Building a Predictive Model Using Random Forests

Swathi V
Dept. of Engineering Design
Indian Institute of Technology Madras
Chennai, India
ed18b034@smail.iitm.ac.in

***Abstract*—Random forests are predictive algorithms that have been popularised because of their ability to work with very high dimensional feature spaces efficiently. In this article, we explore the mathematical framework behind random forest models, and how they can be used to optimize the model. We also discuss various metrics to quantify their performance, and discuss how it can be improved.**

***Index Terms*—Random forests, ensemble models, decision trees, bagging, car evaluation dataset**

I. Introduction

Machine learning tasks are becoming increasingly more complex with applications in rapidly progressing fields such as computer vision, bio-informatics, medical diagnosis, etc. With such elaborate and intricate tasks, the data used for training these models is also increasing manifold in terms of number of features. However, due to resource limitations, it is not practical to increase the number of samples proportionally. Thus, machine learning algorithms which can work efficiently with data containing very high-dimensional feature spaces but small number of samples are becoming increasingly more popular. The random forest algorithm, devised by Brieman in the early 2000s, is one technique popularly used for this purpose.

Random forests are **ensemble models** that can be applied to both regression and classification tasks, very much like **decision trees**, their building blocks. Though random forests are very powerful and high performing models, the key principle behind this approach is building many *weak uncorrelated decision trees* using the training data. For regression models, the *average* of the output of these regression trees is then used to predict the output variable, whereas for classification models, the *majority vote* of these weak classifiers is used.

The major reason for the popularity of random forests is that it can be easily applied to a wide range of prediction problems, and has few hyperparameters to tune. In addition, as mentioned above, it's ability to deal with small sample sizes and high-dimensional feature spaces and ease of scalability have played key roles in popularizing the algorithm. However, it is also to be noted that the black-box nature of the algorithm does not allow for easy interpretability.

In this report, random forest algorithm is used to build a predictive model for the car evaluation dataset, which contains 6 categorical input features. The target variable, or the variable to be predicted, is the class that the car belongs to: *unacc*, *acc*, *good*, *vgood*.

We start by analysing the mathematical formulation of random forests, which we then use to build a predictive model and tune the hyperparameters of the same. The performance of the model is evaluated using various metrics.

II. Random Forests - A Mathematical Overview

Random forests are ensemble models, which use an aggregation of multiple weak decision trees (which have low predictive power), to form a combined model with high performance. In this section, we examine how a decision tree is built and how it works. We then move on to understand how multiple such trees are aggregated into a random forest, and the reason for their high performance.

A. Decision Trees

A decision tree is a flowchart like structure, which can be used to divide the input space into multiple regions using a series of conditions on the input variables.

The tree is grown successively using a greedy approach known as *recursive binary splitting*. It begins at the top of the tree, where all observations belong to a single region, and successively splits the input space. Each split is represented by two new branches further down on the tree. In order to decide what the best split is, measures such as the *residual sum of squares* and *Gini index* are used for regression and classification trees respectively. These measures quantify the accuracy of the split or the purity of the resulting classes respectively. This process of splitting the nodes is continued until a stopping criterion is reached, such as one limiting the maximum depth of the tree.

While making a prediction on a new data point, we start at the top of the tree and successively narrow down one region where the observation belongs. Once in this region, we predict the observation corresponding to the maximum number of training data points in that region in case of classification, and in case of regression, we predict the average output of those data points.

B. Bagging

Though decision trees are well known for their high interpretability, they do not have the predictive accuracy of other machine learning techniques as they suffer from *high variance*. **Bootstrap aggregation**, or **bagging** is a process used to reduce the variance of a statistical learning model, and has proven to have good results when applied to decision trees.

In general, averaging over a set of observations reduces variance, thus, a natural method to reduce variance in a model would be to take many training sets from the population, build separate prediction models on each set, and average the resulting predictions. However, this is not practical as multiple training sets are generally not available. Instead, we can bootstrap, or sample from a single dataset with replacement. Using this approach, we can obtain B different bootstrapped training sets, and build models on each one of them. Finally, we average the predictions of all B models, and make predictions for regression models as

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x) \tag{1}$$

where $\hat{f}^{*b}(x)$ is the b^{th} model. For classification models, we record the class predicted by each of the B trees, and take a *majority vote*, wherein the final prediction is the most commonly predicted one.

