Data Analytics Lab: Assignment-5
A Mathematical Essay on Random Forest

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 Random Forest method, which is applied to a dataset containing different car attributes. The key task is to classify the cars as unacceptable, acceptable, good, or very good based on their 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, Random Forest, Data & Problem, Conclusion**

I. Introduction

This study focuses on a comprehensive empirical analysis of the factors influencing car acceptability, 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. Random forest is the modeling technique used to predict the acceptability of the car 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 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 car safety factors, with valuable insights for car manufacturers and consumers in making informed decisions about vehicle safety and performance.

II. Random Forest

A random forest, also known as a random decision forest, is an ensemble learning technique used for tasks such as classification and regression. It works by creating multiple decision trees during the training phase from sample training data with replacement, and the output is based on the majority voting, this process is known as bagging.

In classification, the random forest's output is determined by the most commonly chosen class among the individual trees. Regression returns the mean or average prediction from the individual trees.

Random decision forests effectively address the tendency of decision trees to overfit their training data, resulting in better generalization. While random forests typically outperform standalone decision trees, it's worth noting that they may not achieve the same level of accuracy as gradient-boosted trees. However, this can vary depending on the characteristics of the data.

A. Bagging

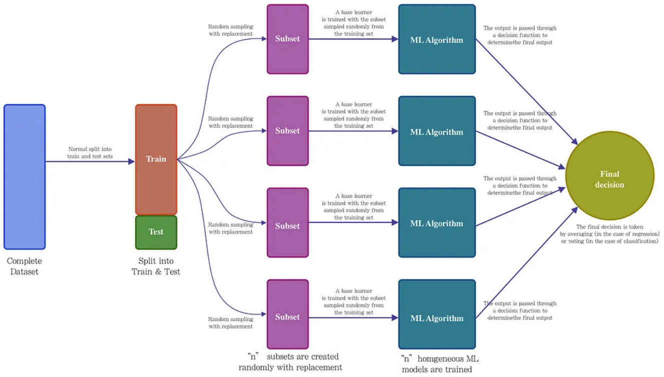

Complete Dataset
Normal split into train and test sets
Train
Test
Split into Train & Test
Random sampling with replacement
Random sampling with replacement
Random sampling with replacement
Random sampling with replacement
Subset
Subset
Subset
Subset
"n" subsets are created randomly with replacement
A base learner is trained with the subset sampled randomly from the training set
A base learner is trained with the subset sampled randomly from the training set
A base learner is trained with the subset sampled randomly from the training set
A base learner is trained with the subset sampled randomly from the training set
ML Algorithm
ML Algorithm
ML Algorithm
ML Algorithm
"n" homgeneous ML models are trained
The output is passed through a decision function to determinethe final output
The output is passed through a decision function to determinethe final output
The output is passed through a decision function to determinethe final output
The output is passed through a decision function to determinethe final output
Final decision
The final decision is taken by averaging (in the case of regression) or voting (in the case of classification)

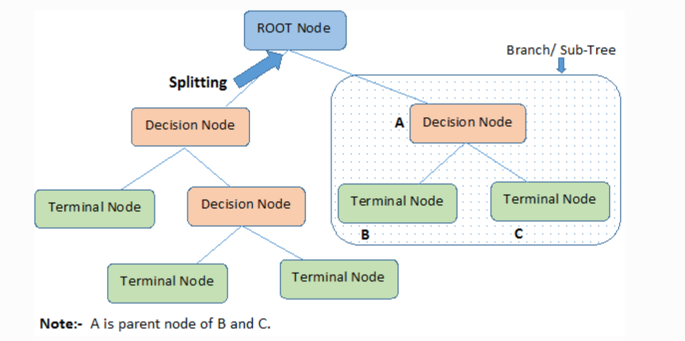
Fig. 1. Steps involved in bagging

Random Forest works on the principle of Bagging, aka Bootstrap Aggregation. It works as an ensemble technique for the random forest algorithm. Here are the steps involved in bagging -

1) **Selection of Subset:** Bagging starts by choosing a random sample, or subset, from the entire dataset.

