Ca' Foscari
University
of Venice

Master's Degree Programme

in Global Development and Entrepreneurship
Curricula Entrepreneurship
LM-56 (Economics)

Final Thesis

Random Forest Implementation for Classification Analysis: Default Predictions Applied to Italian Companies

Supervisors
Prof. Stefano Federico Tonellato
Prof. Roberto Casarin

Graduand
Davide Tramontin
Matriculation number
860126

Academic Year
2019 / 2020

This page is intentionally left blank.

I wish to thank Professor Stefano Federico Tonellato and Professor Roberto Casarin, whose contribution has been essential during this research experience. Your guidance, expertise, and passion have been fundamental in the development of this work. Finally, thank you to VERA academy and Ca' Foscari University of Venice for providing me this great opportunity.

A special thank you to you, mom, papá, Luca, and Marco, for your support and for always believing in me. I wouldn't be where I am today, and I wouldn't be who I am today, without you.
A big hug to my whole family, in Canada and in Italy, and to all my friends. Thank you for making my life so awesome.

This is for you, my little niece. I am waiting for you.

This page is intentionally left blank.

Table of Contents

This page is intentionally left blank.

Abstract

The growing importance of big data and the increased environment complexity have led to an increase in the implementation of machine learning algorithms, given their ability to efficiently deal with entangled situations. This study contributes to the framework regarding the application of random forests and other machine learning algorithms. Specifically, the topic of research is company failure and probability of default. The major impact that the firm's default has on businesses, markets, and societies, underlines the importance of developing models which predict the probability of default. This research attempts to address this topic with two purposes: create an accurate binary model to classify companies in *Defaulted* and *Non-Defaulted*; identify the most important predictors in order to understand the links between the financial ratios considered and the companies' status.

Random forests' ability to deal with big data sets and with various and diverse predictors have led to choosing this algorithm to analyze the topic of research. Building on a literature review of decision trees, random forests, company failure, and the models which predict the probability of default, this study's analysis is constructed through several experiments which permit to tune the model appropriately and construct the final model which provide the highest accuracy. Through its cross-sectional analysis, this research confirms random forests' strong stability and its consistent performance. The final model generated performs well, and identifies in the coverage of fixed assets, gross profit, net working capital, cost of debt, debt to equity ratio, leverage, solvency ratio, and return on assets, the most important default predictors.

Finally, the results and methods applied have been jointly used to extend the purpose of this research. In order to permit further development of this study and of research on random forest and machine learning, an R programming code which permits to reproduce the computations carried out is provided. Importantly, the designed function is applicable to any data set to permit the analysis of different topics as well and provides a visual representation of the results through a Shiny App, permitting an easier interpretation of results.

Keywords: Random Forests, Machine Learning, Bankruptcy Predictions, R programming

This page is intentionally left blank

Introduction

In a globalised and complex environment, the role that big datasets play, and the techniques trough which such data are analysed, have been growing in importance. Specifically, in the case of financial and private equity investments, the complexity of the decision-making process has increased, making it difficult for analyst and investors to obtain a clear picture of an investment. Companies' successes and failures depend on many internal and external factors which increase the difficulty faced by analysists looking for the identification of all the elements that may have an impact on a company or institution. Moreover, one of the main difficulties has been to create an adaptable model, since diverse situations are affected differently by the various factors. To some extent, machine learning algorithms have helped in improving and dealing with such processes by reducing the possibility of errors, making them less time and resource consuming, and by creating models which do not overfit the data (Breiman, 2001), and, therefore, are not data-specific methods.

This study aims to use random forests to predict the probability of a company's default. Useful methodological references can be found in the context of univariate discriminant analysis (Beaver, 1967), multiple discriminant analysis (Altman, 1968), the Black-Scholes-Merton model (Black and Scholes, 1973; Merton, 1974) and the KVM model (Kealhofer, 2003).

Predicting default with certainty is not possible. Nevertheless, it is possible to obtain a quantification of default risk. From a business point of view, then, such methodologies permit to guide management in the decision-making process. From an investor's point of view (e.g. banks, financial institutions, investment funds), it can be assumed that investment decisions are done by assigning scores to the companies according to specific parameters. Such scores, of course, incorporate the probability of default. Predicting a company's default and quantifying the probability of failure is very relevant for financial investors. For instance, it can be noted how such predictions may influence the market for loans and the real estate market.

In the case of start-ups and young ventures, applying machine learning techniques may permit findings that through classical methods are not possible. Uncertainty and unpredictability

involving new ventures complicates the decision-making process regarding both start-up management (from the firm's point of view) and deciding which start-ups should be considered for investments (from investors' point of view, e.g. business angels and venture capitalists). Although the accuracy and reliability of machine learning applications to start-ups will likely be lower, models could be trained to deal specifically with start-ups, permitting to obtain information that would be useful to all parties dealing with new ventures.

Finally, from a macroeconomic point of view, company's bankruptcy has major direct and indirect impacts on societies; among others, for example, it increases unemployment. By identifying which variables are of greater importance in the prediction of a company's default, this would permit to underline where the focus of public policies should be placed to avoid such issue. This, in turn, would permit to run simulations incorporating changes in the identified variables, and therefore evaluate outcomes by trial and error, and, finally, clarifying the best policy measure possible.

This research is a part of a great ensemble of studies that could be carried regarding, for example, the interaction of private equity and financial investments and machine learning. Such investment in fact depends on a much greater number of variables than only credit risk, therefore expanding such research to other variables can provide a clearer view of a company's situation. The main purpose of this study, though, is to prepare the basis which can be used for further research; one important feature of machine learning algorithms is, in fact, their adaptation capabilities. In other words, this study begins as the basis for a greater project. By developing prediction models for all aspects that influence such decisions, unifying such models could provide one strong predictor which will be able to maintain a wider perspective than the single models alone.

This research will mainly focus on random forests. Nevertheless, an application of the decision tree algorithm (Breiman et al., 1984) will be designed to permit comparison across models. Moreover, further machine learning algorithms, such as AdaBoost (Freund and Schapire, 1996a; 1996b), could provide useful insights to compare the model's predictive capabilities. The aim of this research is to use such algorithms with a classification purpose and, therefore, to create a model that permits to answer the question "Has the company defaulted?". By doing so, the goal is to create a model which identifies the patterns within data without overfitting. In fact, this will

generate a model which, given as input the predictors' values, provides useful insights regarding a company's legal status. Finally, the models will be evaluated to identify the most accurate one.

This study is organized as follows. Following this brief introduction, the first chapter will provide a literature review underlining the main features and characteristics of random forests. In this chapter, the focus will be placed on both the theoretical as well as the practical programming aspects using R. The second chapter will provide a literature review on credit risk and company default, in order to permit the reader to obtain an understanding of the topic and begin to comprehend how the models available in the literature apply to the topic of company default. This chapter will present the main models that are being used, and the factors that are believed to describe the wellbeing of a company. The third chapter will show the actual implementation of the first two chapter combined; this can be considered as the data analysis part of the research. This chapter is divided in 1) presentation of the data used in this research, 2) development and construction of the model, 3) implementation of the model in R programming to predict company default.

The final chapter will summarize the procedure, findings, limitations of the research and possible further developments.

This page is intentionally left blank

CHAPTER I

MACHINE LEARNING ALGORITHMS

This chapter focuses on machine learning algorithms, and, specifically, on decision trees and random forests, which are the approaches used in this research. The aim of the following sections is to present the main features and characteristics of the two methods, by presenting both theoretical and practical concepts. For continuity and simplicity, across both sections 1.2 and 1.3, an example will be carried using the `iris` data set available in R.

1.1. MACHINE LEARNING ALGORITHMS APPLICATIONS

In the recent decades, technological, economic, and social changes have enormously increased the complexity of our society (Burgartz et al., 2016). Moreover, Burgartz et al. (2016) show that volatility, uncertainty, and ambiguity, significantly characterize all processes, representing the difficulty of dealing with a globalized environment. The complexity of such society is, among other factors, given by the increasing quantity of data available. While the latter provide great opportunities to exploit, managing and analyzing big data sets efficiently is challenging. The adoption of computational techniques has, therefore, become necessary in order to obtain better performances, and to reduce the misuse of resources. Machine learning algorithms, specifically, permit the creation of models which are efficient and, importantly, adaptable to complex data structures, big and small data sets, and to many fields such as engineering, business and finance, medicine, and many other (Papacharalampous and Tyralis, 2017). Given that machine learning algorithms' performance and accuracy vary according to the situation in which they are applied, it is useful to compare the result provided by the different algorithms (Armitage et al., 2006). This study aims to contribute to the existing literature regarding random forest's performance, and specifically, in the following chapters, on credit risk and company default.

1.2. DECISION TREES

1.2.1. Introduction

With respect to the other machine learnings algorithms, decision trees are relatively simple. The purpose of this study is not to analyze and comprehend the mathematics and deep functioning of the algorithm; for this, refer to studies by Breiman et al. (1984) and by Maimon and Rokach (2005), among others. Nevertheless, this section will provide the necessary information to implement decision trees and, most importantly, it will show the basis for understanding and applying random forests.

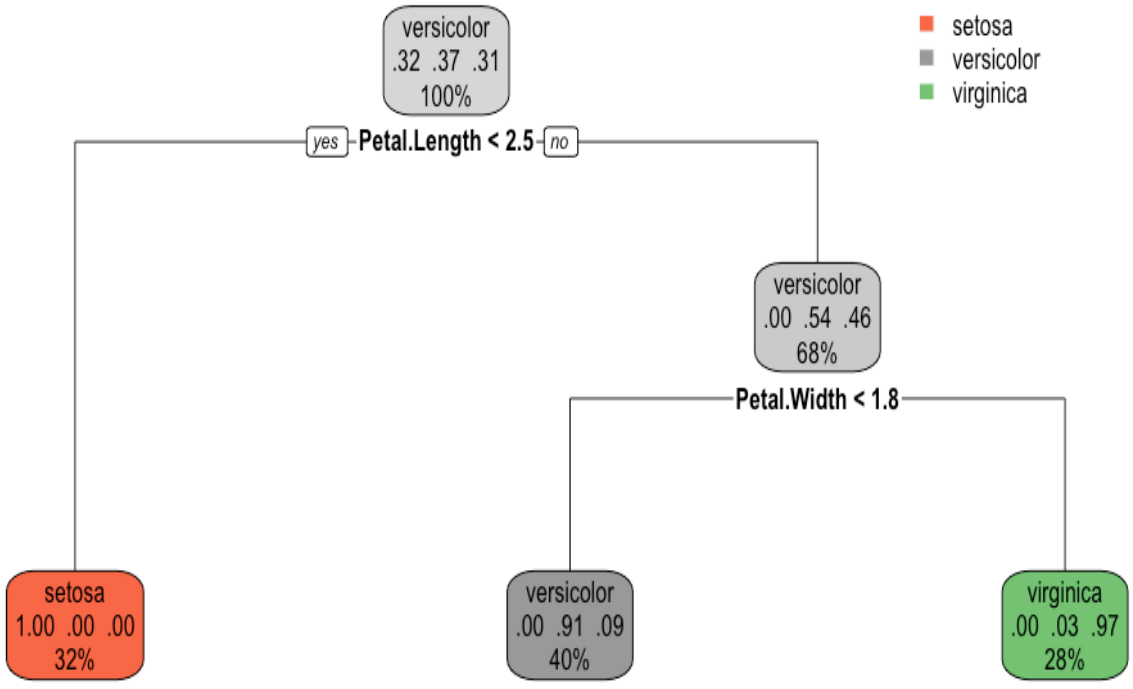
While this study will focus on classification problems, it should be noted that the algorithms considered can be used for regression problems as well. Classification of a data set involves, as the following sections will show, a recursive process whose aim is to split the data in order to extract information and obtain a prediction. Consider, for example, a doctor that has to evaluate the medical condition of a patient (the medical example is often used in the literature to obtain an initial understanding of the problem). Once admitted, several measurements regarding the patient's health conditions are made in order to obtain the necessary data. In order to extract information from the data, and therefore identify the patient's conditions, the doctor evaluates different possibilities according to the recorded values regarding the patient. Therefore, considering the various variables such as, for example, blood pressure, fever, family medical history, and so on, the doctor can identify the appropriate diagnosis. A classification problem, then, involves training an algorithm to interact with specific variables in order to obtain an output class. The reason for using a decision tree is that the algorithm can compute the tasks of the decision maker rather straightforwardly. In the previous example, given the values regarding the patient's health, and given a trained decision tree, the algorithm generated can predict the diagnosis.

Decision trees are classifiers which are expressed as recursive partitions of the instance space. (Breiman et al. 1984). In other words, such algorithms permit the classification of data by recursively splitting them accordingly with specific criteria.

To put it simply, the decision tree process can be imagined as an upside-down tree: starting from its root, the algorithm splits downwards, creating branches and leaves (see Figure 1.1).

The initial node, which comprehends the entire learning sample, is called the root node. Being the first node, it does not have incoming edges. From this point onwards, the data are split into new test nodes recursively, until a class output is obtained. In other words, each new node is either assigned to a class that represents the target value (Maimon and Rokach, 2005), and in such a circumstance it is a terminal node of the tree, also called leaf, or it is further split. Figure 1.1 shows an example of such process; a simple decisional tree has been designed using the `iris` dataset available in the R library datasets. Such data set has been collected by Ronald Fisher in 1936, and contains 150 cases, classified considering 5 variables named `Sepal.Lenght`, `Sepal.Width`, `Petal.Lenght`, `Petal.Width` and `Species`. `Species`, specifically, contains the values which are being predicted: `setosa`, `versicolor`, and `virginica`.

Figure 1.1: Decision tree, `iris` example

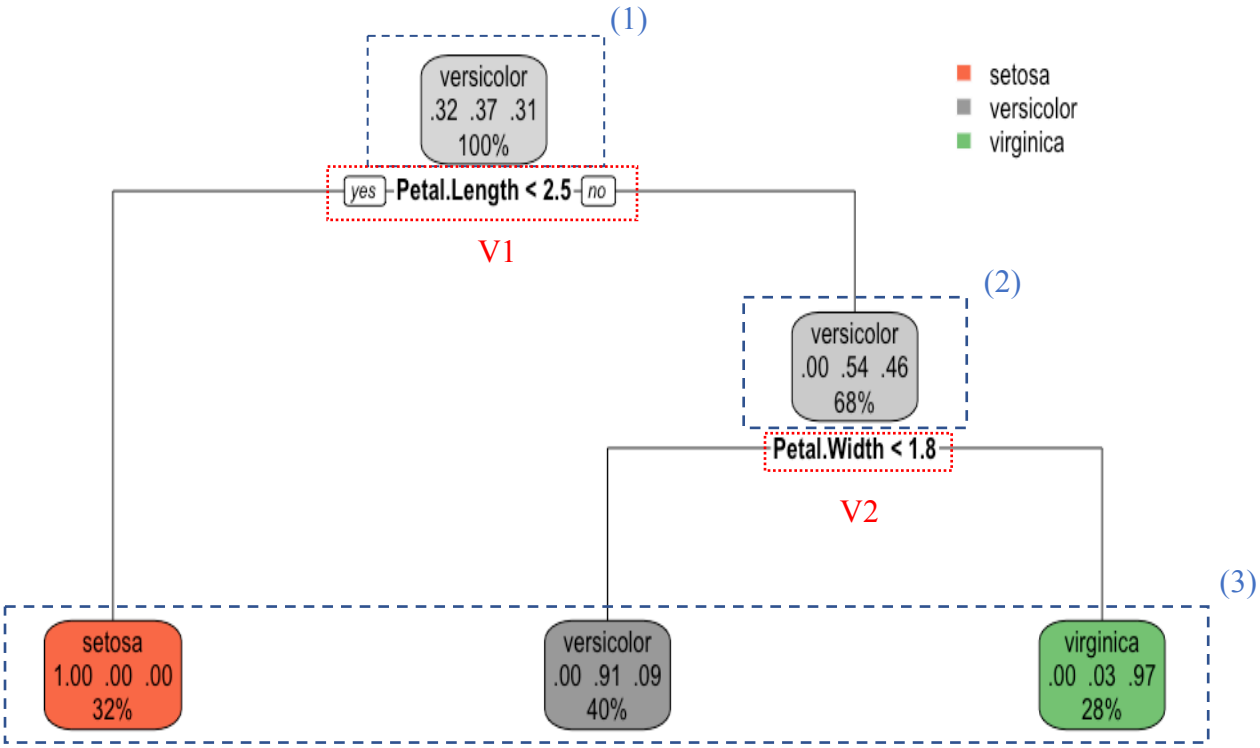


Data Source: `iris` data set available in the R library "datasets"

The R function `rpart.plot()` permits to obtain a useful representation of the decision tree, as shown by the information provided inside the node (full code reported in the Appendix B).

Specifically, from the top down, each node contains the class predicted, the predicted probability of each class, and the percentage of data within the node. As shown in the root node in Figure 1.1., intuitively 100% of the observations are present in such node. While the predicted class and the predicted probabilities are less important in the root node, they are useful in understanding the results of the decision tree's classification. Specifically, consider the leaf nodes at the bottom of Figure 1.1.; the node on the left shows that the predicted class is `setosa` and, importantly, it shows that the split decision "Petal.Lenght < 2.5" permits to obtain a pure node. In fact, the predicted probability of the class `setosa` is equal to 100%, while the ones of `versicolor` and `virginica` are zero. The other two leaf nodes, instead, still account for some level of impurity, given that both `versicolor` and `virginica`'s predicted probabilities are greater than zero in both nodes. Intuitively, the sum of the portion of data present in each leaf node sums to the totality of the data initially available in the root node.

Figure 1.2: Decision tree division and characteristics, `iris` example



Data Source: `iris` data set available in the R library "datasets"

The `iris` dataset contains variables regarding the length and width of the sepal and the petal, and one variable which contains the species of the flower. The decision tree has been constructed in order to predict the `Species` variable, therefore the algorithm has three possibilities of classification: `virginica`, `setosa`, and `versicolor`. Figure 1.1 and 1.2 show the main features of the tree, which permits to clearly identify the three parts previously mentioned: the root node, the decisional nodes, and the class output (predictions). While this case is relatively simple, this division is helpful in deeper (with more decisional nodes) trees.

A decision tree analysis is divided in two parts: training and testing. While the training part involves the creation of the model by providing the algorithm with the information required so that it can find patterns, within the data, between the independent (`Sepal.Lenght`, `Sepal.Width`, `Petal.Lenght`, `Petal.Width`) and the dependent variable (`Species`), testing the model means using new data for which values regarding `Species` are not available. The initial dataset is, therefore, divided in two sub-data sets. There is no general rule regarding the size of the training set, given that it depends on the size of the data set and on the quality of the data (Maimon and Rokach, 2005). For the construction of this case, the guidance provided by Breiman (1996; 2001) and Breiman et al. (1984) has been followed, and 90% of the data set has been used for training; in the following chapters, further developments will be provided in order to choose the appropriate size.

The root node in Figure 1.2 is, therefore, the training set used to construct the model that will predict the class `Species`. Directly connected to the first phase is the beginning of the splitting process, in this case labeled V1. A feature of the decision tree algorithm, and classification machine learning algorithms in general, is that it will select V1 as the decision point that will decrease, as much as possible, the data entropy (Breiman et al., 2001; Maimon and Rokach, 2005), which expresses the amount of disorder and non-homogeneity within the data. This, then, is valid for all nodes throughout the decision tree, as shown in the TreeGrowing (S, A, y) algorithm (Figure 1.3) provided by Maimon and Rokach (p. 169, 2005). The specific terms and steps shown in the algorithm will be addressed throughout the chapter.

While this example is relatively simple and therefore contains only one internal node, decision trees generally have many. As stated above, the choice of V2 is computed by the algorithm with

the aim of obtaining the most information as possible from the data, once V1 has been used to reduce the entropy of the dataset. While internal nodes can lead to leaf nodes, they also lead to new internal nodes which will continue to split the data, by binary recursive partitioning (Moises, 2008). The final part that includes the leaves nodes, meaning the nodes that provide an output class. The tree trained on a data set can be used to predict data not included in the training set. Consider two random units, presented in Table 1.1, which were not included in the training set; therefore, these units represent unseen data for the model. Let's suppose that the species of the two units is not known.

Table 1.1: Random units to test the model, `iris` example

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Unit 35	4.9	3.1	1.5	0.2	setosa
Unit 102	5.8	2.7	5.1	1.9	virginica

Data Source: `iris` data set available in the R library "datasets"

Figure 1.4 provides a visual representation of the steps through which the two units have been assigned to a class.

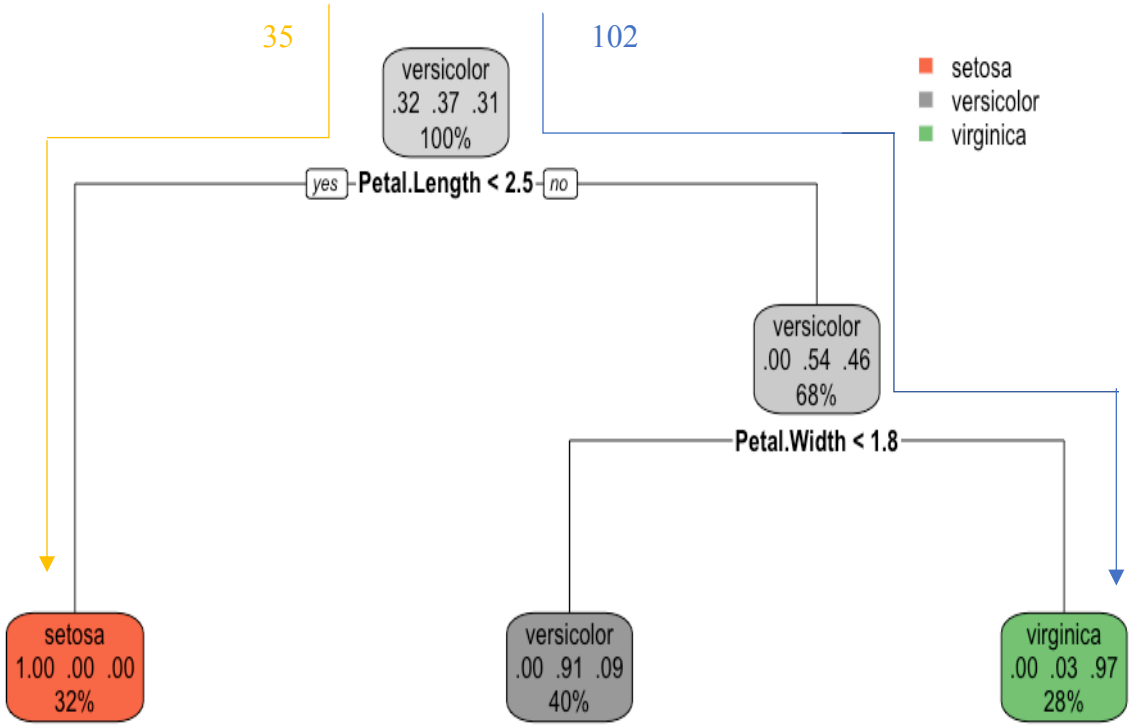
The decision tree identified `Petal.Lenght` as the predictor variable which reduces, as much as possible, the noise present in the data set. Specifically, it does so by recursively splitting the data in smaller samples. The possibilities of metrics used to do so are various (Maimon and Rokach, 2005). In the case shown in Figure 1.1, a univariate splitting approach is used. The procedure is quite straightforward. During the training process it has been found that the `Species` of units with `Petal.Lenght` less than 2.5 is always `Setosa`. Considering the data recorded in Table 1.1, Unit 35 reports `Petal.Lenght` equal 1.5 and, therefore, is predicted as `Setosa`. Regarding Unit 102, instead, the initial split is not enough to make an accurate prediction. The disorder of the data set made of units with `Petal.Lenght` greater than 2.5 can be further reduced. The split which minimizes such disorder is provided by `Petal.Width`. Units with `Petal.Width` less than 1.8 are predicted to belong to the `versicolor` class, otherwise `virginica`. Given Unit 102 reports `Petal.Width` equal to 1.9, it is predicted as `virginica`. The decision tree's accuracy can be evaluated by comparing the predictions made for Unit 35 and 102 with their actual `Species` class.

Figure 1.3: Algorithm for growing a decision tree

```
TreeGrowing (S,A,y)
Where:
S - Training Set
A - Input Feature Set
y - Target Feature
Create a new tree T with a single root node.
IF One of the Stopping Criteria is fulfilled THEN
    Mark the root node in T as a leaf with the most
    common value of y in S as a label.
ELSE
    Find a discrete function f(A) of the input
        attributes values such that splitting S
        according to f(A)'s outcomes (v_1,...,v_n) gains
        the best splitting metric.
    IF best splitting metric > treshold THEN
        Label t with f(A)
        FOR each outcome v_i of f(A):
            Set Subtree_i= TreeGrowing (σ_{f(A)=v_i} S,A,y).
            Connect the root node of t_T to Subtree_i with
                    an edge that is labelled as v_i
        END FOR
    ELSE
        Mark the root node in T as a leaf with the most
            common value of y in S as a label.
    END IF
END IF
RETURN T
```

Source: Maimon and Rokach (2005, p. 169)

Figure 1.4: Prediction of Unit 35 and Unit 102's class, `iris` example



Data Source: `iris` data set available in the R library "datasets"

From this process, the importance of the variables as predictors can be assessed. The variable which is most important in the predictive process is the one that reduces at most the noise within the data (Breiman et al., 1984). Therefore, in this example, the most important predictor is `Petal.Lenght`, followed by `Petal.Width`. `Sepal.Lenght` and `Sepal.Width`, instead, while still being considered in the splitting process, were not actually used to split the data. These predictors, then, are considered less important, meaning that, once the data are split through `Petal.Lenght` and `Petal.Width`, they do not significantly reduce the data set's entropy. Further on in this chapter further considerations regarding the variable importance will be provided, underlying that focusing solely on the importance as measured above is not enough. Further analysis through methods which permit to infer causality among variables may provide useful insights.

1.2.2. Construction of decision trees

While the previous section of this chapter was designed to provide an initial understanding of decision trees and their functioning, this one will present a more technical approach for decision tree construction. A review of Breiman et al. (1984) CART algorithm permits to unravel the process operated by decision trees. Followingly, based on Breiman et al. (1984) and on the survey of Moisen (2008), the fitting process will be analyzed, focusing on the main activities involved, such as choosing the appropriate variables, and splitting and pruning, while providing notions of other issues such as costs, missing data, overfitting, and others.

As previously discussed, the design of decision trees involves two steps, training and testing. Being a supervised machine learner, such algorithm requires inputs in order to create an output. Training the model, therefore, is the initial step in which the algorithm familiarizes and learns from the data, drawing connections and identifying patterns. In order to do so, the tree needs to be appropriately fitted and constructed (Breiman et al., 1984). In other words, then, the performance and correctness of a decision tree depends on the type of data, on the size and depth of the tree, and other elements.

1.2.2.1. CART algorithm

The most important algorithm which provides the functioning of classification and regression trees is the CART algorithm (Breiman et al., 1984). The main differences between classification and regression trees are the structure of the dependent variables and the error measurements. Classification trees are used when the response variable is categorical (e.g. `Species` in the previous example). In this case the error is measured in terms of misclassified units. Regression trees differ from classification trees since they are used with continuous dependent variables, and the error is computed as the squared difference between actual and predicted values (Breiman et al., 1984; Loh, 2011). This distinction is relevant for the determination of the decision tree's output, and the random forests' output which will be analyzed in section 1.3. In this section, for simplicity, the notation is defined as in Breiman et al. (1984).

Breiman et al. (1984) underline two purposes of classification analysis. The most straightforward is to create a classifier which accurately assigns a unit to its class. An additional use of

classification trees is linked to the objective of understanding the underlying reasons for which a unit belongs to a specific class. Often, studies have both purposes. In both cases, the analysis involves the determination of a function which given an input **x** provides an output y.

Suppose a case (in the iris example, a flower unit recorded) is described by a vector **x**, which groups m measurements defined as x_1, x_2, …, x_m; in the classification analysis process, such variables are called predictor variables (in the `iris` example, m = 4: `Petal.Lenght`, `Petal.Width`, `Sepal.Lenght`, and `Sepal.Width`). Define, then, *X* the measurement space including all m cases (in the iris example, X is 4-dimensional). Suppose, finally, that the dependent variable Y assumes 1, 2, …, J values. With classification trees, for instance, j_1, j_2, …, j_J classes (in the iris example, J = 3: `setosa`, `versicolor`, and `virginica`), and let *C* be the set of classes. Finally, Breiman et al. (1984) define a classifier as *"a function d(**x**) defined on X so that for every **x**, d(**x**) is equal to one of the numbers 1, 2, ..., J."* (p. 4). Readapting the definition to the classification purpose of this paper, d(**x**) must be equal to one of the classes j_1, j_2, …, j_J.

As previously stated, the decision tree algorithm involves two steps, training and testing. It should be noted that in both the training and testing process information regarding X and Y is available. The predictive model created, though, will be able to predict Y given only information regarding X.

Through training, the algorithm searches for patterns within the data. Being predictive algorithms, the models constructed find such patterns by using past experience (Breiman et al., 1984). Specifically, the training data works as a learning sample, meaning it provides the information (through X and Y) required so that the model can make predictions on unseen data.

The second step involves testing the model and evaluating its performance. A measure of the accuracy of the CART algorithm is provided by the true misclassification rate $R^*(d)$. Specifically, given the predictive function d(**x**) trained on a fixed learning sample *L*, $R^*(d)$ represents the proportion of cases that are wrongly classified by d(**x**) when it is used on a new sample. Given a new sample (**X**, Y), $R^*(d)$ is defined as (Breiman et al., 1984)

$$R^*(d) = P(d(\boldsymbol{X}) \neq Y|L) \tag{1.1}$$

The right side of the equation represents the theoretical probability of misclassification on the new sample (**X**, Y) given *L*. Breiman et al. (1984) estimate $R^*(d)$ through three internal estimates:

- resubstitution estimate: defining an indicator function *I(·)* which returns 1 if the argument within the brackets is true and 0 if it is false, the estimate R(d) with N cases is given by

$$R(d) = \frac{1}{N}\sum_{n=1}^{N} I(d(\boldsymbol{x_n}) \neq y_n) \tag{1.2}$$

  In other words, for every time the predictive function d(**x**) misclassifies the case, the function *I* returns 1. For all N cases, (1.2) averages the misclassified cases by dividing the count of misclassified cases by N.
- test sample estimate: by dividing the learning sample *L* in L_1 and L_2; L_1 is used for training and the N_2 cases in L_2 are used to estimate $R^*(d)$.

$$R^{ts}(d) = \frac{1}{N_2}\sum_{(x_n, j_n)\in L_2} I(d(\boldsymbol{x_n}) \neq y_n) \tag{1.3}$$

- v-fold cross-validation estimate: the test sample estimate may lead to limitations due to the reduction of data (i.e. split in training and testing sets) available to train the function d(**x**) (intuitively, a higher amount of data contains more information; having more information available, the model can find more reliable and truthful patterns within the data). The v-fold cross-validation methods permits, instead, to use all data. Suppose to divide *L* in V subsamples of the same size $L_1, L_2, \ldots, L_V$. For v = 1, 2, …, V, the v^{th} sub sample is set aside and used for testing, while $L\backslash L_v$ is used for training.

  Suppose to train V functions by using $L\backslash L_v$ as learning sample. A test estimate, then, can be computed

$$R_{cv} = \frac{1}{V}\sum_{v=1}^{V}\frac{1}{N_V}\sum_{(x_n, y_n)\in L_V} I\left(d^{(v)}(\boldsymbol{x_n}) \neq y_n\right) \tag{1.4}$$

  with $N_v = N/V$. Intuitively, this approach permits to use all the information available within the data.

An important feature of the CART algorithms, and machine learning in general, is the ability to deal with diverse data. In 1984 Breiman et al. (1984) observe that *"many of the presently available statistical techniques were designed for small data sets having standard structure with all variables of the same type,* [with] *the underlying assumption* [that] *the phenomenon is homogeneous."* (p. 7), recognising that new statistical models needed to be adaptable and could not lack to deal with high dimensionality, mixture of data types, non-standard data structures, and non-homogeneity. The recent availability increase of big and complex data underlines the importance of both this insight and the advancement of research on machine learning algorithms.

1.2.2.2. Choosing the appropriate variables and variable importance

The construction of a decision tree mainly revolves around its splitting process; through programming (R, Python, and so on), the best split possible is automatically identified by the algorithm, as shown in Figure 1.1 (Breiman et al., 2001; Maimon and Rokach, 2005).

Variable selection refers to the task of providing the algorithm with the best choice of variables possible. Three methods have been determined and are supposed to be effective in determining such explanatory variables:

- a review analysis regarding past data and the environment of research: while this method does not provide statistically significant results, it does provide a deep understanding of the case study, leading to determining which group of variables to focus on;
- creation of a regression model: by constructing a regression, an analysis trough R or other programming language can easily lead to identifying the significant variables and the variability accounted by each variable. The functioning of such process may be hard to achieve for two reasons. Firstly, variables cannot be correlated, and finding similar variables to use instead of correlated ones is not always easy to obtain. Secondly, with a high number of variables, the size of the dataset needs to be big enough so that the regression doesn't overfit;
- trial and error approach: in the case of data sets described by a great number of variables this method may be long and costly; moreover, in the case of decision trees and random forest, the relevance of this approach is undermined given the algorithms ability to asses and quantify variable importance (Breiman et al., 1984; Breiman, 2001). Suppose m

possible variables can be initially used to describe a specific dependent variable. By recursively removing variables, or changing the combination of variables considered, a predictive model can be created for each test. This permits to obtain measurements of variable importance, errors, and accuracy. Comparing the results obtained for each trial, supposing accuracy is the sole indicator used to evaluate the model's performance, the combination of variables that provides the best accuracy can be used for the analysis. Moreover, this approach permits to obtain some information to evaluate the possibility of discarding those variables which are repeatedly assessed as not important. It should be noted, though, that each test will provide a list of variables ordered by importance based on the variable considered by that test. When addressing the concept of variable importance, then, it should be kept in mind that the importance is relative to the decisions made in the creation of the test, specifically which and how many variables are considered. Assumptions regarding true variable importance are possible, then, only if the model contains a great number of predictor variables.

During the analysis presented in Chapter 3, a literature review has been implemented in order to choose the variables on which to construct the predictive model. A trial and error approach, instead, has been used to assess the variable importance and construct a predictive model with fewer variables. Specifically, to reduce the possibility of misjudgments regarding variable importance as explained above, rather than using the results of a single test, a score has been attributed to each variable according to importance provided by the algorithm. By summing the scores of each variable across various tests, a final variable importance measurement has been recorded (see Table 3.12 in Chapter 3).

