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

February 1, 2020

# Contents

# 1 Introduction

Drawing conclusions from data and utilizing it to get predictions is a common practice in Economics alongside with many other fields. As a non-parametric estimation tool, decision trees attract attention in the literature yet we require robust methods to correctly identify patterns in data and to obtain accurate predictions and decision trees suffer from high variance [12]. For prediction purposes having high variance is a crucial problem, thus, several improvements were proposed such as bootstrap aggregation, boosting and most importantly Random Forests which is our focus in this paper. Fundamentally Random Forests are an ensemble of decision trees which are grown from randomly sampled data with randomly selected explanatory variables. Although Random Forest is also capable in regression type problems, from our perspective it gives an account of itself in classification setting and we scale down our focus to classification settings meaning that dependent variable in data is categorical.

We start with explaining the decision trees since the decision tree is the building block and the starting point of Random Forest. We concentrated on the tree-building process with differing splitting criteria rather than pruning as Random Forest uses fully grown trees and does not prune trees. After delving into the variance problem, we describe as one of the solutions, bootstrap aggregating (bagging) because Random forest utilizes the main principle of bagging, bootstrapped data. In the next section, we start with defining Random Forest and discuss the main root of randomness and mention the idea of Out of Bag sample which stems from the usage of bootstrapping. There are two different class determination methods available theoretically in Random Forest. We start the Mathematical Explanation part with defining those voting processes. Although primarily we use soft voting, understanding majority voting provides us with a better insight. Then we define a measure of the decision tree's fit and decompose it to understand the improvement of Random Forest over decision trees. With exploiting bias-variance decomposition of that measure and expanding our findings to Random Forest, we exhibit the working principle of Random Forest. In the next section, we examine variable importance as Random Forest enables us to measure how much each independent variable is important. Finally, we applied the Random Forest algorithm to simulated and real data. In the simulation study, we employed linear and non-linear data generation processes and compared Random Forest's performance with linear regression. In real data study, we used Titanic data [22] and compared Random Forest's performance with two boosting methods; Adaptive and Gradient Boosting.

In the literature, decision trees are covered by several published works including [2] and [12]. As a solution to the problem of high variance in decision trees, Leo Breiman introduced bagging to add randomness to the model with randomly sampling the data[3]. Subsequently, Random Forest is introduced as an extention with also embedding randomness in the variable selection process[6]. While [12] being the main source regarding intuition, Louppe provides us with a re-assesment of published works and insight in mathematical aspects of Random Forest[18]. The bias-variance decomposition and examination of improvement of Random Forest upon decision trees utilizes [15][9][11][17].

# 2 Decision tree

## 2.1 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.

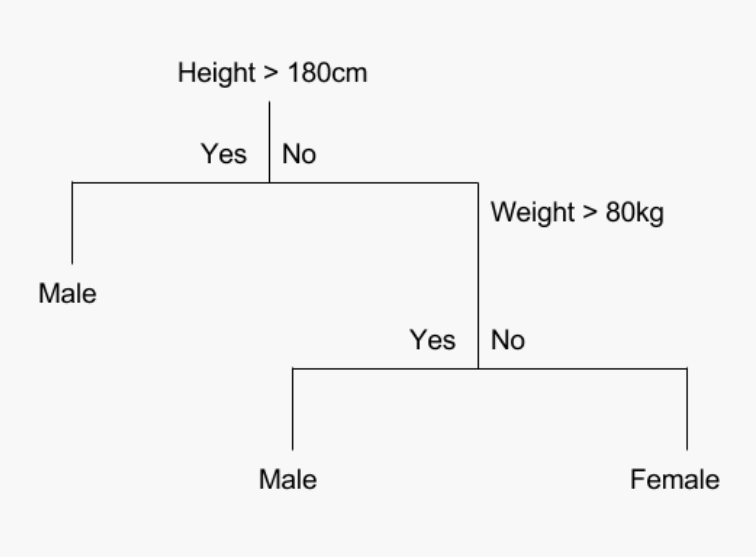


Figure 1: Given a data set with two features height and weight, and gender as the target variable, this example tree stratisfies the two-dimensional feature space into three distinct subset each represented by the terminal nodes at the bottom. The stratification occurs at the two deciding nodes depending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

## 2.2 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.

### 2.2.1 Splitting criteria

Since we are only concerned with classification, $Y$ is therefore categorical. The original CART algorithm uses Gini and Twoing as purity measures for the splitting criterion [2]. However, implementations of the algorithm such as Python's sklearn package [20] also contain entropy and misclassification rate as measures of impurity.

For a given learning sample $D = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ for a $C$ class problem, let $N_c$ be the number of instances $\{x, y\}$ belonging to in class $c$.

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

At node $t$, let the probabilities $p(c,t)$ be estimated by

$$p(c,t) = \frac{\pi(c)N_c(t)}{N_c}. \tag{1}$$

This represents the probability that an 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

$$p(t) = \sum_{c \in C} p(c,t), \tag{2}$$

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

$$p(c|t) = \frac{p(c,t)}{p(t)} = \frac{p(c,t)}{\sum_{c \in C} p(c,t)}. \tag{3}$$

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

$$\sum_{c \in C} p(c|t) = 1 \tag{4}$$

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

$$i(t) = \sum_{c \in C} p(c|t)(1 - p(c|t)) = 1 - \sum_{c \in C} p_c^2 \tag{6}$$

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 $|C| = 2$, $i(t)$ will be maximized by $p(c|t) = 0.5$ for $c \in C$.

