University of Bonn

Master of Science in Economics

Random Forest for Classification Problems

Submitted by

Burak Balaban

Arkadiusz Modzelewski

Raphael Redmer

Supervisor: Prof. Dr. Dominik Liebl

February 6, 2020

Abstract

In this paper, we examine the machine learning algorithm Random Forest and present the results of its application. First, we start with an explanation of the Decision Tree. Then, we proceed to show how it can be improved. More specifically, these improvements for the Decision Tree contain bagging, boosting and Random Forest. There, we show that the Random Forest achieves a better prediction accuracy by introducing randomness in tree ensembling. This randomness provides us with a decorrelated ensemble of trees and the increase in the number of uncorrelated trees yields lower error regarding prediction purposes. By using Random Forest, we can assess the importance of every variable and make conclusions about data. Moreover, we explain the intuition of Random Forest in detail including mathematical clarification. Finally, we apply Random Forest on both simulated and real data and compare it with various methods. In conclusion, this paper aims to introduce the relevant concepts in detail and is essentially a review.

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, the Decision Tree attracts attention in the literature, yet we require robust methods to correctly identify patterns in the data and to obtain accurate predictions. However, the Decision Tree suffers from high variance [12]. For prediction purposes, having high variance is a crucial problem, thus, several improvements were proposed. These improvements are bootstrap aggregation, boosting and most importantly Random Forests, which is our focus in this paper. Fundamentally, the Random Forests is an ensemble of Decision Trees which are grown from randomly sampled data with randomly selected explanatory variables. Although, the Random Forest is also capable of regression, from our perspective, it gives an account of itself in a classification setting Therefore, we narrow our focus to the classification settings, meaning that the dependent variable in data is categorical.

We start by explaining the building block and the starting point of the Random Forest which is the Decision Tree. We focused on the tree-building process with different splitting criteria rather than pruning, since the Random Forest uses fully grown trees and does not prune them. After delving into the variance issue, we describe one of the solutions, bootstrap aggregating (bagging) due to it being one of the main principles of the Random forest. 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 in Random Forest. We start the Mathematical Explanation part by defining those voting processes. Although, we primarily 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 the Decision Tree. By exploiting bias-variance decomposition of this 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 the Random Forest enables us to measure how important each independent variable is. 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 the real data study, we used Titanic data [23] and compared Random Forest's performance with two boosting methods: Adaptive and Gradient Boosting.

In the literature, the Decision Tree is covered by several published works including [2] and [16]. As a solution to the problem of high variance in the Decision Tree, Leo Breiman introduced bagging to add randomness to the model with randomly sampling the data[3]. Subsequently, the Random Forest is introduced as an extension including randomness in the variable selection process [6]. In our paper, while [12] being the main source regarding intuition, Louppe provides us with a reassessment of published works and an insight into the mathematical aspects of the Random Forest[19]. The bias-variance decomposition and the examination of the improvement of the Random Forest upon the Decision Tree makes use of [15], [9], [11], and [18]. Louppe also clarifies the variable importance in [20] and [17] illuminates the variable selection process.

This paper is written as a term paper for the Research Module in Econometrics and Statistics and we would like to be graded as a group.

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 form of a tree structure. Each node in the tree further splits 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 nodes represent the actual classification or decision. The topmost decision node in a tree that 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.

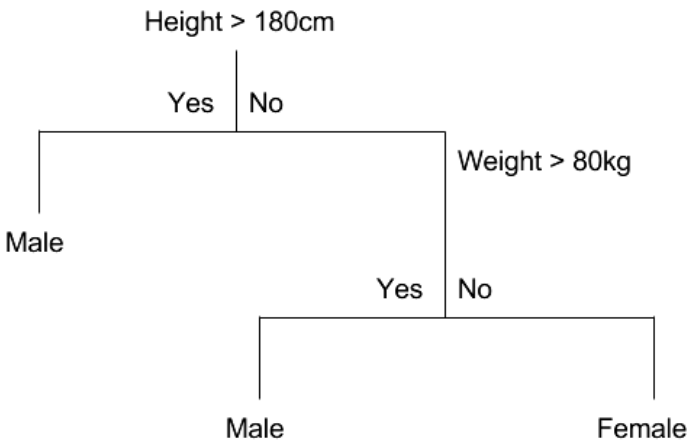

Figure 1: Given a data set with two features height and weight, and gender as the target variable, this example tree stratifies the two-dimensional feature space into three distinct subsets each represented by the terminal nodes at the bottom. The stratification occurs at the two deciding nodes depending either on whether its height is above 180 cm and 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 growth is to choose a split among all the possible splits at each node such 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 in the following algorithm:

Algorithm 1: Binary Splitting [2]

(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 categorical. The original CART algorithm uses Gini and Towing as purity measures for the splitting criterion [2]. However, implementations of the algorithm such as Python's sklearn package [21] 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 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 instances in t. The proportion of the instances belonging to class c in the sample D falling into t equals $N_c(t)/N_j$. The given set of priors $\pi(c)$ specify 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 c 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

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

Further, 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 defined as $p_L = p(t_L)/p(t)$ and $p_R = p(t_R)/p(t)$.