1.2.2.3. The splitting process

The splitting process assumes great importance in the development of the model; given that all splits - from the root node until the leaf nodes - are required to create two new groups with the lowest entropy (Breiman et al., 2001; Maimon and Rokach, 2005), measuring the impurity of the splits gains relevance to improve the model. A node is said to be impure when it contains more than one class of the response variable (Breiman et al., 1984). The reason for which the algorithm

is expected to reduce impurity is that pure nodes, with only one class, provide no prediction errors. In other words, creating dataset with a homogeneous composition with respect to the dependent variable permits to make more accurate classifications and predictions. Nevertheless, the aim is to reduce the noise within the data, and not to obtain perfectly pure nodes, which would lead to overfitting.

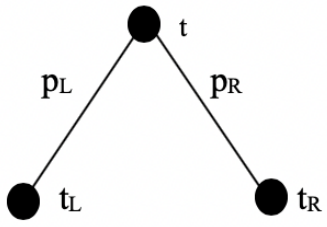
Breiman et al. (1984) initially describe structured classification tress and observe four elements which permit to construct a decision tree through splitting: a set of binary questions on which to split the data, a goodness of split criterion, a stop-splitting rule, and a rule for assigning every terminal node to a class.

Overall, given a node t, for each variable, the best binary questions that split the data are identified. By comparing the various splits, then, the best one is selected according to specific impurity measurements.

> *"An impurity function φ defined on the set of all J-tuples of numbers ($p_1, \ldots, p_J$) satisfying $p_j > 0, j = 1, \ldots, J, \sum_j p_j = 1$ with the properties*
>
> - *φ is a maximum only at the point $\left(\frac{1}{j}, \frac{1}{j}, \ldots, \frac{1}{j}\right)$*
> - *φ is a minimum only at the points (1, 0, ..., 0), (0, 1, 0, ..., 0), ..., (0, 0, ..., 0, 1)*
> - *φ is a symmetric function of $p_1, \ldots, p_j$*" (Breiman et al., 1984, p. 32)

Consider the node t in Figure 1.5; a split s leads to two outcomes: the proportion p_L of cases that meet the criteria defined by the binary question are moved in the left node t_L, otherwise the proportion p_R is sent to the right node t_R.

Figure 1.5: Split of node t in t_L and t_R



Source: Breiman et al. (1984)

Through the impurity function φ, Given the impurity measure i(t)

$$i(t) = \varphi\big(p(1|t), p(2|t), \dots, p(J|t)\big) \tag{1.5}$$

Breiman et al. (1984) provide a measurement of the goodness of split s through the decrease in impurity given by

$$\Delta i(s,t) = i(t) - p_R i(t_R) - p_L i(t_L) \tag{1.6}$$

meaning the impurity provided by a split is given by the difference of impurity between the node t and the sum of impurity provided by the nodes t_R and t_L weighted by the corresponding proportions. Given a series of splits along a binary tree T, and give M terminal nodes 1, …, M, the tree impurity is provided by I(T)

$$I(T) = \sum_{t=1,\dots,M} I(t) = \sum_{t=1,\dots,M} i(t)p(t) \tag{1.7}$$

in which p(t) is the estimated probability that any case will be moved to node t. Finally, by maximizing the decrease in impurity across the nodes, the tree impurity is minimized. In fact, given a tree T, for any node t = 1, 2, …, M, by splitting the node t into t_R and t_L, a new tree T' is obtained. The difference between I(T) and I(T') provides the decrease in impurity in the tree as shown in Breiman et al. (1984)

$$I(T) - I(T') = -I(t) - I(t_L) - I(t_R) \tag{1.8}$$

Funally, as (1.8) depends solely on the node t and on the split s, in order to obtain the maximum impurity reduction, it is sufficient to maximize

$$\Delta I(s,t) = I(T) - I(t_R) - I(t_L) = \Delta i(s,t)p(t) \tag{1.9}$$

These procedures obtained by Breiman et al. (1984) permit to grow a decision tree. Initially, the splitting process is set to stop when the maximum $\Delta I(s,t)$ is less than a set threshold $\beta > 0$.

The limitations of such tree, though, are linked to the threshold set, since this provides the depth of the trees. Intuitively, identifying stopping points a priori highly influences a decision tree's growth.

Finally, the class attributed to a node is linked to the resubstitution estimate, meaning that the class with the highest proportion in a terminal node is assigned to the node.

The previous paragraphs explain the generating steps that lead to the growth of structured classification tress. Although several limitations are linked to this process, it provides a useful introduction to right sized tree found by Breiman et al. (1984). They show, in fact, that better results are obtained by pruning fully grown trees rather than finding the most suitable stopping point for splitting.

Practically, the tree is grown by taking into consideration the Gini Index in the splitting process as the measure for node impurity (Breiman et al., 1984) and

$$i(t) = \sum_{i \neq j} p(i|t)p(j|t) \tag{1.10}$$

provides a measure of the impurity of node t. Moisen (2008) underlines that the Gini Index is widely used in practice, and it is also presented as

$$i(t) = \sum_{j=1}^{J} p_{tj}(1 - p_{tj}) \tag{1.11}$$

with J classes and p_{ij} the proportion of class j in the node t.

A Gini index equal to 0 reflects perfection; higher values of the Gini index mean a higher misallocation of classes within the node. The code used to compute the Gini Index is shown in Appendix B.

Recall the decision tree in Figure 1.1. The first binary question chosen by the algorithm to split the data is "Is `Petal.Length` < 2.5?". If the answer is yes, then the units move to the node on the left, otherwise to the one on the right. Similarly, the second binary question to split the data is "Is `Petal.Width` < 1.8?". The results of computing the Gini Index of the root node and of the following splits are shown in Table 1.2.

Table 1.2: Gini index for splits in Figure 1.1, `iris` example

	Gini Index
Root node	0.6666667
Split: "Is Petal.Length < 2.5?"	0.3333333
Split: "Is Petal.Width < 1.8?"	0.3906076

Data Source: `iris` data set available in the R library "datasets"

Intuitively, the Gini Index for the root node is the highest. Moreover, given that splits are chosen in a decreasing order in terms of amount of impurity reduction, the first split reports the lowest Gini Index (i.e. it accounts for the highest impurity reduction).

While other methods for splitting are available, Breiman et al. (1984) found that misclassification rate is not linked to the splitting rule chosen (p. 94). Therefore, given the frequent use of the Gini Index in practice, the algorithm used in this research will also use the Gini Index. The reader is referred to Breiman et al. (1984, p. 93-129) for additional methods and specifics.

1.2.2.4. The pruning process

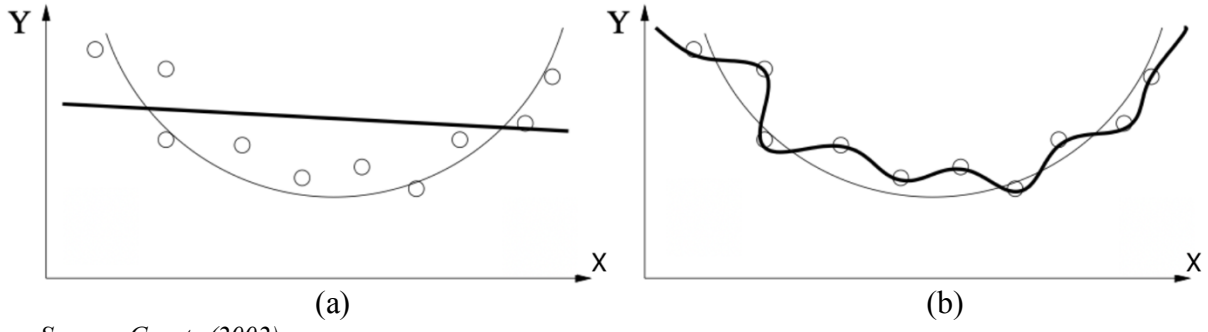
In the previous section important results by Breiman et al. (1984) have been introduced. They found that the tree structured classifiers do not always provide acceptable results due to several limitations. Specifically, the main concern is linked to finding right tree size by searching for the best stop-splitting point. By shifting the focus from finding the best stop-splitting rule to pruning, more efficient classifiers were obtained. In fact, the main issue with setting a threshold β (and choosing a stopping criteria, see Figure 1.3) is that often the tree's growth is stopped too early, therefore the impurity of nodes remains high, or too late, and the trees result being too deep, leading to an problem called overfitting. It should be noted that having pure leaf nodes is not the objective of the splitting process, as will be discussed followingly.

With tree's depth and the splitting process an additional consideration relevant to pruning arises, involving bias and variance. Suppose a model provides a predictive function d(**x**), and suppose the sample (**X**, Y) is given. Two extreme scenarios can be observed (Geurts, 2002):

- Too simple model: suppose the model is linear. As Figure 1.6 (a) shows, such model would not be able to extract significant data patterns from the learning sample. It is, in fact, too simplistic. Such model leads to underfitting;
- Too complex model: suppose the model is highly complex and is able to identify accurately both data patterns and the noise within the data. In other words, the model is so complex that it is suitable for the sole learning set on which it is trained (Figure 1.6 (b)). Such model overfits the data.

Considering the data points and model lines in Figure 1.6, it is intuitive that the application of these two models on new data will not provide accurate results. The reason for which Geurts (2002) considers these models is because they permit to shed some light on prediction errors. In the simple linear model, the error is mainly attributable to the lack of complexity in the model; such error is called bias. In the complex model, instead, the error is linked to the variability that the model manifests when the data is changed; such error is defined variance (Geurts, 2002). Intuitively, there is a trade-off relationship between the two errors, given that they vary in opposite directions according to the complexity of the model. Finally, the pruning process attempts to solve these issues by permitting to avoid the choice of setting a threshold β and using stopping criteria for the tree's growth. Moreover, by finding the best bias-variance trade-off, through pruning an appropriate tree size is found.

Figure 1.6: Too simple and too complex models, examples

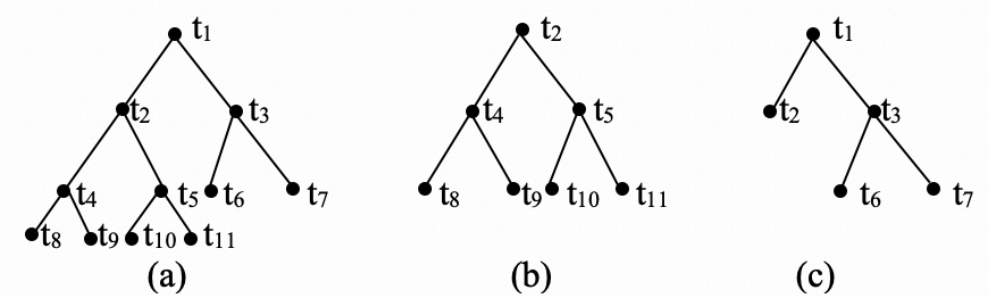


Source: Geurts (2002)

Pruning begins from the terminal leaf nodes of a fully-grown tree, and it reduces the tree's complexity upwards by eliminating branches (Breiman et al., 1984). Let T_{MAX} be a fully-grown

tree, who's terminal nodes are as pure as possible (i.e. no stop-splitting rule is used). Let t_1 be the root node. Let a branch T_t of T be the sub-tree with root node t and its descendants. Pruning a branch provides a pruned tree defined as T - T_t (Breiman et al., 1984). Figure 1.7 provides an example of this process.

Figure 1.7: Branch T_{t_2} pruning



Source: Breiman et al. (1984)

The pruned tree T' in Figure 1.7 (c) is the result of $T - T_{t_2}$, removing the branch T_{t_2}, comprising nodes t_2, t_4 and t_5, and t_8 to t_{11} (Figure 1.7 (b)), from the T_{MAX} tree in Figure 1.7 (a). As it can be noted, when a branch is removed, only branch's root node (i.e. t_2) is not removed. T' is defined a pruned subtree of T_{MAX}.

While Figure 1.7 provides a simple example, consider T_{MAX} to be a fully-grown tree, with many branches and several terminal nodes. In order to choose the best pruned tree, it is necessary to be able to compare the subtrees. Breiman et al. (1984) find that the most appropriate measurement for comparing subtrees is the tree misclassification cost R(T). It can be observed that stating from the terminal nodes, as pruning moves upwards across the branches, R(T) decreases until it reaches a minimum, before increasing and finally reaching the root node. This is linked to the bias-variance trade-off previously described. The best depth for the decision tree is the size which provides the minimum R(T) (i.e. the best trade-off bias-variance).

Breiman et al. (1984) define the minimal cost-complexity pruning.

"For any subtree $T < T_{MAX}$, define its complexity as $|\tilde{T}|$, the number of terminal nodes in T. Let $\alpha \geq 0$ be a real number called complexity parameter and define the cost-complexity measure $R_\alpha(T)$ as

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}| \text{ "} \text{(p. 66)} \tag{1.12}$$

Therefore, for any α, the tree $T(\alpha)$ that minimizes $R_\alpha(T)$ can be found.

$$R_\alpha(T(\alpha)) = min_{T<T_{MAX}} R_\alpha(T) \tag{1.13}$$

Given equation (1.13) three scenarios can be observed with varying α:

- Small $\alpha = 0$: T_{MAX} minimizes $R_0(T)$ assuming that terminal nodes are pure;
- Increasing α: $T(\alpha)$, subtree of T_{MAX}, minimizes $R_\alpha(T)$;
- Large α: T_{MAX} is completely pruned and $T(\alpha)$ corresponds to the root node.

Finally, in order to estimate $R_\alpha(T(\alpha))$ Breiman et al. (1984) find both methods the independent test set and v-fold cross-validation appropriate.

Several R packages provide functions which deal with pruning. In this study, `rpart` has been used to generate and prune decision trees. Specifically, considering the `iris` example, once a decision tree model is constructed through the `rpart()` function

```
DT -> rpart(Species~., data= training_data, method= "class")
```

the function `prune()` permits to effectively prune the decision tree by setting the argument `cp`, which represents complexity parameter. By using the table provided by `rpart()` which contains various complexity parameters associated with error measurements (i.e. the `cptable`) the complexity parameter that minimizes the cross-validation error can be found.

```
prune(DT_model, cp = DT$cptable[which.min(DT$cptable["xerror"]), "CP"])
```

The code above is set to search, among the various `cp` values grouped in `cptable`, for the one that minimizes the cross-validation error (i.e. `xerror`).

1.2.3. Performance evaluation

1.2.3.1. Overfitting and underfitting

As previously stated, an analysis trough machine learning aims to make predictions based on a previous learning steps. In fact, two steps are identified, training and testing. Reconsider the example of Figure 1.1; readapting Dietterich's (1995) considerations to this example, training means providing the model with the information required regarding `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width` (i.e. the predictors), and `Species`, so that connections can be made between the explanatory variables and the dependent one (`Species`). The objective is to create a model which, when provided with new data and no `Species` information, can make accurate classifications.

Importantly, one should focus on the prediction accuracy measured on the test set, and not on the training set. Machine learning algorithms are highly adaptable and generalizable, and such feature should be preserved throughout the test. Recall the "too simple" and "too complex" models presented in the previous section in order to introduce the bias and variance error (Figure 1.6). When a model learns specific features of a dataset, it is said to be overfitted since it tends to work only for the specific data (Dietterich, 1995). In other words, the model is too complex and fixed to a situation (see Figure 1.6 (b)) and will result in non-appropriate predictions when the study involves new data.

Overfitting can generally be identified by observing the training error, which will be low, and the test error, which will be high. On the other hand, similar considerations are true for underfitting, meaning the circumstances in which the model created does not account for enough traits of the training sample (see Figure 1.6 (a)), is of low complexity, and therefore is not able to draw appropriate connections and conclusions on new data (Crowley and Ghojogh, 2019).

A model which overfits (or underfits) the data will provide poor accuracy once used with unseen data. Therefore, these issue lead to a model which underperforms and is used below its potential. The most successful method for avoiding such limitation is pruning the decision tree appropriately.

The overfitting issue leads to one last consideration. The complexity of a model should be included in its evaluation. The application of the Occam Razor principle to machine learning has gained importance when dealing with overfitting (Kelly, 2007).

1.2.3.2. Performance

Machine learning algorithms, with appropriate tuning, can be powerful tools; it is not possible, though, to establish which is the best algorithm (Armitage et al., 2006). The evaluation of the performances of different algorithms, applied to various situations, permits to create a network, connecting the algorithms to the scenarios in which they operate better. Considering decision trees, within such category many different algorithms can be found. While in this research CART types trees are studied, for instance, Quinlan (1996) presents other two different algorithms and their characteristics, the ID3 (Iterative Dichotomiser 3) and the C4.5 algorithms.

1.2.3.2.1. Testing: accuracy and errors

A first approach to identifying how well a model is performing is to run it on a test data set. Returning to the iris example, in Figure 1.3 a first simple application of prediction has been presented. To perform the true test, the function `predict()` has been used in order to generate predictions. Such predictions have then been compared with the actual values of the `Species` variable. Below, the confusion matrix of a cross-sectional analysis with a testing set containing 15 cases is provided.

Table 1.3: Confusion matrix, testing iris - 90% training 10% testing, `iris` example

	setosa	versicolor	virginica
setosa	7	0	0
versicolor	0	0	0
virginica	0	0	8

Source: iris data set

The confusion matrix permits to display the results of the predictions (columns) against the actual values of the dependent variable (rows). This information is then used to compute the accuracy

and the error of the tree. A confusion matrix presents the correct predictions on the diagonal and the wrong predictions elsewhere.

Considering for instance a classification example in which the set of classes C_1 contains two classes j_1 and j_2. Moreover, consider a classification test in which the set of classes C_2 contains m j_1, j_2, …, j_m classes. The confusion matrix for these tests are provided, respectively, by Table 1.4 and 1.5.

Table 1.4: Confusion matrix with two classes

	Predicted j_1	Predicted j_2
Actual j_1	a	b
Actual j_2	c	d

Table 1.5: Confusion matrix with m classes

	Predicted j_1	Predicted j_2	…	Predicted j_m
Actual j_1	$n_{1,1}$	$n_{1,2}$	…	$n_{1,m}$
Actual j_2	$n_{2,1}$	$n_{2,1}$	…	$n_{2,m}$
…	…	…	…	…
Actual j_m	$n_{m,1}$	$n_{m,2}$	…	$n_{m,m}$

Considering the binary classification algorithm (Table 1.4), for example, the confusion matrix counts true positives (i.e. predicted positive and actually positive), true negatives (i.e. predicted negative and actually negative), false negatives (i.e. predicted negative, but actually positive), and false positives (i.e. predicted positives but actually negative).

Having these figures, it is possible to compute the error and accuracy levels for binary classification (Ralescu et al., 2011) as follows:

$$Accuracy = \frac{a + d}{a + b + c + d} \tag{1.14}$$

$$Error = \frac{b + c}{a + b + c + d} \tag{1.15}$$

To generalize, given Table 1.4, let S be the sum of the elements in the confusion matrix. (1.14) and (1.15) can be written as

$$Accuracy = \frac{\sum_{i=1}^{m} n_{i,i}}{S} \tag{1.16}$$

$$Error = \frac{\sum_{i=1}^{m} \sum_{j \neq i} n_{i,j}}{S} \tag{1.17}$$

The predictions of a 15-sample testing set provided no errors. This example shows the importance of considering the size of the dataset. Having used 90% of the data for training, only 15 units are available for testing.

In order to comprehend the difference, an additional test has been carried. The data set has been divided in order to use 70% for training, and 30% for testing.

Table 1.6: Confusion matrix, testing iris - 70% training 30% testing, `iris` example

	setosa	versicolor	virginica
setosa	17	0	0
versicolor	0	14	4
virginica	0	0	15

Data Source: `iris` data set available in the R library "datasets"

These results are more useful to present the classification error. Four times out of 50 units (size of the new testing data set) the decision tree predicted the wrong `Species`. While this method is useful, its results change every time a new random sample is created, highlighting one of the the limitations of such method. A solution to this issue is provided by more integrated and sophisticated algorithms, such as, for example, random forests.

1.3. RANDOM FORESTS

Having understood the functioning and main characteristics of decision trees, the following sections expand the machine learning analysis to random forests. The latter have been an important advancement in the development of regression and classification predictors, overcoming, among other things, the overfitting issue (Breiman, 2001). While the trademark for "Random Forest" has been registered by Leo Breiman and Adele Cutler in 2006, a similar approach to random forest has been developed in Ho (1995).

This section is divided as follows: first, a technical part will provide the definition and the functioning of random forests and will continue the classification of `Species` example started in section 1.2; secondly, the main characteristics and features necessary to construct a random forest will be analyzed, with specific attention to programming using R; thirdly, the main techniques to evaluate the performance of random forests are presented.

1.3.1. Introduction

A random forest algorithm integrates several decision trees, and, opposingly to the use of single decision trees, aims to create many random situations which all provide their own prediction. Many studies in the literature prove that ensemble classifiers have greater accuracy than weak learners. Specifics regarding the methods used to create ensemble models are provided in section 1.3.1.2. Random forests, by constructing a high number of trees, are able to include a much wider array of combination of variables split points, making such algorithm more accurate and reliable than singular decision trees. Importantly, its adaptability and versatility have led to the application of the random forest algorithm developed by Breiman (2001) in various fields, such as engineering (Dixon et al., 2015), environmental sciences (Artemenko et al., 2009), financial studies (Kumar and Thenmozhi, 2006), medicine (Kane et al., 2014), 3D object recognition (Criminisi et al., 2013), and many other.

1.3.1.1. Definition

Decision trees, as previously discussed, by splitting recursively each decision node according to certain parameters, permit to predict a class or a number, given a learning set L = (X, Y).

Random forests provide the same outcome by integrating the results of all the trees it comprehends. Breiman (2001) provides the following definition of random forest:

Definition 1.1

"A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(\boldsymbol{x}, \Theta_k)' \; k = 1, \ldots\}$ *where the* $\{\Theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* $\boldsymbol{x}$*."* (p. 6)

Random forests, like decision trees, are used for both regressions and classifications. For the purpose of this research, only classification will be considered. Classification random forests, expanding Definition 1.1, are ensembles of classification decision trees which, given an input dataset, get one vote. In other words, given a case (**x**, y), and given 1, 2, …, J classes, each tree classifies the case by predicting one class j. Each tree expresses a vote for the class it predicts. The algorithm, then, proceeds to group all votes and outputs the class with the greatest number of votes. An advantage of the random forest algorithm is that for any sequence of decision tree classifiers it does not overfit the data, because of the randomness injected into the model (Breiman, 2001).
The reason for which random forests, and ensemble models in general, have been developed is to create strong complex learners by integrating several weak learners (Breiman, 1996; Freund and Schapire, 1996). Specifically, weak learners are those models that do not perform well on their own, and that are not able to generalize their results. Moreover, ensemble classifiers permit to address the bias-variance trade-off issue introduced in section 1.2. Finally, ensemble models' functioning, and adaptability, provide better results and greater accuracy (Breiman, 1996; Breiman, 2001; Freund and Schapire, 1996).

1.3.1.2. Ensemble methods: bagging

There are several ways in which machine learning algorithms operate and train their models in order to create ensemble models. Specifically, Dietterich (1999) provides a review of bagging (Breiman, 1996) and boosting (Freund and Schapire, 1996), the two main methods for constructing ensemble classifiers, explaining similarities, differences, and the procedures through which ensemble models are created from the union of many weak learners. Weak learners are defined as

classifiers whose performance is slightly better than sampling randomly among j classes. Ensemble methodologies' objective is to create strong learners from the interaction of weak classifiers, therefore obtaining models with higher accuracy (Freund and Schapire, 1995). Moreover, Dietterich (1999) provides a third method, randomization, which has shown performance comparable with those of bagging and boosting.

The purpose of this section is to obtain an understanding of the bagging procedure, relevant to comprehend the random forest approach used in this research. In order to comprehend the wide variety of possible approaches to prediction methods, the reader is referred to Breiman (1996), Freund and Schapire (1996), and Dietterich (1999). The choice of method, in fact, may be linked to various factors; this provides that algorithms using these three methods (e.g. random forests, AdaBoost) perform differently according to the situation or data to which they are applied. For instance, Dietterich (1999) stresses that the three methods considered in this section perform differently according to the stability of the data set (i.e. the noise present in the data). Given a learning set which is unstable, bagging proved to perform better than randomization, and much better than boosting, given its ability of exploiting such noise to create more diverse classifiers. On the other hand, with low noise scenarios, boosting performs favourably to bagging and randomization, which, instead, work similarly (Dietterich, 1999). Algorithms implementing such methods are often compared in the literature (Breiman, 2001; Freund and Schapire, 1996, 1999; Dietterich, 2000; Cheung-Wai Chan and Paelinckx, 2008). Often the performance of algorithms such as AdaBoost, developed by Freund and Schapire (1996), which implements boosting, and random forest, which applies bagging, are observed simultaneously. Moreover, randomization should be considered given that the random forest algorithm used in this research implements bagging and a revised randomization procedure.

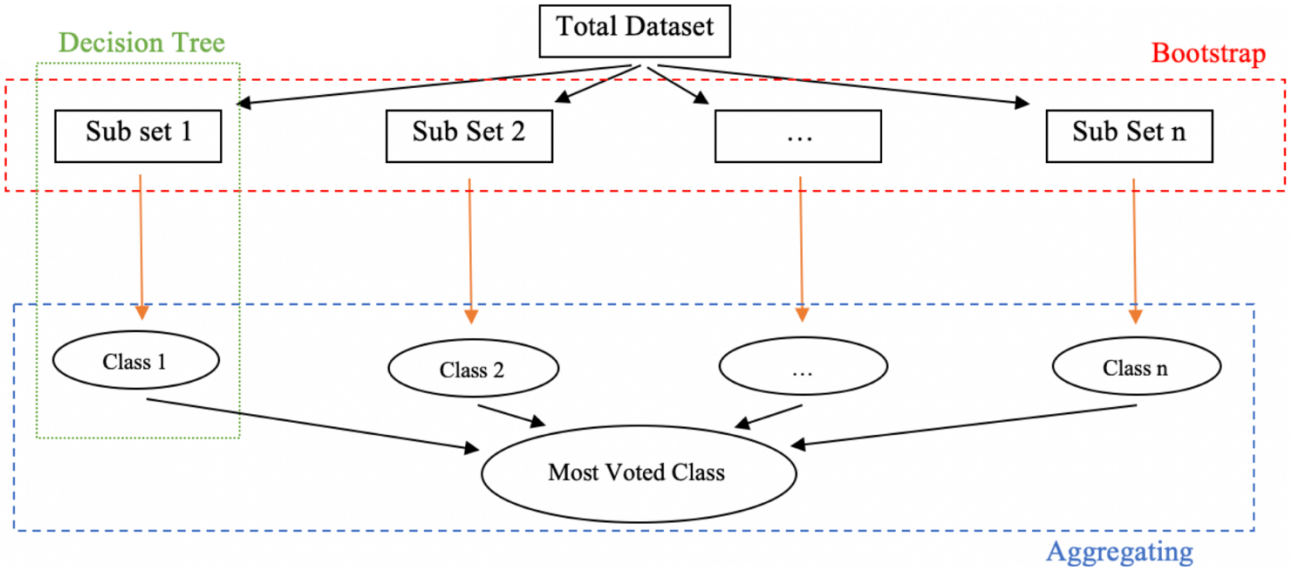
1.3.1.2.1. Bagging

Developed by Breiman (1996), bagging is a procedure that consists in bootstrap aggregating. In other words, bagging permits, by using the same learning sample, to obtain various classifiers. In turn, aggregating the results provided by such classifiers, provides an aggregated strong learner. The ensemble of weak learners consists of two distinct phases:

- Bootstrap: Given a dataset of size N, it involves the random selection, with replacement, of a sub-datasets of size N. These, in turn, will be used as the starting root node for the decision trees. By recursively repeating this step, numerous tree classifiers are generated (Breiman, 1996). While Figure 1.8 provides a single decision tree for each sub set, Breiman (1996) provides that 50 trees are grown on each sub sample.
- Aggregating: the second and final step involves grouping the n independent learners and averaging its results. In fact, by putting together the n classifiers, and counting each one's vote, the class which obtained the most votes is selected (Breiman, 1996).

Figure 1.8 has been designed to show a simple visualization of such procedure. For a more detailed list explaining the bagging procedure, see Breiman (p.125, 1996).

Figure 1.8: Bagging procedure



Let N be the number of cases, let Y be the dependent variable, let **x** be the set of m variables x_1, x_2, …, x_m used to predict Y. Given a learning set $L = \{(y_n, \mathbf{x}_n), n = 1, 2, \ldots, N\}$, suppose a function φ is available such that (Breiman, 1996)

$$y = \varphi(\boldsymbol{x}, L) \tag{1.18}$$

meaning the function φ, given **x** and L, predicts y.

As in Breiman (1996), during the bootstrap step, a sequence of learning sets $\{L_k\}$ is sampled with replacement from the learning sample L. Each learning sample in $\{L_k\}$ contains N cases $(y_n, \mathbf{x}_n)$. In the aggregating step, instead, the procedure provides for an aggregation of the single classifiers' output.

Suppose Y consists of j =1, 2, …, J classes; suppose that K learning set are computed and suppose a classifier is constructed on each learning set. Then, every classifier will provide a class j among the available J classes. Finally, let φ_A be the procedure that provides the aggregation of all votes.

$$\varphi_A(\boldsymbol{x}) = argmax_j N_j \tag{1.19}$$

with $N_j = nr\{k;\ \varphi(\boldsymbol{x}, L) = j\}$, meaning that for each leaning set k, N_j records the output class j. Then, the function argmax() provides the class j for which N_j is maximum (Breiman, 1996). While this study focuses on classification, the same is true for regression, in which the average of predictions provides the model output (Breiman, 1996).

$$\varphi_A(\boldsymbol{x}) = av\ \varphi(\boldsymbol{x}, L_k) \tag{1.20}$$

While boosting trains its learning samples in a sequential way, as shown in Figure 1.8, a distinctive feature of bagging is that it involves training the decision trees independently, and only averaging the outputs in the final step. Breiman (1996) shows how adopting the bagging method permits to obtain higher accuracy, greater model stability, and lower variance. Moreover, bagging has proven to be mostly useful with unstable learning samples. Simply, if changes in L do not generate great changes in the aggregation of the votes, then the learning sample's stability is high (Breiman, 1996). While the difference in the resulting aggregation of votes may not be very different with stable data, in case of unstable data the improvement in accuracy resulting from the addition of bagging is significant.

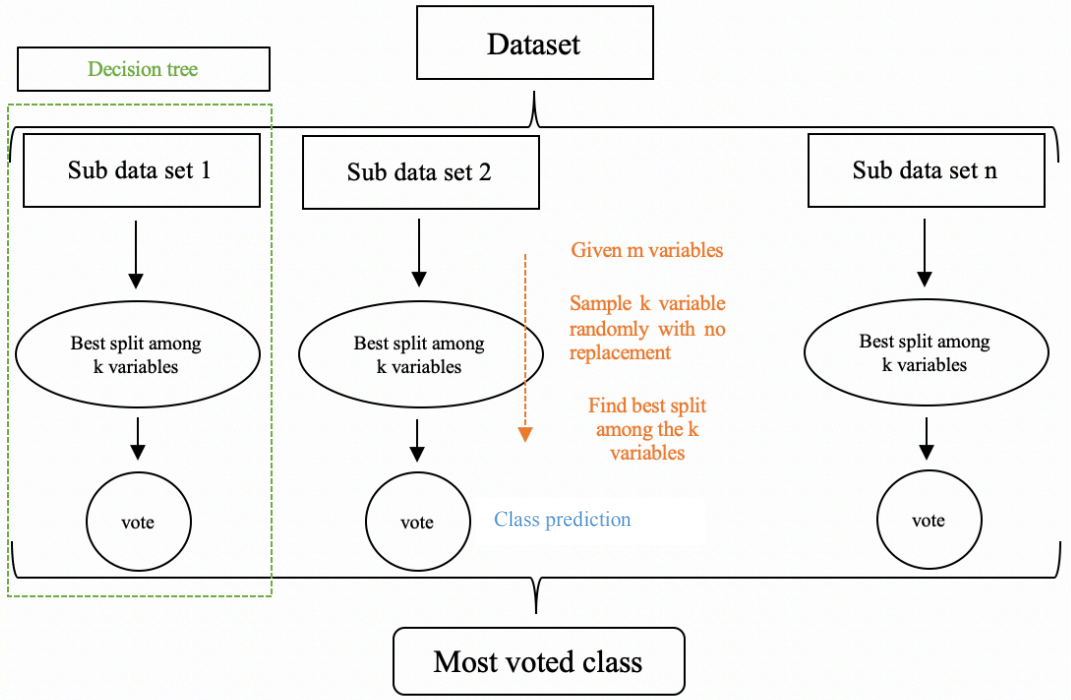
1.3.1.3. Functioning of random forests

Intuitively, the functioning of random forests is in line with that of decision trees. In fact, suppose 100 trees are constructed separately and grown with the characteristics described in the section 1.2. This situation provides 100 class predictions but may be limited in terms of model accuracy and adaptability. Definition 1.1 may lead to believe that a random forest is simply a collection of trees. The methods which compose the random forest algorithm, though, make a significant difference in the efficiency of this machine learning technique.

As previously discussed, tree predictors (i.e. weak learners) are generated in order to create an ensemble classifier (i.e. strong learner). Definition 1.1 states that random vectors Θ_k provide for the growth of such trees (Breiman, 2001). Random forests adopt a bagging approach to create the ensemble model but integrate this through random input selection and linear combination of inputs are analyzed by Breiman (2001). In fact, while bagging provides the random subsets on which the decision trees are grown, in order to differentiate such trees and to reduce the correlation among them, the algorithm does not simply choose the best split (see *Decision Tree - Splitting*). Instead, at each node the algorithm randomly selects the best split among a predefined number of splits (Breiman, 2001; Dietterich, 1990). For example, suppose n predictor variables are available; at each node k variables will be selected randomly. Finally, once the Gini Indexes are computed for each variable, the best one is used to split the node. Breiman (2001) found that such procedure permits to obtain higher accuracy. Moreover, this method permits to construct independent poorly-correlated trees. The importance of this aspect is related to the fact that correlated trees would tend to provide the same predictions, impacting the strength of the ensemble model.

Let k be the number of variables to be selected for each node split. Building on Figure 1.8, Figure 1.19 provides a visual representation of the procedure described. assuming k splits are given to choose from. For simplicity, one split per tree is shown; this does not provide a limitation to understanding the functioning of the random forest algorithm.

