The Saigon International University

FACULTY OF ENGINEERING & COMPUTER SCIENCE

FINAL PROJECT REPORT

SUBJECT: ARTIFICIAL INTELLIGENCE

PROJECT TITLE:

DECISION TREE MODEL UTILIZATION FOR INSIGHT EXTRACTION FROM TABULAR DATASETS

STUDENTS:

DO HOANG VU
STUDENT ID: 31012302902
CLASS: 23DAI

NGUYEN MINH TU
STUDENT ID: 74802302853
CLASS: 23HTDL

SUPERVISOR:
PhD HO LONG VAN

Semester 1 – Academic Year 2024–2025

ACKNOWLEDGEMENT

We would like to extend our profound and heartfelt gratitude to our respected supervisor, PhD Ho Long Van, for his unwavering guidance, insightful advice, and steadfast support throughout the entire process of this project. His deep expertise and thoughtful mentorship have not only illuminated our path in exploring the fascinating domain of Artificial Intelligence but also inspired us to strive for excellence. The knowledge and skills we have gained under his guidance will undoubtedly serve as a cornerstone for our future academic and professional endeavors.

Furthermore, we are sincerely grateful to our peers for their constructive feedback and invaluable camaraderie, which have fostered a collaborative and enriching environment during this project. Lastly, we would like to express our deepest appreciation to our families, whose encouragement, understanding, and unwavering belief in our abilities have been a source of immense strength and motivation. Their support has been instrumental in enabling us to dedicate ourselves fully to this work.

Table of Contents

Decision Tree Model Utilization For Insight Extraction From Tabular Datasets

Vu Do Hoang, Tu Nguyen Minh
The Saigon International University
dohoangvuk16@siu.edu.vn, nguyenminhtuk16@siu.edu.vn

December 24, 2024

Abstract

This research explores the utilization of decision tree models for extracting meaningful insights from tabular datasets, focusing specifically on classification tasks. The study implements and analyzes a decision tree-based approach, incorporating Gini Index and Entropy metrics for node splitting decisions, while addressing stopping conditions and pruning techniques to prevent overfitting.
The project develops an interactive web-based visualization platform that enables users to upload datasets, customize tree parameters, and visualize decision structures. Using the Drug Classification dataset as a case study, the research demonstrates decision trees' practical application, highlighting the model's ability to process both categorical and numerical data while maintaining interpretability.
The methodology encompasses data preprocessing, model construction, and evaluation using accuracy, precision, recall, and F1-score metrics. The implementation addresses technical considerations including missing value handling, outlier removal via IQR method, and strategic data splitting for validation.
Results demonstrate that decision trees effectively facilitate pattern recognition and decision-making in structured data analysis, particularly in domains requiring model transparency. The research contributes both a theoretical framework and practical visualization tool, while identifying future opportunities in scalability enhancement and integration with advanced machine learning techniques.

Keywords: Decision Trees, Classification, Machine Learning, Data Visualization, Pattern Recognition.

1. INTRODUCTION

1.1. Background

Decision trees are among the most influential and interpretable tools in Artificial Intelligence, finding extensive applications in classification and regression tasks. Their ability to model complex decision-making processes in an intuitive, hierarchical manner makes them indispensable for analyzing and extracting valuable insights from tabular datasets.

This project delves deeply into the theoretical foundation and practical implementation of decision tree models, focusing on their capability to uncover patterns and relationships within structured data. By leveraging this model, we aim to demonstrate their utility in solving real-world problems across diverse domains.

In addition to building and evaluating decision tree models, this project also emphasizes the development of an interactive and user-friendly web application. This platform enables users to visualize binary trees generated from any valid tabular dataset, providing a clear representation of the decision-making process. The web interface is designed to be intuitive, allowing users to upload datasets, customize parameters, and explore the resulting decision trees interactively. Such a visualization tool not only enhances understanding but also serves as a bridge between technical model outputs and user-friendly insights, catering to both technical professionals and non-specialist audiences.