**Information Entropy**

The Entropy measure from information theory is defined as

$$i(t) = \sum_{c \in C} p(c|t) log(p(c|t)) \tag{7}$$

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

$$i(t) = 1 - \max_{c \in C} p(c|t). \tag{8}$$

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

### 2.2.2 Bias-variance trade-off

We can measure the the performance of a decision tree by assessing its fit to data. To measure the fit of a model to data, the generalization error is a widely used concept [6]. 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 [13], 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.

## 2.3 Bagging

Decision trees belong to methods which are sensitive to the specific data on which they are trained. 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 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. It was also described by L. Breiman in his work [3]. 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. The fundamental difference between bagging and Random Forest is that in Random Forest injection of randomness is added. This method and mentioned injection of randomness in it will be described in the next section of paper.

# 3 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 [5], [6], and [4] 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.

## 3.1 Main idea and illustration

An ensemble of randomly trained decision trees, so in other words random decision forest was defined by [6]:

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

---

**Algorithm 1:** Random Forest for Regression or Classification [12]

---

1. For $b = 1$ to $B$:
   a) Draw a bootstrap sample $D_b$ of size N from the training data.
   b) Grow the Random Forest tree $T_{D_b,\theta_b}$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached:
      i. Select $m$ variables denoted by $\theta_b$ at random from the $n$ variables
      ii. Pick the best variable/split-point among the $m$
      iii. Split the node into two daughter nodes
2. Output the ensemble of trees $\{T_{D_b,\theta_b}\}_{b=1}^{B}$

---

### 3.1.1 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.

As it is mentioned in [12], the inventors recommend following numbers:

1. For classification: $\lfloor\sqrt{n}\rfloor$ and the minimum node size is one
2. For regression: $\lfloor\frac{n}{3}\rfloor$ and the minimum node size is five

where $n$ is a number of features in the classification or regression problem.

This yields that in classification problem with 10 different variables, only 3 randomly

chosen will be considered for splitting the node. These numbers mentioned above are only the recommendations made by inventors and in practise the best values for these parameters can be very different and it will depend on the problem. Therefore, these parameters should be treated as tuning parameters. 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 [7].

### 3.1.2 Training, testing and prediction

Training and testing is an integral part of building up every machine learning model. In Random forest 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. As written in [8], 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:

$$p(c|\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} p_b(Y = c|X = \mathbf{x}) \tag{9}$$

where in a random forest with the number of decision trees equals to $B$ each tth tree obtains the posterior distribution $p_b(Y = c|X = \mathbf{x})$.

### 3.1.3 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 [12]. Out of Bag is commonly used to monitor error of predictions. Breiman's work [5] shows empirically that Out of Bag estimate is as accurate as a test set of the same size as the training set [5].

## 3.2 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 $\theta$. 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 [17]. As mentioned earlier, Random Forest classifier selects a bootstrapped subset of observations and grow the decision tree with only a subset of regressors. Repeating this tree growing

process $B$ times yields a Random Forest. 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 [18] and [24].

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

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

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:

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

As mentioned in [3], the two aforementioned 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 [18]. We introduced majority voting to enhance our understanding and will use soft voting in further derivations.

### 3.2.1 Properties

Having explained the details of algorithm, we can explicitly show the improvement of Random Forest upon decision trees with exploiting the generalization error. We can measure of a model's fit with the generalization error which also called test error or the expected prediction error, we will start with examining a decision tree's generalization error and expand our finding to Random Forest. The derivations and findings in this section closely follow Louppe's paper [18]. The generalization error of a decision tree $T_{D,\theta}$ which is grown using the set $D$ and parameters $\theta$ 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 jointly. The bias-variance decomposition of both functions are similar and follow the same dynamics [9]. While the bias-variance decomposition of zero-one loss function remains to be relatively unexplanatory, the squared loss provides us a superior insight. Therefore we denoted the findings with using the zero-one loss function with apostrophe. The squared loss function measures the squared difference between the dependent variable and its predicted value by decision tree $T_{D,\theta}$ and can be defined as

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

while the zero-one loss function yields is

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

$L(Y, T_{D,\theta}(X))$ equals to 1 if the class of dependant variable is different than its predicted class by decision tree and equals to 0 if both are the same, meaning that $Y = T_{D,\theta}(X)$. Both loss functions follow the same logic in essence. While the squared loss function yielding a number to measure the error, the zero-one loss function only yields 1 or 0. The generalization 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 expression in the first equation measures the squared difference of true and predicted values and the expression in the second equation is the probability of misclassification of the tree. We will decompose the expected generalization error and investigate the improvement of random forest upon decision trees.

***The Decomposition of Err***$(T_{D,\theta})$
Given the probability distribution of P(X,Y), there exists a model $\phi_\beta$ that minimizes the generalization error and can be derived analytically independent or learning set $D$ [18]. With conditioning on X generalization error for $\phi_\beta$ becomes;

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

Point-wise minimization inner term with respect to Y yields [18];

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

$\phi_\beta$ 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[18]. 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 expected value of Y at X=x since the mean of $Y$ minimizes the squared difference and equals to its expected value. Bayes Model of zero-one loss function denoted as $\phi'_\beta$ 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 class with highest probability in the set $Y$ is chosen by Bayes Model when using the zero-one loss function, we included intermediate steps in section 7.1. Aforementioned 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

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

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 generalization 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 section 7.2, the first term in equation (24) also called squared bias ($bias^2$) shows how the expected prediction of our decision tree differs from Bayes Model and the latter term is the variance of our estimator. Therefore, we can define $\boldsymbol{Err}(T_{D,\theta})$ as follows

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

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

The same decomposition can be conducted for the zero-one loss function in similar fashion and as mentioned in [18],[9], [15], [11] both zero-one and squared loss functions decompositions yield same results. However, without assuming $D$ is normally distributed, bias-variance decomposition cannot be solved for zero-one loss function explicitly as done for squared loss [18]. [17] introduces another decomposition for zero-one loss function (included in section 7.3), 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.

