TU WIEN Informatics

Decision Tree Classification with Missing Values

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Data Science

eingereicht von

Magdalena Fritz, BSc
Matrikelnummer 01630011

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser
Mitwirkung: Dr. Judith Cerdà Belmonte

Wien, 31. August 2023

Magdalena Fritz	Peter Filzmoser

TU WIEN Informatics

Decision Tree Classification with Missing Values

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Data Science

by

Magdalena Fritz, BSc
Registration Number 01630011

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser
Assistance: Dr. Judith Cerdà Belmonte

Vienna, 31st August, 2023

Magdalena Fritz | Peter Filzmoser

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Magdalena Fritz, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 31. August 2023

Magdalena Fritz

Acknowledgements

I want to acknowledge the persons who supported me throughout my studies and the final thesis.

My sincerest gratitude goes to Dr. Judith Cerdá Belmonte, head of the Natural Catastrophe Competence Center of UNIQA Insurance Group, for helping me defining a topic for my thesis and for giving me the chance to work on this project. I'm thankful that she allowed me to combine working on my thesis with my regular work at UNIQA, for her excellent guidance throughout my investigations and her motivating support. I want to thank her for the suggestions she shared with me about areas where I could improve. Her advice and mentoring have been invaluable to me, both professionally and personally.

Likewise, I would like to express my deepest gratitude to my supervisor Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Filzmoser for helping me to define the theoretical part of my thesis, which was needed to effectively solve the practical part. I would like to thank him for sharing his knowledge with me, for offering me advice and guidance throughout my scientific research as well as for calling attention to possible problems. His support and expertise have contributed to gain valuable results.

Finally, I would like to express genuine thanks to my family and close friends for their unwavering support and encouragement throughout the past years.

Kurzfassung

Datenanalysen spielen heutzutage eine äußerst wichtige Rolle, da auf deren Grundlage oft wesentliche Entscheidungen getroffen werden. Die ständig wachsende Menge an Daten führt jedoch zu immer größeren Herausforderungen, wie z.B. Dateninkorrektheit, schwierige Datenbereinigung und -verarbeitung, Laufzeitprobleme, etc. Ein häufiges Problem, mit dem die meisten Datensätze konfrontiert sind, ist die Unvollständigkeit. Datensätze sind sehr häufig unvollständig und ein angemessener Umgang mit fehlenden Werten ist essenziell, um zuverlässige und robuste Ergebnisse zu erzielen.

Die vorliegende Arbeit beschäftigt sich mit der Problematik, geeignete Strategien für den Umgang mit fehlenden Werten in realen Datensätzen zu finden, wenn ein Entscheidungsbaum oder ein Zufallswald trainiert wird. Im theoretischen Teil werden verschiedene Strategien analysiert, die von Entscheidungsbäumen und Zufallswäldern angewandt werden, um fehlende Werte direkt während des Trainierens zu behandeln. Außerdem wird eine einfache und mehrfache Imputationsmethode erläutert, nämlich *k-nearest neighbor* (kNN) Imputation und *multivariate imputation by chained equations* (MICE). Für den praktischen Teil werden reale Datensätze verwendet, die von UNIQA Insurance Group zur Verfügung gestellt wurden. Diese Datensätze wurden zusammengeführt, mit zusätzlichen Informationen angereichert, aufbereitet und analysiert. Der praktische Teil besteht aus zwei Abschnitten: einer Simulationsstudie und einer Fallstudie. Ziel der Simulationsstudie ist, festzustellen, ob die Techniken zur Behandlung fehlender Daten die Klassifizierungsgenauigkeit im Vergleich zu Entscheidungsbäumen und Zufallswäldern, die nur vollständige Beobachtungen der Daten berücksichtigen, verbessern. Außerdem soll untersucht werden, wie sich die Performance von Entscheidungsbäumen und Zufallswäldern, die fehlende Werte direkt verarbeiten, im Vergleich zu Entscheidungsbäumen und Zufallswäldern, die imputierte Daten verwenden, verändert. Zu diesem Zweck werden verschiedene Prozentsätze an fehlenden Werten in dem vollständigen Datensatz künstlich erzeugt, und die Auswirkungen der verschiedenen Methoden zur Behandlung der unvollständigen Daten untersucht. Ziel der Fallstudie ist, zu prüfen, inwieweit die fehlende "Nutzungsart" von Gebäuden in den Daten, d. h. die Nutzung oder der Zweck von Gebäuden wie z.B. Bäckerei, Familienhaus, Krankenhaus, usw., mit Hilfe eines Entscheidungsbaums oder eines Zufallswalds und einer geeigneten Technik zur Behandlung der fehlenden Daten vorhergesagt werden kann. Dafür wird der vielversprechendste Ansatz aus der Simulationsstudie mit dem gesamten bereitgestellten Datensatz trainiert und bewertet.

Die Ergebnisse der Simulationsstudie zeigen, dass die Leistung der getesteten Entscheidungsbäume und Zufallswälder nicht wesentlich von der Technik zur Behandlung der fehlenden Daten abhängt. Entscheidungsbäume bzw. Zufallswälder, welche auf imputierte Daten trainiert sind oder welche die fehlenden Werte direkt beim Trainieren handhaben, weisen keine wesentliche Verbesserung der Performance im Gegensatz zu Entscheidungsbäumen bzw. Zufallswäldern, welche nur vollständige Daten verwenden, auf. Außerdem zeigt sich, dass einfache Strategien zur direkten Behandlung von fehlenden Werten konkurrenzfähig zu komplexen Imputationsmethoden sind. Diese Ergebnisse lassen sich in erster Linie durch das spezielle Muster der fehlenden Werte in den Daten erklären. Die Wahl der konkreten Methode von Entscheidungsbäumen oder Zufallswäldern hat jedoch tatsächlich einen erheblichen Einfluss auf die Performance. Die wichtigsten Ergebnisse der Fallstudie sind, dass Zufallswälder zusammen mit der *Separate Class Method* zur Behandlung der fehlenden Werte, bei der fehlende Daten als eigene Klasse behandelt werden, sehr zufriedenstellende Ergebnisse bei der Vorhersage der Nutzungsart von Gebäuden liefern, wobei sie eine Genauigkeit von bis zu 90% erreichen.

Diese Masterarbeit bietet einen bedeutenden Einblick in die Technologie des maschinellen Lernens und ihre Auswirkungen auf die Zukunft der Industrie, während sie sich mit realen Datensätzen und einer der wichtigsten inhärenten Herausforderungen, nämlich der Unvollständigkeit der Daten, befasst. Sie unterstreicht, wie wichtig es ist, die Limitationen realer Datensätze, wie fehlende Werte und deren Muster, zu verstehen und richtig damit umzugehen. Zusätzlich zeigt die Arbeit, dass bestehende Modelle des maschinellen Lernens erfolgreich angewandt werden können, um Vorhersagen zu treffen, die zum Beispiel zur Verbesserung der Qualität realer Daten genutzt werden können und somit auch zu einer Verbesserung der Genauigkeit der Ergebnisse von Analysen, die mit diesen Daten durchgeführt werden, führen.

Abstract

Nowadays, data analyses are playing a significant role, as important decisions are often taken based on them. The continuous growth of data increases the challenges of the analyses, for example, data incorrectness, difficult data cleaning and processing, runtime issues, etc. A common challenge that most of the data sets face is incompleteness. More often than not, data sets are incomplete and appropriate handling of missing values is very crucial to achieve reliable and robust results.

The present thesis focuses on the problematic of finding appropriate strategies to handle missing values in real data when training a decision tree or random forest classifier. In the theoretical part, it analyzes several strategies applied by decision trees and random forests to handle missing values directly during training time, in addition to outlining a single and multiple imputation approach, namely k-nearest neighbor (kNN) imputation and multivariate imputation by chained equations (MICE). For the study part, real data sets provided by UNIQA Insurance Group are used. These data sets have been merged, enriched with additional information, pre-processed and explored. The study part is divided in two sections: a simulation study and a case study. The goal of the simulation study is to identify if the missing data handling techniques improve the classification accuracy compared to decision trees and random forests that take only complete observations of the data into account. Additionally, it aims to examine how the performance of decision trees and random forests that directly handle missing values changes compared to decision trees and random forests that use imputed data. For that purpose, different percentages of missing values are artifically created on the complete data set to study the effect of the different methods for handling the incomplete data. The aim of the case study is to check the extent to which the missing "occupancy" attributes of buildings in the data, that is, the use or purpose of buildings such as bakery, family house, hospital, etc., can be predicted using a decision tree or random forest classifier with appropriate missing data handling techniques. Therefore, the most promising approach from the simulation study is trained and evaluated with the full provided data set.

The simulation study concludes that the performance of decision trees and random forests is similar for all the missing data handling techniques used. Training decision trees or random forests on imputed data or using techniques applied by decision trees or random forests to handle missing values directly during training, does not considerably improve the performance compared to simply using the data that is complete. Furthermore, simple

strategies of decision trees and random forests to handle missing values directly are competitive to complex imputation approaches. These outcomes are primarily explained by the special missingness pattern reflected in the data. However, the choice of the decision tree or random forest method has indeed a significant impact on the performance. The key findings of the case study are that random forests along with the separate class method, where missing values are treated as a separate category, exhibit highly satisfactory results in predicting the occupancy for buildings by achieving an accuracy of up to 90%.

This master thesis offers a significant insight into machine learning technology and its impact on the future of industry, while dealing with real-world data sets and one of the most relevant inherent challenges, which is incompleteness of the data. It highlights the importance of understanding and properly handling the limitations of real-world data sets, such as missing values and their patterns. Additionally, it shows that existing machine learning models can be successfully applied to make predictions, which can be used, for example, to improve the quality of real-world data and consequently to improve the accuracy of the results of the analyses performed with these data.

Contents

CHAPTER 1

Introduction

The first chapter of the thesis describes the motivation to the topic and defines the problem to solve. It states its contributions and explains the methodological approach before outlining the structure of the thesis.

1.1 Motivation

We live in a world where data and data analysis is playing the most important role ever. Governments and private companies put immense effort to gather large data sets and analyze them to take decisions that influence people's behaviors and quality of life, such as policy formation, public service delivery, marketing efficiency, new pharmaceutical products, etc. [Kau23] [Agg23] Incorporating data and analytics into decision-making and business operations exhibits an extreme amount of benefits across a variety of industries. The amount of data collected is increasing every day, and so is the complexity. One of the main challenges of data scientists is that the data sets are more often than not incomplete and data sets sometimes are not large enough after data cleansing and preparing. Correctly processing data to achieve data correctness and accurateness is demanding. These challenges may particularly appear when dealing with real-world data sets and companies and governments often hire data scientists to cope with them. [Cao17]

This thesis examines missing values in data sets provided by UNIQA Insurance Group (UIG), and more concretely from the Natural Catastrophes Competence Center (NCCC), which funds this thesis. The NCCC is the owner of a large database which contains the buildings insured by UIG along with their attributes such as geographical location, insurance conditions, building characteristics, occupancy, and so on. This data has key applications in insurance such as the determination of capital requirements and the optimization and pricing of protection covers for the purpose of risk transfer, among many others.

A vast literature has been generated on statistical analysis of data with missing values and on general methods that handle missing values, e.g. Little and Rubin (2019) [LR19], Schafer and Graham (2002) [SG02], Van Buuren and Groothuis-Oudshoorn (2011) [VBGO11]. However, in this thesis the focus is on state-of-the-art methods such as decision trees and random forests, and their ability to cope with missing data. Within this context, there is scarce literature. Decision trees and random forests represent commonly used machine learning algorithms that are used to accurately classify unlabeled observations. They are simple and transparent, yet effective. Decision trees and random forests are very flexible and handle well data consisting of variables of mixed types, which contributed to the choice of these models. While missing values can occur in training data as well as in test data and affect learning and classification accuracy. Therefore, handling missing values appropriately is essential for classifier learning. Furthermore, using appropriate missing data handling techniques reduces introducing bias and receiving misleading conclusions. In order to select the correct missing data handling technique, various considerations may be decisive, such as the percentage of missingness, the size of the data, missingness patterns and missingness mechanisms. [Twa09]

1.2 Problem Definition

One of the challenges of real-world data sets is incompleteness, that is, attribute values are missing in the data. Appropriate handling of such omissions is crucial when using machine learning algorithms, like decision trees or random forests, in order to learn a classifier and for enhancing the prediction accuracy.

The most prevalent approach to handle missing values in the data when the aim is to fit a decision tree or random forest classifier is the deletion of all observations containing missing values and imputation of missing values in a separate step before fitting the model. However, decision trees and random forests can also handle missing values directly during the training phase and consequently no explicit imputation step is needed before fitting the model. For that, different techniques can be used.

The present thesis focuses on the problematic of finding appropriate strategies for handling missing values in the data when learning a decision tree or random forest classifier. The study data set provided by UIG consists of buildings and their properties being one of them the "occupancy". The occupancy is defined as the "use or purpose of the building"; e.g. bakery, family house, hospital, etc. Whereas this information is very relevant for any insurer, it is not always included. Therefore, to improve the quality of the provided data set, a classifier should be fitted that predicts the occupancy type for an insured object.

The following research questions should be answered in the present thesis:

- How do decision trees and random forests that use imputed data or that directly handle missing values improve classification accuracy compared to decision trees and random forests that take only complete instances into account?

- How do decision trees and random forests that directly handle missing values perform compared to decision trees and random forests that use imputed data?
- To which extent the missing "occupancy" attributes of buildings can be predicted using a decision tree or random forest classifier with appropriate missing data handling techniques?

1.3 Contributions

The main contribution of this thesis is a summary, analysis and comparison of state-of-the-art techniques for handling missing attribute values using decision trees and random forests whereby these missing values appear in the training data. While in the literature several methods proposed to deal with missing values resort to imputation techniques, this study focuses on multiple imputation and on strategies to deal with missing data directly at the training time without an explicit imputation step, where, on the other hand, the amount of literature is relatively small. The thesis discusses the assumptions behind the different techniques and their appropriateness.

