Ensemble Method for Classification: Random Forest

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

***Abstract*—Classification and regression trees or CART models, also called decision trees, 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 tree, with one leaf per region. A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms. This algorithm is applied in various industries such as banking and e-commerce to predict behavior and outcomes. This article provides an overview of the random forest algorithm and how it works. The article will present the algorithm's features and how it is employed in real-life applications. In this paper, we aim to discover the relationship between different features and a car's safety. We apply one of the efficient Random Forest Classifiers on the given dataset. Changes made- added a new subsection in the *Mathematical Foundation* section, alongwith more visuals across the paper. Accuracy of our model is also improved marginally.**

***Index Terms*—Classification, Decision Tree(s), features, predictors, classifier, ensemble, bagging boosting.**

I. Introduction

A random forest is a machine learning technique for solving regression and classification problems. It makes use of ensemble learning, which is a technique that combines many models (here, classifiers) to solve complex problems. A random forest algorithm is made up of a large number of decision trees. The random forest algorithm's 'forest' is trained using bagging or bootstrap aggregation. Bagging is a meta-algorithm ensemble that improves the accuracy of machine learning algorithms. The outcome is determined by the (random forest) algorithm based on the predictions of the decision trees. It predicts by taking the mode or averaging the output of various trees. Increasing the number of trees increases the precision of the outcome. A random forest eradicates the limitations of a decision tree algorithm. It efficiently tackles over-fitting on data and increase general precision. It generates predictions without requiring many configurations in packages (like scikit-learn). Features of a Random Forest Algorithm include:

- More accuracy than the decision tree algorithm.
- More effective ways of handling missing data.
- Produces reasonable predictions without extensive hyper-parameter tuning.
- Little to none over-fitting, compared to single decision trees.
- In every random forest tree, a subset of features is selected randomly at the node's splitting point.

Random forest classification uses an ensemble methodology to achieve the desired result. The training data is fed into various decision trees for training. This dataset contains observations and features that will be chosen at random during node splitting. A rain forest system is supported by a number of decision trees. Every decision tree is made up of nodes that represent decisions, leaf nodes, and a root node. The leaf node of each tree represents the decision tree's final output. The final output is chosen using a majority-voting system (or mode). In this case, the output chosen by the majority of decision trees becomes the random forest system's final output. Regression is the other task performed by a random forest algorithm. A random forest regression follows the concept of simple regression. Values of dependent (features) and independent variables are passed in the random forest model. Each tree produces a specific prediction. The mean prediction of the individual trees is the output of the regression. This is contrary to random forest classification, whose output is determined by the mode of the decision trees' class. Although random forest regression and linear regression follow the same concept, they differ in terms of functions [4].

This document 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 visualise and analyse the data to see if any specific features strongly influence the target variable. Ultimately, we would like to understand the data characteristics which affect the performance of Random Forest classifier.

This paper is structured as follows. First, 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 *Random Forest Classifier* on the real-world dataset. A summary and conclusions are given in the end in Section IV.

II. Mathematical Foundation behind Random Forest

Bagging or bootstrap aggregation is a technique for reducing the variance of an estimated prediction function. Bagging seems to work incredibly well for high-variance, low-bias procedures, such as trees. For regression, the same regression tree is simply fitted many times to bootstrap sampled versions of the training data and average the result. For classification, a committee of trees each cast a vote for the predicted class. Random forests [1] is a substantial modification of bagging

that builds an extensive collection of de-correlated trees and then averages them. On many problems, the performance of random forests is very similar to boosting, and they are simpler to train and tune. As a consequence, random forests are widespread and are implemented in a variety of packages.

Bagging's objective is to average many noisy but approximately unbiased models, which hence reduces the variance. Trees are ideal candidates for bagging, since they can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias. Since trees are notoriously noisy, they benefit greatly from the averaging. Moreover, since each tree generated in bagging is identically distributed, the expectation of an average of B such trees is the same as the expectation of any one of them. This means the bias of bagged trees is the same as that of the individual trees, and the only hope of improvement is through variance reduction. This is in contrast to boosting, where the trees are grown in an adaptive way to remove bias, and hence are not independent. An average of B i.i.d. random variables, each with variance σ^2, has variance $\frac{1}{B}\sigma^2$. If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, the variance of the average is

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{1}$$

As B increases, the second term disappears, but the first remains, and hence the size of the correlation of pairs of bagged trees limits the benefits of averaging. The idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much. This is achieved in the tree-growing process through random selection of the input variables. After B such trees $\{T(x;\Theta b)\}_1^B$ are grown, the random forest (regression) predictor is

$$\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T(x;\Theta_b) \tag{2}$$

Θ_b characterizes the b^{th} random forest tree in terms of split variables, cut-points at each node, and terminal-node values.

