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

Arjav Singh
Metallurgical and Materials Engineering
Indian Institute of Technology Madras
Chennai, India
mm20b007@smail.iitm.ac.in

***Abstract*—This study is a mathematical essay on the Decision Tree methods wherein it is applied on a dataset containing different attributes of a car, and the key task is to classify it as unacceptable, acceptable, good, or very good based on its safety while considering other factors like buying price, cost of maintenance, number of doors, capacity in terms of persons to carry, size of the luggage boot, and determine whether some characteristics of the car are more likely to make it a good choice.**

***Index Terms*—Introduction, Decision Trees, Data & Problem, Conclusion**

I. Introduction

This study focuses on a comprehensive empirical analysis of the factors influencing car safety, with the Car Evaluation Database serving as the primary dataset. It is observed that several factors, including maintenance price, purchase price, luggage capacity, and seating capacity, are found to be significantly affected by the safety category assigned to different cars. Decision Trees are a modeling technique to predict safety categories based on individual attributes such as maintenance cost, purchase price, and various vehicle capacities.

The research methodology begins with collecting, cleaning, and preparing the raw data. Subsequently, exploratory data analysis is conducted to gain insights into the dataset's characteristics. Statistical models are then constructed, creating visual representations to provide quantitative and visual evidence supporting the relationships observed. In the following section, the fundamental principles that underpin Decision Trees as a modeling tool are discussed.

This study later provides a presentation and discussion of the insights and observations derived from the data analysis and the models that have been developed. The noteworthy findings, patterns, and trends that have emerged throughout the study are highlighted here. The objective is to comprehensively understand how car safety is affected by factors such as maintenance costs, purchase prices, and various vehicle capacities.

The concluding section summarizes the key highlights and significant features of the research. Potential avenues for further investigation are outlined, suggesting areas where future research could expand upon the findings. A contribution is made to a deeper understanding of the factors affecting car safety, with valuable insights provided for car manufacturers and consumers in making informed decisions about vehicle safety and performance.

II. Decision Tree

Decision tree learning, also known as decision tree induction, is widely used in statistics, data mining, and machine learning. It employs a tree structure to make predictions based on data observations. The tree branches represent conditions or features, and the leaves provide the predicted outcomes.

Nowadays, the Decision Tree algorithm is known by its modern name, CART, which stands for Classification and Regression Trees. Classification and Regression Trees or CART is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can be used for classification and regression modeling problems.

Decision trees can visually and explicitly depict decision processes in decision analysis. In data mining, decision trees describe data patterns, which can be used as a basis for decision-making.

Based on the type of the target variable, we have two varieties of decision trees -

1) **Categorical Variable Decision Tree**: A decision tree with a categorical target variable is called a Categorical variable decision tree.
2) **Continuous Variable Decision Tree**: A decision tree with a continuous target variable is called a Continuous Variable Decision Tree.

A. *Decision Tree Terminologies*

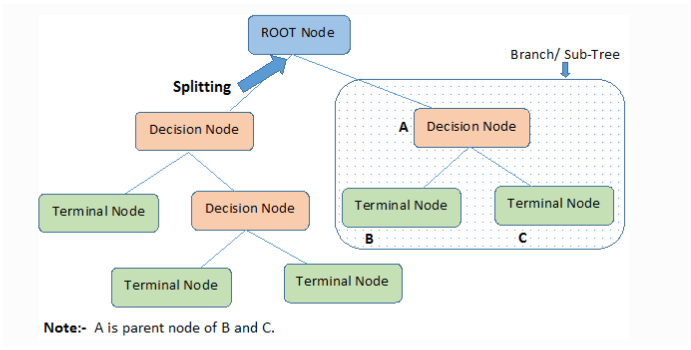

Fig. 1. Basic structure of a Decision Tree

- **Root Node**: It represents the entire population or sample and is divided into two or more homogeneous sets.

- **Splitting**: It is the process of dividing a node into two or more sub-nodes.
- **Decision Node**: When a sub-node splits into further sub-nodes, it is called the decision node.
- **Leaf / Terminal Node**: Nodes that do not split are called Leaf or Terminal nodes.
- **Pruning**: When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
- **Branch / Sub-Tree**: A subsection of the entire tree is called a branch or sub-tree.
- **Parent and Child Node**: A node divided into sub-nodes is called a parent node of sub-nodes, whereas sub-nodes are the child of a parent node.