As already stated above, 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 for a higher value. Conversely, if a node has a 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 measures 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 performance of a Decision Tree by assessing its fit to the data. The generalization error is a widely used concept for evaluating the fit [6]. In this paper, 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. These methods include Bagging, Boosting and Random Forest which 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 Tree belongs to methods that are sensitive to the specific data on which they are trained [12]. In other words, Decision Tree estimates are susceptible to even slight changes in data and that stems from having high variance. Bootstrap Aggregation (Bagging) was designed for dealing with the high variance problem in Decision Tree estimates. L. Breiman introduced bagging for Decision Tree in [3]. We start the bagging procedure by conducting bootstrapping. We draw multiple random subsamples with replacement, and then we use these datasets as training data for our model. Therefore, the number of Decision Trees equals that of the subsamples. Finally, we obtain the final estimate of the bagged model by averaging over each estimate of the bootstrapped trees. As Bagging, Random Forest makes use of bootstrapped sampling and employes bagged Decision Trees. The fundamental difference between bagging and Random Forest is injection of randomness included explanatory variables. We will go into further details about Random Forest and the additional randomness in the next section.

3 Random Forest

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 the work presented by [5], [6], and [4], which propose that ensembles of different decision trees can achieve a meaningful improvement in accuracy in classification problems and other common learning tasks such as regression.

3.1 Main Idea and Illustration

An ensemble of randomly trained Decision Trees, i.e. Random Forest, was defined by [6] as follows:

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

Algorithm 2: 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,θ_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 θ_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 the Random Forest model is further inclusion of randomness which allows for more diverse bootstrapped trees. The two key concepts that make the Random Forest "random" are:

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

In practice, random sampling of training observations is done by bootstrapping. Bootstrapping for random forests means that every single decision tree is trained on a random sample which is drawn with replacement. Further randomness gets added to the model by taking only a subset of all the variables into consideration when splitting each node in the respective Decision Tree. There are different practices for choosing the number of variables for splitting the node. In [12], the following specifications are recommended:

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 variables in the classification or regression problem. Alternatively, this parameter can also be optimized by cross-validation.

These aforementioned specifications are only recommendations from [12] and [6]. In practice, the most suitable values for these parameters can differ and they also will depend on the problem. Therefore, these parameters should be treated as tuning parameters. The number of chosen variables has a decisive impact on the model, since it controls not only the amount of randomness within each tree, but also the amount of correlation between different trees in the forest. As the randomness parameter decreases, the 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.

The last step consists of taking all predictions from every decision tree and combining them into in order to get a single prediction of Random Forest. We can adopt various approaches when combining Decision Tree predictions. We will delve into details regarding those approaches, but in contemporary exercises, Random Forests generate probabilistic output in classification problems [8]. They return not just a single class point prediction, but a whole class distribution. The combination of tree predictions can be described as below:

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

whereas B denotes the number of Decision Trees in the Random Forest, and each tree obtains the posterior distribution $p_b(Y = c|X = \mathbf{x})$.

3.1.3 Out of Bag Sample

In applications of Random Forest, we can use the Out of Bag sample to validate our model. When performing a bootstrap to obtain the random sample from the data training, only 2/3 of data is included in expectation and the left-out part of data is called Out of Bag (OOB) sample. After training the Random Forest model, we use the respective OOB sample used as a validation set. Each Decision Tree in the Random Forest is validated with the corresponding leftover sample and averaged to calculate the Out of Bag estimate [12]. The Out of Bag sample is commonly used to assess the error of predictions. Breiman's work [5] shows that the 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. Further, let $T_{D,\theta}$ be the 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 extend the analysis and provide results for uncountable sets as well [18]. As mentioned earlier, the Random Forest classifier selects a bootstrapped subset of observations and grows the decision tree with only a subset of regressors. Repeating this tree growing process B times yields a Random Forest. Assume that x^* is the value that we want to predict its class membership. There are two rules that can be used to get the prediction: majority voting and soft voting [19][25].

In majority voting, after getting every trees prediction denoted as $\hat{T}(x^*)$, the final prediction is the class that gets the most votes from the 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 $\hat{p}_{D,\theta_b}(Y = c|X = x^*)$ get averaged. The most probable class is predicted as follows:

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

As mentioned in [3], the two aforementioned voting procedures provide similar results, yet using soft voting can provide smoother class probability estimates and can be exploited in a deeper analysis setting such as certainty estimates investigation [19]. 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 the algorithm, we can show the improvement of the Random Forest upon the Decision Tree by exploiting the generalization error. We can measure a model's fit with the generalization error, which is also called test error or the expected prediction error. We will start by examining a Decision Tree's generalization error and expand our findings to the Random Forest. All derivations and findings in this section closely follow Louppe's paper [19]. The generalization error of a Decision Tree $T_{D,\theta}$ which is grown using the set D and parameters θ 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 the classification setting, the zero-one loss function is of interest. However, to get a better understanding, 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 the zero-one loss function remains to be relatively unexplanatory, the squared loss provides us a superior insight. Therefore we denoted the findings by using the zero-one loss function with the 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 yielding

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

$L(Y, T_{D,\theta}(X))$ equals 1 if the class of dependent variable is different than its predicted class by Decision Tree and equals to 0 if both are the same, implying that $Y = T_{D,\theta}(X)$. Both loss functions ultimately follow the same logic. While the squared loss function yields a number to measure the error, the zero-one loss function only yields either 1 or 0. The generalization error for both functions are defined below:

$$\boldsymbol{Err}(T_{D,\theta}) = \mathbb{E}_{X,Y}\{(Y - T_{D,\theta}(x))^2\} \tag{15}$$
$$\boldsymbol{Err'}(T_{D,\theta}) = \mathbb{E}_{X,Y}\{1(Y \neq T_{D,\theta}(X))\} = P(Y \neq T_{D,\theta}(X)) \tag{16}$$

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 the random forest upon decision trees.