Through this combined approach of model implementation and visualization, the project aspires to contribute meaningfully to the understanding and application of decision tree methodologies in data-driven decision-making.

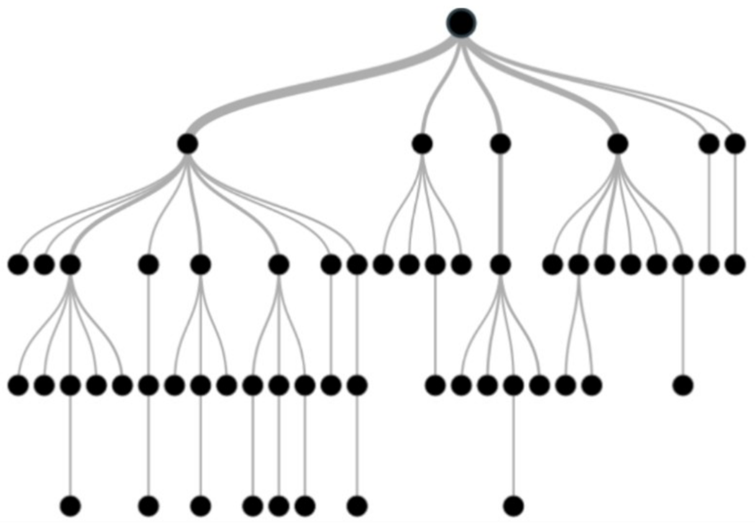


Figure 1.1: The Structure of a Decision Tree

1.2. Objectives

The primary objective of this study is to build and analyze a decision tree model to enhance interpretability and uncover significant patterns within datasets. By developing a decision tree model, the study aims to provide clear, understandable rules that can be easily interpreted and used for future predictions, ultimately facilitating decision-making processes in practical applications.

In general, decision trees are commonly applied to both classification and regression problems. In classification, the results are typically discrete and unordered (e.g., a weather prediction model forecasting whether it will rain on a specific day). In regression, the data includes feature vectors that may contain both categorical and continuous attributes, which are meaningful (e.g., a profit prediction model estimating the potential profit generated from selling a product). Within the scope of this project, our team will focus on the classification model of decision trees (Decision Tree Classification).

