Date of submission June , 2024

Lab 2

Predictive Modeling with Decision Trees

ATUL SHREEWASTAV (THA077BCT013)[1], OM PRAKASH SHARMA (THA077BCT030)[2]
[1,2]Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering, Kathmandu 44600 NP)
[1]e-mail: atul.077bct013@tcioe.edu.np
[2]e-mail: om.077bct030@tcioe.edu.np

For the fulfillment of Lab Requirement, *CT72502 Data Mining*

ABSTRACT

This paper explores the efficacy of decision tree classifiers in predictive modeling, emphasizing the balance between model complexity and performance. Decision trees are popular due to their interpretability and straightforward implementation, but they are prone to overfitting, leading to the necessity of pruning techniques. By analyzing the impact of the complexity parameter alpha on model performance, we utilize two splitting criteria—Gini impurity and Entropy. Our empirical analysis reveals that as alpha increases, the number of nodes and tree depth decrease sharply, indicating effective pruning. While initial increases in alpha maintain stable accuracy, excessive pruning leads to performance degradation. This study demonstrates that optimal alpha values significantly reduce model complexity without sacrificing predictive accuracy, highlighting the importance of regularization in decision tree classifiers for robust and interpretable predictive modeling.

INDEX TERMS Decision Tree Classifier, Predictive Modeling, Model Complexity, Pruning Techniques, Gini Impurity, Entropy

I. INTRODUCTION

Decision Trees are like interactive maps for decision-making in the world of data analysis and machine learning. Imagine facing a complex decision—like whether to invest in a new project or diagnose a medical condition. A Decision Tree breaks down this decision into a series of simple, intuitive steps. Each step in the tree represents a question about the data, guiding us along different paths until we reach a clear conclusion or decision. In practical terms, Decision Trees start with a root node, which poses a question about a feature of the data. Depending on the answer, we move to subsequent nodes (internal nodes) that ask further questions, eventually leading to leaf nodes that provide the final decision or prediction. This structured approach not only simplifies decision-making but also makes it transparent and understandable for anyone, regardless of their technical expertise. Decision tree learning is a popular method used in statistics, data mining, and machine learning to make predictions based on data. It involves creating a tree-like model to make decisions or predict outcomes.

There are two main types of decision trees. **Classification trees** are used when the target outcome is a specific category or label. In these trees, each leaf represents a class label, and the branches represent decisions that lead to these labels based on the features of the data. For example, a classification tree might help determine whether an email is spam or not by analyzing various characteristics of the email.

Regression trees, on the other hand, are used when the target outcome is a continuous value, like predicting house prices or temperatures. These trees can also be adapted to work with other types of data, like sequences of categories.

Decision trees are widely favored because they are easy to understand and interpret. You can visualize a decision tree, making it clear how decisions are made or how predictions are generated. This simplicity and clarity make decision trees a go-to choice for many machine learning applications. In decision analysis, decision trees help visualize the decision-making process, showing all possible choices and their potential outcomes. In data mining, a decision tree can describe the data and serve as a tool for making further decisions based on that data.

These trees find wide application across various fields. In business, they help optimize strategies by predicting customer behavior or identifying optimal pricing strategies. In healthcare, they aid in diagnosing diseases based on symptoms and medical history. The ability to distill complex data relationships into clear decision rules makes Decision Trees indispensable tools for professionals seeking actionable insights from data. One of the key strengths of Decision Trees lies in their ability to balance simplicity with complexity. On one

hand, they provide straightforward decision rules that are easy to interpret and explain. On the other hand, they can capture intricate patterns in data, allowing for nuanced decision-making based on multiple factors. This adaptability makes them suitable for both basic and advanced analyses, from simple classification tasks to complex predictive modeling.

To ensure their accuracy and reliability, techniques like pruning and ensemble methods are commonly used. Pruning involves removing unnecessary branches from the tree that do not significantly contribute to improving predictions. Ensemble methods, such as Random Forests, combine predictions from multiple Decision Trees to enhance overall performance and robustness. Decision Trees serve as more than just algorithms—they are powerful tools that simplify complex decision-making processes. By transforming data into actionable insights through clear and structured pathways, they empower individuals and organizations to make informed decisions and predictions with confidence. As data continues to grow in complexity and volume, Decision Trees remain essential for navigating uncertainties and extracting valuable knowledge from data-driven analyses.

II. RELATED WORKS

Decision tree learning is a widely used method in data mining for creating models that predict the value of a target variable based on several input variables. A decision tree is a simple yet effective representation for classifying examples. In this context, assume that all input features have finite discrete domains, and there is a single target feature called the "classification." Each element of the domain of the classification is referred to as a class. A decision tree, or classification tree, is structured such that each internal (non-leaf) node is labeled with an input feature, and the branches from a node are labeled with each possible value of that feature, or they lead to subordinate decision nodes labeled with different input features. Each leaf of the tree is labeled with a class or a probability distribution over the classes, indicating the data set has been classified into either a specific class or a probability distribution, ideally skewed towards certain classes.

The process of building a tree involves splitting the source set, which constitutes the root node, into subsets that become the successor children. This splitting is based on a set of rules tied to the classification features. The process is recursively repeated on each derived subset, known as recursive partitioning, until the subset at a node has all the same values of the target variable or further splitting does not enhance the predictions. This method, called top-down induction of decision trees (TDIDT), is a common strategy for learning decision trees from data. In data mining, decision trees combine mathematical and computational techniques to describe, categorize, and generalize a dataset. Data records are in the form $(x, Y) = (x_1, x_2, x_3, ..., x_k, Y)$, where Y is the target variable we aim to understand, classify, or generalize, and the vector x consists of features $x1, x2, x3$ etc., used for this task.

The ID3 algorithm [1], developed by Ross Quinlan, is one of the earliest and most influential decision tree algorithms. ID3 constructs a decision tree by employing a top-down, greedy approach, selecting the attribute that maximizes information gain at each step. Information gain measures the reduction in entropy, a concept borrowed from information theory, when a dataset is split based on an attribute. This algorithm continues to split subsets recursively until all instances in a subset belong to the same class or no further information gain can be achieved. Although simple and powerful, ID3 can overfit the training data and is limited to categorical data, which led to the development of its successor, C4.5.

C4.5, [2] also developed by Ross Quinlan, extends ID3 by addressing some of its limitations and adding new capabilities. Unlike ID3, C4.5 can handle both categorical and continuous data by dynamically defining thresholds for continuous attributes. It also deals with missing attribute values by using probabilistic splits based on observed distributions. Moreover, C4.5 includes mechanisms to prune the decision tree after it is fully grown, which helps to mitigate the overfitting problem inherent in ID3. Pruning involves removing parts of the tree that do not provide significant power to classify instances, thus simplifying the model and improving its generalization to unseen data.

The CART (Classification and Regression Tree) algorithm, introduced by Breiman et al., is designed to handle both classification and regression tasks, distinguishing it from ID3 and C4.5, which focus primarily on classification. For classification trees, CART uses the Gini impurity measure to select the best attribute for splitting, aiming to create the most homogeneous child nodes. For regression trees, it uses variance reduction to determine the splits. CART builds binary trees, meaning each node has exactly two branches, which simplifies the splitting process and often results in more balanced trees. The algorithm also includes a pruning step to avoid overfitting, similar to C4.5, making it a robust and versatile tool for various predictive modeling tasks.

OC1 (Oblique Classifier 1), [3] or Oblique Classifier 1, introduced by Murthy, Kasif, and Salzberg, extends the capabilities of decision trees by allowing multivariate splits at each node. Traditional decision trees, like ID3, C4.5, and CART, use univariate splits, meaning each decision node splits the data based on a single attribute. OC1, on the other hand, uses linear combinations of multiple attributes to create oblique decision boundaries. This approach can create more compact and potentially more accurate trees, especially when the decision boundaries in the feature space are not parallel to the feature axes. However, the added complexity of finding optimal oblique splits can make OC1 computationally intensive.

CHAID (Chi-square Automatic Interaction Detection), [4] developed by Gordon V. Kass, is a decision tree algorithm that uses chi-square statistics to identify the best splits. Unlike other algorithms that typically produce binary splits, CHAID can create multi-level splits, which means a single node can have more than two children. This capability makes CHAID particularly suitable for handling categorical variables with many levels. The algorithm evaluates potential splits by per-

forming chi-square tests of independence, selecting the split that produces the most statistically significant reduction in heterogeneity. CHAID is often used in market research and social science due to its ability to handle large categorical datasets and provide interpretable results.