A. In-depth analysis of Random Forests

In this section we analyze the mechanisms at play with the randomization employed by random forests. For this discussion we focus on regression and squared error loss, since this gets at the main points, and bias and variance are more complex with 0–1 loss. Furthermore, even in the case of a classification problem, we can consider the random-forest average as an estimate of the class posterior probabilities, for which bias and variance are appropriate descriptors.

The limiting form ($B \longrightarrow \infty$) of the random forest regression estimator is

$$\hat{f}_{rf}(x) = \mathrm{E}_{\Theta|\mathrm{Z}} T(x;\Theta(\mathrm{Z})) \tag{3}$$

where dependence has been explicated on the training data Z. Here the estimation is considered at a single target point x. From (1) it can be seen that

$$\mathrm{Var}\hat{f}_{rf}(x) = \rho(x)\sigma^2(x) \tag{4}$$

Here

- $\rho(x)$ is the sampling correlation between any pair of trees used in the averaging:

$$\rho(x) = \mathrm{corr}[T(x;\Theta_1(\mathrm{Z})), T(x;\Theta_2(\mathrm{Z}))] \tag{5}$$

- σ^2(x) is the sampling variance of any single randomly drawn tree,

$$\sigma^2(x) = \mathrm{Var}T(x;\Theta(\mathrm{Z})) \tag{6}$$

It is easy to confuse $\rho(x)$ with the average correlation between fitted trees in a given random-forest ensemble; that is, think of the fitted trees as *N*-vectors, and compute the average pairwise correlation between these vectors, conditioned on the data. This is not the case; this conditional correlation is not directly relevant in the averaging process, and the dependence on x in $\rho(x)$ warns us of the distinction. Rather, $\rho(x)$ is the theoretical correlation between a pair of random-forest trees evaluated at x, induced by repeatedly making training sample draws **Z** from the population, and then drawing a pair of random forest trees. In statistical jargon, this is the correlation induced by the sampling distribution of **Z** and Θ.

More precisely, the variability averaged over in the calculations in (5) and (6) is both

- conditional on **Z**: due to the bootstrap sampling and feature sampling at each split, and
- a result of the sampling variability of **Z** itself.

In fact, the conditional covariance of a pair of tree fits at x is zero, because the bootstrap and feature sampling is i.i.d.

1) Bias: As in bagging, the bias of a random forest is the same as the bias of any of the individual sampled trees $T(x;\Theta(\mathrm{Z}))$:

$$\begin{aligned}\mathrm{Bias}(x) &= \mu(x) - \mathrm{E}_{\mathrm{Z}}\hat{f}_{rf}(x) &(7)\\ &= \mu(x) - \mathrm{E}_{\mathrm{Z}}\mathrm{E}_{\Theta|\mathrm{Z}} T(x;\Theta|\mathrm{Z}) &(8)\end{aligned}$$

This is also typically greater (in absolute terms) than the bias of an unpruned tree grown to **Z**, since the randomization and reduced sample space impose restrictions. Hence the improvements in prediction obtained by bagging or random forests are *solely a result of variance reduction*.

Any discussion of bias depends on the unknown true function. Although for different models the shape and rate of the bias curves may differ, the general trend is that as m decreases, the bias increases.

2) Similarity with Nearest Neighbors: The random forest classifier has much in common with the k-nearest neighbor classifier; in fact a weighted version thereof. Since each tree is grown to maximal size, for a particular $\Theta^*, T(x;\Theta^*\mathrm{Z}))$ is the response value for one of the training samples. The tree-growing algorithm finds an "optimal" path to that observation,