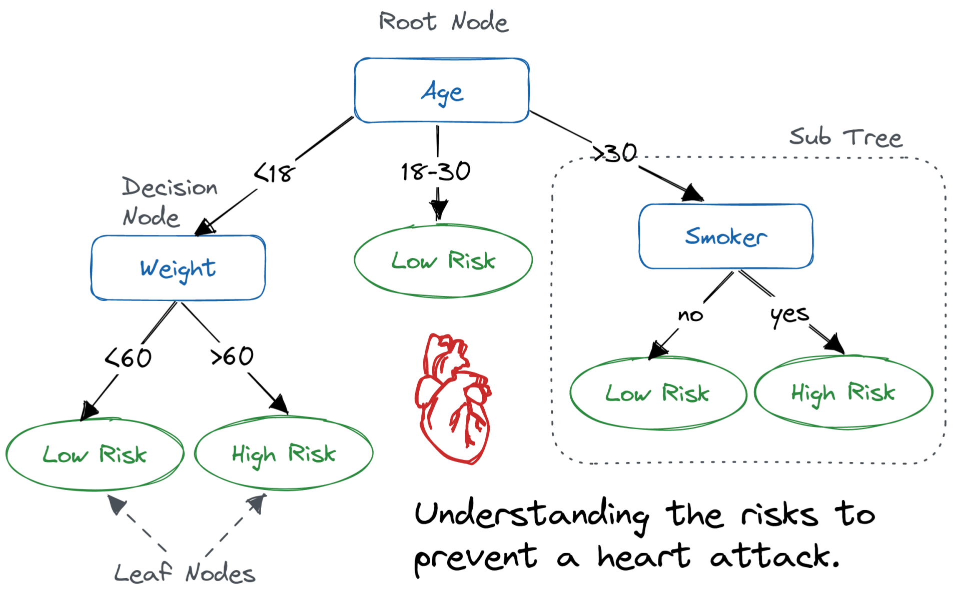


Figure 1.2: Example of a Decision Tree

1.3. Scientific and Practical Significance

- **Scientific Importance:** This research enhances our understanding of Decision Tree models in AI, focusing on how these models can be applied to extract valuable insights from tabular data. The study contributes to both theoretical knowledge and practical applications of AI in data analysis.
- **Practical Importance:** The findings can be applied in a variety of domains where decision-making is essential. Decision Trees can help industries such as finance, healthcare, and marketing make better, data-driven decisions, improve efficiencies, and uncover hidden patterns within datasets.

2. RELATED WORK

In recent years, decision tree models have gained significant traction in the field of artificial intelligence and machine learning due to their interpretability and efficiency in handling tabular datasets. Previous studies have extensively explored the utilization of decision trees for both classification and regression tasks across diverse domains. Quinlan's seminal work on ID3 and later developments such as C4.5 and CART laid the foundational principles of decision tree algorithms. These models introduced mechanisms for handling categorical and numerical data while addressing challenges like overfitting through pruning techniques. Subsequent research has enhanced these methods by incorporating ensemble approaches, such as Random Forests and Gradient Boosted Trees, to improve predictive accuracy. [1]

In the context of tabular datasets, decision tree models have been employed in various sectors including healthcare, finance, and marketing. For example, decision trees have been used for predicting patient outcomes, credit scoring, and customer segmentation, demonstrating their versatility and reliability.

Moreover, recent advancements in hybrid models have integrated decision trees with deep learning frameworks to leverage structured data alongside unstructured inputs. These approaches, while computationally intensive, have shown promise in improving model performance in complex data scenarios.

The methodologies explored in this project draw inspiration from these foundational and contemporary works, aiming to optimize decision tree models for extracting actionable insights from tabular datasets. This report also considers the limitations identified in prior research, such as sensitivity to data noise and bias in feature splits, proposing strategies to mitigate these challenges.

3. THEORETICAL BACKGROUND

3.1. Decision Tree Algorithm

Decision trees are tree-based algorithms widely utilized in both regression and classification tasks. They are part of the family of supervised learning algorithms, which operate by analyzing the features and characteristics of data points and training a model within a tree-like structure. This process involves recursively partitioning the dataset into smaller subsets, and making predictions based on these partitions. Due to their simplicity and interpretability, decision trees are commonly applied to complex datasets and have proven to be effective in various domains, such as finance, healthcare, and marketing. The main focus of this project is to explore the application of decision tree models in classification tasks to extract meaningful insights from tabular datasets.

3.2. Different Stages of the Model

At its core, a decision tree consists of three main components:

- Decision Nodes: Represent decisions.
- Chance Nodes: Represent probabilities or uncertainties.
- Terminal Nodes: Represent outcomes.

The nodes are connected by branches. Nodes and branches can be used repeatedly in various combinations to create trees of varying complexity.

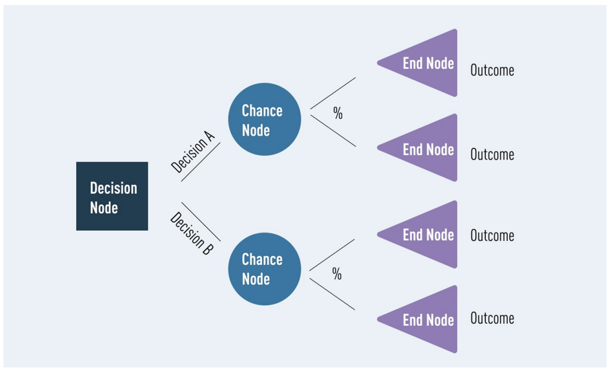


Figure 3.1: Example of Node and Branch Connections

In essence, both classification trees and regression trees operate based on the same underlying model. The key difference lies in their outputs: if the tree's prediction target is categorical, it is a classification tree; if the target is a real number, it is a regression tree.