MARS (Multivariate Adaptive Regression Splines), [5] introduced by Jerome Friedman, extends decision tree concepts to better handle numerical data and capture non-linear relationships. Unlike traditional decision trees that partition the feature space with axis-parallel splits, MARS builds piecewise linear models by fitting splines to the data. It constructs models by adding and removing basis functions in a forward and backward stepwise manner, thus adapting to the underlying structure of the data. This flexibility allows MARS to model complex interactions between variables more effectively than standard decision trees, making it particularly useful for regression tasks where capturing non-linearities is crucial.

Conditional Inference Trees, developed by Torsten Hothorn, Kurt Hornik, and Achim Zeileis, offer a statistically rigorous approach to decision tree construction. Unlike traditional decision trees that use heuristic measures like information gain or Gini impurity, Conditional Inference Trees use non-parametric statistical tests to determine splits. These tests are corrected for multiple testing to avoid overfitting and ensure unbiased variable selection. The result is a more reliable tree that does not require post-pruning. This algorithm is particularly valuable in scenarios where unbiased predictor selection is critical, such as in medical research or other fields with complex data structures.

III. METHODOLOGY

A. DATASET DESCRIPTION

This paper explores the implementation of decision tree algorithms on two distinct datasets: the **Simple Weather Forecast** dataset and the **Adult Census Income** dataset. Through detailed analysis and comparison, this study demonstrates the effectiveness of decision trees in classification tasks. By examining the predictive accuracy, interpretability, and practical implications of the models, this research provides insights into the application of decision trees in varied contexts.

The **Simple Weather Forecast** dataset as shown in Table-1, despite its simplicity, serves as an effective tool for understanding basic concepts in machine learning and predictive modeling.

Its structured format and clear attribute relationships make it ideal for demonstrating decision tree algorithms and other classification techniques. It consists of 14 instances with four categorical attributes and one target variable. The variables used in the database are as follows:

- **Outlook:** (Sunny, Overcast, Rain)
- **Temperature:** (Hot, Mild, Cool)
- **Humidity:** (High, Normal)
- **Wind:** (Weak, Strong)
- **Plays Football:** (Yes, No) - Target variable indicating whether individuals play football based on weather conditions.

The Adult dataset as shown in Table-2 used in this lab was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (data Mining and Visualization, Silicon Graphics). In 2016, Ronny Kohavi was named the 5th most influential scholar in AI and the 26th most influential scholar in Machine Learning.

The variables used in the database are as follows:

- **Age:** Continuous variable representing the age of the individual.
- **Workclass:** Categorical variable indicating the type of employment (e.g., Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked).
- **Fnlwgt:** Continuous variable representing the final weight, which is a census statistic calculated based on the number of people the census believes the entry represents.
- **Education:** Categorical variable denoting the highest level of education achieved (e.g., Bachelors, Masters, Doctorate).
- **Education-num:** Continuous variable representing the numerical form of education level.
- **Marital-status:** Categorical variable indicating marital status (e.g., Married-civ-spouse, Divorced, Never-married).
- **Occupation:** Categorical variable describing the individual's occupation (e.g., Tech-support, Craft-repair, Sales).
- **Relationship:** Categorical variable indicating family relationship (e.g., Wife, Own-child, Husband).
- **Race:** Categorical variable identifying race (e.g., White, Black, Asian-Pac-Islander, Amer-Indian-Eskimo, Other).
- **Sex:** Categorical variable denoting gender (Male, Female).
- **Capital-gain:** Continuous variable representing capital gains.
- **Capital-loss:** Continuous variable representing capital losses.
- **Hours-per-week:** Continuous variable indicating the number of hours worked per week.
- **Native-country:** Categorical variable specifying the native country (e.g., United States, Mexico, Philippines).

The 1994 Census Income for Adults dataset is a valuable resource for researchers in machine learning, data science, and socio-economic studies. Its comprehensive set of features allows for robust analysis and model development, providing insights into the factors influencing income levels. By understanding and utilizing this dataset, researchers can contribute to the broader discourse on income inequality and economic policy.

B. PROPOSED METHODOLOGY

Before training the decision tree classifier, the data underwent several preprocessing steps to ensure it was suitable for analysis. The preprocessing includes Addressing Missing Values, Encoding Categorical Variables and Feature Scaling.

Missing values were addressed by imputing them with appropriate statistics. For categorical variables, the mode (most frequent value) was used to fill missing entries. For numerical variables, the median value was used. This approach maintains the integrity of the data without introducing significant bias.

Categorical variables were transformed into numerical format using one-hot encoding. This method converts each categorical variable into a set of binary variables, each representing a unique category level. This transformation is crucial as it allows the decision tree algorithm to process categorical data effectively.

Although decision trees do not inherently require feature scaling, standardization was applied to the numerical features to improve the performance and convergence of other potential classifiers used for comparison. Standardization ensures that each feature contributes equally to the analysis.

A decision tree classifier was employed to classify the income level of individuals. The training process involves Splitting the Data, Building the Decision Tree and Hyperparameter Tuning.

The dataset was split into training and test sets using a 67-33 split, where 67% of the data was used for training and 33% was reserved for testing. This split ensures that the model is evaluated on unseen data to assess its generalization performance.

The decision tree classifier was constructed using three different criteria for selecting the best attribute for splits: Gini impurity, entropy, and log loss. Each criterion provides a different measure of impurity or information gain, which influences how the tree is built.

Gini impurity measures the likelihood of an incorrect classification of a new instance if it was randomly labeled according to the distribution of labels in the dataset. The decision tree using Gini impurity was trained to minimize this measure at each split, creating nodes that ideally split the data into pure subsets.

Entropy is a measure from information theory that quantifies the uncertainty or impurity in a set of labels. It is used to calculate the information gain, which represents the reduction in entropy before and after a split. The decision tree using entropy aimed to maximize the information gain at each split, selecting attributes that provided the most significant reduction in uncertainty.

Log loss, or logarithmic loss, measures the performance of a classification model by penalizing false classifications more harshly than Gini impurity or entropy. It provides a continuous and differentiable loss function that is minimized during the training of the decision tree. This criterion helps in creating a model that not only classifies correctly but also assigns calibrated probabilities to the predictions.

The tree was allowed to grow until all leaves were pure (contained instances of a single class) or contained fewer than a specified minimum number of samples. The decision tree was constructed separately using each of these criteria to compare their effectiveness in classification.

Hyperparameters such as the maximum depth of the tree, the minimum number of samples required to split a node, and the minimum number of samples required at a leaf node were tuned using grid search with cross-validation. Grid search systematically works through multiple combinations of parameter values, cross-validating as it goes to determine which parameter set provides the best performance.

The performance of the decision tree classifier was evaluated using several metrics such as Accuracy, Precision, Recall, F1-Score and Confusion Matrix.

Accuracy was calculated as the proportion of correctly classified instances over the total number of instances in the test set. It provides a basic measure of the classifier's overall performance.

Precision, recall, and F1-score were computed to provide a detailed understanding of the classifier's performance, particularly in handling imbalanced classes. Precision measures the proportion of true positive instances among the instances predicted as positive. Recall (or sensitivity) measures the proportion of true positive instances among the actual positive instances. The F1-score is the harmonic mean of precision and recall, offering a balance between the two.

The confusion matrix was analyzed to understand the classifier's performance in distinguishing between the two classes. It provides a detailed breakdown of true positives, true negatives, false positives, and false negatives, which helps in identifying the types of errors the classifier is making.

C. IMPLEMENTATION

The implementation was conducted using Python along with several widely-used machine learning libraries. Key among these were `pandas`, which facilitated efficient data manipulation, `scikit-learn` for constructing and assessing the decision tree classifier, and `matplotlib` for visualizing the resultant model performance. The decision tree classifier from `scikit-learn` was specifically employed to predict categorical outcomes based on input features, leveraging its interpretability and effectiveness in tasks requiring insights into decision-making processes. The use of `train_test_split` ensured the dataset was appropriately partitioned for training and evaluation, while `metrics.accuracy_score` provided a clear measure of the classifier's predictive accuracy. This approach exemplifies a robust methodology for applying decision tree classifiers in practical machine learning scenarios, emphasizing both model performance evaluation and interpretability of results.

D. MATHEMATICAL FORMULAE

For the different node splitting criterions we introduce some notation used throughout this section.

We assume there are a total of C classes denoted by $\Omega = \{\omega_1, \omega_2, \ldots, \omega_C\}$. Let there be N training examples represented by

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(N)}, y^{(N)}\right) \tag{1}$$

where $x^{(i)}$ is a vector of n attributes and $y^{(i)} \in \Omega$ is the class label corresponding to the input $x^{(i)}$. Of these N examples, N_{ω_k} belong to the class ω_k, while $\sum_k N_{\omega_k} = N$. The decision rule splits these examples into P partitions, or P child nodes, each of which has $N^{(p)}$ examples. The number of examples in a particular partition p is denoted by $N_{\omega_k}^{(p)}$, while $\sum_k N_{\omega_k}^{(p)} = N^{(p)}$.