Figure 1.9: Random forests functioning and split randomization



1.3.1.3.1. Out-of-bag estimates

Breiman (2001) identifies two reasons for adopting a bagging approach when growing random forests:

1) when random features are used, bagging improves accuracy. In Breiman (2001) the random features tested are: random input selection and linear combination of inputs. Regarding the random selection of features, Breiman (2001) study has been influenced by Ho (1995) and Amit and Geman (1996);
2) bagging permits to acquire continuous estimates of the generalization error PE^* of the ensemble method by computing the out-of-bag estimates.

Suppose K classifiers are trained on the training space (Y, **X**). Given an indicator function $I(\cdot)$, define the margin function $mr(\boldsymbol{X}, Y)$ as the *"extent to which the average number of votes for the right class exceeds the average votes for any class"* (Breiman, 2001, p.7).

$$mg(\boldsymbol{X}, Y) = av_k I(h_k(\boldsymbol{X}) = Y) - \max_{j \neq Y} av_k I(h_k(\boldsymbol{X}) = j) \tag{1.21}$$

The generalization error PE* of the ensemble method, then, is the probability that the margin function over the space (**X**, Y) is negative (Breiman, 2001). The generalization error, also known as out-of-bag error (OOB error) is a measurement of the model's accuracy when tested on unseen data.

$$PE^* = P_{\boldsymbol{X},Y}(mg(\boldsymbol{X}, Y) < 0) \tag{1.22}$$

As previously introduced, one of the advantages of using random forest is that they do not tend to overfit the data when the number of trees in the forest increases. This is linked to an important finding by Breiman (2001) regarding the generalization error's convergence to

$$P_{\boldsymbol{X},Y}\left(P_\Theta(h(\boldsymbol{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\boldsymbol{X}, \Theta) = j) < 0\right) \tag{1.23}$$

for any set of classifiers Θ (see Definition 1.1). Generalizing (1.22), the margin function over Θ is defined as

$$mr(\boldsymbol{X}, Y) = P_\Theta(h(\boldsymbol{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\boldsymbol{X}, \Theta) = j) \tag{1.24}$$

The result obtained in (1.23), then, represents the probability over the **X**, Y space of the margin function for the random forest (Breiman, 2001). In other words, the generalization error PE* has an upper bound. Beside the mathematical reasoning behind (1.23), such finding is relevant to comprehend random forests' behavior.

Having defined PE* and (1.23), an important characteristic of random forest can be analyzed. The trees in random forests are fully grown (Breiman, 2001). Generally, fully grown trees overfit the data and are characterized by high variance (Image 1.6 (b)). Decision trees, in fact, require proper tuning in order to find a proper trade-off between bias and variance. The same is not true for random forests. Creating fully grown trees, together with random selection of features and training on different samples, permits to reduce the variance and maintain the low association among classifiers. While including a great number of fully-grown trees would lead to believe that

the random forest's error should be increasing with the size of the forest, (1.23) provides stability to the model. In practice, with big data sets, in order to reduce the computation cost, the depth of the trees is slightly reduced. This permits to reduce the time needed to generate the forests, while not impacting significantly the result.

Moreover, (1.23) permits to understand the second reason for which bagging is used in random forests. Suppose k bootstrap samples are extracted from the learning sample $L = \{(\mathbf{X}, \mathbf{Y}), x_i \in X , y \in Y\}$. Let k classifiers $h_k(\mathbf{x}, L_k)$ grow on the L_k samples. Given that the L_k samples are created by randomly selecting with replacement inputs from L, not all classifiers h_k are trained using all the inputs. It is therefore possible to select, for each (y, **x**), the final votes provided by the classifiers which were not trained on (y, **x**). In other words, to these classifiers, (y, **x**) is unseen data and can therefore be used to obtain an error measurement. Given that such classifiers are called out-of-bag classifiers, the generalization error PE^* is also called out-of-bag error.

An alternative approach to bagging and out of bag samples is provided by k-fold cross validation. Recall from section 1.2 that such method is used by CART decision trees to evaluate the misclassification rate through (1.4). Similar to bagging, the training set is divided in K sub samples of equal (or approximately equal) size. For k = 1, 2, …, K, the k^{th} sub sample is set aside and used for testing, while the other are used for training. By repeating this step K times, cross-validation permits obtain ongoing error measurements similar to out-of-bag estimates.

The main difference between bagging and cross-validation is the size used for testing the classifiers. Nevertheless, some studies suggest that with a high number of trees, the use of the out-of-bag error or the error computed through the cross-validation procedure are very similar (Kim, 2009). Overall, both methods permit to reduce the variance of model prediction.

Building on (1.23), Breiman (2001) introduces the concept of random forest's strength and correlation. Moreover, his finding shows that the generalization error decreases, and therefore the model's accuracy improves, when correlation decreases, and strength increases.

Provided that the margin function is a measure of how well the classifier predicts a class (i.e. higher margin provides better classification), Breiman (2001) introduces the concept of strength s as the expected value of the margin function; given (1.21),

$$s = E_{\boldsymbol{X},Y}(mr(\boldsymbol{X}, Y)) \tag{1.25}$$

Intuitively, the value of s is proportional to the value of the margin function, as with an increase in correctly classified classes, the strength of the random forest increases. Finally, the strength provides a measurement of the tree's accuracy.

In order to define the correlation ρ, it is necessary to introduce the raw margin function. Differently from the margin function in (1.21), the raw margin function is the difference between the number of votes for the right class and the number of votes for the most probable voted class other than Y $\hat{j}(\boldsymbol{X}, Y)$.

$$mrg(\Theta, \boldsymbol{X}, Y) = I(h(\boldsymbol{X}, \Theta) = Y) - I(h(\boldsymbol{X}, \Theta) = \hat{j}(\boldsymbol{X}, Y)) \tag{1.26}$$

By considering two independents Θ and Θ', once the correlation $\rho(\Theta, \Theta')$ is computed, Breiman (2001) provides for a mean value $\bar{\rho}$

$$\bar{\rho} = \frac{E_{\Theta,\Theta'}(\rho(\Theta, \Theta')\mathrm{sd}(\Theta)\mathrm{sd}(\Theta'))}{E_{\Theta,\Theta'}(\mathrm{sd}(\Theta)\mathrm{sd}(\Theta'))} \tag{1.27}$$

and, adjusting (1.23), for the upper bound for the generalization error PE^*

$$PE^* \leq \frac{\bar{\rho}(1 - s^2)}{s^2} \tag{1.28}$$

The result above permits to consider generalization error, strength, and correlation jointly. Moreover, such relationship will be useful in the analysis of random feaature selection.

1.3.1.3.2. Example: `iris`

Let's reconsider the `iris` example, expanding it to random forest. Several packages have been developed in R and are available to conduct such analysis and predictions; the mostly used are `randomForest` and `caret`. In this research's analysis, the `caret` package has been used. While the forest will be constructed, and an initial analysis and evaluation will be presented in this section, the tuning of its parameters to train the forest and the final testing will be displayed in sections 1.3.2 and 1.3.3 respectively (the full code is presented in Appendix B).

The data set is divided into a training set and a testing set. According to the size of the data set, different sizes of training and testing sets may be used. In general, it is suggested to adopt a ratio between 70%-30% and 90%-10% for training and testing respectively (Breiman, 1996; Breiman, 2001).

```
iris_RFModel <- randomForest(Species ~., data = trainingDataset)
```

The command[1] above permits to grow a forest using default parameters. The defaulted parameters computed as provided by the `randomForest` package generally provide a good degree of accuracy. In this study, in order to find the best accuracy, the main parameters have been tuned. The main parameters used to tune random forests are: mtry, ntree, maxnodes, and nodesize. These will be thoroughly explained in the following sections.
Figure 1.10 and Table 1.7 report the results obtained by training the model.

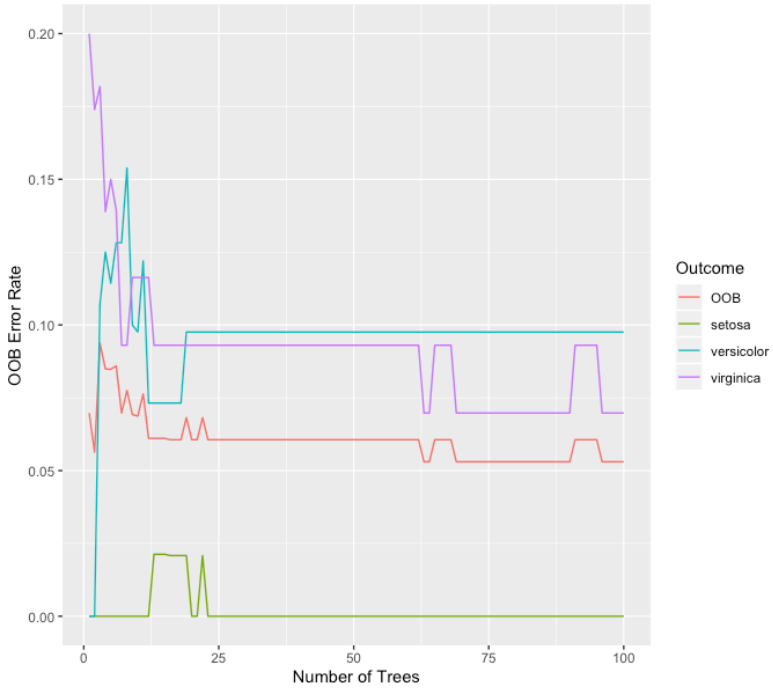
Table 1. 7: Confusion matrix iris and class error, `iris` example

	setosa	versicolor	virginica	Class Error
setosa	48	0	0	0.00000000
versicolor	0	37	4	0.09756098
virginica	0	3	40	0.06976744

Data Source: `iris` data set available in the R library "datasets"

[1] Specifics regardin the randomForest() function and its parameters can be found in the RDocumentataion, available at: https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest

Figure 1.10: Random Forest Error Plot, `iris` example



Data Source: `iris` data set available in the R library "datasets"

Figure 1.10 shows, along the classification errors for each class, the out-of-bag error estimate. This result is interesting since it confirms that as the number of trees increases the generalization error PE^* converges (see 1.23). As shown by the orange line in Figure 1.10, in fact, the out-of-bag error is higher between 1 and 25 trees, until it decreases and stabilizes from 25 trees onwards. Moreover, the reported average out-of-bag error is 5.3%. This result shows the high accuracy obtained by the random forest created, underlining the efficiency of using weak learners in order to create a strong classifier.

The confusion matrix in Table 1.7 shows the misclassified and the correctly classified cases. Specifically, it can be noted that the model is showing no difficulties in identifying the `setosa` cases. On the other hand, 7 misclassifications have been done regarding versicolor and virginica. While these results are promising, final evaluations should be conducted once the model is tested on the testing set.

1.3.2. Construction of random forests

1.3.2.1. Variables selection and splits

The way in which trees are grown and the randomization of features are key elements for the proper functioning of the random forest algorithm. Regarding the size of the dataset, random forests have proven good performance both with small and big data sets. Some debate, though, occurred in the literature regarding the use of datasets with p > n, with p predictor variable and n cases. Nevertheless, in general, with the proper number of cases, random forests are able to deal with many variables.

Additionally, through the `importance()` function in R, the algorithm provides the variables in order of importance. In the prediction analysis, this function permits to observe which variables account for most impurity reduction, and, therefore, are most reliable for the final prediction. While it is debated whether it is possible or not to obtain information regarding causal relationships between the predictor variables X and the response variable Y, the information provided by this function is useful for an initial understanding of the correlation between X and Y (Gregorutti and Saint-Pierre, 2013). An interesting paper by Hastie and Zhao (2019), on the other hand, suggests relevant information regarding causal relationship can be obtained by observing the partial dependence plot (PDP) and the Individual Conditional Expectation (ICE) plot. This possibility will be analyzed in section 1.3.3.

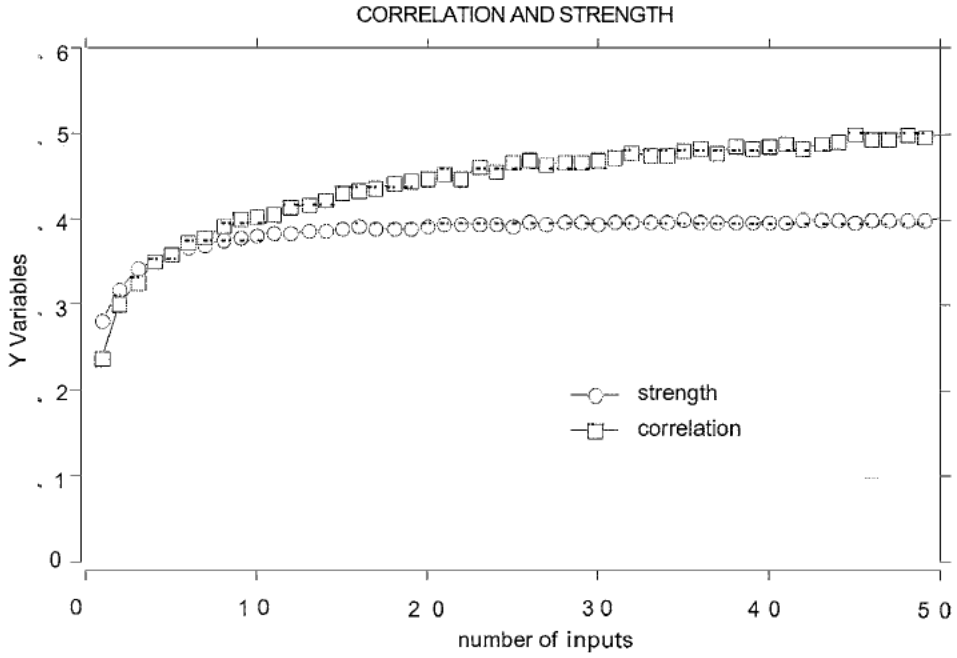
In order to obtain trees that are independent, Breiman (2001) adopted a random feature selection approach to split the decision trees. Simply, given m predictor variables, the algorithm selects randomly k variables, and finds the best split among the k variables by computing the Gini Index. Practically, in R, such parameter is called mtry. By default, matry is set equal to $\sqrt{m}$ for classification analysis and m/3 for regression analysis. Nevertheless, through the `caret` package the parameter can be tuned.

```
trControl <- trainControl(method = "cv",
                          number = 10,
                          search = "grid")

set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(1:4)) # select range
_mtry <- train(Species ~.,
               data = trainingDataset,
               method = "rf",
               metric = "Accuracy",
               tuneGrid = tuneGrid,
               trControl = trControl,
               importance = TRUE,
               nodesize = 1,
               ntree = 300)

print(Sample1_RF_mtry)
```

The function `trControl()` permits to choose the method to use between bootstrapping, cross validation, among others. The `tuneGrid()` function permits to create custom grids in order to test the model on different values for the parameter to be tested. Therefore, by changing the range of values required (considering the default value is $\sqrt{m}$, in order to save computing time, the range should be chosen according to m), R outputs all models generated through the `tuneGrid()` function. By visualizing such output, the mtry value that provides the best accuracy can be identified.

Recalling (1.28) and relationship between generalization error, strength and correlation, Breimam (2001) provides the reason for which mtry should be tuned. For simplicity consider the result obtained by Breiman in Figure 1.11. Specifically, it can be observed that as the number of variables chosen for the split increases, the random forest's strength remains approximately constant, while the correlation increases, showing that an optimal mtry exists, given that as mtry increases the generalization error increases (Breiman, 2001).

Figure 1.11: Correlation and strength



Source: Breiman (2001, p.17)

Importantly, Breiman's (2001) results have been confirmed in a series of experiments on small and large datasets by Papacharalampous and Tyralis' (2017), in which the variable selection question is directly assesed, and by Adam et al. (2010) which focuses specifically on the concept of strength and correlation and finds that (1.28) holds for both small and big datasets, and with both a small and big number of trees.

1.3.2.2. Trees' depth

As described in section 1.2, the parameters which permit to tune a tree's depth are maxnodes and nodesize. Respectively, they represent the maximum number of terminal nodes a tree should have and the minimum size of terminal nodes. While decision trees require pruning to avoid overfitting, random forests are strong learners which do not overfit the data. Breiman (2001), in fact, writes that trees in random forests should be fully grown. While this remains a possibility which still leads to good performances, these parameters can be tuned in order to select the best

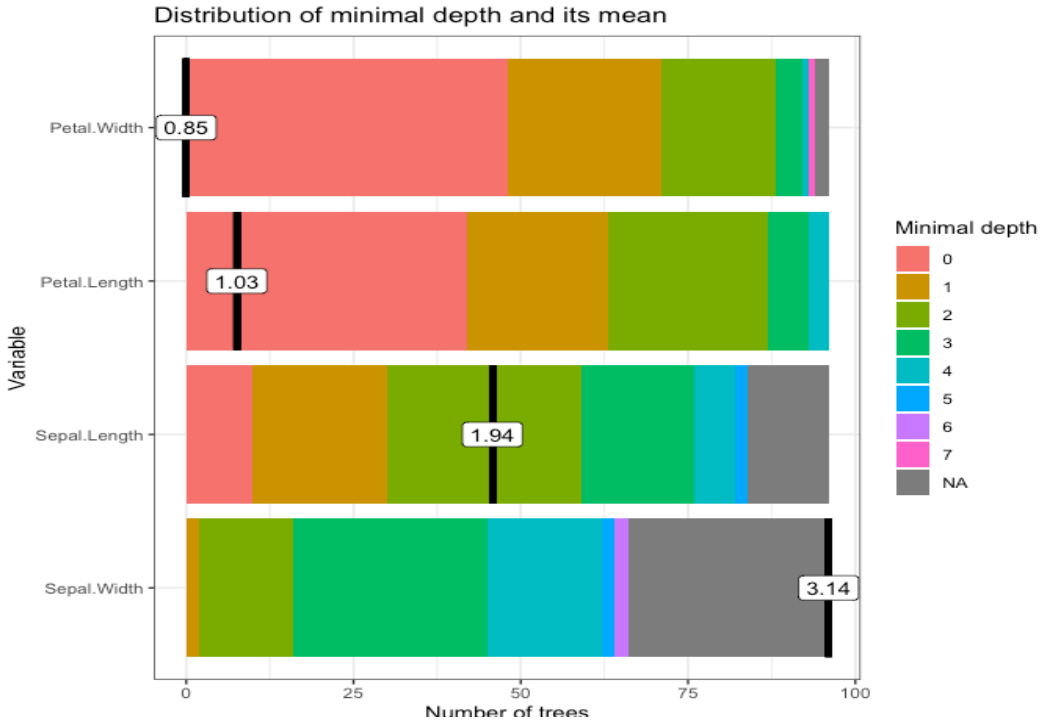
accuracy. Given random forest's ability to deal with overfitting, the nodesize parameter can be set at the default value 1. In order to tune maxnodes, though, the code below can be used.

```
store_maxnode <- list()
tuneGrid <- expand.grid(.mtry = 9) # the result obtained previously must be used
for (maxnodes in c(1:10)) { # change range
  set.seed(1234)
  RF_maxnode <- train(Species ~.,
                      data = trainingDataset,
                      method = "rf",
                      metric = "Accuracy",
                      tuneGrid = tuneGrid,
                      trControl = trControl,
                      importance = TRUE,
                      nodesize = 1,
                      maxnodes = maxnodes,
                      ntree = 300)
  key <- toString(maxnodes)
  store_maxnode[[key]] <- RF_maxnode
}
results_node <- resamples(store_maxnode)
summary(results_node)
```

Similar to mtry, the output generated through the code above permits to visualize and select the number of nodes that provides the highest accuracy.

An interesting insight to the random forest is shown in the minimal depth graph (Figure 1.12). Such graph permits to observe which variables were used to split the data closest to the root node. In the `iris` example, Figure 1.12 shows that the variables which were mostly used in the first two splits are `Petal.Width` and `Petal.Lenght`. This information is linked to the results of the variable importance analysis which will be presented in the final section. While `Sepal.Length` was used on average on the third split, Figure 1.12 shows that `Sepal.Width` was used very few times to split the data, and, on average, for the last split.

Figure 1.12: Tree depth, `iris` example



Data Source: `iris` data set available in the R library "datasets"

1.3.2.3. Number of trees

Breiman's (2001) research provides important findings regarding the number of trees issue. Expanding on section 1.2.2.4, an hypothesis may be that an increase in the number of trees could result in a model which is overly dependent on a specific dataset. As shown in Breiman (2001), random forests generally do not overfit when more trees are added. In fact, as the number of trees grown tends to infinite, the generalization error for a random forest is limited by an upper bound. Therefore, the number of trees used does not have a major impact on the accuracy of the model, as long as a sufficient large value is provided. Some studies have tried to suggest appropriate values for the `ntree` parameter: Oshiro et. al. (cited in Papacharalampous and Tyralis, 2017) identified the required number of trees in the range [64, 128], while Kuhn and Johnson (cited in Papacharalampous and Tyralis, 2017), instead, suggests using at least 1000 trees.

Having obtained a value for `mtry` and `maxnodes`, the code for tuning `ntree` is provided below.

```
tuneGrid <- expand.grid(.mtry = 2)
store_maxtrees <- list()
for (ntree in c(500, 1000, 1500, 2000)) {
  set.seed(1234)
  RF__ntrees <- train(Species ~.,
                      data = trainingDataset,
                      method = "rf",
                      metric = "Accuracy",
                      tuneGrid = tuneGrid,
                      trControl = trControl,
                      importance = TRUE,
                      nodesize = 1,
                      maxnodes = 4,
                      ntree = ntree)
  key <- toString(ntree)
  store_maxtrees[[key]] <- RF__ntrees
}
results_tree <- resamples(store_maxtrees)
summary(results_tree)
```

1.3.3. Performance and evaluation of random forests

Several possibilities are available to evaluate the performance and structure of random forests. While from a theoretical point of view, Breiman (2001) focuses on the generalization error and on the relationship between strength and correlation with the number of variables used, from a practical point of view testing the model on unexplored data and using the confusion matrix to analyze correct and wrongful predictions remain the most used approaches. While Breiman's (2001) theoretical view has been already presented previously, the following sections will focus on testing and on analyzing the relationship between predictor variables and the dependent variable.

1.3.3.1. Testing

Relying solely on the results based on the training set is not appropriate. With decision trees, in fact, it is possible to obtain a model that seems to perform well, but that when applied to unseen data shows its limitations (e.g. overfitting). The same is not true for random forests, given that they

do not overfit and they allow to find a trade-off between bias and variance. Nevertheless, providing a model through unseen data is the best possibility for testing it.

```
tuneGrid <- expand.grid(.mtry = 2) # insert best mtry
FinalRF <- train(Species ~.,
                 data = trainingDataset,
                 method = "rf",
                 metric = "Accuracy",
                 tuneGrid = tuneGrid,
                 trControl = trControl,
                 importance = T,
                 nodesize = 1, # insert best nodesize
                 ntree = 500, # insert best ntree
                 maxnodes = 4) # insert best maxnodes

# Prediction on new data (testing dataset)
irisPrediction <- predict(FinalRF, newdata = testingDataset, type = "prob")
irisConfusionMatrix <- confusionMatrix(irisPrediction, testingDataset$Species)
irisConfusionMatrix # confront prediction with actual values
```

The code presented above constructs the final random forest and tests it on unseen data provided by the testing data set that has been extracted from the data set initially. Results are provided in Table 1.8 and 1.9. It should be noted that by using 90% of the data for training, very little data is left for testing (17 cases). While studies have shown that the random forest algorithm performs well with small datasets as well, this suggested to test the model to change the proportion of training/testing data.

Table 1.8: Final testing confusion matrix, iris example, `iris` example

	setosa	versicolor	virginica
setosa	3	0	0
versicolor	0	5	1
virginica	0	1	7

Data Source: `iris` data set available in the R library "datasets"

Table 1.9: Main statistics, iris example

Accuracy	88.24%
No Information Rate	36.36%
p-value (Acc > NIR)	$6.292e^{-14}$

Data Source: `iris` data set available in the R library "datasets"

The no information rate, as defined by the description of the `confusionMatrix()` function of the `caret` package in R, represents the largest class percentage in the data. Intuitively, a model with accuracy below the no information rate should not be considered. Overall 88.24% accuracy indicates the model's good performance.

Throughout this chapter, for simplicity and continuity, the iris example has been carried. The purpose of this example was to provide a practical view, sustaining the theoretical background analysed. Such a small dataset, though, could bring limitations. The latter are not related to the fact that random forests are not an appropriate method for small datasets; Breiman (2001), Papacharalampous and Tyralis (2017), and other studies have shown this is not the case. Rather, it raises the issue of deciding the appropriate size of the training and testing set, specifically with small data sets. To prove this, the code has been run with 70% of the dataset being used for training, and 30% for testing. The results confirm such theory, since the accuracy increased to 93.18%.

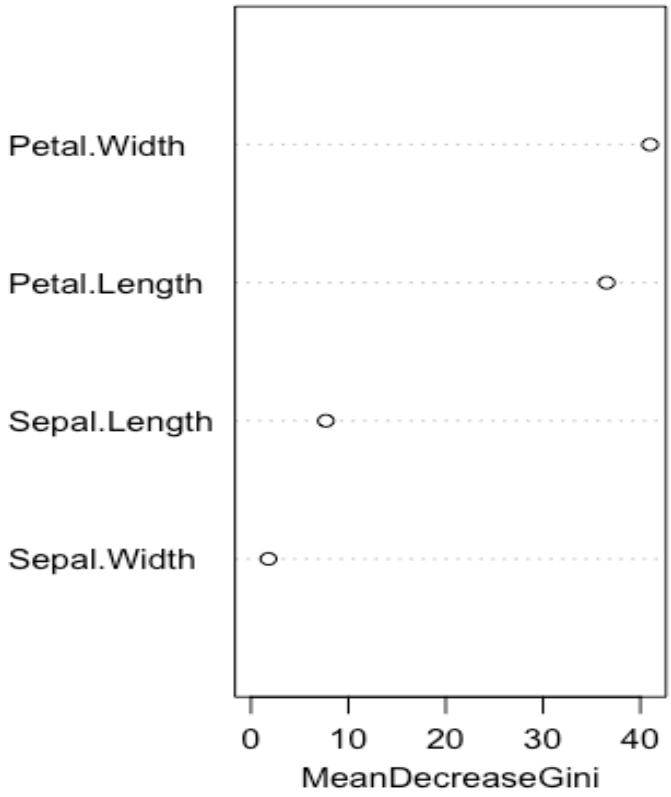
1.3.3.2. Variable importance and predictor's impact on the model

One of the main criticisms to machine learning is that it generally behaves like a "black box", meaning it is not easy to comprehend the underlying functioning of the model (Hastie and Zhao, 2019). In other words, given a trained model, given a new input I(**x**), the model provides a prediction Y based on a predictive function g(**x**), but it is not simple to extract information regarding the relationship between I(**x**) and Y. This argument has often been used to state that it is not generally possible to identify causal relationship between predictor variables and the dependent variable through machine learning algorithms. While the purpose of this study and the depth of the topic do not permit to fully address such analysis, an introduction will be provided by drawing the partial dependence plots, introduced in Friedman (2001), and the individual conditional expectation. Further researching this topic could be interesting; by analyzing the methods which

test causal relationship in machine learning models, a possible research question may be "Can causality be inferred through random forests?".

Firstly, an important tool available in R permits to start this analysis by identifying the most important variables for the classification of the dependent variable. Figure 1.13 depicts the results obtained for the `iris` example. The importance of a predictor is measured as its ability to reduce the tree impurity, and, in random forests, is measured through the Gini Index. Therefore, Figure 1.13 shows the mean decrease of the Gini Index for each predictor (i.e. a greater decrease in the Gini Index indicates the high importance of the predictor.

Figure 1.13: Variable Importance, iris example



Data Source: `iris` data set available in the R library "datasets"

While `Petal.Width` and `Petal.Length` account for most of the decrease of the nodes' impurity, `Sepal.Length` and `Sepal.Width` do not account for much importance. Figure 1.13 confirms that the results provided by the variable importance plot are in line with those of the distribution of

minimal depth plot (Figure 1.12). While Figure 1.13 permits to observe some degree of correlation between the predictor variables and `Species`, it is not possible with this information to infer causality and to view how the single classes `setosa`, `versicolor`, and `virginica` are affected by the variables. The importance of variable, though, permits to begin to draw conclusions regarding the fact that some variables may not be considered in the construction of the random forest given their little influence on the predicted variable. In the `iris` example, for instance, Figure 1.13 suggests that the variable `Sepal.Width` could be excluded from the set of predictors.

An additional approach to evaluating the removal of variables is provided by the partial dependence plots (hereafter PDP) and the individual conditional expectation (hereafter ICE) plots, which show how much the predicted variable is dependent from the predictor variables. Moreover, by plotting PDP and ICE plot as logit contribution of the variables to the model, it's possible to deduct the likeliness of positive influence of a specific predictor variable to the predictive model. Following a brief introduction to the PDP, the latter will be used to conduct two tests. First, the PDP will be used to observe the relationship between the predictor variables and the classes within the dependent variable. Then, the PDP and the ICE plots will be used to observe the relationship between the predictive model and the predictor variables.

Given the previously defined output g(**x**) = Y, *"the partial dependence of g on a subset of variables X_s is defined as (let X_C be all the subsets of variables minus X_s)*

$$g_S(x_S) = E_{x_C}[g(x_S, X_C)] = \int g(x_S, x_C) dP(x_C) \tag{1.29}$$

[meaning] *g_S is the expectation of g over the marginal distribution of all variables other than X_S."* (Hastie and Zhao, 2019, p. 2-3). Finally, (1.29) provides a prediction of Y based on X_s.

For the purpose of this study, it's sufficient to understand that PDP show the extent to which a model partially depends on a predictor variable. For classification analysis, the PDP shows the average logit contribution of **x** on the class k. In the `iris` example this permits to answer questions such as "How does the prediction for `setosa` change if `Petal.Lenght` is increased?" and so on.

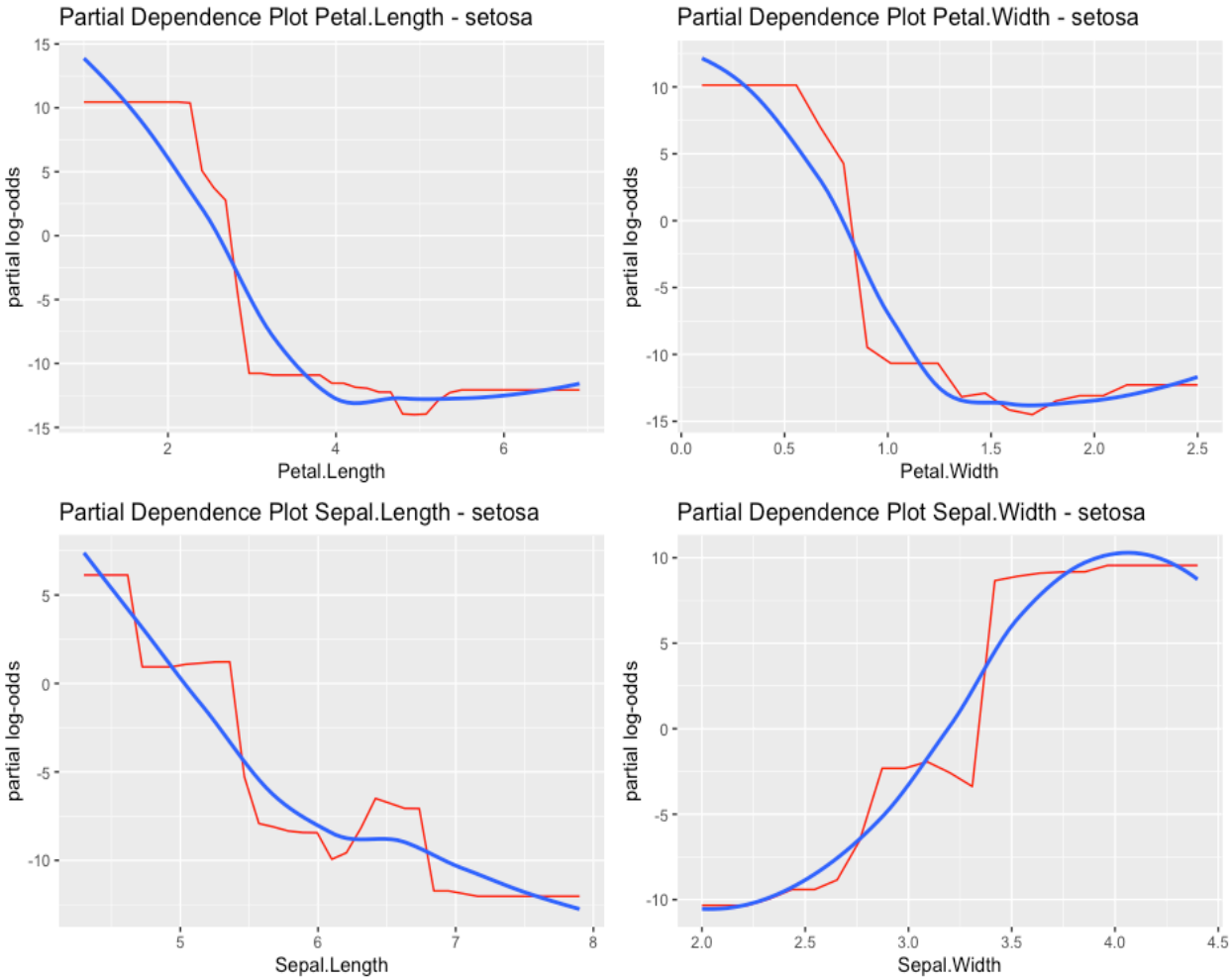
Given our classification example, the PDP for each class have been graphed and are shown in Figures 1.14, 1.15, and 1.16, in relation to the four predictors `Petal.Width`, `Petal.Length`, `Sepal.Length` and `Sepal.Width`. The blue curve has been added to show a smoother version of

the line to simplify interpretation. The following list provides the key for interpreting Figures 1.14, 1.15, and 1.16 by observing the log odds:

- log odds > 0 means the class considered is more likely to be predicted correctly;
- log odds < 0 means the class considered is less likely to be predicted correctly;
- log odds = 0 indicates that the class prediction is not impacted, on average.

Moreover, values of x which provide positive log odds influence positively the classification process.