3.3. Key Characteristics of Decision Trees

3.3.1. Gini Index

The Gini Index is a metric used to evaluate the impurity of a dataset by measuring the probability of incorrectly classifying a randomly chosen element if it were labeled according to the class distribution. It assesses the inequality or diversity within the dataset. A Gini Index value of 0 represents perfect purity (all elements belong to a single class), while a value close to 1 indicates high impurity. The formula for calculating the Gini Index is as follows:

$$Gini(S) = 1 - \sum_{i=1}^{n} p_i^2$$

Where n is the total number of classes, and p_i represents the proportion of elements belonging to the i-th class. The Gini Index is widely used in algorithms such as CART (Classification and Regression Trees) for splitting nodes in a decision tree.

3.3.2. Entropy Index

Entropy is a concept derived from information theory, measuring the level of disorder, uncertainty, or impurity within a dataset. It quantifies the homogeneity of a dataset, with lower entropy indicating higher purity and higher entropy reflecting greater impurity. Decision tree algorithms such as ID3, C4.5, and C5.0 utilize entropy to evaluate splits and select attributes that maximize data homogeneity. The formula for entropy is given by:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Where p_i is the probability of occurrence of the i-th class, and c is the total number of classes. When entropy is combined with information gain, it becomes a powerful tool for guiding the construction of decision trees.

3.3.3. Information Gain

Information Gain measures the reduction in entropy achieved by partitioning a dataset based on a specific attribute. It quantifies how effectively an attribute separates the dataset into subsets containing elements of a single class. Attributes with higher information gain are more effective at reducing impurity and are therefore preferred as decision nodes in the tree. The formula for information gain is:

$$\text{Information Gain} = E(Y) - E(Y|X)$$

Where $E(Y)$ is the entropy of the target variable Y, and $E(Y|X)$ is the conditional entropy of Y given the attribute X. By selecting attributes that maximize information gain, decision tree algorithms prioritize splits that provide the most significant improvement in data homogeneity.

3.3.4. Stopping Conditions

The growth of a decision tree must be carefully controlled to prevent overfitting, improve generalization, and reduce computational complexity. Common stopping conditions include:

- **Maximum Tree Depth:** The depth of the tree is limited using the 'max_depth' parameter to avoid excessive growth and overfitting. A shallower tree generalizes better than unseen data.
- **Minimum Samples for a Split:** A node will only split if the number of samples it contains meets a predefined threshold, controlled by the 'min_samples_split' parameter. This prevents splitting on nodes with insufficient data, reducing noise.
- **Minimum Samples per Leaf:** The 'min_samples_leaf' parameter specifies the minimum number of samples that must be present in a leaf node. This ensures that each leaf node contains enough data to make reliable predictions.
- **Pure Leaf Nodes:** Splitting stops when a node achieves purity, meaning all samples in the node belong to the same class. This indicates that further division will not improve classification performance.
- **Maximum Number of Splitting Nodes:** To limit model complexity, the tree stops splitting when the number of decision nodes reaches a predefined limit, ensuring the tree remains interpretable.
- **Maximum Number of Leaf Nodes:** The tree's growth is constrained by limiting the total number of leaf nodes using the 'max_leaf_nodes' parameter. This prevents over-complexity and improves computational efficiency.

By employing these stopping conditions, decision trees strike a balance between model accuracy and interpretability, ensuring their practical utility in real-world applications.

3.4. Advantages and Disadvantages

Advantages:

- **Simplicity and Interpretability:** Decision trees are intuitive and easy to understand, even for non-experts. The structure represents decision rules, making them transparent and explainable.
- **Minimal Data Preprocessing:** They require minimal preprocessing, such as handling missing values or using categorical data directly without normalization.
- **Versatility:** Decision trees work with both numerical and categorical data, making them widely applicable.
- **White-Box Nature:** Their transparent nature allows easy interpretation and explanation of predictions using Boolean logic.
- **Scalability:** They handle large datasets efficiently and are suitable for real-world applications.

