University of Bonn

Master Programme in Economics

Random Forest for Classification Problems

Submitted by

Burak Balaban

Arkadiusz Modzelewski

Raphael Redmer

Supervisor: Prof. Dr. Dominik Liebl

January 9, 2020

Contents

1 Abstract

As a non-parametric estimation tool, decision trees attract attention in the economics literature. Yet, decision trees suffer from high variance and, for prediction purposes higher variance seems to be a crucial problem, thus, several improvements were proposed such as bootstrap aggregation, boosting and most importantly random forests. In this project, while the main focus is being on the random forest. The elements of statistical learning by [6] [14] [11], [9] and as expected [4] are the main literature that will be utilized in this project.

To explain the concept of random forests in full extent, primarily decision trees should be discussed. Exploiting the main idea and struggles with bias-variance trade-off, random forests' importance can be emphasized as a more stable prediction tool [11]. Conceptual comparison of random forests with bagging and boosting can deliver a better understanding of its unique features as [8] shows in a similar fashion. To get a further understanding, random forests' estimation process can be mathematical explained [1] and likewise, examining the consistency of estimator and showing the properties can be included [3], [5]. Also, variable importance in the tree growing process is another area that needs to be delved into [7] and [10].

2 Introduction

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

3 Decision tree

3.1 Main idea

The Decision Tree is a non-parametric supervised learning method used for classification and regression. It predicts the response with a set of if-then-else decision rules derived from the data. The deeper the tree, the more complex the decision rules and the closer the model fits the data. The decision tree builds classification or regression models in the form of a tree structure. Each node in the tree further partions the feature space into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and terminal nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor is called the root node. Decision trees can handle both categorical and numerical data.

An example of such a tree is depicted below in figure 1.

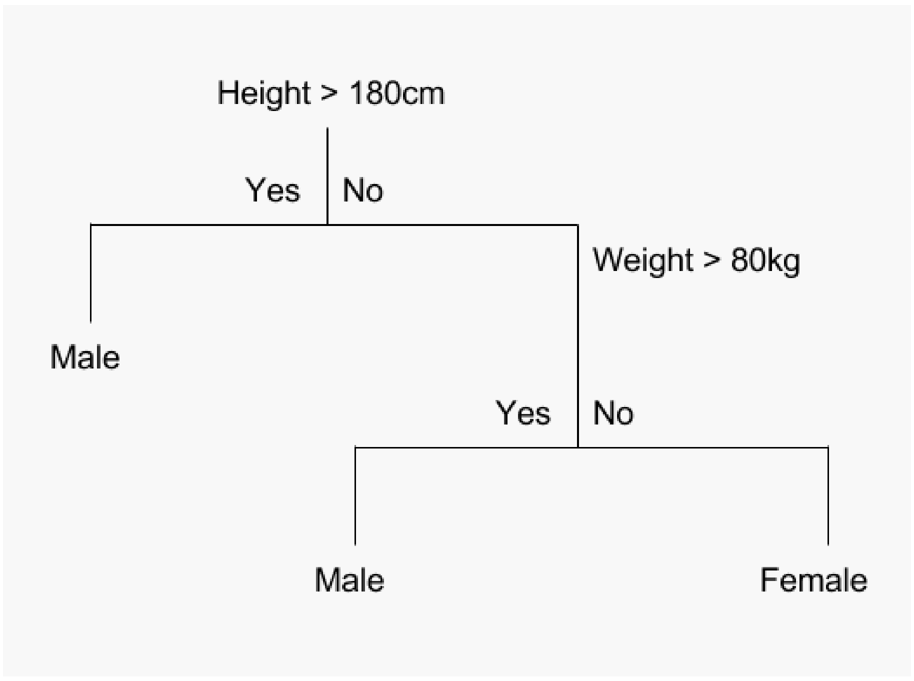

Figure 1: Given a data set with two features height and weight, and gender as the target variable, this example tree stratisfies the two-dimensional feature space into three distinct subset each represented by the terminal nodes at the bottom. The stratification occurs at the two deciding nodes depending either on whether its height is above 180 cm and or its weight is above 80kg.

3.2 Tree Building Process

This chapter describes the CART algorithm for tree building as specified in [2]. The basic idea of tree growing is to choose a split among all the possible splits at each node so that the resulting child nodes are the "purest". In this algorithm, only univariate splits are considered. That is, each split depends on the value of only one predictor variable. All possible splits consist of possible splits of each predictor.

A tree is grown starting from the root node by repeatedly using the following steps on each node (also called binary splitting)

(i) **Find best split s for each feature X_m:** For each feature X_m, there exist $K-1$-many potiential splits whereas K is the number of different values for the respective feature. Evaluate each value $X_{m,i}$ at the current node t as a candidate split point (for $x \in X_m$, if $x \leq X_{m,i} = s$, then x goes to left child node t_L else to right child node t_R). The best split point is the one that maximize the splitting criterion $\Delta i(s,t)$ the most when the node is split according to it. The different splitting criteria will be covered in the next chapter.

(ii) **Find the node's best split:** Among the best splits for each feature from Step (i) find the one s^*, which maximizes the splitting criterion $\Delta i(s,t)$.

(iii) **Satisfy stopping criterion:** Split the node t using best node split s^* from Step (ii) and repeat from Step (i) until stopping criterion is satisfied.

3.2.1 Splitting criteria

Since we are only concerned with classification, Y is categorical. The original CART algorithm uses Gini and Twoing as purity measures for the splitting criterion. However, implementations of the algorithm such as Python's sklearn package also contain entropy and misclassification rate as measures of impurity.

