Multi-class Classification with Decision Trees

Vaibhav Mahapatra
Data Science Department
Indian Institute of Technology, Madras
Chennai, India
me19b197@smail.iitm.ac.in

***Abstract*—Classification and regression trees (CART models in short) are defined by recursively partitioning the input space and defining a local model in each resulting region of input space. This can be represented by a decision tree, with one leaf per region. Most of the classification approaches focus on feature selection or reduction. Some features are irrelevant and redundant, resulting in a lengthy detection process and degrading the classifier's performance. The purpose of this study is to identify essential input features in building a practical car safety predictor. In this article, we aim to discover the relationship between different features and a car's safety. We apply one of the efficient Decision Tree Classifiers on datasets. Improvements: Mathematical modelling section along with visuals and improved performance of model.**

***Index Terms*—Classification, Entropy, Gini Index, features, predictors, classifier, trees.**

I. Introduction

A decision tree is a predictive modelling approach that is used in statistical machine learning. There are two types of decision trees, namely Classification and Regression Trees. A classification tree is used when the predicted outcome is the class (categorical), and a Regression tree, when the expected outcome is a continuous variable. A tree representation is used to solve the problem in which each leaf node corresponds to a class label, and features are represented on the internal node of the tree. One of the significant advantages of using the decision tree algorithm is that it can visually and explicitly represent decision-making with ease.

A decision tree is built by splitting the data set, constituting the *root node* of the tree, into subsets—which include the successor children. This splitting is based on a set of splitting rules based on predictive features. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a *node* has all the same values of the target variable or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees (*TDIDT*) is an example of a *greedy algorithm*, and it is by far the most common strategy for learning decision trees from data.

This article looks at a conditional probability model that works on data extracted from the UCI Machine Learning Repository. The key focus of our work is to classify a car based on its physical qualifications and buying price. We perform exploratory data analysis to visualise the data and see if any specific features strongly influence the target variable. Ultimately, we would like to draw insights on most relevant attributes that affect the performance of the Decision Tree classifier.

This document is structured as follows. In the next section, we provide the necessary background and definitions. Section II focuses on the data description, analysis, handling of missing values and visualization to gain more insights into the problem, while Section III demonstrates that the use of *Decision Tree Classifier*. The summary and conclusions are given in the last Section V.

II. Mathematical Modelling with Decision Trees

To explain the CART approach, consider the tree in fig. 1. This simple example elucidates how a hypothetical person would choose a job among various jobs. The first node asks if the salary is within some range. The child node asks if the job location is near home and subsequently if the job offers a cab service. If we pay attention to the logic here, we can intuitively conclude that each question partitions the domain with respect to one parameter to take a decision. Recursively, partitioned spaces are partitioned further based on some other criteria and so on.

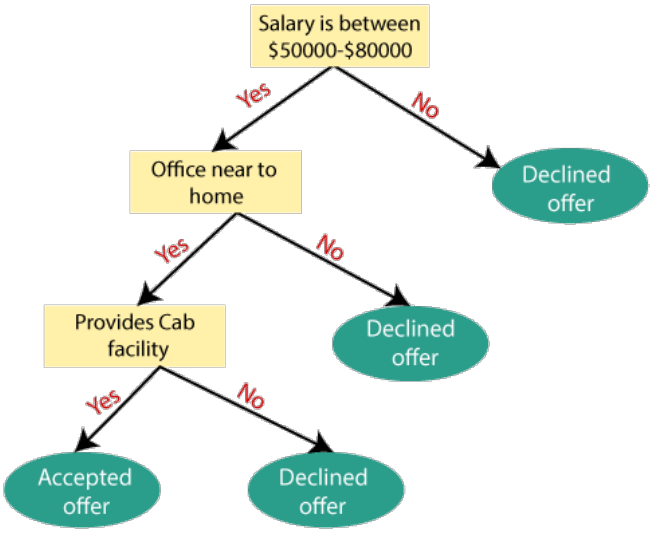

Fig. 1. A simple decision tree

In general, a decision tree can be mathematically repre-

sented as

$$f(x) = \mathbb{E}[y|x] = \sum_{m=1}^{M} w_m \mathbb{I}(x \in R_m) = \sum_{m=1}^{M} w_m \phi(x; V_m) \tag{1}$$

where R_m is the m^{th} region, w_m is the mean response in this region, and v_m encodes the choice of variable to split on, and the threshold value, on the path from the root to the m^{th} leaf. This makes it clear that a CART model is just a an adaptive basis-function model, where the basis functions define the regions, and the weights specify the response value in each region.