- **Non-Parametric Nature:** As non-parametric models, they adapt to data distributions without assuming specific patterns.

Disadvantages:

- **Sensitivity to Data Variations:** Small changes in data can lead to significant changes in tree structure, affecting consistency.
- **Overfitting Risk:** Decision trees may grow overly complex, capturing noise in the data and reducing generalization. Techniques like pruning and depth limitation can mitigate this.
- **Bias Risk:** Imbalanced datasets can result in biased trees favoring dominant classes. Balancing data is essential.
- **Computational Cost:** Building a tree can be computationally intensive for large or high-dimensional datasets.
- **Outlier Sensitivity:** Extreme values can distort splits, reducing accuracy and robustness.
- **Piecewise Predictions:** Predictions are less smooth due to the discrete nature of tree splits.

4. METHODOLOGY

The research employs a systematic empirical approach, with clearly defined steps to ensure the robustness and reliability of the data analysis process. This chapter outlines the methodology adopted, with a focus on data preprocessing and preparation for the Decision Tree model.

4.1. Data Preprocessing

The dataset undergoes a series of preprocessing steps to ensure it is clean, consistent, and suitable for modeling. The steps are as follows:

4.1.1. Reading and Cleaning Data

The data preprocessing begins with loading the dataset and cleaning it to remove inconsistencies:

- **Loading Data:** The dataset is read from a CSV file. It supports flexible delimiters such as commas or semicolons to accommodate various data formats.
- **Removing Missing Values:** Rows with missing values are removed to ensure data integrity and prevent errors during model training.
- **Dropping Unnecessary Columns:** Columns irrelevant to the target analysis are dropped, as specified in the preprocessing configuration.

Algorithm 1: Reading and Cleaning Data

```
  Data: Input CSV file containing the dataset
  Result: Cleaned DataFrame without missing or unnecessary columns
1 Load dataset from the specified CSV file; Remove rows with missing values;
2 If configured, drop specified unnecessary columns;
3 Print column names for verification;
```

4.1.2. Removing Outliers

Outliers are removed based on the Interquartile Range (IQR) method to improve the robustness of the model:

- **Compute IQR:** Calculate the first (Q1) and third quartiles (Q3) for each numerical column.
- **Define Bounds:** Determine the lower and upper bounds as $[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$.

- **Filter Data:** Retain only rows with values within the defined bounds for each column.

Algorithm 2: Removing Outliers

```
   Data: DataFrame with numerical columns
   Result: DataFrame without outliers
1  Copy the original DataFrame; Identify numerical columns in the DataFrame; For
     each numerical column: begin
2      Compute Q1 and Q3; Calculate IQR as $Q3 - Q1$;
3      Define lower bound as $Q1 - 1.5 \times IQR$;
4      Define upper bound as $Q3 + 1.5 \times IQR$;
5      Filter rows within these bounds;
6  end
```

4.1.3. Data Splitting

The dataset is split into training and testing subsets to evaluate model performance:

- **Define Split Ratio:** The proportion of data allocated to training (default: (80%) and testing (20%)).
- **Random Shuffling:** Rows are randomly shuffled to prevent order bias.
- **Split Data:** Partition the data into training and testing sets based on the split ratio.

Algorithm 3: Data Splitting

```
   Data: Cleaned DataFrame, split ratio (default: 0.8)
   Result: Training and testing sets (X_train, X_test, y_train, y_test)
1  Randomly shuffle the dataset;
2  Calculate the split index based on the ratio;
3  Split feature matrix $X$ and target vector $y$ into training and testing sets;
```

4.2. Model Construction

The Decision Tree model is implemented and constructed through the following detailed steps.