***Extending findings to Random Forest***

In regression setting, random forest shares the same idea with classification prediction with soft voting. Random forest for regression setting 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

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

where $\mu_{D,\theta}(x)$ is the average prediction of all ensembled trees. Since $\theta$'s are random, independent and have the same distribution [18], when we extend this finding bias of a random forest we can state that

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

Intiutively, when we get the average prediction of all ensembled trees, we are able to consider $\mu_{D,\theta}(x)$ as one decision tree in simplest terms and make this inference. We can interpret equation (28) as the squared bias cannot be decreased with ensembling randomized models, namely, an ensemble of trees does not guarantee having lower bias compared to only one tree [12]. Although random forest is inadequate to propose any structure to decrease the generalization error so far regarding *noise* and $bias^2$, it displays promising performance in reducing the last remaining part of the generalization error. Thus, we can continue our exploration with variance of random forest, yet, we need to define the correlation coefficient $\rho(x)$ before delve into variance since the correlation coefficient have significant role in variance. For any two trees $T_{D,\theta'}$ and $T_{D,\theta''}$ trained with the same data $D$ and different growing parameters $\theta'$ and $\theta''$, 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 $\theta'$ and $\theta''$ following the same distribution in the intermediate steps in section 7.4. $\sigma_{D,\theta}(x)$ is the variance of a single decision tree and associated with prediction variability stemming from randomness of the set $D$ and randomness introduced with $\theta$ [18]. Therefore, $\rho(x)$ represents the effect of randomization in the learning algorithm in general. 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, trees are non-correlated and the prediction of trees are perfectly random in the sense that not dependent on the predictions of other trees. We can decompose variance of a random forest as follows:

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

We included derivations in detail in section 7.5. 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 generalization error of a random forest is less than

the generalization 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 $\sigma^2_{D,\theta}(x)/B$ which can be decreased with increasing $B$ as mentioned. Conclusively, Random Forest improves the performance of decision tree with decreasing variance and keeping bias unaffected.

## 3.3 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. Random Forest allows 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 based on work by [19] and [1].

### 3.3.1 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 [6]. 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 [6] is given by:

$$\widehat{MDI}(X_j) = \frac{1}{B}\sum_{b=1}^{B}\sum_{t\in T_b : v(s_t)=X_j} p(t)\Delta i(s_t,t) \tag{31}$$

where $v(s_t)$ 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 $p(t)\Delta i(s_t,t)$ for all nodes $t$ where $X_j$ is used. Mean in the name of MDI comes from averaging over all $B$ 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.

# 4 Application and Comparison

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

In section 4.1, we apply the random forest on simulated data, and show how its performance develops over increasing sample sizes.

Then in section 4.2, we apply the random forest on the real Titanic data set [22] and evaluate its performance.

In section 5.1 and 5.2, we apply AdaBoost and Gradient Boosting respectively on the Titanic data set and compare their performance with that of the random forest.

## 4.1 Application of Random Forest on 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:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon, \tag{32}$$

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 2 and 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.

As one can see in figure 2, 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:

$$y = \beta_0 + \beta_1 I(x_1 >= 0, x_2 >= 0) + \beta_2 I(x_1 >= 0, x_2 < 0) + \beta_3 I(x_1 < 0) + \epsilon, \tag{33}$$

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

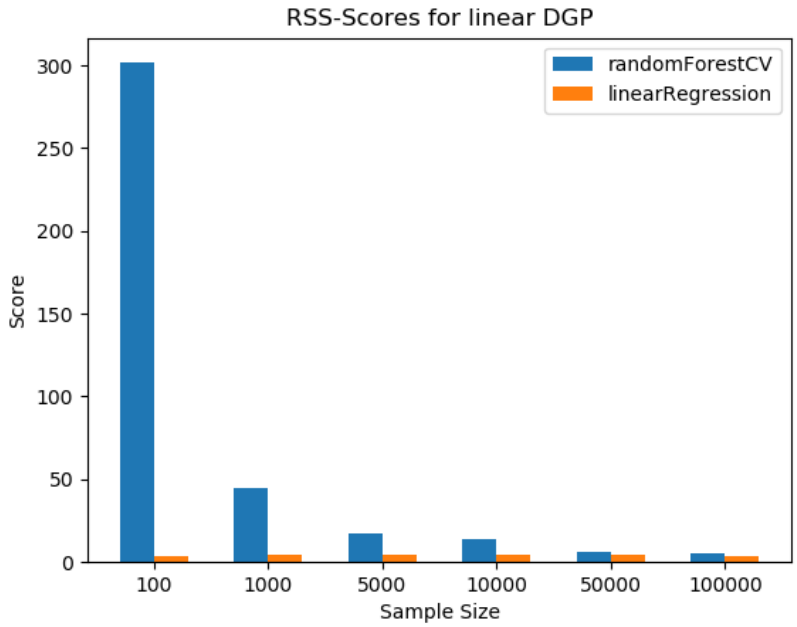


Figure 2: This plot illustrates the RSS for different training sample sizes for Random Forest and OLS. These samples where drawn from a linear DGP in accordance to equation (32). The holdout set for calculating the RSS were drawn again for each training sample from the same DGP. It always contained 100 observations. In case of the Random Forest, for each sample the parameters got optimized again via cross validation.