2) **Bootstrap Sampling:** Each model is created from these samples, called Bootstrap Samples, which are taken from the original data with replacement. This process is known as row sampling.
3) **Bootstrapping:** The step of row sampling with replacement is referred to as bootstrapping.
4) **Independent Model Training:** Each model is trained independently on its corresponding Bootstrap Sample. This training process generates results for each model.
5) **Majority Voting:** The final output is determined by combining the results of all models through majority voting. The most commonly predicted outcome among the models is selected.
6) **Aggregation:** This step, which involves combining all the results and generating the final output based on majority voting, is known as aggregation.

B. Decision Tree Terminologies



Note:- A is parent node of B and C.

Fig. 2. 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.

C. 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.

D. 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.

The subset that yields the minimum Gini Index is chosen as the splitting attribute for discrete-valued attributes. 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.

E. Working of Random Forest

The major idea behind random forest is that each tree might do a relatively good job predicting but will likely overfit on the part of the data. If many trees are built, all of which work and overfit in different ways, we can reduce the amount of overfitting by averaging their results. The random forest algorithm follows the following steps -

1) **Step 1:** In the Random forest model, a subset of data points and a subset of features are selected for constructing each decision tree.
2) **Step 2:** Individual decision trees are constructed for each sample.
3) **Step 3:** Each decision tree will generate an output.
4) **Step 4:** Final output is considered based on Majority Voting or Averaging for Classification and regression, respectively.

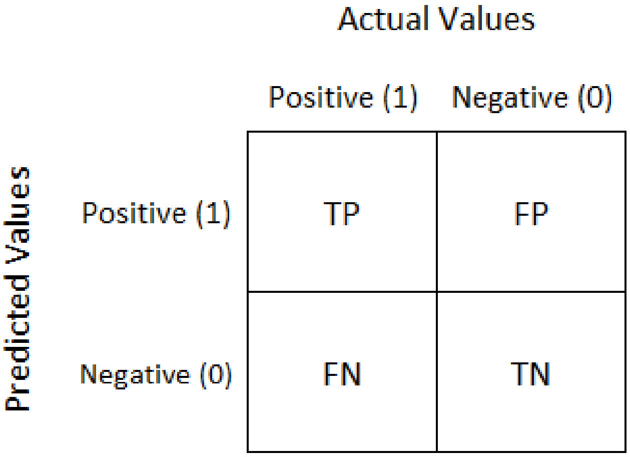
F. Overfitting in Decision Tree algorithm

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



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Fig. 3. Confusion Matrix.

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:
   - **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.

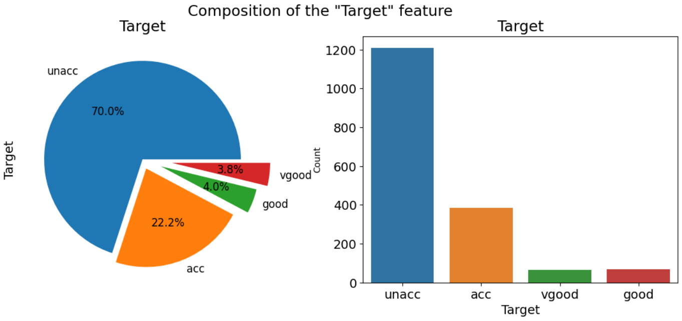


Fig. 4. Distribution of Target Variable

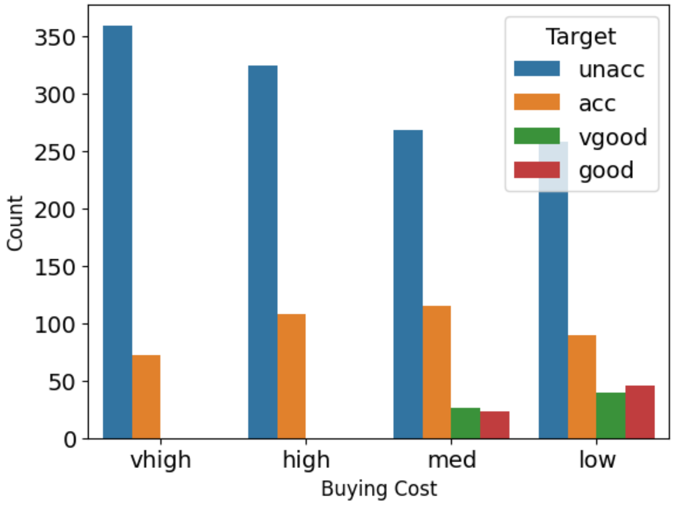


Fig. 5. Target vs Buying

Univariate analysis is initiated by generating a barplot 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

