Data Analytics Lab : Assignment 4
A Mathematical Essay on Decision Trees

Nikhesh Kumar M
Indian Institute of Technology, Madras
Roll No - GE23C011
ge23c011@smail.iitm.ac.in

***Abstract*—In this project, we present a comprehensive analysis of a dataset with a primary focus on assessing automobile safety. Our analytical approach centers around the utilization of Decision Tree Modeling, a robust and interpretable tool in the realm of predictive analytics. Throughout this study, we endeavor to shed light on the core principles of Decision Trees and their central role in evaluating car safety.**

Furthermore, we embark on an exploration of the dataset, methodically unearthing insights, patterns, and trends associated with car safety, which encompasses vehicle attributes, maintenance considerations, and safety features. As we conclude this study, we contribute to a more profound comprehension of safety evaluations for various car models, facilitating well-informed decisions within the automotive industry.

***Index Terms*—Decision trees, predictive modelling, Car safety assessment, Decision Tree interpretation, Data-driven insights, data analysis, classification accuracy.**

I. Introduction

A. Broad Overview

Decision Tree modeling is a powerful technique for evaluating and predicting car safety in the given dataset. By recursively partitioning the data based on specific attributes, we obtain valuable insights into the factors influencing car safety ratings.

B. Decision Trees

Implementing decision tree model offers a transparent and easily interpretable way to understand the complex relationship between various features of a car and their impact on safety, making it a valuable tool for safety assessment and decision-making in the automotive industry. Two advantages of decision tree modelling is that :

a) Robustness: Decision trees are highly robust in terms of interpretability. They provide a clear and intuitive representation of decision-making processes, making them suitable for tasks where transparency is crucial. This transparency is valuable for understanding model predictions and ensuring the model aligns with domain knowledge.

b) Data distribution and feature handling: Decision trees do not make strong assumptions about the data distribution, which can make them robust when dealing with various types of data, including both categorical and continuous features. They can handle data with missing values and outliers as well. Decision trees are robust to irrelevant features. They can naturally filter out less important attributes when making splits, leading to efficient feature selection.

C. Problem statement

This report aims to employ machine learning model for classifying vehicles into distinct safety categories using a Car Evaluation Database. The central goal is to implement a Decision Tree model to deliver an interpretable solution for car safety assessment, with substantial implications for both consumers and the automotive industry.
Key steps include a comprehensive database analysis, addressing missing data, encoding categorical variables, and partitioning the dataset into training and testing subsets while ensuring accurate labeling of the target variable representing safety classes.
Additionally, a comparative evaluation will be conducted to assess the Decision Tree model's strengths and limitations in the context of car safety classification, including performance bench-marking against alternative classification algorithms.

II. Decision Trees

Implementation of Decision Trees

Decision Trees employ algorithms to create a tree structure that optimally splits the data by minimizing impurity measures such as Gini impurity or entropy at each node. These calculations enable Decision Trees to quantitatively assess feature importance and make predictions regarding car safety, thus enhancing data-driven decision-making in the automotive industry. To implement Decision tree for the given dataset (df), we follow the below steps:

a) Data preprocessing: We begin by data pre-processing, a crucial initial step. This encompasses tasks such as addressing missing values, encoding categorical variables, and segmenting the dataset into distinct training and testing subsets. It is imperative to ensure that the target variable y is appropriately designated, denoting the class labels we aim to predict using the information from feature vector X.

b) Model Training: We create an instance of the Decision Tree classifier model, `DecisionTreeClassifier` in the `scikit-learn` imported library. Then we customize the model based on the type of decision tree we need, whether