4.2.1. Defining the Tree Structure

The Decision Tree is initialized with customizable parameters such as maximum depth, minimum samples per split, and the criterion for splitting (Gini Index or Entropy).

Algorithm 4: Tree Initialization

```
Input: max_depth, min_samples_split, criterion, max_features
Output: Initialized Decision Tree
Procedure InitializeTree(max_depth, min_samples_split, criterion, max_features)
    self.max_depth ← max_depth;
    self.min_samples_split←min_samples_split;
    self.criterion ← criterion;
    self.tree ← None;
```

4.2.2. Measuring Impurity

The impurity of a dataset is measured using either the Gini Index or Entropy.

Algorithm 5: Measuring Impurity

```
Input: Class labels y
Output: Impurity value
Function GiniIndex(y)
    gini ← 1.0;
    foreach class c in unique(y) do
        $p_c$ ← count(c)/len(y);
        gini ← gini $-p_c^2$;
    end
    return gini;
Function Entropy(y)
    entropy ← 0.0;
    foreach class c in unique(y) do
        $p_c$ ← count(c)/len(y);
        if $p_c > 0$ then
            entropy ← entropy $-p_c \cdot \log_2(p_c)$;
        end
    end
    return entropy;
```

4.2.3. Finding the Best Split

The algorithm evaluates all possible splits across features and their unique values to maximize information gain.

Algorithm 6: Finding the Best Split

```
Input: Feature matrix X, labels y
Output: Best gain, best split column, best split value
Function FindBestSplit(X, y)
    best_gain ← 0;
    best_split_col, best_split_value ← None, None;
    current_uncertainty ← CalculateUncertainty(y);
    foreach column col in range(X.shape[1]) do
        foreach value val in unique(X[:, col]) do
            left, right ← Split (X, y, col, val);
            if len(left) = 0 or len(right) = 0 then
                continue;
            end
            left_y, right_y ← labels from splits;
            gain ← InformationGain(left_y, right_y,
              current_uncertainty);
            if gain > best_gain then
                best_gain, best_split_col, best_split_value ← gain, col,
                  val;
            end
        end
    end
    return best_gain, best_split_col, best_split_value;
+
```

4.2.4. Recursive Tree Building

The tree is constructed recursively by finding the best split at each node until a stopping condition is met.

Algorithm 7: Recursive Tree Building

```
Input: Feature matrix X, labels y, depth
Output: Decision Tree Node
Function BuildTree(X, y, depth)
    if len(unique(y)) = 1 or depth = max_depth or len(y) <
      min_samples_split then
        return Leaf Node;
    end
    best_gain, col, value ← FindBestSplit (X, y);
    if best_gain = 0 then
        return Leaf Node;
    end
    left, right ← Split (X, y, col, value);
    return Node with BuildTree (left, depth+1) and BuildTree (right,
      depth+1);
```

4.3. Evaluation Metrics

The performance of the implemented Decision Tree model is thoroughly evaluated using several standard classification metrics. These metrics include accuracy, precision, recall, and F1-score. Each metric provides unique insights into the model's classification capabilities, ensuring a comprehensive evaluation.

4.3.1. Accuracy

Accuracy is the proportion of correctly classified samples out of the total samples. It is calculated as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of samples}}$$

Accuracy provides a general measure of performance but can be misleading in imbalanced datasets.

4.3.2. Precision

Precision measures the proportion of true positive predictions among all positive predictions. It is given by:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Precision is particularly useful when the cost of false positives is high.

4.3.3. Recall

Recall, or sensitivity, measures the proportion of actual positive samples correctly identified. It is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

High recall indicates the model is effective in identifying positive samples.