For a give learning sample L for a J class problem, let N_j be the number of instances $\{x, y\}$ belonging to in class j.

In node t, let $N(t)$ be the total number of instances with $\{x, y\} \in t$ and $N_j(t)$ the number of class j cases in t. The proportion of the class j instances in the sample L falling into t is $N_j(t)/N_j$. For a given set of priors, $\pi(j)$ is interpreted as the probability that an instance belongs to class j.

At node t let the probabilities $p(j, t)$, $p(t)$ and $p(j|t)$ be estimated by using Thus, let

$$p(j, t) = \frac{\pi(j) N_j(t)}{N_j} \tag{1}$$

be the estimate for the probability that na instance will both be in class j and fall into node t. Therefore, the estimate for the probability that any instance falls into node t is defined by

$$p(t) = \sum_j p(j, t), \tag{2}$$

The estimate $p(t)$ for the probability that an instance belongs to class j given that it falls into node t is defined by

$$p(j|t) = \frac{p(j, t)}{p(t)} = \frac{p(j, t)}{\sum_j p(j, t)}. \tag{3}$$

It holds that the conditional probability $p(j|t)$ must satisfy

$$\sum_j p(j|t) = 1 \tag{4}$$

Let $i(t)$ be an impurity measure evaluated at note t. Then, the decrease of impurity (i.e. the splitting criterion) is defined as

$$\Delta i(s, t) = i(t) - p_L i(t_L) - p_R i(t_R), \tag{5}$$

where p_L and p_R are probabilities of sending a case to the left child node t_L and to the right child node t_R respectively. They are estimated as $p_L = p(t_L)/p(t)$ and $p_R = p(t_R)/p(t)$.

As already stated abobe, the goal is to maximize $\Delta i(s, t)$ In the following, different measures for impurity will be presented.

Gini Measure

The Gini impurity measure is defined as

$$i(t) = \sum_{j} p(j|t)(1 - p(j|t)) = 1 - \sum_{j=1}^{J} p_j^2 \tag{6}$$

The intuition behind this measure is to assign nodes for which its probabilities are more skewed towards a particular group a higher value. Conversely, if a node has more balanced distribution, then $i(t)$ will turn out to be lower. For example, in the case of $J = 2$, $i(t)$ will be maximized with $p(j|t) = 0.5$ for $j = 1, 2$.

Information Entropy

The Entropy measure from information theory is defined as

$$i(t) = \sum_{j} p(j|t) log(p(j|t)) \tag{7}$$

which measures the average rate at which information is produced by a stochastic source of data . Thus, it can also be used for measuring impurity.

Rate of Misclassification

The rate of misclassification is defined as

$$i(t) = 1 - \max_{0<j\leq J} p(j|t). \tag{8}$$

which measure the proportion of instances node t not belonging to the dominant group in t.

3.2.2 Bias-variance trade-off

To measure the fit of a model to data, the generalization error is a widely used concept(Breiman, 2001). Principally, we use generalization error to compare methods and adjust parameters to improve methods. Under certain assumptions, we can decompose the generalization error into three main components; the Bayes Error, the squared bias ($bias^2$) and the variance. We will explain the mathematical aspects of the generalization error in the following chapters. For now, we only need to mention the Bayes Error is irreducible and independent of the classifier, and there is a trade-off between $bias^2$ and variance. Since decision trees suffer from high variance and decreasing variance causes $bias^2$ to increase, we need methods to bypass this trade-off. Methods including Bagging, Boosting and Random Forest exploits mathematical properties to decrease variance without increasing $bias^2$. We will explain bagging briefly, exhibit mathematical dynamics of Random Forest and use Boosting in the application section.

3.3 Bagging

It is easy to guess that decision trees belong to methods which are sensitive to the specific data on which they are trained. It means that when we change training data we can get much different resulting decision trees and as a result of that also predictions are different. Bagging was created for these kind of methods which has high variance, like classification and regression trees. Bagging or in other words Bootstrap Aggregation is a procedure that can be used for reducing the variance. As a first step in bagging procedure we do bootstrapping, so we have to create many random subsample datasets with replacement and then we use these datasets as training data for our model for example in our case decision tree. As a result we have the same number of decision trees as the number of created datasets. Last step is to use new dataset to predict the result from each model and finally average the result over all unit results. After introducing bagged decision trees, another improvement to this method was created. Method which is an improvement over bagged decision trees is commonly known as the Random Forest which is the topic of our next section of paper.

4 Random Forest

Decision trees, which we mentioned in the previous section, have been used for a long time. Deployment of decision trees is visible in simple situations and also in more complex scientific or real life and industrial affairs. The recent popularity of decision trees is due to work presented by Breiman between 1996 and 2004 that ensamples of different decision trees can get a meaningful improvement in accuracy in classification problems and other common learning tasks such as regression. As the unit in this procedure is based on decision tree and includes an injection of randomness, this method is known as a random forest.