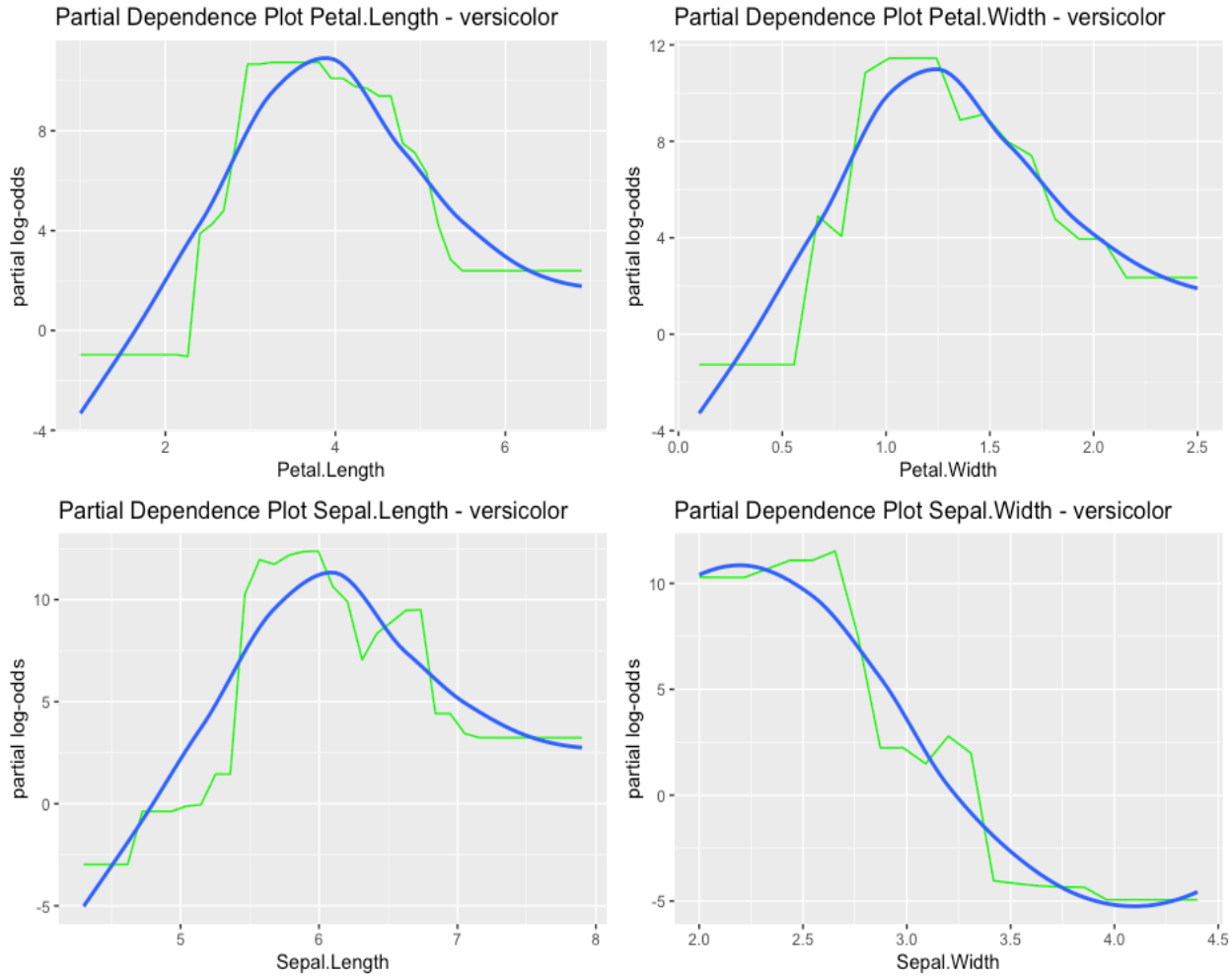
Figure 1.14: Dependence plots k = `setosa`, iris example



Data Source: `iris` data set available in the R library "datasets"

Figure 1.14 provides some useful insights. Comparing Figures 1.14 and 1.16, it is shown that the influence that the predictor variables have on the predicted class has opposite trends in the case of `setosa` and `virginica`; `versicolor` behaves differently. Moreover, the PDP for `Sepal.Lenght` shows that it is the variable which has less influence in classifying `setosa`.

Figure 1.15: Dependence plots k = `versicolor`, `iris` example



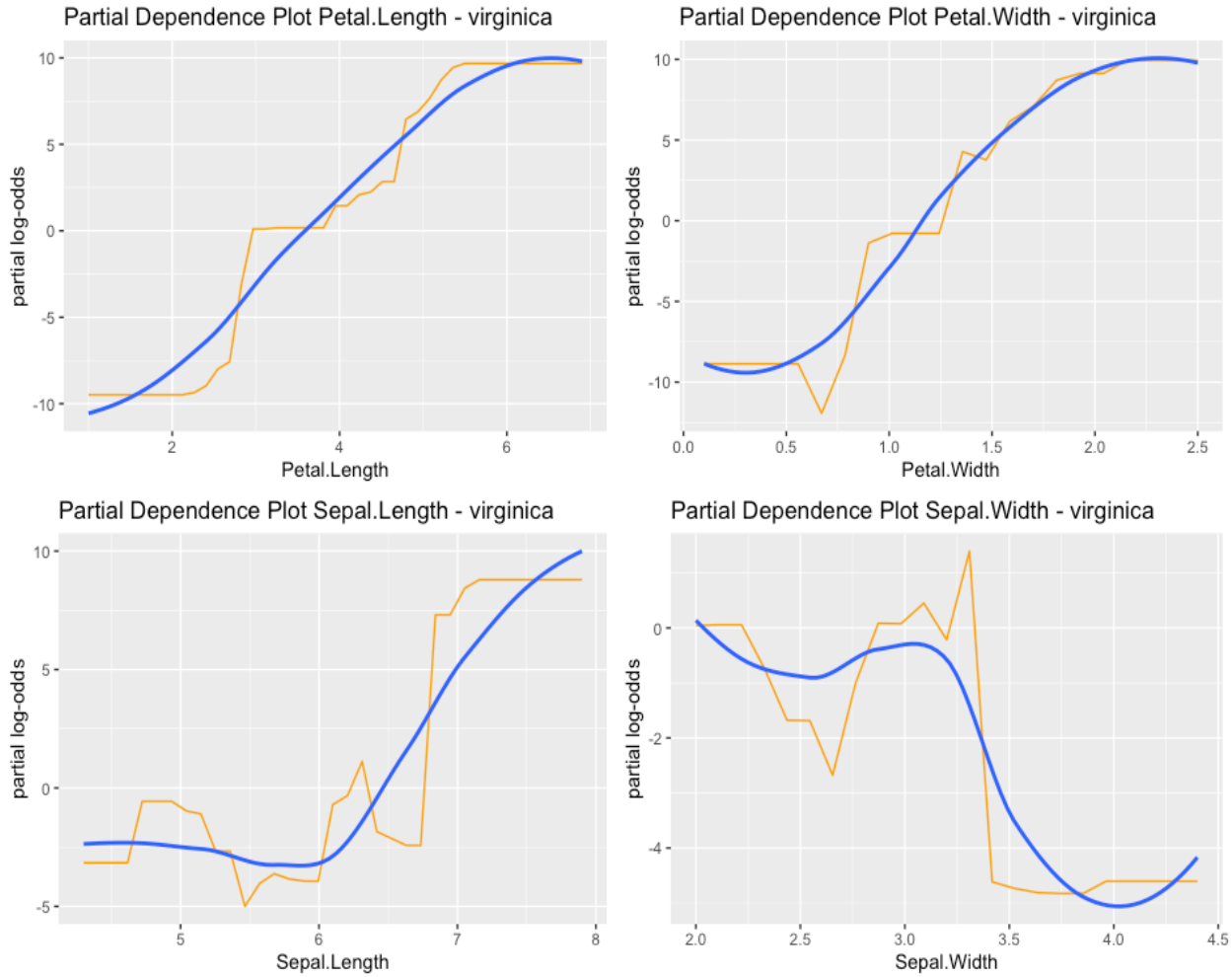
Data Source: `iris` data set available in the R library "datasets"

The relationship between the `versicolor` class and the variables `Petal.Length`, `Petal.Width`, and `Sepal.Length` seems constant. For `Petal.Length` values above ~1.5, for `Petal.Width` above ~ 0.3, for `Sepal.Length` above ~ 4.75, values begin to have a positive influence for classification.

This influence, then, increases until a specific point before showing a positive but decreasing influence on the `versicolor` prediction.

The PDP for `Sepal.Width`, instead, indicated a continuously decreasing influence on the class prediction.

Figure 1.16: Dependence plots k = `virginica`, `iris` example



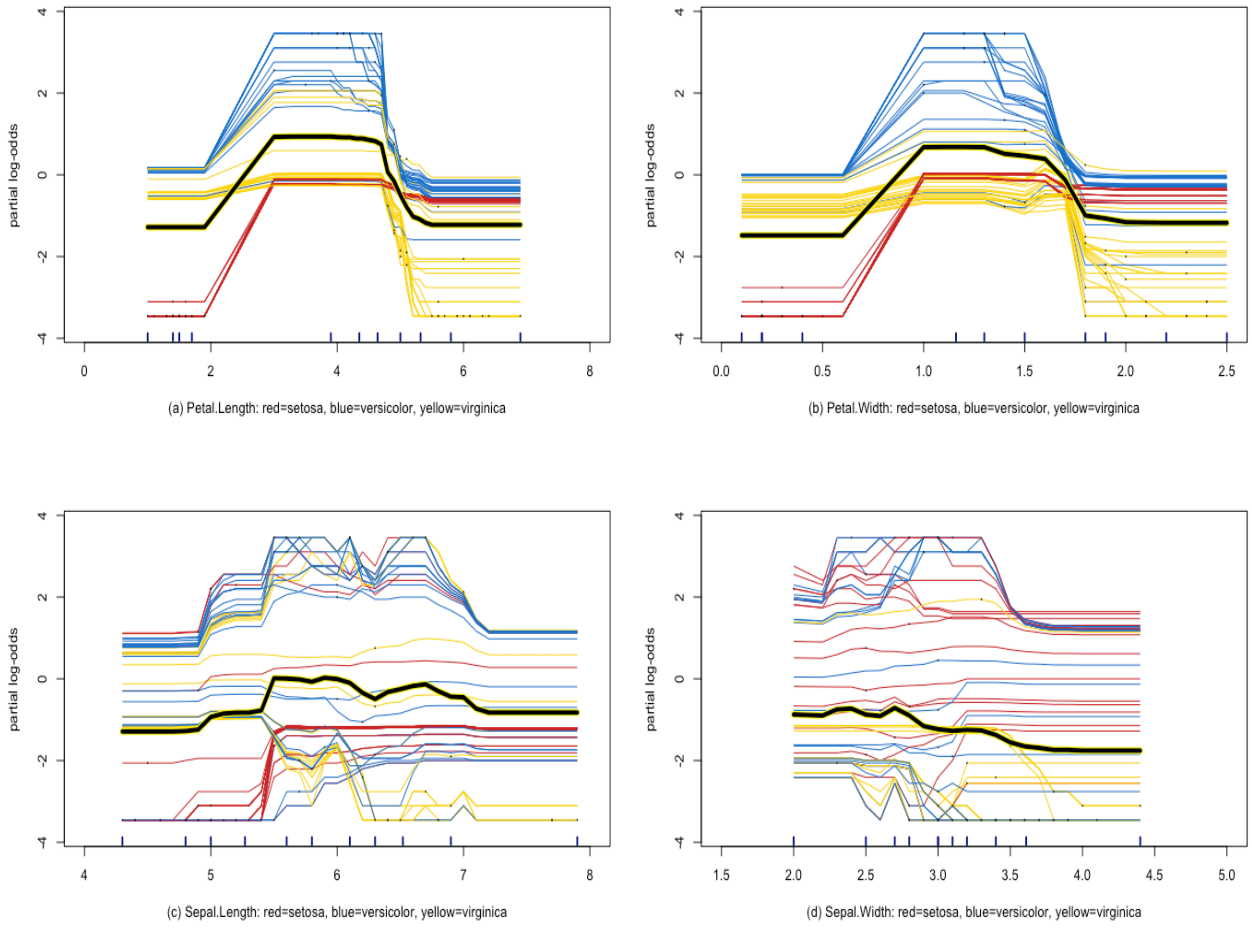
Data Source: `iris` data set available in the R library "datasets"

The PDP for the class `virginica` provides interesting results. Specifically, `Sepal.Width` does not have any influence in the classification process. Moreover, considering `Petal.Length` and `Petal.Width`, a strong positive relationship is shown in Figure1.16.

While the limitations of using PDP should be further analyzed to draw conclusions, Figures 1.14, 1.15, and 1.16 permit to extend the objective of obtaining a model with high accuracy. In fact, they provide useful insights regarding the influence that the predictor variables have on each class, permitting to understand the underlying reason for which a specific case is classified, for example, as `virginica` and not as `setosa`.

While the PDP shows the average influence of a variable on the model, the ICE plot permits to visualize the single influence of each case.

Figure 1.17: Individual conditional expectations, `iris` example



Data Source: `iris` data set available in the R library "datasets"

Differently from the previous individual analysis on the classes in `Species`, in this case the influence on the model overall is observed, and the units are colored in Figure 1.17 according to the class. The PDP in Figure 1.17 is provided by the highlighted black line.

Figure 1.17 provides useful results in line with those of variable importance of Figure 1.13. While the messiness of the ICE curves for `Sepal.Length` and `Sepal.Width` suggest that it is difficult to identify relationships with the three classes in Species, the PDP in black shows the little relevance these two variables have in relation to the random forest constructed. In fact, while `Sepal.Length` may have a slightly positive influence on the model approximately between [5.5;6], `Sepal.Width` never has a positive influence on the random forest. This may suggest that `Sepal.Width` could be removed from the predictor variables. A new random forest has therefore been constructed in order considering as predictors only `Petal.Length`, `Petal.Width` and `Sepal.Length`. This forest performed favourably (95.56% accuracy) compared to the results shown in the previous section (93.18% accuracy).

1.4. CONCLUSIONS

The purpose of this chapter is to provide the knowledge and understanding of decision trees and random forest in order to prepare the basis for the comprehension of the analysis conducted in Chapter 3. Such chapter could be widely extended to other machine learning algorithms in order to compare the performances of various algorithms.

Both decision trees and random forests perform well. Nevertheless, the fact that random forests do not overfit provides a great advantage compared to decision trees. Through random feature selection and bagging, it has been shown that random forests are a powerful tool for creating an accurate and stable model. Moreover, an attempt to analyze the underlying functioning of the random forest algorithm has been done by observing each predictor variable's influence on the classification model through partial dependence plots and individual conditional expectation plots. These provide a useful tool since they permit to extend the conclusions drawn from variable importance measurements.

Throughout the chapter an analysis on the `iris` data set has been carried. This allowed to confront decision trees and random forest. Moreover, its simplicity permitted to implement the

methods explained and, most importantly, to focus on understanding the algorithms used. Further analysis and clarifications will be provided in Chapter 3. The latter will implement the knowledge and methods used in this chapter to conduct a binary classification analysis in order to predict a company's probability of default. The `iris` example will be introduced again following the main analysis of Chapter 3, where the code providing the function to repeat the analysis is provided.

CHAPTER II
COMPANY FAILURE AND PROBABILITY OF DEFAULT

In the literature numerous studies and empirical researches (Altman, 1968; Balcaen and Ooghe, 2006; Beaver, 1967; Ciampi and Gordini, 2008; Edmister, 1972; Hazak and Mannasoo, 2007; Kealhofer, 2003; Kliestik et al., 2014) address credit risk and firm default. The determination of correlation and causation between financial statement figures and default has triggered the development of methodologies and models with the objective of predicting company's failure and quantifying the probability of default. Regardless of the complexity of the argument, many advancements have been achieved. The latter have been led by the importance of identifying relationships between company's credit risk and probability of default, and the various internal and external factors. The value that relies in such discoveries is linked, among others, to businesses, markets, and to public policy.

The following paragraphs will provide a literature summary and review, aiming to lay out the topic and the main characteristics of the models developed. Furthermore, an attempt to finding connections and possible complementarities between such models and random forests, which this paper will focus on, is presented.

2.1. MODELS FOR QUANTIFICATION OF PROBABILITY OF DEFAULT

2.1.1. Introduction and definitions

Bankruptcies have a great impact on modern societies (Hazak and Mannasoo, 2007). In fact, besides the cost and losses faced by shareholders, the network generated through relationships with stakeholders extends the repercussions at a macroeconomic level. Such enlargement has increased the popularity of both researches on the impact of business' bankruptcy on societies, and on the prediction of company failures. Reasonably, the importance of the two fields of study is interconnectedly strong.

Specifically, regarding the methods developed to identify and quantify the probability of default and company failure, several are available. Within such group, two approaches can be

noted. While some models mainly use the firms' financial information, accounting ratios, and company structural figures, for example the Z-score model (Altman, 1968), the univariate and multivariate discriminant analysis (Beaver, 1967), the logit regressions (cited in Ciampi and Gordini, 2013), other methods have a more finance and market based approach, such as Merton's theory of the risk structure of interest rate (1974) and the KMV model (cited in Kealhofer, 2003). See Balcaen and Ooghe (2004; 2006) for a detailed summary of the classical and alternative methods developed on the topic.

For the purpose of this paper and for simplicity, bankruptcy, business and corporate failure, liquidation, and default, are assumed to describe the same situations: a company who, by any reason, is forced to terminate its operation and its lifecycle. Therefore, the methods mentioned above, are grouped together since they all have the same objective of predicting, classifying, and finding correlation or causal relationship that describe company's default. This may entail some limitations. While the methods do aim to predict company's default, the assumptions and situations underlying some of the methods' application may not permit comparison across the two groups.

2.1.2. Models predicting company's failure and probability of default

The complexity of the environments and societies have, in turn, increased the difficulty of analyzing business. Such complicatedness is related to the fact that much more interconnected networks make businesses, among others, highly dependable on a wide variety of factors. Burgartz et al. (2016) discuss this by describing today's operations as influenced by volatility, uncertainty, complexity, and ambiguity. Extrapolating from this information, the ability to be able to deal with numerous situations simultaneously required to managers is fundamental for the survival and success of a business. Moreover, the latter's ability to adapt and take advantage of new possibilities has become the mean to obtain competitive advantages. Finally, this underlines the complexity of creating static models which consider limited influences and poor variables. As will be discussed further on in the chapter, this complicates the prediction process of a company's default, making it impossible to know with certainty, based solely on the companies figures, whether a company will default or not. Moreover, given that not all businesses operate in the same environment, and therefore are not influenced by the various factors to the same extent, the possibility to create an adaptable method, applicable to all business, is demanding to achieve. Consider, for example, that

some industries require specific costly investments, which are possible by creating debt. Such firms are affected in different ways and by different factors (e.g. financial crisis) than enterprises which do not incur in such situations. Examples of this type are abundant and diverse. This highlights a limitation of researches in such topic. Nevertheless, the numerous studies that have been conducted have tried to identify which variables should be considered in the analysis of this topic. Such studies, as will be presented subsequently, contribute in identifying a framework for predicting credit risk. The study provided in the Chapter 3 will try to integrate such framework in order to consider all most relevant variables. By consistently improving such framework, the possibility of creating accurate models and predictions has increased over time (Breiman, 2001; Papacharalampous and Tyralis, 2017).

2.1.2.1. Univariate and multivariate discriminant analysis

Methodologies such as univariate and multivariate analysis, in this field of study, use financial ratios to predict corporate failure, assuming that such ratios are appropriate to express a firms' health. While such methods are advantageous for their simplicity, they often are too subjective (Balcaen and Ooghe, 2006). For specifics regarding such methods, refer to Balcaen and Ooghe (2006), Altman (1968), and Beaver (1967). Specifically, Beaver (1967) used cash-flow ratios, net-income ratios, debt to total-asset ratios, liquid-asset to total-asset ratios, liquid-asset to current debt ratios, and turnover ratios. Beaver (1967) underlines both the limitations and the strengths of such approach. While he conducted an analysis on the ratio's prediction power individually (therefore univariate analysis), he suggests that a multi-ratio analysis may be more useful and provide better results. Moreover, he clarifies that given the different nature of financial ratios and, especially, the different ways (if any) in which companies use such ratios, the results are affected by external factors which are complex to quantify. Moreover, analysis conducted on financial ratios often do not consider other external relevant factors (Hazak and Mannasoo, 2007). Nevertheless, the results obtained are promising.

Beaver (1967) has contributed significantly in developing the univariate discriminant analysis model and has shown that financial ratios may have a predictive power up to five years before the failure event. His model, in fact, attempted to make predictions over five years. Table 2.1

summarizes some of Beaver (1967) analysis' results, showing the percentage of misclassified defaulted companies. Moreover, regardless of the fact that they are considered individually, Table 2.1 still permits to identify which ratios are most effective in predicting company failure, and which are consistent through time. Other studies have studied the relationship between financial ratios and company default and found similar results (Altman, 1968; Ciampi and Gordini, 2008; Edmister, 1972).

The functioning of the univariate analysis mainly revolves around its classification procedure. Specifically, the firms considered in the study are classified as failing or non-failing for every ratio individually, by finding a level of ratio (i.e. decision point) which minimizes the classification error (Beaver, 1967).

Table 2.1: Findings of Univariate Analysis[2]: % error

Ratio	Years before Failure				
	1	2	3	4	5
$\frac{Cash\ Flow}{Total\ Assets}$	.10 (.10)	.20 (.17)	.24 (.20)	.28 (.26)	.28 (.25)
$\frac{Cash\ Flow}{Total\ Debt}$	.13 (.10)	.21 (.18)	.23 (.21)	.24 (.24)	.22 (.22)
$\frac{Net\ Income}{Sales}$	.13 (.09)	.22 (.16)	.28 (.24)	.35 (.28)	.31 (.27)
$\frac{Net\ Income}{Total\ Assets}$	.13 (.12)	.21 (.15)	.23 (.22)	.29 (.28)	.28 (.25)
$\frac{Current + Long\ Term\ Liabilities}{Total\ Assets}$	.23 (.19)	.30 (.26)	.36 (.29)	.39 (.31)	.38 (.33)

Source: Beaver (1967), see p.106-107 for full results

One of the main disadvantages of such model is the fact that it only considers the ratios individually (Balcaen and Ooghe, 2006). In other words, it cannot make an appropriate prediction since it is impossible to find a scenario in which a company's situation can be described with one single ratio. Nevertheless, it behaves well as a guidance to understand which ratios (i.e. which figures

[2] Beaver's (1967) analysis consisted in two test runs. The results of the first test are presented in the second row in brackets, while the second test's results are provided by the first row.

from the balance sheet) have a greater impact. Such aspect, though, is limited due to the correlation among the variables composing the ratios; in fact, no causation relationship can be derived.

Realizing that the disadvantages of the univariate method lays in the use of the ratios individually, other methods appeared to solve such inefficiencies. While the Risk index model does so by providing each ratio with a score, therefore providing a final evaluation of a firm's score including all ratios, the multiple discriminant analysis has been the most successful in solving such issue.

The multiple discriminant analysis has been introduced by Altman (1968), with the development of the Z-score model. Later, such procedures have been further researched and has been accustomed as the standard method for identifying company default (Balcaen and Ooghe, 2006). This method consists in finding a linear relationship which provides a score $Z \in [-\infty; +\infty]$ to the firm. Such score represents a firm's health (Altman, 1968). Such discriminant function is provided by (2.1) derived by Altman (1968, p.592, p.594).

$$Z = v_1 x_1 + v_2 x_2 + \cdots + v_n x_n \tag{2.1}$$

where $x_1, \ldots, x_n$ are the independent variables and $v_1, \ldots, v_n$ are the discriminant coefficients. (2.1) permits to visualize the overcome the main limitations of the univariate analysis previously discussed, in which the variables are observed individually. Moreover, an optimal discriminant function (2.2) has been obtained in Altman (1968).

$$Z = .012X_1 + .014X_2 + .033X_3 + .006X_4 + .999X_5 \tag{2.2}$$

where X_1 = Working Capital/Total Assets

X_2 = Retained Earnings/Total Assets

X_3 = Earnings Before Interest and Taxes/Total Assets

X_4 = Market Value Equity/Book Value of Total Debt

X_5 = Sales/Total Assets

Finally, Altman's (1968) Z-score model has proven great accuracy, correctly classifying 95% of the firms analyzed. Nevertheless, Balcaen and Ooghe (2006) highlight its disadvantages, such as

that the linearity of the discriminant function is in contrast with the fact that not all variables are linear with respect to a company's financial health, and that although the method resembles a regression analysis, it differs in its computations and it does not permit to assess the relative importance of the variables. It should also be noted that merely using accounting increases the risk and uncertainty of the study: regulations may differ across countries, a certain level of subjectivity, and the fact that ratios often differ according to industry may lead such measurements to impact the final results. This issue can be contained, to some extent, by creating a sample of firms which belong to the same industry in countries which adhere to the same accounting regulation. Nevertheless, this leads to poor adaptability of the model, requiring the creation of a new study for every industry analyzed.

2.1.2.2. Survival Analysis Model

While Hazak and Mannasoo's (2007) paper analyzes the predictability of financial ratios, classifying them as *"Company financial indicators"*, it also expands the research to *"Company structural indicators"* and *"Macroeconomic indicators"*. For the purpose of this research, the survival analysis model will not be presented, since its relevance is provided by the variables it uses. In fact, such model goes beyond financial ratios, underlying the importance of other figures. Table 2.2 briefly summarizes such variables.

Table 2.2: Variables to Include in an Analysis

Variable	Description
EMPL	Employees
TYPE	Binary variable: A for large businesses and B for small ones.
QUOTE	Binary variable: 1 for quoted, 0 for non-quoted
GDP GROWTH	Real GDP growth
LEND RATE	Real domestic lending rate
CREDIT GDP	Private credit share of GDP
EXP IMP GDP	Exports + Imports (as % of GDP)
REER	Real effective exchange rate

Source: Hazak and Mannasoo, 2007, p.9-10

2.1.2.3. Black-Scholes, Merton, and the KVM Model

The Black-Scholes-Merton and the KVM model take a different approach to quantifying credit risk, and such theories have gained a lot of importance in modern finance. While the aim remains to quantify the probability of default, these methods are mostly used in the financial analysis sector. For instance, one of their application relates to bank's loan approval (Bandyopadhyay, 2007). In other words, they aim to improve the capacity of making good investments and obtaining the knowledge necessary to hedge and protect a position in case adversities. In addition to the previous definition of bankruptcy provided, in this section the explanation for bankruptcy mainly relies in fall of asset value and/or liquidity shortages (Bandyopadhyay, 2007). Therefore, in the calculation of the probability of default, three main elements are considered: value of assets, uncertainty or risk of the asset value, and leverage, meaning the extent of a firm's liabilities (Bandyopadhyay, 2007; Kealhofer, 2003). The methods analyzed in this section differ from the previous ones because the basis for their computations is provided by valuing debt and equity as call options. Such procedure has been developed by Black and Scholes in 1973. Building on these findings, the Black-Scholes-Merton developed a model which led to winning the Noble Prize for Economics in 1997[3]. Elaborating on their model, the KVM model has been developed.

The underlying importance of Black and Scholes' results is given by the fact that the equity of a company can be viewed as a call option on the company's underlying assets (Kealhofer, 2003). Consider the following example, extracted and modified from Kealhofer (2003) for simplicity al illustrated by Figure 2.1.

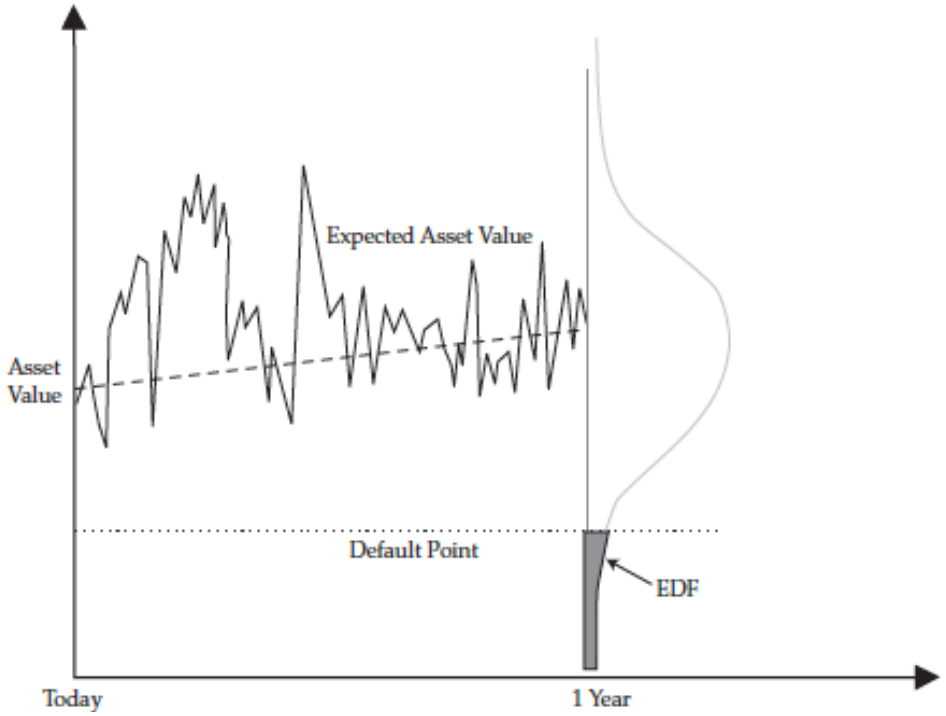
> *"Suppose a company has a single asset consisting of 1 million shares of Microsoft stock. Furthermore, it has a single fixed liability—a one-year discount note with a par amount of $100 million—and is otherwise funded by equity. In one year's time, the market value of the company's business will either be sufficient to pay off the note or it will not, in which case the company will default. One can observe that the equity of*

[3] The Noble Prize has been awarded to Robert Merton and to Myron Scholes. Fisher Black died in 1995, therefore not being eligible for the prize.

the company is logically equivalent to 1 million call options on Microsoft stock, each with an exercise price of $100 and a maturity of one year." (p.30)

As shown in Figure 2.1, in 1 year there is a wide range of values which the asset can obtain. Simply, if such future value will be lower than the liability, the firm will fail. As the Gauss curve provides, there is a greater possibility that the asset's future value will be close to the starting value, but extreme outcomes are possible and depend on the volatility of asset (Kealhofer, 2003).

Figure 2.1: Frequency distribution of asset value at horizon and probability of default



Note: The distance from the expected asset value to default is three standard deviations
Source: Kealhofer, 2003, p.31

Figure 2.1 permits to visualize how three main elements influence default: the asset, the volatility of such asset, and the liability.

The application of Black-Scholes-Merton's model goes beyond the possibilities of development of this research. The reader may refer to Black and Scholes (1973), Merton (1974),

and Young and Tudela (2003). On the other hand, the KVM model, developed in 1974 by Kealhofer, McQuown, and Vasicek (Kliestik et al., 2014), and the distance to default, provide interesting insights. Firstly, the KVM model attempts to find the probability of default of the company as a whole. Secondly, it introduces the concept of distance to default. In line with Figure 2.1, such model involves three steps: computation of an estimate of the market value, finding the distance to default, and scaling such distance to an actual probability.

The distance to default is defined as the *"distance between the expected value of assets of the company and the default point and then splits this difference by an estimation of the volatility of the company in a time horizon. At that time, the distance to default is replaced into a cumulative density function to compute the likelihood that the value of the company will be fewer than the face value of debt at the maturity of the debt"* (Ammann, cited in Kliestik et al., 2014, p.239). In other words, it provides an assessment of the business' default risk, measuring the distance between the expected value of the asset and the default point d* (i.e. value of debt). As Kliestik et al. (2014) provide

$$d^* = debt_{short\ term} + \frac{1}{2} debt_{long\ term} \tag{2.3}$$

and the distance to default DD

$$DD = \frac{E\big(F(t)\big) - d^*}{\sigma_F} = \frac{\ln\left({}^{F(t)}/_{D}\right) + \left(r - \frac{1}{2}\sigma_F^2\right)(T-t)}{\sigma_F\sqrt{T-t}} \tag{2.4}$$

where r = risk-free rate

F(t) = current asset value

D = face value of debt

σ_F = annualized company value volatility

Finally, the probability of default can be quantified as the probability that the current value of the asset F(t) falls below the face value of the debt D.

$$PD(t) = P[F(t) \leq D] \tag{2.5}$$

2.2. RANDOM FOREST AND APPLICATIONS

Machine learning applications can be viewed as a powerful tool in the credit and default risk prediction. While the adaptability of random forest could provide models, which are alternatives to the two different approaches previously described, likely a more appropriate comparison can be done with methods of the first group. In fact, random forest permit to use numerous variables and provides the relative variables' importance, therefore avoiding some of the limitations of the other methods.

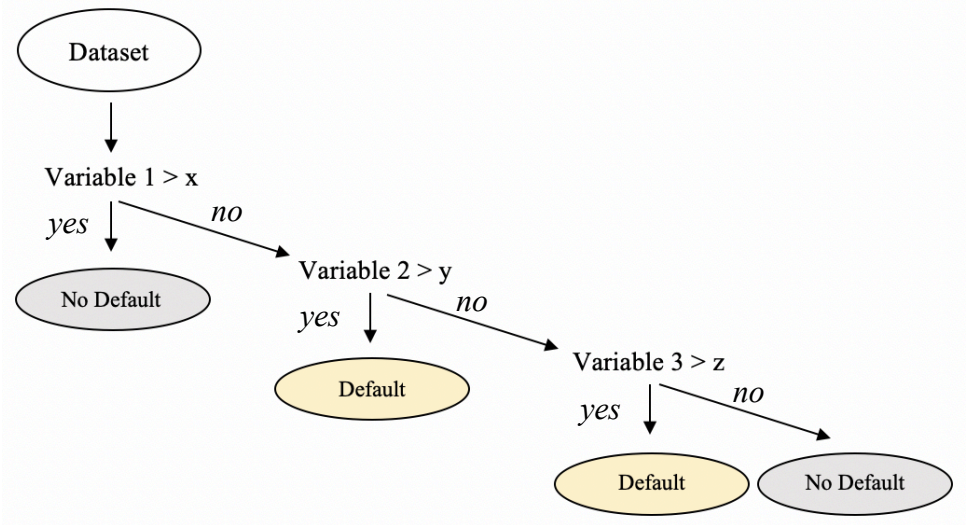
Identifying with certainty, at the time of the research, whether a company will default or not in the future is not possible. The number of variables that influences a company's lifecycle are numerous; therefore, not having information regarding future decisions taken by the management of a firm, and not knowing the impact and effects that external factors will have on the company, the only possibility is to estimate, based on the on the data available at the time of the analysis, the company's probability of default and its distance to default. As the previous sections present, this has been the objective of KVM's model.