The second relevant contribution is a comparison of the performance of decision tree and random forest classifiers that use imputation approaches for the missing values in the data and the performance of decision tree and random forest classifiers that directly handle missing values. This outlines the effect of the different methods for handling incomplete data. It highlights the most promising approaches for different proportions, patterns, and mechanisms of missing data.

Additionally, many experimental studies, e.g. Twala et al. (2008, 2009) [TJH08] [Twa09] or Gavankar and Sawarkar (2015) [GS15], focus on selected techniques of decision trees to handle the missing values in the data directly. However, a full overview and comparison of existing applicable strategies for dealing with missing values while growing a decision tree or random forest is missing in the literature.

And finally, the third contribution worthwhile to mention is the demonstration of an appropriate strategy to deal with the missing values in the input data set provided by UIG. A random forest classifier that predicts the occupancy of such buildings based on various attributes with mixed characteristics is trained. The extent is checked to which the occupancy of the buildings can be predicted based on the information given, while appropriate methods are used for the missing values.

1.4 Methodological Approach

The first part of the thesis which comprises Chapter 2, 3 and 4 is a review of peer reviewed scientific literature where the most widely used and promising techniques for dealing with incomplete data in decision trees and random forests are summarized, analyzed and compared. [Sny19] The study of the similarities and differences between the methods

and the discussion of the assumptions of each method contribute to the understanding of their appropriateness.

In the second part of the thesis which comprises Chapter 5, a simulation study is conducted. For a correct design, analysis and reporting, the paper of Morris et at. (2019) [MWC19] is considered. Promising imputation approaches and different strategies of handling missing data directly at learning time are compared to the complete-case analysis approach, where the observations with missing values are deleted. Different proportions, patterns, and mechanisms of missing data are synthetically created on a complete real-world data set with mixed characteristics in order to study the effects of the different methods for handling the incomplete data. The performance of the different methods is then compared. Afterwards, it is analyzed how missing data handling techniques along with decision trees and random forests improve the classification accuracy compared to classification taking only complete instances into account.

Finally, based on the understanding gained from the literature review and the outcome of the simulation study, the thesis deals in Chapter 6 with the case study of learning a decision tree or random forest classifier for predicting the missing occupancy attributes in the supplied data set by UNIQA Insurance Group. [Fid84] Based on the information included in the data set such as location, insurance conditions, building properties, and other building characteristics the aim is to predict the missing occupancy attribute for buildings using decision trees or random forests. The data set consisting of variables of mixed types contains missing values for several attributes in different amounts. With the use of different missing data handling techniques, it is analyzed to which extent the occupancy attribute can be classified based on the available information using a decision tree or a random forest. After comparing the most auspicious approaches from the simulation study a final decision tree is learned within the case study.

Figure 1.1: Project Organization

Figure 1.1 gives an overview of the time taken for each of the described parts. While the author worked full time on this project for a total of 6 months, a large part of the time, about 40%, was spent on business understanding and data mining, cleaning and preparing. This part was very time-consuming, but extremely important in this project to achieve well-performing and reliable results.

1.5 Structure of the Thesis

The thesis starts with an introduction for the reader in Chapter 2 regarding the theoretical basics of decision trees and random forests. Additionally, it outlines the fundamentals of missing data. It continues with Chapter 3 with missing data imputation techniques which are theoretically explained and discussed. Single imputation and multiple imputation are compared and specific methods are analyzed. In Chapter 4, decision tree and random forest missing data handling techniques are introduced which can handle missing values directly during training phase without a separate imputation step. All theoretically analyzed methods are tested on the data set provided by UNIQA Insurance Group within a simulation study in Chapter 5. In Chapter 6, an appropriate strategy to deal with the missing values in the UIG data set is used and a final classifier is fitted to predict the missing occupancy attribute of individual buildings based on various attributes. Finally, the conclusions and the correspoding discussion are included in Chapter 7.

CHAPTER 2

Methodological Foundations

In this chapter an introduction to the methodological foundations of decision tree and random forest classification is given. Different performance criteria to evaluate a machine learning model are discussed. The concept of missing data is introduced with a main focus on different patterns and mechanisms of missing data.

2.1 Decision Tree Classification

Decision trees represent a simple yet effective methodology for supervised classification. In a classification problem, measurements on explanatory variables are available for multiple observations and based on these, the aim is to predict the values for the categorical response variable, that is, to which class out of a finite, predefined set of classes the observations belong. [Bre17]

A decision tree follows a tree structure and it is created with a root, nodes, and branches. It starts from the root, which corresponds to the first node of the tree, and moves downwards. For each node of the tree, multiple branches can originate from it into other nodes. The node where the branches begin is called parent node and the nodes where the branches end are called child nodes. The nodes with no extended branches are called leaf or terminal nodes. All nodes that are no leaf nodes are called internal nodes. [AKAM12]

Decision trees partition the space of the explanatory variables by applying a sequence of splits on single variables. The root of the tree contains the full space and the final partition is defined by the leaf nodes of the tree. Since each internal node is split based on the value of one explanatory variable, these are interpreted as certain characteristics. On the other hand, the branches define ranges of values and consequently define the outcomes for the characteristics. The variable on which the split is based is often called split variable and the ranges of the values that are represented by the branches are defined by so-called split points. For a continuous split variable X_i, the binary split with

split point c can be illustrated as in Figure 2.1. All observations with a value smaller than c for the split variable are assigned to the left child node, and all observations with a higher value than c for the split variable go to the right child node. If the split variable X_i is categorical with a finite set of categories, then a binary split can be defined by assigning all observations with a value for X_i from a subset of categories $S_i \subset S$ to the left child node and all other observations to the right child node as in Figure 2.2.

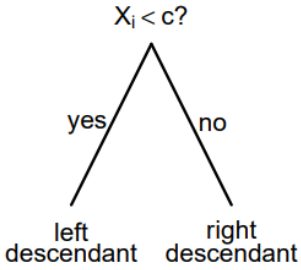


Figure 2.1: Split on a continuous split variable X_i with split point c [CCS12]

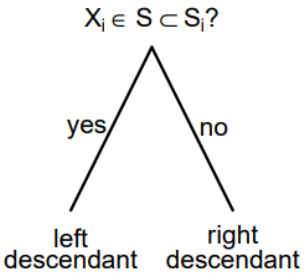


Figure 2.2: Split on a categorical split variable X_i using $S_i \subset S$ for split definition [CCS12]

Usually, the splits for the nodes in the tree are defined by doing an exhaustive search over all possible split variables and split points, and by selecting the one that 'best' splits the data. 'Best' is based on a *splitting criterion*, which is a measure of the purity of a node, that is, the homogeneity of the instances within a node with respect to their target values. The exact definition is given later in Chapter 4. [CCS12]

A node is not split further and is declared to be a terminal node if it is 100% pure, that is, if it contains only observations from the same class, or if a split does not result in an improvement of the node purity or if a stopping criterion is reached. Possible stopping criteria are the minimum number of observations in a node or the maximal depth of the

tree, which is defined by the length of the longest path from the root of the tree to one leaf node.

After a tree is grown, it is often very complex, especially when it is fully grown until all nodes are pure and no stopping criterion is used. Then the tree often overfits the data which means that more structure is inferred to the tree by the training data than is justified. By pruning, that is, by removing branches of the tree that do not provide much power to classify instances, overfitting is tackled, and at the same time the tree is simplified. This improves the accuracy for new unseen data. [Qui93]

Once a decision tree is grown and therefore trained, it can be used to make predictions for new observations where the output label is not known. This classification for a new observation x, which is assumed to be complete without any missing values, is done by passing it down the tree until it reaches a terminal node. All the observations of that terminal node are used to make the classification for the new variable by calculating the category that appears most often. [CCS12]

One major advantage of decision trees is their interpretability. They are simple and yet effective. Decision trees can handle different types of data and there are no assumptions about the distribution of the data. Therefore, decision trees are recommended to be used when a simple and flexible approach for making predictions, which is highly interpretable and can handle nonlinear relationships, is needed. [ZZ08] However, caution is required in terms of overfitting, since too complex trees often exhibit poor performance on new data, and in terms of instability, since small changes in the data can lead to significantly different trees and this to inconsistent results. Decision trees might not be advisable when dealing with high-dimensional or complex data, since they could become too big and complex to generalize well for new, unseen data or could have difficulties to cope with the data, which results in high training-time complexity costs. [Dec23]

2.2 Random Forest Classification

A *Random Forest* is a methodology for supervised classification. As in decision trees, the goal is to find a function of the explanatory variables to predict the output variable, which is categorical. A random forest consists of an ensemble of decision trees, where each of the trees in the forest depends on multiple random elements. The prediction function is created by this collection of trees, which are the so-called *base learners*. So, the base learners are combined to form the *ensemble predictor*.

Each of the decision trees is grown as usual. However, in contrast to using all the given data for fitting the tree, only a random subspace of the data is used for each of the trees. So, an independent bootstrap sample from the original data is chosen for each of the trees to train them. Furthermore, when splitting a node, a search for the best split variable is not done among all variables of the data but rather among only randomly selected ones. These two steps are necessary to uncorrelate the trees and minimize the variance. Each of the trees in the forest is fully grown until no more improvement can be made with a

split of a node or until a stopping criterion is reached. Pruning is not done for the trees in a random forest, although this is important for individual decision tree classifiers. The reason for this is that the two layers of randomization applied when creating a random forest lead to a diverse set of decision trees which prevents overfitting anyway.

Once all trees are grown a prediction for a new unseen observation with unknown class label can be made. For each of the trees, the new observation is passed down until it reaches a terminal node and the observations in the specific terminal node are used to specify a label. So, a label is returned by each of the trees and these are combined by majority voting to form the prediction. Majority voting means to take the most frequently predicted class label.

Random forests have multiple advantages. They are powerful and flexible. They can handle high-dimensional problems well and can be implemented in parallel. Therefore, the training and prediction is rather fast. [CCS12] Random forests are recommended to be used when the accuracy of single decision trees should be improved and when dealing with complex data with possible outliers, since random forests are robust and reduce overfitting. However, random forests are not advisable when interpretability is extremely crucial, since it could be challenging to comprehend specific outcomes, especially if a forest consists of a vast number of trees. [Ran23]

2.3 Performance Criteria

Performance indicators are needed to evaluate and compare the performance of different machine learning methods or the performance of the same method using different tuning parameters.

One important performance indicator used to evaluate a classifier is called *confusion matrix*. It is a cross table that counts the number of occurrences between the true class labels and the predicted class labels of a classification. Figure 2.3 is showing an example. The true classes are listed in the rows and the predicted class labels are in the columns, both in the same order. The diagonal, highlighted in green, is denoting the number of observations for which the prediction and the true class label agree. For a good classifier, the values are high in the diagonal and low everywhere else. The confusion matrix forms the basis for many other performance indicators as it contains all the relevant information about the performance of an algorithm.

The *accuracy* is a very simple and intuitive metric that can be derived from the confusion matrix. It is defined as the ratio of the number of the correctly classified observations and the total number of observations. Therefore, it is the ratio of the sum of the diagonal elements of the confusion matrix and the sum of all elements of the confusion matrix. It takes values within [0,1]. The accuracy is a measure for the overall performance, when all observations have the same weight and contribute equally. Thus, it is useful when the aim is to correctly predict as many observations as possible regardless of the class distribution. However, if the aim is that the algorithm should work for all classes equally well, then

		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Figure 2.3: Example of a confusion matrix [GBV20]

the accuracy is not suitable, especially in case of imbalanced data sets. Classes with many observations would have a higher weight compared to classes with less observations and consequently, classification errors for small classes where the algorithm is performing poorly are not noticeable.

Another performance criterion that accounts for this is the *F1-Score*. It is derived from the *precision* and the *recall*. The precision for one class k is the fraction of the correctly as class k predicted elements, denoted as TP_k (true positives), divided by the total number of observations predicted to have class k. This is composed of the sum of the correctly as class k predicted observations TP_k and the incorrectly as class k predicted observations FP_k (false positives). Thus, the precision for class k is definied as

$$Precision_k = \frac{TP_k}{TP_k + FP_k}. \tag{2.1}$$

The precision is a measure of trust of the model's prediction for an observation belonging to class k.

The recall for class k is the fraction of the correctly as class k predicted observations divided by the total number of observations belonging to class k, which is the sum of the correclty classified observations TP_k and the incorrectly classified observations FN_k from class k (false negatives):

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{2.2}$$

Thus, the recall is a measure of the accuracy for class k and measures the performance of the model in finding all observations of class k.

The *F1-Score* is a weighted average between precision and recall. It is definied as

$$F1\ Score = \frac{2}{Precision^{-1} + Recall^{-1}} = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{2.3}$$

Precision and recall could refer here either to the ones of a binary classification problem or to the ones of a specific class of a multi-class problem. The F1-score takes values within [0,1]. If precision and recall are both high, the F1-score will also be high, being 1 the maximum possible value. This is the ideal scenario, indicating high accuracy and high trust of the model's prediction. If both the precision and the recall are low, the F1-score will be low. If the recall is high and the precision is low, or the other way around, the F1-score will be low. High recall and low precision indicates good performance in finding observations of a specific class but also predicting a significant number of observations wrongly as this class. Low recall and high precision indicates overall less observations being predicted to be of a specific class but a higher trust in these predictions. If either the precision or the recall is 0, the F1-score will be 0, which is the lowest possible value.

The so-called *macro average precision* and *macro average recall* are then defined as

$$Macro\ Average\ Precision = \frac{\sum_{k=1}^{K} Precision_k}{K} \tag{2.4}$$

and

$$Macro\ Average\ Recall = \frac{\sum_{k=1}^{K} Recall_k}{K}, \tag{2.5}$$

where K is the total number of different classes.