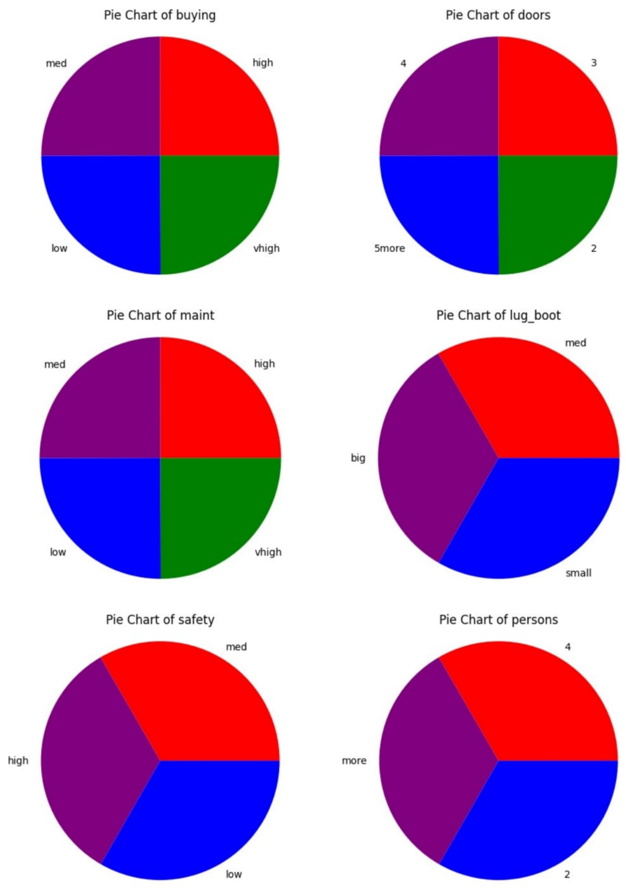

Fig. 1: Pie chart visualisation of categorical variables

it's a classification or regression tree. In this case, we use the classification tree to classify the target (qualitative response) into different classes. We train the model on the training dataset `(X_train,y_train)`, where `X_train` contains the feature vectors, and `y_train` contains the corresponding class labels or target values.

c) Model Prediction: After training, use the trained Decision tree model to make predictions on new, unseen data. We pass the feature vectors of the test dataset `X_test` to the `predict` method, which splits the nodes into different classes and assigns the class with the highest probability as the predicted label.

d) Model Evaluation: We evaluate the model's performance using appropriate metrics (e.g., accuracy, precision, recall, F1-score) from the `metrics` module in the chosen machine learning library. Compare the predicted labels `y_pred` to the actual labels `y_test` to assess how well the model generalizes to new data.

A. Architecture of decision trees

The top node of the tree is called the root node. It represents the entire dataset. At this node, a decision is made about which feature to split the data on. The points along the tree where the predictor space is split are referred to as internal nodes.