Random forests are able to address the default topic with a wider perspective. Consider the machine learning application given by classification random forests (refer to Chapter 1 for specifics regarding random forests along with decision trees). Simply, given a dataset, given two outcome classes 'default' and 'no default', the process through which a decision tree predicts a class is provided by several decision points. At each decision point, the data is split; such process occurs until final leaves nodes are reached. Figure 2.2 provides a visualization of a decision trees' prediction-making process readapted to the topic discussed. The decision tree, then, based on the input data provided, votes one class.

Classification decision trees treat company's default as a binary variable, assigning the sample companies either to the defaulting class or to the non-defaulting class. A random forest is an ensemble of decision trees in which each tree votes a class and, therefore, the prediction output is most voted class (Breiman, 2001).

Random forests base their prediction on the input data provided. In other words, the number of variables through which predictions can be made regarding a company's default vary according to the researcher's discretion. Therefore, this shows the possibility of analyzing the topic without disregarding any factors[4]. Finally, this permits the creation of an adaptable model.

Figure 2.2: Decision Tree



2.3. CONCLUSIONS

The purpose of this research is to evaluate how well random forests predict if a company defaulted or is still active. In other words, once properly trained, the model will be used with new data, and its accuracy will be quantified based on the correctly and wrongly classified data. In line with this objective, this chapter summarizes and highlights the main concepts concerning companies' default prediction, by providing building blocks which will help the reader throughout the paper. Besides providing a general view of the literature on company default prediction, this chapter has been helpful in starting to develop the main analysis that will be presented in Chapter

[4] It should be noted that this does not mean that an infinite number of variables can be used. As provided in Chapter 2, the choice of variables is related to their importance. Nevertheless, by using a trial and error approach the analyst can be sure to not misjudge, and not include, variables whose importance is relevant.

3. Specifically, the elements described in this section were thoroughly used to select the variables used as predictors.

Random forests can be seen as an expansion and improvement to the univariate models' classification procedure. A possible further development could be, in fact, to repeat Beaver's (1967) study by creating a random forest using as variables the same financial ratios used by Beaver and by using the same data. This, in fact, would provide a great opportunity for comparison between the two methods. With respect to multiple discriminant analysis, random forests also do not have an equivalent to regression analysis' beta. Nevertheless, as provided in the previous chapter, a powerful tool in random forest is the computation of variable importance through the Gini index. Moreover, one of random forests advantages is its adaptability and its ability to discern, through its algorithms, the variables that have greater importance.

Finally, random forest, through machine learning, provides a method which seems to be reliable for the analysis of a complex topic such as company default prediction, given it masters a fundamental characteristic which is adaptability. In fact, random forests permit to consider many variables, therefore obtaining a model with a wide perspective and which describes the default topic as well as possible. One of the main disadvantages of random forest, as reported by Balcaen and Ooghe (2006), is that machine learning decision trees cannot measure causation relationships. A paper by Hastie and Zhao (2019), confirming Friedman (1996) findings, though, found that, in contrast with the conventional view, causal information can be extracted by drawing partial dependence plots and individual conditional expectation plots.

CHAPTER III

CLASSIFICATION OF COMPANY DEFAULT THROUGH RANDOM FORESTS

This chapter presents the development of the main study of this research. As previously discussed, companies' default has great relevance in many fields. The importance and applications of company failure prediction, then, has increased in order to improve both monitoring of indicators that have an impact, and the efficiency of actions taken to prevent company default.

The data, the methodology, and the tests used to obtain a clear view of the application of random forests to company default prediction are introduced and explained, so result and finding can be finally displayed and comprehended. Additionally, an R code has been developed in order to permit the reproduction of this analysis on other topics. The aim of the function created is to permit an easier and quicker process for the application of the random forest algorithm.

3.1. RESEARCH QUESTION

The possible perspectives from which to analyze such topic are numerous. Such variety permits to identify many reasons for doing such research and several ways in which the study can be assessed, as provided in Chapter 2.

With the aim of extending the literature on such topic, in this study random forests are used to address the research question "Can the random forest algorithm predict correctly company default?". Within this question, several others will be addressed by consequence, regarding both firm failure and Breiman's (2001) algorithm: "What are the main predictors of company default?", "How does missing data imputation affect the analysis?", "Does the random forest overfit?" are some of them.

Company's failure and probability of default depend on macroeconomic factors as well as specific financial indicators (Hazak and Mannasoo, 2007). For this reason, firm failure is linked to a great number of factors which vary from across countries (e.g. investments in

company development and innovation, competition between companies, and so on). In order to address the research question considered, this study will focus on Italian based companies.

3.2. DATA AND METHODOLOGY

3.2.1. Data

3.2.1.1. The AIDA data set

The AIDA database is part of the Bureau van Dijk, founded in 1991, and acquired by Moody's in 2017. Such organization provides consultation and help to clients in addressing a great array of issues and situations. For example, it provides assistance and coverage in strategy and business developments, managerial decision-making processes, mergers and acquisitions, financial investments, and many others. It does so thanks to the big data available (covering, worldwide, 365 million companies) [5].

Specifically, regarding the Italian market, the AIDA - Analisi Informatizzata dell'Azienda - data set contains information regarding both Italian public and private companies. Overall, it covers over 2 million companies, reporting information, among others, related to companies' structure, balance sheet, and financial ratios. The following paragraph will extend the understanding of variable selection. To obtain an initial comprehension of the data, Table 3.1 summarizes the predictors main statistics.

The data downloaded from the AIDA Database required some preparation before it could be used on R[6]. Firstly, *Legal Status*, the variable to be predicted, shows the company's status, distinguishing active companies form inactive ones. It has 10 values: *Active*, *Active (default of payments)*, *Active (receivership), Dissolved (merger), In Liquidation*, *Bankruptcy*, *Dissolved (bankruptcy)*, *Dissolved*, and *Dissolved (liquidation)*. These have been changed to binary for simplicity; in section 2.1.1 of Chapter 2 it has been assumed that bankruptcies, company failures, and defaults, are considered equal, beyond the reasons for such events. While this assumption carries strong limitations, it is to some extent in line with the classification made in the AIDA database. In fact, in AIDA the data is grouped in two secgions, which in this research will be used as follows:

- 1 has been assigned to *Active*, *Active (default of payments)*, *Active (receivership)*;

[5] AIDA Database https://www.bvdinfo.com/it-it/chi-siamo#secondaryMenuAnchor2
[6] See Appendix 2.2 for the codes.

- 0 has been assigned to *Dissolved (merger), In Liquidation*, *Bankruptcy*, *Dissolved (bankruptcy)*, *Dissolved*, and *Dissolved (liquidation)*.

This provides both a limitation to the research and some useful insights. In fact, this underlines the complexity of predicting company default. The latter is an event which may happen in many different forms and is linked to various factors. Intuitively, any model trying to predict the probability of default will incur in a certain level of error due to this complexity.

Secondly, missing values were recorded as *n.a.* and *n.s.*, which is not readable by R. These have all been transformed to NA. Thirdly, except for *Legal Status* which has been transformed to factor, all other variables have been assumed to be numeric (with the exception of the ATECO code, which has been solely used to classify the data in section 3.2.2.1.5).

3.2.1.2. Variable selection

The AIDA Database provides a numerous array of financial, structural, and strategical indicators, which permitted an initial approach with numerous variables. According to Breiman (2001), and confirmed by Papacharalampous and Tyralis (2017), one of the random forest algorithm's strength is that it performs well with many variables. In the recursive splitting process, the random feature selection provides that a random group of n variables is taken at each node; followingly, each variable is analyzed in order to find the split that reduces at most the impurity of the data (i.e. Gini Index and Information Gain). According to the mean reduction of the Gini Index that each variable provides, the predictors are assigned a level of importance. The `randomForest` and the `caret` packages in R both provide useful functions that asses such importance. Linked to this, importantly, random forests do not overfit (Breiman, 2001; Moisen, 2005). If properly tuned, in fact, the model will focus on finding patterns within the data, but it will not be strictly connected solely on the training data. The methodology section will provide a proof that the number of variables does not influence significantly the results in a random forest analysis.

The characteristics of the random forest algorithm, therefore, theoretically permit to include in this analysis all the indicators and figures present in the AIDA dataset. Nevertheless, some considerations are required. Firstly, the nature of the data and missing

values do not permit to include all indicators. Secondly, increasing the number of variables of the model significantly increases the time required for computations. Such additional time, as will be presented, does not provide an equivalent trade-off in terms of accuracy. Thirdly, in order to permit future comparison of predictive models, this study will consider variables in line with, among others, Beaver (1967), Altman (1968), Ciampi and Gordini (2008) and Edmister's (1972) studies. In other words, this study will mainly focus on financial ratios, assuming these describe appropriately a company's health situation. Finally, while the intent was to consider macroeconomic indicators as well (e.g. employees), due to missing data such variables could not be considered; moreover, imputing such data did not seem appropriate

Followingly, the lists containing the variables that were initially considered for this analysis is provided[7]. In the end, some we not included in the analysis.

Financial Indicators
Liquidity Ratio
Current Ratio
Current Liabilities/Tot Assets
Leverage
Coverage of Fixed Assets
Cost of Debt
Interest/Turnover
Solvency Ratio[8]
Share Funds/Liabilities
Net Financial Position
Debt to equity
Debt /EBITDA ratio
Banks/turnover
Management Ratios
Tot assets turnover (times)
Working cap turnover (times)

Profitability ratio
Earnings Before Inter. Tax Depr. Amm.
EBITDA/Sales
Return on Asset
Return on Investment
Return on Sales
Return on Equity
NetP&L/Operating P&L

Productivity Ratio
Number of employees
Turnover per employee
Added value per employee

Other significant data
Net working capital
Gross profit
Net short-term assets
Share funds - Fixed Assets
Cash Flow

[7] The ratios are listed and defined as reported in the AIDA database.
[8] By solvency ratio, the AIDA dataset provides the ratio between shareholder funds and total assets in percentage. Such ratio may also be known as the Shareholder Equity Ratio.

Banks/Turnover and has then been eliminated from the variables list due to a too high number of missing values. Productivity ratios, similarly, have not been included (see Appendix A for specifics regarding how these ratios and indicators have been computed).

Overall the ratios and figures considered express the way in which a company uses its assets (efficiency ratios), its ability to meet short term obligations (liquidity ratios), its ability to not miss debt commitments (solvency ratios), and its profitability (profitability ratios). While other figures may be useful, they may not be appropriate given the variety of size and industry the companies are sampled from; the ratios, instead, permit comparisons. This is not due to scaling reasons, since random forests deal appropriately with mixed variables and different scales (Breiman, 2001). Rather, it is linked to the nature of the data (i.e. if a more homogeneous sample of companies were taken, likely, more variables could be included in the analysis). Nevertheless, increasing the number of variables does not mean a certain increase in accuracy.

3.2.2. Methodology

In order to increase the generality of the model, this study's main dataset is made of 552,500 companies, which have been selected randomly from the AIDA dataset. Given the aim of testing the random forests algorithm's ability in predicting company default, the latter permitted to choose a sample of companies which covers, assumingly, the greatest number of company characteristics available in the AIDA dataset. These are, for instance, as many variations in the ATECO 2007 classification, both private and public companies, and a representation in the data of both small, medium, and large companies. It would be interesting to try the model on a more limited homogeneous dataset as well.

While the model has been developed and trained using the main dataset, an addition of 103,446 companies' information and figures have been downloaded for the final testing of the model.

The following segments will focus on the methodology of the research, results will be provided later on in the chapter. The quantitative part of this research has been developed in two sections. Firstly, a study of the random forest applied to the data permitted to identify

the best scenario for its application. Secondly, the model has been used on new data in order to verify the consistency of the model and test its accuracy.

3.2.2.1. Development of the model

The following section will present the steps which were followed throughout the analysis in order to identify the best characteristics to increase the model's accuracy. While the main analysis is limited to the first two steps, the other ones have been implemented in order to sustain the previous ones and provide helpful insights. In fact, step 3 and step 4 strengthen the methodology and approaches taken in the main analysis. Step 5, finally, shows an analysis a homogeneous dataset.

3.2.2.1.1. Step 1: preparation of data and samples

At first, rather than fixing the missing values issue, the latter have been eliminated from the data set and the algorithm has been run. The reason for which this has not been considered as an acceptable result is presented in the section 3.2.3. For computational reasons, a dataset of 552,500 companies was not feasible for analysis. Therefore, it has been divided in 9 smaller datasets sampled randomly with replacement from the main data set. This permitted to extend the research to various opportunities of analysis. Finally, a collection of companies has been created, which are all classified with similar ATECO 2007 descriptions (for simplicity, this will be referred to as Sample 10). Table 3.1 groups the size information and the purpose for which the sub-datasets were used.

For each data set an imputation of the missing data has been implemented. This was a required step given that, in relation to random forests, R requires the data clean of *NA*. Specifics will be provided in the section 3.2.3, covering both missing values and additional computation time.

Table 3.1: Data information: size, predictors, purpose

Dataset Name	nrows (companies)	ncol (variables/predictors)	Purpose
Main Dataset	55,250	26	
Sample 1	55,250	26	Sample 1 to 5 have been used to test random forest with 80% training and 20% testing data sets.
Sample 2	55,250	26	
Sample 3	55,250	26	
Sample 4	55,250	26	
Sample 5	55,250	26	
Sample 6	55,250	26	Used to test random forest with increased imputation parameters.
Sample 7	55,250	26	Used to test random forest with 70% training and 30% testing data sets.
Sample 8	55,250	26	Used to test random forest with 90% training and 10% testing data sets.
Sample 9	55,250	26	Used to test random forest with fewer predictors.
Sample 10	55,250	26	Used to test random forest on a homogeneous data set.

Data Source: AIDA database

3.2.2.1.2. Step 2: analysis across five samples

This step involves the analysis on the first five samples. While Breiman (1996; 2001) and Breiman et al. (1994) suggests 90% of the data should be used in training, such a big dataset permits to reduce such value. In fact, Breiman (2001) advises the use of high percentages of data for training so thata as much as possible of the information available within the data is used to improve the model. Nevertheless, a random forest computation has been run with training and testing set at, respectively, 90% and 10%; the results show no significant improvement.

For each random forest created, the parameters have been tuned, so that each parameter would lead to the greatest accuracy possible. Specifically, this involved finding the best *mtry*, the best *maxnodes*, and the best *ntree*. No significant difference occurs by changing the *nodesize*, which has been kept constant at 27. While Breiman (2001) states the trees in random forests can be fully grown, identifying the best values for the parameters regarding the trees' depth permitted to maintain the model's accuracy while reducing the computation

time. Finally, the random forest with the appropriate parameters have been generated, and its results and time of computation have been recorded.

3.2.2.1.3. Step 3: change of training and testing data set size

As previously mentioned, Breiman (1996; 2001) and Breiman et al. (1994) advocate for a training set which amounts to 90% of the data. This section has been carried through in order to explore the possibilities that better accuracy can be achieved changing these parameters. For this reason, both an increase and a decrease of the percentage of data used for training has been considered.

The equivalent steps taken in step 2 have been repeated, adapting accordingly the training and testing data sets.

3.2.2.1.4. Step 4: reduction of number of variables and predictors

An important feature of random forests is its assessment of variable importance. By recursively generating trees, the variables used which accounted for the greatest impurity reduction, on average. By constructing n trees, the variable importance is quantified n times and average by the random forest algorithm. This information has been used to construct this step.

The variable importance has been assessed from Sample 1 to Sample 5, and a score from 1 to 26 has been awarded to the predictors. By summing the results of each run, a list of variables has been obtained, in which the ones with the lowest score are the most important predictors. This creates a limitation given that the difference in importance across variables is not equal for all variables. Nevertheless, assumedly, by averaging over five different samples this limitation is lessened. Finally, a random forest is generated following the procedures of step 2 considering only the first 15 predictors in order of importance.

3.2.2.1.5. Step 5: a homogeneous dataset

This step has been mainly designed to provide an initial view of the application of random forests in homogeneous scenarios. Therefore, this should be view as incentive and basis to try to prove if such algorithm is able to predict default more precisely in some sectors rather than others.

Table 3.2. summarizes the composition of the dataset. Choosing one single ATECO 2007 code seemed reductive. Therefore, all companies belonging to the hotels and renting industries have been included.

Table 3.2: Composition of data of Sample 10

ATECO 2007 code	ATECO 2007 description (Italian)	ATECO 2007 description (English)
550000	Alloggi	Accomodation/Housing
551000	Alberghi	Hotels
552000	Alloggi per vacanze e altre strutture per brevi soggiorni	Holiday accommodation and other facilities for short stays
552010	Villaggi turistici	Resorts
552020	Ostelli della gioventù	Youth Hostels
552030	Rifugi di montagna	Mountain Huts
552040	Colonie marine e montane	Marine and Mountain Colonies
552050	Affittacamere per brevi soggiorni, case ed appartamenti per vacanze, bed and breakfast, residence	Guest houses for short stays, holiday homes and apartments, bed and breakfasts, residences
552051		

Data Source: AIDA database

The equivalent steps taken in step 2 have been repeated, using the same 80% of the data for training and 20% for testing.

3.2.2.2. Training and testing of the final model

Having identified the main characteristics and the parameters which provide the highest accuracy, these have been used to test the algorithm. A training set made of companies extracted in the previous analyses has been prepared, selecting 110,533 companies, and it has been used to train the model. The parameters used are those that, by observing all steps of the "Development of Model" phase, provide greater accuracy. Finally, rather than diving the data in two sets for training and testing, a new dataset has been prepared, composed of 103,446 companies which have never been used in the previous steps. The algorithm has followingly been used to make a prediction of the status of such companies.

3.2.3. Main issues

3.2.3.1. Missing values

While random forests can handle, in theory, missing data (Breiman, 2001; Biau & Scornet, 2010), from a practical point of view in R programming considerations are required given that assumptions and adjustments need to be done.

In theory, an estimation of the missing values is implemented by the algorithms by constructing a proximity matrix, which groups the proximity of pairs of observations computed (Biau & Scornet, 2010). In other words, the algorithm will search for similar ones and replace the respective missing value.

From a practical point of view, it should be noted that both the R packages that provide functions for this analysis, `caret` and `randomForest`, require either a dataset with no missing values or specifications regarding how to deal with eventual NA encountered. Given the unmissable presence of missing data, this issue has been addressed as follows.

Firstly, a trial has been done specifying that all companies with at least one missing value should not be considered in the analysis. This, of course, led to reduce the dataset (64,631 companies). The results obtained are displayed in Table 3.3.

Table 3.3: Eliminating NAs

		Value
Errors	Type I	91.02%
	Type II	0.56%
	OOB Estimate	11.48%
	Accuracy	89.06%
	NIR	88.44%
	p-value	0.0215

Data Source: AIDA database

With the `caret` and the `randomForest` packages in r, several figures are shown when the model's results are printed. Specifically, among others, two sections are relevant: one provides information regarding the model errors, quantifying the out of bag error and providing measurements of the Type I error, or falsely predicted positives, (i.e. a company is predicted defaulted when it is active) and the Type II error, or falsely predicted negatives (i.e. a company is predicted active when it is defaulted). The other is linked to the model's

accuracy and significance. In this case, as described in the `caret` package description available through the `help()` function in R, a 95 percent confidence interval is used when computing the accuracy of the model, and a one-sided test is run to see if the model performs better than the no information rate prediction. Therefore, by observing the p-value, it is possible to verify, or not, the correctness of the hypothesis.

An 89.06% accuracy and a relatively low OOB error estimate may lead to believe the model performed well. Firstly, the p-value is not small enough to guarantee statistical significance of the result. Secondly, the No Information Rate is really high, and not much lower than the accuracy. This underlines the likelihood that the elimination of the companies led to considering mostly companies which are still active (i.e. defaulted companies are more incline to having missing data). Consequently, confirming this theory, the errors underline where the model does not predict well. Almost the totality of the error is done on predicting, as defaulted, active companies. This should lead to raise the attention regardless of the relatively low OOB error. Therefore, these considerations show how in such conditions the model does not identify properly the patterns regarding the defaulted companies, leading, in fact, to many firms classifies as defaulted but actually still active. It should be noted that these results are not linked to the size of the dataset used, as will be shown in the results section, rather on the composition and nature of the dataset.

The purpose of this study is to test random forests and their performance, and to find a model which is appropriate and applicable with various and diversified conditions, rather than showcasing high accuracy measurements which do not have statistical significance. To do so, trials have been carried out in different situations. Such experiments will be presented throughout the chapter.

In order to maintain as much data information possible, an imputation approach has been implemented. Specifically, the missing values have been imputed using random forest iterations, therefore predicting recursively the missing figures. Such algorithm stops its process either when the maximum number of iterations is reached, or when the model senses that no improvement in the prediction is reached by computing the differences in the data across the iterations (Stekhoven, 2011; Stekhoven and Buhlmann, 2012) The imputation with the smallest differences generates the dataset to be used.

```
Imputated_SampleData <- missForest(Sample, maxiter = 10, ntree = 100,
                                   variablewise = T, verbose = T)
```

Specifically, `maxiter` refers to the maximum number of iterations the code should implement, and `ntree` reports the number of trees the forest constructed is made of. As provided in Stekhoven (2011) with a big data set less iterations and trees are required. For computation reasons, these parameters have been set to 3 and 20 respectively. In order to prove this, a trial has been done (Sample 6) with 5 iterations and 50 trees. The results, displayed in Table 3.4, are in line with those of the study's main tests on Samples from 1 to 5 which will be displayed in the following sections. Therefore, there is not a useful trade-off, in this case, between the accuracy of the model and the additional computation time required (about 3 hours above the average).

An important aspect that should be underlined is that the imputation of the data does not increase the information of the data. On the other hand, it should be noted that imputation of data may influence negatively the result of the analysis by increasing the error. For this reason, in order to verify the correctness of using `missForest` with this type of data, a test has been run by comparing a complete dataset (Sample 2) and the imputed dataset, quantifying the error present between the two. Such error result to be quite high; in a five-iteration run it falls within the range of [0.46; 0.53]. This will have an impact on the final results of the study.

Table 3.4: Imputation of data with maxiter=5 and ntree=50

			Time (s)	Time (h/m/s)
Sample 6	Imputation of NAs	Iteration 1	9778.683	2:42:59
		Iteration 2	8681.916	2:24:42
		Iteration 3	8518.775	2:21:59
		Iteration 4	6988.686	1:56:29
		Iteration 5	6756.202	1:52:36
		Total	40724.262	11:18:44
	Best mtry		5281.858	1:28:02
	Best maxnodes		22824	6:20:24
	Best ntrees		6608.394	1:50:08
	Final Random Forest		1115.472	0:18:35

		Value
Errors	Type I	51.85%
	Type II	8.55%
	OOB Estimate	22.40%
Accuracy		78.01%
NIR		67.66%
p-value		< 2.2e-16

Data Source: AIDA database

As provided by Mukherjee et al. (2013) `missForest` tends to provide better results than other imputation methods. To gain a deeper insight, another imputation method has been implemented, through the R package `mice`, which the objective of finding a lower error and a better imputed dataset. While such function generally seems to work appropriately, its results are not acceptable in this case, given that the data is not missing randomly, which is an important assumption for the package `mice`. Moreover, given the type of data, collinearity and possible linear dependencies prevent the use of such method, which provides high errors and inaccurate predictions. This provides possibilities for further research and development of this study, trying to find more suitable imputation methods.

Given the previous assumption that most NAs are linked to defaulted companies, such error will likely result in an increase in the type I error (i.e. companies predicted defaulted but actually active). The reason for this assumption is linked to the fact that by imputing many values for defaulted companies and given the fact that the imputation process generates high errors, the difficulty in identifying patters which permit default prediction increases. Unfortunately, the nature of the data, and its analysis through the random forest algorithm, encounter such issue. An additional complication due to the imputation of the data of such a large dataset is linked to the high computation time it required. Such topic will be addressed in the following section. In evaluating the final models, then, these should be taken into consideration.

The following Table 3.5 reports the differences computed in the imputation of the first five samples. Highlighted are, in fact, the smallest differences, and, therefore, the imputed data sets which were used. As shown, some samples (i.e. Sample 1, 4, and 5) did not require more than three iterations to detect the best imputation, confirming the assumptions previously made regarding the imputation of large data sets.

Tables 3.5: Differences and imputation parameters

		Differences (predictor variables)	Differences (dependent variable)	Characteristics (max n. iterations, n. trees)
Sample 1	Iteration 1	0.001155294	0	maxiter = 3 , ntree = 20
	Iteration 2	0.000289698	0	maxiter = 3 , ntree = 20
	Iteration 3	0.000296456	0	maxiter = 3 , ntree = 20
Sample 2	Iteration 1	0.01837136	0	maxiter = 3 , ntree = 20
	Iteration 2	0.01270615	0	maxiter = 3 , ntree = 20
	Iteration 3	0.00141429	0	maxiter = 3 , ntree = 20
Sample 3	Iteration 1	0.006264864	0	maxiter = 3 , ntree = 20
	Iteration 2	0.003706305	0	maxiter = 3 , ntree = 20
	Iteration 3	0.002901615	0	maxiter = 3 , ntree = 20
Sample 4	Iteration 1	0.01743522	0	maxiter = 3 , ntree = 20
	Iteration 2	0.002134286	0	maxiter = 3 , ntree = 20
	Iteration 3	0.002766341	0	maxiter = 3 , ntree = 20
Sample 5	Iteration 1	0.001288294	0	maxiter = 3 , ntree = 20
	Iteration 2	0.000169625	0	maxiter = 3 , ntree = 20
	Iteration 3	0.000364519	0	maxiter = 3 , ntree = 20

Data Source: AIDA database

It can be noted that the differences for the dependent variable are always equal to 0, meaning that the dataset does not have any missing values regarding the *Legal Status* of the companies.

3.2.3.2. Computing time

One important issue related to this research is the time required to run computations. While this does not influence the results, this should be regarded as a limitation rather than an issue impacting the final result. The timing has been registered by the R package `tictoc`.

The time required to prepare the data (without considering the time to download the data) is not significant.

Table 3.6: Computation times for imputation, tuning, and random forest

			Time (s)	Time (h/m/s)
Sample 1	Imputation of NAs	Iteration 1	4485.709	1:14:46
		Iteration 2	2424.989	0:40:25
		Iteration 3	4428.683	1:13:49
		Total	11339.381	3:08:59
		Best mtry	2892.313	0:48:12
		Best maxnodes	2270.123	0:37:50
		Best ntrees	7618.174	2:06:58
		Final Random Forest	2127.423	0:35:27
Sample 2	Imputation of NAs	Iteration 1	2279.553	0:38:00
		Iteration 2	2688.883	0:44:49
		Iteration 3	6325.12	1:45:25
		Total	11293.556	3:08:14
		Best mtry	4202.287	1:10:02
		Best maxnodes	4378.586	1:12:59
		Best ntrees	8994.032	2:29:54
		Final Random Forest	1215.69	0:20:16
Sample 3	Imputation of NAs	Iteration 1	4768.542	1:19:29
		Iteration 2	4094.993	1:08:15
		Iteration 3	3958.735	1:05:59
		Total	12822.27	3:33:42
		Best mtry	6608.087	1:50:08
		Best maxnodes	2917.912	0:48:38
		Best ntrees	5568.229	1:32:48
		Final Random Forest	1006.133	0:16:46
Sample 4	Imputation of NAs	Iteration 1	4511.137	1:15:11
		Iteration 2	3756.001	1:02:36
		Iteration 3	4756.921	1:19:17
		Total	13024.059	3:37:04
		Best mtry	4171.258	1:09:31
		Best maxnodes	1961.774	0:32:42
		Best ntrees	6608.394	1:50:08
		Final Random Forest	1615.472	0:26:55
Sample 5	Imputation of NAs	Iteration 1	5518.775	1:31:59
		Iteration 2	5788.686	1:36:29
		Iteration 3	5756.202	1:35:56
		Total	17063.663	4:44:24
		Best mtry	5281.858	1:28:02
		Best maxnodes	2824.138	0:47:04
		Best ntrees	5318.651	1:28:39
		Final Random Forest	1115.472	0:18:35

Data Source: AIDA database

For these calculations a machine with the following characteristics has been used: Dual Core Intel Core i5, with a processor speed of 2.3 GHz, and 8GB of RAM.

The topic should be divided in two parts. Firstly, the random forest algorithm itself does not require much time for computation. As Table 3.7 reports, on average, in the main analysis, it took 23 minutes to generate the forest. The result differs considering the tuning process. Specifically, finding the best number of trees to be constructed required almost 2 hours. A dispute in the literature is ongoing in relation to the number of trees to be constructed. For instance, Oshiro et. al. (cited in Papacharalampous and Tyralis, 2017) believes only 64 to 128 trees are necessary, while Kuhn and Johnson (cited in Papacharalampous and Tyralis, 2017) suggests using at least 1000 trees. In this research, not being able to detect and value which approach is the most suitable, the number of trees has been computed for every case; this, of course, led to an increase in computational cost. Secondly, the imputation of the data accounts itself for 3 hours of computation, on average.

Table 3.7: Sample 1 to 5 average times and total average times

	Time (h/m/s)
Sample 1	7:17:27
Sample 2	8:21:24
Sample 3	8:02:03
Sample 4	7:36:21
Sample 5	8:46:44

	Average Time (h/m/s)
Average Time per Sample	8:00:48
Tot Iterations	3:38:29
Best mtry	1:17:11
Best maxnodes	0:47:51
Best ntrees	1:53:41
Total Tuning	3:58:43
Final Random Forest	0:23:36

Data Source: AIDA database

On average, the entire process for each sample required 8 hours of computations. The timing for steps 3, 4, and 5 are equivalent.

While a stronger calculator would likely reduce the average time, the time required for these computations are high. As previously stated, this does not affect the results. In comparing different models, though, for the same accuracy and error estimate the computational cost of running a model may be a key factor in the choice between two models.

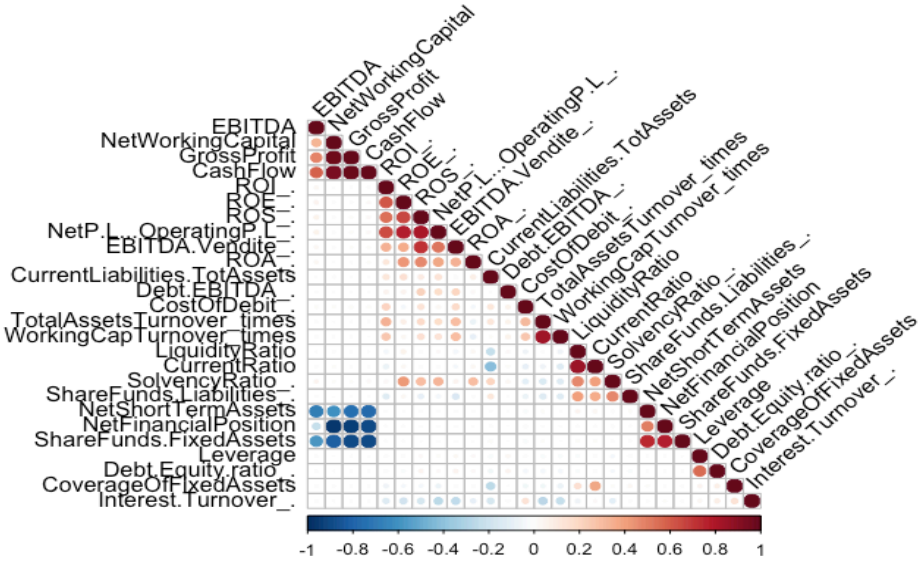
In general, considering that the imputation of the data was necessary for the nature of the data in this case, but may not be required in every study, and assuming a fixed number of trees as provided by various studies, generating a random forest with a dataset of about 55 thousand units would require approximately 2 hours and a half.

3.2.3.3. Correlation among predictors

The final considerations required before presenting the results of this study concern the correlation among predictors. Constructing a great number of trees, each on its own data random forests are able to obviate significantly such issue. Nevertheless, the correlation among predictors may still have an impact. For this reason, the correlation between the variables has been computed and graphed for all five samples used in the main analysis in order to consider all possibilities. Figure 3.1 shows the results for Sample 1[9]. Given that the financial indicators and ratios are computed on figures from the company's balance sheet, a certain level of collinearity occurs.

[9] The correlation graphs for the other samples did not show significant differences from Figure 3.1, therefore have not been presented.

Figure 3.1: Correlation among predictors in Sample 1



Data Source: AIDA database

The reason for which this information is observed is to detect how many variables are correlated and the extent to which they are correlated. Consider the splitting process in the decision trees. Once a predictor is used to decrease the impurity of a node, it directly means that all variables which positively correlated to it will be less likely to be used given that two correlated predictors both reduce the same impurity.

As it can be observed in Figure 3.1, not many variables are highly correlated (e.g. *Gross Profit* with *Cash Flow*, *Gross Profit* with *Net Working Capital*, *Share Funds - Fixed Assets* with *Net Financial Position*). Given the small number of such group, it is reasonable to assume that with its randomization procedures the algorithm will obviate entirely such issue.

3.3.RESULTS AND FINDINGS

3.3.1. Results: development of the model

3.3.1.1. Step 2 results: analysis across five samples

This part of the analysis concerns five sub-data sets, sampled randomly with replacement from the main data set. The reason for repeating the analysis over various samples is to test the consistency of the model. Table 3.8 summarizes the results obtained for each sample.

Table 3.8: Accuracy and other results, Sample 1 to 5, Step 2

			Value (80%train - 20%test)
Sample 1	Errors	Type I	51.75%
		Type II	8.39%
		OOB Estimate	22.40%
		Accuracy	77.85%
		NIR	67.72%
		p-value	< 2.2e-16
Sample 2	Errors	Type I	49.65%
		Type II	8.69%
		OOB Estimate	21.85%
		Accuracy	78.25%
		NIR	67.42%
		p-value	< 2.2e-16
Sample 3	Errors	Type I	47.61%
		Type II	9.51%
		OOB Estimate	21.83%
		Accuracy	79.09%
		NIR	68.46%
		p-value	< 2.2e-16
Sample 4	Errors	Type I	48.52%
		Type II	9.03%
		OOB Estimate	20.98%
		Accuracy	79.12%
		NIR	67.23%
		p-value	< 2.2e-16
Sample 5	Errors	Type I	47.67%
		Type II	8.71%
		OOB Estimate	20.95%
		Accuracy	80.03%
		NIR	67.87%
		p-value	< 2.2e-16

Data Source: AIDA database

In order to evaluate a predictive model, the accuracy measurement itself is not sufficient. By introducing the No Information Rate (NIR), though, the assessment of the model's predictive power can be contextualized. The NIR represents the prediction accuracy based exclusively on the dependent variable's distribution. In other words, it represents the best guess at predicting, in this case, companies' status (*Active* or *Defaulted*) in the five samples considered. The NIR, then, is equivalent at the percentage of the majority's class (i.e. in this case, *Active*). Intuitively, a model with accuracy below the NIR level shouldn't be considered. The model trained in this analysis, on average, performed 11.13% above the NIR.