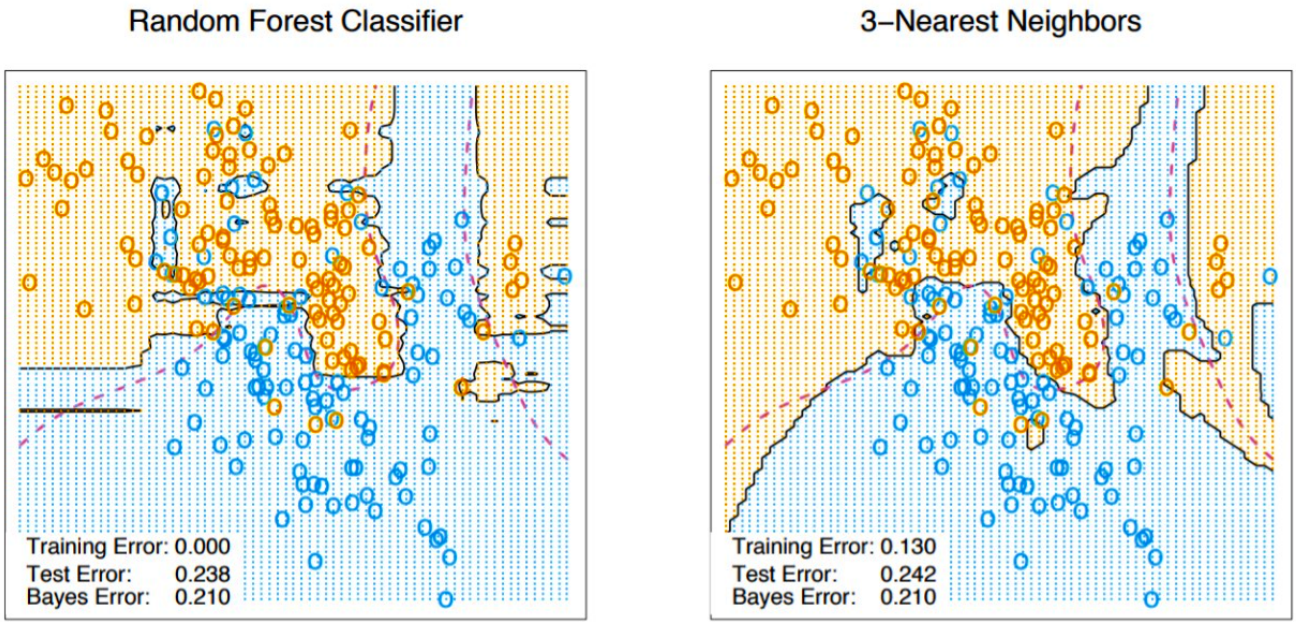

Fig. 1. Random forests versus 3-NN. The axis-oriented nature of the individual trees in a random forest lead induce decision regions with a similar flavor

choosing the most informative predictors from those at its disposal. The averaging process assigns weights to these training responses, which ultimately vote for the prediction. Hence via the random-forest voting mechanism, those observations close to the target point get assigned weights—an equivalent kernel—which combine to form the classification decision. Fig. 1 demonstrates the similarity between the decision boundary of 3-nearest neighbors and random forests on some data [3].

3) Comparison with Logistic Regression: Logistic Regression was the first algorithm that was devised for classification tasks. Random Forest(RF), a rather new algorithm, trumps Logistic Regression on the following fronts:

- Logistic Regression can't learn non linear decision boundaries and has high bias as compared to RF which has low bias and is flexible enough to learn highly nonlinear decision boundaries. The variance in RF is also reduced due to bootstrapping and voting.
- Most of the limitations of linear regression are applied to LR, that are heteroskedasticity, serial correlation, and non-normality of error terms. All of them contribute to the standard errors of the estimated parameters.
- Can't handle missing values unlike RF which is immune to it as its underlying are decision trees. Similarly, RF are unaffected by outlier values, whereas outliers may severely affect a Logistic Regression's performance.
- In LR, features need to be scaled and normalized unlike RF which is unaffected by it.
- To build Logistic Regression models, appropriate features must be selected before fitting the model unlike RF which select features in its decision tress. Or as an alternative the Logistic Rgression model should be regularized with lasso to select the features.
- When classes are completely separable, the estimation of parameters becomes unstable in LR due to the use of logistic function which then becomes close to a Step Function and it forces the derivatives to be infinite and hence becomes computationally unstable. So, LR works when the classes are almost linearly separable but not exactly. In RF there is no problem if the classes are completely separable. Rather it helps to reduce the computations when appropriate tree pruning methods are used.