4.1 Main idea and illustration

An ensemble of randomly trained decision trees, so in other words random decision forest was defined by Breiman in his work ,,Random Forests" from 2001 as follows:

Theorem 1. *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{g(\boldsymbol{x}, \Theta_k), k = 1, 2, ...\}$ *where the* $\{\Theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* $\boldsymbol{x}$ *.*

4.2 Randomness in the model

The main aspect of a random decision forest model is an injection of randomness which allows to have all unit trees different from the others. Two key concepts that makes decision forest "random" are:

1. Random sampling of training data points when building trees
2. Random subsets of features considered when splitting nodes

In practise, random sampling of training observations is done by using bootstrapping. Bootstrapping for random forests means that every single decision tree learns from a random sample which is drawn with replacement. The second important injection of randomness into model is taking into consideration only a subset of all the variables, when splitting each node in every unit decision tree. It means that number of variables for splitting the node has to be chosen and for classification recommended number is $\lfloor\sqrt{n}\rfloor$, where n is a number of features in the classification problem. This yields that in problem with 10 different variables, only 3 randomly chosen will be considered for splitting the node. Randomness parameter, so the number of chosen variables has a meaningful impact on the model, because except controlling the amount of randomness within each tree, it controls also the amount of correlation between different trees in the forest. As randomness parameter decreases, trees become more decorrelated [Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning].

4.3 Training, testing and prediction

Training of all trees is done independently and testing consists in the fact that each test point $\mathbf{x}$ is pushed through every tree included in forest until it ends in corresponding leaves. As a last step is taking all predictions from every unit decision tree and combining them into prediction of single random forest. Combination of different tree predictions can be done in different ways. In random forests for classification problems, forests generate probabilistic output. It means that they return not just a single class point prediction, but whole class distribution, so combination of tree predictions can be described as below:

$$p(c|\mathbf{x}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{x}) \tag{9}$$

where in a random forest with the number of decision trees equals to T each tth tree obtains the posterior distribution $p_t(c|\mathbf{x})$. The class label here is symbolized by c such that $c \in \mathbf{C}$ with $\mathbf{C} = \{c_k\}$

4.4 Out of Bag sample

Another important idea which is used in applications of random forest is validating model using Out of Bag sample. When performing a bootstrap for getting a sample of data for training, remaining part of data is our Out of Bag (OOB) sample. After training random forest model, OOB sample will be used as not known data for prediction. Leftover sample will be passed through every possible decision tree in the model that not include this data in the bootstrap training sample. Out of Bag is commonly used to monitor error of predictions. Breiman's work on error estimates from 1996, shows empirically that Out of Bag estimate is as accurate as a test set of the same size as the training set [Leo Breiman, OUT-OF-BAG ESTIMATION, 1996].

5 Mathematical explanation

Let $D = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ be the set, we want to train our model on and $T_{D,\theta}$ decision tree produced by using the set D and parameters θ. We assume that D is countable which normally is the case especially for Y values although replacing sums with integrals can extent the analysis and provide results for uncountable sets as well (Kohavi, 1996). As mentioned earlier, Random Forest classifier selects a bootstrapped subset of observations denoted as $\boldsymbol{D'}$ and grow the decision tree with only a subset of regressors. Repeating this tree growing process B times gives a Random Forest Estimator. Assume x^* is the value that we want to predict its class, there are two rules that can be used to get the prediction; majority voting and soft voting (Louppe, 2014; Zhou,2012).

In majority voting, after getting every trees prediction denoted as $\hat{T}(x^*)$ final prediction is the class that gets most votes from trees;

$$\boldsymbol{RF}_{D,\theta_1,\theta_2,...,\theta_B}(x^*) = \underset{c\in Y}{argmax} \sum_{b=1}^{B} 1(\hat{T}_b(x^*) = c) \tag{10}$$

In soft voting, probability estimates of a tree denoted as $\hat{p}_{D,\theta_b}(Y = c|X = x^*)$ is estimated and after all are averaged and most likely class is predicted;

$$\boldsymbol{RF}_{D,\theta_1,\theta_2,...,\theta_B}(x^*) = \underset{c\in Y}{argmax} \frac{1}{B} \sum_{b=1}^{B} \hat{p}_{D,\theta_b}(Y = c|X = x^*) \tag{11}$$

As mentioned in Breiman(1994), aformentioned two voting procedures provides similar results, yet using soft voting can provide smoother class probability estimates and be exploited in a deeper analysis setting such as certainty estimates investigation (Louppe, 2014).

5.1 Properties

The generalization error which also called test error or the expected prediction error of $T_{D,\theta}$ is;

$$\boldsymbol{Err}(T_{D,\theta}) = \mathbb{E}_{X,Y}\{L(Y, T_{D,\theta}(X))\} \tag{12}$$

where L is the loss function measuring the difference between its two arguments. Since we focus on classification setting, the zero-one loss function is our interest, however, to get a better understanding, widely used in regression type predictions, the squared loss function will be examined first. The bias-variance decomposition of both functions are similar and follow the same dynamics(Domingos, 2000). The squared loss function can be defined as

$$L(Y, T_{D,\theta}(x)) = (Y - T_{D,\theta}(x))^2 \tag{13}$$

while the zero-one loss function is

$$L(Y, T_{D,\theta}(X)) = 1(Y \neq T_{D,\theta}(X)) \tag{14}$$

The expected prediction error for both functions

$$\boldsymbol{Err}(T_{D,\theta}) = \mathbb{E}_{X,Y}\{(Y - T_{D,\theta}(x))^2\} \tag{15}$$

$$\boldsymbol{Err}(T_{D,\theta}) = \mathbb{E}_{X,Y}\{1(Y \neq T_{D,\theta}(X))\} = P(Y \neq T_{D,\theta}(X)) \tag{16}$$

where the last expression in the second equation is the probability of misclassification of the tree.