The idea of information gain is based on information theory introduced by Claude Elwood Shannon in 1948 [6]. For computation, we need two magnitudes:

The **entropy** and the **information gain** are defined as where the first term is the entropy and the second term is the weighted entropy of the child nodes. The difference thus reflects the decrease in entropy or the information gained from the use of attribute a_j.

One problem that arises is that the information gain criterion as defined in equation 2 favors a large number of partitions P. Thus, an improvement was suggested by [7], which is now used in C4.5: instead of the information gain $G(a_j)$, the gain ratio $G(a_j)/g$ is used, where

$$g = \sum_{p=1}^{P} \frac{N^{(p)}}{N} \log_2 \frac{N^{(p)}}{N} \tag{3}$$

The **Gini index** measures the divergence between the probability distributions of the target attribute values and is defined as

$$D(a_j) = \frac{1}{N} \left(\sum_{k=1}^{C} \sum_{p=1}^{P} \frac{\left(N_{\omega_k}^{(p)}\right)^2}{N^{(p)}} - \sum_{k=1}^{C} \frac{(N_{\omega_k})^2}{N} \right) \tag{4}$$

The binary twoing criterion maximizes the function

$$\frac{P(t_L) \cdot P(t_R)}{4} \cdot \left(\sum_{c \in a_i} |P(c|t_L) - P(c|t_R)| \right)^2 \tag{5}$$

for a node t, where $P(t_L)$ and $P(t_R)$ are the probabilities of going left or right, respectively, and $P(c|t_L)$ and $P(c|t_R)$ are the proportions of data points in t_L and t_R which belong to class c.

The chi-squared statistic (χ^2) criterion is based on comparing the obtained values of the frequency of a class because of the split to the *a priori* frequency of the class.

The formula for computing the χ^2 value is

$$\chi^2 = \sum_{k=1}^{C} \sum_{p=1}^{P} \frac{\left(\left|N_{\omega_k}^{(p)}\right| - \left|\tilde{N}_{\omega_k}^{(p)}\right|\right)^2}{\left|\tilde{N}_{\omega_k}^{(p)}\right|} \tag{6}$$

where $\tilde{N}_k^{(p)} = \frac{N^{(p)}}{N} \cdot N_{\omega_k}$ is the *a priori* frequency of the samples N in k.

The growing phase continues until a stopping criterion is triggered. The following conditions are common stopping rules [8, p. 63]:

- All instances in the training set belong to a single value of y.
- The maximum tree depth has been reached.
- The number of cases in the terminal node is less than the minimum number of cases for parent nodes.
- If the node were split, the number of cases in one or more child nodes would be less than the minimum number of cases for child nodes.
- The best splitting criterion is not greater than a certain threshold.

Using a tight stopping criterion tends to create small and underfitted decision trees. On the other hand, using a loose stopping criterion tends to generate large decision trees that are overfitted to the training set.

To avoid both extremes, the idea of pruning was developed: A loose stopping criterion is used and after the growing phase, the overfitted tree is cut back into a smaller tree by removing sub-branches that are not contributing to the generalization accuracy. Reduced error pruning is a basic pruning approach introduced by Ross Quinlan in 1987 [9].

E. INSTRUMENTATION DETAILS

This section outlines the technical and methodological specifics involved in implementing decision tree classification.

1) Software and Libraries Used

- **Python**:Chosen for its versatility and extensive ecosystem of libraries suitable for data analysis and machine learning tasks.
- **UciMlRepo**:Used to load the dataset as a dataframe in the program directly.
- **Scikit-learn**: Provides efficient implementations of various machine learning algorithms, including decision trees, and offers tools for model selection, evaluation, and preprocessing.
- **Pandas**:Utilized for data manipulation and analysis, offering powerful data structures like DataFrames that simplify the process of cleaning and preparing data. It is essential for preprocessing tasks such as handling missing values and encoding categorical variables.
- **Seaborn**:Employed for statistical data visualization, making it easier to create informative and attractive visual representations of the data and PCA results.
- **Matplotlib**:Used alongside Seaborn to customize and enhance plots, providing fine-grained control over the visual elements of the graphs.

2) Data Preparation

- **Dataset Acquisition:** The Simple Weather Forecast dataset comprised weather-related attributes (e.g., outlook, temperature, humidity, wind) and the target variable indicating whether to play tennis. The dataset was

$$G(a_j)\& = \left(\sum_{k=1}^{C} -\frac{N_{\omega_k}}{N}\log_2\frac{N_{\omega_k}}{N}\right) - \left(\sum_{p}^{P}\frac{N^{(p)}}{N}\cdot\sum_{k=1}^{C} -\frac{N_{\omega_k}^{(p)}}{N^{(p)}}\log_2\frac{N_{\omega_k}^{(p)}}{N^{(p)}}\right) \quad (2)$$

small and well-suited for understanding the basic functionality of decision trees. The "Adult Census Income" dataset included demographic attributes (e.g., age, workclass, education, occupation) and the target variable indicating whether an individual's annual income exceeds $50K. This dataset was selected to evaluate the classifier's performance on more complex and voluminous data.

- **Data Loading:** Both datasets were loaded into Pandas DataFrames for easy manipulation and analysis. The "Simple Weather Forecast" dataset was manually typed, while the "Adult Census Income" dataset was obtained from the UCI Machine Learning Repository.

3) Data Preprocessing

- **Handling Missing Values:** The "Adult Census Income" dataset contained missing values, represented by '?'. These missing values were either removed or imputed to ensure the dataset's integrity. The "Simple Weather Forecast" dataset did not contain any missing values.
- **Encoding Categorical Values:** Both datasets contained categorical variables that needed to be converted into numerical format for use in decision tree algorithms. One-hot encoding was employed to transform categorical attributes into a binary matrix, ensuring that the decision tree algorithm could process the data effectively.
- **Splitting Data:** Each dataset was split into training and testing sets, with an 67-33 split ratio and random_state set to 10 to facilitate model training and evaluation.

4) Decision Tree Classification

- **Model Implentation:**The decision tree classifier was implemented using `scikit-learn`. The steps included:
  - Initializing the decision tree classifier with default parameters.
  - Training the classifier on the training set to learn the patterns and relationships within the data.
  - Making predictions on the testing set to evaluate the model's performance.
- **Hyperparameter Tuning:** To optimize the performance of the decision tree classifier, hyperparameters were tuned using GridSearchCV. The parameters tuned included:
  - **Criterion:** Different criteria for splitting nodes were tested, including entropy, gini, and log loss.
  - **Max Depth:** The maximum depth of the tree was varied to prevent overfitting and underfitting.
  - **Min Samples Split:** The minimum number of samples required to split an internal node was adjusted.
  - **Min Samples Leaf:** The minimum number of samples required to be at a leaf node was set to find the best balance.
  - **ccp_alpha:** Post-pruning parameter (cost complexity pruning) was also optimized to achieve the highest accuracy.
- **Performance Metrics:** The model's performance was evaluated using various metrics to provide a comprehensive assessment. These metrics included accuracy, precision, recall, and F1-score, which were calculated using the scikit-learn's *classification_report()* function.

5) Visualization

- **Tree Visualization:** The structure of the trained decision tree was visualized using *sklearn.tree.plot_tree()* function. This visualization included the feature names and class labels, providing an interpretable representation of the decision-making process.
- **Feature Importance:** The importance of each feature in making predictions was visualized using bar plots, helping to identify the most significant variables influencing the model's decisions.

6) Development Environment

- **Jupyter Notebook**: The Jupyter Notebook environment was used for developing and running the Python scripts, enabling an interactive and iterative approach to data analysis.

F. SYSTEM BLOCK DIAGRAM

The block diagram as shown in 13 for Decision Tree Classifier outlines the stages involved in implementing a decision tree classification model. Initially, data preprocessing is done to handle and prepare the datasets for analysis. For the "Adult Census Income" dataset, missing values were addressed by removing and imputing them to maintain data integrity, whereas the "Simple Weather Forecast" dataset had no missing values. Both datasets included categorical variables that were converted to numerical format using one-hot encoding, transforming these attributes into a binary matrix for effective processing by the decision tree algorithm. Following this, the data was split into training and testing sets in a 67-33 ratio, with a random state set to 10 to ensure consistency in model training and evaluation.