Although the above points may seem like Random Forest has the upper hand over Logistic Regression, LR does have some pros over RF:

- High interpretability compared to RF as it is a linear model.
- Lesser computation than RF.
- No hyperparamer tunning needed except for the threshold of class labels which is usually chosen as 0.5. In RF the sampling proportion of samples and features, depth of the trees, and an appropriate measure like Information Gain or Gini Index has to be chosen for splitting the independent variables.
- Less chance of over fitting where RF is prone to overfitting

For a sample case, you can see the performance of a RF model compared to a LR one in Fig. 2.

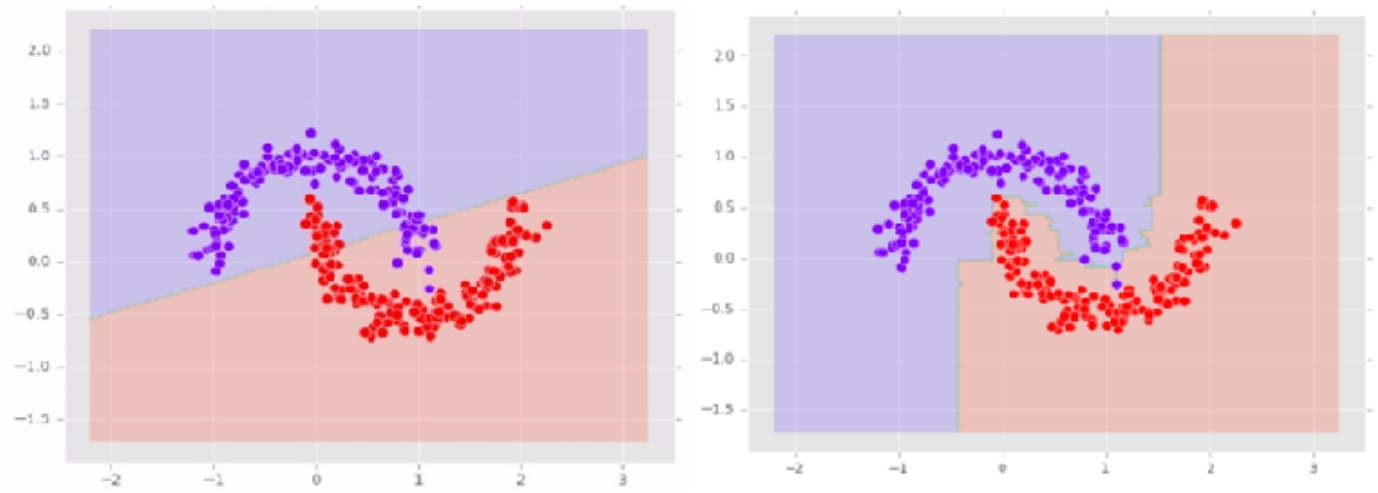

Fig. 2. Logistic Regression vs Random Forest

III. Real-life application of Random Forest

A. Data Analysis and Feature Engineering

The dataset 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 analysis aims 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 no are missing values in the columns. From Fig. 3 it is observed that most cars are classified as **unacc**, and only a few cars live up to the safety standards.

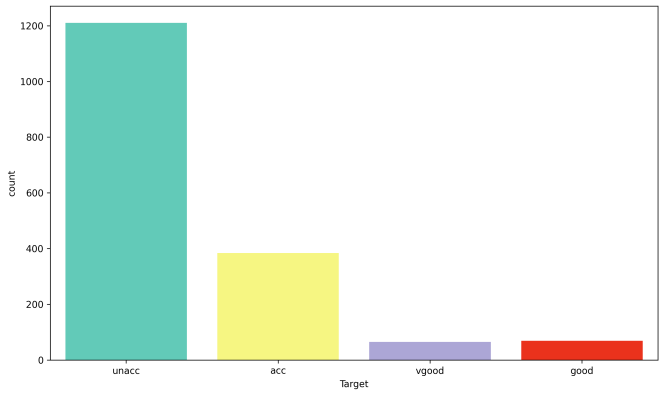

Fig. 3. Frequency of each target variable

We will now focus on establishing relationships between different features and identifying the most suited for modelling purposes.