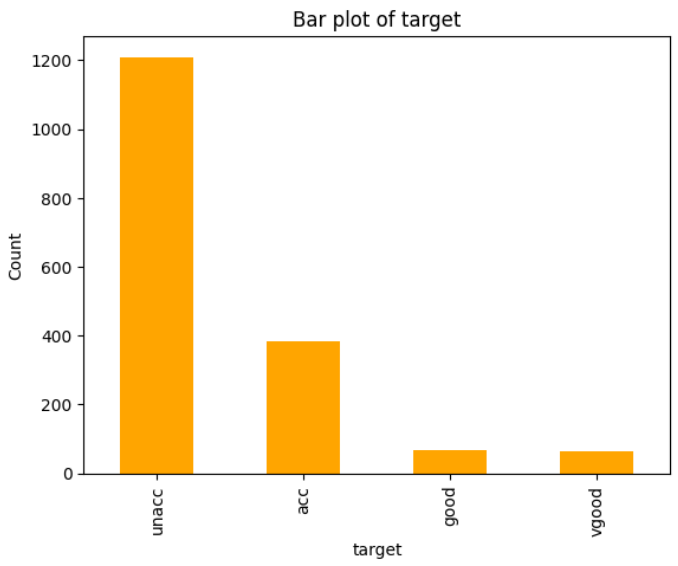

Fig. 2: Target variable bar plot

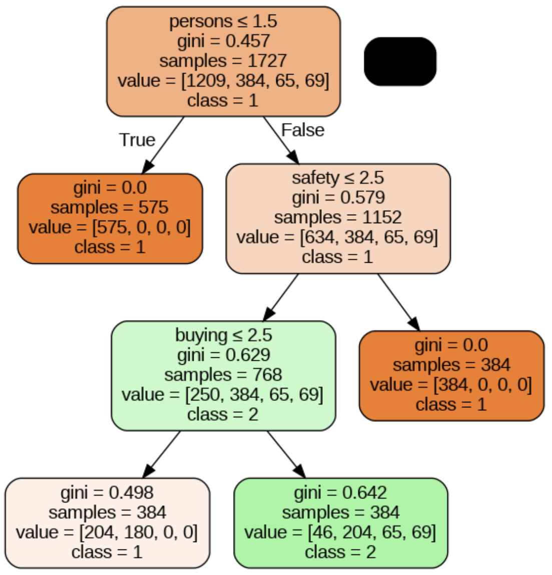

Fig. 3: Decision tree for the training dataset under the Gini index criterion

We refer to the segments of the trees that connect the nodes as branches. Leaf nodes are the terminal nodes of the tree and do not have any child nodes. These nodes represent the final prediction or classification. The decision tree model assigns a class label or a regression value to instances that reach a leaf node.

B. Splitting rules

Different algorithms can be used to determine the best splitting rule at each node. For example, the ID3 algorithm uses information gain, C4.5 uses gain ratio, and CART uses Gini impurity for classification tasks and mean squared error for regression.

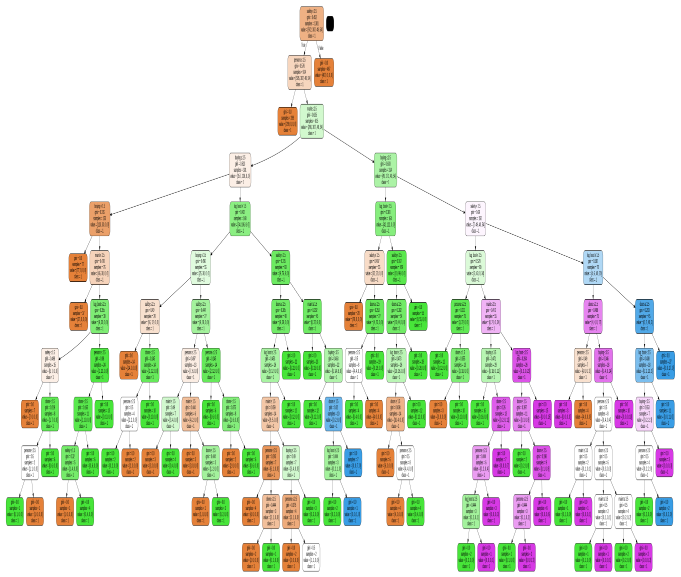
Fig. 4: Decision tree model under the Gini index criterion

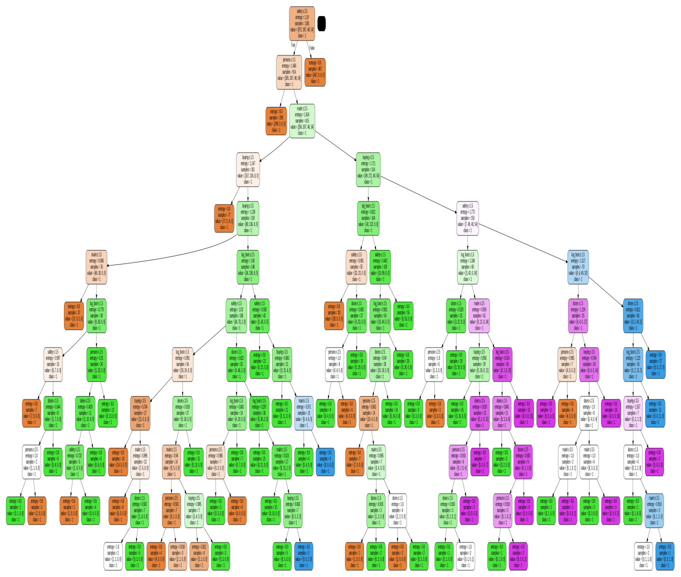
Fig. 5: Decision tree model under the Entropy criterion