C. Random Forests

Random forests improve over bagged trees by using a technique to *decorrelate* the trees. Similar to bagging, we build a number of trees on bootstrapped training samples. However, at every split, only a small proportion of the features are considered as candidates. Thus, a random sample of predictors is chosen from the full set, out of which the best split candidate is chosen. This means that at each split, the algorithm is not allowed to even consider a majority of the features available.

The reason behind this is that in case of bagging, if there is a very strong predictor, it will be used by almost every tree in the bagged model, which will make the models highly correlated. Averaging the predictions of many correlated models does not, however, produce a significant improvement over a single model.

Random forests, on the other hand, ensure that the strongest feature is not even considered in a majority of the splits, giving the other features a chance. This decorrelating process makes the resulting model less variable.

As in bagging, the output is predicted using the average or the majority depending on whether it is a regression or a classification setting.

D. Estimating Error

In most machine learning models, the validation or cross-validation framework is used to estimate the error of the model on unseen data. However, the test error of a bagging or random forest model can be estimated in a much more straightforward way.

Since bagging uses a bootstrapped subset of the training data, only a portion of the data is used for training each tree. The remaining data points, which are not used to train a particular tree, are known as **Out-of-Bag** (OOB) observations. For every observation in the training data, it can be shown that there are approximately $B/3$ trees for which the observation is OOB. The predictions of these trees can then be aggregated to predict the response for the observation. This can be used to calculate the mean square error or classification error using other metrics such as accuracy.

The advantage of using OOB is particularly significant for large datasets where cross-validation might be computationally difficult, or for very small datasets in which splitting into a train and validation dataset might not be feasible.

E. Interpreting Random Forest Models

Random forests generally lead to improved predictive accuracy, which is, however, at the cost of interpretability. Since we cannot represent the model as a single tree like structure, we lose the ability to identify the variables which were most crucial for the prediction process.

Though it is much more difficult to interpret a random forest than a single tree, we can obtain a summary of the significance of each predictor, using metrics such as RSS (for regression models) and Gini index (for classification models). The total reduction in the above-said measures due to a given predictor, averaged over all trees, can be used as a measure of the importance of a particular feature, a high value of which indicates that the variable in question was more important than one with a lower value.

III. Data

The data used for this task is the *Car Evaluation Model*, derived from a hierarchical decision model. The various input features and their definitions are given in Table I. The *target variable*, or the variable to be predicted is the class that the car belongs to: *unacc*, *acc*, *good*, *vgood*.

TABLE I
VARIOUS INPUT FEATURES THAT ARE PRESENT IN THE DATASETS

Variable	**Type and Description**	**Key (if applicable)**
buying	Buying Price: Categorical	vhigh, high, med, low
maint	Price of Maintenance: Categorical	vhigh, high, med, low
doors	Number of Doors: Categorical	2, 3, 4, 5more
persons	Capacity: Categorical	2, 4, more
lug_boot	Size of the luggage boot: Categorical	small, med, big
safety	Estimated safety of the car: Categorical	low, med, high

A. Preliminary Data Analytics

The dataset consists of **1727** data points, with no missing values. This is divided into a training and test data of proportions 70% and 30% respectively. The distribution of data points in the training data, for each input feature is given in Fig. 1.

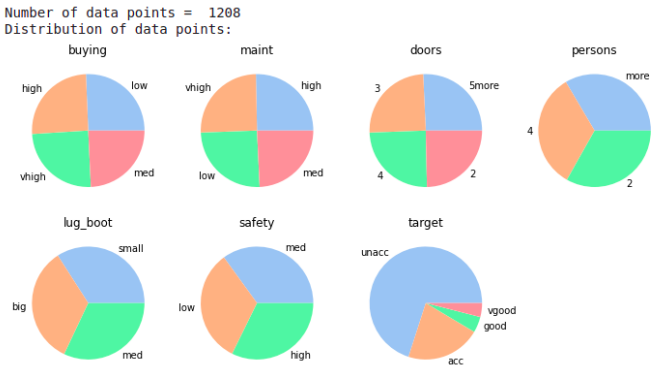

Fig. 1. Distribution of training data with respect to input and target features

IV. The Problem

For the given dataset, we try to predict the target variable using the given input variables. The steps followed for the same are:

- Data cleaning
- Data Visualization and Analysis
- Feature Engineering
- Building random forest models and tuning hyperparameters
- Using the final model to make predictions on the test data
- Identifying the most important features using the random forest

A. Data Cleaning

The dataset does not contain any missing values. However, as shown in Fig. 1, the dataset is extremely skewed. For the same dataset, using synthetic minority oversampling technique (SMOTE) technique for training data augmentation, a decision tree model was previously built that gave 98% accuracy on test data. Hence, in this report, we try to use a random forest model for the same dataset without data augmentation, to compare the performance with that of a decision tree.