B. Working of Decision Tree

Decision trees classify the problem by sorting them from the root to some leaf/terminal node, with the leaf/terminal node classifying the subproblems.

Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This recursive process is repeated for every subtree rooted at the new node.

The accuracy of a tree depends significantly on the strategic choices made when deciding how to split it. These choices differ for classification and regression trees.

Decision trees employ various algorithms to determine the splits, aiming to divide a node into two or more sub-nodes. This process enhances the similarity or homogeneity among the resulting sub-nodes. In simpler terms, it increases the purity of each node concerning the target variable. The decision tree assesses all available variables for potential splits and chooses the one that leads to the most homogenous sub-nodes.

C. Attribute selection measures

The primary challenge in Decision Tree implementation lies in selecting attributes for each level, including the root node, a process known as attribute selection. Two prominent attribute selection measures are commonly used:

1) Information Gain: When Information Gain is employed as the criterion, attributes are treated as categorical. In contrast, the Gini Index assumes attributes to be continuous. Let's delve into these attribute selection measures:

Information Gain: Information Gain is an attempt to estimate the information contained in each attribute. To grasp this concept, it's essential to understand another concept called Entropy.

Entropy: Entropy quantifies the impurity within a given dataset. In Physics and Mathematics, entropy is synonymous with the randomness or uncertainty of a random variable (X). In information theory, it signifies the impurity within a group of examples. Information Gain, in this context, is the reduction in entropy. It calculates the difference between the entropy before a split and the average entropy after a split based on the given attribute values.

The formula for Entropy is represented as:

$$Entropy = -\sum_{i=1}^{c} p_i \cdot \log_2(p_i)$$

Here, 'c' is the number of classes, and p_i is the probability associated with the i^{th} class.

The ID3 (Iterative Dichotomiser) Decision Tree algorithm employs entropy to compute information gain. By evaluating the decrease in entropy for each attribute, the attribute with the highest information gain is chosen as the splitting attribute at the node.

2) Gini Index: Another attribute selection measure used by CART (Categorical and Regression Trees) is the Gini Index. It uses the Gini method to create split points.

Gini Index: The Gini Index can be represented with the following formula:

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

Here, 'c' is the number of classes, and p_i is the probability associated with the i^{th} class.

The Gini Index implies that when two items are randomly selected from a population, they must belong to the same class with a probability of 1 if the population is pure. Gini Index works with a categorical target variable such as "Success" or "Failure" and performs binary splits. A higher Gini value indicates greater homogeneity.

Here are the steps to calculate the Gini Index for a split:

1) Calculate the Gini Index for sub-nodes using the formula, which is the sum of the squares of the probabilities for success and failure ($p^2 + q^2$).
2) Calculate the Gini Index for the split using the weighted Gini score of each node within that split.

For discrete-valued attributes, the subset that yields the minimum Gini Index is chosen as the splitting attribute. In the case of continuous-valued attributes, the strategy considers each pair of adjacent values as a potential split point, and the point with the smaller Gini Index is chosen as the splitting point. The attribute with the minimum Gini Index is ultimately selected as the splitting attribute.

D. Random Forest

Bootstrap aggregating, commonly known as bagging (derived from "bootstrap aggregating"), is an ensemble meta-algorithm in machine learning. Its purpose is to enhance the stability and accuracy of machine learning algorithms, particularly those used in statistical classification and regression. An additional benefit of bagging is its ability to reduce variance and mitigate overfitting issues. While it is often associated with decision tree methods, it is adaptable to various algorithms. Bagging represents a specific case within the broader model averaging approach.

As suggested by its name, Random Forest comprises many individual decision trees that function as a collective ensemble. Each tree within the random forest generates a class prediction, and the class with the most votes is considered the model's prediction. This ensemble approach leverages many relatively uncorrelated models (trees) operating as a committee, resulting in superior performance to any constituent models.

E. XGBoost

Gradient boosting is a versatile machine-learning technique for regression, classification, and other tasks. It produces a prediction model as an ensemble of weak prediction models, typically decision trees. When decision trees serve as the weak learners, this approach is known as gradient-boosted trees, and it often surpasses the performance of random forests.

The process of building a gradient boosting model is stage-wise, similar to other boosting methods. However, it extends their capabilities by enabling the optimization of arbitrary differentiable loss functions.