The Decomposition of Err$(T_{D,\theta})$
Given the probability distribution of $P(X, Y)$, there exists a model ϕ_β which minimizes the generalization error and can be derived analytically [19]. By conditioning on X generalization error for ϕ_β 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 the inner term with respect to Y yields;

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

ϕ_β is denotes the Bayes Model and $\boldsymbol{Err}(\phi_\beta)$ is the residual error, the minimum obtainable error with any model.The irreducible error, $\boldsymbol{Err}(\phi_\beta)$, arises solely due to random deviations in the data [19]. We will exploit the irreducible concept when examining the dynamics of random forests and decision trees. In that sense, with a couple of manipulations, the Bayes Model for squared loss can be reduced to

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

For the 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 its expected value. The Bayes Model of zero-one loss function denoted as ϕ'_β 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 the highest probability in the set Y is chosen by the Bayes Model when using the zero-one loss function. We included intermediate steps in section 6.1.

The aforementioned residual error can be computed for both functions

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

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

By 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, the first term in the equation corresponds to 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 by 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 the Random Forest classifier use a bootstrapped dataset D, thus, if we further examine the second term with taking an 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 shown in section 6.2. 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 that 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 a similar fashion and as mentioned in [19], [9], [15] and [11]. Both zero-one and squared loss functions decompositions yield the same results. However, without assuming that D is normally distributed, bias-variance decomposition cannot be solved for zero-one loss function explicitly as done for squared loss [19]. [18] introduces another decomposition for zero-one loss function (included in section 6.3), but still, it remains to be unexplanatory compared to squared loss, thus, we explain the dynamics by using squared loss, although the main focus of the paper remains to be on classification setting.

Extending findings to Random Forest

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

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

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

where $\mu_{D,\theta}(x)$ is the average prediction of all ensembled trees. Since θ's are random, independent and have the same distribution, expected values for the bootstrapped trees are equal [19]. When we consider the bias of a random forest, we can state that

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

Intuitively, when we get the average prediction of all ensembled trees, we are able to consider $\mu_{D,\theta}(x)$ as one Decision Tree in the simplest terms and make this inference. We can interpret equation (28) as the squared bias cannot be decreased by ensembling randomized models, namely, an ensemble of trees does not guarantee to have a lower bias compared to only one tree [12]. Furthermore, the squared bias of an ensemble of randomized model equals the squared bias of that model in theory [19]. Although the Random Forest is inadequate to propose any structure to decrease the generalization error so far regarding *noise* and $bias^2$, it shows promising performance in reducing the last remaining part of the generalization error. Thus, we can continue our exploration

with the variance of Random Forest, yet, we need to define the correlation coefficient $\rho(x)$ before delving into variance since the correlation coefficient has a 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 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 Pearson's correlation coefficient and the property of θ' and θ'' following the same distribution in the intermediate steps in section 6.4. $\sigma_{D,\theta}(x)$ is the variance of a single Decision Tree and associated with prediction variability stemming from the randomness of the set D and randomness introduced with θ [19]. 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 they are 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 6.5. 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 $\rho(x)\sigma^2_{D,\theta}(x)$ and due to randomization in the algorithm $\rho(x) < 1$. Thus, the variance of the Random Forest is less than the variance of the Decision Tree. This inference implies that the generalization error of the Random Forest is less than the generalization error of the Decision Tree. If the decision trees in the random Forest are independent, therefore $\rho(x) \to 0$, the variance is reduced to $\sigma^2_{D,\theta}(x)/B$ which can be decreased by increasing B as mentioned. Conclusively, the Random Forest improves the performance of the Decision Tree by 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, to get a better insight of the processes of interest, we need to understand which explanatory variables are more important and useful to make mentioned predictions. The Random Forest allows to not only build an accurate model with reliable predictions, but also to provide variable importance measures. In this section, we use [20] and [1], and after presenting the Mean Decrease Impurity, we will examine variable importance for randomized tree ensembles. Then, we will discuss these ideas' link to the Random Forest algorithm.

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 proposed by [6]; Mean Decrease Impurity (MDI) and Mean Decrease Accuracy (MDA). For MDI, we get the total decrease in node impurity resulting from splitting into further nodes and we average that total over all trees. While closely following Louppe's notation [20], we can define MDI as

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

where $v(s_t)$ is the variable used in split s_t and $p(t)$ is the proportion N_t/N of samples reaching t. By definition, $MDI(X_j)$ evaluates the importance of a variable X_j for predicting Y by summing over the weighted impurity decreases $p(t)\Delta i(s_t, t)$ for all nodes t where X_j is used and averaging that sum over all B trees in the Random Forest. We can use the above definition for various impurity measures $i(t)$ including the Gini Index and the Information Entropy. Furthermore, in the contemprorary exercises, MDI is commonly used and also embedded into the Random Forest algorithm in skilearn package [21]. 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 the model's prediction accuracy. MDA uses Out-of-Bag error estimate to calculate a variable's importance. 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 observations in the tree. $MDA(X_j)$ is calculated by averaging the differences in the Out-of-Bag error estimation after and before the permutation in all trees. Because of these permutations, the measure is also referred to as the permutation importance.

