocessing steps. This stage involves scaling numerical features, encoding categorical variables, and any other transformations deemed necessary for algorithmic compatibility.

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

	Sex	BP	Cholesterol	Drug	NatoKIntervals	AgeInterval
0	F	HIGH	HIGH	DrugY	3	0
1	M	LOW	HIGH	drugC	2	2
2	M	LOW	HIGH	drugC	0	2
3	F	NORMAL	HIGH	drugX	0	0
4	F	LOW	HIGH	DrugY	3	3

	Sex	BP	Cholesterol	Drug	NatoKIntervals	AgeInterval
0	0	0	0	0	3	0
1	1	1	0	1	2	2
2	1	1	0	1	0	2
3	0	2	0	2	0	0
4	0	1	0	0	3	3

Figure 1: Dataset before and after preprocessing

Figure 2 presents the distribution of the target attribute within the dataset before and after data augmentation.

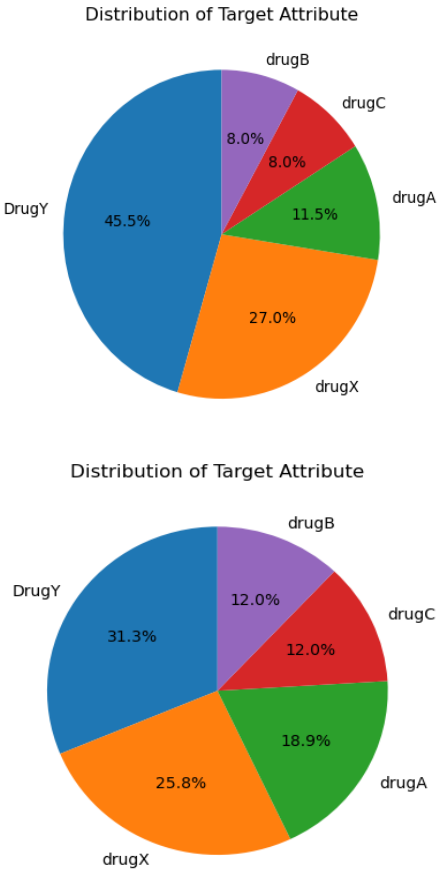
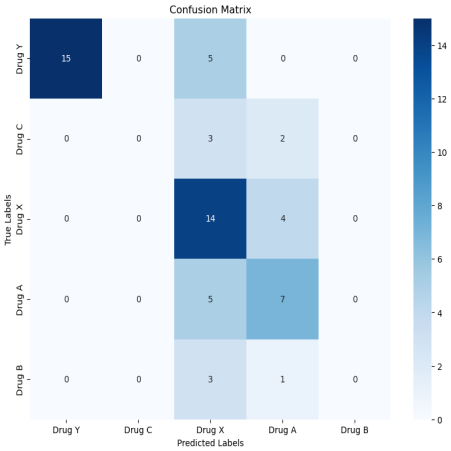


Figure 2: Distribution of target attribute before and after SMOTE

Figure 3 presents the outcomes derived from the classification process executed using the custom-built algorithm.

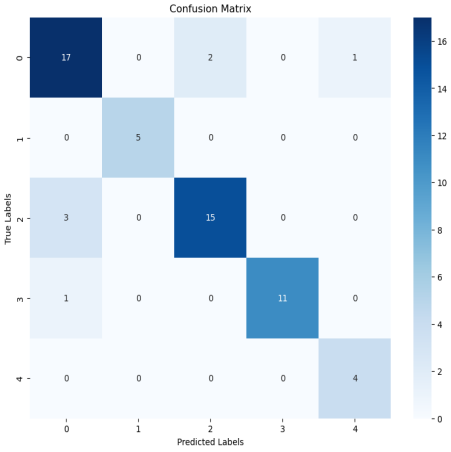


Classification Report:

	precision	recall	f1-score	support
0	1.00	0.75	0.86	20
1	0.00	0.00	0.00	5
2	0.47	0.78	0.58	18
3	0.50	0.58	0.54	12
4	0.00	0.00	0.00	4

Figure 3: Metrics of the self-made algorithm

Figure 4 presents the results from the classification that was performed using the sklearn algorithm.



Classification Report:

	precision	recall	f1-score	support
0	0.81	0.85	0.83	20
1	1.00	1.00	1.00	5
2	0.88	0.83	0.86	18
3	1.00	0.92	0.96	12
4	0.80	1.00	0.89	4

Figure 4: Elbow method results for Clarans

Discussion

Preprocessing

In the preprocessing phase, several crucial steps were undertaken to enhance the quality of the dataset. First, numerical values were transformed into intervals, a strategic decision aimed at reducing the overall number of observations. Additionally, the Synthetic Minority Over-sampling Technique (SMOTE) was employed to address class imbalance while carefully controlling the generation of new data to prevent an excessive increase in sample size. Furthermore, the numerical attribute "Na_to_K" underwent scaling using the StandardScaler to ensure consistent and comparable features. The cumulative impact of these preprocessing steps resulted in improved classification results.

Classification Results

Figure 3 and Figure 4 depict the classification results obtained from both the scikit-learn (sklearn) classifier and the custom-built classifier. Notably, the custom classifier exhibited a limitation by not predicting two variables and providing predictions for only three labels instead of the five present in the dataset. Several factors may contribute to this discrepancy.
The unbalanced nature of the dataset could have influenced the custom classifier's performance. Class imbalance can lead to biased models that prioritize the majority class. Additionally, the inherent complexity of the custom algorithm may have affected its ability to handle certain patterns in the data. Adjustments to hyperparameters, such as `max_depth`, could play a crucial role in the algorithm's behavior.
The `max_depth` hyperparameter in decision trees represents the maximum depth of the tree. It determines the number of nodes and branches the tree can have. A higher `max_depth` allows the tree to capture more complex relationships in the data, potentially leading to overfitting. Conversely, a lower `max_depth` may result in a simpler tree, potentially underfitting the data.

Conclusion

The preprocessing steps, including numerical value transformation, controlled SMOTE application, and feature scaling, significantly improved the dataset's quality.
The comparison with the scikit-learn classifier highlighted the trade-offs between the custom solution and a well-established library. The scikit-learn classifier demonstrated superior performance, emphasizing the importance of leveraging established tools for certain tasks. The observed discrepancies in the custom classifier's predictions warrant further investigation, particularly regarding hyperparameter adjustments and addressing class imbalance.
Future work may involve refining the custom algorithm, exploring additional preprocessing techniques, and fine-tuning hyperparameters to achieve more robust and generalizable results.

References

[1] C. C. Aggarwal and C. K. Reddy, *Data clustering: Algorithms and applications*. Chapman and Hall/CRC, 2014. (Book)

[2] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*, pp. 25-71, Springer, 2006. (Book Chapter)

[3] M. A. Aceves Fernández, "Inteligencia Artificial para Programadores con Prisa," 2023.