This model can be generalised to the classification setting by storing the distribution over class labels in each leaf instead of the mean response. This is illustrated in fig. 2. This model can be used to classify the data. For example, first, the colour of the object is checked. If it is blue, the left branch is followed and end up in a leaf labelled "4,0", which means it has 4 positive examples and 0 negative examples which match this criterion. Hence the prediction is $p(y = 1|x) = 4/4$ if x is blue. If it is red, the shape is checked: if it is an ellipse, a leaf labelled "1,1" is reached, so $p(y = 1|x) = 1/2$ is predicted. If it is red but not an ellipse, then $p(y = 1|x) = 0/2$ is predicted; If it is some other colour, the size is checked: if less than 10, then $p(y = 1|x) = 4/4$ is predicted, otherwise $p(y = 1|x) = 0/5$. These probabilities are just the empirical fraction of positive examples that satisfy each conjunction of feature values, which defines a path from the root to a leaf. In the tree in fig. 2, most of the leaves are "pure", meaning they only have examples of one class or the other, the only exception being the leaf representing red ellipses, which has a label distribution of (1, 1). We could distinguish positive from negative red ellipses by adding a further test based on size. However, it is not always desirable to construct trees that perfectly model the training data, due to overfitting.

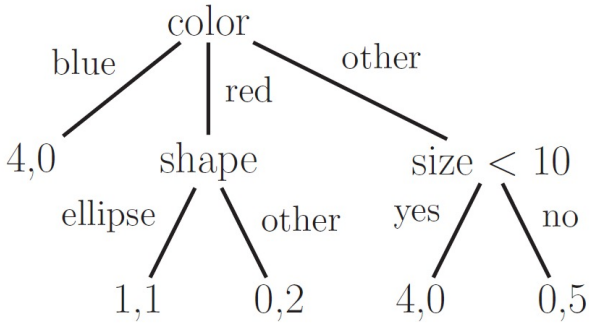

Fig. 2. A simple decision tree. A leaf labeled as (n_1, n_0) means that there are n_1 positive examples that match this path, and n_0 negative examples..

A. Growing or Learning of a tree

The split function chooses the best feature, and the best value for that feature, as follows:

$$(j^*, i^*) = arg \min_{j \in \{1,\ldots,D\}} \min_{t \in T} \text{cost}(\{X_i, y_i : x_{ij} <= t\}) + \text{cost}(X_i, y_i : x_{ij} > t) \tag{2}$$

where the cost function for a given dataset will be defined below. For simplicity, it is assumed that all inputs are real-valued or ordinal, so it makes sense to compare a feature x_{ij} to a numeric value t. The set of possible thresholds T_j for feature j can be obtained by sorting the unique values of x_{ij}. For example, if feature 1 has the values $\{4, -23, 72, -23\}$, then we set $T_1 = \{-23, 4, 72\}$. In the case of categorical inputs, the most common approach is to consider splits of the form $x_{ij} = c_k$ and $x_{ij} \neq ck$, for each possible class label c_k. Although multi-way splits (resulting in non-binary trees) are allowed, this would result in data fragmentation, meaning too little data might "fall" into each subtree, resulting in overfitting.