C. *Performance of the Model*

The CART (Classification and Regression Trees) algorithm uses a cost function to evaluate the quality of potential splits when building decision trees for both classification and regression tasks. In classification, the Gini impurity is commonly used as the cost function for CART.

$$Gini(p) = 1 - \sum_{i=1}^{J} (p_i)^2$$

where Gini(p) is the Gini impurity for a node with probability distribution p across J different classes. p_i is the probability of class i in the node. The Gini impurity measures the level of disorder in a set of data points. A lower Gini impurity indicates that the data points are more pure and belong to the same class, while a higher Gini impurity suggests a mixture of different classes. When building a decision tree, the goal is to minimize the Gini impurity by selecting splits that result in more pure child nodes.

Another criterion alternative to the Gini index is the Entropy criterion. Entropy measures the impurity or disorder in a dataset. The idea is to minimize entropy when making splits in the dataset, just as with the Gini impurity. A lower entropy indicates that a node is more pure and contains mostly one class, while a higher entropy suggests a more mixed set of classes in the node.

$$H(S) = -\sum_{i=1}^{J} p_i \log(p_i)$$

$H(S)$ is the entropy for a set S with probability distribution $p_1, p_2, ..., p_J$ across J different classes. p_i is the probability of class i in the set.

Entropy measures the amount of information or randomness in the data. The more mixed the classes are in a node, the higher the entropy. The goal in decision tree construction is to minimize the entropy by selecting splits that result in more homogeneous child nodes, leading to more pure and informative decisions in the tree.

Log loss, also known as cross-entropy loss, is a cost function commonly used for classification tasks, especially in the context of binary and multi-class classification. It quantifies how well a classification model's predicted probabilities align with the actual class labels. It is particularly useful when dealing with probabilistic models, such as logistic regression and tree-based classifiers like decision trees.

$$\text{Log-Loss} = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

Log-Loss is the log loss cost function. N is the number of samples or data points in the dataset. y_i is the actual binary class label for the i-th sample (0 or 1 in binary classification). p_i is the predicted probability of the i-th sample belonging to class 1 (for binary classification) or the corresponding class (for multi-class classification).

Log-loss is a commonly used metric for evaluating the performance of classification models, and it is often used in scenarios where probabilistic predictions are essential, such as in big data applications where understanding the confidence of the model's predictions is critical.

a) Training Complexity: During the training phase, a Decision Tree algorithm compares all features on all training samples at each node. This process continues recursively as the tree is constructed. As a result, training complexity is

$$O(n \times m \times log(m))$$

where: n is the number of training instances, m is the number of features, $log(m)$ represents the depth of the tree.