XGBoost is a notable algorithm that has gained prominence in applied machine learning and Kaggle competitions, particularly for structured or tabular data. It efficiently implements gradient-boosted decision trees designed for speed and performance. Created by Tianqi Chen, it has seen contributions from numerous developers.

XGBoost operates differently from traditional gradient boosting, which works in function space using the Newton-Raphson method. In contrast, gradient boosting functions in function space employ gradient descent. Using a second-order Taylor's approximation in the loss function establishes a connection to the Newton-Raphson method.

F. Overfitting in Decision Tree algorithm

Overfitting is a practical concern when constructing Decision Tree models. It manifests when the algorithm delves too deeply into the tree structure, attempting to minimize training-set errors yet inadvertently increasing test-set errors, resulting in reduced predictive accuracy. Overfitting often occurs when the model creates excessive branches due to data outliers and irregularities.

To address overfitting, two common approaches are employed:

1) **Pre-Pruning**: Pre-pruning involves halting tree construction prematurely. Nodes are not split if their quality measure falls below a predefined threshold. However, selecting the appropriate stopping point can be challenging.
2) **Post-Pruning**: Post-pruning, on the other hand, builds the tree to its full depth. If the tree exhibits overfitting, pruning is applied as a post-processing step. Cross-validation data is used to assess pruning impact. Decisions are made based on whether expanding a node improves accuracy. If it does, expansion continues; otherwise, the node is converted into a leaf node.

G. Metrics for model evaluation

1) **Confusion Matrix**: It is used to summarize the performance of a classification algorithm on a set of test data for which the true values are previously known. Sometimes it is also called an error matrix. Terminologies of the Confusion matrix (Figure 1) are:

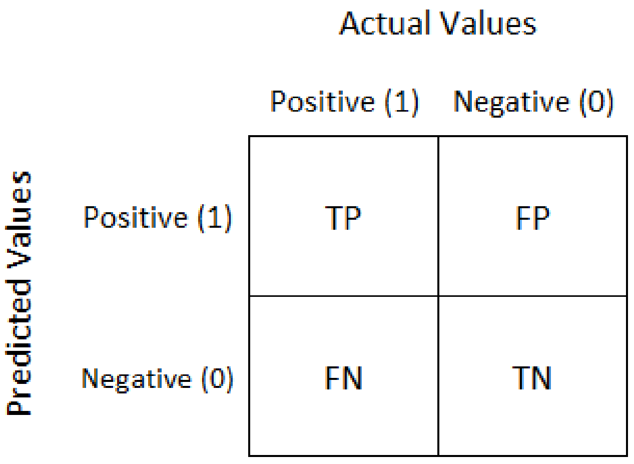

Fig. 2. Confusion Matrix.

- **True Positive**: TP means the model predicted yes, and the actual answer is also yes.
- **True negative**: TN means the model predicted no, and the actual answer is also no.
- **False positive**: FP means the model predicted yes, but the actual answer is no.
- **False negative**: FN means the model predicted no, but the actual answer is yes.

The rates calculated using the Confusion Matrix are:

a) **Accuracy**: (TP+TN/Total) tells about overall how classifier Is correct.
b) **True positive rate**: TP/(actual yes) it says about how much time yes is predicted correctly. It is also called "sensitivity" or "recall."
c) **False positive rate**: FP/(actual number) says how much time yes is predicted when the actual answer is no.
d) **True negative rate**: TN/(actual number) says how much time no is predicted correctly, and the actual answer is also no. It is also known as "specificity."
e) **Misclassification rate**: (FP+FN)/(Total) It is also known as the error rate and tells about how often our model is wrong.
f) **Precision**: (TP/ (predicted yes)) If it predicts yes, then how often is it correct.
g) **Prevalence**: (actual yes /total) how often yes condition actually occurs.
h) **F1-score**: f1 score is defined as the weighted harmonic mean of precision and recall. The best achievable F1 score is 1.0, while the worst is 0.0. The F1 score serves as the harmonic mean of precision and recall. Consequently, the F1-score consistently yields lower values than accuracy measures

since it incorporates precision and recall in its computation. When evaluating classifier models, it is advisable to employ the weighted average of the F1 score instead of relying solely on global accuracy.