3.3.2 Variable importance for totally randomized tree ensembles

Let us now consider Mean Decrease Impurity as defined by equation (31) and assume that U is a set of categorical input variables and is equal to $\{X_1, ..., X_p\}$ In addition, assume that categorical output Y. For simplicity, the impurity measure which we use is the Shannon Entropy. Let us assume that we created an infinitely large ensemble of trees that are completely randomized and fully developed. Further, assume that for the creation of these trees, we used training sample D consisting of N joint observations $X_1, ..., X_p, Y$ independently taken from the joint distribution $P(X_1, ..., X_p, Y)$. A totally randomized and fully developed tree as defined in [20] is a Decision Tree which partitioned every node t using variable X_j chosen uniformly and randomly among those which were not yet used at the parent nodes of t and where every t is split into $|\mathcal{X}_j|$ sub-trees, with $\mathcal{X}$ denoting the image set. Given the above assumptions, the two following theorems have been proven.

Theorem 1. *The MDI of $X_j \in U$ for Y as computed with an infinite ensemble of fully developed totally randomized trees and an infinitely large training sample is:*

$$MDI(X_j) = \sum_{k=0}^{p-1}\frac{1}{C_p^k}\frac{1}{p-k}\sum_{O\in\mathcal{P}_k(U^{-j})} I(X_j; Y|O) \tag{32}$$

where U^{-j} denotes the subset $U - \{X_j\}$, $\mathcal{P}_k(U^{-j})$ is the set of subsets of U^{-j} of cardinality k, and $I(X_j; Y|O)$ is the conditional mutual information of X_j and Y given the variables in O as defined in [17].

Theorem 2. *For any ensembles of fully developed trees in asymptotic learning sample size conditions (i.e. in the same conditions as those of Theorem 1), we have:*

$$\sum_{j=1}^{p} MDI(X_j) = I(X_1, ..., X_p; Y) \tag{33}$$

3.3.3 Relevant and irrelevant variable

As it was done in [17], we define relevant and irrelevant variables. The variable X_j for which exists at least one subset $O \subseteq U$ (possibly empty) s.t. $I(X_j; Y|O) > 0$ is a variable which is relevant to Y with respect to U. In contrast, a variable which is irrelevant to Y with respect to U is a variable X_i for which, for all $O \subseteq U$, $I(X_i; Y|O) = 0$ holds. Remark that if a variable is relevant then by definition it can not be irrelevant at the same time, so $j \neq i$. Remark also that if X_i is irrelevant to Y with respect to U, then also by definition it is always irrelevant to Y with respect to any subset of U. Louppe et al. using these definitions showed that below holds.

Theorem 3. *$X_i \in U$ is irrelevant to Y with respect to U if and only if its infinite sample size importance as computed with an infinite ensemble of fully developed totally randomized trees built on U for Y is 0.*

Theorem 4. *The infinite sample size importance of any variable $X_j \in U_R$ (where $U_R \subseteq U$ is the subset of all variables in U which are relevant to Y with respect to U) as computed with an infinite ensemble of fully developed totally randomized trees built on U_R for Y is the same as its importance computed in the same conditions by using all variables in U. It means:*

$$MDI(X_j) = \sum_{k=0}^{p-1} \frac{1}{C_p^k} \frac{1}{p-k} \sum_{O \in \mathcal{P}_k(U^{-j})} I(X_j; Y|O) = \sum_{l=0}^{r-1} \frac{1}{C_r^l} \frac{1}{r-l} \sum_{O \in \mathcal{P}_l(U_R^{-j})} I(X_j; Y|O) \tag{34}$$

where r is the number of relevant variables in U_R.

The two theorems above inform us that the variable importance obtained with an ensemble of completelyy randomized trees depends only on the relevant variables. It shows that the importance of irrelevant variables is equal to 0. Accordingly, it does not affect the importance of relevant variables. In practice, the mentioned properties are desirable conditions for a reliable criterion for assessing the variable importance. As should be expected, any noise should not have importance higher than zero and it should not influence any other variable importance.

3.3.4 Variable importance of non-totally randomized trees

In the previous two subsections, analysis of variable importance does not directly relate to the Random Forest [6], because we mention the analysis of completely randomized

ensembles of trees. As proved in [17], it is not true for the Random Forest that variable is irrelevant if and only if its importance is equal to 0. It is true that the importance of variables is dependent on the number of irrelevant variables, such that variable importance as obtained from totally randomized trees has properties that the Random Forest does not possess. Asymptotically, it is more appropriate to use totally randomized trees to assess the importance of the variable. In finite examples, where we have a limited number of samples and a limited number of trees, guiding the choice of the splitting variables is still a reliable strategy. In these cases, we can not measure $I(X_j; Y|O)$ for all X_j and O. Hence, even if the resulting variable importance can be biased, it is pragmatic to promote those that seem to be the most promising.

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 [23] 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:

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

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 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.