Given the probability distribution of P(X,Y), there exists a model ϕ_β that minimizes the expected prediction error and can be derived analytically independent or learning set D (Louppe, 2014). Conditioning on X gives;

$$\mathbb{E}_{X,Y}\{L(Y, \phi_\beta(X))\} = \mathbb{E}_X\{\mathbb{E}_{Y|X}\{L(Y, \phi_\beta(X))\}\} \tag{17}$$

Minimizing the term with conditional expectation with respect to Y;

$$\phi_\beta = \underset{c \in Y}{argmin} \mathbb{E}_{Y|X=x}\{L(Y, c)\} \tag{18}$$

ϕ_β is defined as Bayes model and $\boldsymbol{Err}(\phi_\beta)$ is the residual error, the minimum obtainable error with any model, which is considered as the irreducible error due to random deviations in the data(Louppe, 2014). We will exploit the irreducible concept when examining the dynamics of random forests and decision trees. In that sense, with couple of manipulations, Bayes Model for squared loss is

$$\begin{aligned}\phi_\beta &= \underset{c \in Y}{argmin}\ \mathbb{E}_{Y|X=x}\{(Y - c)^2\} \\ &= \mathbb{E}_{Y|X=x}\{Y\}\end{aligned} \tag{19}$$

For squared loss function, bayes model predicts the average value of Y at X=x. In zero-one loss function case Bayes Model is

$$\begin{aligned}\phi_\beta &= \underset{c\in Y}{argmin}\ \mathbb{E}_{Y|X=x}\{L(Y,c)\}\\ &= \underset{c\in Y}{argmax}\ P(Y = c\,|X = x) \end{aligned} \tag{20}$$

The most likely class in the set Y is chosen by Bayes Model when using the zero-one loss function. Aformentioned residual error can be computed for both functions;

$$\boldsymbol{Err}(\phi_\beta) = \mathbb{E}_{Y|X=x}\{(Y - \phi_\beta(x))^2\} \tag{21}$$

$$\boldsymbol{Err}(\phi_\beta) = P(Y \neq \phi_\beta(x)) \tag{22}$$

With using the squared loss function, $\boldsymbol{Err}(T_{D,\theta}(x))$ can be written as

$$\begin{aligned}\boldsymbol{Err}(T_{D,\theta}(x)) &= \mathbb{E}_{Y|X=x}\{(Y - T_{D,\theta}(x))^2\}\\ &= \boldsymbol{Err}(\phi_\beta(x)) + (\phi_\beta(x) - T_{D,\theta}(x))^2\end{aligned} \tag{23}$$

Intermediate steps are included in the Appendix. As mentioned above, first term in the equation corresponds the irreducible error and the second term is due to the prediction differences between Bayes Model and our decision tree estimation. The expected prediction error increases with an increase in that difference. Since the result does not depend on the Y-values, it can be also expressed without conditional expectation. Decision trees in random forest classifier uses a bootstrapped dataset and D is a random variable, thus, if we further examine the second term with taking expectation over D, it decomposes as

$$\begin{aligned}&\mathbb{E}_D\{(\phi_\beta(x) - T_{D,\theta}(x))^2\}\\ &= (\phi_\beta(x) - \mathbb{E}_D\{T_{D,\theta}(x))^2 + \mathbb{E}_D\{(\mathbb{E}_D\{T_{D,\theta}(x)\} - T_{D,\theta}(x))^2\}\end{aligned} \tag{24}$$

With intermediate steps being in Appendix, the first term in (23) shows how the expected prediction of our decision tree differs from Bayes Model also called squared bias and the latter term is the variance of our estimator. Therefore, we can define $\boldsymbol{Err}(T_{D,\theta})$ as follows

$$\boldsymbol{Err}(T_{D,\theta}) = noise(x) + bias^2(x) + var(x) \tag{25}$$

where

$$\begin{aligned}&noise(x) = \boldsymbol{Err}(\phi_\beta)\\ &bias^2(x) = (\phi_\beta(x) - \mathbb{E}_D\{T_{D,\theta}(x)\})^2\\ &var(x) = \mathbb{E}_D\{(\mathbb{E}_D(T_{D,\theta}(x)) - T_{D,\theta}(x))^2\}\end{aligned}$$

The same decomposition can be conducted for the zero-one loss function and as mentioned in (Louppe,2014), (Domingos,2000), (James,2003), (Friedman,1997) both zero-one and squared loss functions can be decomposed in similar fashion. However, since the distribution of D is unknown, bias-variance decompostion cannot be solved explicitly as done for squared loss(Louppe, 2014). (Kohavi,1996) introduces another decomposition for zero-one loss function (included in the Appendix), but, still it remains to be unexplanatory compared to squared loss, thus, we explain the dynamics with using squared loss although the main focus of the paper remains to be on classification setting.