2) **ROC curve (Receiver Operating Characteristic)**: The Receiver Operating Characteristic (ROC) curve is a useful tool for assessing a model's performance by examining the trade-offs between its True Positive (TP) rate, also known as sensitivity, and its False Negative (FN) rate, which is the complement of specificity. This curve visually represents these two parameters.
The Area Under the Curve (AUC) metric to summarize the ROC curve concisely. The AUC quantifies the area under the ROC curve. In simpler terms, it measures how well the model can distinguish between positive and negative cases. A higher AUC indicates better classifier performance.
In essence, AUC categorizes model performance as follows:
   - If AUC = 1, the classifier correctly distinguishes between all the Positive and Negative class points.
   - If 0.5¡ AUC ¡ 1, the classifier will distinguish the positive class value from the negative one because it finds more TP and TN than FP and FN.
   - If AUC = 0.5, the classifier cannot distinguish between positive and negative values.
   - If AUC =0, the classifier predicts all positive as negative and negative as positive.

III. Problem

We have been tasked to analyze various attributes of different cars, such as their purchasing price, maintenance costs, number of doors, passenger capacity, trunk size, and safety ratings. The goal is to identify which car characteristics are more likely to indicate a wise choice.

A. *Exploratory Data Analysis and Feature Generation*

The data is initially read into a pandas data frame. A total of 1728 data points are observed, with 7 columns encompassing various car-related features. When the distributions of the target variable are visualized, a multi-class imbalanced dataset problem is evident. Around 70% of the total cars are classified as unacceptable, 22.2% as just acceptable, 4% as good, and 3.8% as very good (as shown in Figure 1). It is observed that all 6 features are categorical and are most likely ordinal. Subsequently, the data is checked for null values, and it is found that no NaN values are present. The next step involves checking for features with high cardinality, which refers to the number of unique values each feature can take. It is discovered that most features have 3/4 unique classes, most of which are balanced. Therefore, it is concluded that this is indeed clean data.

- *Buying*: The car's purchase price - 'vhigh,' 'high,' 'med,' and 'low'
- *Maintenance*: The cost of maintenance of the car - 'vhigh' 'high' 'med' 'low'
- *Persons*: Seating capacity of the car - '2' '4' 'more'
- *Doors*: The number of doors in the car - '2' '3' '4' '5more'
- *Lug boot*: The car's boot space - 'small' 'med' 'big'
- *Safety*: 'low' 'med' 'high'
- *Target*: 'unacc' 'acc' 'vgood' 'good'

The aim is to predict the multiclass feature Target.

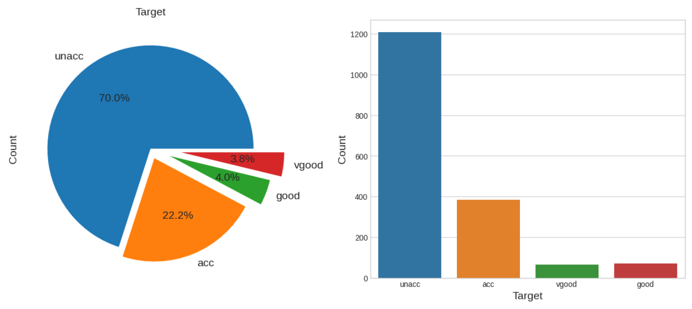

Fig. 3. Distribution of Target Variable

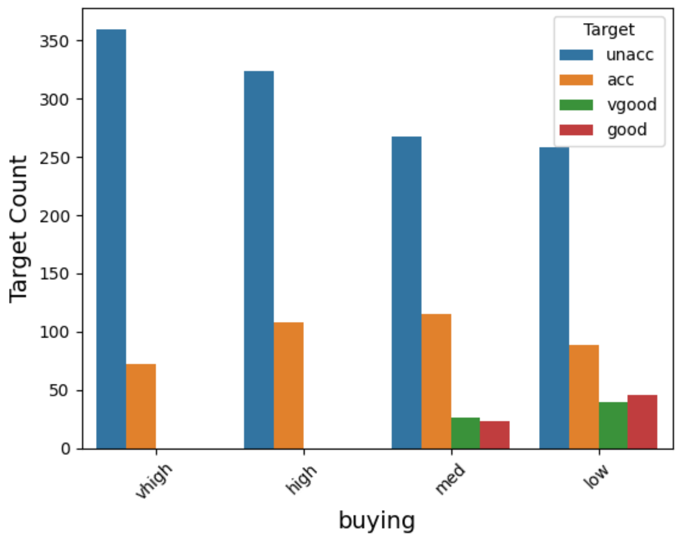

Fig. 4. Target vs Buying

Univariate analysis is initiated by generating histplots for each of the six features, employing the seaborn library, with the target column as the hue. It becomes apparent that certain classes in some features lack specific target classes, which is highly beneficial for our decision tree model, as the model aims to achieve pure leaves. For instance, in the case of the 'buying' feature, it is observed that high and very high buying costs are associated only with unacceptable and acceptable cars. Similarly, a small luggage boot capacity is exclusively linked to cars not being classified as 'very good.' Cars designed for 2 persons are solely seen as unacceptable. In contrast, very high-maintenance cars are either unacceptable or acceptable, and high-maintenance cars are not classified as 'very good.'