The function that checks if a node is worth splitting can use several stopping heuristics, such as the following:

- Is the reduction in cost too small? Typically gain of using a feature to be a normalized measure of the reduction in cost is defined as:

$$\delta \triangleq \text{cost}\left(\mathcal{D}) - (\frac{|\mathcal{D}_L|}{\mathcal{D}}\text{cost}(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{\mathcal{D}}\text{cost}(\mathcal{D}_R\right) \tag{3}$$

- Has the tree exceeded the maximum desired depth?
- Is the distribution of the response in either $\mathcal{D}_L$ or $\mathcal{D}_R$ sufficiently homogeneous (e.g., all labels are the same, so the distribution is pure)?
- is the number of examples in either $\mathcal{D}_L$ or $\mathcal{D}_R$ too small?

All that remains is to specify the cost measure used to evaluate the quality of a proposed split. This depends on whether our goal is regression or classification. Both cases are discussed below

B. Regression cost function

In regression setting cost is defined as follows:

$$\text{cost}(\mathcal{D}) = \sum_{i \in \mathcal{D}} (y_i - \bar{y}) \tag{4}$$

where $\bar{(}y) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} y_i$ is the mean of the response variable in the specified set of data. Alternatively, a linear regression model can be fitted for each leaf, using as inputs the features that were chosen on the path from the root, and then measure the residual error.

C. Classification cost function

In the classification setting, there are several ways to measure the quality of a split. First, a multinoulli model is fitted to the data in the leaf satisfying the test $X_j < t$ by estimating the class-conditional probabilities as follows:

$$\hat{\pi}_c = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i = c) \tag{5}$$

where $\mathcal{D}$ is the data in the leaf. Given this, there are several common error measures for evaluating a proposed partition:

- **Misclassification rate**. The most probable class label is defined as $\hat{y}_c = arg\max_c \hat{\pi}_c$. The corresponding error rate is then

$$\frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}}\mathbb{I}(y_i \neq \hat{y}) = 1 - \hat{\pi}_y \tag{6}$$

- **Entropy**, or **deviance**:

$$\mathbb{H}(\hat{\pi}) = -\sum_{c=1}^{C}\hat{\pi}_c \log \hat{\pi}_c \tag{7}$$

One important think to note is that minimizing the entropy is equivalent to maximizing the information gain between test $X_j < t$ and the class label Y, defined by

$$\text{infoGain}(X_j < t, Y) \triangleq \mathbb{H}(Y) - \mathbb{H}(Y|X_j < t) \tag{8}$$

$$= \left(\sum_c p(y=c)\log p(y=c)\right) + \left(\sum_c p(y=c|X_j<t)\log p(c|X_j<t)\right) \tag{9}$$

since $\hat{\pi}_c$ is an MLE for the distribution $p(c|X_j < t)$.

- **Gini Index**

$$\sum_{c=1}^{C}\hat{\pi}_c(1-\hat{\pi}_c) = \sum_c \hat{\pi}_c - \sum_c \hat{\pi}_c^2 = 1 - \sum_c \hat{\pi}_c^2 \tag{10}$$

This is the expected error rate. To see this, $\hat{\pi}_c$ is the probability a random entry in the leaf belongs to class c, and $1 - \hat{\pi}_c$ is the probability it would be mis-classified.

In the two-class case, where $p = \pi_m(1)$, the misclassification rate is $1 - max(p, 1-p)$, the entropy is $\mathbb{H}_2(p)$, and the Gini index is $2p(1-p)$. It is observed that the cross-entropy and Gini measures are very similar, and are more sensitive to changes in class probability than is the misclassification rate. For example, consider a two-class problem with 400 cases in each class. Suppose one split created the nodes (300,100) and (100,300), while the other created the nodes (200,400) and (200,0). Both splits produce a misclassification rate of 0.25. However, the latter seems preferable, since one of the nodes is pure, i.e., it only contains one class. The cross-entropy and Gini measures will favor this latter choice [2].

D. Pruning of a Decision Tree

To prevent overfitting, the tree's growth can be stopped if the decrease in the error is not sufficient to justify the extra complexity of adding an extra subtree. Therefore, the standard approach is to grow a "full" tree and then perform pruning. This can be done using a scheme that prunes the branches giving the most minuscule increase in the error. To determine how far to prune back, the cross-validation error can be evaluated on each subtree, and then pick the tree whose CV error is within 1 standard error of the minimum. The point with the minimum CV error corresponds to the simple tree.