In regression setting, random forest classifier shares the same idea with classification prediction with soft voting. Random forest classifier for regression can be written as

$$\boldsymbol{RF}_{D,\theta_1,\theta_2,\ldots,\theta_B}(x) = \frac{1}{B}\sum_{b=1}^{B}(T_{D,\theta_b}(x)) \tag{26}$$

When we take the average prediction in this case equals to expectation in terms of training set, we get

$$\begin{aligned}\mathbb{E}_{D,\theta_1,\theta_2,\ldots,\theta_B}\{\boldsymbol{RF}_{D,\theta_1,\theta_2,\ldots,\theta_B}(x)\} &= \mathbb{E}_{D,\theta_1,\theta_2,\ldots,\theta_B}\{\frac{1}{B}\sum_{b=1}^{B}(T_{D,\theta_b}(x))\}\\ &= \frac{1}{M}\sum_{b=1}^{B}\mathbb{E}_{D,\theta_b}\{T_{D,\theta_b}(x)\}\\ &= \mu_{D,\theta}(x)\end{aligned} \tag{27}$$

where $\mu_{D,\theta}(x)$ is the average prediction of all ensembled trees as a random forest since θ's are random, independent and have the same distribution(Louppe, 2014). When we extend this finding bias of a random forest we can state that

$$bias^2(x) = (\phi_\beta(x) - \mu_{D,\theta}(x))^2 \tag{28}$$

meaning that squared bias cannot be decreased and will be same for any randomized models. Although random forest is inadequate to propose any structure to decrease the prediction error so far regarding $noise(x)$ and $bias^2$, it displays phenomenal performance in reducing the last remaining part of the prediction error. Thus, we can continue our exploration with variance of random forest and need to define the correlation coefficient $\rho(x)$. For any two trees $T_{D,\theta'}$ and $T_{D,\theta''}$ trained with the same training data and different growing parameters θ' and θ'', we can define the correlation coefficient as follows

$$\rho(x) = \frac{\mathbb{E}_{D,\theta',\theta''}\{T_{D,\theta'}(x)T_{D,\theta''}(x)\} - \mu^2_{D,\theta}(x)}{\sigma^2_{D,\theta}(x)} \tag{29}$$

We utilize the definition of the Pearson's correlation coefficient and the property of θ' and θ'' following the same distribution in the intermediate steps which are included in the Appendix. $\sigma_{D,\theta}(x)$ is the variance of a decision tree and generally $\rho(x)$ represents the effect of randomization in the learning algorithm. In our case, it is close to 1 when predictions of two decision trees are highly correlated and implies that randomization does not have a significant effect. On the other hand, if it is close to 0, the prediction of two trees are perfectly random and non-correlated.

We can decompose variance of a random forest as follows;

$$\mathbb{V}_{D,\theta_1,\theta_2,\ldots,\theta_B}\{\boldsymbol{RF}_{D,\theta_1,\theta_2,\ldots,\theta_B}(x)\} = \rho(x)\sigma^2_{D,\theta}(x) + \frac{1-\rho(x)}{B}\sigma^2_{D,\theta}(x) \tag{30}$$

Increasing the number of ensembled trees B, will lower the latter expression in the equation and in the extreme case where $B \to \infty$, the variance of a random forest equals to $\rho(x)\sigma^2_{D,\theta}(x)$ and due to randomization in the algorithm $\rho(x) < 1$, thus, variance of a random forest is less than variance of a decision tree. This inference implies that the expected prediction error of a random forest is less than the expected prediction error of a decision tree. If the decision trees in the random forest are independent and consequently

$\rho(x) \to 0$, the variance is reduced to $\frac{\sigma^2_{D,\theta}(x)}{B}$ which can be decreased with increasing B as mentioned.

6 Interpretation

In many cases, the main purpose of using a random forest model is its usefulness in performing predictions of a dependent variable based on a set of explanatory variables. In addition, very often, to understand the processes under study we need to understand which explanatory variables are the most important and useful to make mentioned predictions. Thanks to random forest we are often capable to not only build an accurate model with reliable predictions, but also to provide variable importance measures which are very important in the process of interpreting the model and its prediction results. This section of the paper will be fully based on work “Understanding variable importances in forests of randomized trees” presented by G. Louppe, L. Wehenkel, A. Sutera and P. Geurts and ”A Random Forest Guided Tour” presented by G. Biau and E. Scornet.

6.1 Variable importance

In order to rank the importance of the explanatory variables in classification problems using random forest we use two possible measures. Both the first and second measure have been proposed by Breiman (2001, 2002). First of them is known as Mean Decrease Impurity (MDI). MDI measure is based on the total decrease in node impurity from splitting on the given explanatory variable and moreover MDI is averaged over all trees. The second measure is known as Mean Decrease Accuracy (MDA). This measure assumes that if the explanatory variable is not relevant to the study of the problem, then rearranging its values should not demolish prediction accuracy. Mean Decrease Impurity proposed by Breiman (2001, 2002) is given by:

$$\widehat{MDI}(X^{(j)}) = \frac{1}{N_T}\sum_T \sum_{t\in T:v(s_t)=X^{(j)}} p(t)\Delta i(s_t,t) \tag{31}$$

where $v(st)$ is the variable used in split s_t and $p(t)$ is the proportion $\frac{N_t}{N}$ of samples reaching t. By definition $MDI(X^{(j)})$ evaluates importance of a variable $X^{(j)}$ for predicting Y by adding up the weighted impurity decreases $p(t)i(s_t,t)$ for all nodes t where $X^{(j)}$ is used. Mean in the name of MDI comes from averaging over all N_T trees in the random forest. Above definition can be used for any impurity measure $i(t)$ for example: the Gini index, the Shannon entropy. The second mentioned measure MDA uses out-of-bag error estimate. In this case to measure the importance of the $X^{(j)}$ variable, we have to permute its values in the out-of-bag example and use these all observations in the tree. $MDA(X^{(j)})$ is counted thanks to averaging the differences in out-of-bag error estimation after and before the permutation in all trees. Because of used permutations, this measure is sometimes called also as the permutation importance. In order to represent MDA measure mathematically, consider an explanatory $j-th$ variable $X^{(j)}$ and stand $OOB_{k,n}$ as a symbol for the out-of-bag data set of the $k-th$ tree and $OOB^j_{k,n}$ as the same data set, but with randomly permuted values of $X^{(j)}$. Denote also by $m_n(\cdot;\Theta_k)$ the $k-th$ tree estimate. Then definition of MDA is given by:

$$\widehat{MDA}(X^{(j)}) = \frac{1}{N_T}\sum_{T}\Big[C_n\big[m_n(\cdot;\Theta_k), OOB^j_{k,n}\big] - C_n\big[m_n(\cdot;\Theta_k), OOB_{k,n}\big]\Big], \tag{32}$$

where R_n is defined for $OOB = OOB^j_{k,n}$ or $OOB = OOB_{k,n}$ by:

$$C_n\big[m_n(\cdot;\Theta_k), OOB\big] = \frac{1}{|OOB|}\sum_{i:(\boldsymbol{X_i},Y_i)\in OOB}(Y_i - m_n(\boldsymbol{X_i};\Theta_k)). \tag{33}$$

Above formulas are true for regression and classification purposes. It is also visible that population version for $\widehat{MDA}(X^{(j)})$ is as given below:

$$MDA(X^{(j)}) = \mathbb{E}[Y - m_n(\boldsymbol{X'_j};\Theta_k)]^2 - \mathbb{E}[Y - m_n(\boldsymbol{X};\Theta_k)]^2, \tag{34}$$

where $\boldsymbol{X'_j} = (X^{(1)}, ..., X'^{(j)}, ..., X^{(p)})$ and $X'^{(j)}$ is here independent copy of $X^{(j)}$.

7 Application and Comparison

This chapter covers the application of random forest regression and the evaluation of its performance.

In subsubsection 7.1.1, we apply the random forest on simulated data, and show how its performance develops over increasing sample sizes.

Then in subsubsection 7.1.2, we apply the random forest on the real Titanic data set [13] and evaluate its performance.

In 8.1 and **??**, we apply AdaBoost and Gradient Boosting respectively on the Titanic data set and compare their performance with that of the random forest.

7.1 Application of random forest

7.1.1 Simulated data

In the simulation, we use a linear and a non-linear data generating process (DGP) for random forest regression. The linear DGP generates the data tuples (y, x_1, x_2, x_3) as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon, \tag{35}$$

whereas $(\beta_0, \beta_1, \beta_2, \beta_3) = (0.3, 5, 10, 15)$, $x_1, x_2, x_3 \sim \mathcal{N}(0,\, 3)$, and $\epsilon \sim \mathcal{N}(0,\, 1)$.

The performance of the Random Forest over an increasing sample is illustrated below in Figure 3 and Figure 3. For each sample drawn from the linear DGP, a set of parameters were optimized via cross validation. Then, the residual sum of squares (RSS) gets calculated based on the holdout set of 100 instances.

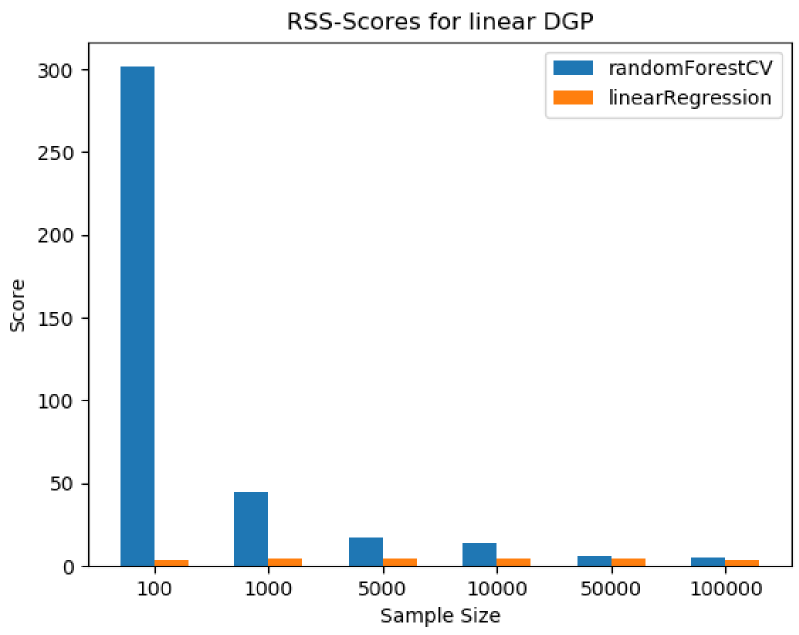

Figure 2: This plot illustrates the RSS for different training sample sizes for Random Forest and OLS. These samples where drawn from a linear DGP in accordance to Equation 35. The holdout set for calculating the RSS were drawn again for each training sample from the same DGP. It always contained 100 observations. In case of the Random Forest, for each sample the parameters got optimized again via cross validation.

As one can see in Figure 2 above, the RSS of the Random Forest converges for the linear DGP to that of the OLS for increasing sample sizes.

The non-linear DGP generates the data tuples (y, x_1, x_2) as follows:

$$y = \beta_0 + \beta_1 I(x_1 >= 0, x_2 >= 0) + \beta_2 I(x_1 >= 0, x_2 < 0) + \beta_3 I(x_1 < 0) + \epsilon, \qquad (36)$$

whereas $(\beta_0, \beta_1, \beta_2, \beta_3)$, x_1, x_2 and ϵ are the same in the previous DGP.