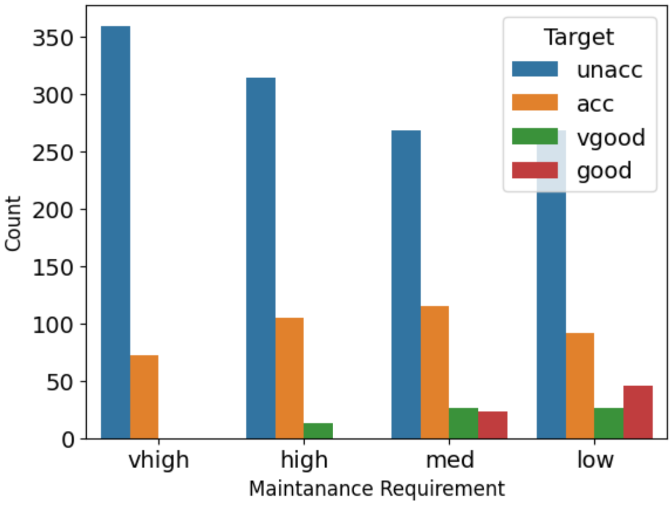


Fig. 6. Target vs Maintenance

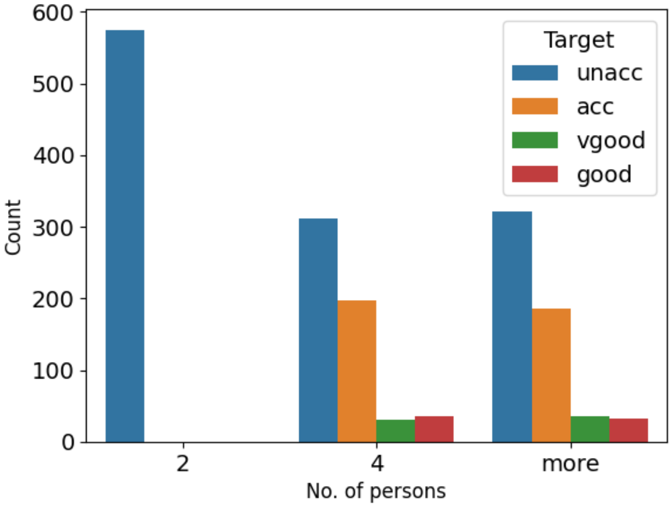


Fig. 7. Target vs No. of people

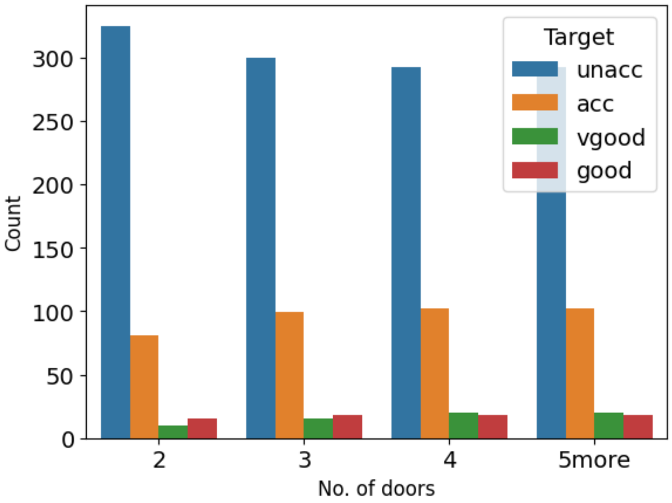


Fig. 8. Target vs No. of doors

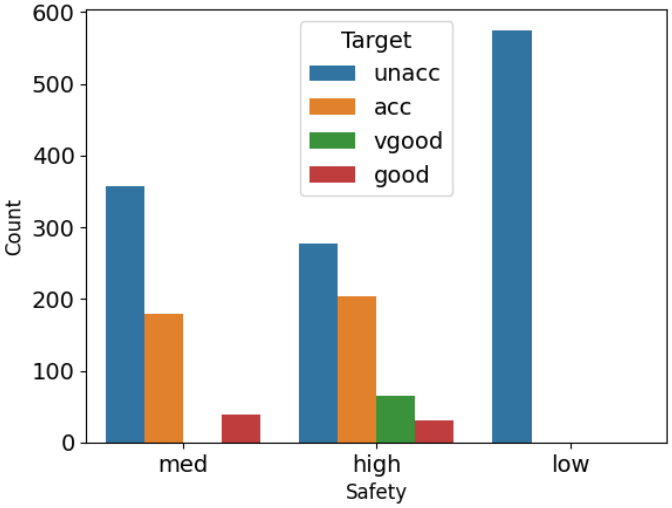


Fig. 9. Target vs Safety

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

B. Post-Processing and Feature Selection

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.