The *Macro F1-Score*, which includes all classes of a multi-class problem, is defined as the harmonic mean of the macro average precision and the macro average recall, thus

$$Macro\ F1\ Score = 2 * \frac{Macro\ Average\ Precision * Macro\ Average\ Recall}{Macro\ Average\ Precision^{-1} + Macro\ Average\ Recall^{-1}}. \tag{2.6}$$

In contrast to the accuracy, all classes have the same weight in the macro F1-score and huge classes have the same importance as small classes. It takes values between 1 and 0, whereby high macro F1-scores indicate good performance on all different classes and low macro F1-scores indicate poor performance for the classes. More precisely, if precision and recall are high across all classes, the macro F1-score will be high, being 1 the maximum possible value. If either the macro average precision and/or the macro average recall is low, the macro F1-score will be low. If either the macro average precision or the macro average recall is 0, the macro F1-score will be 0, being the lowest possible value.

All in all, it is important to assess in addition to the accuracy the (macro) F1-score to obtain a good understanding of the performance of a classifier, especially in case of imbalanced classes. [GBV20]

2.4 Missing Data

Missing data is a common problem when working with real-world data sets, and its origin is due to various reasons such as missing answers in a questionnaire, failures in a manual

data entry process, failures in measurements or experiments, and censored or anonymous data. [Kai14]

Little and Rubin (2019) [LR19] defined missing data as unobserved values that would be meaningful for the analysis if they were observed and stated that missing values hide meaningful values. If this applies, they state that it makes sense to fill in the missing values appropriately to gain back this information.

Assuming to have a rectangular data set matrix, where the rows represent the observations or units and the columns represent the variables or characteristics measured for each observation, missing values are usually indicated in the data matrix as *NULL*, *NA*, empty strings or as symbols like question marks. Missing values expressed in one of these ways are easy to detect. However, sometimes missing values are indicated as outliers with values like 999, and then it is more difficult to detect them. [Kai14]

Missing values can occur in training data as well as in test data or in both. Therefore, handling them appropriately is very important for classifier learning to achieve good prediction accuracy. Incomplete data may have an impact on the interpretations of the data and the models resulting from this data as well. Using appropriate missing data handling techniques avoids introducing bias. Furthermore, misleading and invalid conclusions for a research study are avoided and limitations of the generalizability of the research findings are prevented. [Twa09]

For the choice of the correct missing data technique, various considerations may be decisive, such as the percentage of missingness in the data or the size of the sample. [Twa09] According to Pyle (1999) [Pyl99], percentages of missing values of 0-1% are trivial and of 1-5% are manageable. Rates of 5-15% require sophisticated methods while rates of more than 15% may influence the interpretation severely.

To assess the potential impact of missing data for choosing the correct missing data techniques, it is very common to distinguish the missingness pattern and the missingness mechanisms. While the missingness pattern simply defines which values in the data matrix are observed and which values are missing, the missingness mechanism descibes the possible relationship between missingness and other variables. [LR19]

2.4.1 Patterns of Missing Data

The pattern of missing data defines which values in a data matrix are missing and which values are observed. Some missing data techniques are applicable to any missingness pattern, other techniques, however, are only intended for specific patterns.

The pattern of missing data is defined as follows. Let $X = (x_{ij})$ be a rectangular data set matrix of dimension $(n \times p)$ without any missing values and therefore a complete data set. The rows $x_i = (x_{i1}, ..., x_{ip})$, for $i = 1, ..., n$, represent the observations while x_{ij} denotes the value of variable j measured for observation i. In case of missing data in the matrix X, let $M = (m_{ij})$ denote the missingness indicator matrix with $m_{ij} = 1$ if x_{ij} is

missing and $m_{ij} = 0$ if x_{ij} is not missing. Thus, this missingness indicator matrix M specifies the pattern of missing data. [LR19]

Some examples of missingness patterns can be seen in Figure 2.4. The rows in the figure correspond to observations and the columns to variables. The most frequently occurring patterns of missing data are univariate, monotonic, and arbitrary. Univariate missing data means that missingness is limited to a single variable. Monotonic missing data means that if the data is missing for the variable j, then the data is also missing for the variables $j + 1, ..., j + l$, and a so-called staircase line can divide the data matrix into missing and observed data. When any variable values may be missing for any observations, then the pattern is arbitrary or general. [Twa09]

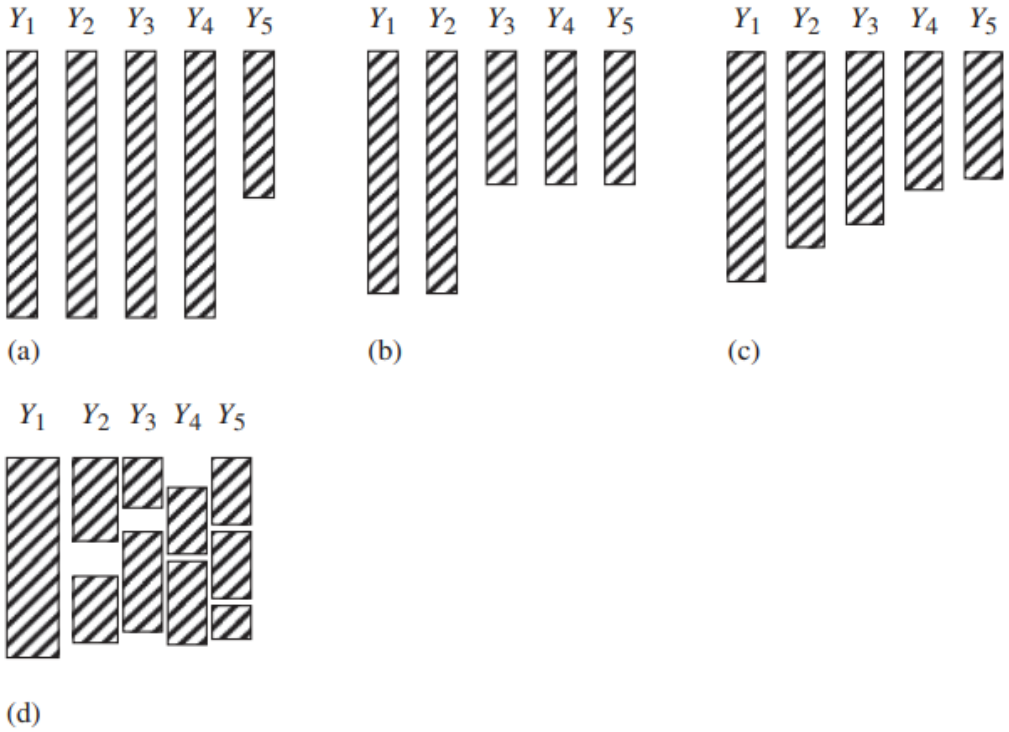


Figure 2.4: Examples of missingness patterns. (a) Univariate, (b) multivariate with two patterns, (c) monotone, (d) general. [LR19]

2.4.2 Mechanisms of Missing Data

Mechanisms of missing data describe the relationships between missingness of variables and the underlying values of the variables in the data set. The dependencies in these mechanisms severely influence the properties of missing data imputation techniques. Therefore, it is crucial to investigate the mechanism of missingness and thus the process

that could have generated the missing data. Little and Rubin (2019) [LR19] summarized three different missingness mechanisms: missing completely at random (MCAR), missing at random (MAR), missing not at random (MNAR).

These missingness mechanisms are defined as follows. Let $X = (x_{ij})$ denote the complete data matrix with rows x_i and $M = (m_{ij})$ be the missingness indicator matrix with rows m_i as specified in Section 2.4.1. Let the rows (x_i, m_i) be independent and identically distributed over i. Then the mechanisms of missing data can be explained with the conditional distribution of m_i given x_i, thus with $f_{M|X}(m_i|x_i, \phi)$, while ϕ represents unknown parameters.

Data is missing completely at random (MCAR) if the missingness does not relate to the missing or observed values of the data, which means that for all i and any distinct values x_i and x_i^* in the sample space of X the equation

$$f_{M|X}(m_i|x_i, \phi) = f_{M|X}(m_i|x_i^*, \phi) \tag{2.7}$$

applies.

For observation i let $x_{(0)i}$ be the components of x_i that are observed and $x_{(1)i}$ be the components of x_i that are missing. Then the data is missing at random (MAR) if the missingness depends only on the observed components $x_{(0)i}$ of observation i. This means that for all i and any distinct missing components $x_{(1)i}$ and $x_{(1)i}^*$ in the sample space of $x_{(1)i}$,

$$f_{M|X}(m_i|x_{(0)i}, x_{(1)i}, \phi) = f_{M|X}(m_i|x_{(0)i}, x_{(1)i}^*, \phi). \tag{2.8}$$

If the distribution of m_i depends on $x_{(1)i}$, the missing components of x_i, the mechanism is denoted as missing not at random (MNAR). This means that Equation (2.8) does not apply to all i and all distinct missing components $x_{(1)i}$ and $x_{(1)i}^*$. [LR19]

MCAR is the strongest assumption of the three missingness mechanisms and Twala (2009) stated that missingness is ignorable if it is MCAR or MAR, which means that the reasons for missing data can be ignored in the analysis of the data and this simplifies the methods used for estimating missing values. In practice, however, the assumption of data being MCAR is rarely met. The least restrictive assumption is MNAR. This mechanism is sometimes also called informatively missing and cannot be ignored. [Twa09]

CHAPTER 3

Missing Data Imputation

The main approaches to deal with missing values when learning a classifier are omission and imputation of missing values, or using classifier methods that can directly handle missing values. In this chapter, the focus is on the former one. Popular methods of single and multiple imputation are analyzed to enable later comparisons with decision tree methods that directly handle missing values.

3.1 Listwise Deletion/Complete Case Analysis

One of the simpler approaches to handle missing values in a data set is called *Listwise Deletion* or *Complete Case Analysis*. In this approach all observations containing missing values are ignored and deleted to obtain one complete data set for further analysis. Therefore, all observations with missing values are simply not used for training a classifier. Although listwise deletion is a frequently used method, a large amount of useful information could be lost with this methodology. This could result in a loss of prediction power, especially if the data size is small and the number of missing values is high. However, if the number of missing values is small and the sample is large, it could be a reasonable strategy only if the missing data are missing completely at random (MCAR), otherwise it may cause bias in the results when applying it to data that do not fulfill the assumption of MCAR. [GS16]

3.2 Single Imputation - kNN

In single imputation, each missing value is replaced by one imputed value, which results in one complete data set. [TJH08] A very common and well-known method of single imputation is *k-nearest neighbor* (kNN) imputation. In this method, a missing value is imputed by taking the k nearest neighbors into account and calculating their aggregation. Depending on the type of the missing variable, a different aggregation is used.

To find the k nearest neighbors, that is, the k observations that are most similar to the observation with the missing value to impute, a distance measure is needed. For that purpose, kNN uses an extension of the *Gower distance*, which calculates the distance between two observations by taking the weighted mean of the contributions of each variable. Thus,

$$d_{ij} = \frac{\sum_{t=1}^{p} w_{ijt}\delta_{ijt}}{\sum_{t=1}^{p} w_{ijt}} \tag{3.1}$$

specifies the distance between the ith and jth observation with w_{ijt} denoting the weight of the tth variable and δ_{ijt} denoting the contribution of the tth variable. The weights w_{ijt} in the distance measure are defined by

$$w_{ijt} = \begin{cases} 1 & \text{if } x_{it} \text{ and } x_{jt} \text{ not missing} \\ 0 & \text{if } x_{it} \text{ or } x_{jt} \text{ or both missing.} \end{cases} \tag{3.2}$$

The distance measure can deal with distance variables of all types such as continuous, ordinal, binary, nominal and semi-continuous. For continuous variables, the contribution of the tth variable in the distance measure between the ith and jth observation is defined by

$$\delta_{ijt} = \frac{|x_{it} - x_{jt}|}{r_t}, \tag{3.3}$$

where x_{it} denotes the value of the tth variable of observation i and r_t denotes the range of the variable t. Thus, the contribution is the absolute distance of the variable values for the observations divided by the total range for the variable. The same is used for ordinal variables but converting them to integer variables beforehand. For binary and nominal variables, the contribution of the tth variable δ_{ijt} is calculated by a 0/1 distance as

$$\delta_{ijt} = \begin{cases} 0 & \text{if } x_{it} = x_{jt} \\ 1 & \text{if } x_{it} \neq x_{jt}. \end{cases} \tag{3.4}$$

A semi-continuous variable is a variable that consists of a continuously distributed part and a probability mass at one point, e.g. a variable that is 0 for one part of the observation and continuously distributed for the other part of the observations. The contribution for such a variable, where s_t denotes the point of the probability mass, e.g. 0, is defined as

$$\delta_{ijt} = \begin{cases} 0 & \text{if } x_{it} = s_t \wedge x_{jt} = s_t \\ 1 & \text{if } x_{it} \neq s_t \wedge x_{jt} = s_t \\ 1 & \text{if } x_{it} = s_t \wedge x_{jt} \neq s_t \\ |x_{it} - x_{jt}|/r_t & \text{if } x_{it} \neq s_t \wedge x_{jt} \neq s_t. \end{cases} \tag{3.5}$$

All distances d_{ij} between two observations are in [0,1], since all contributions are inside this interval.

Once the k nearest neighbors are defined for the observation with missing value for the tth variable, their values are aggregated to obtain the imputation for the missing value.

If the variable is continuous, the median of the values of the k nearest neighbors is taken. For categorical variables, a majority voting is used, where the category that appears most often in the k nearest neighbors is selected. In case of a tie between two categories, one of them is selected at random.

In this method, the distances are only calculated for the observations with missing values. Therefore, not the whole distance matrix has to be calculated and therefore the method is appropriate for large data sets. [KT16] [D'O21]

kNN imputation as described above is implemented in the `R` software for statistical computing in the package `VIM` and can be applied by using the function `kNN` (see Chapter 5.2). [KT16]

In single imputation, each missing value is replaced by one imputed value, which is treated as true value. However, this approach is not taking the uncertainty of the imputation into account and therefore underestimates the variance. This could lead to invalid tests and confidence intervals. [Twa09]

3.3 Multiple Imputation - MICE

A popular class of imputation methods that should overcome limitations of single imputation is *multiple imputation* (MI). Rubin (1996) [Rub96] defined multiple imputation by three main steps. These are imputation, analysis, and pooling. In contrast to single imputation, the missing values are imputed not once but multiple times, more precisely m times, with plausible values in the first step of the process. The variable m can take integer values, and theoretically higher values yield better results. However, high values for m increase the runtime and the required memory enormouosly, and often a value between 5 and 20 leads to sufficiently good results. [VB18] The imputed values are drawn from a distribution, that is separately modelled for each of the missing values. This results in m complete data sets, which are identical for the non-missing values of the original data set and differ in the imputed values for the missing values. In the second step, each of the imputed data sets is analyzed by using a method that would have been used if the data had been complete such as linear regression or decision tree. This leads to one analysis per imputed data set, so in total in m analyses. These outcomes differ from each other as their input data differ for each analysis. Lastly, the m results are pooled or combined to one output. By using m imputations and analyses, a measure of the variability, e.g. calculated with the standard deviation, that reflects the uncertainty of the imputation, is obtained in the pooling step of multiple imputation. [Twa09] Figure 3.1 illustrates the three-step process of multiple imputation for $m = 5$.