From Fig. 4 it is evident 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

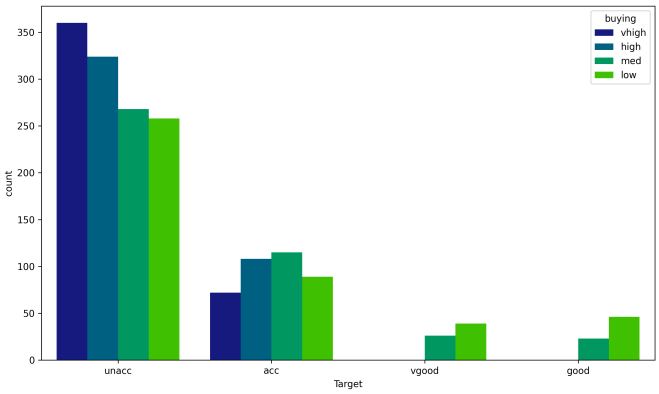

Fig. 4. Target variable vs buying price

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. 5).

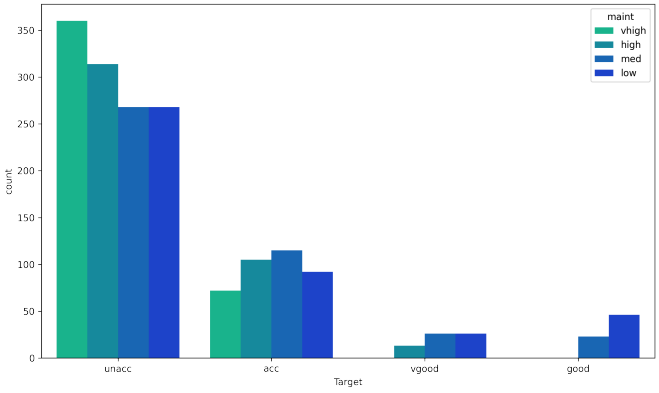

Fig. 5. Target variable vs maintenance cost

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, but from the Fig. 6 it is observed that all the cars are equally safe irrespective of the number of doors present.

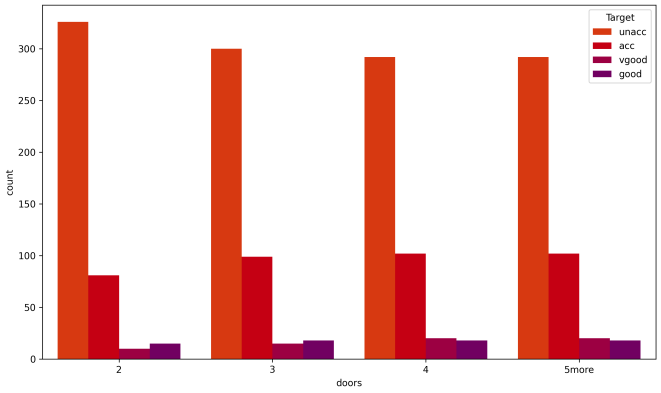

Fig. 6. Target variable vs No. of doors

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 7, whereas the space for luggage boot is evenly distributed across all the four target variable categories.

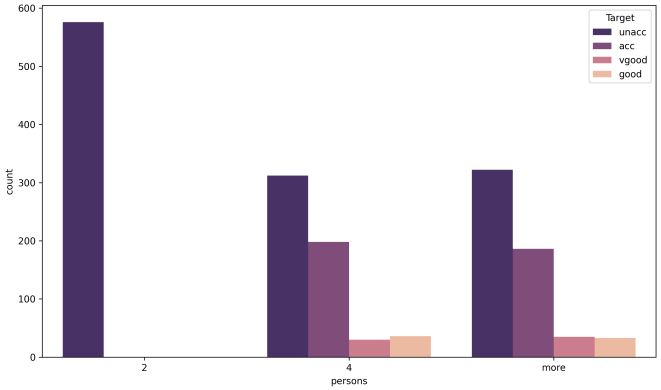

Fig. 7. Target variable vs person carrying capacity

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. 8

B. Random Forest Classifier Model and Results

The purpose of this paper is to predict the safety of car and to identify its relationship with the other features. To do so, several generative models are fitted (which aims at learning the distribution of target classes rather than just identifying the decision boundary). The mathematics behind the estimation of the best fit is described in detail in Sec. II.