The Out-Of-Bag error (i.e. the error computed during the bagging process on the data not sampled for training) resulted from these analysis amounts to about 21% on average. Building on this, the OOB score (i.e. OOB accuracy) can be easily derived and quantified at about 79%. Such results underline the consistency of the model. In fact, the predictive accuracy on both the non-sampled training set, and on the testing dataset, are approximately

equal. This result is promising. Regarding the OBB error, it should be noted that its importance in a binary classification test may be limited (Hornung and Janitza, 2018). The authors, in fact, state that in some tests, and in binary classification predictions specifically, the OOB error may be overestimated. This result suggests that the accuracy of this model may be greater than those provided in Table 3.9, averaging the results of the five samples.

Table 3.9: Average results, Step 2

			Value (80%train - 20%test)
Sample 1:5	Errors	Type I	49.04%
		Type II	8.87%
		OOB Estimate	21.60%
		Accuracy	78.87%
		NIR	67.74%
		p-value	< 2.2e-16

Data Source: AIDA database

As previously assumed, type I errors result to be quite high. Given the premises regarding the nature of the data and the imputation errors, the relevance of such measurement in this analysis should be carefully weighted. The promising results in terms of model consistency and accuracy, in fact, should not be run down by this error. Nevertheless, further analysis should be carried in order to quantify the true value of the wrongly predicted defaulted companies.

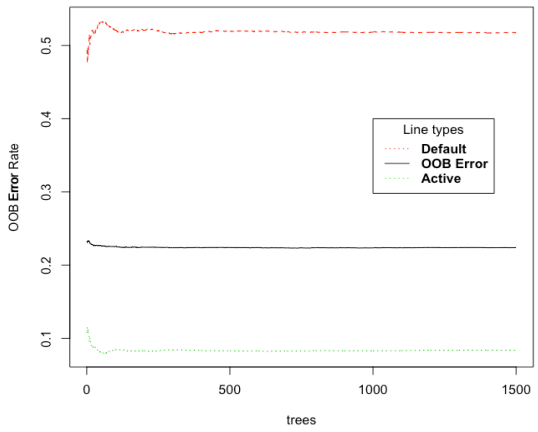
A representation of the errors can be found in Figures 3.2 to 3.6. In terms of error quantification, the consistency of the model still holds.

Overall, these performances provide promising results in generating a random forest with the following parameters[10]: mtry = 10, maxnodes = 34, nodesize = 27, and ntree = 1000.

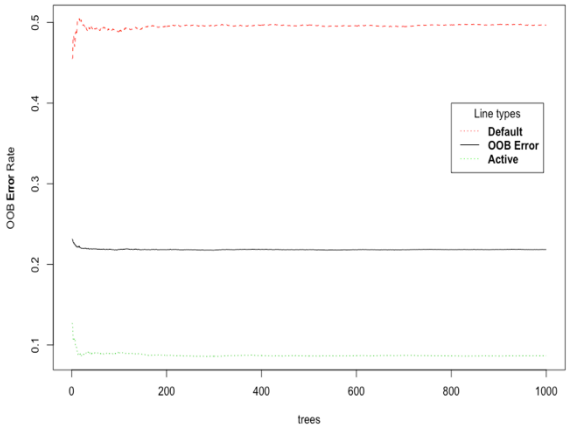
[10] Such values have been obtained averaging the parameters of the five samples. Specifically, the results obtained are mtry = 9.6, maxnodes = 33.6, nodesize = 27, and ntree = 1000.

Figures 3.2 to 3.6

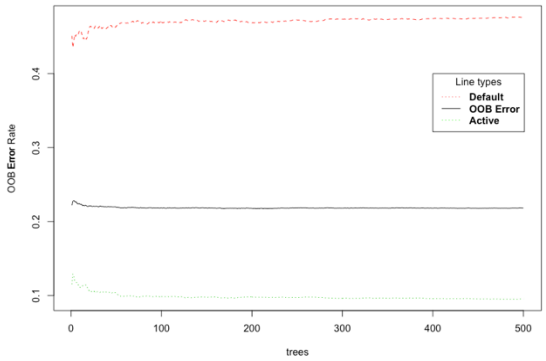
3.2 Sample 1: class errors and OOB errors



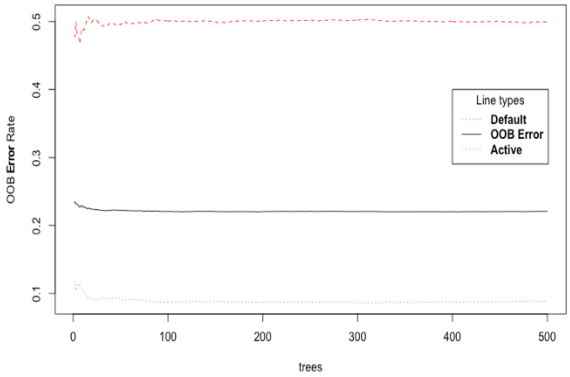
3.3 Sample 2: class errors and OOB errors



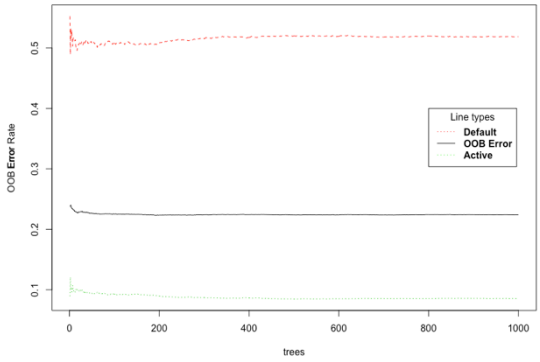
3.4 Sample 3: class errors and OOB errors



3.5 Sample 4: class errors and OOB errors



3.6 Sample 5: class errors and OOB errors



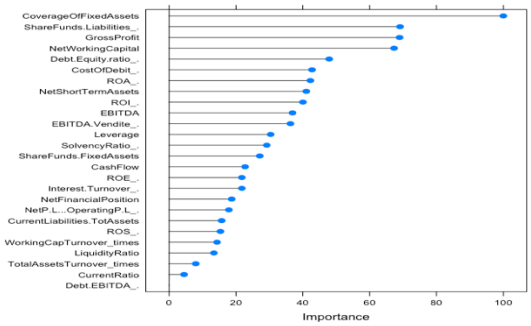
Data Source: AIDA database

Finally, the importance of the variable has been assessed and graphed (Figures 3.7 to 3.11). This result is helpful in understanding which variables the algorithm considers more relevant as predictors. Moreover, this result has been used in Step 4 to test the model with a lower number of variables. The results shown in Figures 3.7 to 3.11, though, may suggest

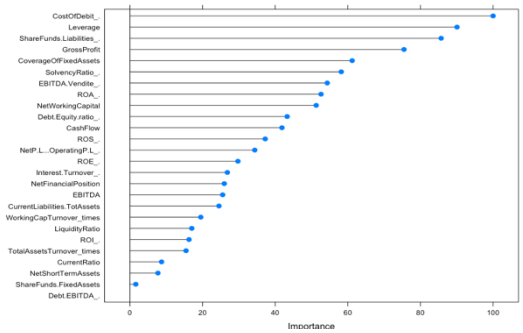
the inconsistency of random forest across samples. Nevertheless, by taking into consideration the limitations of the datasets and the adjustments made in this research, it can be observed that often the position of a predictor variable in one figure with respect to the others figures changes of approximately 2 to 5 positions (e.g. observe *Coverage of Fixed Assets*, *Gross Profit* and *Current Ratio*). Those whose position varies significantly across Figures 3.7 to 3.11 may be heavily impacted by the complexity and diversity of the data.

Figures 3.7 to 3.11: variable importance, Sample 1 to 5

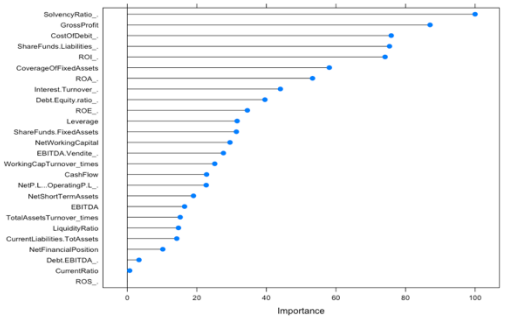
3.7 Sample 1: variable importance



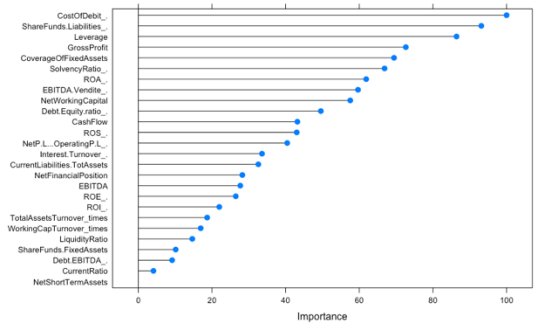
3.8 Sample 2: variable importance



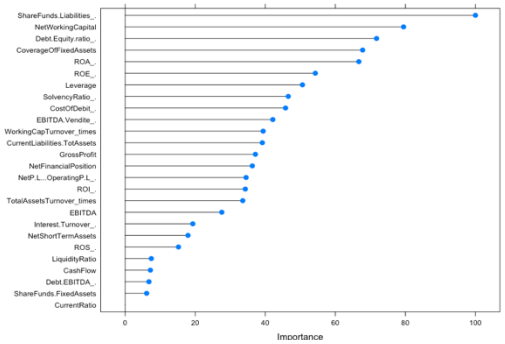
3.9 Sample 3: variable importance



3.10 Sample 4: variable importance



3.11 Sample 5: variable importance



Data Source: AIDA database

Table 3.12 presents these results averaged across the five samples. Specifically, the variables which are considered most important in predicting the company's default are *Share Funds/Liabilities*, *Coverage of Fixed Assets*, *Gross Profit*, *Cost of Debit*, *Return on Assets* and the *Solvency Ratio*. Nevertheless, as Figures 3.7 to 3.11 show, few variables have very low importance. Surprisingly, the *Current Ratio*, *Liquidity Ratio*, and the *Return on Sales*, in some samples, tend to account for low importance.

3.3.1.2. Step 3 results: change of training and testing data set size

Given the extensive amount of data, no significant differences emerge from changing the percentage of data used for training and testing the model. As Table 3.10 and 3.11 show, results are very similar to those of Table 3.9.

Table 3.10: Results 70% train 30% test, Step 3

			Value (70%train - 30%test)
Sample 7	Errors	Type I	50.10%
		Type II	8.88%
		OOB Estimate	22.19%
		Accuracy	78.07%
		NIR	67.74%
		p-value	< 2.2e-16

Data Source: AIDA database

Table 3.11: Results 90% train 10% test, Step 3

			Value (90%train - 10%test)
Sample 8	Errors	Type I	49.94%
		Type II	8.46%
		OOB Estimate	22.07%
		Accuracy	78.02%
		NIR	67.74%
		p-value	< 2.2e-16

Data Source: AIDA database

In the application of this algorithm in relation to company default, with this dataset, then, this parameter is not relevant in the ranges analyzed.

3.3.1.3. Step 4 results: reduction of number variables and predictors

Step 4 has been designed to confirm random forests' strength of not being influenced significantly by a high number of predictors. Ideally, the experiment should be run several times and with a different number of variables. Nevertheless, as Table 3.14 shows, the results are in line with the other obtained previously.

As represented in Figures from 3.7 to 3.11, the difference, in term of importance, from one predictor to the other is not equal. By recursively considering the importance across five samples though, we may assume that assigning a score from 1 to 26 does not alter the analysis. Specifically, it is safe to assume the first 15 variables highlighted in table 3.12 are truly the most important variables. Table 3.12 reports the order of importance for each sample and provides a final list in which the predictors are ordered from the most important to the least important.

Table 3.12: Variable Importance

	Var Imp (Sample 1)	Var Imp (Sample 2)	Var Imp (Sample 3)	Var Imp (Sample 4)	Var Imp (Sample 5)	TOTALE		
Coverage of Fixed Assets (%)	1	5	6	5	4	12	Share Funds/Liabilities (%)	1
Share Funds/Liabilities (%)	2	3	4	2	1	21	Coverage of Fixed Assets (%)	2
Gross Profit (EUR)	3	4	2	4	13	26	Gross Profit (EUR)	3
Net Working Capital (EUR)	4	9	13	9	2	20	Cost of Debit (%)	4
Debt/Equity Ratio (%)	5	10	9	10	3	34	Return On Assets (%)	5
Cost of Debit (%)	6	1	3	1	9	34	Solvency Ratio (%)	6
Return On Assets (%)	7	8	7	7	5	35	Leverage	7
Net Short Terms Assets (EUR)	8	24	18	26	20	37	Net Working Capital (EUR)	8
Return On Investment (%)	9	21	5	19	16	37	Debt/Equity Ratio (%)	9
EBITDA (EUR)	10	17	19	17	18	50	EBITDA/Sales (%)	10
EBITDA/Sales (%)	11	7	14	8	10	64	Return On Equity (%)	11
Leverage	12	2	11	3	7	70	Return On Investment (%)	12
Solvency Ratio (%)	13	6	1	6	8	73	Interest/Turnover (%)	13
Share Funds - Fixed Sssets (EUR)	14	25	12	23	25	76	Cash Flow (EUR)	14
Cash Flow (EUR)	15	11	16	11	23	77	Net P&L/Operating P&L (%)	15
Return On Equity (%)	16	14	10	18	6	81	EBITDA (EUR)	16
Interest/Turnover (%)	17	15	8	14	19	87	Current Liabilities/Total Asssets (%)	17
Net Financial Position (EUR)	18	16	23	16	14	87	Net Financial Position (EUR)	18
Net P&L/Operating P&L (%)	19	13	17	13	15	88	Working Capital Turnover (times)	19
Current Liabilities/Total Asssets (%)	20	18	22	15	12	92	Return On Sales (%)	20
Return On Sales (%)	21	12	26	12	21	96	Net Short Terms Assets (EUR)	21
Working Capital Turnover (times)	22	19	15	21	11	99	Share Funds - Fixed Sssets (EUR)	22
Liquidity Ratio	23	20	21	22	22	103	Total Assets Turnover (times)	23
Total Assets Turnover (times)	24	22	20	20	17	108	Liquidity Ratio	24
Current Ratio	25	23	25	25	26	124	Current Ratio	25
Debt/Ebitda Ratio (%)	26	26	24	24	24	124	Debt/Ebitda Ratio (%)	26

Data Source: AIDA database

Then, variables from position 16 onwards have not been included in the code. Finally, the output for this run are provided in Table 3.13.

Table 3.13: Results, Step 4

			Value (80%train - 20%test)
Sample 9	Errors	Type I	49.19%
		Type II	9.07%
		OOB Estimate	21.98%
		Accuracy	78.11%
		NIR	67.74%
		p-value	< 2.2e-16

Data Source: AIDA database

As it can be observed, the results of this experiment are in conformity with those averaged in Table 3.9. Importantly, these results confirm the theory that Breiman's (2001) algorithm handles correctly both runs with many of fewer predictors.

3.3.1.4. Step 5 results: a homogeneous data set

One of the main purposes of this research is to provide a model which is generally applicable. In other words, a method which delivers good performance across different industries and sectors. This experiment presents a starting point for evaluating the possibility of developing specified sectors' random forest and compare performances with the model established in Step 2.

While this test could be run on any sector, here the focus is placed on the hotels and renting sector. With the purpose of comparison with the results of Table 3.10, 80% of the data has been used for training and 20% for testing.

By observing results in Table 3.14, some considerations are required. Firstly, regardless of the reduction of data (30,349 companies), the results show that the accuracy is statistically lower than the no information rate. Secondly, the model provides for a great accuracy (4.16% higher than that of Table 3.9). Nevertheless, the No Information Rate increased by 11.06%, meaning that the chances of predicting correctly based solely on *Legal Status*' distribution has increased greatly. Thirdly, regarding the error, the OOB estimate decreased significantly.

Table 3.14: Results, Step 5

			Value (80%train - 20%test)
Sample 10	Errors	Type I	69.35%
		Type II	2.79%
		OOB Estimate	16.53%
		Accuracy	83.03%
		NIR	78.80%
		p-value	< 2.2e-16

Data Source: AIDA database

Considering these observations, in this case, the increase in accuracy does not provide for a better model, given it generates a higher error in defaulted companies' prediction (see type I error). This may be linked to the fact that in Sample 10 24,034 companies are *Active*, and only 6,315 are *Defaulted*. Moreover, in relation to the no information rate, it is shown that this experiments provides a smaller difference between the accuracy and the NIR, compared to the main analysis.

3.3.2. Results: training and testing of final model

Building on the results and considerations obtained from the previously explained steps, a final model has been implemented. The results of this test, together with those of the analysis across five samples, provide a clearer picture of the application of Breiman's (2001) algorithm to predicting company's default.

This final implementation of the algorithm has been done to obtain both further proof of the consistency of the model. For these reasons, a training and testing data sets have been designed. Moreover, a different approach has been developed for this section:

- Given that the purpose of the analysis of this section is to develop a model built on the information and features obtained in the previous steps, the random forest has been trained using a sample of 110,533 companies obtained from the main dataset.
- In addition to the first dataset of 552,500 companies, a new test set of 103,446 firms has been downloaded. This permits to remove the chance that the model is familiarized with the data.

The results of the main analysis shown in step 2 show that the parameters which provide greater accuracy are: mtry = 10, maxnodes = 34, nodesize = 27, and ntree = 1000. Therefore, these have been used to tune the random forest. Table 3.15 reports the results obtained. As it can be observed, the results do not differ much from those of Table 3.10, providing promising results both in terms of consistency and accuracy. It should be kept in mind the data used for training and testing still required data imputation, given that not enough companies with no missing data were available to perform an analysis without imputation. Likely, by improving the imputation process, and by finding alternatives to the use of the `missForest` approach, both type I error and the OOB error should decrease, and the accuracy should increase. Nevertheless, the accuracy of 78.13% obtained is 12.01 percentage points higher than the no information rate.

Most prediction errors, though, are done in predicting as defaulted active companies (e.g. type I error). Given the model seems to perform appropriately, this requires further research in order to identify the reason for such high value. For instance, some possible hypotheses are:

- Such increase is generated by the imputation process;
- The complexity of the topic will likely be characterized by a relatively high type I error.

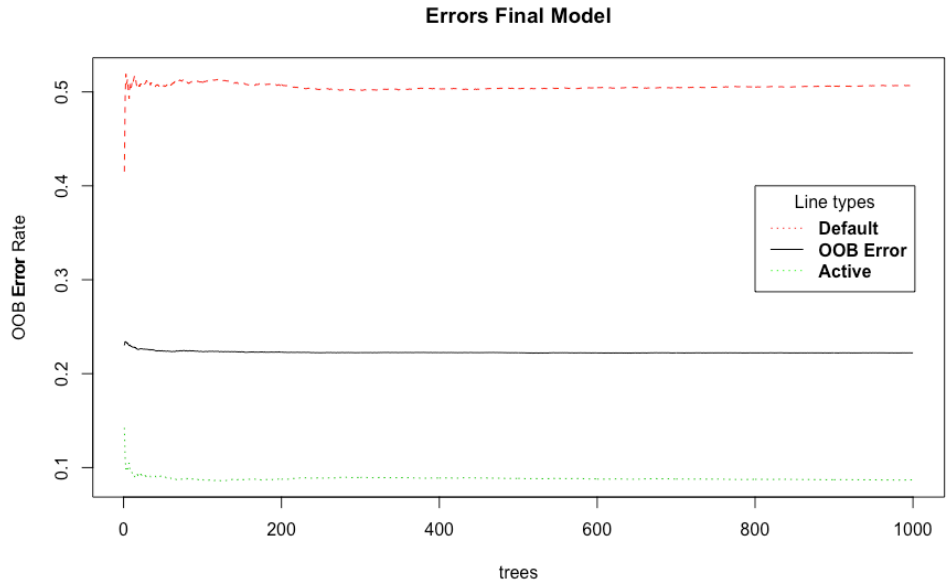
Table 3.15: Random Forest Final Results

		Value
Errors	Type I	50.02%
	Type II	8.72%
	OOB Estimate	21.96%
	Accuracy	78.13%
	NIR	66.12%
	p-value	<2.2e-16

Data Source: AIDA database

In addition to the results shown in Table 3.15, Figure 3.12 provides a visual representation of the errors.

Figure 3.12: Type 1, Type II, and OOB error of Final Model



Data Source: AIDA Database

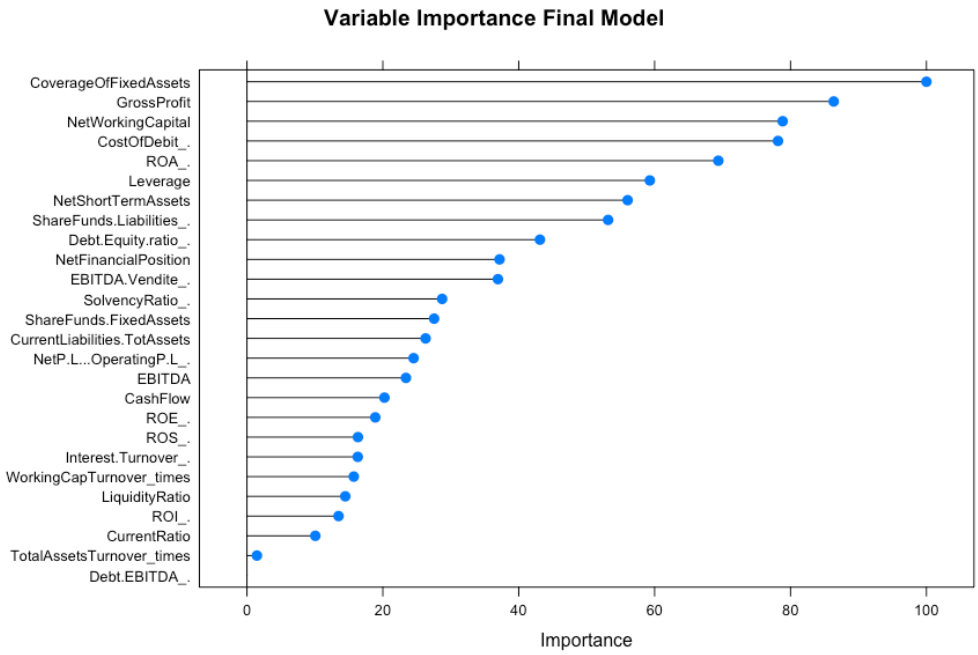
As Figure 3.12 shows, the error on defaulted predictions is much higher than the out-of-bag error and the error on active company's prediction. As described in Chapter 1, the results obtained by Breiman (2001) regarding the upper bound limit for the generalization error PE* is confirmed by Figure 3.12: by growing a forest with more than approximately 250 tress the out-of-bag error remains constant.

3.3.3. Variable importance and causality interpretation

Figure 3.13 provides useful results to compare this model with those of Table 3.13. Within the first fifteen predictors found in the previous analysis, 13 out of 15 have obtained a position in the list of the fifteen most important predictors.

Considering the totality of tests run, all predictors used in this study are attributed some level of importance in at least one test. This underlines that no variables selected was wrongly introduced in the analysis. Further analyses are required in order to confirm this assumption. Finally, Figures 3.13, and 3.7 to 3.11 show some level of consistency among the attribution of importance to predictor variables across the tests.

Figure 3.13: Variable Importance of Final Model



Data Source: AIDA database

The following list presents the variables which appear to be most important in the description of company failure and a brief description in order to observe the reason for which they are considered important:

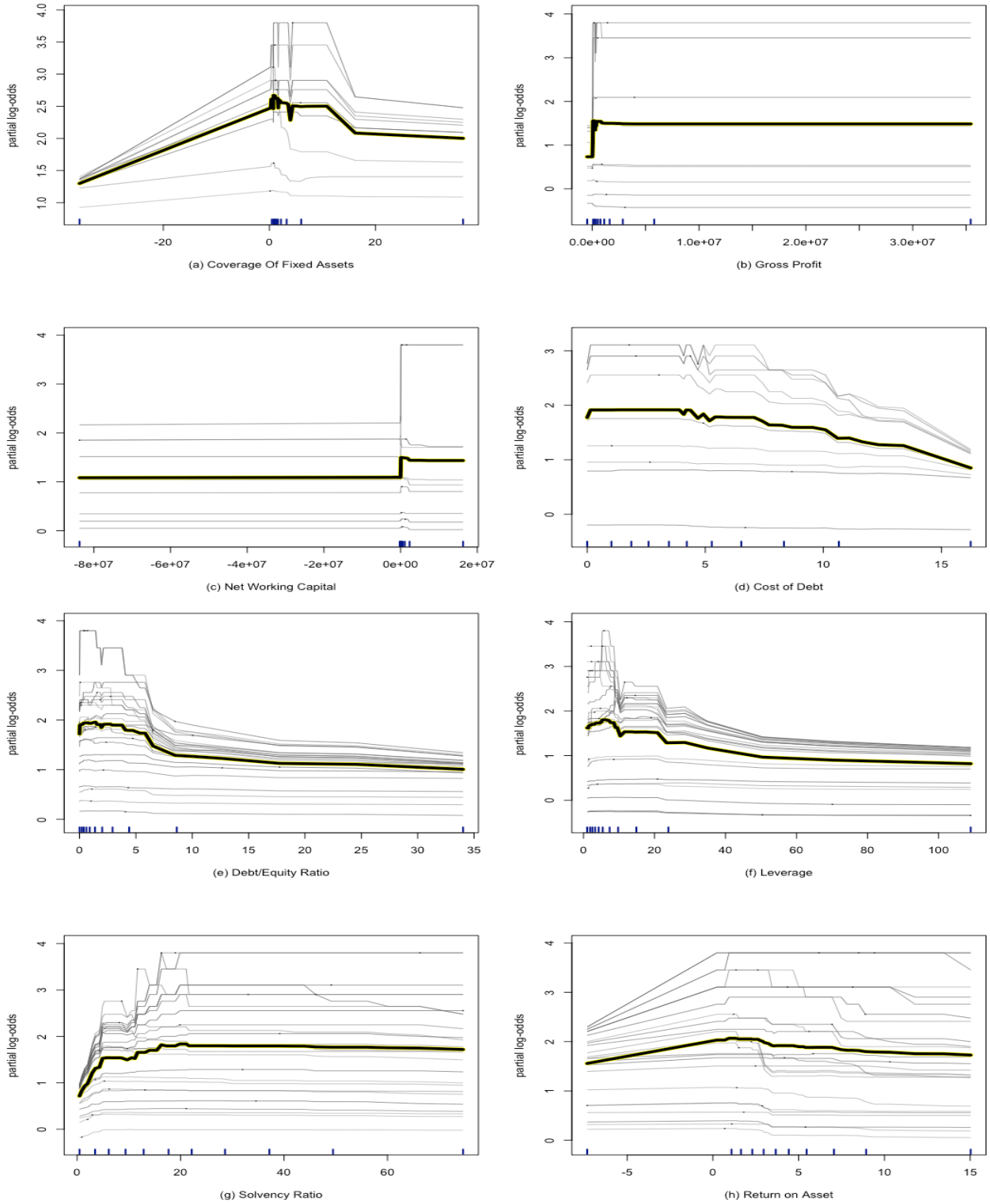
- *Coverage of Fixed Assets*: it's a measurement of a company's ability to repay its debts by selling or liquidating its assets. It is not a surprise that this variable was among the best six predictors in all five samples considered initially and in the final test. This indicator provides an indication of how risky a company is;
- *Gross Profit*: this underlines the relevance of both revenues and costs of goods sold, fundamental for a company's success;
- *Net Working Capital*: this measurement includes both current assets and current liabilities. While surprisingly Cash Flow was not considered as one of the most important variables, through net working capital we can observe that it is taken into consideration. Moreover, the same is true for the Liquidity Ratio variable. A high liquidity level manifests its potential to grow; on the other hand, a low liquidity availability leads to default;

- *Cost of Debt*: it is linked to a company's capital structure and provides an indication of a company's ability to obtain debt appropriately. Likely, the importance of this variable is connected to riskiness of funding with debt;
- *Debt to Equity Ratio*, *Leverage,* and *Solvency Ratio*: these figures are all linked to a company's financial leverage. In fact, they move together. By providing an indication of the proportion of equity and debt funding, therefore revealing a company's level of risk, given that equity funding is safer than debt funding;
- *Return on Asset*: a measure of efficiency; by observing a company' ROA, an evaluation of the firm's management can be done.

In order to further evaluate the importance of these variables listed above, the individual conditional expectation plot have been graphed (Figure 3.14). Analyzing these results, though, it should be considered that some variables are correlated (see Figure 3.1). This presents a limitation to using such approach (Friedman, 2001; Hastie and Zhao, 2019) While the correlation is significant only for some of the variables, further considerations should be done in order to obtain significant results.

Figure 3.14 provides some useful insights regarding the predictors which were most important in decreasing the impurity of the final random forest generated in this study. First, all variables have a positive impact on the model's classification of default. This, to some extent, confirms the validity of the process that led to choosing these variables. Secondly, *Coverage of Fixed Assets*, the variable which often provided the highest reduction of the Gini Index, appears to have the highest positive influence on the model, compared to the other variables.

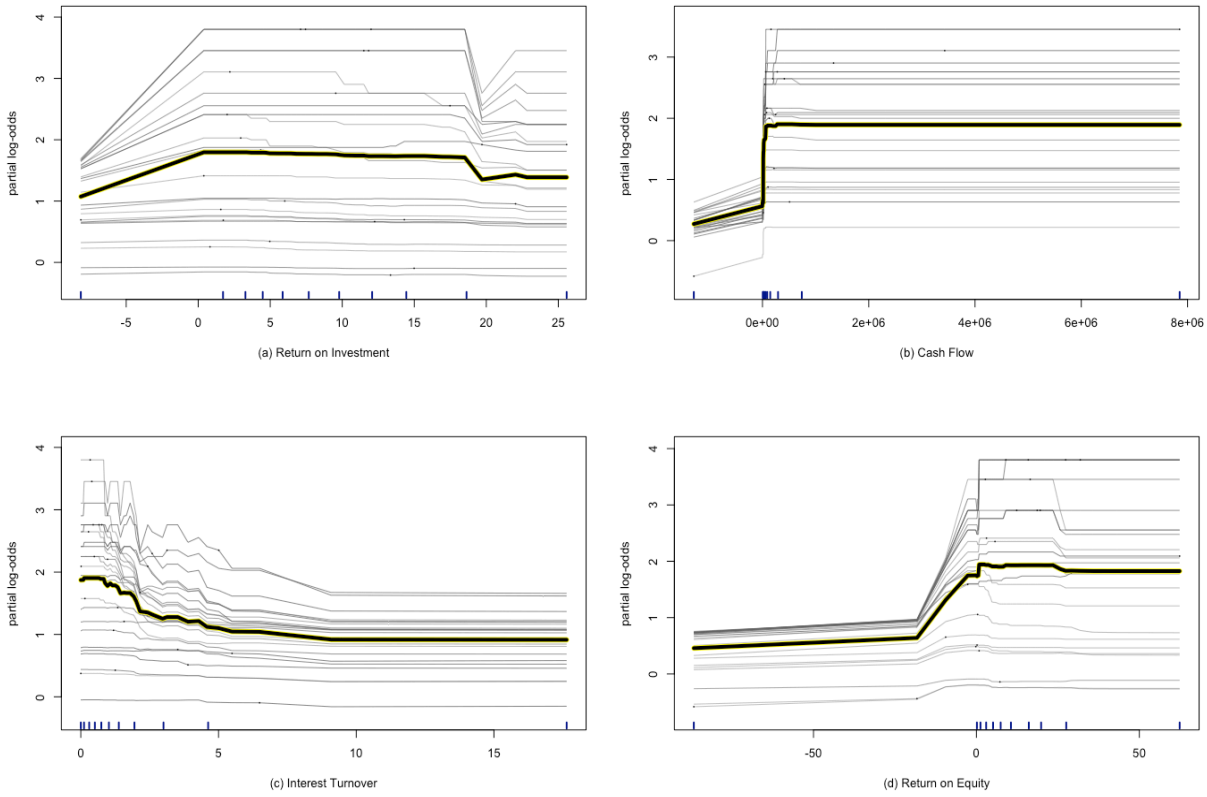
Figure 3.14: ICE plots *Coverage of Fixed Assets, Gross Profit, Net Working Capital, Cost of Debt, Debt to Equity Ratio, Leverage, Solvency Ratio,* and *Return on Asset*



Data Source: AIDA database

The always positive influence of the predictors considered in Figure 3.14 suggested to extend the causality analysis to other variables which were not deemed important by the variable importance measurements. The result provided in Figure 3.15 indicate that *Return on Investment*, *Cash Flow*, *Interest Turnover*, and *Return on Equity*, regardless of the poor importance in the reduction of the nodes impurity, have a positive influence on the model. This suggests that variable importance shown in Figures 3.7 to 3.11 and Figure 3.13 should be addressed as "relative importance". In other words, the fact that a variable may be less important in the classification process does not mean that such variable may not have a positive influence on the prediction of company failure.

Figure 3.15: ICE plot *Return on Investment*, *Cash Flow*, *Interest Turnover*, and *Return on Equity*



Data Source: AIDA database

Figure 3.15 (a) and (b) provide useful information regarding the relationship between defaulted companies and *Cash Flow* and *Return on Equity*, respectively. Specifically, it is

shown the difference between positive and negative values. The steep increase in Figure 3.15 (a), likely, is linked to the assumption that the increases in *Cash Flow* permit easier classification of Active companies. As a reminder, an increase in the y-axis is not equal to an increase in the classification accuracy, rather it shows the change in partial influence that a variable has on the model. *Return on Equity* has the same behavior. Similar relationship is shown by *Gross Profit* (Figure 3.14 (b)) and *Net Working Capital* (3.14 (c)). On the other hand, *Interest Turnover* in Figure 3.15 (c) manifests a decreasing influence in the model as the value of x increases.