B. Data Visualization and Analysis

Fig. 3 shows a pairwise plot showing the frequency of the different target variables for each pair of input variables. In these scatter plots, the size of the bubble varies with the frequency at which the two input features occur together. This is colour coded by the target variable. The bar graphs along the diagonal show the count of each input feature, again colour coded by the target variable.

This allows us to better visualise the distribution of the data, instead of only plotting a histogram of each input variable. We see that for many variable pairs, (for example: buying = *vhigh*, maint = *vhigh*), there is only one output class in the training data. There are many more input variable pairs which point to only two possible output classes. This shows that the data can be fit well using a decision tree or ensemble models using the same, where the input space can be split into various regions.

C. Feature Engineering

As all the variables are categorical, we will encode them to numerical class labels. This is done for all features, including the target variable. The numerical values have been assigned in the same order as the nominal variable names (for example: buying - *low* = 0, *med* = 1, *high* = 2, *vhigh* = 3).

D. Building Random Forest Models and Tuning Hyperparameters

The major hyperparameters to be tuned in a Random Forest model are the number of estimators (decision trees), maximum depth of each tree, and the number of features to be considered for random sampling. In this analysis, we build various models to tune the number of features and the number of estimators, while the maximum depth of the trees will be adjusted accordingly by the model. The hyperparameters considered are:

- Number of estimators: 5 - 100
- Maximum features considered at each split:
  - *sqrt*: Square root of the total number of features available, i.e., 2.
  - *log2*: Logarithm of the total number of features to the base 2, i.e., 2.
  - *0.5*: Half of the total number of features available, i.e., 3.
  - *None*: All the available input features will be considered, i.e., 6.

Fig. 2 - Fig. 5 shows the performance of the model for the training data (out-of-bag error) and test data (accuracy, f1 score).

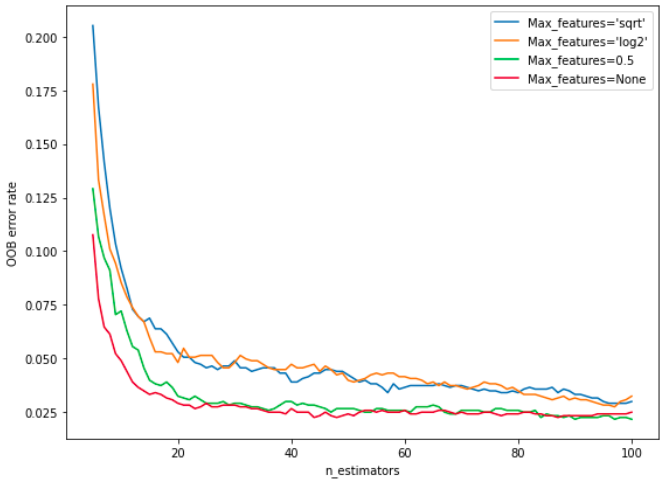

Fig. 2. Out-of-bag error of models with various hyperparameters

We see that a random forest with approximately 30 estimators and considering all features at each split performs very well on the given data. Even though using all features will most likely fail to decorrelate the individual trees, since the number of input features is less in this case (6), this gives the best results. These hyperparameters will be used for further analysis.