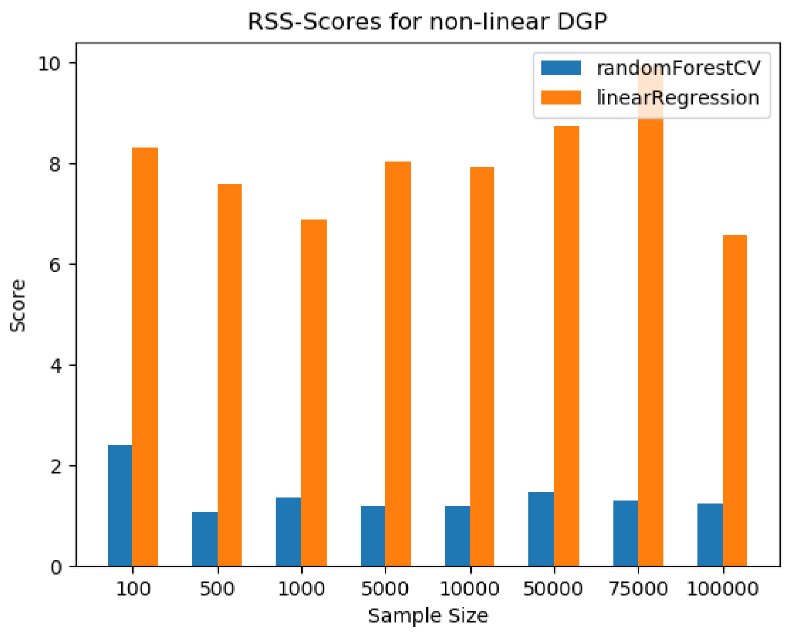

Figure 3: This plot illustrates the RSS for different training sample sizes for Random Forest and OLS. These samples where drawn from a non-linear DGP in accordance to Equation 36. The holdout set for calculating the RSS were drawn again for each training sample from the same DGP. It always contained 100 observations. In case of the Random Forest, for each sample the parameters got optimized again via cross validation.

As one can see above in Figure 3, the Random Forest performs strictly better than the OLS for any sample size. Due to this DGP resembling a stratification similar to that of a Descision Tree, the RSS of the Random Forest converges relatively quickly while that of the OLS remains unstable and high.

7.1.2 Real data example

As previously mentioned, we applied the Random Forest on the Titanic data set [13] in order to determine the survival of the passengers based on reported attributes like name title or booked cabin. In order to use the data to its fullest extent, we conducted additional feature engineering. Without that, many features remain unusable for our methods, because they contain missing values or values that are formatted as text. For the implementation of the feature engineering and the classification, one can consult our code repository [12]. The Random Forest managed to achieve a total classification accuracy of 84,32% on the holdout set. The holdout set consists of 15% of the total data. According to the confusion matrix in Figure 4, for passengers that died, the accuracy was slightly higher compared to those that survived. This is to be expected, since deaths outnumber survivals considerably.

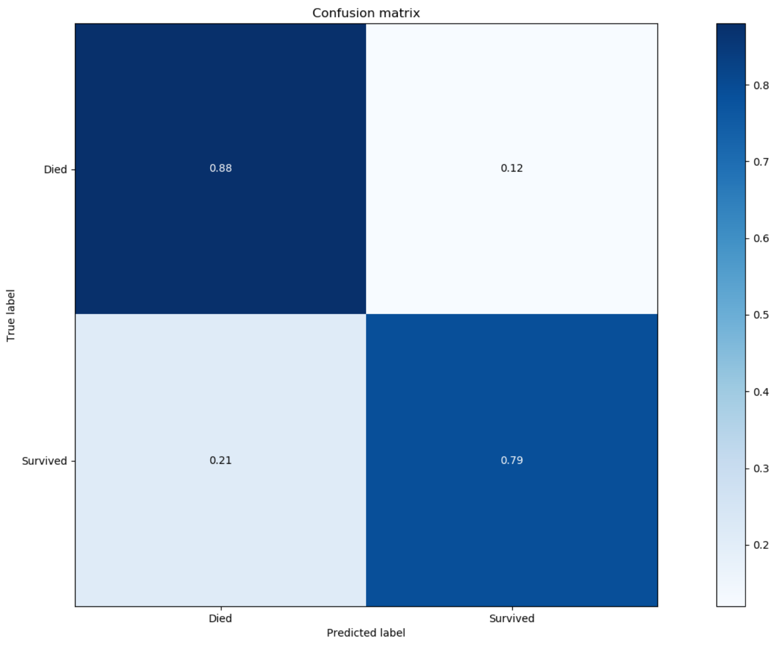

Figure 4: This plot illustrates the accuracy of the Random Forest's prediction on the Titanic data set.

8 Comparison to two boosting methods

Before introducing AdaBoost and Gradient Boosting methods firstly, it is essential to understand what boosting is. Like bagging, boosting is an approach which can be applied to many machine learning methods for classification or regression. Bagging uses bootstrap to create multiple datasets for training the method. As a next stage bagging fits a separate decision tree to each training dataset, and then it combines all decision trees in order to create a single predictive model. Every decision tree is independent to others thanks to using bootstrap to create different training datasets. Boosting method works similarly, but in our case decision trees are grown sequentially. It means that each tree is built using information from previously built trees. Boosting method does not involve bootstrap sampling. In this method instead of bootstrap each decision tree is fit on a modified version of the original dataset.