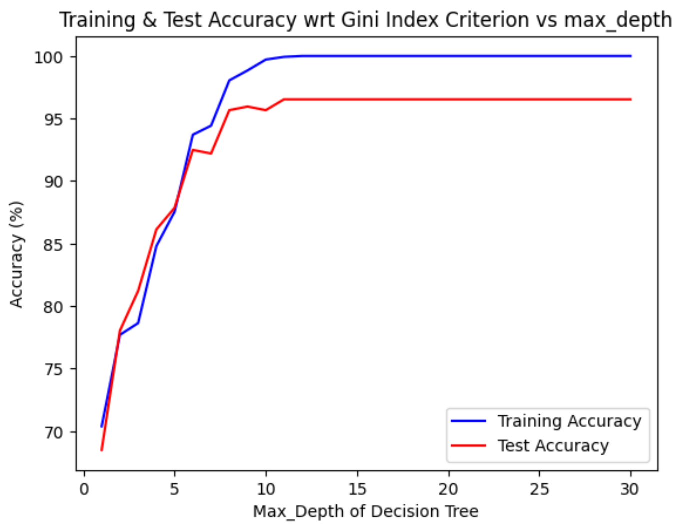

Fig. 6: Training and Test dataset accuracy under Gini index criterion

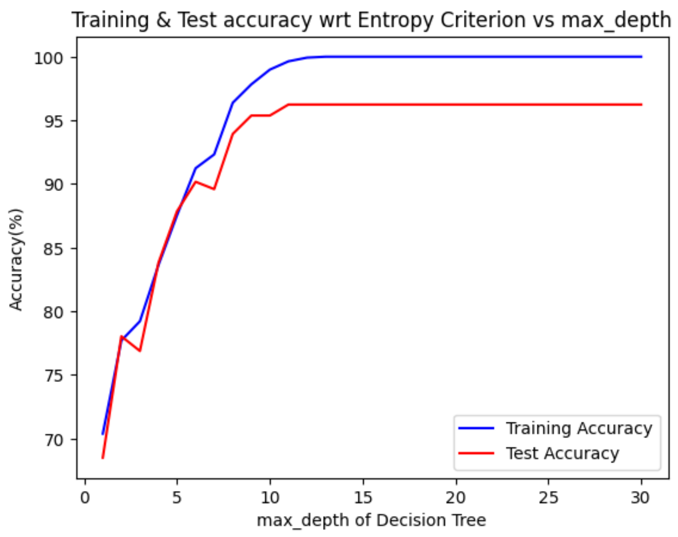

Fig. 7: Training and Test dataset accuracy under Entropy criterion

This implies that the training complexity can be quite high for data sets with many features or a large number of training instances.

b) Prediction Complexity: In contrast, the prediction phase with Decision Trees is highly efficient. To make predictions, one needs to traverse the tree from the root node to a leaf node. Decision Trees are typically balanced, so this traversal requires examining approximately $O(log(m))$ nodes. Since each node only involves checking the value of one feature, the prediction complexity is $O(log(m))$. Importantly, this prediction complexity is independent of the number of features, making predictions very fast, even for datasets with many features.

We observe that `scikit-Learn` offers an option to speed up training for small datasets with the ‘presort‘ parameter set to ‘True‘. However, using ‘presort‘ can slow down training considerably for larger datasets due to the extra computational overhead of sorting the data.

In summary, while training Decision Trees can be computationally intensive, especially for large datasets, their prediction phase is highly efficient and can be performed quickly. Adjusting the ‘presort‘ parameter is an option to speed up training for smaller datasets, but it may not be suitable for larger ones due to the associated slowdown.

D. *Limitations of Decision trees model*

- Over-fitting: Decision trees are prone to over-fitting, especially when they are deep or have many branches. Over-fitting occurs when the tree captures noise and idiosyncrasies in the training data, resulting in poor generalization to unseen data. Techniques like tree pruning and setting maximum tree depth can help address this issue.
- High Variance: Decision trees can have high variance, which means they are sensitive to small changes in the training data. Different splits and tree structures can result from slight variations in the input data, making the model less stable.
- Not Suitable for Continuous Output: While decision trees can handle regression tasks, they may not perform well when the relationship between features and the target variable is continuous and complex. Other regression algorithms may be more appropriate in such cases.
- Large Trees: For complex data sets, decision trees can grow to become very large and deep, making them challenging to visualize and interpret. Large trees are more likely to over-fit, and their complexity may hinder their usefulness
- Sensitive to Irrelevant Features: Decision trees can be sensitive to irrelevant or noisy features in the dataset, which can lead to sub-optimal splits and reduced model performance.
- Greedy Nature: Decision tree algorithms use a greedy approach to select the best split at each node. This means they may make locally optimal decisions without considering the global structure of the tree, potentially leading to sub-optimal overall performance.
- Limited Extrapolation: Decision trees are not well-suited for extrapolation, meaning they may not make reliable predictions outside the range of values present in the training data. This can be a limitation when dealing with data that spans a wide range of values.