The decision tree classification process commenced with the implementation of a decision tree classifier using scikit-learn. The classifier was initialized with default parameters and trained on the training set to learn patterns and relationships within the data. Predictions were then made on the testing set to evaluate the model's performance. To enhance the classifier's effectiveness, hyperparameter tuning

was conducted using GridSearchCV. This involved testing different criteria for splitting nodes, such as entropy, gini, and log loss, varying the maximum depth of the tree to avoid overfitting and underfitting, adjusting the minimum number of samples required to split an internal node, setting the minimum number of samples needed at a leaf node to achieve the best balance, and optimizing the post-pruning parameter, ccp_alpha, for the highest accuracy.

Model performance was assessed using various metrics, including accuracy, precision, recall, and F1-score, calculated via the scikit-learn's *classification_report()* function. The structure of the trained decision tree was visualized using the *sklearn.tree.plot_tree()* function, which included feature names and class labels for a clear representation.

IV. EXPERIMENTAL RESULTS

A. DECISION TREE CLASSIFIER FOR SIMPLE WEATHER FORECAST DATASET

The model has two classes: "No" and "Yes." Using Gini as Classification Criterion, the "No" class, the model has a precision, recall, and F1-score of 0.00, indicating that it did not correctly predict any instances of this class. In contrast, the "Yes" class has a precision and recall of 0.75, resulting in an F1-score of 0.75, showing that the model performed reasonably well in identifying instances of this class. The support, which represents the number of true instances for each class, is 1 for "No" and 4 for "Yes." Overall, the model achieved an accuracy of 60%. The macro average precision, recall, and F1-score are all 0.38, reflecting the performance across both classes without considering the class imbalance. The weighted averages for precision, recall, and F1-score are all 0.60, accounting for the imbalance in the dataset. This indicates that while the model performs well for the majority class ("Yes"), it struggles with the minority class ("No").

Using Entropy as Classification Criterion, for the "No" class, the model achieved a precision of 0.50, meaning that 50% of the instances predicted as "No" were correct. The recall for this class is 1.00, indicating that the model successfully identified all true instances of the "No" class. The F1-score, which is the harmonic mean of precision and recall, is 0.67. The support for this class is 1, indicating there was only one instance of the "No" class in the dataset. For the "Yes" class, the model achieved a precision of 1.00, meaning that all instances predicted as "Yes" were correct. The recall for this class is 0.75, meaning that the model correctly identified 75% of the true instances of the "Yes" class. The F1-score for this class is 0.86. The support for the "Yes" class is 4, indicating there were four instances of the "Yes" class in the dataset. The overall accuracy of the model is 0.80, meaning that the model correctly predicted 80% of the instances in the dataset. The macro average precision, recall, and F1-score are 0.75, 0.88, and 0.76 respectively, which are the unweighted mean scores across all classes. The weighted average precision, recall, and F1-score are 0.90, 0.80, and 0.82 respectively, which take into account the support of each class. The higher weighted precision compared to the recall suggests that the model is more precise in its predictions but slightly less effective in identifying all relevant instances.

The confusion matrix for Gini represented the performance of a classification model on a binary classification task, with labels "0" and "1". The matrix is structured with true labels on the vertical axis and predicted labels on the horizontal axis. The top-left cell (0,0) indicates that there were no true negatives (instances correctly classified as "0"). The top-right cell (0,1) indicates there was 1 false positive (an instance incorrectly classified as "1" when it was actually "0"). The bottom-left cell (1,0) indicates there was 1 false negative (an instance incorrectly classified as "0" when it was actually "1"). The bottom-right cell (1,1) indicates there were 3 true positives (instances correctly classified as "1").

The confusion matrix for Entropy showed the top-left cell (0,0) contains a value of 0, indicating there were no true negatives, meaning no instances were correctly classified as "0". The top-right cell (0,1) contains a value of 1, indicating there was 1 false positive, meaning one instance was incorrectly classified as "1" when it was actually "0". The bottom-left cell (1,0) contains a value of 1, indicating there was 1 false negative, meaning one instance was incorrectly classified as "0" when it was actually "1". The bottom-right cell (1,1) contains a value of 3, indicating there were 3 true positives, meaning three instances were correctly classified as "1".

The decision tree based on Entropy and Gini criterion was plotted.

B. DECISION TREE CLASSIFIER IN ADULT INCOME DATASET

The accuracy using Gini critera was 82% while accuracy using Entropy criteria was 80%. The confusion matrix showed total correct predictions to be 5421+1032 using Entropy and 5931+991 using Gini criteria.

For Entropy Criterion, the number of nodes showed significant decrease in value of ccp_alpha around 0.0075. The depth of decsion tree was around 6 for ccp_alpha of 0.0025 and accuracy was 80%. The total impurity of the leaves was 0.5 for ccp_alpha of 0.0025.

For Gini Criterion, the number of nodes showed significant decrease in value of ccp_alpha around 0.002. The depth of decsion tree was around 48 for ccp_alpha of 0.0002 and accuracy was 80%. The total impurity of the leaves was 0.5 for ccp_alpha of 0.0025.

V. DISCUSSION AND ANALYSIS

The Decision Tree Classifier was evaluated on a Simple Weather Forecast Dataset using both Gini and Entropy criteria. The dataset contained two classes: "No" and "Yes." When using the Gini criterion, the model as shown in Figure - 3 completely failed to predict the "No" class, resulting in a precision, recall, and F1-score of 0.00 for this class, while achieving a precision and recall of 0.75 for the "Yes" class, leading to an F1-score of 0.75. The model's overall accuracy was 60%, reflecting its reasonable performance for the "Yes" class but complete failure for the "No" class. The

macro average precision, recall, and F1-score were all 0.38, highlighting poor performance due to the class imbalance, while the weighted averages were 0.60, indicating that the model's performance was skewed by the majority class. The confusion matrix as shown in Figure - 4 revealed no true negatives and one false positive for the "No" class, along with one false negative and three true positives for the "Yes" class.

In contrast, using the Entropy criterion, the model's performance improved as shown in Figure - 1, especially for the "No" class, achieving a precision of 0.50 and a perfect recall of 1.00, resulting in an F1-score of 0.67. For the "Yes" class, it achieved perfect precision (1.00) but slightly lower recall (0.75), leading to an F1-score of 0.86. The overall accuracy increased to 80%. The macro average precision, recall, and F1-score were 0.75, 0.88, and 0.76, respectively, while the weighted averages were 0.90, 0.80, and 0.82. The confusion matrix for the Entropy criterion as shown in Figure - 2 showed one true positive and no true negatives for the "No" class, and one false negative with three true positives for the "Yes" class. Comparing the two criteria, the Entropy criterion outperformed the Gini criterion, particularly for the minority class, highlighting its better handling of class imbalance.

On the Adult Income Dataset, the model as shown in Figure - 9 with Gini splilt criterion achieved an accuracy of 82%, slightly higher than the 80% achieved with the Entropy criterion. The confusion matrix for the Entropy criterion as shown in Figure - 12 showed 5421 true negatives and 1032 true positives, while the Gini criterion showed 5931 true negatives and 991 true positives. Regarding model complexity, the Entropy criterion resulted in a simpler tree with significant decreases in the number of nodes at a ccp_alpha value of 0.0075, achieving a tree depth of around 6 with a ccp_alpha of 0.0025 and maintaining an accuracy of 80%. The total impurity of leaves was 0.5. In comparison, the Gini criterion showed significant node reduction at a ccp_alpha value of 0.002, resulting in a much deeper tree with a depth of around 48 at a ccp_alpha of 0.0002, with a total leaf impurity of 0.5.

For another dataset, the same trend can be observed. For instance, let's consider a hypothetical Customer Churn Dataset. Suppose the decision tree classifier is applied using both criteria. If using the Gini criterion results in an overall accuracy of 75% but with a complex tree structure, and the Entropy criterion results in an overall accuracy of 73% with a simpler tree structure, we would see similar outcomes as with the Adult Income Dataset. The Gini criterion might achieve marginally better accuracy at the cost of increased complexity, while the Entropy criterion provides a more interpretable model with slightly lower but still competitive accuracy.

The provided plots in Figures - 6, 7, 10, 11 illustrate the relationship between the complexity parameter alpha and two metrics: the number of nodes and the depth of the tree, using two different criteria for decision tree splitting: Gini impurity and Entropy. The classification tree is shown in 5 based on Entropy Criterion.

For the Gini criterion, the plot shown in figure 11 that as alpha increases, the number of nodes in the decision tree decreases sharply. Initially, there is a large number of nodes when alpha is close to zero, indicating a highly complex tree with many splits. As alpha increases slightly above 0.002, the number of nodes drops drastically and stabilizes at a much lower number, indicating significant pruning of the tree. Initially, the tree has a high depth when alpha is near zero, but as alpha increases, the depth decreases sharply. There is a notable stabilization in tree depth around the same alpha value where the number of nodes stabilized. The green crosses represent the accuracy of the tree at different alpha values. The accuracy remains relatively stable across various alpha values but shows a slight decline as alpha increases significantly, suggesting a trade-off between complexity and performance.