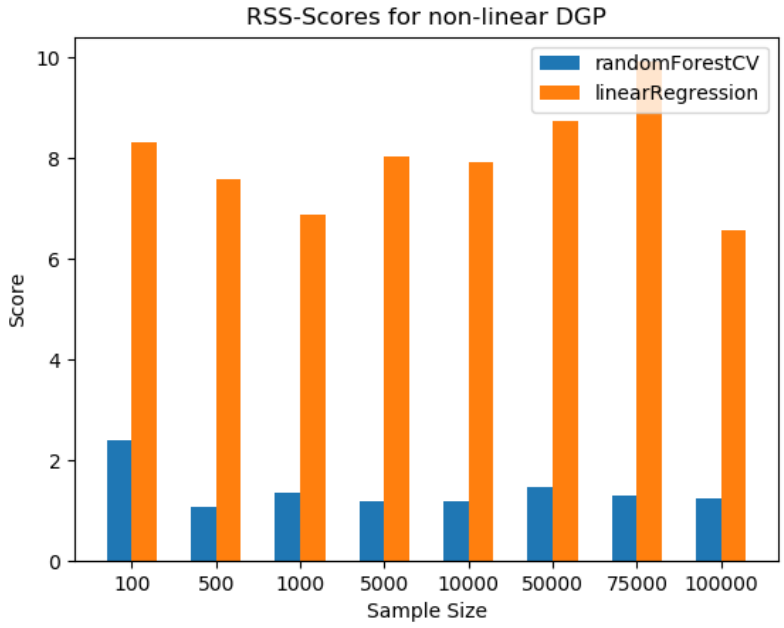


Figure 3: This plot illustrates the RSS for different training sample sizes for Random Forest and OLS. These samples where drawn from a non-linear DGP in accordance to equation (33). The holdout set for calculating the RSS were drawn again for each training sample from the same DGP. It always contained 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.

## 4.2 Application of Random Forest on real data

As previously mentioned, we applied the Random Forest on the Titanic data set [22] in order to determine the survival of the passengers based on reported attributes like name title or booked cabin. A major reason for choosing this data set was due to the fact that it consists of many categorical features. 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. Further, we split up the data set randomly into five folds via the K-fold method. Then, the random forest got applied on each of the folds to obtain the classification rates. Finally, we averaged these results to get a more representative performance evaluation of the Random Forest. For the implementation of the feature engineering and the classification, one can consult our code repository [21]. The Random Forest managed to achieve a total classification accuracy of 82.71% on the holdout set. 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.

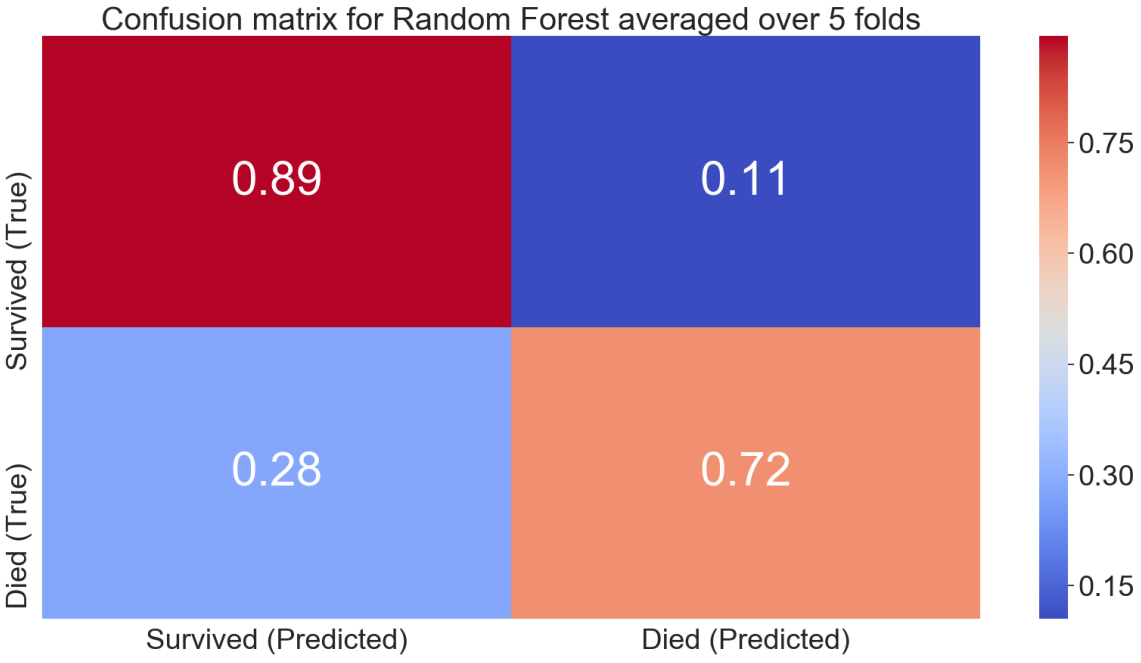


Figure 4: This plot illustrates the accuracy of the Random Forest's prediction on the Titanic data set. The data set got split into five folds on which the Random Forest got applied. Then, the resulting combination of classification errors got averaged. The left axis indicates the true class membership, while the bottom one indicates the predicted one.

# 5 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 [16].

## 5.1 AdaBoost Classifier

### 5.1.1 An introduction to the AdaBoost method

Adaptive Boosting (Adaboost) was introduced by [10] 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 [12]. 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.