E. Advantages and Disadvantages of CART method

CART models are popular for several reasons: they are easy to interpret, they can easily handle mixed discrete and continuous inputs, they are insensitive to monotone transformations of the inputs (because the split points are based on ranking the data points), they perform automatic variable selection, they are relatively robust to outliers, they scale well to large data sets, and they can be modified to handle missing inputs. However, CART models also have some disadvantages. The primary one is that they do not predict very accurately compared to other kinds of models. This mainly due to the greedy nature of the tree construction algorithm. A related problem is that trees are unstable: minor changes to the input data can significantly affect the tree's structure due to the hierarchical nature of the tree-growing process, causing errors at the top to affect the rest of the tree. In frequentist terminology, it can be said that trees are high variance estimators [2]. Popular approaches to deal with these disadvantages is to use trees in either a bagging or a boosting manner. These methods cleverly leverage the high variance characteristics to fit models that generalise better.

When comparing CART method with Logistic Regression for classification, we can see the following pros:

- Decision Trees bisect the space into smaller and smaller regions, whereas Logistic Regression fits a single line to divide the space exactly into two. Of course for higher-dimensional data, these lines would generalize to planes and hyperplanes. A single linear boundary can sometimes be limiting for Logistic Regression.
- A decision tree is much more interpretable than logistic regression, when there are various variables involved.
- They usually fit the data much better than LR, even tend to overfit. This can be changed by pruning the tree.
- They are immune to null values and outliers.

Despite all these pros, decision trees are harder to hyperparameter tune. They are also more prone to overfit as they have high variance and low bias.

III. Decision Tree on the Cars-dataset

A. Data Description

The data-set comprises information about cars safety divided into four categories based on the car's condition. The categories are **unacc**, **acc**, **good** and **vgood** indicating the increased safety with **uacc** being the least safe and **vgood** being the safest. Information about the car's buying price, maintenance price, number of doors, the person carrying capacity, size of luggage boot and estimated safety. The aim is to devise a generalised decision tree model that can determine the car's safety rating. Once the data is imported, it is highlighted that there are no missing values in the columns. From Fig. 4 it is observed that most cars are classified as **unacc**, and only a few cars live up to the safety standards.

B. Exploratory Data Analysis

The following section will focus on establishing relationships between different features and identify the most suited