In the following, let's assume to have observed a $(n \times p)$ matrix $X^{\text{obs}} = (X_1^{\text{obs}}, ..., X_p^{\text{obs}})$ which contains data values on p variables X_j, $j = 1, ..., p$, for n observations, and which contains missing values. The hypothetically complete matrix is denoted by $X = (X_1, ..., X_p)$ and is obtained by taking together the observed and the missing data part as $X = (X^{\text{obs}}, X^{\text{mis}})$, where $X^{\text{mis}} = (X_1^{\text{mis}}, ..., X_p^{\text{mis}})$ is a matrix of the same size

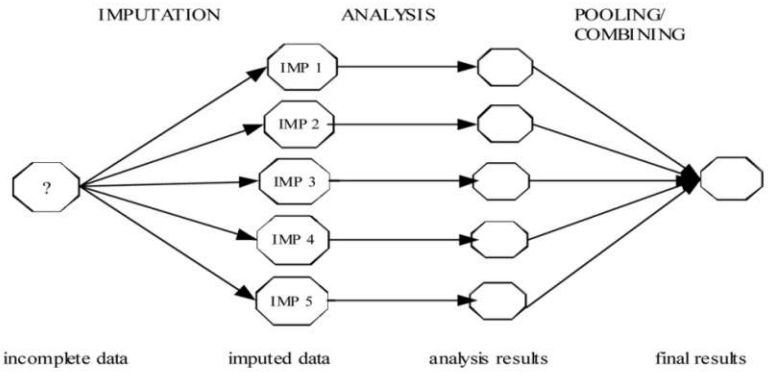


Figure 3.1: Three-step process of multiple imputation [Twa09]

containing only the actual true values for all the ones that are missing in X^{obs}. One could imagine stacking the two matrices on top of each other to get X. Furthermore, assume that $X^{(h)}$ denotes the hth of the m imputed data sets. The $p-1$ variables, where variable j is excluded, are denoted as $X_{-j} = (X_1, ..., X_{j-1}, X_{j+1}, ..., X_p)$ and the quantity or model of scientific interest is denoted by Q. These can be, for example, regression coefficients.

Therefore, when applying multiple imputation, m imputed data sets $X^{(1)}, ..., X^{(m)}$ are created and for each of them an analysis is done to receive an estimate for the quantity of interest, that is, $\hat{Q}^{(1)}, ..., \hat{Q}^{(m)}$. These m estimates are pooled into one estimate, denoted by $\bar{Q}$, by e.g. calculating the mean in case of approximately normally distributed quantities of interest. [VBGO11] In case of decision trees the same approach as for ensemble learning is followed for pooling the results of the individually trained trees, as described in detail in Chapter 4.5. [Twa09]

A very common approach to specify the imputation model needed in step one of the multiple imputation process is the technique of *chained equations*. The method is then called *multivariate imputation by chained equations* (MICE). It defines the imputation model separately for each variable containing missing values by conditional densities. After starting with an initialization, imputations are drawn by iterating over these conditional densities. Often only a few iterations are needed.

In more detail, let $P(X|\theta)$ be the p-variate distribution, from which X is a partially observed random sample, that is hypothetically complete. Further, let's assume that this distribution is completely specified by θ, which is an unknown vector. In the MICE method the posterior distribution of θ is then derived by iteratively sampling from the conditional distributions

$$\begin{aligned} &P(X_1|X_{-1}, \theta_1) \\ &\vdots \\ &P(X_p|X_{-p}, \theta_p). \end{aligned} \tag{3.6}$$

The parameters $\theta_1, ..., \theta_p$ in the conditional distributions are different for the individual conditional densities and their product may differ from the joint distribution. The initial imputation is done by drawing from the observed marginal distributions, while a Gibbs sampler defines the tth iteration of chained equations by drawing

$$\begin{aligned}
\theta_1^{*(t)} &\sim P(\theta_1 | X_1^{\text{obs}}, X_2^{(t-1)}, ..., X_p^{(t-1)}) \\
X_1^{*(t)} &\sim P(X_1 | X_1^{\text{obs}}, X_2^{(t-1)}, ..., X_p^{(t-1)}, \theta_1^{*(t)}) \\
&\vdots \\
\theta_p^{*(t)} &\sim P(\theta_p | X_p^{\text{obs}}, X_1^{(t)}, ..., X_{p-1}^{(t)}) \\
X_p^{*(t)} &\sim P(X_p | X_p^{\text{obs}}, X_1^{(t)}, ..., X_p^{(t)}, \theta_p^{*(t)}),
\end{aligned} \tag{3.7}$$

one after the other. A Gibbs sampler is a Markov chain Monte Carlo algorithm that is usually used to create a sample approximated from a multivariate probability distribution, where directly sampling is not possible. $X_j^{(t)} = (X_j^{\text{obs}}, X_j^{*(t)})$, which is the observed variable completed with the imputations $X_j^{*(t)}$ for the contained missing values, denotes the imputed variable j at iteration t. In the formulas above one can observe that in $X_j^{*(t)}$ the imputations $X_j^{*(t-1)}$ appear only indirectly through the relation of $X_j^{*(t)}$ with other variables. The algorithm typically converges after 10-20 iterations. The algorithm is quite powerful and can work with variables of mixed types. However, the computational speed can be an issue for large data sets.

In `R`, the method is implemented in the package `mice` (see Chapter 5.2). By using the function `mice`, multiple imputations by chained equations can be received. With the function `with` of the same package one can apply an analysis to all imputed data sets and the results can be combined by the function `pool`. [VBGO11]

CHAPTER 4

Decision Tree/Random Forest Missing Data Handling Techniques

In this chapter existing decision tree and random forest methods for classification are explored and analyzed. Each method has a specific built-in strategy to handle missing values in the data. By making use of them, it is not required to have a separate process for handling the missing values and learning the classifier.

4.1 CART - Classification and Regression Trees

The well-known CART - Classification and Regression Trees - algorithm, introduced by Breiman et al. in 1984 [BFSO84], constructs trees by recursive partitioning. Binary trees, that are trees where every node has at most two child nodes, are built by a two-step procedure: In the first step a single variable is found that best splits the data in a node - the definition of best used by CART is stated in Section 4.1.1 - and then the observations are separated into two subsets (nodes) defined by a split. Subsequently, these steps are applied again to each of these subsets separately and recursively repeated until no improvement can be made or until a stopping criterion is reached, such as a minimum number of observations in the final subsets. After the tree has reached its full size, it is trimmed back again in the second step of the procedure. By using cross-validation, estimates of the risk for trees of different sizes are computed and the smallest tree with the lowest estimate of risk is chosen. [TA+97]

The fundamentals of CART are implemented in the packages `rpart` [TARR22] and `tree` [RR16] of the `R` software for statistical computing (see Table 5.2). The two implementations differ mainly in its handling of missing data. More details about this are given in Section 4.1.2.

4.1.1 Building the Tree

For splitting the observations in a node of the tree into two distinct subsets a variable that best splits the data has to be found. 'Best' is based on a splitting criterion, that gives a measure of the purity of a node. Purity refers to the homogeneity of the instances within a node with respect to their target values. In the context of CART, the purity is defined and measured with either the Gini impurity or the information impurity, and is specified in the following paragraph.

CART is building the tree by choosing the split with maximum impurity reduction. For a node A that is split into two child nodes A_L (left) and A_R (right) the impurity reduction is given by

$$\Delta I = P(A)I(A) - P(A_L)I(A_L) - P(A_R)I(A_R), \tag{4.1}$$

whereby P denotes the probability of node A for future observations and I denotes the impurity of node A. For some impurity function f, the impurity for a node A is defined as

$$I(A) = \sum_{k=1}^{K} f(p_{kA}). \tag{4.2}$$

p_{kA} is the proportion of observations in A that belong to class k for future samples and K is the number of classes.

When A is completely pure and thus all observations in A belong to the same class, the equation $I(A) = 0$ should be fulfilled, and therefore the impurity function f must be concave with $f(0) = f(1) = 0$.

Two options for f proposed in CART are the *Gini index* and the *information index*. The Gini index is given by

$$f(p) = p(1-p) \tag{4.3}$$

while the information index is defined as

$$f(p) = -p \log(p). \tag{4.4}$$

Here p denotes the probability of a given class. For a two-class problem, the two impurity functions are nearly the same. Both can also handle multi-class problems and work well in practice.

While Gini is used as default splitting criterion for both R packages `tree` and `rpart`, there is also the option to use the information index in `rpart`.

4.1.2 Missing Data

Instead of just discarding all observations containing missing attribute values, CART is able to cope in a more ambitions way. When training a classification tree, two steps are affected by missing values: the step of finding a variable that best splits the data in a

node and, once a splitting variable and split point are selected, the step of separating the observations into two different subsets when the values of that relevant attribute might be missing.

For CART, all observations with at least one value for an explanatory variable are used for modelling. The criterion to be maximized when looking for a split of node A into two child nodes A_L and A_R is still the impurity reduction

$$\Delta I = P(A)I(A) - P(A_L)I(A_L) - P(A_R)I(A_R). \tag{4.5}$$

While the term $P(A)I(A)$ is equal for all variables and splits, no matter if there are missing values or not, the terms $P(A_L)I(A_L)$ and $P(A_R)I(A_R)$ are adjusted in case of missing values. For the impurity measures $I(A_L)$ and $I(A_R)$ as well as for the calculation of the probabilities $p(A_L)$ and $p(A_R)$ only the observations with known values for the relevant attribute are taken into account. Then the probabilities are modified in a way that the sum is equal to $p(A)$. This guarantees that the sum of the probabilities of the terminal nodes is equal to 1.

Once a split is selected, CART uses *surrogate variables* to decide how to separate the observations with missing values for the split variables. Surrogate variables are variables used as alternative split variables that replace the original split variables as good as possible. For finding the surrogate variables the partitioning algorithm is applied again to predict the two categories defined by the split with the help of the other explanatory variables. An optimal split point and the corresponding misclassification error are determined for each explanatory variable. In addition, the prediction based on the majority denoted as 'go with the majority' is evaluated. For this blind rule the misclassification error is $\min(p, (1-p))$ with $p = (\#$ in A assigned to $A_L)/|A|$, where $|A|$ is the number of observations in node A. All surrogate variables that predict worse than the 'go with the majority' rule are tossed out and the remaining ones are ranked by misclassification error.

So, when the attribute value of an observation for the original splitting variable is not known, then the first surrogate variable is used instead. If this value is missing too, the next best surrogate variable is invoked and so on. If all surrogate variables are missing for an observation, then the 'go with the majority' rule is used. [TA+97]

In the `R` package `tree` surrogate variables are not implemented. Observations with missing values for the splitting variable are simply not sent further down the tree. [RR16] In the package `rpart` three different options are given for handling the observations with missing values for the split variable. Observations can either be not sent further down the tree as in the package `tree` or they can be sent down with the help of surrogate variables. If for an observation the values for all surrogate variables are missing, there are two options: the observation is not sent further down or the observation is sent down the tree with the 'go with the majority' rule. The last opportunity is the default in `rpart` and suggested by CART. [TARR22]

4.2 Conditional Inference Trees

Conditional inference trees were introduced by Hothorn et al. in 2006 [HHZ06]. They use recursive binary partitioning, that is, recursively building binary trees like CART, embedded in the theory of permutation tests by Strasser and Weber (1999) [SW99]. The relation of the explanatory variables to the response variable is determined via the conditional distribution of statistics and is the basis for selecting the split variables among all the variables. This enables a comparison independent of the different variable scales and overcomes the bias towards continuous variables or variables with many different outcomes. After the strongest relation between a variable and the response is found, multiple test procedures are used to check the significance. The recursion stops as soon as no further significant association between any variable and the response can be found.

Conditional inference trees are implemented in `R` in the function `ctree` (see Table 5.2). While the original implementation is available in the `R` package `party`, the new and improved reimplementation can be found in the package `partykit`. Only the latter one will be improved and developed further in the future. [HHZ15]

4.2.1 Building the Tree

Let the response variable Y be from the sample space $\mathcal{Y}$ and the variable vector $X = (X_1, ..., X_p)$ consisting of the p explanatory variables from the sample space $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_p$, while both of them can be of different scales. The conditional distribution of the response variable, given the p explanatory variables, is denoted by $D(Y|X)$ and dependent on a function f, so that

$$D(Y|X) = D(Y|X_1, ..., X_p) = D(Y|f(X_1, ..., X_p)). \tag{4.6}$$

It is assumed to have partition based regression relationships, so the variable space $\mathcal{X} = \cup_{l=1}^{r} B_l$ is partitioned by the disjoint sets $B_1, ..., B_r$. Based on the training set $\mathcal{L} = \{(x_1, y_1), ..., (x_n, y_n)\}$ consisting of n independent and identically distributed observations with y_i denoting the response and $x_i = (x_{i1}, ..., x_{ip})$ denoting the values for the p explanatory variables for observation i, a regression relationship model needs to be trained.

Recursive binary partitioning can be formulated with the use of case weights $w = (w_1, ..., w_n)$ for the individual observations, which are non-negative integer values. So, each node in the tree consists of case weights w_i, $i = 1, ..., n$, where each weight w_i is non-zero if the observation i is part of the node and zero otherwise. The steps are:

1. Given the weights w_i in a node, the global null hypothesis that the response variable is independent of all p explanatory variables is tested. The algorithm is stopped if this cannot be rejected. If this hypothesis can be rejected, the variable X_{j^*} that is most strongly related to the response is selected.