First, one-hot encoding is performed, and correlation is checked among all the features to check their association with others. It is found that for the 'Safety low' category, as expected, the target category 'unacc' has the highest positive correlation, rest categories have a negative association. For the other two categories of the Safety feature, the 'acc' and 'vgood' have a positive correlation, whereas the 'good' category remains the same. Also, a car with only a two-person capacity is highly unaccepted, resulting in a high negative correlation with 'acc,' 'good,' and 'vgood.' A car with 4 or more person capacity is 'acc' and considered good and 'vgood.' It is found that doors have a very minimal effect on the cars' acceptability since it has an association close to 0 with the target feature. Buying and maintenance significantly affect the cars' acceptability, and it was expected.

Since the categorical features in this dataset are ordinal,

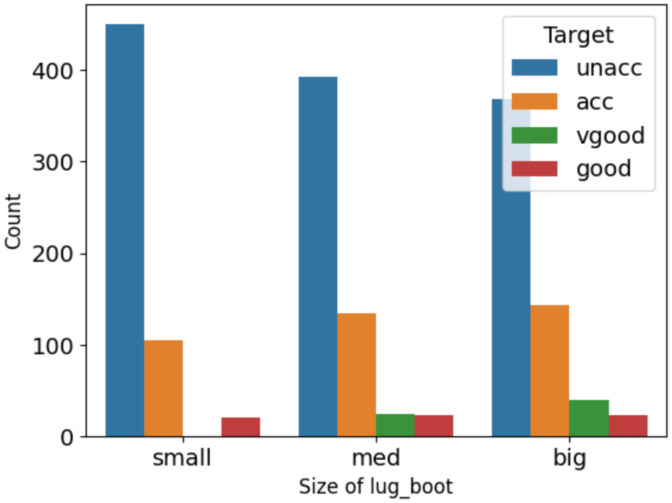


Fig. 10. Target vs Lug boot

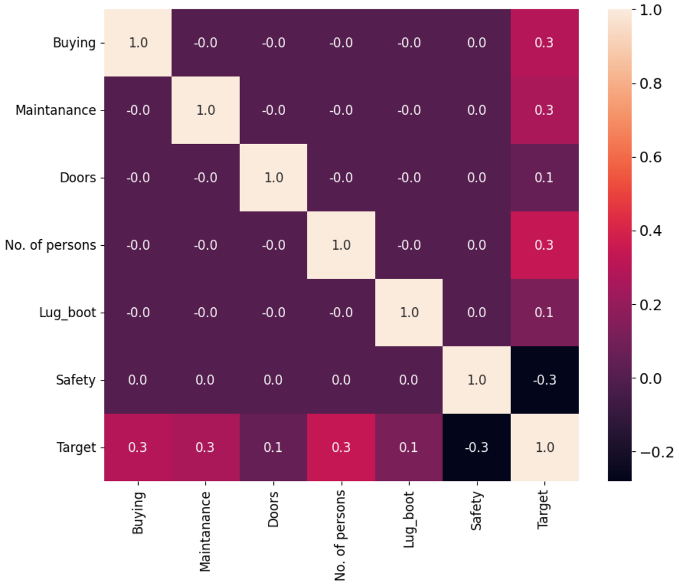


Fig. 11. Heatmap for all features

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.

Based on the above observation, feature selection was performed with the help of the mutual information score. As discussed before, it was found that doors are not significant enough to be considered in the modeling.

Subsequently, the data is divided using an 80/20 split, resulting in a final dataset with 1382 examples in the training set and 345 in the cross-validation set.