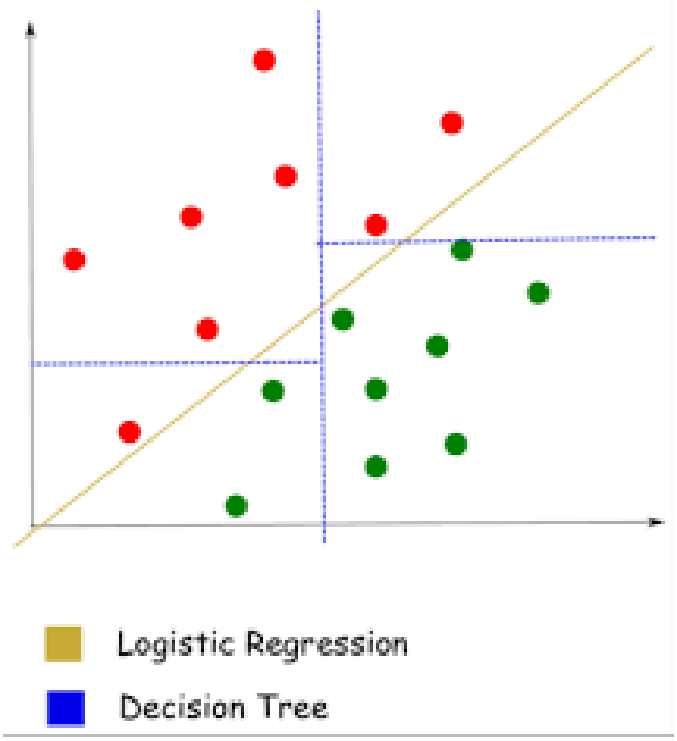

Fig. 3. Sample case of Decision Tree vs Logisitic Regression

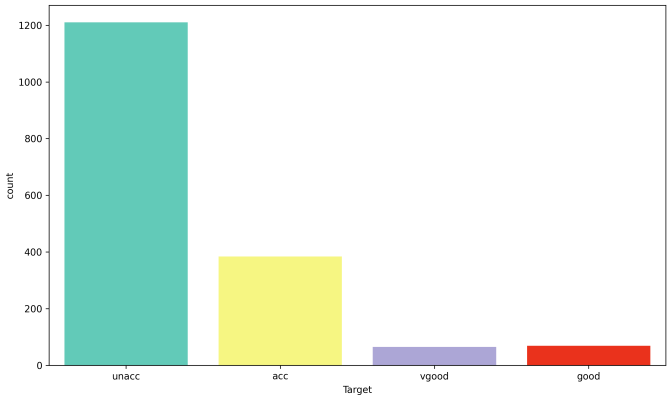

Fig. 4. Count-plot of Target Variable

ones for modelling purposes. The number of cars from each category identify the relationship between safety and given variables.

From Fig. 5 it is observed that the maximum numbers of cars bought fall in **unacc** category irrespective of their buying price. In contrast, surprisingly, the vehicles that are most up to the safety standards are not the expensive ones but are in the low and medium price range. A similar pattern is observed with respect to the cost of maintenance (as seen in Fig. 6).

The variable **doors** conveys information about the number of doors in the car. It is a common notion that 2 door cars are expensive and therefore have to safer than the other ones. From the Fig. 7 it is seen that all the cars are equally safe irrespective

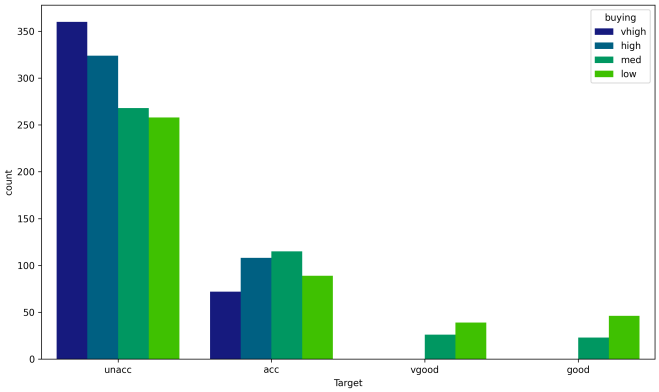

Fig. 5. Target variable vs Buying price

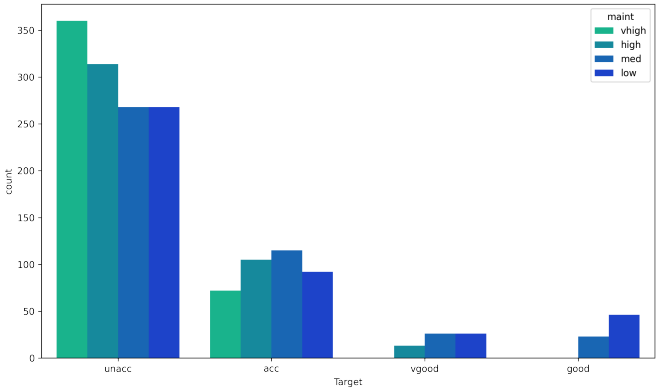

Fig. 6. Target variable vs Maintenance cost

of their price and maintenance cost. Safety standards are distributed equally across all the four door categories.