III. Data preprocessing and cleaning

1) Data Collection and pre-processing Before proceeding to analyse the data, data was preprocessed and cleaned. This encompassed visualizing data distributions, detect-

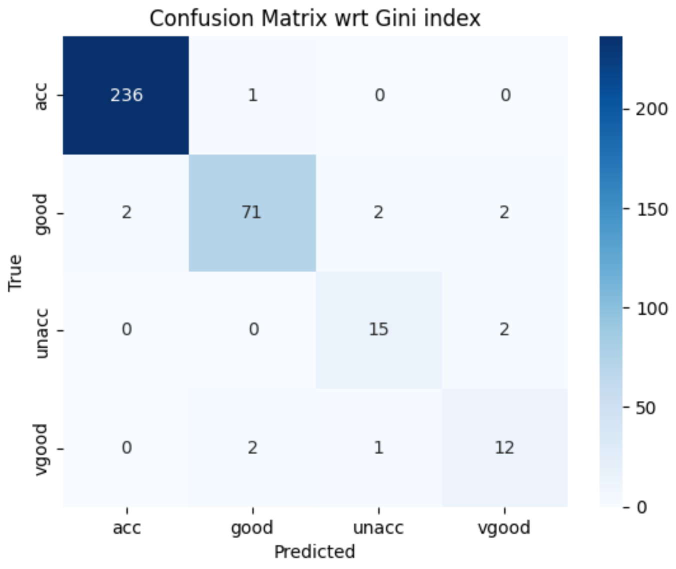

Fig. 8: Confusion matrix under Gini index criterion

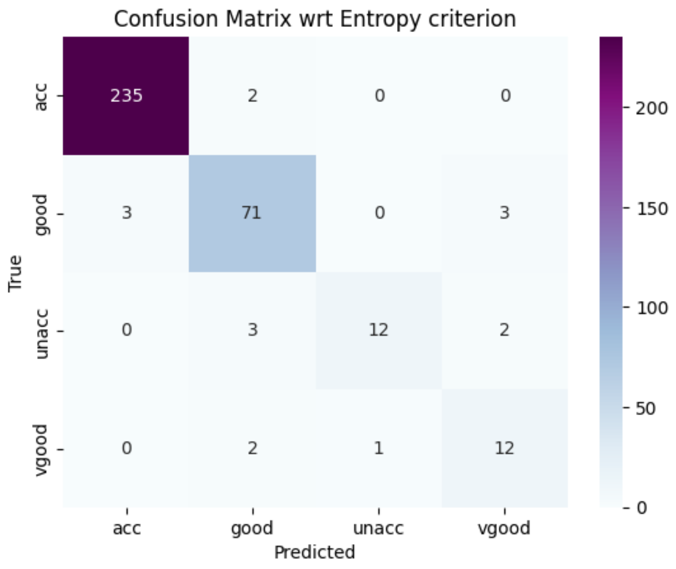

Fig. 9: Confusion matrix under Entropy criterion

ing missing data, and recognizing any potential anomalies. There was no missing data observed and we used

2) To convert the categorical variables into numerical variables, `category_encoders` library was imported which specializes in encoding categorical variables for machine learning models. Specifically, `ce.OrdinalEncoder` is a class within the `category_encoders` library that is used here to perform ordinal encoding. We perform Ordinal encoding, which is a method of converting categorical data into numerical values while preserving the ordinal relationship between categories. Ordinal encoding assigns a unique integer to each categorical variable, maintaining the order of the categories. In the given dataset, ”low” is encoded as 1, ”medium” as 2, and ”high” as 3, and ”vhigh” as 4. After fitting and transforming the categorical variables, the data is now primed for training and testing predictive algorithms.