3.4. COMPARISON: DECISION TREE

This section will provide a brief application of an alternative methods Decision trees are closely related to random forests, but often incur in overfitting issues. As for random forests, the software used to run the algorithm is R. In order to compare the different algorithms' performances, the imputed datasets used for the random forest analysis have been used for decision trees as well.

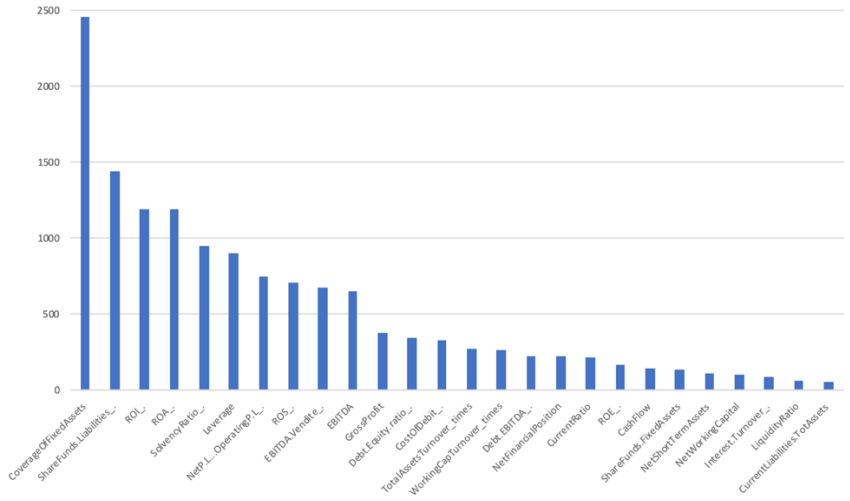
3.4.1. Decision Tree

As described in the first chapter, decision trees and random forests are very similar. Specifically, decision trees are the foundation of random forests. With its algorithm, Breiman (2001) was able to use decision trees to implement a model which does not overfit, and, therefore, can be used in more generalized cases. The reason for creation of random forest was mainly provided by the fact than decision trees tend to overfit. By focusing on the data and their patters, trees memorize the data and obtain good performances only on that dataset. This is mainly linked to the depth of trees.

Figure 3.17 plots a tree which was generated, trained, and tested using Sample 1. Such tree has been pruned in order to find the best situation possible, meaning the tree model resulted is neither too simple (with very few splits, resembling the model shown in Figure 1.6 (a) in Chapter 1) and a tree which would certainly overfit (see Figure 1.6 (b)). In order to do so, at first, the tree was fully grown. Subsequently, following the theory presented in Chapter 1, the complexity parameter α ha been computed and used for pruning. Figure 1.17

is provided since it permits to observe the different splits and, together with Figure 3.16, draw conclusions regarding variable importance, by comparing it with the variable importance shown in Figure 3.7.

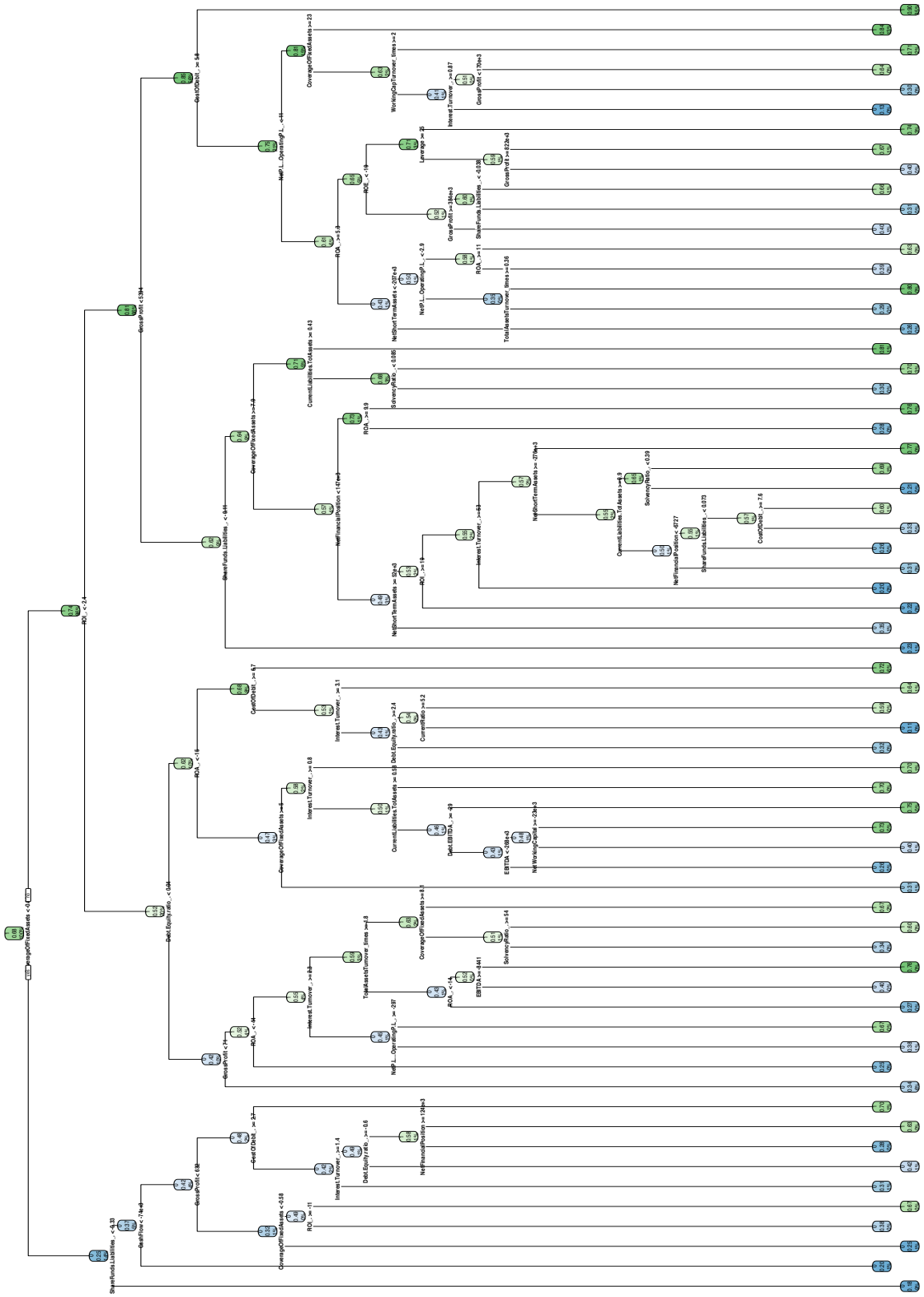
Figure 3.16: Variable importance, decision tree



Data Source: AIDA database

As Figure 3.17 shows, the root node has been split by using *Coverage of Fixed Assets*. Moreover, similarly to Figure 3.7, *Coverage of Fixed Assets* accounts for a great reduction in the data impurity. Followingly, the difference in importance between the remaining variables is not as strong as *Coverage of Fixed Assets*. Regardless of the similarities, there are several differences between the results shown in Figure 3.16 and those of Figure 3.7. In fact, beyond the first two variables in order of importance, the importance of the other predictor variables is measured equally by the two algorithms.

Figure 3.17: Decision Tree on Sample 1



Data Source: AIDA Database

Finally, the model was tested on the testing set.

Table 3.16: Decision Tree on Sample 1

		Predicted	
		Default	Active
Actual	Default	3123	1187
	Active	2157	10176

Data Source: AIDA database

Table 3.16 reports the confusion matrix generated by testing the decision tree on the testing dataset put aside in Sample 1. Formula (1.14) and (1.15) in Chapter 1 permit to compute the accuracy and the error rate of the model. The accuracy of this first test amounts to 79.91% and the error rate 20.09%. These results, as expected, are in line, and slightly better, with those of Table 3.15. Nevertheless, further tests were run similarly to the analysis done with the final model of random forests (see *Training and testing of final model*). The pruned decision tree has been used with unseen data. The data set used were Sample 2 and Sample 3. The results reported in Tables 3.17 and 3.18 show the difficulty encountered by the model. The reason for the decrease in performance is, assumedly, due to the complexity of the company default scenario.

Table 3.17: Decision tree on Sample 2

		Predicted			
		Default	Active	Accuracy	Error Rate
Actual	Default	10174	7674	72.39%	27.61%
	Active	7528	29678		

Data Source: AIDA database

Table 3.18: Decision tree on Sample 3

		Predicted			
		Default	Active	Accuracy	Error Rate
Actual	Default	1037	7537	67.15%	32.85%
	Active	7755	30221		

Data Source: AIDAdDatabase

The accuracy decreases significantly and the error increase for both tests. This should not lead to the conclusion that the decision tree algorithm does not perform well, but it underlines the difficulty of predicting company default with only one tree.

3.5. R CODE: A REPRODUCIBLE FUNCTION

Once the analysis has been concluded, the single codes used have been integrated in one function. The aim of this is to provide the reader with a function which permits to reproduce such analysis. Moreover, this function has been written in order to contribute to the framework in the literature regarding the application of machine learning algorithms and, specifically, random forests. By continuously improving such function, then, it would be possible to compare several machine learning algorithms with one command, and on a great variety of research questions and study fields.

The function, named `RandomForestAnalysisDT()`, can be found at the following link.

https://drive.google.com/file/d/1m80ZESGlIxKHJwF04vDwrS2VGWQPqWg1/view?usp=sharing

Specifically, the function in R, once the code is launched, can be used as

```
RandomForestAnalysisDT(dataset, classvariable)
```

Notes:

- The variables in dataset must correspond to the columns and should be expessed as numeric or factors;
- The response variable (classvariable) should be a factor;
- classvariable must be a character object. Example: suppose the variable to classify is Species, then: classvariable = "Species".

Simply, this function will:

- Impute missing data by predictions through random forests (uses missForest());
- Create random forest with training and testing dataset respectively

  (70% , 30%) , (80% , 20%) , (90% , 10%)

  and it will tune each random forest;
- Create a Decision Tree model;
- Create an AdaBoost model;
- Create a shiny app presenting a useful dashboard to visualize and compare results.

Moreover, all results (e.g. tuning parameters) and plots[11] are shown. Further information regarding the function's functioning and the R packages required to run the function are provided in the introduction to the function.

Finally, by considering the `iris` example introduced in Chapter 1, the function has been used to present a visualization of the output once the function is used. In order to fully use the features of the function, missing values are introduced in the data set.

```
irisNA <- prodNA(iris, noNA = 0.20) # insert randomly NA in iris
aRandomForestAnalysisDT(dataset= irisNA, classvariable= "Species")
```

Once the computations are concluded, a new page (i.e. a shiny app) is generated showing the results obtained. The page will contain six different sections:

1) Data Summary & Preparation;
2) Random Forest 70% train and 30% test;
3) Random Forest 80% train and 20% test;
4) Random Forest 90% train and 10% test;
5) Decision Tree (80%/20%);
6) AdaBoost.

[11] By choosing the directory, all plots will be saved and named in the directory chosen.

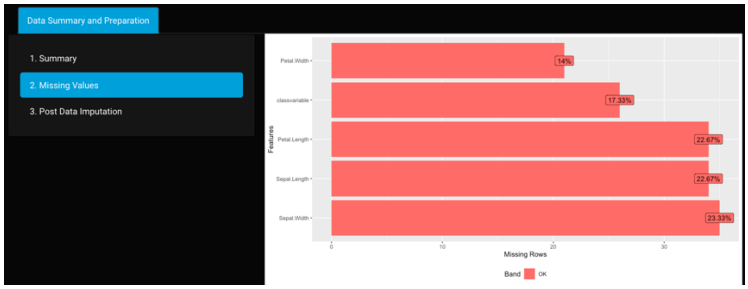
While the decision tree and AdaBoost's section provide an introduction the two approaches, the main use of visualizing through a shiny app is that it permits to compare different methods easily. Consider, for instance, it is needed to compare the error plot across the three random forest constructed with different training and testing sizes. Figure 3.18 provides a screen image of the output generated.

Figure 3.18: function's output

Data Source: `iris` data set available in the R library "datasets"

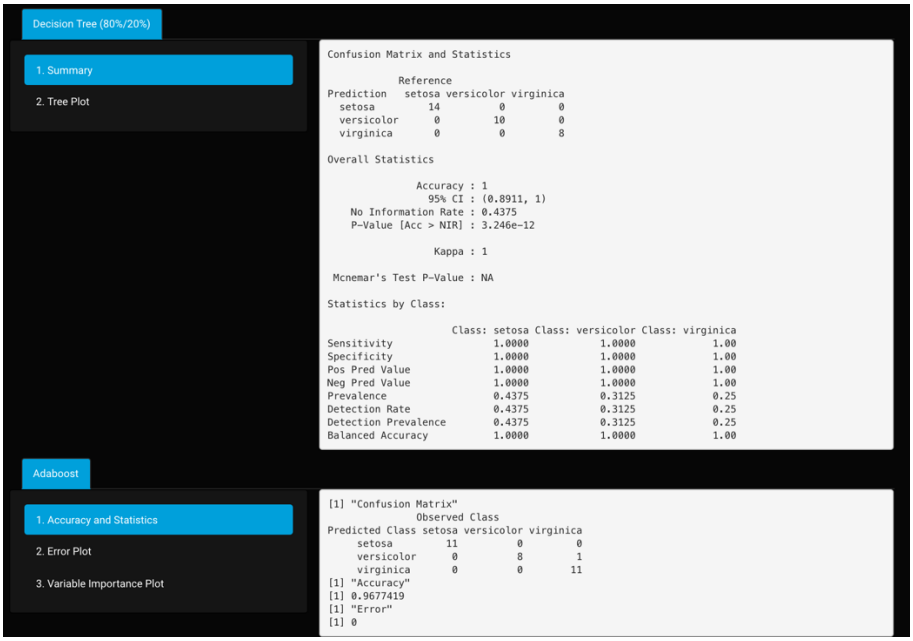
As the menu list in Figure 3.18 shows, the random forest models are thoroughly analyzed, and each result is printed in the shiny app. Other examples are shown in Figure 3.19 and 3.20.

Figure 3.19: irisNA missing values



Data Source: `iris` data set available in the R library "datasets"

3.20: Main statistics, decision trees and AdaBoost



Data Source: `iris` data set available in the R library "datasets"

The possibilities for improving such function are numerous. Possibly, by expanding the number of methods and by further developing the ones already included, `aRandomForestAnalysisDT()`can provide a wide and detailed analysis into one single function.

This page is intentionally left blank.

CONCLUSIONS AND FINDINGS

4.1. CONCLUSIONS

4.1.1. Discussion

This research has been conducted in order to analyze the topic of credit risk and company default through an application of machine learning algorithms. Specifically, this analysis focused on firm failure classification and prediction through financial indicators (i.e. ratios). The reasons for this study are numerous and many research questions which were addressed. The main objective of this paper has been to construct a random forest model to verify its ability to predict whether a company failed or is still active. Within this area of study, attention has been placed on the importance of predicting company failure for business, investors, and governments.

Given that random forests require proper tuning in order to perform well and given that machine learning methods can perform differently across various fields, the way this study has been structured provides to cover as many situations as possible. As provided in chapter 3, the analysis has been divided in different steps, which vary for size and type of data considered and for tuning parameters. By doing so, a wider array of scenarios was analyzed. Once the data had been downloaded, the dataset required preparation and cleaning. Both companies and predictors which presented too many missing data were not considered. Followingly, the remaining missing values have been imputed by predicting the data through random forests. This provided for the main limitation of this study, increasing the out-of-bag error. Regardless of the latter, the results are promising.

One of the main achievements of this study is linked to the stability of the application of random forests with the data. As the results in chapter 3 present, the model shows high level of consistency with the data. This can be noted not only in terms of accuracy and performance on the validation test, but also by the fact that the accuracy on the validation test (i.e. accuracy on testing phase) out-of-bag score obtained (i.e. accuracy on training phase). In the final model construction, an accuracy of 78.13% has been obtained, which is 12.01 percentage points above the no information rate, with a 21.96% out-of-bag error estimate (see Table 3.16). Likely, given how well random forest operate in terms of consistency, the accuracy of the final model could be increased. Nevertheless, given

the complexity of the topic, and given the fact that a company's default is linked to many factors, achieving an accuracy that ranges between 77.85% and 80.03% (see Table 3.9) is a positive result. Also, it should be noted that this outcome is given solely by internal financial indicators. It is reasonable that a certain level of error is present given the impact of external factor (e.g. a recession) and of events which the model cannot consider (e.g. wrong business decisions).
Overall all predictors used in this study are attributed some level of importance. This underlines that no variables selected was wrongly introduced in the analysis. The variables which appear to be most important in the description of company failure are *Coverage of Fixed Assets, Gross Profit*, *Net Working Capital*, *Cost of Debt*, *Debt to Equity Ratio*, *Leverage,* and *Solvency Ratio*, *Return on Asset*. The fact that many of the variables which are considered to be the most important predictors are linked to assets, their utilization, and leverage, is in line with financial methods such as the BSM model and the KVM model, which quantify a company's probability of default based on assets, volatility of the assets, and leverage.

As presented in the introduction to this study, studying company default has implications for three different parties. In fact, it can be noted that the variables listed above cover some fundamentals which should be considered in developing a firm strategy and organization. These are: profitability and potential to grow, efficiency, risk, and financial leverage. Businesses' management, financial and private equity investors, and national and local governments should, therefore, focus on these variables.

4.1.2. Limitations of the Research

Overall the random forest algorithm seems an appropriate methodology to classify company default, but some considerations are required in order to comprehend the reasons for stating this theory. In fact, some error values, mainly obtained during the training phase of the models, in this study fairly raise the concerns in favor of the opposite. Some limitations impact the analysis and therefore some clarifications are required. Through this section an explanation of the factors that influenced the research will be carried, focusing briefly, at first, on the minor ones, to then conclude with the major limitation of the study.

This study aimed to provide additional elements to evaluate the performance of machine learning in predicting and classifying company default. Moreover, random forests, and the approach used in this research, permit the comparison with methods such as Beaver (1967) or Altman's (1968). Nevertheless, the initial development of the model aimed to consider both internal and external factors influencing the company's lifecycle (i.e. macroeconomic variables). Given that very few companies have values reported regarding employment, added values per employer, and similar, the research is limited to internal factors (i.e. financial ratios).

It would be interesting to seek for causal relationship between the predictor independent variables and the dependent classes *Active* and *Defaulted*. This, though, is not possible with classification random forests since they only quantify correlation relationships.

Finally, a higher computational power would have permitted to increase the correctness of the research and to reduce the computational cost. Specifically, greater datasets, therefore carrying more information, could have been analyzed. Nevertheless, in terms of computing time, this should be addressed as a complication rather than a limitation.

Given the error of the imputation of missing data, it is reasonable to assume that the main limitation of this study is given by the nature of the data. The latter, in fact, has played a great role in the results obtained, impacting the study both directly and indirectly. Not being able to use `randomForest` or `caret`'s functions if missing value were present, an imputation approach has been used to complete the dataset. Imputation of missing data is an appropriate method since it permits to carry on the analysis, but it does not increase the information present in the data, therefore it does not lead to overestimation of positive results. On the other hand, though, the high imputation error may have an effect on the results. Given that the purpose of this study is to test random forest's predicting capability in relation to company default, the random forest algorithm was also used for the imputation of data, by creating, on each sample, random forests which predicted the missing values. As provided, the error generated by this step is high. Likely, the complexity of the data considered in this study impacted this process, increasing the difficulty of identifying patterns among the data. Given each company has its own structure, history, and characteristics, among others, one assumption may be that the imputing missing data, with such data, is a complex task. Moreover, by observing the data, it was noted that most missing data

recorded are attributable to defaulted companies. Therefore, likely, a high error in the imputation of missing data followingly increased the errors in predicting defaulted companies.
Likely, in this case, random forests are not the best choice for imputing the data; therefore, trying alternative imputation methods may help improve the results obtained. This may also be linked to the fact that the missing values are not randomly distributed across the dataset. Some variables, in fact, account for most of the missing data.

Another limitation, which is still linked to the nature of the data, is the way which company build their financial statements and report their figures. By observing the data, some companies reported zeros for some variables. In this moment, and given the high number of observations and variables, it cannot be determined whether those numbers carry information regarding the company or should instead be considered as missing values. For this reason, no action has been taken and the zeros were not substituted. While this may have an impact on the analysis, likely the great number of companies included in the dataset outsets such limitation.

4.2. RESEARCH DEVELOPMENTS

When dealing with complex and such widely spread topics difficulties may arise in finding the focus on the research. On the other hand, this provides opportunities to expand and continuously build on the study. This research is innovative in two ways: firstly, it focuses on machine learning and the random forest algorithms, methodologies which are increasingly being used since they provide a good asset to deal with big data and complex scenarios; secondly, it applies such approaches buy modernizing models on credit risk and company default which were often static and challenging to use.

The main development would be to replicate the study on new data or by applying different methods for the imputation of missing values. This would provide a clarification regarding the errors and limitations encountered in this study, assessing whether the errors are due to the data, to the imputation process, or other. Moreover, two extensions could be added replicative studies. Firstly, the replication of this study could include tests with other machine learning methods and classifiers, such as eXtreme Gradient Boosting, Neural Network, Linear Discriminant Analysis,

and glmnet, among others. Secondly, a test including external factors as predictors could try to confirm the theory that the accuracy achieved in this study can be increased.

As presented in the second chapter, two main groups have been identified regarding methods for identifying credit risk and company default. The first group (e.g. Altman Z-score) assumes a company-based approach, focusing on company figures and macroeconomic factors to predict company default. The second group (e.g. Black-Scholes-Merton model, KVM model), instead, focus on the financial character of firms and markets. In fact, they assume credit default as impacted by the value of assets, the volatility of such assets, and liabilities. The possibility or not to compare the models of the two groups goes beyond this research. Nevertheless, the elements provided by both groups could be used in the development of random forests. As conducted in this study, the random forest has been built on the guidelines provided by the first group. In fact, the variables used as predictors are in line with those used in studies by Breiman (1967), Altman (1968), Edmister (1972), Ciampi and Gordini (2008).
Connecting to the second group, the chance to quantify the probability of default and find causal relationship arises. Therefore, a study in which random forests and methods such as the KVM model are used in parallel could provide interesting results.

Finally, a relevant development would be to introduce the concept of time in the analysis. Firstly, as shown by papers such as Beaver's (1967), quantifying the risk of default across a certain range of years in the future could provide crucial information for a company's management. Secondly, the analysis could be tried with time series. Random forests, though, view predictor variables as independent and do not distinguish the various years. Likely, this would improve the research, providing more information within the data for the training process.

DATA SOURCES

AIDA: Access provided by Ca' Foscari University of Venice. Database available online at https://www.bvdinfo.com/en-gb/our-products/data/national/aida

REFERENCES

[1] Adam, S., Bernard, S. and Heutte, L. (2010) *A Study of Strenght and Correlation in Random Forests*. International Conference on Intelligent Computing, August 2010, 93(11) p.186-191. [online] Available at: https://www.researchgate.net/publication/220776168_A_Study_of_Strength_and_Correlation_in_Random_Forests [Downloaded March 27th, 2020]

[2] Altman, E.I. (1968) Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. The Journal of Finance, Wiley, 23(4), p.589-609. [online] Available at: https://www.jstor.org/stable/pdf/2978933.pdf?refreqid=excelsior%3Ab232540f026689ed7945fa0ab00de413 [Downloaded April 4th, 2020]

[3] Amit, Y and Geman, D. (1996) Shape quantization and recognition with randomized trees. Neural Computation 9, n.7, p. 1545-1588. [online] Available at: http://www.cis.jhu.edu/publications/papers_in_database/GEMAN/shape.pdf [Downloaded March 2nd, 2020]

[4] Arie, B.D. (2008) *Comparison of Classification Accuracy Using Cohen's Weighted Kappa*, Expert Systems with Applications 34, 825-832. [online] Available at: https://www.sciencedirect.com/science/article/pii/S0957417406003435 [Downloaded March 21st, 2020]

[5] Armitage, A., Williams, N. and Zander, S. (2006) *A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification*, ACM SIGCOMM Computer Communication Review, 36(5) October [online] Available at: https://dl.acm.org/doi/abs/10.1145/1163593.1163596 [Downloaded March 11th, 2020]

[6] Balcaen, S. and Ooghe, H. (2004) Alternative Methodologies in Studies on Business Failure: Do They Produce Better Results Than the Claasical Statistics Methods?, Vlerick Leuven Gent Management School, Working Papers, No. 16. [online] Available at: https://pdfs.semanticscholar.org/4ed3/a64119fbc1409586e26187175d49e7db91c3.pdf?_ga=2.249487175.1152388759.1586037390-2003424513.1586037390 [Downloaded April 4th, 2020]

[7] Balcaen, S. and Ooghe, H. (2006) 35 Years of Studies on Business Failure: An Overview of the Classical Statistical Methodologies and Their Related Problems. British Accounting Review, 38(1), p.63-93. [online] Available at: https://pdfs.semanticscholar.org/4c19/705cff7bd30815864f12ee10e9be2c3ca1b9.pdf [Downloaded April 4th, 2020]

[8] Bandyopadhyay, A. (2007) Mapping Corporate Drift Towards Default. The Journal of Risk Finance, 8(1), p.35-45. [online] Available at: https://www.emerald.com/insight/content/doi/10.1108/15265940710721064/full/pdf?title=mapping-corporate-drift-towards-default-part-1-a-marketbased-approach [Downloaded April 1st, 2020]

[9] Beaver, W.H. (1967) *Financial ratios predictors of failure*. Journal of Accounting Research, Vol.4, p.71-111 [online] Available at: https://www.jstor.org/stable/pdf/2490171.pdf?refreqid=excelsior%3A4bfb1df97a8e3da309a85e76f6b0ad60 [Downloaded April 4th, 2020]

[10] Biau, O. and D'Elia, A. (2010) *Euro Area GDP Forecasting Using Large Survey Dataset. A Random Forest Approach*. Eurostat European Commission, 6th Eurostat Colloquium on Modern Tools for Business Cycle Analysis: the Lessons From Global Economic Crisis, September 2010 [online] Available at: https://ideas.repec.org/p/ekd/002596/259600029.html [Downloaded February 19th, 2020]

[11] Biau, G. and Scornet, E. (2010) *A Random Forest Guided Tour*. Mathematics Subject Classification. [online] Available at: https://arxiv.org/pdf/1511.05741.pdf [Downloaded March 15th, 2020]

[12] Breiman (1996) Bagging Predictors. Ross Quinlan. [online] Available at: https://link.springer.com/content/pdf/10.1023/A:1018054314350.pdf [Downloaded February 20th, 2020]

[13] Breiman, L. (2001) *Random Forest*. Machine Learning 45 5-32 [online] Available at: https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf [Downloaded January 20th, 2020]

[14] Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984) Classification And Regression Trees, *Chapman & Hall/CRC*.

[15] Burgartz, T., Khare, A., Kramer, A. and Mack, O. (2016) Managing in a VUCA World, *Springer*, p. 3-17, 100

[16] Cheung-Wai Chan, J. and Paelinckx, D. (2008) Evaluation of Random Forest and Adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery. Remote sensing of Environment, 122(6). [online] Available at: https://www.sciencedirect.com/science/article/pii/S0034425708000679?via%3Dihub [Accessed March 25th, 2020]

[17] Ciampi, F. and Gordini, N. (2008) Using Economic-Financial Ratios for Small Enterprise Default Prediction Modeling: An Empirical Analysis. 2008 Oxford & Economics Conference Program. [online] Available at: https://www.researchgate.net/profile/Niccolo_Gordini/publication/281147699_Using_Economic-Financial_Ratios_for_Small_Enterprise_Default_Prediction_Modeling_An_Empirical_Analysis/links/560118d808aeafc8ac8c806d/Using-Economic-Financial-Ratios-for-Small-Enterprise-Default-Prediction-Modeling-An-Empirical-Analysis.pdf [Downloaded April 1st, 2020]

[18] Ciampi, F. and Gordini, N. (2013) Small Enterprise Default Prediction Modelling through Artificial Neural Networks: An Empirical Analysis of Italian Small Enterprises, Journal of Small Business Management 2013 51(1), pp. 23–45. [online] Available at: https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1540-627X.2012.00376.x [Downloaded April 1st, 2020]

[19] Cohen, J. (1960) *A Coefficient of Agreement for Nominal Scales*. Educational and Psychological Measurement, Vol XX, No. 1. [online] Available at: https://journals.sagepub.com/doi/pdf/10.1177/001316446002000104 [Downloaded March 21st, 2020]

[20] Cohen, J. and Fleiss, J.L. (1973) The Equivalence of Weighted Kappa and the Intraclass Correlation oefficient as Measures of Relaibility, Educational and Psychological Measurement, 33, 613-619. [online] Available at: https://journals.sagepub.com/doi/pdf/10.1177/001316447303300309 [Accessed May 10th, 2020]

[21] Crowley, M. and Ghojogh, B. (2019) The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial. arXiv. [online] Available at: https://arxiv.org/pdf/1905.12787.pdf [Accessed March 10th, 2020]

[22] Dietterich, T. (1995) *Overfitting and Undercomputing in Machine Learning*. ACM Computing Surveys, September. [online] Available at: https://dl.acm.org/doi/10.1145/212094.212114 [Downloaded February 19th, 2020]

[23] Dietterich, T. (1999) An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. Machine Learning, 1-22. [online] Available at: http://web.engr.oregonstate.edu/~tgd/publications/mlj-randomized-c4.pdf [Accessed March 26th, 2020]

[24] Dietterich (2000) *Ensemble Methods in Machine Learning*. Springer, 1-15. [online] Available at: https://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf [Accessed March 26th, 2020]

[25] Dokuchaev, N. and Luong, C. (2018) *Forecasting of Realised Volatility with the Random Forest Algorithm*. J. Risk Financial Manag, 11(4), 61 [online] Available at: https://www.mdpi.com/1911-8074/11/4/61 [Downloaded February 19th, 2020]

[26] Edmister, R.O. (1972) An Empirical Test of Financial Ratio Analysis for Small Business Failure Prediction, The Journal of Financial and Quantitative Analysis, Cambridge University Press, 7(2), pp. 1477-1493. [online] Available at: https://www.jstor.org/stable/pdf/2329929.pdf?refreqid=excelsior%3Acc98e37e485c033e123844176a8deb55 [Downloaded April 4th, 2020]

[27] Fleiss, J.L., Levin, B. and Paik, M.C. (1976) Statistical Methods for Rates and Proportions, Third Edition, Wiley. p. 608-609. [online] Available at: https://pdfs.semanticscholar.org/efad/747e8231b5bb930b97fb1b3eb02eb89169bd.pdf?_ga=2.153201584.940612771.1589279324-895608351.1588941941 [Accessed May 10th, 2020]

[28] Freund, Y. and Schapire, R.E. (1995) *A decision-theoretic generalization of on-line learning and an application to boosting*. Journal of Computer and System Sciences, 55(1):119-139, March 1997. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=4BF3325D8222B3234BB95971FCAD8759?doi=10.1.1.56.9855&rep=rep1&type=pdf Accessed March 26th, 2020]

[29] Freund, Y. and Schapire, R.E. (1996) *Experiments with a new boosting algorithm.* AT&T Research. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.6252&rep=rep1&type=pdf [Accessed March 26th, 2020]

[30] Freund, Y. and Schapire, R.E. (1999) *A Short Introduction to Boosting*. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September 1999. [online] Available at: https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf [Accessed March 11th, 2020]

[31] Fiedman, J.H. (2001) Greedy function approximation: a gradient boosting machine. The Annals of Statistics, vol.29, n.5, p.1189-1232. [online] Available at: https://www.jstor.org/stable/pdf/2699986.pdf?refreqid=excelsior%3Aa3150146e539a339503f93a4aabbd08f [Accessed May 14th, 2020]

[32] Geurts, P. (2002) Contributions to Decision Tree Induction: Bias/Variance Tradeoff and Time Serie Classification. Ph.D. thesis, University of Liège. [online] Available at: https://orbi.uliege.be/bitstream/2268/25737/1/geurts-phd2002.pdf [Accessed April 21th, 2020]

[33] Gregorutti, B. and Saint-Pierre, P. (2013) Correlation and variable importance in random forest. Statistics and Computing. [online] Available at: https://www.researchgate.net/publication/258083104_Correlation_and_variable_importance_in_random_forests [Downloaded May 15th, 2020]

[34] Hastie, T. and Zhao, Q. (2019) Causal interpretations of black-box models. Journal of Business & Economic Statistics. [online] Available at: https://web.stanford.edu/~hastie/Papers/pdp_zhao.pdf [Downloaded April 7th, 2020]

[35] Hazak, A. and Mannasoo, K. (2007) Indicators of Corporate Default - EU-based Empirical Study, Transformations in Business and Economics. [online] Available at: https://www.researchgate.net/profile/Aaro_Hazak/publication/4739160_Indicators_of_corporate_default_-_EU-based_empirical_study/links/00b4952a8eb79f33d3000000.pdf [Downloaded April 1st, 2020]

[36] Ho, T.K. (1995) Random decision forest. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282. [online] Available at: https://www4.stat.ncsu.edu/~lu/ST7901/reading%20materials/Ho1995.pdf [Downloaded March 2nd, 2020]

[37] Hornung, R. and Janitza, S. (2018) On the Overstimation of Random Forest's Out-Of-Bag Error, Y-h. Taguchi, PloS One, 13(8). [online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6078316/ [Accessed April 21st, 2020]

[38] Kealhofer, S. (2003) Quantifying Credit Risk I: Default Prediction, Financial Analyst Journal, 59(1), p. 30-44. [online] Available at: https://www.tandfonline.com/doi/pdf/10.2469/faj.v59.n1.2501?needAccess=true [Downloaded April 1st, 2020]

[39] Kelly, T.K. (2007) Ockham's Razor, Empirical Complexity, and Truth-Finding Efficiency, Theoretical Computer Science 383, 270-289. [online] Available at: https://reader.elsevier.com/reader/sd/pii/S0304397507003222?token=27E31F81B3BF3D5C59927641ECE1AE1582AA78EAF2DB10BAC7778747C4D02AF9D4E199ADBEF22DB7C634BAB9E91CE446 [Accessed April 13th, 2020]

[40] Kim, J.H. (2009) Estimating classification error rate: repeated cross-validation, repeated hold-out and bootstrap. Computational Statistics & Data Analysis, vol 53, n 11, p. 3735-3745. [online] Available at: https://www.sciencedirect.com/science/article/abs/pii/S0167947309001601 [Accessed May 14th, 2020]