4.3.4. F1-score

The F1-score is the harmonic mean of precision and recall, balancing the trade-off between the two metrics. It is calculated as:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. EXPERIMENTATION

5.1. Dataset Description

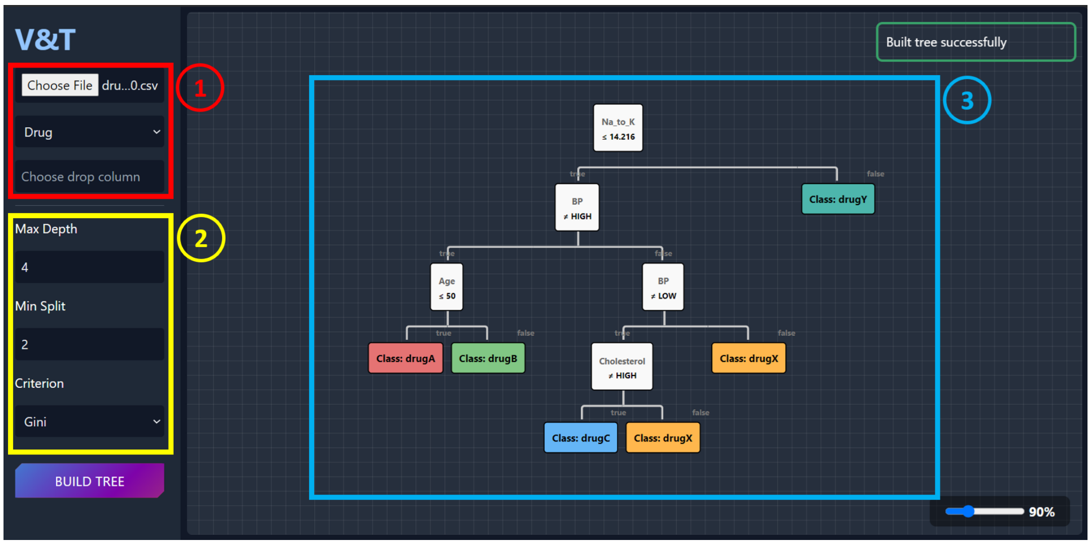
The primary dataset used is the Drug Classification dataset, which represents a common example of tabular data for decision tree learning. It contains six features and a target variable representing prescribed drug types. The features are as follows:

- **Age:** The age of the patient.
- **Sex:** The gender of the patient (Male/Female).
- **BP:** The blood pressure level of the patient (High, Normal, Low).
- **Cholesterol:** The cholesterol level of the patient (High, Normal).
- **Na_to_K:** The sodium-to-potassium ratio in the patient's blood.
- **Drug:** The type of drug prescribed (DrugA, DrugB, DrugC, DrugX, DrugY). **This is the target variable.**

This dataset serves as a representative example for understanding how decision trees operate on tabular data.

5.2. Introduction to the Decision Tree Visualization Website

To facilitate the process of building and visualizing decision trees, we developed an interactive web application. This application allows users to upload datasets, customize parameters like tree depth and splitting criteria (Gini or Entropy), and visualize the results in a clear decision tree format. The interface is divided into three main sections, as described below:



Drug

Max Depth
4

Min Split
2

Criterion
Gini

Built tree successfully

Figure 5.1: Overview of the Web Interface for Decision Tree Visualization

https://hoangvu.id.vn/decision-tree

- **(1) Data Upload and Target Column Selection Area:** Users can upload a dataset in CSV format using the *Choose File* button. Once the dataset is uploaded, the list of columns will appear in the "Choose drop column" dropdown menu, allowing users to select the target column (in this example, the **Drug** column).

- **(2) Parameter Configuration Area:** Users can customize parameters for building the decision tree, including:

  - **Max Depth:** The maximum depth of the decision tree.
  - **Min Split:** The minimum number of samples required to split a node.
  - **Criterion:** The splitting criterion, which can be either Gini or Entropy.

  Once these parameters are configured, users can click the *Build Tree* button to construct the decision tree.

- **(3) Result Display Area:** After successfully building the tree, the result is displayed in this area. Each node represents a splitting condition (or a classification result at the leaf), with *True/False* labels indicating the left and right branches.