### 5.1.2 Real Data Application of Adaboost

The application of Adaboost is analogous to that of the Random Forest. That means that it got applied on the same five folds from the data set and its results got averaged. Thus, when we employ Adaboost to predict the survival outcome of the passenger on Titanic, we achieve a mean classification rate of 82.94% 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.

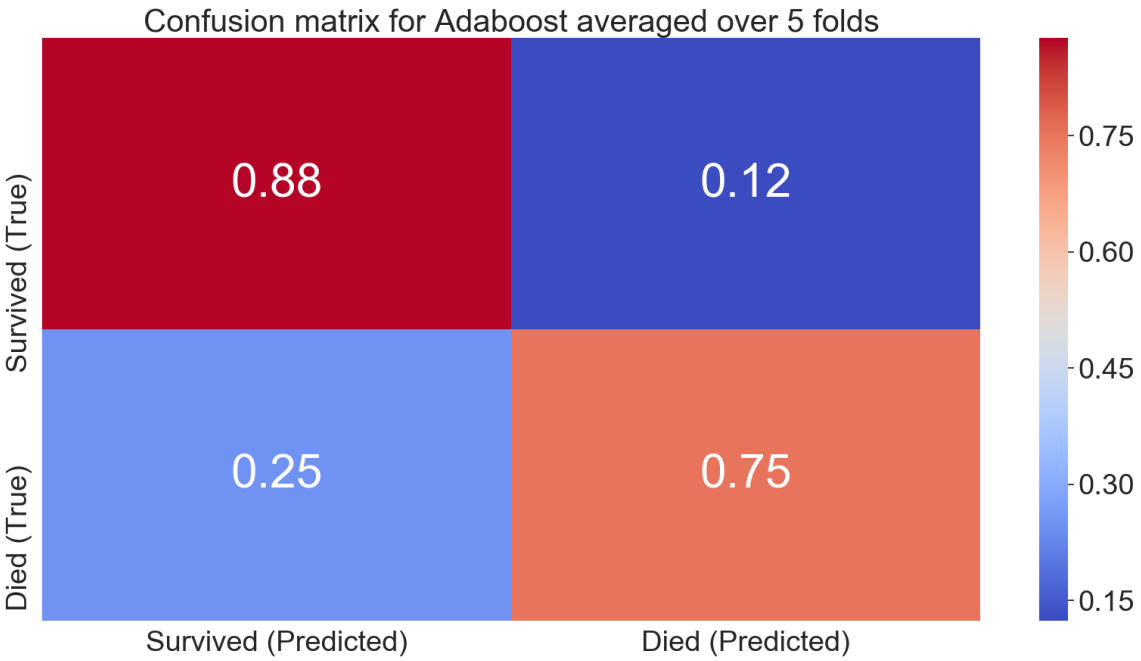


Figure 5: This plot illustrates the accuracy of the AdaBoost's prediction on the Titanic data set. The data set got split into five folds on which the Random Forest got applied. Then, the resulting combination of classification errors got averaged. The left axis indicates the true class membership, while the bottom one indicates the predicted one.

## 5.2 Gradient Boosting Classifier

### 5.2.1 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, we 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 algorithm. In this algorithm, many weak learners are created by 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.

### 5.2.2 Real data example

The application of Gradient Boosting is analogous to that of the other methods. That means that it got applied on the same five folds from the data set and its results got averaged. Thus, when we employ Gradient Boosting to predict the survival outcome of the passenger on Titanic, we achieve a mean classification rate of 82.15% accuracy.

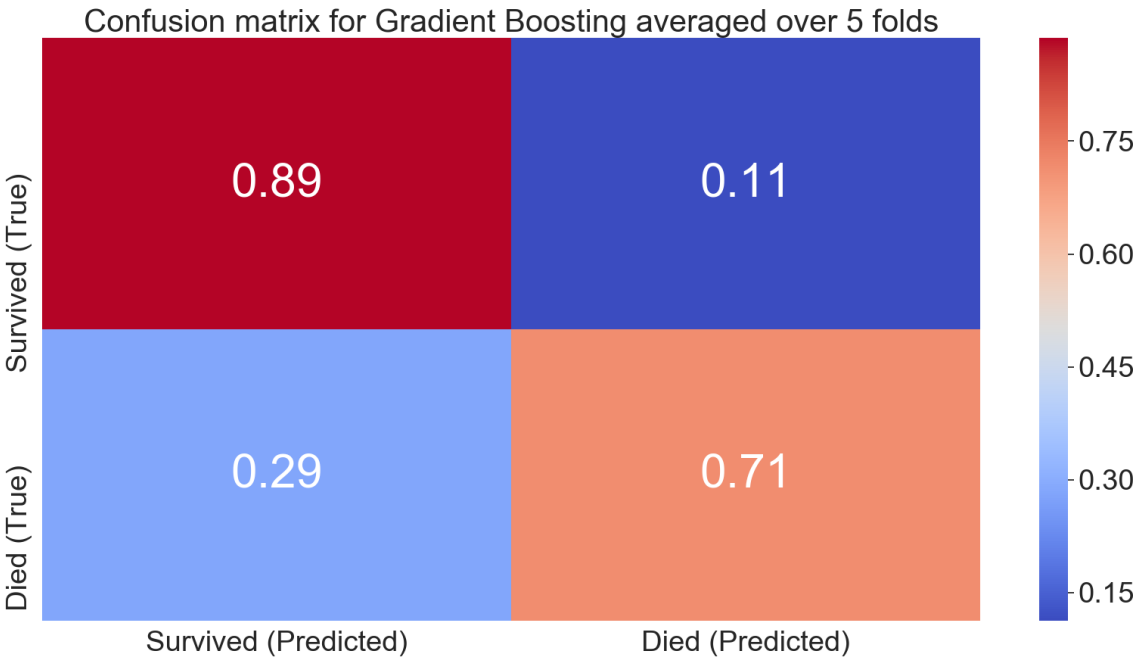


Figure 6: This plot illustrates the accuracy of the Gradient Boosting's prediction on the Titanic data set. The data set got split into five folds on which the Random Forest got applied. Then, the resulting combination of classification errors got averaged. The left axis indicates the true class membership, while the bottom one indicates the predicted one.