[41] Kliestik, T., Kocisova , K. and Misankova, M. (2014) Calculation of Distance to Default. Procedia Economics and Finance, 23, p.238-243. [online] Available at: https://www.researchgate.net/profile/Tomas_Kliestik/publication/282555079_Calculation_of_Distance_to_Default/links/5660180f08aeafc2aaca9fee/Calculation-of-Distance-to-Default.pdf [Downloaded April 1st, 2020]

[42] Loh, W.Y. (2011) Classification and Regression Trees, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1 (1): 14-23. [online] Available at: https://www.researchgate.net/publication/227658748_Classification_and_Regression_Trees [Downloaded May 10th, 2020]

[43] Maimon, O. and Rokach, L. (2005) Decision Trees, in The Data Mining and Knowledge Discovery Handbook, *Springer*, ch. 9. [online] Available at: https://www.researchgate.net/publication/225237661_Decision_Trees [Downloaded February 10th 2020]

[44] Moisen, G.G. (2008) *Classification and Regression Trees*. Elsevier B.V. [online] Available at: https://www.fs.fed.us/rm/pubs_other/rmrs_2008_moisen_g001.pdf [Downloaded March 4th, 2020]

[45] Mukherjee, A., Singal, A., Waljee, A. and Zhang, Y. (2013) Comparison of Imputation Methods for Missing Laboratory Data in Medicine, BMJ Open, [online] Available at: https://www.researchgate.net/publication/254264046_Comparison_of_imputation_methods_for_missing_laboratory_data_in_medicine [Downloaded April 22nd, 2020]

[46] Papacharalampous, G. and Tyralis, H. (2017) *Variable Selection in Time Series Forecasting Using Random Forests*. Algorithms 10(4), 114. [online] Available at: https://www.mdpi.com/1999-4893/10/4/114/htm [Downloaded January 20th, 2020]

[47] Ralescu, A., Ramsay, B., van der Knaap, E. and Visa, S. (2011) *Confusion Matrix-based Feature Selection*, in Proceedings of the Twenty-Second Midwest Artificial Intelligence and Cognitive Science Conference, April 16-17 2011. P.120-127 [online] Available at: https://www.researchgate.net/profile/Atsushi_Inoue/publication/220833227_Page_Ranking_Refinement_Using_Fuzzy_Sets_and_Logic/links/54b743480cf24eb34f6e9e80.pdf#page=126 [Accessed March 19th, 2020]

[48] Wei, W.S.W. (2006) Time Series Analysis: Univariate and Multivariate Methods, 2nd ed., *Pearson Addison Wesley*. Available at: https://www.researchgate.net/publication/236651810_Time_Series_Analysis_Univariate_and_Multivariate_Methods_2nd_edition_2006 [Downloaded March 11th, 2020]

CODE REFERENCES

[1] For the Gini Index:
Decision Tree Flovours: Gini Index and Information Gain, *Learn by Marketing,* February 27, 2016, [online] Available at http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/

[2] For mtry, maxnodes, and ntrees:
R Random Forest Tutorial with Example, *Guru99*, February 27, 2016, [online] Available at https://www.guru99.com/r-random-forest-tutorial.html

[3] For correlation graphs:
Correlation Matrix: An R function to do all you need, *Statistical Tools for High-Throughput Data Analysis* [online] Available at: http://www.sthda.com/english/wiki/correlation-matrix-an-r-function-to-do-all-you-need

[4] For missForest information:
Stekhoven Daniel J. in *Using the* `missForest` *Package* (May, 2011) Available at: https://stat.ethz.ch/education/semesters/ss2012/ams/paper/missForest_1.2.pdf

Stekhoven D.J. and Buhlmann, P (2012) MissForest? Non-Parametric Missing Value Imputation for Mixed-Type Data. Available at https://www.researchgate.net/publication/51753782_MissForestNon-parametric_missing_value_imputation_for_mixed-type_data

BIBLIOGHRAPHY

[1] Artemenko, A.G., Kolumbin, O.G., Kuz'min, V.E., Muratov E.N., Muratov, N.N. and Polishchuk, P.G. (2009) Application of random forest approach to QSAR prediction of aquatic toxicity. J. Chem. Inf. Model., p. 2481-2488. [online] Available at: https://pubs.acs.org/doi/pdf/10.1021/ci900203n [Accessed May 15th, 2020]

[2] Black, F. and Scholes, M. (1973) The Pricing of Options and Corporate Liabilities. The Journal of Political Economy, 81(3), p.637-654. [online] Available at: https://www.cs.princeton.edu/courses/archive/fall09/cos323/papers/black_scholes73.pdf

[3] Criminisi, A., Fitzgibbon A., Glocker, Izadi, S., B., Shotton, J. and Zach, C. (2013) Scene coordinate regression forest for camera relocalization in RGB-D images. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2930-2937 [online] Available at: http://openaccess.thecvf.com/content_cvpr_2013/papers/Shotton_Scene_Coordinate_Regression_2013_CVPR_paper.pdf [Accessed May 15th, 2020]

[4] Dixon, B., Naghibi, S.A. and Pourghasemi, H.R. (2015) GIS-based groundwater potential mapping using boosted regression tree, classification and regression tree, and random forest machine learning models in Iran. Environmental Monitoring and Assessment 188, n.44 [online] Available at: https://link.springer.com/article/10.1007/s10661-015-5049-6 [Accessed May 15th, 2020]

[5] Kane, M.J., Price, N., Rabinowitz, P. and Scotch, M. (2014) Comparison of ARIMA and random forest time series prediction of avian influenza H5N1 outbreaks. BMC Bioninformatics 15, n.276. [online] Available at: https://link.springer.com/article/10.1186/1471-2105-15-276 [Accessed May 15th, 2020]

[6] Kumar, M. and Thenmozhi, M. (2006) Forecasting stock index movement: a comparison of support vector machines and random forest. Indian Institute of Capital Markets 9th Capital Markets Conference Paper. [online] Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=876544 [Accessed May 15th, 2020]

[7] Merton, R.C. (1974) On the Pricing of Corporate Debt: The Risk Structure of Interest Rates, The Journal of Finance, 29(2), p.449-470. [online] Available at: https://onlinelibrary.wiley.com/doi/10.1111/j.1540-6261.1974.tb03058.x

[8] Young, G. and Tudela, M. (2003) A Merton-Model Approach to Assessing the Default Risk of UK Public Companies. Bank of England, Working Paper n. 194. [online] Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=530642&download=yes

APPENDIX

Appendix A: *variable formulas*

Financial Indicators

$$Liquidity\ Ratio = \frac{Tot\ Current\ Asset - Tot\ Inventories}{Payables\ due\ Within\ 12\ Months}$$

$$Current\ Ratio = \frac{Tot\ Current\ Asset}{Payables\ due\ Within\ 12\ Months}$$

$$\frac{Current\ Liabilities}{Tot\ Assets} = \frac{Payables\ due\ Within\ 12\ Months}{Payables\ due\ Within\ 12\ Months + Payables\ due\ Beyond\ 12\ Months}$$

$$Leverage = \frac{Tot\ Assets}{Shareholder's\ Funds}$$

$$Coverage\ of\ Fixed\ Assets = \frac{Shareholder's\ Funds + Payables\ due\ Within\ 12\ Months}{Tot\ Fixed\ Assets}$$

$$Cost\ of\ Debt = \frac{Borrowing\ Costs}{Due\ to\ Banks + Due\ to\ Banks\ Beyond\ 12\ Months} * (-100)$$

$$\frac{Interest}{Turnover} = \frac{Borrowing\ Costs}{Revenues + other\ Revenue\ and\ Income} * (-100)$$

$$Solvency\ Ratio = \frac{Shareholder'sFunds}{Tot\ Assets} * 100$$

$$\frac{Share\ Fund}{Liabilities} = \frac{Shareholder's\ Funds}{Tot\ Assets} * 100$$

$$Net\ Financial\ Posit = Due\ to\ Banks + Due\ to\ Banks\ Beyond\ 12\ Months + Due\ to\ Other + Due\ to\ Other\ Lenders\ Beyond\ 12\ Months - Tot\ Liquid\ Funds$$

$$Debt\ to\ Equity = \frac{Due\ to\ Banks + Due\ to\ Banks\ Beyond\ 12\ Months + Due\ to\ Others + Due\ to\ Other\ Lenders\ Beyond\ 12\ Months}{Shareholder's\ Funds}$$

$$Debt/_{EBITDA} = \frac{Due\ to\ Banks + Due\ to\ Banks\ Beyond\ 12\ Months + Due\ to\ Others + Due\ to\ Other\ Lenders\ Beyond\ 12\ Months}{Operating\ Margin + Depreciation, Amortization\ and\ Writedowns}$$

$$Banks/_{Turnover} = \frac{Due\ to\ Banks + Due\ to\ Banks\ Beyond\ 12\ Months}{Revenues + other\ Revenue\ and\ Income} * 100$$

Profitability Ratios

$$EBITDA = Operating\ Margin + Depreciation, Amortization\ and\ Writedowns$$

$$EBITDA/_{Sales} = \frac{Operating\ Margin + Depreciation, Amortization\ and\ Writedowns}{Revenues + other\ Revenues\ and\ Income} * 100$$

$$Return\ on\ Assets = \frac{Operating\ Margin}{Tot\ Assets} * 100$$

$$ROI = \frac{Operating\ Margin}{\begin{array}{c}(Shareholder'sFunds + Bonds + Bonds\ Beyond\ 12\ Months + \\ due\ to\ other\ lenders + due\ to\ other\ lenders\ beyond\ 12\ months + \\ due\ to\ shareholders\ for\ loans + due\ to\ shareholders\ for\ loans\ beyond\ 12\ months\end{array}} * 100$$

$$Return\ on\ Sales = \frac{Operating\ Margin}{Revenues + other\ Revenues\ and\ Income} * 100$$

$$Return\ on\ Equity = \frac{Profit\ (Loss)}{Shareholder's\ Funds} * 100$$

$$Net\ P\&L/_{Operating\ P\&L} = \frac{Profit\ (Loss)}{Operating\ Margin} * 100$$

Other Significant Financial Indicators

$$\begin{array}{l}Net\ Working\ Capital \\ \quad = Tot\ Current\ Assets \\ \quad - Payables\ due\ within\ 12\ months\end{array}$$

$$\begin{array}{l}Gross\ Profit \\ = Revenues + changes\ in\ contract\ work\ in\ progress \\ + changes\ in\ inventories \\ - changes\ in\ inventory\ of\ raw\ and\ consumable\ materials \\ - raw, comsumable\ materials\ and\ goods\ for\ resale\end{array}$$

$$\begin{array}{l}Net\ Short\ Term\ Asset \\ \quad = Trade\ Account + Receivable\ due\ from\ others \\ \quad + tot\ liquid\ funds \\ \quad - payables\ due\ within\ 12\ months\end{array}$$

$$Share\ Fund - Fixed\ Assets = Shareholder\ Funds - Tot\ Fixed\ Assets$$

$$\begin{array}{l}Cash\ Flow = Profit\ (Loss) of\ the\ year \\ \quad + Depreciation, Amortization\ and\ Writedowns\end{array}$$

Productivity Ratios

$$Turnover\ per\ Employee = \frac{Revenues}{Number\ of\ Employees}$$

$$Added\ Value\ per\ Employee = \frac{Added\ Value}{Number\ of\ Employees}$$

Management Ratios

$$Tot\ Assets\ Turnover\ (times) = \frac{Net\ Sales}{Tot\ Assets}$$

$$Working\ Cap\ Turnover\ (times) = \frac{Net\ Sales}{Tot\ Current\ Assets}$$

Appendix B: *codes*

```
# packages installed, and libraries used throughout this research
install.packages("missForest")
devtools::install_github("collectivemedia/tictoc")
install.packages("randomForest")
install.packages("caret")
install.packages("e1071")
install.packages("ggRandomForests")
install.packages("randomForestExplainer")
install.packages("dplyr")

library(caret)
library(e1071)
library(ggplot2)
library(ggRandomForests)
library(randomForestExplainer)
library(randomForest)
library(missForest)
library(tictoc)
library(dplyr)
```

B.1 Chapter 1:

1.1. Decision trees

```
attach(iris)
# Creation of train (90%) and test (10%) dataset
iris_sample <- sample(2, nrow(iris), replace = T, prob = c(0.9, 0.1)
iris_training <- iris[iris_sample,]
iris_test <- iris[-iris_sample,]

decisionaltree_model <- rpart(Species~., iris_training, method = "class") # Create decision tree model
decisionaltree_model

# To obtain a graphical representation of the tree (Image 2.1)
rpart.plot(decisionaltree_model)

# Gini Index[12]
gini_process <- function(classes, splitvar = NULL){
  #Assumes Splitvar is a logical vector
  if (is.null(splitvar)){
    base_prob <- table(classes)/length(classes)
    return(1-sum(base_prob**2))
  }
  base_prob <- table(splitvar)/length(splitvar)
  crosstab <- table(classes, splitvar)
  crossprob <- prop.table(crosstab, 2)
  No_Node_Gini <- 1-sum(crossprob[,1]**2)
  Yes_Node_Gini <- 1-sum(crossprob[,2]**2)
  return(sum(base_prob * c(No_Node_Gini,Y es_Node_Gini)))
}

gini_process(iris$Species) # Obtain values of the Gini index
gini_process(iris$Species, iris$Petal.Length < 2.5)
gini_process(iris$Species, iris$Petal.Width < 1.8)

Testing Predictions
pred <- predict(decisionaltree_model, iris_test, type = "class")
```

[12] The following code has been provided by the publication on Learn by Marketing on February 27, 2016, available at http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/

```
table(iris_test, pred) # provides output

# predictions with training 70% and testing 30%
iris_sample <- sample(2, nrow(iris), replace = T, prob = c(0.7, 0.3)
iris_training <- iris[iris_sample,]
iris_test <- iris[-iris_sample,]

decisionaltree_model <- rpart(Species~., iris_training, method = "class")

new_prediction <- predict(decisionaltree_model, iris_test, type = "class")
table(iris_test, new_prediction)
```

1.2 Random forests

```
# use ?randomForest for specifics
# Creation and OOB error
ind <- sample(2, nrow(iris), replace = T, prob = c(0.9,0.1))
trainingDataset <- iris[ind==1,]
testingDataset <- iris[ind==2,]

iris_RFModel <- randomForest(Species ~., data = trainingDataset, ntree = 100,
                             proximity = T, importance = T)

# to obtian a grapgh of the error (Graph 1.1)
plot(gg_error(iris_RFModel))

# Variable Importance, Graph 1.2
varImpPlot(iris_RFModel)

# Tree Depth
MinimalDepthFrame <- min_depth_distribution(iris_RFModel)
plot_min_depth_distribution(MinimalDepthFrame, mean_sample = "relevant_trees") # Graph 1.3

# Testing with RF (with caret package)
# Training with parameters tuning
trControl <- trainControl(method = "cv", number = 10, search = "grid")
tuneGrid <- expand.grid(.mtry = 3)
iris_FinalRF <- train(Species~.,
                      trainingDataset,
                      method = "rf",
                      metric = "Accuracy",
                      tuneGrid = tuneGrid,
                      trControl = trControl,
                      importance = T,
                      nodesize = NULL,
                      ntree = 1000,
                      maxnodes = NULL)

# Predictions and Confusion Matrix
prediction <- predict(iris_FinalRF, newdata = testingDataset)
confusionMatrix(prediction, testingDataset$Species)

# Repeat for Example with training 70% and testing 30%
```

B.2. Chapter 3

2.1 Step 1: preparation of data and samples

```
attach(CompanyData)

# Transformation of legal status
CompanyData$LegalStatus[CompanyData$LegalStatus == "Active" ] <- 1
CompanyData$LegalStatus[CompanyData$LegalStatus == "Dissolved (merger)" ] <- 1
```

```
CompanyData$LegalStatus[CompanyData$LegalStatus == "Active (default of payments)" ] <- 1
CompanyData$LegalStatus[CompanyData$LegalStatus == "Active (receivership)" ] <- 1
CompanyData$LegalStatus[CompanyData$LegalStatus == "Dissolved (demerger)" ] <- 0
CompanyData$LegalStatus[CompanyData$LegalStatus == "In liquidation" ] <- 0
CompanyData$LegalStatus[CompanyData$LegalStatus == "Bankruptcy" ] <- 0
CompanyData$LegalStatus[CompanyData$LegalStatus == "Dissolved (bankruptcy)" ] <- 0
CompanyData$LegalStatus[CompanyData$LegalStatus == "Dissolved" ] <- 0
CompanyData$LegalStatus[CompanyData$LegalStatus == "Dissolved (liquidation)" ] <- 0

# Conversion to NA: repeat for all variables
CompanyData$LegalStatus[CompanyData$LegalStatus == "n.s." ] <- NA
CompanyData$LegalStatus[CompanyData$LegalStatus == "n.a." ] <- NA

CompanyData$LiquidityRatio[CompanyData$LiquidityRatio == "n.s." ] <- NA
CompanyData$LiquidityRatio[CompanyData$LiquidityRatio == "n.a." ] <- NA

CompanyData$CurrentRatio[CompanyData$CurrentRatio == "n.s." ] <- NA
CompanyData$CurrentRatio[CompanyData$CurrentRatio == "n.a." ] <- NA

CompanyData$`CurrentLiabilities/TotAssets`[CompanyData$`CurrentLiabilities/TotAssets` == "n.s." ] <- NA
CompanyData$`CurrentLiabilities/TotAssets`[CompanyData$`CurrentLiabilities/TotAssets` == "n.a." ] <- NA
# …

# Transformation of variables to numeric, or factor as needed
sapply(NewCompanyData, class) # check class
NewCompanyData <- transform(
    NewCompanyData,
    LegalStatus = as.factor(LegalStatus),
    LiquidityRatio = as.numeric(LiquidityRatio),
    CurrentRatio = as.numeric(CurrentRatio),
    `CurrentLiabilities/TotAssets` = as.numeric(`CurrentLiabilities/TotAssets`),
    Leverage = as.numeric(Leverage),
    CoverageOfFixedAssets = as.numeric(CoverageOfFixedAssets),
    `Banks/Turnover_%` = as.numeric(`Banks/Turnover_%`),
    `CostOfDebit_%` = as.numeric(`CostOfDebit_%`),
    `Interest/Turnover_%` = as.numeric(`Interest/Turnover_%`),
    `SolvencyRatio_%` = as.numeric(`SolvencyRatio_%`),
    `ShareFunds/Liabilities_%` = as.numeric(`ShareFunds/Liabilities_%`),
    NetFinancialPosition = as.numeric(NetFinancialPosition),
    `Debt/Equity ratio_%` = as.numeric(`Debt/Equity ratio_%`),
    `Debt/EBITDA_%` = as.numeric(`Debt/EBITDA_%`),
    TotalAssetsTurnover_times = as.numeric(TotalAssetsTurnover_times),
    WorkingCapTurnover_times = as.numeric(WorkingCapTurnover_times),
    EBITDA = as.numeric(EBITDA),
    `EBITDA/Vendite_%` = as.numeric(`EBITDA/Vendite_%`),
    `ROA_%` = as.numeric(`ROA_%`),
    `ROI_%` = as.numeric(`ROI_%`),
    `ROS_%` = as.numeric(`ROS_%`),
    `ROE_%` = as.numeric(`ROE_%`),
    `NetP&L / OperatingP&L_%` = as.numeric(`NetP&L / OperatingP&L_%`),
    NetWorkingCapital = as.numeric(NetWorkingCapital),
    GrossProfit = as.numeric(GrossProfit),
    NetShortTermAssets = as.numeric(NetShortTermAssets),
    `ShareFunds-FixedAssets` = as.numeric(`ShareFunds-FixedAssets`),
    CashFlow = as.numeric(CashFlow)
)
```

2.3 Table 3.4: eliminating NA

```
# Repeat code Appendix 2.2 for the entire dataset. Then
```

```
AdjustedCompanyData <- na.omit(NewCompanyData)
dim(AdjustedCompanyData)

# continue with creation and tuning of forest as in Appendix 2.6.
```

2.4 Table 3.4: imputation of data with maxiter=5, ntree=50

```
Imp_Sample6 <- missForest(Sample6, maxiter = 5, ntree = 50,
                          variablewise = T, verbose = T)

# continue with creation and tuning of forest as in Appendix 2.6.
```

2.5 Step 2 results: main analysis across five samples

```
# Repeat for each sample from 1 to 5. The code requires adjustments for each sample. The
# structure remains the same, the names of the objects created and called by the functions need
# to change.

# Sample 1
# Imputation of missing values
Imp_Sample1 <- missForest(Sample1, maxiter = 3, ntree = 20,
                          variablewise = T, verbose = T)
write.csv(Imp_Sample1$ximp, "Data_Sample1.csv", row.names = FALSE) # save dataset

DataSample1 <- read_csv("Data_Sample1.csv")
NewSample1 <- as.data.frame(DataSample1)
attach(NewSample1)
sapply(NewSample1, class)

NewSample1 <- transform(
  NewSample1,
  LegalStatus = as.factor(LegalStatus))

colSums(is.na(NewSample1)) # no NA

# Tuning and Random Forest generation
# Creating of training and testing dataset, respectively 80% and 20% of the data
Sample1_trainingDataset <- sample_n(NewSample1, size = 0.8*nrow(NewSample1))
Sample1_testingDataset <- sample_n(NewSample1, size = 0.2*nrow(NewSample1))

tic("Best mtry") # start recording time
# Tuning of mtry (nodesize set constant at 27)
trControl <- trainControl(method = "cv", number = 10, search = "grid")

set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(2: 13))
Sample1_RF_mtry <- train(LegalStatus ~.,
                         data = Sample1_trainingDataset,
                         method = "rf",
                         metric = "Accuracy",
                         tuneGrid = tuneGrid,
                         trControl = trControl,
                         importance = TRUE,
                         nodesize = 27,
                         ntree = 300)
print(Sample1_RF_mtry)
toc() # stop recording time

# Output
        Random Forest

        44043 samples
```

```
           26 predictor
            2 classes: '0', '1'

        No pre-processing
        Resampling: Cross-Validated (10 fold)
        Summary of sample sizes: 39639, 39639, 39639, 39638, 39639, 39638, ...
        Resampling results across tuning parameters:

          mtry  Accuracy
           2    0.7974545
           3    0.7974543
           4    0.7982452
           5    0.7983217
           6    0.7993163
           7    0.7985766
           8    0.7993416
           9    0.7993418
          10    0.7987432
          11    0.7987432
          12    0.7987011
          13    0.7987120

        Accuracy was used to select the optimal model using the largest value.
        The final value used for the model was mtry = 9.
# best mtry = 9

tic("Best maxnodes")
# Tuning of maxnodes
store_maxnode <- list()
tuneGrid <- expand.grid(.mtry = 9) # the result obtained previously must be used
for (maxnodes in c(10:36)) {
  set.seed(1234)
  Sample1_RF_maxnode <- train(LegalStatus ~.,
                              data = Sample1_trainingDataset,
                              method = "rf",
                              metric = "Accuracy",
                              tuneGrid = tuneGrid,
                              trControl = trControl,
                              importance = TRUE,
                              nodesize = 27,
                              maxnodes = maxnodes,
                              ntree = 300)
  key <- toString(maxnodes)
  store_maxnode[[key]] <- Sample1_RF_maxnode
}
results_node <- resamples(store_maxnode)
summary(results_node)
toc()

# Output
        Call:
        summary.resamples(object = results_node)

        Models: 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 31,
        32, 33, 34, 35, 36
        Number of resamples: 10

        Accuracy
                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
        10 0.7522959 0.7578597 0.7683967 0.7655713 0.7732084 0.7760204    0
        11 0.7564397 0.7598367 0.7676317 0.7674334 0.7742285 0.7809232    0
```

```
        12 0.7595001 0.7636294 0.7692896 0.7689892 0.7741218 0.7804132    0
        13 0.7600102 0.7629280 0.7701824 0.7702646 0.7762688 0.7827085    0
        14 0.7604592 0.7633744 0.7723504 0.7706216 0.7768639 0.7819434    0
        15 0.7602041 0.7636630 0.7710751 0.7697544 0.7736115 0.7811783    0
        16 0.7617347 0.7644283 0.7733705 0.7718204 0.7783307 0.7811783    0
        17 0.7602652 0.7640121 0.7740083 0.7719224 0.7777210 0.7850038    0
        18 0.7596939 0.7632165 0.7729884 0.7713613 0.7776929 0.7847488    0
        19 0.7612245 0.7653847 0.7746460 0.7722284 0.7767009 0.7832186    0
        20 0.7617955 0.7656399 0.7714578 0.7718460 0.7785220 0.7824535    0
        21 0.7610303 0.7663414 0.7732432 0.7723306 0.7783091 0.7824535    0
        22 0.7625606 0.7660862 0.7720956 0.7729682 0.7812005 0.7839837    0
        23 0.7622449 0.7665965 0.7760204 0.7741668 0.7796480 0.7855139    0
        24 0.7646009 0.7696575 0.7750284 0.7750342 0.7793152 0.7852589    0
        25 0.7630706 0.7666601 0.7763040 0.7746515 0.7814830 0.7865340    0
        26 0.7640306 0.7672639 0.7746459 0.7747791 0.7822845 0.7867891    0
        27 0.7651109 0.7710604 0.7766866 0.7758504 0.7792870 0.7885743    0
        28 0.7663861 0.7700401 0.7743911 0.7758249 0.7804987 0.7878092    0
        29 0.7656210 0.7693045 0.7750287 0.7758250 0.7819017 0.7865340    0
        30 0.7674063 0.7698980 0.7766867 0.7767431 0.7814830 0.7883193    0
        31 0.7666412 0.7699126 0.7749011 0.7763605 0.7826945 0.7870441    0
        32 0.7661311 0.7710604 0.7766865 0.7761819 0.7798890 0.7870441    0
        33 0.7668962 0.7692110 0.7751562 0.7761309 0.7817104 0.7875542    0
        34 0.7660714 0.7690836 0.7763041 0.7760545 0.7836236 0.7867891    0
        35 0.7656210 0.7710605 0.7777069 0.7770748 0.7829859 0.7890844    0
        36 0.7653660 0.7700893 0.7741360 0.7757994 0.7808814 0.7865340    0
# best accuracy for maxnode = 35

tic("Best ntrees")
# Tuning of ntree (use best mtry and best maxnodes results)
tuneGrid <- expand.grid(.mtry = 9)
store_maxtrees <- list()
for (ntree in c(500, 1000, 1500, 2000)) {
  set.seed(1234)
  Sample1_RF__ntrees <- train(LegalStatus ~.,
                              data = Sample1_trainingDataset,
                              method = "rf",
                              metric = "Accuracy",
                              tuneGrid = tuneGrid,
                              trControl = trControl,
                              importance = TRUE,
                              nodesize = 27,
                              maxnodes = 35,
                              ntree = ntree)
  key <- toString(ntree)
  store_maxtrees[[key]] <- Sample1_RF__ntrees
}
results_tree <- resamples(store_maxtrees)
summary(results_tree)
toc()

# Output
        Call:
        summary.resamples(object = results_tree)

        Models: 500, 1000, 1500, 2000
        Number of resamples: 10

        Accuracy
                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
        500  0.7658761 0.7699126 0.7760490 0.7764371 0.7816467 0.7880643    0
        1000 0.7663861 0.7695937 0.7750287 0.7763095 0.7820292 0.7875542    0
        1500 0.7666412 0.7699763 0.7756664 0.7766666 0.7816106 0.7885743    0
        2000 0.7666412 0.7704866 0.7755388 0.7765901 0.7814831 0.7880643    0
```

```
# Best ntrees = 1500

tic("Final Random Forest")
# Final tree training post parameter tuning
tuneGrid <- expand.grid(.mtry = 9)
Sample1_FinalRF <- train(LegalStatus ~.,
                         data = Sample1_trainingDataset,
                         method = "rf",
                         metric = "Accuracy",
                         tuneGrid = tuneGrid,
                         trControl = trControl,
                         importance = T,
                         nodesize = 27,
                         ntree = 1500,
                         maxnodes = 35)

# Prediction on new data (testing dataset)
Sample1_prediction <- predict(Sample1_FinalRF, newdata = Sample1_testingDataset, type = "raw")
Sample1_ConfusionMatrix <- confusionMatrix(Sample1_prediction, Sample1_testingDataset$LegalStatus)
Sample1_ConfusionMatrix # confront prediction with actual values
toc()

print(Sample1_FinalRF$finalModel) # for class error and OOB error

plot(Sample1_FinalRF$finalModel, main = "Errors Sample3") # plot errors (Graphs 3.2 to 3.6)
legend(800,0.4, legend=c("Default", "OOB Error", "Active"),
           col=c("red", "black","green"), lty=c(3,1,3), cex=1,
           title="Line types", text.font=2)
title( ylab = "OOB Error Rate")

rquery.cormat(NewSample3[,-1]) # plot correlation (Graph 3.1)

# Computation and plot of variable importance (Graphs 3.7 to 3.11)
impSample3 <- varImp(Sample3_FinalRF)
plot(impSample3, main = "Variable Importance Sample 3" , cex = 1.15)
```

2.6 Step 3 results: change of training and testing dataset size

```
# Table 3.11: 70%/30%
Sample7_trainingDataset <- sample_n(NewSample7, size = 0.7*nrow(NewSample7))
Sample7_testingDataset <- sample_n(NewSample7, size = 0.3*nrow(NewSample7))

# Table 3.12: 90%/10%
Sample8_trainingDataset <- sample_n(NewSample8, size = 0.9*nrow(NewSample8))
Sample8_testingDataset <- sample_n(NewSample8, size = 0.1*nrow(NewSample8))
```

2.7 Step 4 results: reduction of number of variables and predictors

```
# Table 3.14
# Apply Appendix 2.2 then in tuning and random forest modify the number of predictors to be considered

Sample9_FinalRF <- train(LegalStatus ~ ShareFunds.Liabilities_. + CoverageOfFixedAssets + GrossProfit +
                         CostOfDebit_. + ROA_.+ SolvencyRatio_.+ Leverage + NetWorkingCapital +
                         Debt.Equity.ratio_. + EBITDA.Vendite_. + ROE_. + ROI_. + Interest.Turnover_. +
                         CashFlow + NetP.L...OperatingP.L_.,
                         data = Sample9_trainingDataset,
                         method = "rf",
                         metric = "Accuracy",
                         tuneGrid = tuneGrid,
                         trControl = trControl,
```

```
                        importance = T,
                        nodesize = 27,
                        ntree = 1000,
                        maxnodes = 33.2)
```

2.8 Training and testing of the final model

```
# Testing data: test1, test2, and test3, are datasets downloaded from AIDA Database
Imp_Test1 <- missForest(test1, maxiter = 1, ntree = 20, variablewise = T, verbose = T)
write.csv(Imp_Test1$ximp, "Data_Test1.csv", row.names = FALSE)

Imp_Test2 <- missForest(test2, maxiter = 1, ntree = 20, variablewise = T, verbose = T)
write.csv(Imp_Test2$ximp, "Data_Test2.csv", row.names = FALSE)

Imp_Test3 <- missForest(test3, maxiter = 1, ntree = 20, = T, verbose = T)
write.csv(Imp_Test3$ximp, "Data_Test3.csv", row.names = FALSE)

data1 <- as.data.frame(Data_Test1)
data2 <- as.data.frame(Data_Test2)
data3 <- as.data.frame(Data_Test3)
write.csv(rbind(data1, data2, data3), "Data_Testing.csv", row.names = FALSE)

Datatest <- read_csv("Data_Testing.csv")
NewDataTest <- as.data.frame(Datatest) # data containing all data in test1, test2, and test3

NewDataTest <- transform(
  NewDataTest,
  LegalStatus = as.factor(LegalStatus))

sapply(NewDataTest, class)
colSums(is.na(NewDataTest)) # no NA


# Trqining data: data from Sample 1 and Sample 2
data4 <- as.data.frame(Data_Sample1)
data5 <- as.data.frame(Data_Sample2)
write.csv(rbind(data4, data5), "Data_Training.csv", row.names = FALSE)

Datatrain <- read_csv("Data_Training.csv")
NewDataTrain <- as.data.frame(Datatrain)

NewDataTrain <- transform(
  NewDataTrain,
  LegalStatus = as.factor(LegalStatus))

sapply(NewDataTrain, class)
colSums(is.na(NewDataTrain)) # no NA

# Random Forest with parameters obtained from the “Development of the model” steps
trControl <- trainControl(method = "cv", number = 10, search = "grid")
tuneGrid <- expand.grid(.mtry = 10)
Test1_FinalRF <- train(LegalStatus ~.,
                       data = NewDataTrain,
                       method = "rf",
                       metric = "Accuracy",
                       tuneGrid = tuneGrid,
                       trControl = trControl,
                       importance = T,
                       nodesize = 27,
                       ntree = 1000,
                       maxnodes = 34)
```

```
FinalTest_prediction <- predict(Test1_FinalRF, newdata = NewDataTest, type = "raw")
FinalTest_ConfusionMatrix <- confusionMatrix(FinalTest_prediction, NewDataTest$LegalStatus)
FinalTest_ConfusionMatrix
```

2.19 Comparison, decision trees

```
data1 <- as.data.frame(Data_Sample1)
data2 <- as.data.frame(Data_Sample2)
data3 <- as.data.frame(Data_Sample3)
attach(data1)
attach(data2)
attach(data3)

Tree_train <-  sample_n(data1, size = 0.8*nrow(data1))
Tree_test <- sample_n(data1, size = 0.2*nrow(data1))

?rpart
modelTree <- rpart(LegalStatus~., data = Tree_train, method = "class", minsplit = 20, maxdepth = 30, cp
= 0.0006)

modelTree <- prune(modelTree)

print(modelTree)
rpart.plot(modelTree)

# prediction on Sample 1
a <- predict(modelTree,Tree_test, type = "class")
summary(a)
table(a,Tree_test[,1])

# prediction on Sample 2
b <- predict(modelTree,testingprova, type = "class")
summary(b)
table(b,testingprova[,1])

# prediction on Sample 3
c <- predict(modelTree,data3, type = "class")
summary(c)
table(c,data3[,1])
```

2.10 Comparison, Adaboost

```
# train on data1
model_adaboost <- boosting(LegalStatus~.,data = data1, mfinal = 100, trees = 500)

# prediction on Sample 2
prediction2 = predict(model_adaboost, data2)
print(prediction2$confusion)
print(prediction2$error)

# prediction on Sample 3
Prediction3 = predict(model_adaboost, data3)
print(prediction3$confusion)
print(prediction3$error)
```