C. Random Forest Modelling

In this paper, we modeled the Random Forest Classifier. Random grid search is used to find the best parameters for the classifier 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 and check for model classification accuracy on the cross-validation dataset. Starting with the class weight, the hyperparameter is balanced as it automatically calculates the class weights according to their distribution in the training dataset. The number of jobs is kept at -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 to find the best hyperparameters by defining a range to check. The hyperparameters are:

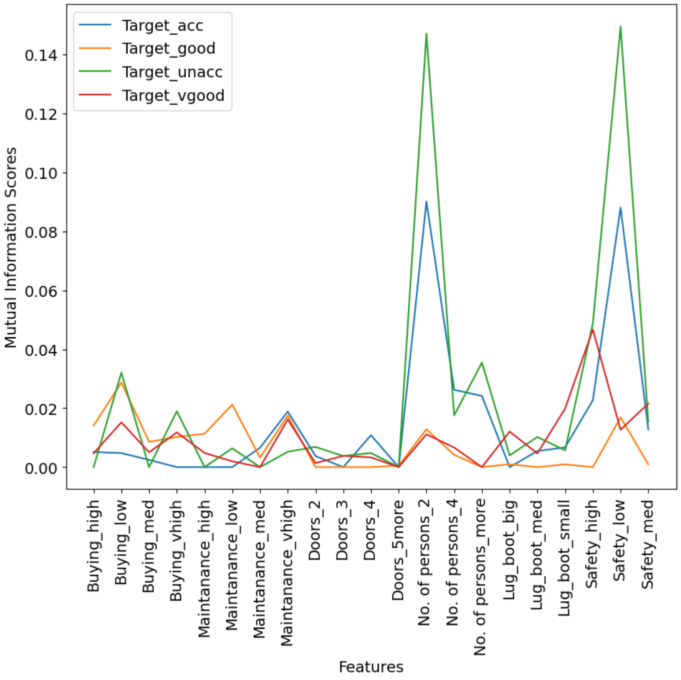


Fig. 12. Mutual Information Score for different features

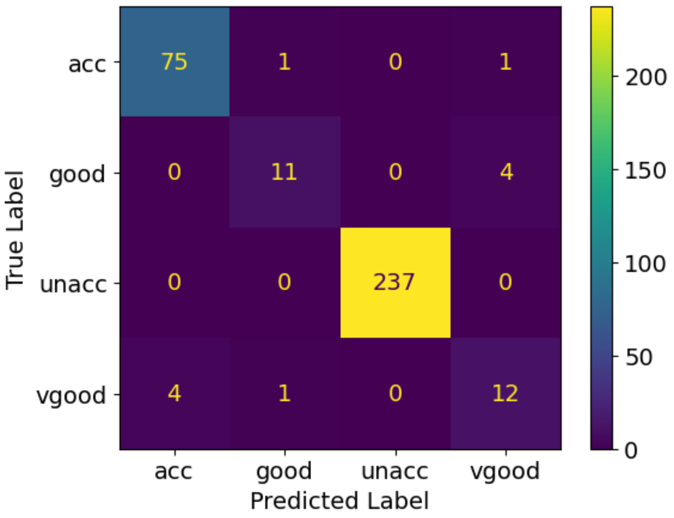


Fig. 13. Confusion Matrix after Random Search

- n estimators: Number of trees in the forest.
- Max depth: The maximum depth of the tree. If None, 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 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.
- Max features: Maximum number of features considered for splitting a node.
- Bootstrap: Method for sampling data points (with or without replacement).