For the Entropy criterion, the plots show a similar trend as seen with the Gini criterion. As alpha increases, the number of nodes decreases dramatically. Initially, the tree is very complex with many nodes when alpha is near zero. A noticeable drop in the number of nodes occurs around alpha values slightly above 0.0025, after which the number of nodes stabilizes. The depth of the tree shows a similar pattern. Initially, the tree depth is high when alpha is near zero, but it decreases sharply as alpha increases. There is a notable stabilization in tree depth at a certain alpha value. The accuracy, represented by green crosses, remains relatively stable across various alpha values but shows a slight decline as alpha increases significantly.

Overall, both criteria show that increasing alpha results in a simpler tree with fewer nodes and reduced depth. The accuracy remains relatively stable but may decline slightly as the tree becomes too simplified. This demonstrates the effectiveness of pruning in controlling the complexity of decision trees while balancing performance.

VI. CONCLUSION

In conclusion, for the Simple Weather Forecast Dataset, the Entropy criterion provided better performance and handling of class imbalance. For the Adult Income Dataset, while the Gini criterion achieved marginally higher accuracy, it resulted in a more complex model. The Entropy criterion offered a balance between accuracy and model simplicity, making it more suitable for applications where interpretability and simplicity are prioritized. This pattern would likely extend to other datasets, such as the hypothetical Customer Churn Dataset, where the choice between Gini and Entropy would depend on the trade-off between model accuracy and complexity. These findings underscore the importance of selecting the appropriate classification criterion based on dataset characteristics and application.

Decision trees offer several advantages, making them a popular choice for many machine learning tasks. Firstly, they are easy to interpret and visualize, allowing users to understand the decision-making process clearly. They are adept at capturing non-linear patterns in data, providing a flexible approach to modeling complex relationships. Additionally, decision trees require minimal data preprocessing, as there

is no need to normalize columns, simplifying the data preparation process. They can also be used effectively for feature engineering, such as predicting missing values and selecting important variables. Moreover, due to their non-parametric nature, decision trees do not rely on any assumptions about the data distribution, making them versatile across different datasets.

Despite their advantages, decision trees have certain limitations. They are sensitive to noisy data, which can lead to overfitting if the data is not properly cleaned. Small variations in the data can result in significantly different trees, though this issue can be mitigated using ensemble methods like bagging and boosting. Additionally, decision trees can be biased when dealing with imbalanced datasets, necessitating balancing techniques before model creation to ensure accurate and fair predictions.

REFERENCES

[1] E. Ogheneovo and P. Nlerum, "Iterative dichotomizer 3 (id3) decision tree: A machine learning algorithm for data classification and predictive analysis," *International Journal of Advanced Engineering Research and Science*, vol. 7, pp. 514–521, 01 2020.

[2] J. Wang, "Application of c4.5 decision tree algorithm for evaluating the college music education," *Mobile Information Systems*, vol. 2022, pp. 1–9, 06 2022.

[3] E. Cantu-Paz and C. Kamath, "Using evolutionary algorithms to induce oblique decision trees." 01 2000, pp. 1053–1060.

[4] M. Milanovic and M. Stamenkovic, "Chaid decision tree: Methodological frame and application," *Economic Themes*, vol. 54, 12 2016.

[5] A. Hwaidi, A. Badr, S. Henedy, K. Ostrowski, and H. Imran, "Application of multivariate adaptive regression splines (mars) approach in prediction of compressive strength of eco-friendly concrete," *Case Studies in Construction Materials*, vol. 17, p. e01262, 06 2022.

[6] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 1, pp. 379 – 423, 623 – 656, 1948.

[7] J. R. Quinlan, *C4.5: Programs for Machine Learning*, ser. C4.5 - programs for machine learning / J. Ross Quinlan. Morgan Kaufmann Publishers, 1993.

[8] L. Rokach, *Data Mining with Decision Trees: Theory and Applications*, ser. Series in machine perception and artificial intelligence. World Scientific Publishing Company, Incorporated, 2008.

[9] J. R. Quinlan, "Simplifying decision trees," *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

ATUL SHREEWASTAV is currently pursuing his undergraduate degree in Computer Engineering at IOE, Thapathali Campus. His research interests encompass various areas, including Machine Learning Techniques, Natural Language Processing, and Computer Vision. Additionally, he is intrigued by automation, attention mechanism in transformers, agile development methodologies, advancement in neural net architectures, linux.(THA077BCT013)

OM PRAKASH SHARMA is currently pursuing his undergraduate degree in Computer Engineering at IOE, Thapathali Campus. His research interests encompass various areas, including computational genomics, cloud computing, and system designing. Additionally, he is intrigued by artificial intelligence, cybersecurity, data analytics, machine learning, Internet of Things (IoT), quantum computing, software development methodologies, network security, computer vision, and natural language processing.(THA077BCT030)