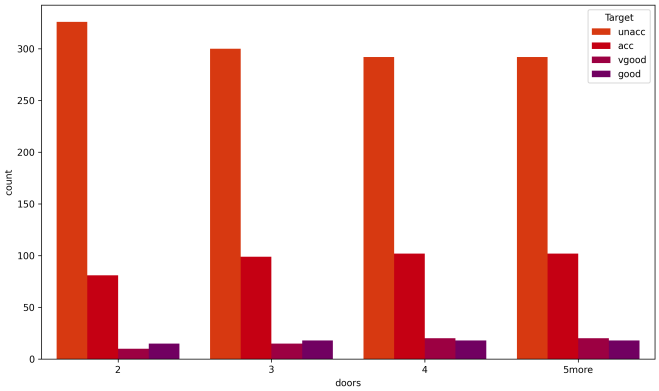

Fig. 7. Target variable vs No. of Doors

On the other hand, the more the safety standards improve as the carrying capacity increases, and the 4 seater cars are better in terms of safety than others. A slight reduction is also observed as the number of seats go beyond 4 as seen in 8, whereas the space for luggage boot is evenly distributed across all the four target variable categories.

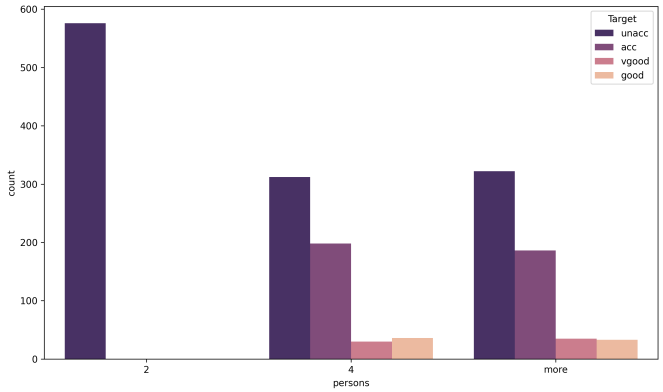

Fig. 8. Target variable vs No. of persons

The safety variable indicates that the cars with lower safety standards fall into the category of **unacc**. At the same time, it's almost distributed evenly across all the categories for higher safety ratings as seen in Fig. 9

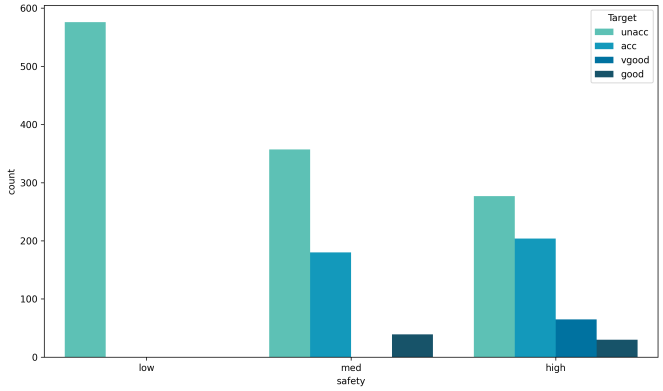

Fig. 9. Target variable vs Safety carrying capacity

C. Model Building and Evaluation

This section will focus on the relationship between different features and identify the most suited for modelling purposes. The purpose of this paper is to predict the safety of car and to identify its relationship with the other features. To do so, a CART model is fitted, which aims at learning the distribution of target classes. The mathematics behind the splits at each node is described in Sec. II.

The python package *sklearn* is used for performing the task of fitting a Decision Tree. Scikit-learn (Sklearn) is one of the most valuable and robust libraries for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling, including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is primarily written in Python, is built upon NumPy, SciPy and Matplotlib. The sklearn package provides different classes for the classification problem, including **DecisionTreeClassifier**. Before proceeding with the modelling, all categorical features are encoded. Once that is done the data is splited in train and test data. Afterwards a Decision tree classifier is fitted on the training data and the labels for test data is predicted.

The mean accuracy score on the test data obtained by both **entropy** and **Gini** criteria using Decision tree is $\mathbf{0.8719}$, whereas that on the training dataset is $\mathbf{0.8025}$. The majority of cars lies in the category of *unacc*. The **null accuracy** obtained is $\mathbf{0.6792}$, which further indicates that the classifier is doing a decent job. The confusion matrix looks as follows:

$$\begin{bmatrix} 65 & 0 & 18 & 0 \\ 11 & 0 & 0 & 0 \\ 17 & 0 & 218 & 0 \\ 17 & 0 & 0 & 0 \end{bmatrix}$$

The classification report suggest the following

	precision	recall	f1-score	support
acc	0.59	0.78	0.67	83
good	0.00	0.00	0.00	11
unacc	0.92	0.93	0.93	235
vgood	0.00	0.00	0.00	0.00
accuracy			0.82	346

The following graph shows the splits made by the decision tree classifier to obtain the results stated above. It is observed that safety is the most essential criterion followed by the person carrying capacity and maintenance cost.