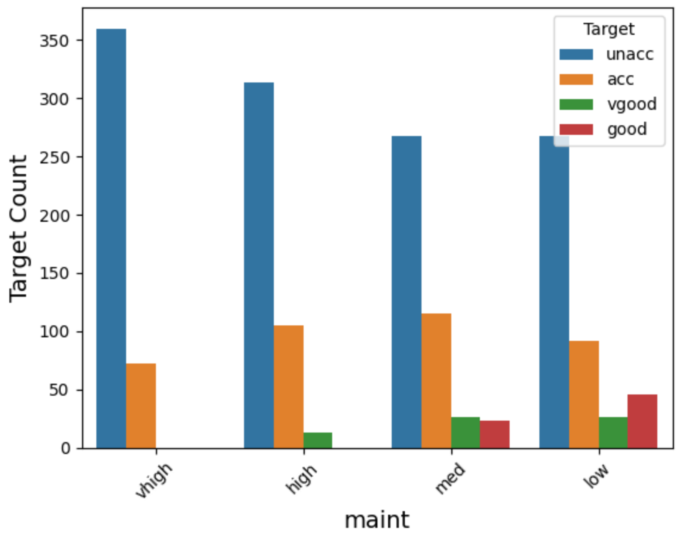

Fig. 5. Target vs Maintenance

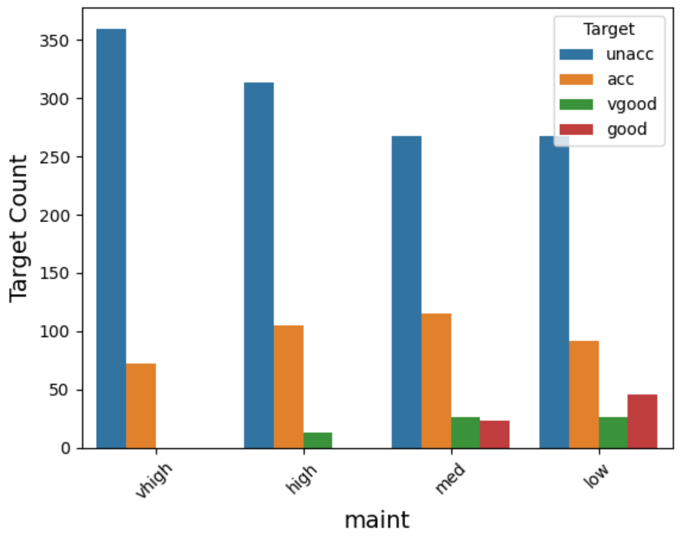

Fig. 7. Target vs Np. of doors

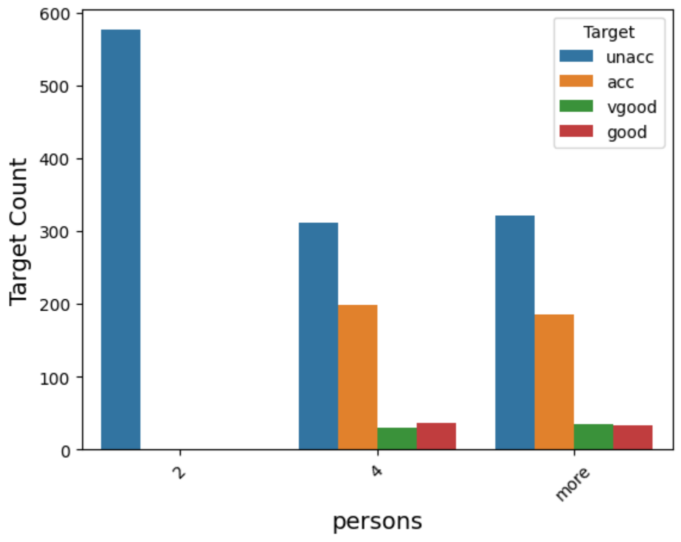

Fig. 6. Target vs No. of people

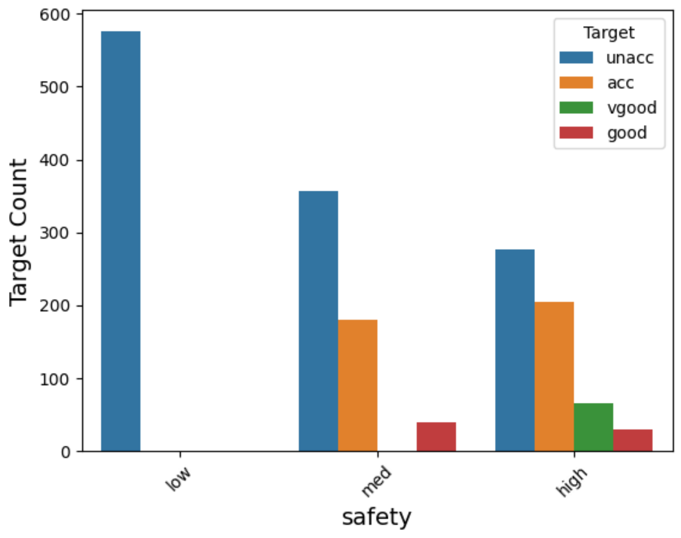

Fig. 8. Target vs Safety

Additionally, cars with low safety ratings are also considered unacceptable.

Proceeding to bivariate analysis, strip plots are constructed for each of the 15 feature pairs, with the target column as the hue, using the seaborn library, while utilizing jitter and dodge values as true. This analysis reveals a similar trend as the univariate analysis, where certain target classes are absent, as expected. A comprehensive analysis has been provided for further examination.

B. Post-Processing

Since the dataset comprises categorical features, it is necessary to encode them suitably. There are two primary methods of encoding:

1) One-hot encoding.
2) Label encoding.

Since the categorical features in this dataset are ordinal, meaning they can be ranked or ordered, label encoding is the appropriate choice. If there were nominal features, one-hot encoding would have been preferred. For this encoding, we utilize the Category Encoders library.

Subsequently, the data is divided using a 70/30 split, resulting in a final dataset with 1209 examples in the training set and 519 in the cross-validation set.

C. Decision Tree Modelling

In this paper, we model three versions of decision-based classifiers, namely Decision Trees, Random Forests, and XG-Boost. We start with modeling the Decision Tree classifier first. We use grid search for finding the best parameters for