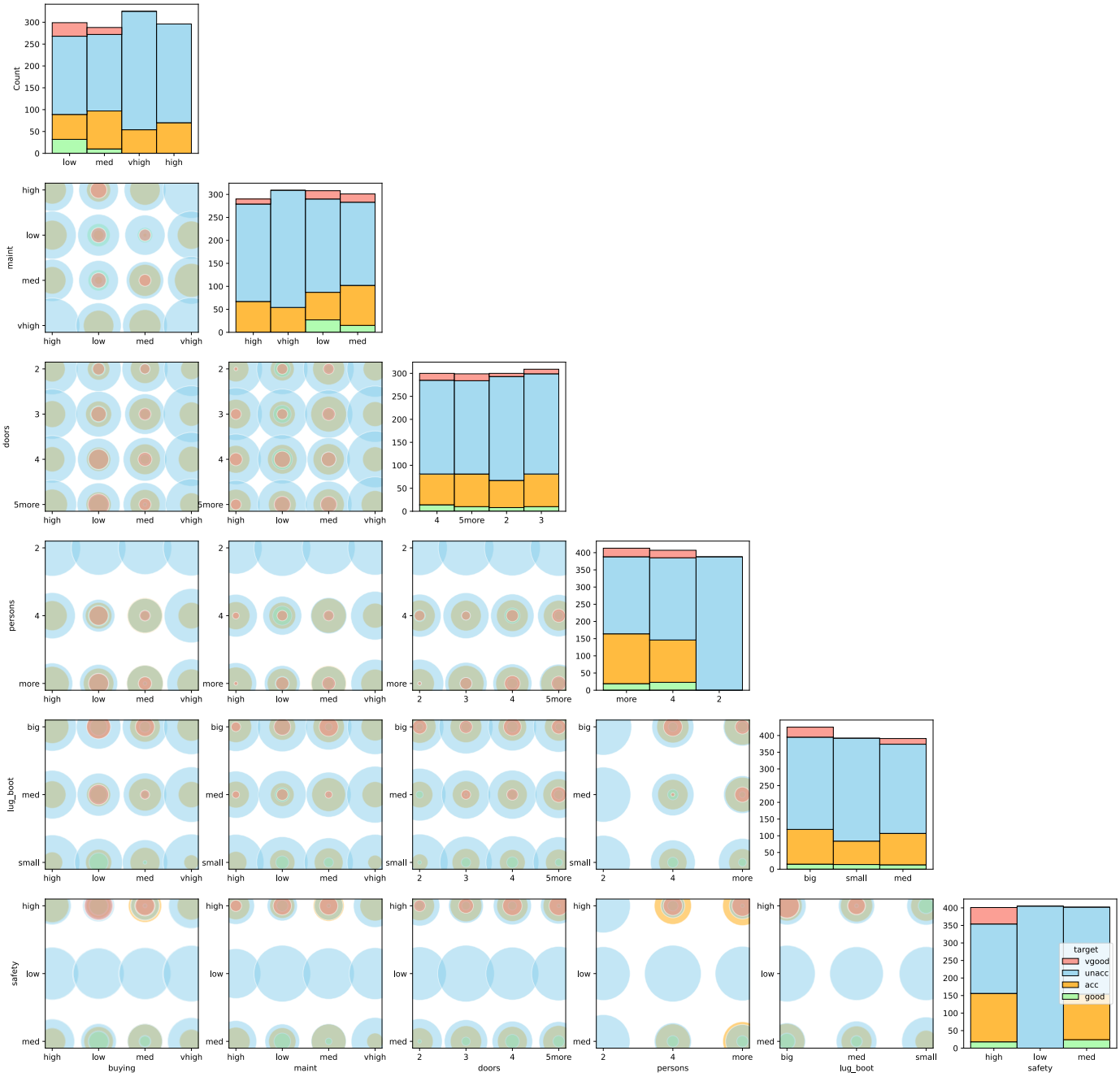

Fig. 3. Pairwise plot of the input features, colour coded by the target variable

E. Using the Best Model to Make Predictions on the Test Data

The metrics obtained using the best model for both the training and test data are shown in Fig. 6. In addition, the classification summary and confusion matrix are plotted in Fig. 7 and Fig. 8 respectively.

We see that the model does not overfit much to the training data. In addition, the model is not biased towards any one class, and that though the support for classes is highly skewed, the performance is similar. In addition, we also observe that even without data augmentation, the random forest model performs at least as well as a decision tree with data augmentation.

F. Identifying the Most Important Features

The importance of each feature (calculated using the reduce in Gini index due to that particular variable averaged across all trees) is shown in Fig. 9.

V. Conclusions

In this report, we analysed the mathematical formulation behind random forests, and examined how to tune various hyperparameters for the same. The model was a powerful predictor that generalised well to unseen data, whose performance