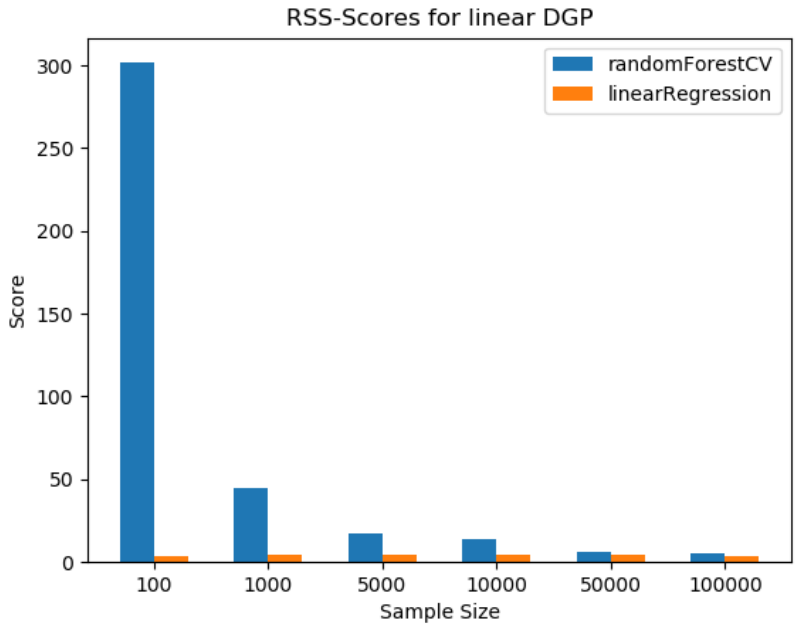

Figure 2: This plot illustrates the RSS for different training sample sizes for Random Forest and OLS. These samples were drawn from a linear DGP in accordance with equation (35). The holdout set for calculating the RSS was drawn again for each training sample from the same DGP. It always contained 100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

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

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

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

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

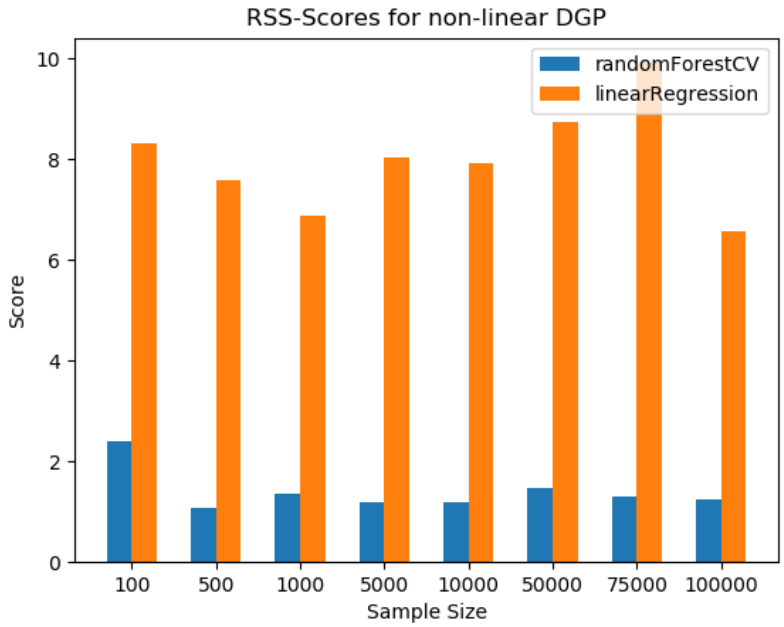

Figure 3: This plot illustrates the RSS for different training sample sizes for Random Forest and OLS. These samples where drawn from a non-linear DGP in accordance to equation (36). The holdout set for calculating the RSS was drawn again for each training sample from the same DGP. It always contained 100 observations. In the 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 Decision Tree, the RSS of the Random Forest converges relatively quickly while that of the OLS remains unstable and high.

4.1.1 Bias Variance Decomposition

For both simulation setups, we decomposed the expected generalization error of the Random Forest estimator. We included the figures for both cases in the appendix (6). Figure 7 illustrates the decomposition of the linear setup and figure 8 that of the non-linear setup. In both figures, the expected generalization error is denoted as a loss. In the linear case, as sample size increases, both the expected generalization error and bias tend to decrease, yet while being regularly low, the variance of Random Forest estimator does not show any pattern. Theoretically, we expect low variance and the figure is in line with our expectations. On the other hand, the case with non-linear DGP emphasizes more the power of Decision Trees embedded in Random Forest. The expected generalization error is driven primarily by variance and bias remains low for all sample sizes. Since the Decision Tree yields low bias estimates, and the variance decreases with sample size, the figure is consistent with our expectations. In addition, is a better showcase of Random Forest compared to linear DGP.

4.2 Application of Random Forest on real data

As previously mentioned, we applied the Random Forest on the Titanic data set [23] 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 [22]. 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.