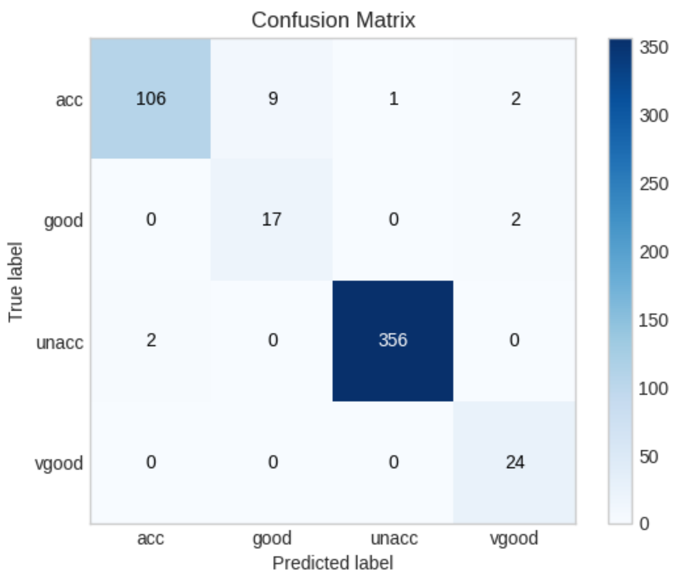

Fig. 9. Confusion Matrix for Decision Tree

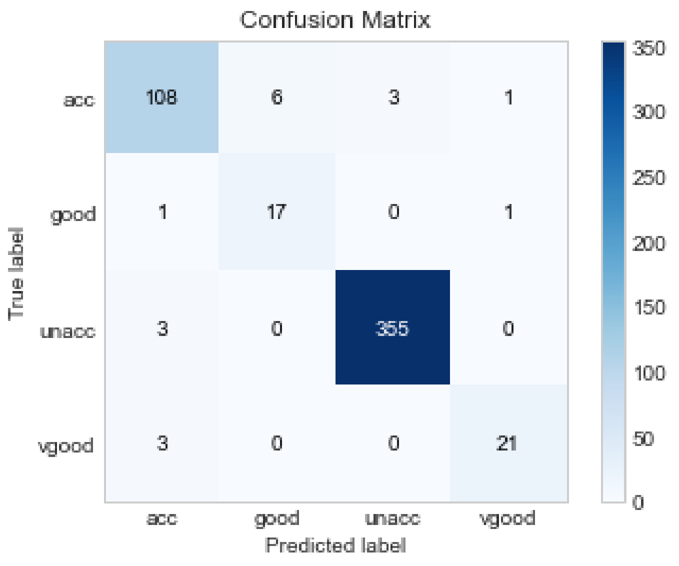

Fig. 10. Confusion Matrix for Random Forest

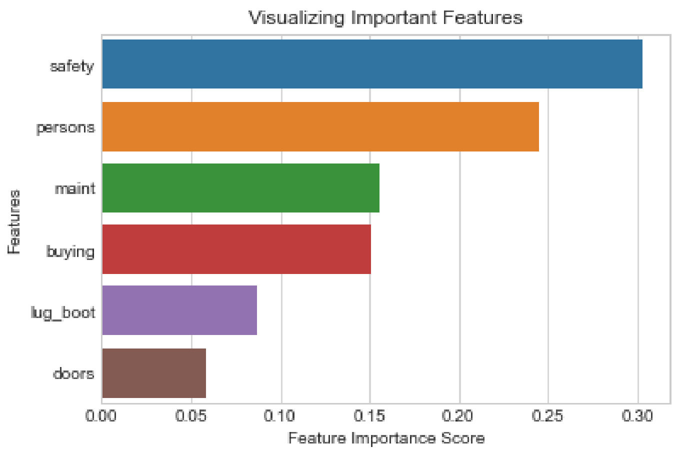

Fig. 11. Feature Importance chart (higher is better)

our decision tree model and checking for model classification accuracy on the cross-validation dataset. Starting with the class weight hyperparameter, we keep it as balanced as this automatically calculates the class weights according to their distribution in the training dataset. The number of jobs is kept as -1 so that it chooses all the CPU cores available for fitting the model. The scoring metric used by us is accuracy. We use grid search for finding the best parameters by defining a range to check. The parameters are:

- Max depth: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain fewer than the minimum samples for a split.
- Min samples split: The minimum number of samples required to split an internal node. If an integer, then consider min samples split as the minimum number. If a float, then min samples split is a fraction, and $\lceil \text{min samples split} \times n \text{ samples} \rceil$ is the minimum number of samples for each split.
- Min samples leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min samples leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

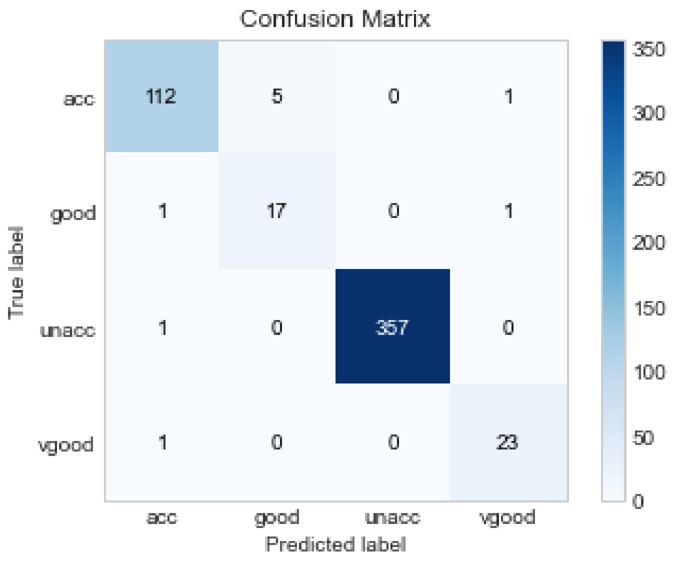

Fig. 12. Confusion Matrix for XGBoost

Finally, we get the best parameters as max depth 9, min samples leaf 2, min samples split 5. Using this, we are able to achieve a train set accuracy of 0.98 and test accuracy of 0.969.