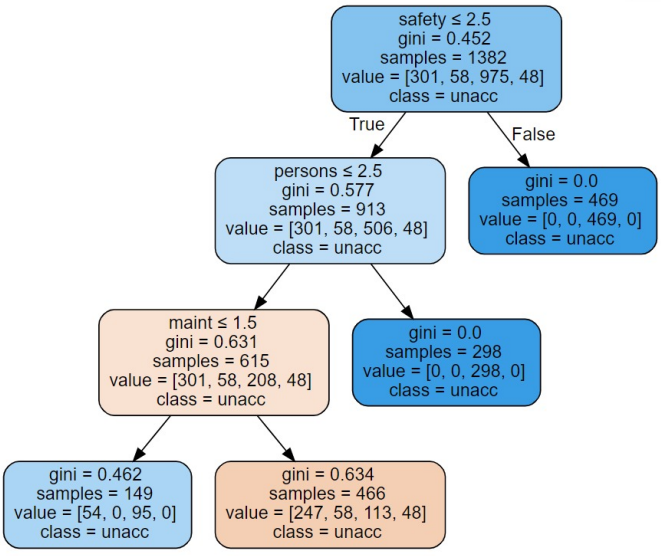

Fig. 10. Decision Tree

IV. Improvements on the Base Model

Further in this paper the RandomSearchCV algorithm is utilised to find the best suitable hyperparameters for the decision tree classifier. Hyperparameters are the variables that the user usually specify while building the Machine Learning model. Thus, hyperparameters are set before specifying the parameters, or we can say that hyperparameters are used

to evaluate optimal parameters of the model. the best part about hyperparameters is that their values are decided by the user who is building the model. Random Search uses a different combination of all the specified hyperparameters and their values, calculates each variety's performance, and selects the best value for the hyperparameters. This makes the processing time-consuming and expensive based on the number of hyperparameters involved.

Classification Report of the training data:

	precision	recall	f1-score	support
acc	0.71	0.90	0.79	301
good	0.00	0.00	0.00	58
unacc	0.97	0.93	0.95	975
vgood	0.53	0.77	0.63	48
accuracy			0.88	1382
macro avg	0.55	0.65	0.59	1382
weighted avg	0.86	0.88	0.86	1382

Classification Report of the test data:

	precision	recall	f1-score	support
acc	0.74	0.76	0.75	83
good	0.00	0.00	0.00	11
unacc	0.94	0.94	0.94	235
vgood	0.58	0.88	0.70	17
accuracy			0.86	346
macro avg	0.56	0.65	0.60	346
weighted avg	0.84	0.86	0.85	346

Fig. 11. Classification Report for Train and Test Dataset

Along with the Search, cross-validation is also performed. The most popular type of Cross-validation is K-fold Cross-Validation. It is an iterative process that divides the train data into k partitions. Each iteration keeps one partition for testing and the remaining k-1 partitions for training the model. The next iteration will set the next partition as test data and the remaining k-1 as train data, and so on. In each iteration, it will record the model's performance and, in the end, give the average of all the arrangements. After applying RandomSearchCV the following hyperparameters were obtained:

- max_depth = 7.0
- max_features = 5
- min_samples_leaf = 8
- min_samples_split = 70

The **entropy** criteria is used to build a decision tree classifier and the accuracy attained is **0.90** for the training set and **0.92** for the test set. The classification report for the train and test set is shown in the Fig. 11.