# 6 Conclusion and outlook

We examined and applied one of the infamous machine learning algorithms, Random Forest. We started with explaining the decision trees and the room for improvement in it. As solutions of high-variance in decision trees, bagging, boosting and Random Forest are developed. Random Forest shows better predictions can be achieved with introducing randomness into the picture. That randomness provides us with a decorrelated ensemble of trees and the increase in the number of trees yields lower error considering prediction purposes. With using Random Forest, we are able to assess the importance of every variable and draw conclusions about data. We explained the intuition of Random Forest in detail and included mathematical clarification. Finally, we applied Random Forest on both simulated and real data and compared with various methods. Considering results, Random Forest appears to be employed in the future as well, thus, a better understanding of the idea and the dynamics can give us a chance to improve. This paper aims to introduce the idea in detail and essentially a review and a showcase of the Random Forest algorithm.

# 7 Appendix

## 7.1 Bayes Model explanation

We identified Bayes Model in equation (20) as such

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

## 7.2 Bias-variance decomposition of Squared Loss Function

We derived the result in equation (23) as follows;

$$\begin{aligned}\boldsymbol{Err}(T_{D,\theta}(x)) &= \mathbb{E}_{Y|X=x}\{(Y - T_{D,\theta}(x))^2\} \\ &= \mathbb{E}_{Y|X=x}\{(Y - \phi_\beta(x) + \phi_\beta(x) - T_{D,\theta}(x))^2\} \\ &= \mathbb{E}_{Y|X=x}\{(Y - \phi_\beta(x))^2\} + \mathbb{E}_{Y|X=x}\{(\phi_\beta(x) - T_{D,\theta}(x))^2\} \\ &\quad + \underbrace{\mathbb{E}_{Y|X=x}\{2(Y - \phi_\beta(x))(\phi_\beta(x) - T_{D,\theta}(x))\}}_{=0 \ \text{ since } \mathbb{E}_{Y|X=x}(Y-\phi_\beta(x))=0 \text{ from (18)}} \\ &= \underbrace{\mathbb{E}_{Y|X=x}\{(Y - \phi_\beta(x))^2\}}_{\text{from (20) equals to } \boldsymbol{Err}(\phi_\beta(x))} + \mathbb{E}_{Y|X=x}\{(\phi_\beta(x) - T_{D,\theta}(x))^2\} \\ &= \boldsymbol{Err}(\phi_\beta(x)) + (\phi_\beta(x) - T_{D,\theta}(x))^2\end{aligned}$$

We adopted following steps in the futher derivations of bias-variance decomposition in equation (24).

$$\begin{aligned}&\mathbb{E}_D\{(\phi_\beta(x) - T_{D,\theta}(x))^2\} \\ &= \mathbb{E}_D\{(\phi_\beta(x) - \mathbb{E}_D\{T_{D,\theta}(x)\} + \mathbb{E}_D\{T_{D,\theta}(x)\} - T_{D,\theta}(x))^2\} \\ &= \mathbb{E}_D\{(\phi_\beta(x) - \mathbb{E}_D\{T_{D,\theta}(x))^2\} + \mathbb{E}_D\{(\mathbb{E}_D\{T_{D,\theta}(x)\} - T_{D,\theta}(x))^2\} \\ &\quad + \mathbb{E}_D\{2(\phi_\beta(x) - \mathbb{E}_D\{T_{D,\theta}(x)\})(\mathbb{E}_D\{T_{D,\theta}(x)\} - T_{D,\theta}(x))\} \\ &\text{since } \mathbb{E}_D\{\mathbb{E}_D\{T_{D,\theta}(x)\} - T_{D,\theta}(x)\} = \mathbb{E}_D\{T_{D,\theta}(x)\} - \mathbb{E}_D\{T_{D,\theta}(x)\} = 0 \\ &= \mathbb{E}_D\{(\phi_\beta(x) - \mathbb{E}_D\{T_{D,\theta}(x))^2\} + \mathbb{E}_D\{(\mathbb{E}_D\{T_{D,\theta}(x)\} - T_{D,\theta}(x))^2\} \\ &= (\phi_\beta(x) - \mathbb{E}_D\{T_{D,\theta}(x))^2 + \mathbb{E}_D\{(\mathbb{E}_D\{T_{D,\theta}(x)\} - T_{D,\theta}(x))^2\}\end{aligned}$$