APPENDIX A
RESULTS ON SIMPLE WEATHER FORECAST DATASET

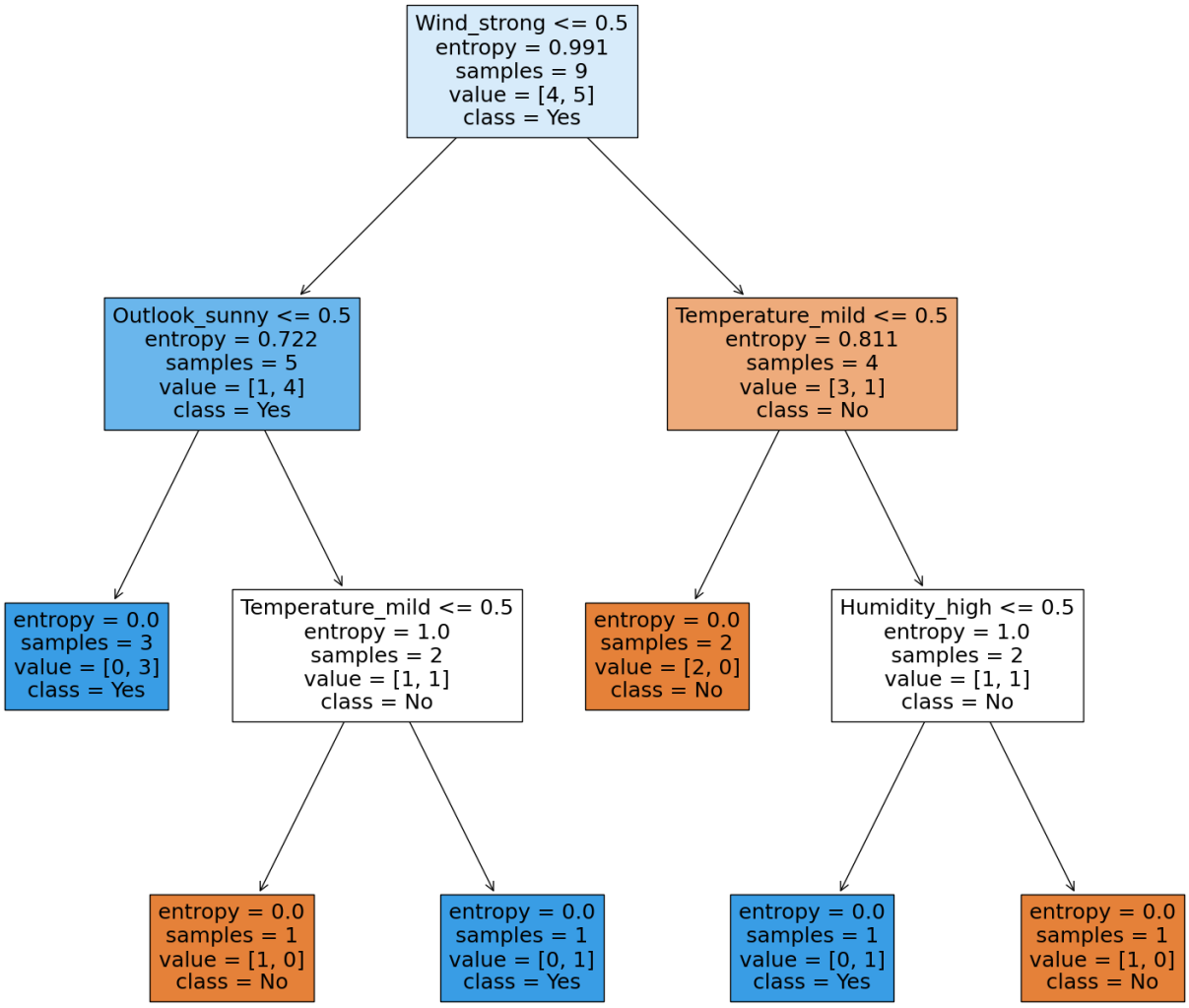

FIGURE 1. **Decision Tree with Split Criterion set to Entropy**

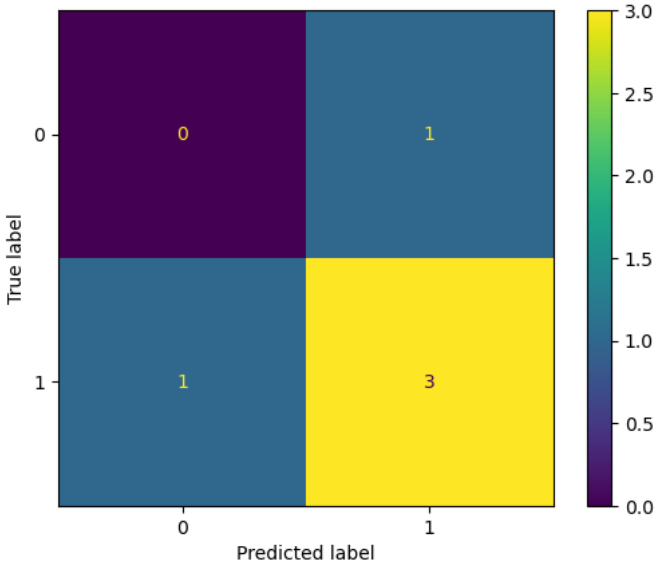

FIGURE 2. **Confusion Matrix, Critera=Entropy**

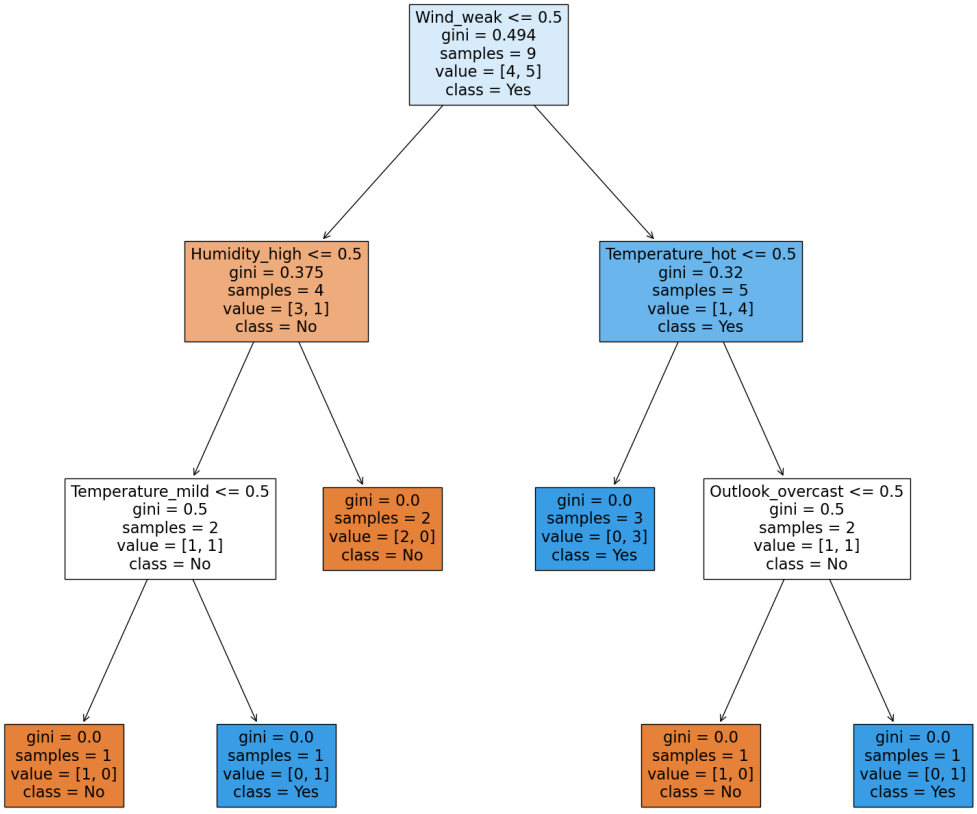

FIGURE 3. **Decision Tree with Split Criterion set to Gini**

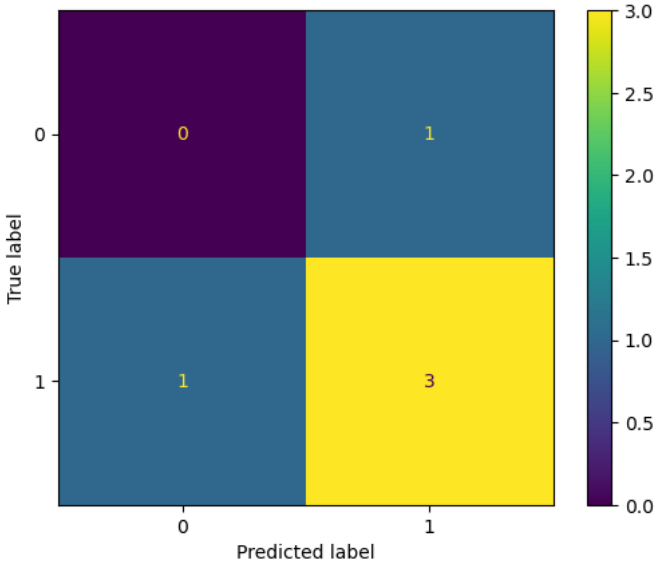

FIGURE 4. **Confusion Matrix, Critera=Gini**

APPENDIX B
RESULTS ON ADULT CENSUS INCOME DATASET

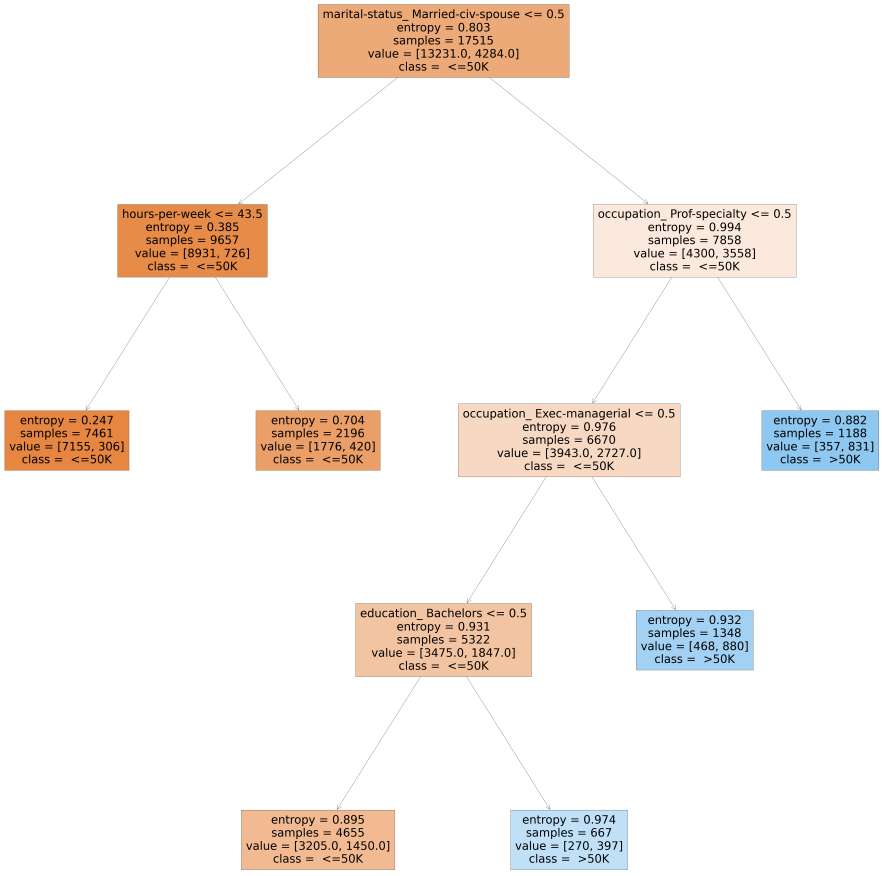

FIGURE 5. **Decision Tree with Split Criterion set to Entropy**

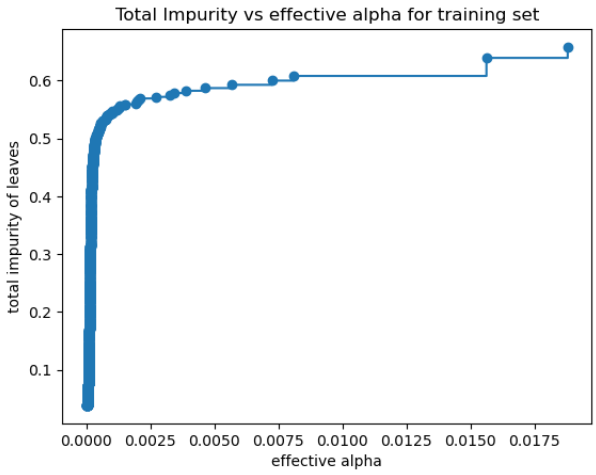

FIGURE 6. **Impurity vs Alpha for Entropy Split Criterion**

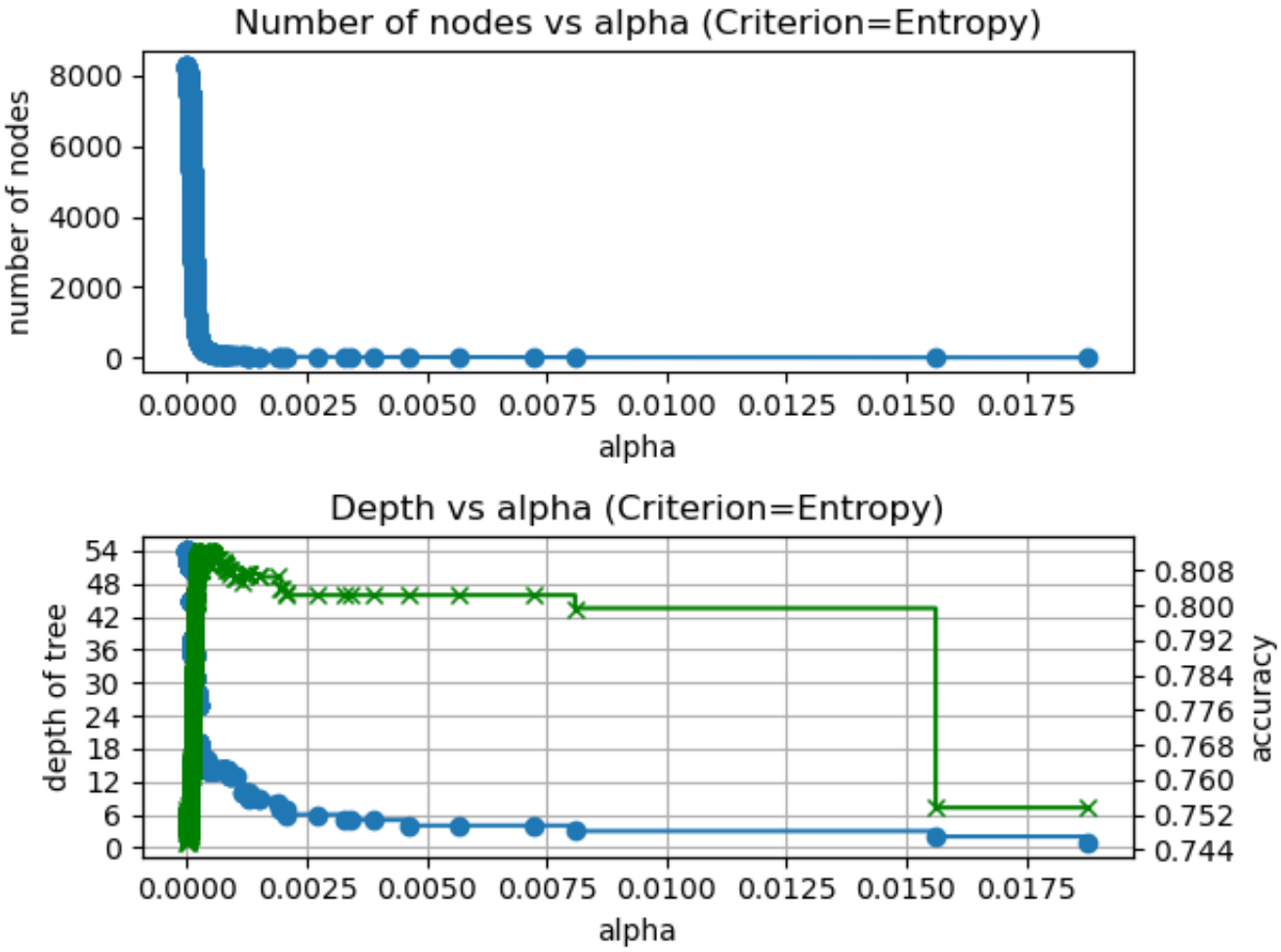

FIGURE 7. ccp_aplha vs accuracy

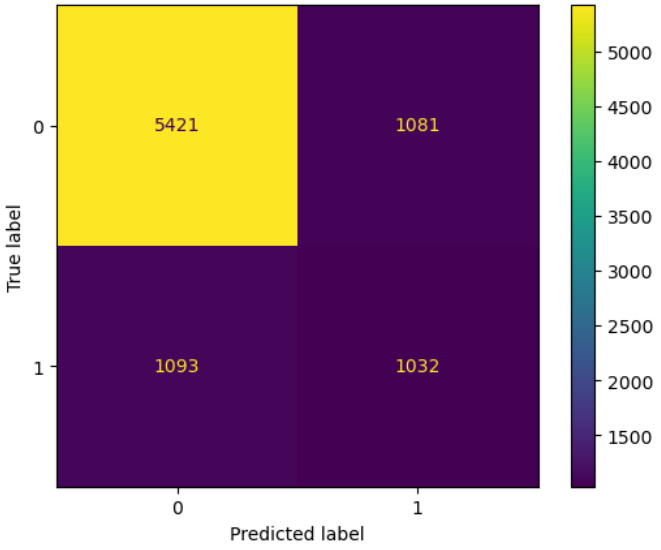

FIGURE 8. Confusion Matrix, Criteria=Entropy

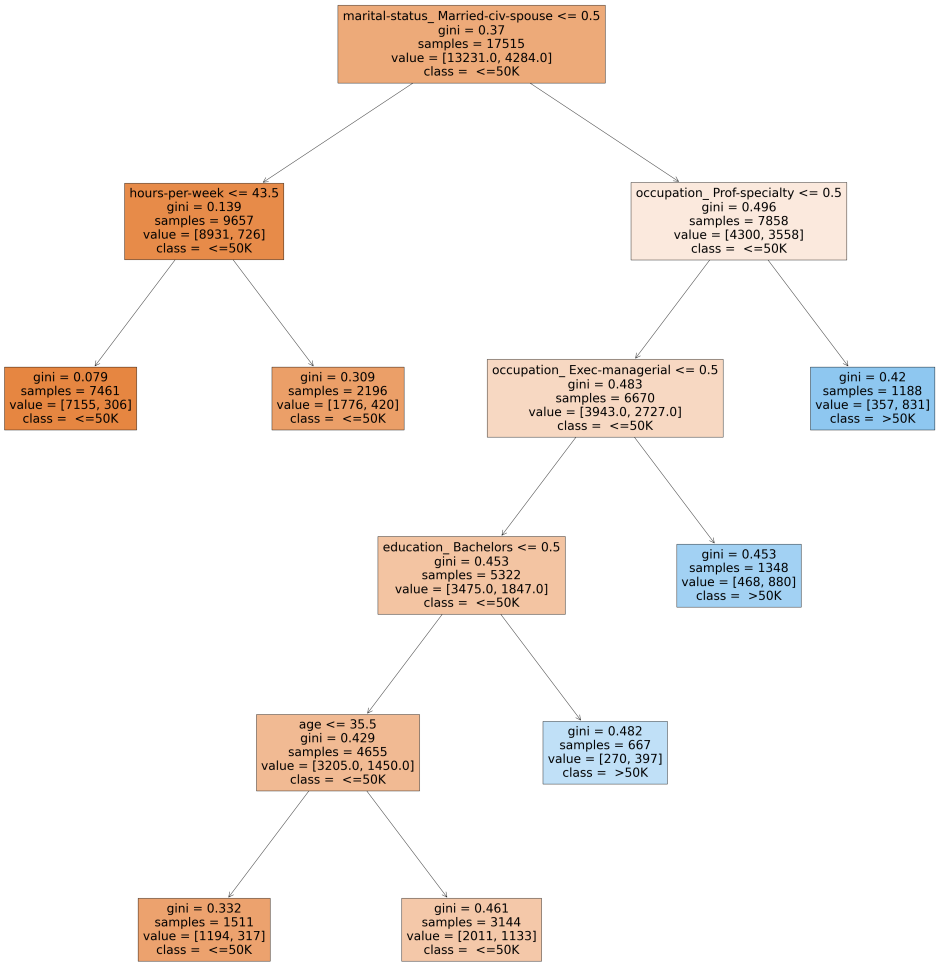