Then we move on to modeling using the Random Forest classifier, which is a bagged version of Decision Trees. In this case, we use randomized search as this does not search the whole space but tries to get to the optimal using randomly sampled values of hyperparameters as this model is expensive to train. The hyperparameters being searched are:

- n estimators: Number of trees in the forest.
- Max features: Maximum number of features considered for splitting a node.
- Max depth: Maximum number of levels in each decision tree.
- Min samples split: Minimum number of data points placed in a node before the node is split.
- Min samples leaf: Minimum number of data points allowed in a leaf node.
- Bootstrap: Method for sampling data points (with or without replacement).

Fitting 3 folds for each of 100 candidates, totaling 300 fits, we find the best parameters as 'n estimators': 800, 'min samples split': 3, 'min samples leaf': 1, 'max features': 'auto', 'max depth': None, 'bootstrap': False. Using this, we are able to achieve a train set accuracy of 1 and test accuracy of 0.969. We then proceed to check for the feature importance as this is provided by the implementation of random forests in the sklearn library. We see that 'safety' has the highest importance with 'doors' having the least. We then train a model dropping the least important feature, but we observe reduced accuracy to 0.969 in the train set and 0.933 in the test set.

At last, we take a look at the boosted algorithm very famously called as XGBoost. XGBoost provides a large range of hyperparameters. We can leverage the maximum power of XGBoost by tuning its hyperparameters. The most powerful ML algorithm like XGBoost is famous for picking up patterns and regularities in the data by automatically tuning thousands of learnable parameters. In this paper, we use the Hyperopt library for finding the best hyperparameters, some of which are:

- Max depth: The maximum depth of a tree, same as GBM. It is used to control over-fitting as higher depth will allow the model to learn relations very specific to a particular sample.
- Gamma: A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split.
- Reg alpha: L1 regularization term on weights (analogous to Lasso regression). It can be used in case of very high dimensionality so that the algorithm runs faster when implemented. Increasing this value will make the model more conservative.
- Reg Lambda: L2 regularization term on weights (analogous to Ridge regression).
- Colsample bytree: is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.
- Min Child Weight: It defines the minimum sum of weights of all observations required in a child. This is similar to min child leaf in GBM but not exactly. This refers to the min "sum of weights" of observations while GBM has min "number of observations".

The best hyperparameters are 'colsample bytree': 0.959166117786024, 'gamma': 7.417890672096632, 'max depth': 13.0, 'merror': 6, 'min child weight': 8.0, 'reg alpha': 53.0, 'reg lambda': 0.73987405692127. Using these, we get an accuracy value of 1 on the training and 0.98 on the test sets.

IV. Conclusion

After conducting a comprehensive analysis of the tree-based models in their raw form, with bagging, and with boosting, it is observed that all of these models perform well. However, XGBoost, which is a boosting-based model, outperforms the others with an accuracy value that is 1.1 percent higher on the test dataset. It is noted that these models exhibit significantly higher accuracy on the training set, suggesting a likelihood of overfitting. Nonetheless, by employing three different hyperparameter search methods, namely grid search, random search, and hyperopt, the hyperparameters that yield the highest accuracy on the cross-validation sets are determined.

It is also evident that the bagging-based method, Random Forest, does not contribute to achieving a better score on the test dataset. Tuning these hyperparameters provides granular control over the trees that are built within the ensemble, resulting in enhanced performance.

As indicated by our univariate and multivariate exploratory data analysis, certain features have classes that do not encompass all the target classes. Furthermore, the distribution of these classes is mostly uniform, which aids the tree-based models in achieving pure leaves and, consequently, high accuracy values. Some of the variables exhibiting this behavior include 'buying' with 'high' and 'vhigh,' 'small' luggage boot, 'very high' and 'high' maintenance, and 'low' safety.

For future work, further avenues of growth could involve exploring additional features that might provide a better explanation of the target variable.

References

[1] KDNuggets, "Decision Tree Algorithm: Explained," KDNuggets, https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html, January 2020.

[2] GeeksforGeeks, "XGBoost," GeeksforGeeks, https://www.geeksforgeeks.org/xgboost/. Accessed: October 11, 2023.

[3] "AUC-ROC Curve & Confusion Matrix Explained in Detail." [Online]. Available: https://www.kaggle.com/code/vithal2311/auc-roc-curve-confusion-matrix-explained-in-detail.

[4] Analytics Vidhya. "K-Fold Cross-Validation Technique and Its Essentials." [Online]. Available: https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/.