## 7.3 Kohavi's decomposition of zero-one loss function

Kohavi proposed an alternative decomposition for zero-function [17]. Our notation differs in some extent from Kohavi's paper to be coherent with out prior findings. The zero-one

loss function is defined as

$$L(\phi_\beta(x), T_{D,\theta}(x)) = 1 - \delta(\phi_\beta(x), T_{D,\theta}(x))$$

where $\delta(\phi_\beta(x), T_{D,\theta}(x)) = 1$ if $\phi_\beta(x) = T_{D,\theta}(x)$ and 0 otherwise. The generalization error (misclassification rate in the paper) is defined and extented as

$$\begin{aligned}
\boldsymbol{Err}(T_{D,\theta}(x)) &= L(\phi_\beta(x), T_{D,\theta}(x) P(\phi_\beta(x), T_{D,\theta}(x)) \\
&= \sum_{\phi_\beta(x), T_{D,\theta}(x)} [1 - \delta(\phi_\beta(x), T_{D,\theta}(x))] P(\phi_\beta(x), T_{D,\theta}(x)) \\
&= 1 - \sum_{y \in Y} P(\phi_\beta(x) = T_{D,\theta}(x)) = y) \qquad (34)
\end{aligned}$$

Even if in the Kohavi's paper it is stated that the last step is a simplified version of the extended Bayesian Formalism [23], there is not enough explanation to explicitly show the mathematical derivation nor the intuition. If we continue examine the equation (34) we get the following decomposition;

$$\begin{aligned}
\boldsymbol{Err}(T_{D,\theta}(x)) &= 1 - \sum_{y \in Y} P(\phi_\beta(x) = T_{D,\theta}(x)) = y) \\
&= \sum_{y \in Y} -P(\phi_\beta(x) = T_{D,\theta}(x)) = y) + \sum_{y \in Y} P(\phi_\beta(x) = y) P(T_{D,\theta}(x)) = y) \\
&+ \sum_{y \in Y} [-P(\phi_\beta(x) = y) P(T_{D,\theta}(x)) = y) + \frac{1}{2} P(T_{D,\theta}(x) = y)^2 + \frac{1}{2} P(\phi_\beta = y)^2] \\
&+ [\frac{1}{2} - \frac{1}{2} \sum_{y \in Y} P(T_{D,\theta}(x) = y)^2] + [\frac{1}{2} - \frac{1}{2} \sum_{y \in Y} P(\phi_\beta(x) = y)^2] \qquad (35)
\end{aligned}$$

Rearranging equation (35) yields

$$\begin{aligned}
\boldsymbol{Err}(T_{D,\theta}(x)) &= \sum_{y \in Y} [P(T_{D,\theta}(x) = y) P(\phi_\beta(x) = y) - P(T_{D,\theta}(x) = \phi_\beta(x) = y)] \qquad (*) \\
&+ \frac{1}{2} \sum_{y \in Y} [P(T_{D,\theta}(x) = y) - P(\phi_\beta(x) = y)]^2 \\
&+ \frac{1}{2} (1 - \sum_{y \in Y} P(T_{D,\theta}(x) = y)^2) + \frac{1}{2} (1 - \sum_{y \in Y} P(\phi_\beta(x) = y)^2) \\
& \qquad (36)
\end{aligned}$$

The $*$ term is the covariance between Bayes Model and decision tree which equals to zero since by construction Bayes Model cannot be dependent of any model. Then the decomposition becomes;

$$\boldsymbol{Err}(T_{D,\theta}(x)) = \sum_x P(x)(\boldsymbol{Err}(\phi_\beta) + bias^2 + variance)$$

where

$$\boldsymbol{Err}(\phi_\beta) = \frac{1}{2}(1 - \sum_{y \in Y} P(\phi_\beta(x) = y)^2) \qquad \text{(Bayes Error)}$$

$$bias^2 = \frac{1}{2}\sum_{y \in Y}[P(T_{D,\theta}(x) = y) - P(\phi_\beta(x) = y)]^2$$

$$variance = \frac{1}{2}(1 - \sum_{y \in Y} P(T_{D,\theta}(x) = y)^2)$$

Since the outcome of Kohavi's decomposition is not as explanatory as our prior findings and without assuming a certain type of distribution for the sample [18], we wanted to explain mathematical dynamics of random forest with using the squared loss function alongside zero-one loss function.

## 7.4 Correlation Coefficient

Using the definition of the Pearson's correlation coefficient and the property of $\theta'$ and $\theta''$ following the same distribution, we derived correlation coefficient as such

$$\begin{aligned}
\rho(x) &= \frac{\mathbb{E}_{D,\theta',\theta''}\{(T_{D,\theta'}(x) - \mu_{D,\theta'}(x))(T_{D,\theta''}(x) - \mu_{D,\theta''}(x))\}}{\sigma_{D,\theta'}(x)\sigma_{D,\theta'}(x)} \\
&= \frac{\mathbb{E}_{D,\theta',\theta''}\{T_{D,\theta'}(x)T_{D,\theta''}(x) - T_{D,\theta'}(x)\mu_{D,\theta''}(x) - T_{D,\theta''}(x)\mu_{D,\theta'}(x) + \mu_{D,\theta'}(x)\mu_{D,\theta''}(x))\}}{\sigma^2_{D,\theta}(x)} \\
&= \frac{\mathbb{E}_{D,\theta',\theta''}\{T_{D,\theta'}(x)T_{D,\theta''}(x)\} - \mu^2_{D,\theta}(x)}{\sigma^2_{D,\theta}(x)}
\end{aligned}$$

With an alternative decomposition, we can state that $\rho(x)$ is non-negative [18] [14]. Being that said the findings we explored with decomposing the variance of the squared loss function are robust.