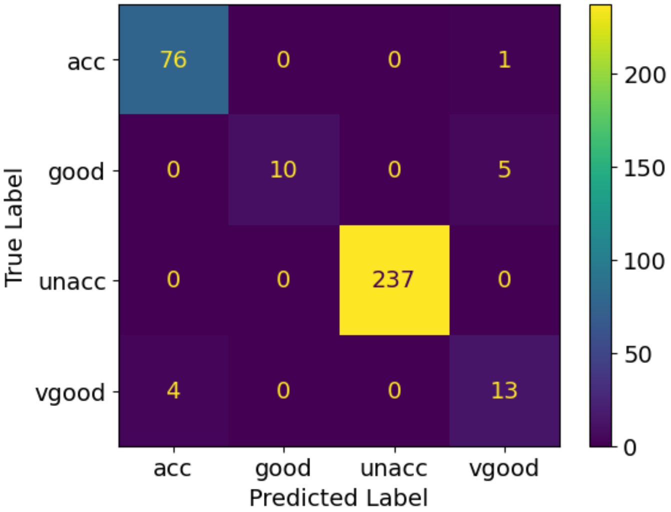


Fig. 14. Confusion Matrix for Random Forest with best features

Fitting 3 folds for each of 100 candidates, totaling 300 fits, we find the best parameters as 'n estimators': 200, 'min samples split': 3, 'min samples leaf': 1, 'max features': 'auto,' 'max depth': None, 'bootstrap': False. This generated a train set accuracy of 1 and a test accuracy 0.969. Feature importance is then checked by implementing random forests in the sklearn library. It is observed that 'safety' is most important, with 'doors' having the least. Finally, the model is trained by dropping the least important feature, but a reduction in accuracy to 0.969 in the train set and 0.933 in the test set is observed.

IV. Conclusion

After conducting a comprehensive analysis of the tree-based model in their raw form and after feature selection, it is observed that all of these variations perform well. However, it is observed that after removing the feature with a low mutual information score, the model's accuracy is reduced. It is noted that these models exhibit significantly higher accuracy on the training set, suggesting a likelihood of overfitting. Nonetheless, by employing hyperparameter search methods random search, 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 built within the ensemble, resulting in enhanced performance.

As the exploratory data analysis indicates, certain features

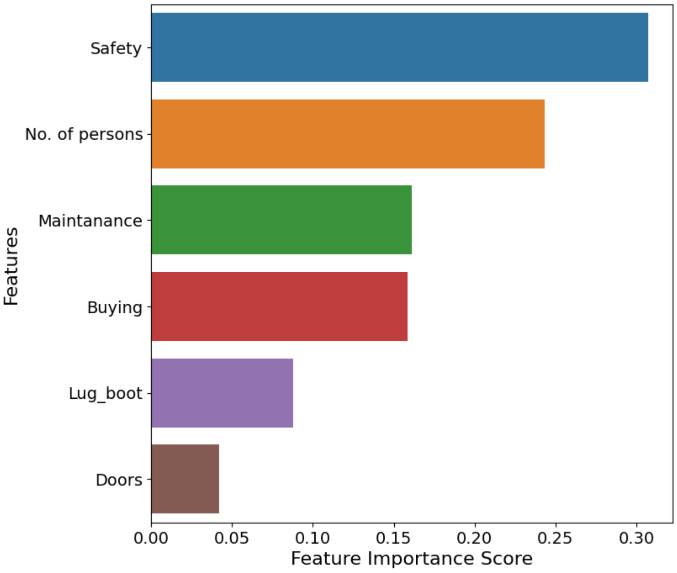


Fig. 15. Feature Importance chart (higher is better)

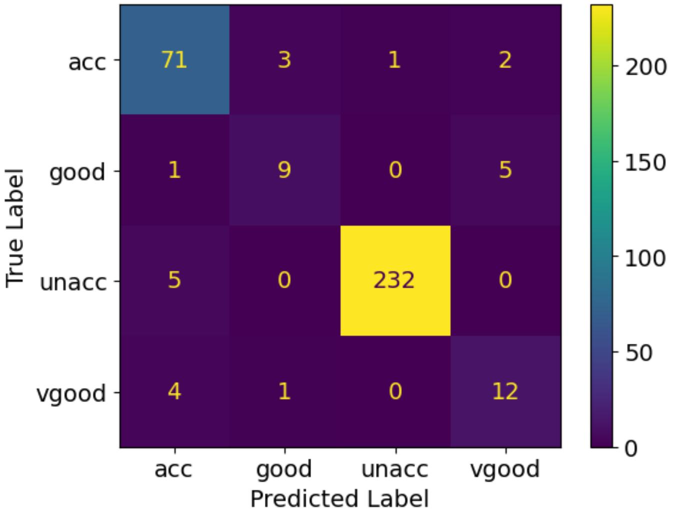


Fig. 16. Confusion Matrix for Random Forest after feature selection

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 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 better explain the target variable.

References

[1] ”Random forest,” Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Random_forest.

[2] ”Understanding Random Forest,” Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

[3] ”Random Forests,” Google Developers. [Online]. Available: https://developers.google.com/machine-learning/decision-forests/random-forests.

[4] ”Ensemble Learning: Bagging Boosting,” Towards Data Science. [Online]. Available: https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422.

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

[6] 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/.