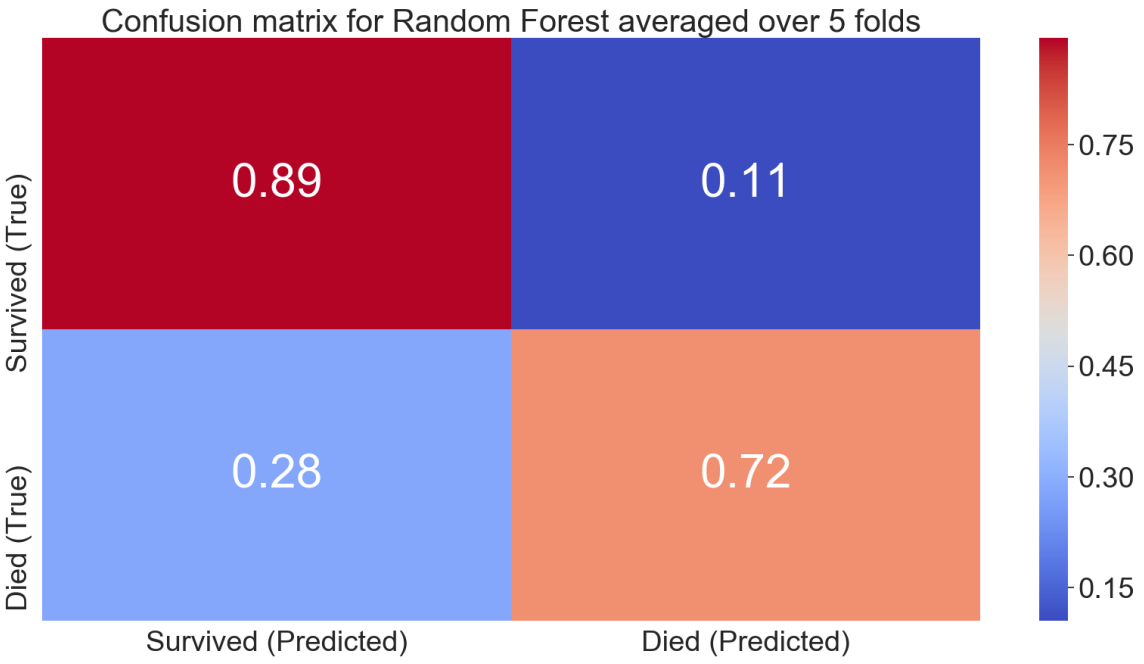

Figure 4: This plot illustrates the accuracy of the Random Forest's prediction on the Titanic data set. The data set got split into five folds on which the Random Forest got applied. Then, the resulting combination of classification errors got averaged. The left axis indicates the true class membership,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

5 Comparison to two boosting methods

Before introducing AdaBoost and Gradient Boosting methods, 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. In the 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 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 are 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 by 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 that predicts 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. When we employ Adaboost to predict the survival outcome of the passenger on Titanic, we achieve a mean classification rate of 82.94% accuracy. By utilizing the confusion matrix of the classification and compare it with that of the Random Forest, we can see that for the non-survivals the result is almost the same, but the Random Forest is slightly better when it comes to identifying survivors.

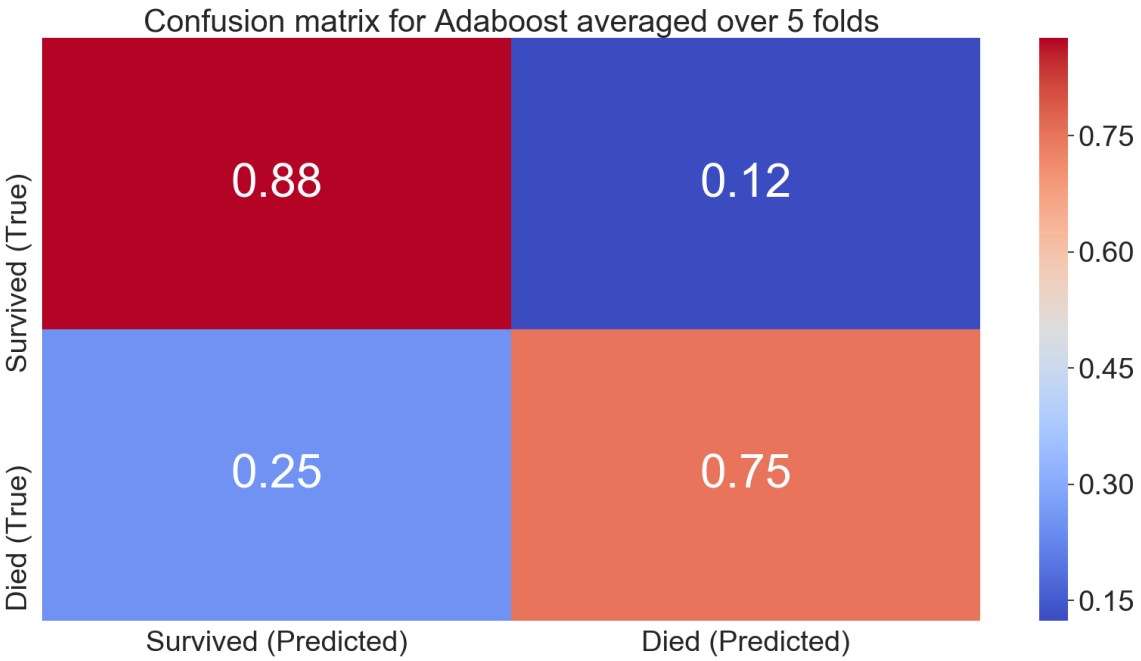

Figure 5: This plot illustrates the accuracy of the AdaBoost's prediction on the Titanic data set. The data set got split into five folds on which the Random Forest got applied. Then, the resulting combination of classification errors got averaged. The left axis indicates the true class membership, while 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 a 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 the method creates a new weak learner and tests 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 that 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 a similar way, like 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.