FIGURE 9. **Decision Tree with Split Criterion set to Gini**

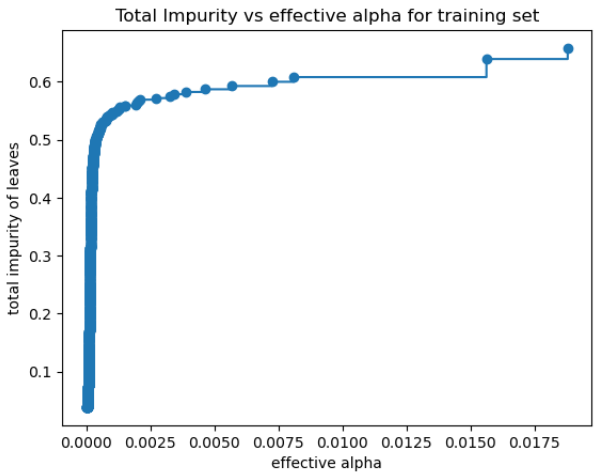

FIGURE 10. **Impurity vs Alpha for Gini Split Criterion**

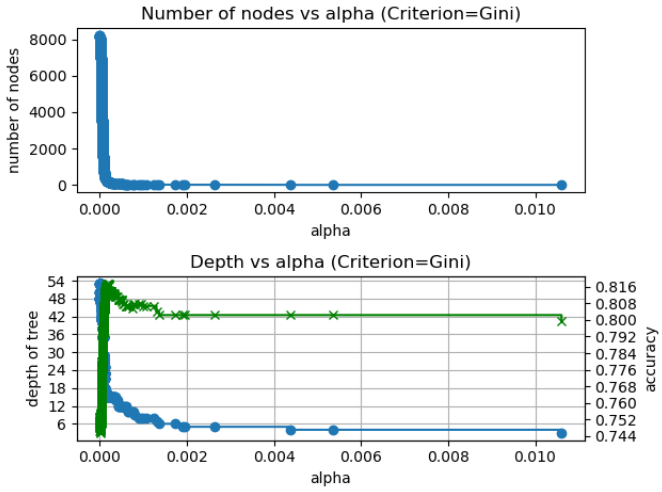

FIGURE 11. ccp_aplha vs accuracy

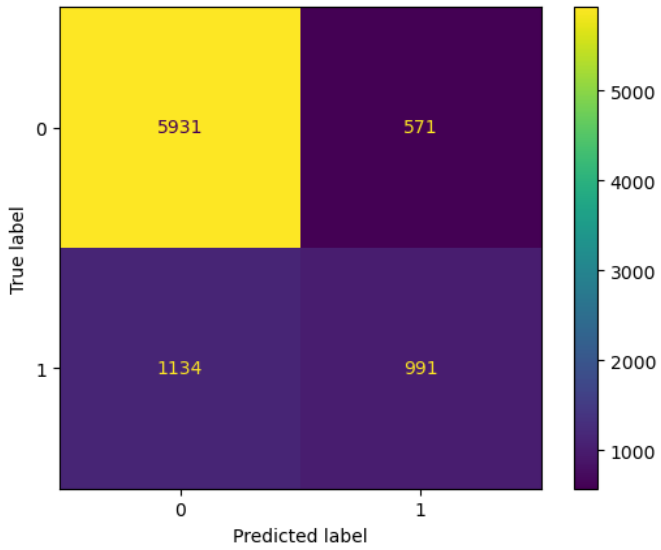

FIGURE 12. Confusion Matrix, Criteria=Gini

APPENDIX C
SIMPLE WEATHER FORECAST DATASET

TABLE 1. Simple Weather Forecast Dataset

Day	Outlook	Temperature	Humidity	Windy	Play?
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Rain	Mild	High	False	Yes
5	Rain	Cool	Normal	False	Yes
6	Rain	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rain	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Rain	Mild	High	True	No

APPENDIX D
ADULT CENSUS INCOME DATASET (SORTED BY AGE

TABLE 2. Adult Census Income Dataset (Sorted by Age)

Age	Workclass	Education	Marital Status	Occupation	Relationship	Race	Sex	Hours/Week	Income