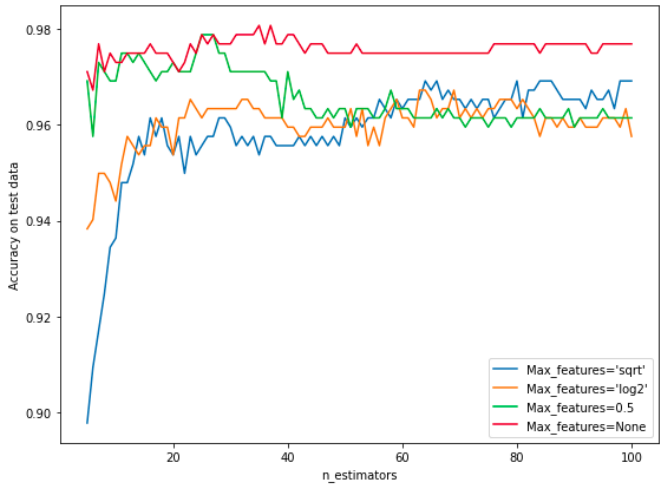

Fig. 4. Accuracy of models with various hyperparameters on test data

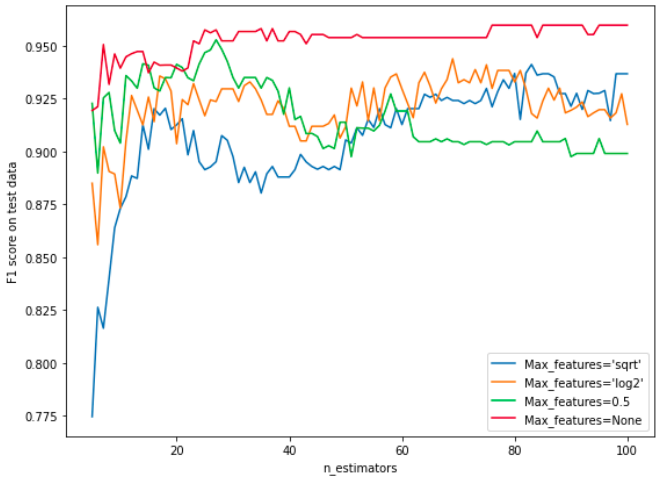

Fig. 5. F1 score of models with various hyperparameters on test data

```
Training:
Accuracy =  1.0
Precision =  1.0
Recall =  1.0
F1 Score =  1.0
Test:
Accuracy =  0.9788053949903661
Precision =  0.9544850143237202
Recall =  0.9547750949694946
F1 Score =  0.953738748492531
```

Fig. 6. Performance of the selected model on training and test data

	precision	recall	f1-score	support
unacc	0.99	0.99	0.99	367
acc	0.95	0.96	0.95	118
good	0.88	0.95	0.91	22
vgood	1.00	0.92	0.96	12
accuracy			0.98	519
macro avg	0.95	0.95	0.95	519
weighted avg	0.98	0.98	0.98	519

Fig. 7. Classification report showing the F1 score, accuracy, and other metrics for each class

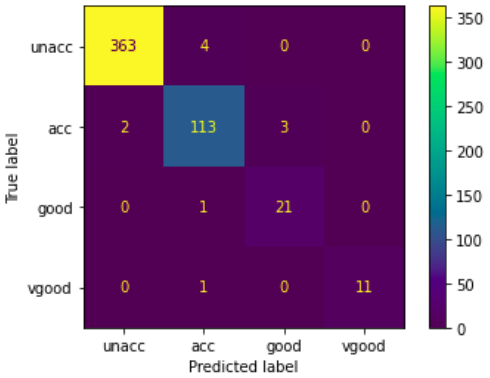

Fig. 8. Confusion matrix showing the distribution of true classes vs predicted classes

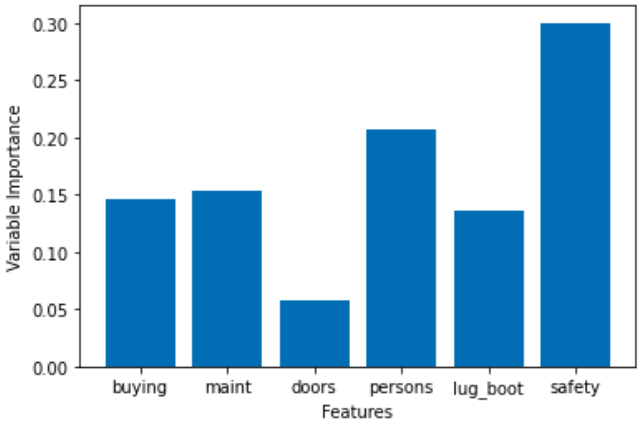

Fig. 9. Importance of each input feature

was measured using metrics such as accuracy, precision, recall and F1 score.

A. *Avenues for Future Work*

The performance of the model can be compared to that of various other ensemble methods such as bagging and boosting, all based on decision trees. In addition, a model using lesser features can also be examined by performing feature selection.

REFERENCES

[1] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, "Tree-Based Methods" in *An Introduction to Statistical Learning,* New York, Springer, 2013.

[2] Biau, G., Scornet, E., *A random forest guided tour*, TEST 25, 197–227 (2016). https://doi.org/10.1007/s11749-016-0481-7

[3] Tony Yiu, *Understanding Random Forest*, https://towardsdatascience.com/understanding-random-forest-58381e0602d2