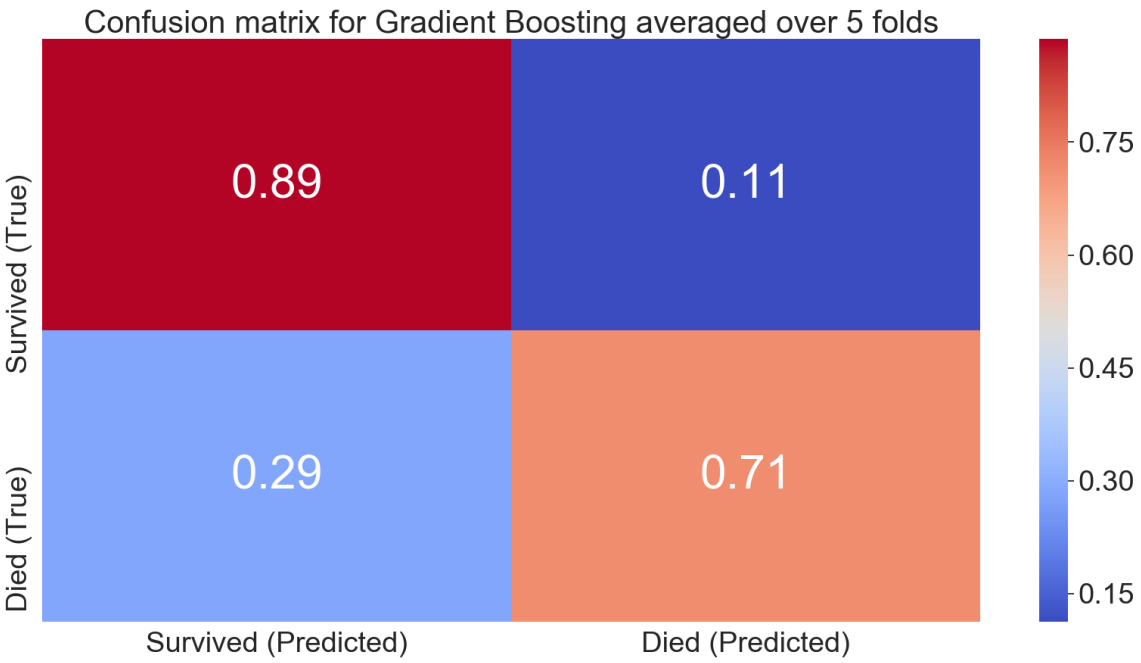

Figure 6: This plot illustrates the accuracy of the Gradient Boosting's prediction on the Titanic data set. The data set got split into five folds on which the Random Forest got applied. Then, the resulting combination of classification errors got averaged. The left axis indicates the true class membership,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

6 Appendix

6.1 Bayes Model explanation

We identified Bayes Model in equation (20) as such

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

6.2 Bias-variance decomposition of Squared Loss Function

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

$$\begin{aligned}\boldsymbol{Err}(T_{D,\theta}(x)) &= \mathbb{E}_{Y|X=x}\{(Y-T_{D,\theta}(x))^2\}\\ &= \mathbb{E}_{Y|X=x}\{(Y-\phi_\beta(x)+\phi_\beta(x)-T_{D,\theta}(x))^2\}\\ &= \mathbb{E}_{Y|X=x}\{(Y-\phi_\beta(x))^2\}+\mathbb{E}_{Y|X=x}\{(\phi_\beta(x)-T_{D,\theta}(x))^2\}\\ &\quad+\underbrace{\mathbb{E}_{Y|X=x}\{2(Y-\phi_\beta(x))(\phi_\beta(x)-T_{D,\theta}(x))\}}_{=0\ \text{ since } \mathbb{E}_{Y|X=x}(Y-\phi_\beta(x))=0 \text{ from 19}}\\ &= \underbrace{\mathbb{E}_{Y|X=x}\{(Y-\phi_\beta(x))^2\}}_{\text{from 21 equals to } \boldsymbol{Err}(\phi_\beta(x))}+\mathbb{E}_{Y|X=x}\{(\phi_\beta(x)-T_{D,\theta}(x))^2\}\\ &= \boldsymbol{Err}(\phi_\beta(x))+(\phi_\beta(x)-T_{D,\theta}(x))^2\end{aligned}$$

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

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

6.3 Kohavi's decomposition of zero-one loss function

Kohavi proposed an alternative decomposition for zero-function [18]. 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

$$\begin{aligned}
\boldsymbol{Err}(T_{D,\theta}(x)) &= L(\phi_\beta(x), T_{D,\theta}(x) P(\phi_\beta(x), T_{D,\theta}(x)) \\
&= \sum_{\phi_\beta(x), T_{D,\theta}(x)} [1 - \delta(\phi_\beta(x), T_{D,\theta}(x))] P(\phi_\beta(x), T_{D,\theta}(x)) \\
&= 1 - \sum_{y \in Y} P(\phi_\beta(x) = T_{D,\theta}(x)) = y)
\end{aligned} \tag{37}$$

Even if in the Kohavi's paper it is stated that the last step is a simplified version of the extended Bayesian Formalism [24], there is not enough explanation to explicitly show the mathematical derivation nor the intuition. If we continue examine the equation (37) we get the following decomposition;

$$\begin{aligned}
\boldsymbol{Err}(T_{D,\theta}(x)) &= 1 - \sum_{y \in Y} P(\phi_\beta(x) = T_{D,\theta}(x)) = y) \\
&= \sum_{y \in Y} -P(\phi_\beta(x) = T_{D,\theta}(x)) = y) + \sum_{y \in Y} P(\phi_\beta(x) = y) P(T_{D,\theta}(x)) = y) \\
&+ \sum_{y \in Y} [-P(\phi_\beta(x) = y) P(T_{D,\theta}(x)) = y) + \frac{1}{2} P(T_{D,\theta}(x) = y)^2 + \frac{1}{2} P(\phi_\beta = y)^2] \\
&+ [\frac{1}{2} - \frac{1}{2} \sum_{y \in Y} P(T_{D,\theta}(x) = y)^2] + [\frac{1}{2} - \frac{1}{2} \sum_{y \in Y} P(\phi_\beta(x) = y)^2]
\end{aligned} \tag{38}$$

Rearranging equation (38) yields

$$\begin{aligned}
\boldsymbol{Err}(T_{D,\theta}(x)) &= \sum_{y \in Y} [P(T_{D,\theta}(x) = y) P(\phi_\beta(x) = y) - P(T_{D,\theta}(x) = \phi_\beta(x) = y)] \quad (*) \\
&+ \frac{1}{2} \sum_{y \in Y} [P(T_{D,\theta}(x) = y) - P(\phi_\beta(x) = y)]^2 \\
&+ \frac{1}{2} (1 - \sum_{y \in Y} P(T_{D,\theta}(x) = y)^2) + \frac{1}{2} (1 - \sum_{y \in Y} P(\phi_\beta(x) = y)^2)
\end{aligned} \tag{39}$$

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 [19], we wanted to explain mathematical dynamics of random forest with using the squared loss function alongside zero-one loss function.