2. X_{j^*} is splitted into two sets A^* and $X_{j^*} \setminus A^*$ and the case weights of the two generated subsets are calculated via $w_{\text{left},i} = w_i \mathbb{I}(x_{ij^*} \in A^*)$ and $w_{\text{right},i} = w_i \mathbb{I}(x_{ij^*} \notin A^*)$ with the indicator function $\mathbb{I}$. x_{ij^*} refers to the value of observation i for variable X_{j^*}.

3. The steps 1. and 2. are repeated recursively with updated weights for each round.

With this algorithm the bias to variables with many possible outcomes is overcome. A partition of the variable space $\mathcal{X}$ into $\{B_1, ..., B_r\}$ is created, where a vector of case weights w exists for each B_l.

The first two steps of this recursive binary partitioning algorithm are examined now in more detail. By the usage of the permutation test framework by Strasser and Weber (1999) [SW99] independence tests by means of the conditional distribution of linear statistics, that are explained in detail in the following section, are conducted and the best split point is determined based on standardized linear statistics. [HHZ15]

Variable Selection

In each node that consists of case weights w, the global null hypothesis $H_0 = \cap_{j=1}^{p} H_0^j$ with the p partial hypotheses $H_0^j : D(Y|X_j) = D(Y)$, which is checking the independence between the response variable and the p explanatory variables, is tested in the first step of the algorithm. If the global null hypothesis cannot be rejected at a predefined significance level α, the recursion is stopped. If it is rejected, the relations between the response Y and all the p variables X_j are measured. This is done by test statistics or p-values specifying the deviation of the partial hypotheses H_0^j.

Assume that the case weights are either 0 or 1 and let $S(\mathcal{L}, w)$ be the group of all permutations of the elements $(1, ..., n)$ with case weights $w_i = 1$, $i = 1, ..., n$. This group of permutations is symmetric. The linear statistic that is specifying the relation between Y and X_j is given by

$$T_j(\mathcal{L}, w) = \text{vec}\left(\sum_{i=1}^{n} w_i g_j(x_{ij}) h(y_i, (y_1, ..., y_n))^T\right) \in \mathbb{R}^{m_j q}, j = 1, ..., p. \tag{4.7}$$

$g_j : \mathcal{X}_j \to \mathbb{R}^{m_j}$ represents a transformation of X_j of the form $g_{ji}(x) = x$ if variable X_j is numeric and $g_{ji}(t) = e_T(t)$ if variable X_j is categorical with T different classes. $e_T(t)$ is a T dimensional unit vector with a value equal to 1 at the t-th position. $h : \mathcal{Y} \times \mathcal{Y}^n \to \mathbb{R}^q$ denotes an influence function that is based on the responses in a permutation symmetric way. If K is the number of different classes, the influence function is defined as $h(y_i, (y_1, ..., y_n)) = e_K(y_i)$, which is a K dimensional unit vector with 1 at the y_i-th element. The function 'vec' transforms a matix into a column vector by columns. In Equation (4.7), the matrix with dimension $(m_j \times q)$ is transformed via the function 'vec' to a vector with dimension $m_j q$.

The distribution of $T_j(\mathcal{L}, w)$ under the partial null hypothesis H_0^j can be derived by fixing the variables X_j and having a condition on all potential permutations of the responses.

This results in so-called *permutation tests*. Given all these permutations $S(\mathcal{L}, w)$, Strasser and Weber (1999) [SW99] defined the conditional expectation of $T_j(\mathcal{L}, w)$ as

$$\mu_j = \mathbb{E}(T_j(\mathcal{L}, w)|S(\mathcal{L}, w)) = \text{vec}\left(\left(\sum_{i=1}^{n} w_i g_j(x_{ij})\right)\mathbb{E}(h|S(\mathcal{L}, w))^T\right) \in \mathbb{R}^{m_j q}, \tag{4.8}$$

while the influence function has the conditional expectation

$$\mathbb{E}(h|S(\mathcal{L}, w) = w_.^{-1} \sum_{i=1}^{n} w_i h(y_i, (y_1, ..., y_n)) \in \mathbb{R}^q \tag{4.9}$$

with $w_. = \sum_{i=1}^{n} w_i$.

The conditional covariance of $T_j(\mathcal{L}, w)$ under the H_0 is given as

$$\begin{aligned}\Sigma_j &= \mathbb{V}(T_j(\mathcal{L}, w)|S(\mathcal{L}, w)) \\ &= \frac{w_.}{w_. - 1}\mathbb{V}(h|S(\mathcal{L}, w)) \otimes \left(\sum_{i=1}^{n} w_i g_j(x_{ij}) \otimes w_i g_j(x_{ij})^T\right) \\ &- \frac{w_.}{w_. - 1}\mathbb{V}(h|S(\mathcal{L}, w)) \otimes \left(\sum_{i=1}^{n} w_i g_j(x_{ij})\right) \otimes \left(\sum_{i=1}^{n} w_i g_j(x_{ij})\right)^T \in \mathbb{R}^{m_j q \times m_j q}\end{aligned} \tag{4.10}$$

with

$$\begin{aligned}\mathbb{V}(h|S(\mathcal{L}, w)) = w_.^{-1} \sum_{i=1}^{n} w_i &(h(y_i, (y_1, ..., y_n)) - \mathbb{E}(h|S(\mathcal{L}, w))) \\ &(h(y_i, (y_1, ..., y_n)) - \mathbb{E}(h|S(\mathcal{L}, w)))^T \in \mathbb{R}^{q \times q}\end{aligned} \tag{4.11}$$

and $\otimes$ denoting the Kronecker product.

Then a univariate, standardized test statistic c depending on the observed multivariate linear test statistic $t \in \mathbb{R}^{mq}$, specified with Equation (4.7) for a $m \in \{m_1, ..., m_p\}$, can be formulated by

$$c_{\max}(t, \mu, \Sigma) = \max_{s=1,...,mq} \left|\frac{(t - \mu)_s}{\sqrt{(\Sigma)_{ss}}}\right|, \tag{4.12}$$

using the conditional expectation μ and the conditional covariance Σ for the considered variable, where $(.)_s$ denotes the element of the input vector on position s and $(.)_{ss}$ denotes the element of the input matrix on position ss.

Since the explanatory variables may be of different scale, the test statistics $c(t_j, \mu_j, \Sigma_j)$ for $j = 1, ..., p$ cannot be compared directly without creating a bias. However, p-values for the conditional distribution of the test statistics can be compared among variables with different scales in an unbiased way and therefore the partial null hypotheses H_0^j are tested with the use of p-values. Thus, in step 1 of the procedure the variable with strongest association to the response is identified by selecting the variable X_{j^*} with

smallest p-value of the conditional test for H_0^j, that is X_{j^*} with $j^* = \text{argmin}_{j=1,...,p} P_j$ and

$$P_j = \mathbb{P}_{H_0^j}(c(T_j(\mathcal{L}, w), \mu_j, \Sigma_j) \geq c(t_j, \mu_j, \Sigma_j)|S(\mathcal{L}, w)). \tag{4.13}$$

However, before testing each partial hypothesis H_0^j in step 1 of the procedure, the global null hypothesis H_0 needs to be tested to check if there is an association between any of the explanatory variables and the response at all or if the recursion needs to stop. Aggregating the transformations g_j, $j = 1, ..., p$, leads to the following linear statistic to test the H_0.

$$T(\mathcal{L}, w) = \text{vec}\left(\sum_{i=1}^{n} w_i (g_1(x_{i1})^T, ..., g_p(x_{ip})^T)^T h(y_i, (y_1, ..., y_n))^T\right) \tag{4.14}$$

As this approach is inconvenient when dealing with missing values, test procedures on the basis of the p-values P_j, $j = 1, ..., p$, are used. Applying a Bonferroni-adjustment on the p-values as $1 - (1 - P_j)^p$ at first, the global null hypothesis H_0 is rejected when the smallest adjusted p-value is less than α, which is a predefined level. If the null hypothesis cannot be rejected the recursion needs to stop. Therefore, the parameter α may also be seen as control parameter for the size of the tree. [HHZ15]

Variable Split Point

Once the variable X_{j^*} that best splits the data is selected, the optimal split point that separates the data of a node into two subsets needs to be found in step 2 of the procedure. This is done by considering the goodness of a split that is assessed by two-sample linear statistics similar as in Equation (4.7). The discrepancy between the sets $\{y_i|w_i > 0 \wedge x_{ij} \in A; i = 1, ..., n\}$ and $\{y_i|w_i > 0 \wedge x_{ij} \notin A; i = 1, ..., n\}$ is determined by a two-sample statistic that is generated by the linear statistic

$$T_{j^*}^A(\mathcal{L}, w) = \text{vec}\left(\sum_{i=1}^{n} w_i I(x_{ij^*} \in A) h(y_i, (y_1, ..., y_n))^T\right) \in \mathbb{R}^q \tag{4.15}$$

for all subsets A of the sample space $\mathcal{X}_{j^*}$, while the conditional expectation $\mu_{j^*}^A$ and the conditional covariance $\Sigma_{j^*}^A$ are calculated in analogous to Equations (4.8) and (4.10). So, for all subsets A the test statistics $c(t_{j^*}^A, \mu_{j^*}^A, \Sigma_{j^*}^A)$ are evaluated and the subset A^* with the maximum test statistic is chosen for the split, that is,

$$A^* = \text{argmax}_A c(t_{j^*}^A, \mu_{j^*}^A, \Sigma_{j^*}^A). \tag{4.16}$$

[HHZ15]

4.2.2 Missing Data

Conditional inference trees can deal with missing values in the explanatory variables in a similar way as CART. Both, step 1 and step 2, of the procedure of learning a conditional

inference tree or generally of recursive binary partitioning are affected when there are missing values in the data.

Let the value x_{ij} of the variable X_j be missing for some $i \in \{1, ..., n\}$ and $j \in \{1, ..., p\}$. Then in the first step of the procedure when calculating $T_j(\mathcal{L}, w)$, only the observations with known values for X_j are taken into account. Therefore, the case weights w_i for the observations with missing values are set to zero in the calculation of $T_j(\mathcal{L}, w)$. Similarly, once a split variable X_j is selected the case weights w_i are set to zero when calculating $T_j^A(\mathcal{L}, w)$ in the second step of the procedure.

As soon as the subset A^* that is defining the split for the split variable X_j is selected, conditional inference trees use *surrogate variables* to separate the observations with missing values for X_j. So, as already stated for CART, surrogate variables with corresponding split points are variables that produce the split that is most similar to the original split. For evaluating them, the binary variable $\mathbb{I}(x_{ij} \in A^*)$ is used as response variable and predicted with the help of the other explanatory variables using the same procedure as described before. Then, when for an observation i the value x_{ij} for the original split variable is missing, the first surrogate variable is used to assign it to a subset. If the value for this is missing too, then the second surrogate is used and etc. [HHZ15]

While in the package `party` only surrogate variables can be used to deal with the missing values, there is an additional approach implemented in the package `partykit`. It is called *missingness incorporated in attributes* (MIA). [HZH23]

MIA is a generalization of the separate class method defined in Chapter 4.4. While MIA can be used for every decision tree classification method, it is explained here with the notation of conditional inference trees. Let X_j be the split variable for a node in the decision tree. Then the approach is considering 3 possible splits. While all the observations with missing values for variable X_j are handled as separate group, this group can be either put to one of the two subsets defined for the observations with no missing values or can define a separate split based on the missingness information. So, when deciding on a split, one out of the following 3 options is selected for a subset A^* of the sample space of X_j.

- Split A: $\{x_{ij} \in A^* \text{ or } x_{ij} \text{ missing}\}$ vs. $\{x_{ij} \notin A^*\}$
- Split B: $\{x_{ij} \in A^*\}$ vs. $\{x_{ij} \notin A^* \text{ or } x_{ij} \text{ missing}\}$
- Split C: $\{x_{ij} \text{ missing}\}$ vs. $\{x_{ij} \text{ not missing}\}$

While MIA is a rather simple method, the literature shows that it can also be extremely effective. [TJH08]

4.3 C4.5 and C5.0 Decision Trees

Quinlan's C4.5 algorithm is an advancement of the so-called ID3 algorithm of Quinlan (1986) [Qui86] and C5.0 decision trees are again an advancement of the C4.5 decision

trees. The ideas to build and train a C4.5 classification tree are similar as for CART. By applying the so-called divide-and-conquer algorithm a variable that best splits the data in a node, denoted as *test* by Quinlan, and multiple split points are found and then the observations are separated into the multiple subsets defined by the split. Once the subsets are created the same steps are again applied separately to all of them and repeated recursively until all observations are from the same class or until a stopping criterion is reached. C4.5 decision trees can also handle missing values in a more ambitious way than just deleting the observations with missing values. But instead of using surrogate variables like CART, C4.5 uses so-called *fractional cases* (observations) to treat the missing values, which are defined in Section 4.3.2. [Qui93]

In `R` two implementations of such algorithms are available. In the package `RWeka` the function `J48` fits a C4.5 decision tree, while a C5.0 decision tree can be generated with the function `C5.0` of the package `C50` (see Table 5.2). [HBH+23] [KWC+23]

C5.0 decision trees rely on the same principles as C4.5 decision trees, however, they exhibit some advances. Compared to C4.5, C5.0 shows improvements in speed and memory usage. While C5.0 performs similar to C4.5, its built decision tree is substantially smaller. C5.0 also allows the usage of methods to improve the tree and of weighting of different variables and misclassification types. [PT13]

4.3.1 Building the Tree

To find the variable that best splits the data in a node A with corresponding split points, Quinlan (1993) [Qui93] relied on information-based methods, that are explained in the following.