The confusion matrix is obtained as follows:

$$\begin{bmatrix} 71 & 0 & 7 & 3 \\ 2 & 7 & 0 & 5 \\ 10 & 0 & 230 & 0 \\ 2 & 0 & 0 & 18 \end{bmatrix}$$

The graph in Fig. 12 shows the splits made by the decision tree classifier to obtain the results stated above. We can observe that the decision tree is faily complex and thanks to K-fold Cross Validation, it isn't overfitting on the data (because training accuracy is somewhat equal to test accuracy)

V. Conclusion

The task of *classifying the cars based on their safety* is essential to understand the consumer behaviour while they purchase a vehicle. The given data shows that the car designed for 4 or more people tends to be safer than the rest. Similarly, the maintenance cost for 2 seater cars are higher (that do match the real world where 2 seater cars are considered a luxury), but despite that, they have lower safety standards. The other variables like luggage space and the number of doors don't influence the safety rating much. The EDA and Decision tree classifier approach gives some decent idea about the relationship between predictors and regressors. It doesn't explain the underlying pattern thoroughly and doesn't predict very well for the cars which can fall in *vgood* and *good* category due to lack of data.

References

[1] Murphy, Kevin P. Machine Learning: A Probabilistic Perspective. Cambridge, MA: MIT Press, 2012. pp.544–552

[2] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013, pp.311–315

[3] https://towardsdatascience.com/understanding-decision-trees-for-classification-python-9663d683c952

[4] https://www.datacamp.com/tutorial/decision-tree-classification-python

[5] https://www.section.io/engineering-education/how-to-implement-k-fold-cross-validation/

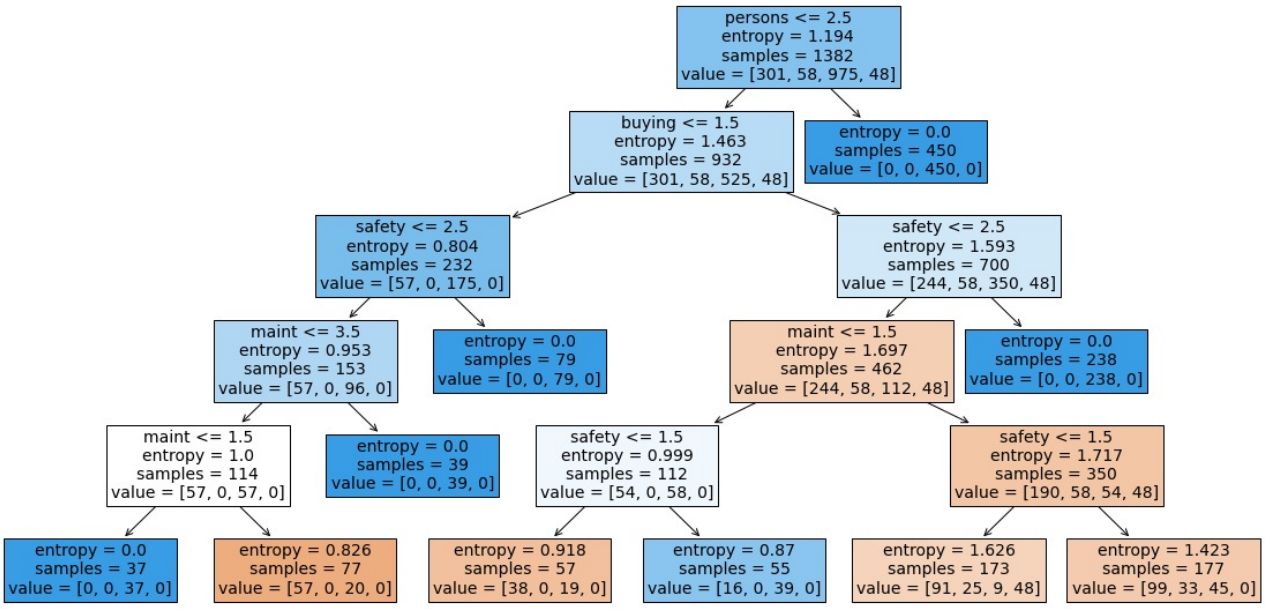

Fig. 12. Final Decision Tree post Hyperparameter tuning