6.4 Correlation Coefficient

Using the definition of the Pearson's correlation coefficient and the property of θ' and θ'' 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 [19] [14]. Being that said the findings we explored with decomposing the variance of the squared loss function are robust.

6.5 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 θ_i and θ_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).

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

6.6 Bias-Variance Decomposition Figures

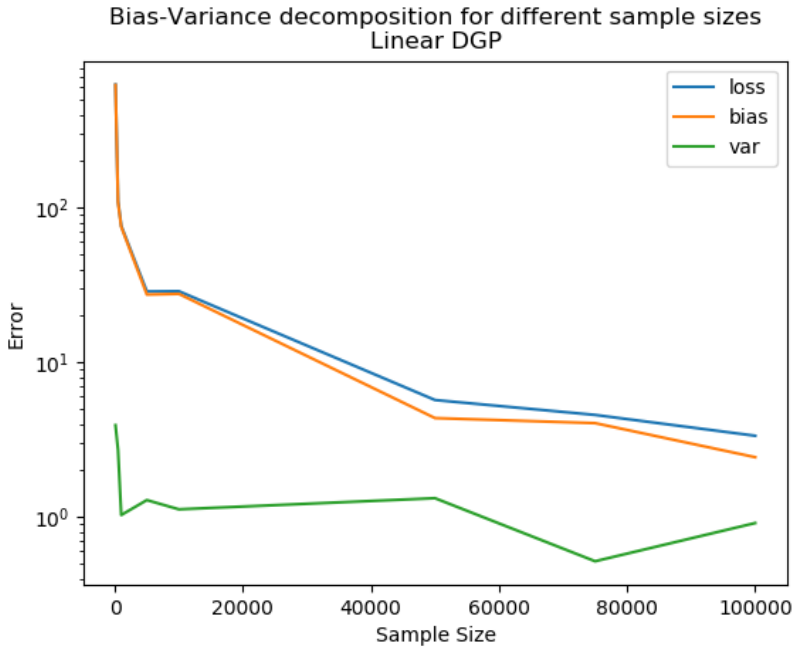

Figure 7: This figure demonstrates the decomposition of the expected generalization error in linear DGP. The expected generalization error is denoted with loss. As sample size increases both the expected generalization error and bias tend to decrease. Although the variance does not follow a pattern, in theory, we expect low variance and variance remains to be low for all sample sizes.

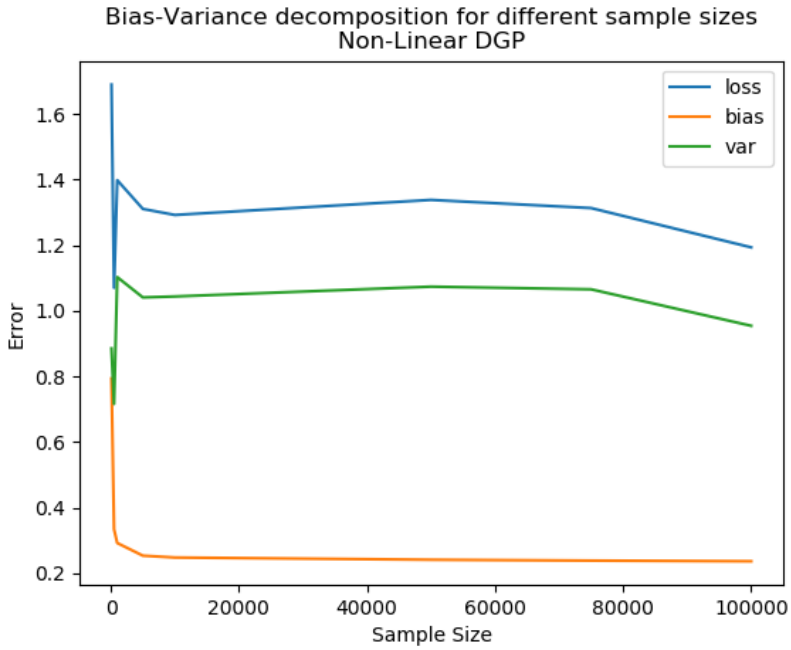

Figure 8: This figure demonstrates the decomposition of the expected generalization error in nonlinear DGP. The expected generalization error is denoted with loss. The expected generalization error is driven primarily by variance and bias remains to be low for all sample sizes. As expected, Random Forest decreases the variance and bias remains to be low due to low biasness of Decision Tree estimates.

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 and George H. John. "Wrappers for Feature Subset Selection". In: *Artif. Intell.* 97.1–2 (Dec. 1997), pp. 273–324. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(97)00043-X. URL: https://doi.org/10.1016/S0004-3702(97)00043-X.

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

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

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

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

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

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

[24] 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.

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