## 7.5 Decomposition of Variance

The derivation of equation (30) is as follows;

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

Using $\mathbb{V}\{aX\} = a^2\mathbb{V}\{X\} = a^2(\mathbb{E}\{X^2\} - \mathbb{E}\{X\}^2$, variance of random forest equals to

$$\mathbb{V}\{\boldsymbol{RF}\} = \frac{1}{B}\left[\mathbb{E}_{D,\theta_1,\theta_2,\ldots,\theta_B}\{(\sum_{b=1}^{B} T_{D,\theta_b}(x))^2\} - \mathbb{E}_{D,\theta_1,\theta_2,\ldots,\theta_B}\{(\sum_{b=1}^{B} T_{D,\theta_b}(x))\}^2\right]$$

Following Louppe's derivations, we can rewrite the variance as pairwise products of any two trees using parameters $\theta_i$ and $\theta_j$;

$$\mathbb{V}\{\boldsymbol{RF}\} = \frac{1}{B}[\mathbb{E}_{D,\theta_1,\theta_2,\ldots,\theta_B}\{\sum_{i,j} T_{D,\theta_i}(x)T_{D,\theta_j}(x)\} - (B\mu_{D,\theta}(x))^2]$$

where $\mu_{D,\theta}$ is the average prediction of all ensembled trees and derived in equation (27).

$$\begin{aligned}
\mathbb{V}\{\boldsymbol{RF}\} &= \frac{1}{B^2}\left[\sum_{i,j}\mathbb{E}_{D,\theta_i,\theta_j}\{T_{D,\theta_i}(x)T_{D,\theta_i}(x)\} - B^2\mu^2_{D,\theta}(x)\right] \\
&= \frac{1}{B^2}\left[B\mathbb{E}_{D,\theta}\{T_{D,\theta}(x)^2\} + (B^2-B)\mathbb{E}_{D,\theta_i,\theta_j}\{T_{D,\theta_i}(x)T_{D,\theta_i}(x)\} - B^2\mu^2_{D,\theta}(x)\right] \\
&= \frac{1}{B^2}\left[B(\sigma^2_{D,\theta}(x) + \mu^2_{D,\theta}(x)) + (B^2-B)(\rho(x)\sigma^2_{D,\theta}(x) + \mu^2_{D,\theta}(x)) - B^2\mu^2_{D,\theta}(x)\right] \\
&= \frac{\sigma^2_{D,\theta}(x)}{B} + \rho(x)\sigma^2_{D,\theta}(x) - \rho(x)\frac{\sigma^2_{D,\theta}(x)}{B} \\
&= \rho(x)\sigma^2_{D,\theta}(x) + \frac{1-\rho(x)}{B}\sigma^2_{D,\theta}(x)
\end{aligned}$$

# References


[1] Gerard Biau and Erwan Scornet. "A random forest guided tour". In: *Test* (2016), pp. 1–31. URL: https://hal.sorbonne-universite.fr/hal-01307105.

[2] L. Breiman et al. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984. ISBN: 9780412048418. URL: https://books.google.de/books?id=JwQx-WOmSyQC.

[3] Leo Breiman. "Bagging Predictors". In: *Machine learning* 24 (1996), pp. 123–140. URL: https://doi.org/10.1023/A:1018054314350.

[4] Leo Breiman. "Consistency for a simple model of random forests". In: (2004).

[5] Leo Breiman. "OUT-OF-BAG ESTIMATION". In: (1996).

[6] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[7] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. *Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*. Foundations and Trends® in Computer Graphics and Vision: Vol. 7: No 2-3, pp 81-227. Vol. 7. 2-3. NOW Publishers, Jan. 2012, pp. 81–227. URL: https://www.microsoft.com/en-us/research/publication/decision-forests-a-unified-framework-for-classification-regression-density-estimation-manifold-learning-and-semi-supervised-learning/.

[8] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. "Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning". In: *Foundations and Trends® in Computer Graphics and Vision* 7.2–3 (2012), pp. 81–227. ISSN: 1572-2740. DOI: 10.1561/0600000035. URL: http://dx.doi.org/10.1561/0600000035.

[9] Pedro M. Domingos. "A Unified Bias-Variance Decomposition for Zero-One and Squared Loss". In: (2000). URL: http://www.aaai.org/Library/AAAI/2000/aaai00-086.php.

[10] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: (1997). URL: https://doi.org/10.1006/jcss.1997.1504.

[11] Jerome H. Friedman. "On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality". In: (1997). URL: https://link.springer.com/article/10.1023/A:1009778005914.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[13] Stuart Geman, Elie Bienenstock, and René Doursat. "Neural networks and the bias/variance dilemma". In: *Neural computation* 4.1 (1992), pp. 1–58.

[14] Pierre Geurts. "Contributions to decision tree induction: bias/variance tradeoff and time series classification". PhD thesis. University of Liège Belgium, 2002.

[15] Gareth M James. "Variance and bias for general loss functions". In: *Machine learning* 51.2 (2003), pp. 115–135.

[16] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

[17] Ron Kohavi, David H Wolpert, et al. "Bias plus variance decomposition for zero-one loss functions". In: *ICML*. Vol. 96. 1996, pp. 275–83.

[18] Gilles Louppe. "Understanding random forests". In: *University of Liège* (2014).

[19] Gilles Louppe et al. "Understanding variable importances in forests of randomized trees". In: *Advances in neural information processing systems*. 2013, pp. 431–439.

[20] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[21] *Research Module Application*. URL: https://github.com/RaRedmer/Research_Module_Application.

[22] *Titanic: Machine Learning from Disaster*. URL: https://www.kaggle.com/c/titanic/data.

[23] David H Wolpert. "The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework". In: *The mathematics of generalization*. CRC Press, 2018, pp. 117–214.

[24] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.