The original ID3 algorithm utilizes the information gain as splitting criterion. The information gain measures the information that is gained by the partition of a node A defined by a split on variable X and is given by

$$gain(X) = info(A) - info_X(A). \tag{4.17}$$

In this formula, $info(A)$ denotes the average amount of information that is needed for the class identification of an observation in node A and can be derived as follows: Assume that one observation is randomly chosen from set A of observations and assume that it's from class k. Then the probability of this message is given as

$$\frac{freq(k, A)}{|A|} \tag{4.18}$$

where $freq(k, A)$ is the number of observations in node A belonging to class k and $|A|$ is the total number of observations in node A. The information of the message can be calculated as

$$-\log_2\left(\frac{freq(k, A)}{|A|}\right). \tag{4.19}$$

according to the information theory. [Kul97] It is measured in bits.

The sum over all K classes relatively to their frequencies in A yields the expected information from such a message:

$$info(A) = -\sum_{k=1}^{K} \frac{freq(k,A)}{|A|} * \log_2\left(\frac{freq(k,A)}{|A|}\right) \text{ bits} \tag{4.20}$$

This measure is also called *entropy*.

On the other hand, $info_X(A)$ denotes the expected information after separating the observations in A according to split X into m different subsets that form the nodes A_i. It is given by

$$info_X(A) = -\sum_{i=1}^{m} \frac{|A_i|}{|A|} * info(A_i). \tag{4.21}$$

All in all, the ID3 algorithm selects the variable X for which the information gain is maximized and declares it to be the split variable at node A.

Using the information gain as splitting criterion, however, produces a bias in favor of variables with many different output capabilities. To overcome this issue, the C4.5 algorithm uses the information gain ratio as splitting criterion instead. In the information gain ratio, the gain attributed to such a variable with many outcomes is revised by a normalization. Let's assume to select one observation at random from set A and to announce that it belongs now to a specific outcome of the splitting variable X instead of announcing the class. Similar to Equation (4.20) the expected information produced by splitting A into m subsets A_i can be defined as

$$split\ info(X) = -\sum_{i=1}^{m} \frac{|A_i|}{|A|} * \log_2\left(\frac{|A_i|}{|A|}\right). \tag{4.22}$$

The information gain ratio is then given by

$$gain\ ratio(X) = gain(X)/split\ info(X). \tag{4.23}$$

Finally, C4.5 selects the split for which this ratio is maximized with the constraint that the information gain is large. The additional constraint ensures that the ratio will not be unstable for splits that are nearby trivial, since for such splits the split information is small. [Qui93]

According to Quinlan (1993) [Qui93], the information gain ratio is more robust and chooses better splits than the gain ratio.

4.3.2 Missing Data

C4.5 is dealing with missing values in the data by taking two questions into account when learning a classification tree. The first question that needs to be addressed is how

to compute the splitting criterion and how to take the number of missing values for the specific variables into account when weighting their attractiveness for being the split variable. The second question is how to treat the observations with missing value for the split variable in the partitioning step.

For choosing the variable that best splits the data in a node A that can contain missing values for some attributes, an adjusted form of the information gain ratio is used. At first, the measures $info(A)$ and $info_X(A)$ are calculated only over the observations with known values for the relevant attribute that performs the split. Then the information gain is adjusted by

$$gain(X) = F * (info(A) - info_X(A)) \tag{4.24}$$

where F denotes the fraction of observations in A with known value for the relevant attribute. So, the information gain looking only at the observation with known values is adjusted by the fraction of such observations. The split information, $split\ info(X)$, is calculated by treating the missing values as additional group. So, when a variable is splitting the data into m subsets, the information split is calculated over $m + 1$ subsets. The final information gain ratio is then calculated as in Equation (4.23) with the adjusted measures.

Once a split variable is selected based on the information gain ratio, a probabilistic approach is used to separate the observations. Assume that based on the split variable and split points, a node A is split into subsets A_i with outcomes O_i of the split variable. For each observation of node A, probabilities are measured with that the observation belongs to each subset A_i of A. These probabilities are also called weights and are assigned as follows: If the value O_i of the split variable is known for an observation and if it is assigned to subset A_i, then the probability that the observation is part of A_i is 1 and the probability that this observation belongs to any of the other subsets is 0. If the value of the split variable for an observation is not known, then only weaker and not clear assignments to the individual subsets A_i via weights can be made. The weight of an observation for each subset A_i is the probability of outcome O_i at A, for which the estimate is given below. Each subset A_i consists therefore of a collection of weights which indicate the probabilities with that the observations belong to this subset and $|A_i|$ in the formulas above needs to be interpreted as the sum of the fractional weights of all observations in A_i.

Since A might be a subset of an earlier partition and therefore might consist of nonunit weights for the observations, the new weight for an observation with weight w where the outcome O_i of the split variable is not known is calculated by

$$w * \text{probability of outcome } O_i. \tag{4.25}$$

The probability of outcome O_i can be estimated by the sum of the weights of the observations in A with known value for the split variable and outcome O_i, divided by the sum of the weights of all observations in A with known value for the split variable. [Qui93]

This strategy of handling the missing attribute values in the data is often called *fractional cases*. [Fee99]

4.4 Separate Class Method/Null Value Strategy

A strategy to deal with missing values in the data that is not dependent on a particular decision tree algorithm is called *separate class method* or *null value strategy*. It treats missing values as a separate class or value. When the variable with the missing values is categorical, then the missing values are just simply replaced with a new category denoted for example as 'missings'. When the variable with missing values is numerical, then the missing values are replaced by a new value that is clearly outside the range. This gap between the original data range and this extreme value ensures the separation between the missing and non-missing attribute values. [DS10]

This method has some disadvantages since it deals with the missing values just like with the non-missing values and does not try to determine the actual true value. Furthermore, it treats all missing values as 'missing' although there might be more than one actual missing value category. [GS15] Advantages are, however, that the method is easy to implement and use and can deal with missing values at training and testing time. According to Ding and Simonoff (2010) [DS10], this technique has the best performance when attribute values are missing not at random and the missingness is dependent on the response variable.

4.5 Random Forests

Random Forests were introduced by Breiman in 2001. [Bre01] A random forest is a combination of decision trees such that each tree depends on a randomly selected subset of the data. It is called to be a tree-based ensemble.

While the aim is to predict the response Y with a function f of the given input data X, the function f is constructed by an ensemble of so-called *base learners* $h_1, ..., h_J$, where J is the number of base learners. These base learners are combined by so-called 'voting' for classification and form the *ensemble predictor* f. Constructing such ensembles of base learners can substantially improve the classification performance.

In random forests each of these base learners is a decision tree, denoted as $h_j(X, \Theta_j)$, $j = 1, ..., J$. All Θ_j's represent a set of random variables and they are all independent of each other. Let $\mathcal{L} = \{(x_1, y_1), ..., (x_n, y_n)\}$ be the training data with y_i denoting the response and $x_i = (x_{i1}, ..., x_{ip})^T$ denoting the values for the p explanatory variables for observation i. For fitting the base learner decision trees, independent bootstrap samples $\mathcal{L}_j$, $j = 1, ..., J$, of size n are taken from the training data $\mathcal{L}$. For each of these bootstrap samples a decision tree is fitted by using binary recursive partitioning and the Gini index as splitting criterion, similar as for CART explained in Section 4.1. The only difference is that the best split for a node in the decision tree is not found by a search over all p

variables but rather by a search over only m variables that were randomly selected from the p explanatory variables. This random selection is done independently at each node of the tree. Pruning is done for none of the trees, since the two layers of randomization applied when building the individual trees create a diverse forest that already prevents overfitting. While Breiman stated to grow each of the trees until each leaf node is pure, that is, each leaf node contains only observations with the same class label, more recent literature stated to use a maximum number of leaf nodes as stopping criteria.

The finally fitted trees are denoted by $\hat{h}_j(x, \theta_j, \mathcal{L})$, $j = 1, ..., J$, where x denotes input data for which the tree can be used to make a prediction and where θ_j is a realization of Θ_j, indicating the randomness in a tree.

To make a classification for a new instance x in the end, the J decision trees are combined by voting to form the ensemble predictor as

$$\hat{f}(x) = \text{argmax}_y \sum_{j=1}^{J} \mathbb{I}(\hat{h}_j(x) = y), \tag{4.26}$$

where $\hat{h}_j(x)$ denotes the prediction of the outcome variable for x with the j-th tree and where $\mathbb{I}(\hat{h}_j(x) = y) = 1$ if the prediction $\hat{h}_j(x)$ is equal to y and 0 otherwise. [CCS12]

CART decision trees handle missing values in the data with surrogate variables, however, this is not appropriate for random forests. The reason is that finding surrogate variables at each node is computationally expensive, especially when many trees are built. Moreover, when looking for the variable that best splits the data of a node for one base learner, the search is done over p randomly selected explanatory variables. Thus, it could be that there are no meaningful surrogate splits as the variables in a node may be uncorrelated. [IKBL08]

In Breiman's random forests missing values are handled by using proximities between the observations. The proximity between two observations in the training data is defined as the proportion of times in which the two were determined to be in the same leaf node among all trees of the forest. Their proximity is 1 if they end up in the same leaf node for all the trees. When the two observations are in two different leaf nodes for all the trees, their proximity is 0. Thus, the proximity defines the closeness of two observations in the explanatory variable space, while the variables with high importance for predicting the response get more weight.

The proximities between all observations are used to impute missing values in random forests using an iterative procedure. Initially, all missing values are imputed by using median imputation, which is the replacement of all missing values of a variable with the median of that specific variable. Then a random forest is fitted and the proximities between all observations are calculated. With the use of the proximities, new imputations for the missing values are done by using the proximity-weighted average for the continuous variables and the proximity-weighted vote for the categorical variables. Then a new random forest is fitted, the new proximities are calculated for this new model and

new imputations for the missing values are done as before. These steps are iteratively repeated until stable results are achieved, which is usually already after a few iterations. So, random forests use a kind of proximity-based nearest neighbors imputation for dealing with the missing values in the data. This is an indication that the method is valid for data that contains missing values that are missing at random (MAR). [CCS12]

Breiman's random forest algorithm is implemented in the `R` package `randomForest` (see Table 5.3). While previous versions could not handle missing values, the above described strategy of using proximities for imputation should be implemented from version 4.0 on and possible to use with the function `rfImpute` from the same package. [BC11]

Imputing missing values with proximities has several advantages, as this strategy works without specific modifications of the random forest algorithm. Moreover, proximity imputation utilizes the property of random forests of clustering the data. However, this strategy has also some drawbacks. Firstly, the out-of-bag (OOB) error rate is biased, as it is overestimated. This out-of-bag error rate is the average error for the observations using the predictions only from the trees that do not contain this observation in their bootstrap sample. Further, the random forest cannot classify test observations with missing values.

To overcome these issues, Ishwaran et al. (2008) [IKBL08] found a new algorithm to handle missing values in Breiman's random forests. They developed an approach called *adaptive tree imputation*. When a tree of the forest is grown, missing data are imputed by drawing randomly from all the nonmissing data that are contained in the respective bootstrap sample, anew at each node. The data in a specific bootstrap sample are also called in-bag data. Since for the missing data imputation in one tree, only the in-bag data are used, the OOB error rate is not biased as for proximity imputation. It is assumed that only a limited number of all variables contain missing values. Then the following steps are done:

1. Assume to be at node A for one tree in the training process. Before the split is performed, the missing values of the data in the node are imputed. For each variable z of the variables that contain missing values, let $S_{z,A}$ be the set of nonmissing values for variable z that are contained in the bootstrap sample of that specific tree with the empirical distribution function $\mathbb{F}_{z,A}$. For each observation in node A with missing value for the variable z, the value is imputed with a random value from $\mathbb{F}_{z,A}$. As soon as all missing values are imputed, the split of the data in the node into two subsets is performed as without having any missing values. So, both the imputation and the performance of the split is only based on the in-bag data.

2. The OOB data is also imputed by randomly drawing from $\mathbb{F}_{z,A}$ but does not play an active role in the learning process of the tree.

3. As soon as the split is performed the imputed values are again considered as missing in the child nodes of A and the steps 1 and 2 are repeated for each of the created

subsets separately. These steps of imputing and splitting are repeated until no more improvement in a node of the tree can be made or until a stopping criterion is reached.

The final imputation for a missing value in the data is considered to be created out of all in-bag imputations for this value in the terminal nodes across all trees of the forest. For missing values of continuous variables the average of the imputed in-bag values is taken, while for categorical variables the most frequently occurring in-bag value is chosen. When two imputed in-bag values occur equally often, then one of them is selected at random.

This strategy can also handle missing values in the test data. An observation from the test data is dropped down the tree while the missing value for variable z at node A of the tree is imputed by randomly drawing from $\mathbb{F}_{z,A}$.

The adaptive tree imputation algorithm can also be iteratively repeated, which makes sense if there are many missing values in the data, because the higher the amount of missing values the lower is the accuracy of the imputation. So, after applying the steps of adaptive tree imputation explained above, an initial forest is built. The missing values are then imputed with the OOB summary values. These are the averaged or frequently occurring imputations of the OOB data in the terminal nodes of all trees. Using the imputed data, a new forest is built. Then for each observation where a value was originally missing, a random value is drawn from all nonmissing in-bag observations in the same terminal node for each tree of the forest. By using these drawn values and the values of the OOB data the average or the frequently occurring value is used as imputation. This imputed data is used to grow a new forest and the steps are repeated iteratively. [IKBL08]

Breiman's random forest using the missing data imputation method from Ishwaran et al. is implemented in the `R` package `randomForestSRC` (see Table 5.3). It does fast parallel computing of random forests for survival analysis, regression, and classification. [IKK23]

4.6 Conditional Random Forests

A *Conditional Random Forest* is a tree-based ensemble like a Breiman's random forest described in Chapter 4.5 but where the base-learners $h_1, ..., h_J$ are conditional inference trees as described in Chapter 4.2.

After the J base-learners $\hat{h}_j(x, \theta_j, \mathcal{L})$, $j = 1, ..., J$ are fitted with the realizations θ_j of Θ_j, they are combined for making predictions. However, instead of averaging for a new observation the predictions from the J trees directly as in Breiman's random forests, in conditional random forests the observation weights are averaged.