6. CONCLUSION AND FUTURE WORK

6.1. Summary of Findings

Decision tree models have long been a cornerstone of machine learning for classification and regression tasks, particularly in structured tabular datasets. Their interpretability and ability to generate human-readable results make them invaluable for extracting insights in domains such as finance, healthcare, and marketing. Classic algorithms, such as ID3 and C4.5, introduced by Quinlan [1], laid the foundation for decision trees by using information gain and entropy to create split rules. Despite their success, these early models were prone to overfitting, especially when applied to noisy or imbalanced datasets.

To mitigate overfitting, advancements like pruning and ensemble methods have significantly improved decision tree performance. Random Forests [2], for example, aggregate predictions from multiple trees through bagging to reduce variance and increase robustness. Similarly, boosting methods such as AdaBoost [3] and Gradient Boosting Machines (GBM) [4] enhance decision tree accuracy by focusing on challenging examples and optimizing sequential learning. These techniques demonstrate the versatility of decision trees when combined with ensemble approaches.

Even in their standalone form, decision trees excel in scenarios where interpretability is critical. For example, in finance, they assist in credit scoring and fraud detection by providing transparent decision-making processes [5]. In healthcare, they are employed for disease prediction and treatment recommendations, offering clear insights into contributing factors. Moreover, decision trees are valuable tools for exploratory data analysis, as they help identify feature importance and relationships, such as predicting customer behavior in retail or analyzing market trends.

Recent advancements have addressed the scalability of decision trees to handle large-scale datasets, leveraging techniques such as parallelized construction to reduce training times while maintaining interpretability [6]. Additionally, innovative applications have emerged, such as using decision trees to optimize resource allocation in cloud computing environments. For instance, they predict resource usage patterns and improve virtual machine allocation, reducing issues like resource contention [7].

Despite their strengths, decision trees face challenges, such as handling missing data, encoding categorical features, and tuning hyperparameters. Techniques like imputation and feature scaling have been proposed to address these limitations, further improving the utility of decision trees across various domains.

6.2. Future Directions

In the future, decision tree models could be enhanced for better scalability and computational efficiency, leveraging parallel and distributed algorithms, and hardware acceleration like GPUs or TPUs. This would improve training speed and performance on large datasets without sacrificing interpretability.

Handling missing and categorical data will remain a challenge, but advanced imputation techniques (e.g., deep learning-based) and novel encoding methods can improve model accuracy. Automated hyperparameter tuning through methods like Bayesian optimization or reinforcement learning could reduce manual intervention and improve performance.

Hybrid models combining decision trees with deep learning could provide both interpretability and the ability to process complex, unstructured data like images or text, enabling explainable AI in advanced applications.

Expanding decision tree applications to areas like environmental monitoring, autonomous transportation, and personalized education could offer valuable insights in these emerging fields.

Finally, improving the explainability of ensemble methods like Random Forests and Gradient Boosting will be key, with future research focusing on better visualization tools to help users understand how individual trees contribute to predictions, especially in high-stakes fields like healthcare and finance.

7. REFERENCES

1. Quinlan, J. R. (1986). ”Induction of decision trees.” *Machine Learning*, 1(1), 81-106.

2. Breiman, L., et al. (2001). Random forests. *Machine learning*, 45(1), 5-32.

3. Freund, Y., Schapire, R. E. (1996). ”Experiments with a new boosting algorithm.” In *ICML*.

4. Friedman, J. H. (2001). *Greedy function approximation: A gradient boosting machine.* Annals of statistics, 1189-1232.

5. Chen, J., & Song, J. (2017). *Decision tree based credit scoring and fraud detection in finance.* Journal of Financial Engineering, 4(2), 123-135.

6. Lai, H. C., & Huang, K. Y. (2018). *Parallelizing decision tree construction for big data analysis.* Journal of Computational Science, 9(4), 275-286.

7. Zhou, Z. H., et al. (2010). *Machine learning in resource allocation: Decision trees for predicting cloud resource usage.* IEEE Transactions on Cloud Computing, 4(1), 18-30.