8.1 AdaBoost Classifier

8.1.1 An introduction to the AdaBoost method

Adaptive Boosting (Adaboost) was introduced by Freund and Schapire (1997) and we employ Adaboost as a comparison technique against Random Forest. In the Adaboost algorithm, instead of fully-grown trees as in Random Forest, we use trees with only one internal node and two leaves also called stumps. We consider those as weak-learners compared to trees since its depth and power is limited. Before generating stumps, we assign weights to observations in the sample, normally the weight of each observation takes the value of $1/N$ as N is the sample size. We generate a stump for each classifier in the data and compare them regarding their misclassification rate. After selecting the best classifier, with using its stump's misclassification rate we compute its stump's significance. With that significance, we compute new weights for the sample. We repeat the algorithm sequentially for the sample with new weights until a stopping criterion is achieved. Generally, using the number of classifiers as the number of iteration is a common practice (Elements of statistical learning). After generating multiple stumps, we can predict an observation's class. We get the decision of every stump and form groups accordingly. Every outcome class has a group of stumps which predict that class and every stump has its significance. After summing significance values of stumps in each group, the prediction is the class with the total highest significance. Although stumps are weak-learners, we exploit the error of a weak-learner to generate another weak-learner and iterating multiple times provides us with a powerful algorithm.

8.1.2 Real Data Application of Adaboost

When we employ Adaboost to predict the survival outcome of the passenger on Titanic, we achieve approximately a rate of 82% accuracy. With utilizing the confusion matrix of Adaboost and compare it with Random Forest's, we can see that for the non-survivals the result is almost the same but Random Forest is better when it comes to identify survivors.

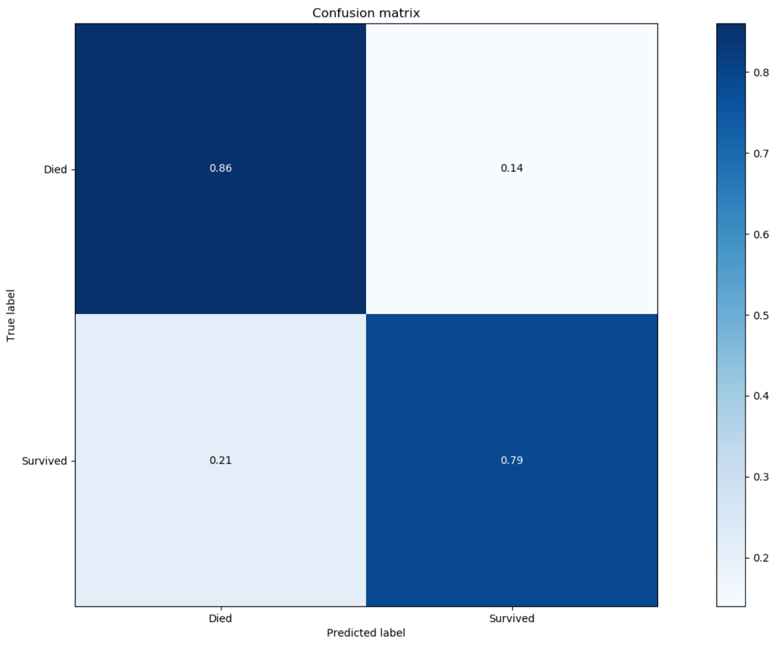

Figure 5: This plot illustrates the accuracy of the Adaboost's prediction on the Titanic data set.

8.2 Gradient Boosting Classifier

8.2.1 An introduction to the Gradient Boosting method

In Gradient Boosting the idea is to take weak learning algorithm or hypothesis and make some corrections that will improve the power of this algorithm/hypothesis. In hypothesis boosting, you check every observation on which statistical learning method was trained on, then you leave the observations which were correctly classified. Then method creates new weak learner and test it only on the observations that were poorly classified. Next, the examples that were correctly classified are kept. The idea described above was used in the AdaBoost (Adaptive Boosting) algorithm. In this algorithm, many weak learners are created thanks to many decision trees that only have a single split. Created instances in the training dataset are weighted in the way that larger weights are assigned to instances which were difficult to classify. To the most difficult training instances weaker learners are added sequentially. Gradient boosting classifiers are the Adaptive Boosting method, but it is combined with weighted minimization. After weighted minimization, all classifiers and weighted inputs are again calculated. The aim of gradient boosting classifiers is to minimize the loss and it operates in the similar way, as gradient descent in a neural network.

8.2.2 Real data example

After a simple and short introduction it is time to apply Gradient Boosting Classifier on the Titanic data set. As in every application in the paper, after conducting feature engineering, we try to determine the survival of the passengers using available explanatory variables. Gradient Boosting Classifier if it comes to a total classification accuracy is slightly worse than the Random Forest method. In this case achieved accuracy is equal approximately 82,84%. As for the Random Forest, the confusion matrix in Figure 6 shows higher accuracy for passengers that died than passengers who survived and it is respectively 85% and 80%.

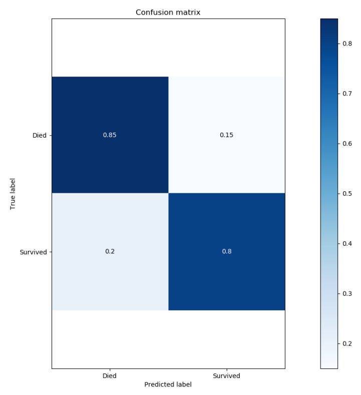

Figure 6: This plot illustrates the accuracy of the Gradient Boosting Classifier's prediction on the Titanic data set.