When there are missing values in the data, conditional random forests deal with them by simply allowing the individual base learners to handle them. As the base learners are

conditional inference trees, missing values are handled by using either surrogate variables or the MIA approach as described in Section 4.2.2. When the base learners are fitted with the appropriate strategy for the missing values, they are combined to build the forest. As in this strategy each base learner must deal with the missing values separately, this approach is computationally very intensive, especially when a large number of trees is grown.

Since Breiman's random forests use CART decision trees as base learners, they are also biased in favor of variables with many different outcomes, whereas conditional random forests overcome this problem since they use conditional inference trees as base learners, which are unbiased regarding this.

In `R` conditional random forests are implemented in the function `cforest` (see Table 5.3). Similar as for conditional inference trees, the original implementation can be found in the package `party`, while the new implementation, which is written completely in `R`, is in the package `partykit`. [HZH23] According to Hothorn (2023) [HZH23], conditional random forests have not yet been tested in much detail and their use is rather experimental.

CHAPTER 5

Simulation Study

The use and comparison of the different missing data handling techniques of decision trees and random forests is explored in an extensive simulation study by using data from the UNIQA Insurance Group (UIG). The simulation study is thoroughly explained and then the results are discussed.

5.1 Data

UNIQA Insurance Group, as most large insurances, is host of several large data sets. The data owner of the provided data for the present study is the Natural Catastrophes Competence Center (NCCC) which belongs to the Group Actuarial Division. The data consists of a list of buildings and their physical attributes such as location, type of construction, number of floors, etc. and of insurance-based attributes such as sum insured, limits and deductibles, etc. As most real-world data sets, the provided data contains missing fields because of, for example, missing answers or failures in manual data entry processes. The missing building attribute the present study focused on is the occupancy of the building, that is, the usage of the building, e.g. bakery, school, family house, etc. Therefore, the aim of this study was to improve the data quality of the provided data set by predicting the missing occupancy attributes in the data.

The use and comparison of different techniques for dealing with missing values in the data while training a decision tree or random forest classifier are illustrated with the provided data.

The data consists of various information assigned per building. These are

- the location, thus country, postal code, street, number, and latitude and longitude coordinates

- the insurance conditions such as the value of the building, limits[1] and deductibles[2]
- the building use or occupancy
- building physical properties such as type of construction, number of floors, year built and existence of a basement
- a classification of the insurance line of business[3]

UNIQA provided three data sets, data set A, B and C, of this type. As a first step of data preparation, the three data sets A, B and C were merged. Only the insurance conditions such as limits and deductibles change from data set to data set. Therefore, the data set A was taken as a basis and columns with the insurance conditions given in data set B and C were added in addition to 3 new binary columns specifying if the buildings are included in each of the data sets or not.

The data was enriched with additional information, namely with the industry category and the type of business, that was extracted from an industrial-based database from UNIQA. The industry category for a building is assigned out of a predefined list of options such as food, electronic, etc. and the type of business is a description of the business done in the building. Each building has a unique identifier where the 3 first digits represent the insurance branche, which was stored as an additional categorical variable.

To meaningfully make use of the address information in the context of this study, the publicly available data set called Corine Land Cover (CLC) 2018[4] was used. The data set is coordinated by the European Environment Agency (EEA) and is part of the Pan-European component of the Copernicus Land Monitoring Service. It provides detailed information on land cover and land use status across Europe assigning every unit of area one of 44 CLC classes[5]. Using the exact latitude and longitude coordinates when available or the postal code when not, the address for each building in the data set was mapped to one of these 44 CLC classes. An illustration of this data can be seen in Figure 5.1.

In the original data there are more than 375 different occupancy classes. To avoid facing computational limitations, excessive complexity, or difficulties in evaluating performance or with highly imbalanced classes, the different occupancy classes had to be grouped to end up with a lower number of classes for the response variable. This was done with a mapping of the occupancy to the NACE code. NACE denotes the French term "nomenclature statistique des activités économiques dans la Communauté européenne" and

[1] https://www.insuranceopedia.com/definition/450/policy-limit (accessed 20.08.2023)
[2] https://en.wikipedia.org/wiki/Deductible (accessed 20.08.2023)
[3] https://en.wikipedia.org/wiki/Line_of_business (accessed 20.08.2023)
[4] https://land.copernicus.eu/pan-european/corine-land-cover/clc2018 (accessed 15.06.2023)
[5] https://land.copernicus.eu/user-corner/technical-library/corine-land-cover-nomenclature-guidelines/html/ (accessed 15.06.2023)

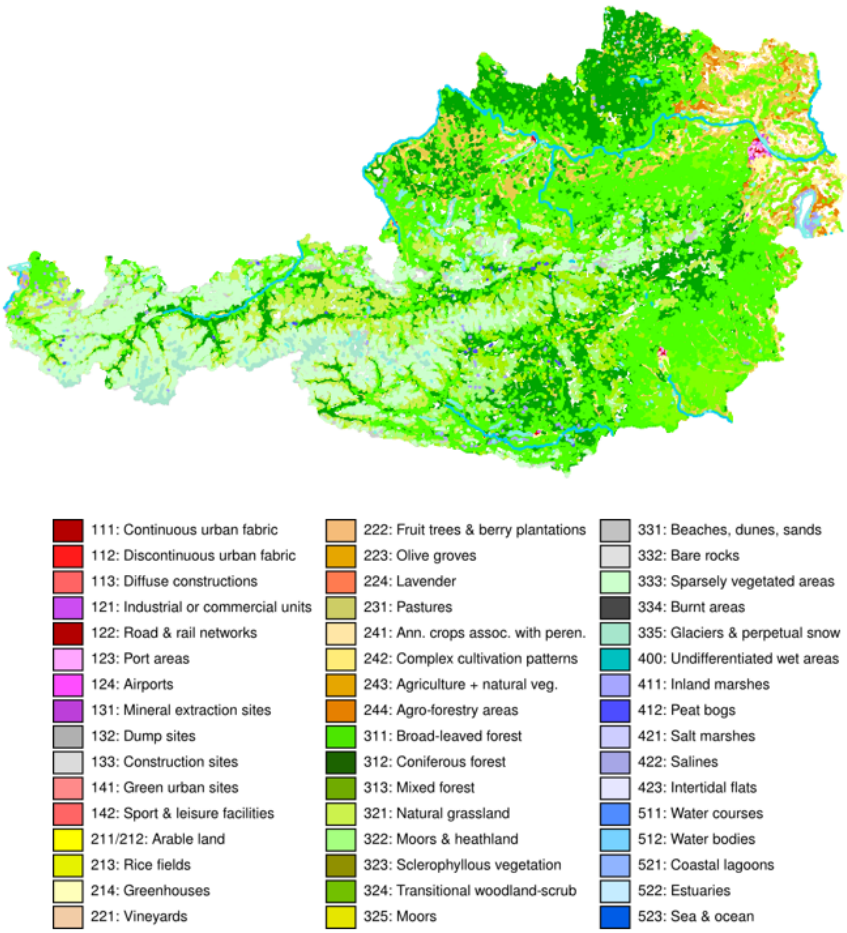


Figure 5.1: Corine land cover 2018

refers to the *Statistical Classification of Economic Activities in the European Community*. It is a classification system of economic activities in the Eurpean Union (EU) which uses four hierarchical levels. In this study only level 1 of the hierarchy was used, which consists of 21 sections identified by alphabetical letters from A to U. The list of the level 1 sections with a description can be found in Table 5.1. [Nac]

The mapping of the occupancy to the NACE code was done by taking information from several different resources. The first data source, that was used, is a commercial ESG database. Via the screening solution it was possible to extract all company names included in the database along with their NACE code. After adding the company name for the individual buildings as additional column to the UNIQA data set, the information of the NACE code from the commercial ESG database could be added. However, not all companies included in the data are also contained in the ESG database.

For all other buildings, for that the company name was not found in the ESG database,

Code	Economic Area
A	Agriculture, Forestry and Fishing
B	Mining and Quarrying
C	Manufacturing
D	Electricity, Gas, Steam and Air Conditioning Supply
E	Water Supply; Sewerage, Waste Management and Remediation Activities
F	Construction
G	Wholesale and Retail Trade; Repair of Motor Vehicles and Motorcycles
H	Transportation and Storage
I	Accommodation and Food Service Activities
J	Information and Communication
K	Financial and Insurance Activities
L	Real Estate Activities
M	Professional, Scientific and Technical Activities
N	Administrative and Support Service Activities
O	Public Administration and Defence; Compulsory Social Security
P	Education
Q	Human Health and Social Work Activities
R	Arts, Entertainment and Recreation
S	Other Service Activities
T	Activities of Households as Employers; Undifferentiated Goods and Services Producing Activities of Households for Own Use
U	Activities of Extraterritorial Organisations and Bodies

Table 5.1: Statistical classification of economic activities in the European community

the industrial-based database from UNIQA was used as a second data source and the NACE code was taken from there.

A third data source was used for completion. By taking the descriptions of the NACE code for all the different hierarchy levels, a list of the most discriminating keywords was created for each level 1 NACE code. These keyword lists were prepared using natural language processing. So, for each level 1 NACE code the descriptions of the codes of all the sublevels were joined at first and then tokenized. Case folding was applied to have all words in lowercase letters and all the stop words were removed. Afterwards, punctuation, numbers, dashes, and quotes were removed. To remove affixes from a word so that only the stem of a word is left, stemming was used. Since the occupancy in the data provided can be given in German or English, two different keyword lists were created per level 1 NACE code, one in German and one in English, and they were merged. After applying the same natural language processing functions to the occupancy column of the data set, it could be checked if the occupancy is included in one of the keyword lists for the different level 1 NACE codes. If the occupancy was uniquely found in one keyword list,

the corresponding NACE code was assigned. However, there were still buildings for that the occupancy was not mapped to a NACE code up to this step.

For these buildings the occupancy group was used, which is a side information in the data set denoting one group out of a predefined list of options for the occupancy of each building. By using the occupancy group a manual mapping of all the occupancies in a specific occupancy group to one NACE code could be defined. For all buildings with no given occupancy group that were left, the occupancy was manually mapped to a NACE code. In the end, the number of different output classes, that should be classified, could be reduced from more than 375 to maximal 21.

After performing all the steps explained above, the prepared and processed data set consists of 1620 buildings in total. After dropping the location variables, except the state, as this information was already translated to the land use information, and the unique identifier, as this was transcribed to the insurance branche, 52 explanatory variables were left.

The number of observations for each class label is clearly not balanced. While there are just a few observations in the data set with class label A or S, there are more than 500 observations with class label C. This unequal distribution of classes is depicted in Figure 5.2. A classification problem with unequal class distribution is called imbalanced classification.

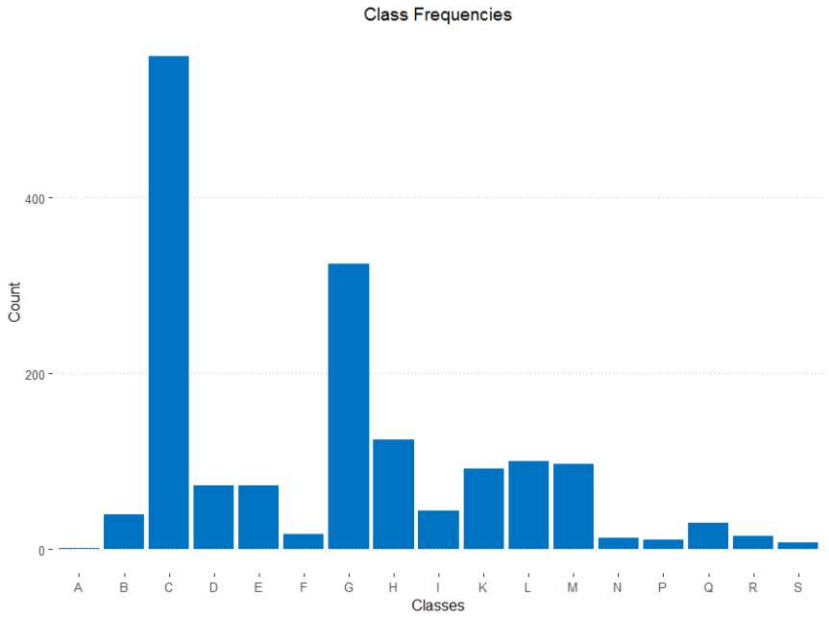


Figure 5.2: Class frequencies

The data set has also missing values in the building physical properties, in the industry category and type of business variables and in the state and land use features. Specifically, the proportion of missing values in the building physical properties is higher than 90%

and therefore they were removed before further study. The industry category and the type of business is missing for 11.4% and 11.8% of the buildings, respectively. While the land use could not be extracted for 6.6% of the buildings, the proportion of the missing values for the state is 5.6%. Interestingly, a multivariate missingness pattern can be observed. In all cases where industry category is missing, the type of business is also missing. So, except for 6 instances either the values for both variables are given, or the values are missing for both of them, which is the case for 170 buildings. A similar phenomenon is given for the variables state and land use. For 75 buildings both values for these variables are missing, while only land use is missing for 17 buildings. The missingness pattern does not automatically indicate a special missingness mechanism. While no relation between the missingness of variables and other observed or unobserved information is known, graphical comparisons of the missingness pattern across different subgroups of the data do not clearly indicate any dependencies of the missingness of the specified variables and other variables in the data, while taking the missingness pattern into account. An illustration of the proportions of missing values and of the missingness pattern can be seen in Figure 5.3.