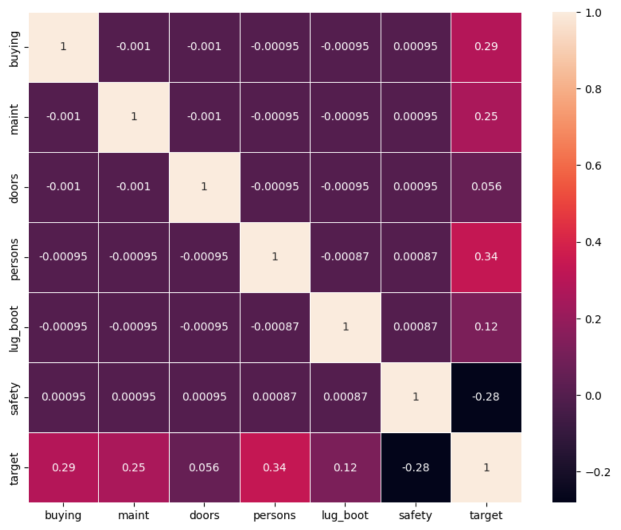

Fig. 10: Correlation matrix

IV. Data analysis and visualization

The performance of decision tree model Car safety evaluation data set was evaluated with the help of three metrics: Accuracy, confusion matrix and predicted classification of classes via decision tree.

In figures 1 and 2, we present the visualisation of categorical variables in the form of pie charts and bar plot.

In figures 3,4,5, Decision tree modelling is visualised under the Gini index and entropy criterion.

Figures 6,7,8 and 9 highlight the accuracy metric and confusion matrix under the different criteria of decision trees.

The last figure 10, gives the correlation matrix between different categorical variables.

V. Conclusion

The Gini criterion model demonstrates robust predictive capabilities in car safety evaluation, achieving an impressive overall accuracy of 97%. It excels in high-precision classification of ’acc’ vehicles (0.99) and maintains substantial recall rates for ’good’ (0.92) and ’unacc’ (0.88) classes. However, the ’vgood’ class exhibits a slightly lower F1-score (0.77). The weighted average F1-score of 0.97 effectively accounts for class imbalance, emphasizing the model’s exceptional overall performance in this multifaceted task.

In contrast, the Entropy criterion model demonstrates commendable predictive abilities in car safety assessment, achieving an impressive overall accuracy of 95%. It excels in high-precision classification of the ’acc’ vehicles (0.99) and maintains respectable precision for the ’good’ class (0.91). However, precision for ’unacc’ (0.92) and ’vgood’ (0.71) classes lags slightly. The model exhibits an exemplary recall rate for the 'acc' class (0.99) and 'good' class (0.92), encapsulating a significant portion of instances. Nonetheless, it faces challenges with lower recall rates for 'unacc' (0.71) and 'vgood' (0.80) classes, indicating room for improvement. The weighted average F1-score of 0.95 underlines the model's substantial overall performance, considering class distribution intricacies, while the macro average F1-score of 0.86 signifies commendable overall performance with nuanced variations.

In summary, the Gini criterion model performs well in classifying "acceptable" and "good" cars, with room for improvement in identifying "unacceptable" and "very good" cars. Conversely, the Entropy criterion model excels in assessing "acceptable" and "good" cars, with opportunities for improvement in evaluating "unacceptable" and "very good" vehicles.

References

[1] Géron, Aurélien. (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O'Reilly Media, Inc., 144-153.

[2] Daniela Witten, Gareth M. James, Trevor Hastie, Robert Tibshirani., (2023) *An Introduction to Statistical Learning.* Springer Texts in Statistics, Springer Nature Switzerland AG, 337-341.

[3] Pedregosa et al., (2011) *Scikit-learn: Machine Learning in Python.* JMLR 12, pp. 2825-2830.

[4] Byron Roots, (2020) *Lecture 3* CSE446: Machine Learning, Winter 2020 Lecture Notes, University of Washington.

[5] John F. Magee (1964) *Decision Trees for Decision-Making.* The Harvard Business Review Magazine, 1 Jul, 1964.