The python package *sklearn* is used for performing the task of fitting a Random Forest classifier. Scikit-learn (Sklearn) is one of the most valuable and robust libraries for machine

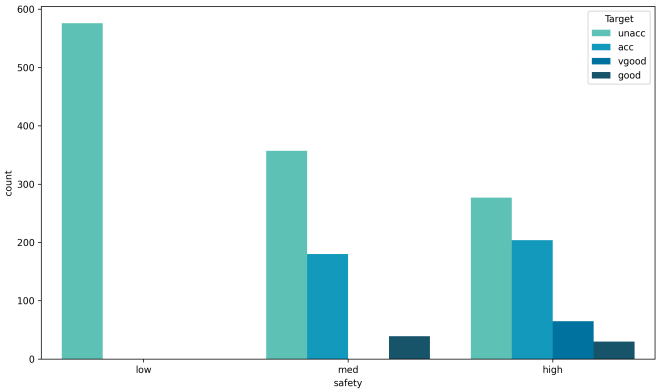

Fig. 8. Target variable vs safety carrying capacity

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 **RandomForestClassifier**. Before proceeding with the modelling, all categorical features are encoded. Once that is done the data is splited in train and test data. Afterwards Random Forest 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 using 10 decision trees and 100 decision trees to generate a Random Forest Classifier turn out to be $\mathbf{0.9583}$, whereas that on the training dataset is $\mathbf{1.000}$, which indicates that the model is overfitting. 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:

$$\begin{bmatrix} 74 & 6 & 3 & 0 \\ 0 & 10 & 0 & 1 \\ 1 & 0 & 234 & 0 \\ 2 & 0 & 0 & 15 \end{bmatrix}$$

The classification report suggest the following

	precision	recall	f1-score	support
acc	0.96	0.89	0.92	83
good	0.62	0.91	0.74	11
unacc	0.99	1.00	0.99	235
vgood	0.94	0.88	0.91	17
accuracy			0.96	346

The following graph shows the importance imparted by the Random Forest classifier with 100 decision trees to the features. It is observed that the most essential features are **safety** and **persons**, whereas the number of **doors** in the car doesn't affect its safety rating by much.

As seen in the Fig. 9 it is observed that the value associated with **doors** is very small (0.066137). Therefore, another Ran-

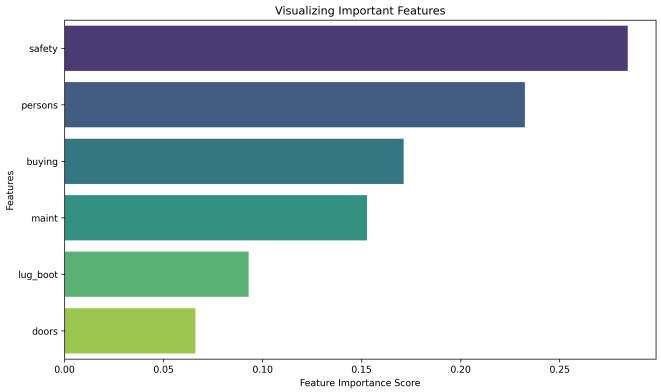

Fig. 9. Important Features

dom Forest Classifier is constructed where the doors feature is excluded. The test accuracy turned out to be **0.9463**, and train accuracy is **0.9581**. The current model addresses the overfitting problem that arises in the earlier model. Therefore, the model performs better when the number of doors is not included in decision making. The confusion matrix for this is

$$\begin{bmatrix} 69 & 9 & 1 & 1 \\ 0 & 11 & 0 & 1 \\ 0 & 0 & 238 & 0 \\ 2 & 3 & 0 & 14 \end{bmatrix}$$

The classification report suggest the following

	precision	recall	f1-score	support
acc	0.96	0.81	0.88	83
good	0.42	0.82	0.53	11
unacc	0.99	1.00	0.99	235
vgood	0.81	0.71	0.73	17
accuracy			0.96	348

IV. 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 is higher (intuitively matches the real world wscenario as 2-seaters are typically luxury cars), but they have lower safety standards. The other variables like luggage space and the number of doors don't influence the safety rating much and should not be considered for modelling purposes. The EDA and Random Forest classifier approach give a good idea about the relationship between predictors and regressors. It explains the underlying pattern thoroughly and predicts very well for the cars which can fall in *vgood* and *good* category even after the availability of significantly less data. Future work will focus on finding the most needed features for modelling purposes.

References

[1] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
[2] Murphy, Kevin P. Machine Learning: A Probabilistic Perspective. Cambridge, MA: MIT Press, 2012. pp.548–552
[3] Random Forest, Department of Mathematics, McGill University, https://www.math.mcgill.ca/yyang/resources/doc/randomforest.pdf
[4] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer, 2013, pp.316–321
[5] https://towardsdatascience.com/a-practical-guide-to-implementing-a-random-forest-classifier-in-python-979988d8a263