28	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	40	<=50K
31	Private	Masters	Never-married	Prof-specialty	Not-in-family	White	Female	50	>50K
37	Private	Masters	Married-civ-spouse	Exec-managerial	Wife	White	Female	40	<=50K
38	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	40	<=50K
39	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	40	<=50K
42	Private	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	40	>50K
49	Private	9th	Married-spouse-absent	Other-service	Not-in-family	Black	Female	16	<=50K
50	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	13	<=50K
52	Self-emp-not-inc	HS-grad	Married-civ-spouse	Exec-managerial	Husband	White	Male	45	>50K
53	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	40	<=50K

APPENDIX E
SYSTEM BLOCK DIAGRAM

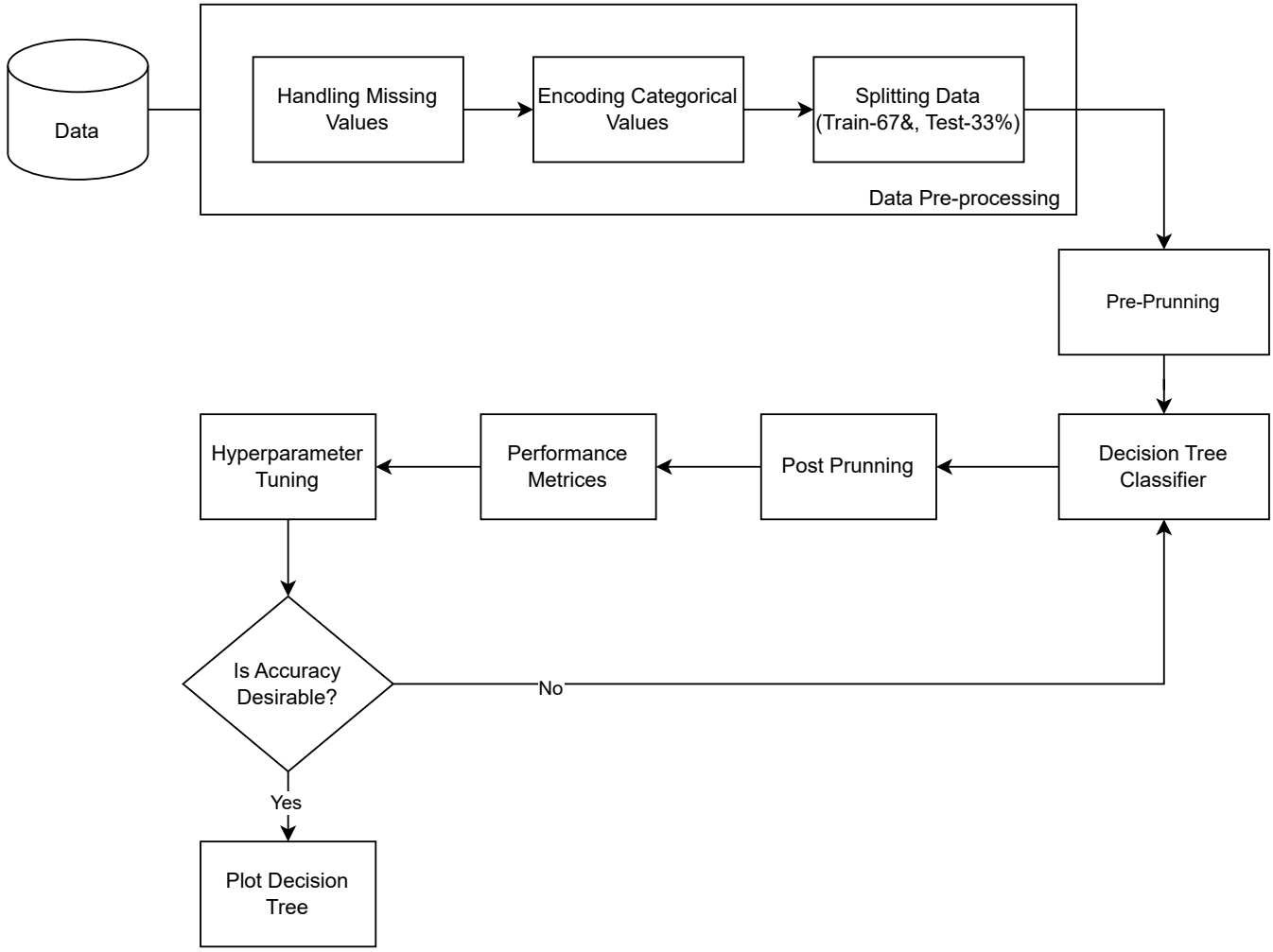

FIGURE 13. Block Diagram for Decision Tree Classifier