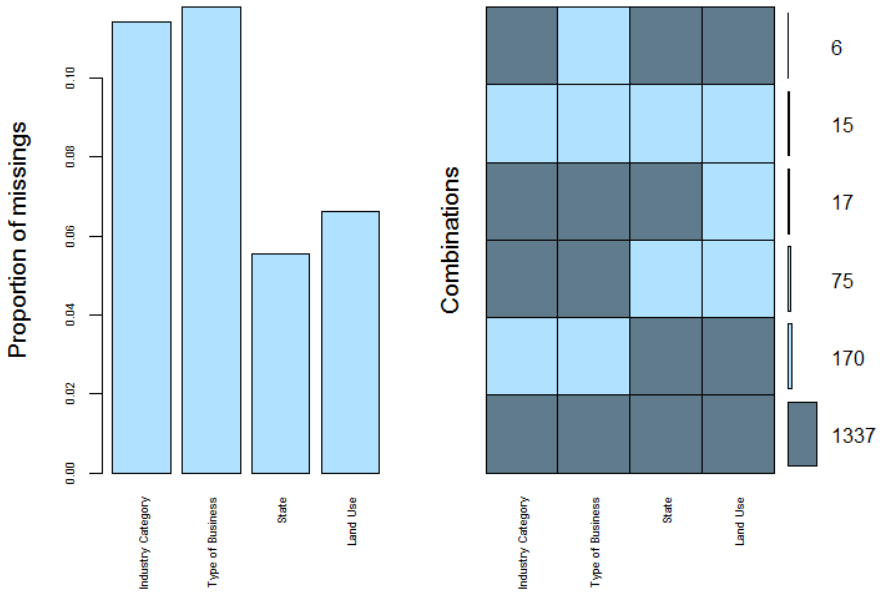


Figure 5.3: Proportions and combinations of missing values

For almost all categorical variables in the data set a one hot encoding was used, that converts a categorical variable to multiple binary variables, one for each possible attribute value of the variable, indicating whether or not that attribute value is assigned to an observation. For the type of business variable, a manual embedding had to be found. Several binary variables, 30 to be exact, were created by checking if specific keywords such as hotel, hospital, bank etc. are included in the description of type of business or not. The embeddings increased the number of variables to 136. As a missing value for

a variable for one observation creates also missing values in all associated columns of the embedding for this specific observation, an even more extreme missingness pattern is observed after the embedding. An illustration is seen in Figure 5.4. Every column illustrates one of the 136 variables, and the rows refer to the observations. Known values are grey, while the missing values are highlighted in black. Before the embedding there were 170 observations with missing values for the two variables industry category and type of business and 75 observations with missing values for the two variables state and land use, as can be seen in Figure 5.3. After the embedding, the 170 observations have 47 missing values for all the columns of the embeddings for industry category and type of business and the 75 observations have missing values for the 30 variables of the embeddings for state and land use.

Figure 5.4: Missingness pattern after the embedding. The x-axis is showing the 136 variables.

5.2 Study Design

The aim was to find a suitable classification method to classify the NACE code, as grouping of the occupancy, for the buildings in combination with an appropriate technique for handling the missing values using the data set provided. An extensive simulation study was conducted where different decision tree methods and random forest methods in combination with various missing data handling techniques were compared. All methods used in the simulation study are theoretically analyzed in Chapter 3 and 4.

The use and comparison of the different techniques are illustrated through an analysis performed with the `R` software for statistical computing[6], version 4.2.3.

In order to conduct a controlled simulation study, all observations containing missing values for any variables were deleted to end up with a complete data set. This complete data set consists of 1337 observations.

The data was randomly split into training and test data with a ratio of 80:20, meaning 80% of the observations were used for training the model, and the remaining 20% of the observations were used to evaluate the model's performance. For the simulation, different

[6]https://www.r-project.org/ (accessed 10.08.2023)

percentages of artificial missing values were created in the training data. Artificially creating missing values means that specific known values are deleted and considered as missing. More precisely, missing values were introduced in 5%, 10%, 20%, 30% and 50% of the observations, respectively. The missingness pattern in the originally prepared data set, as illustrated in Figure 5.3, showed that for approximately two-thirds of the observations with missing values, both the industry category and the type of business missing are missing, while for around one-third, both the state and land use are missing. In the simulation study, this missingness pattern of the originally prepared data set was taken into account in order to simulate a similar pattern. Therefore, in 5%, 10%, 20%, 30% and 50% of the observations of the training data, respectively, missing values were artificially created in all columns of the embeddings of industry category and type of business for two-thirds of these observations, and in all columns of the embeddings for state and land use for the remaining one-third of these observations. Using the training data with varying amounts of artificially created missing values and taking the missingness pattern into account, different decision trees and random forests, in combination with various missing data handling techniques, were trained, and evaluated on the test data. All these steps were repeated 100 times to obtain an overview of the variance and the stability of the methods and of the results.

The focus of the first part of the simulation study was on strategies of decision trees to handle missing values directly during the training phase, without the need for explicitly imputating them before fitting the model. The decision tree methods that were analyzed are: conditional inference trees, CART (classification and regression trees), C4.5 decision trees and C5.0 decision trees. For all these methods pre-implemented functions of `R` packages with their default parameter values were used. The manually implemented missing data handling techniques are the listwise deletion (LD) strategy and the separate class (SC) method. Listwise deletion refers to deleting all rows of the data set containing any missing values, which is easy to perform. By applying the separate class method, the missing values in the categorical variables of the data set were replaced by a separate category. This separate category can be denoted arbitrarily and it only has to be different from the two categories that already denote the two classes of the binary variables. Additionally, each of the aforementioned decision tree methods has a different built-in strategy to handle missing values directly during the fitting. Whereas conditional inference trees can incorporate missingness in attributes (MIA), CART decision trees apply surrogate variables (SV) or eliminate observations with missing values (El), while C4.5 and C5.0 decision trees can handle missing values using fractional cases (FC). All these missing data handling strategies are pre-implemented in the respective `R` functions of the decision trees. An overview about the different tested combinations of decision trees, with their implementation in `R` and the version used, and the corresponding missing data handling techniques is provided in Table 5.2.

Next, the focus was on strategies of random forests that handle the missing values directly without an explicit imputation step. The random forest methods tested are Breiman's random forest and conditional random forests. For both of these methods

Decision Tree Method	R Function	R Package (version)	Missing Data Handling Technique
Conditional Inference Tree	ctree()	partykit (1.2.19)	Listwise Deletion
CART	rpart()	rpart (4.1.19)	Listwise Deletion
CART	tree()	tree (1.0.43)	Listwise Deletion
C4.5	J48()	RWeka (0.4.46)	Listwise Deletion
C5.0	C50()	C50 (0.1.8)	Listwise Deletion
Conditional Inference Tree	ctree()	partykit (1.2.19)	MIA
CART	rpart()	rpart (4.1.19)	Surrogate Variables
CART	tree()	tree (1.0.43)	Elimination
C4.5	J48()	RWeka (0.4.46)	Fractional Cases
C5.0	C50()	C50 (0.1.8)	Fractional Cases
Conditional Inference Tree	ctree()	partykit (1.2.19)	Separate Class
CART	rpart()	rpart (4.1.19)	Separate Class
CART	tree()	tree (1.0.43)	Separate Class
C4.5	J48()	RWeka (0.4.46)	Separate Class
C5.0	C50()	C50 (0.1.8)	Separate Class

Table 5.2: Tested combinations of decision tree methods and missing data handling techniques

pre-implemented functions of R packages with their default parameter values were used. Similar to decision trees, the manually implemented missing data handling techniques applied are listwise deletion and the separate class method. Each of the aforementioned random forests has also a different built-in strategy to handle missing values directly during training without the need for imputation. Whereas conditional random forests can incorporate missingness in attributes for each of its trees, Breiman's random forests can handle missing values through adaptive tree imputation. These strategies of handling missing values in random forests are pre-implemented in the corresponding R functions of the random forest methods. All tested combinations of random forests and missing data handling techniques, along with their R implementations and the versions used, are listed in Table 5.3. In the table, random forest is abbreviated with RF.

In addition to these various strategies of decision trees and random forests to deal with the missing values in the data directly during the learning phase, two different imputation methods were tested, namely single imputation and multiple imputation. Imputation refers to the process of replacing the missing values with estimated reasonable values, resulting in a complete imputed data set that can be used for further analysis, such as training decision trees or random forests. The selected single imputation method is the k-nearest neighbor (kNN) imputation, and the chosen multiple imputation method is multivariate imputation by chained equations (MICE). Pre-implemented R packages were used for both imputation methods, while the default parameter values of the

Random Forest Method	R Function	R Package (version)	Missing Data Handling Technique
Random Forest	randomForest()	randomForest (4.7.1.1)	Listwise Deletion
Conditional RF	cforest()	partykit (1.2.19)	Listwise Deletion
Random Forest	randomForestSRC()	randomForestSRC (3.2.1)	Listwise Deletion
Conditional RF	cforest()	partykit (1.2.19)	MIA
Random Forest	randomForestSRC()	randomForestSRC (3.2.1)	Adaptive Tree Imputation
Random Forest	randomForesst()	randomForest (4.7.1.1)	Separate Class
Conditional RF	cforest()	partykit (1.2.19)	Separate Class
Random Forest	randomForestSRC()	randomForestSRC (3.2.1)	Separate Class

Table 5.3: Tested combinations of random forest methods and missing data handling techniques

corresponding R functions were not changed. For kNN imputation, the default value for the parameter k - representing the number of nearest neighbors used - is 5. For MICE, the default number of multiple imputations, denoted by m, is 5. A classical CART decision tree and Breiman's random forest, respectively, were fitted on the imputed data sets generated by kNN imputation and MICE imputation. Comparisons were made with the strategies of CART decision trees and Breiman's random forests that deal with missing values directly without an explicit imputation step. All tested combinations of decision trees and random forests, respectively, and strategies applied to handle missing values in the data, along with the used R implementations and versions, are summarized in Table 5.4.

Classification Model	R Function	R Package (version)	Missing Data Handling Technique (R Package version)
CART	rpart()	rpart (4.1.19)	Listwise Deletion
CART	rpart()	rpart (4.1.19)	Surrogate Variables
CART	rpart()	rpart (4.1.19)	Separate Class
CART	rpart()	rpart (4.1.19)	kNN (VIM 6.2.2)
CART	rpart()	rpart (4.1.19)	MICE (mice 3.15.0)
Random Forest	randomForest()	randomForest (4.7.1.1)	Listwise Deletion
Random Forest	randomForest()	randomForest (4.7.1.1)	Separate Class
Random Forest	randomForest()	randomForest (4.7.1.1)	kNN (VIM 6.2.2)
Ranodm Forest	randomForest()	randomForest (4.7.1.1)	MICE (mice 3.15.0)

Table 5.4: Tested imputation methods

After training the models using the training data, the model's performance was measured using the test data. The performance indicators used are the accuracy, the F1-Score, and the confusion matrix to gain a more accurate understanding. The effectiveness of the models was assessed and insights into the model's strengths and weaknesses were gained by analyzing the output of these performance indicators.

5.3 Results

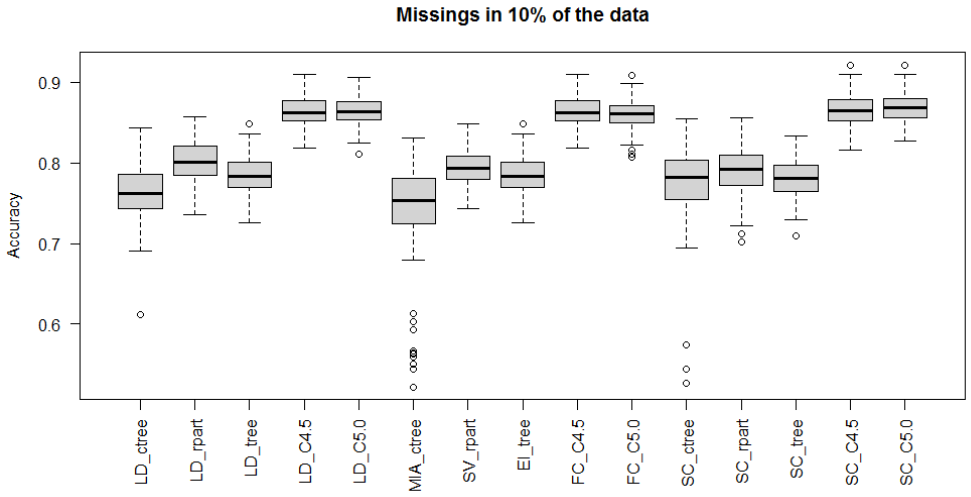
In this chapter, the results of the simulation study are presented and analyzed. In the first section, the performance of decision trees and random forests in predicting the NACE code for buildings included in the data set is assessed. Various strategies for handling missing values in the data directly during the learning phase are tested. Afterwards, the performance of a classical decision tree and random forest is evaluated by training them on imputed data sets resulting from kNN imputation and MICE imputation, respectively.

5.3.1 Decision Trees/Random Forests Missing Data Handling Techniques

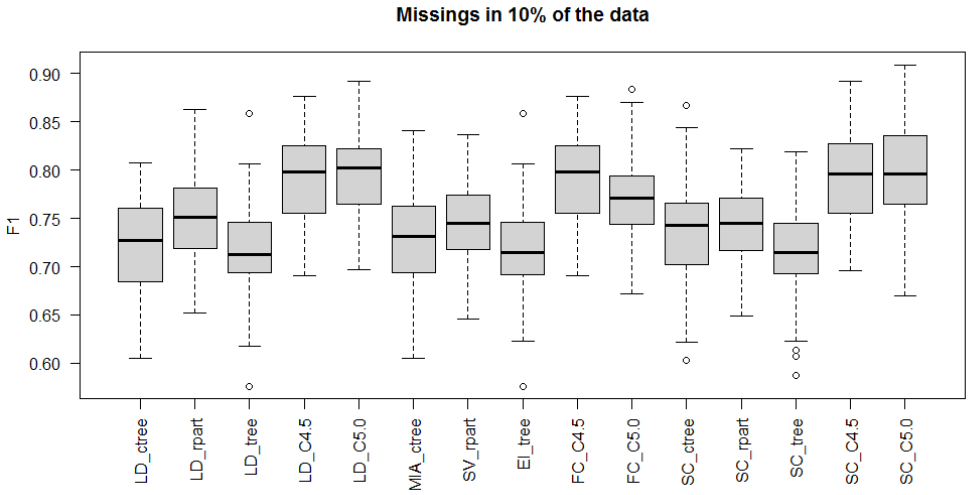
First, the performance of the aforementioned decision trees in predicting the NACE code, along with the specified strategies for handling the missing values in the data directly, is presented. As described in Chapter 5.2, the training data contains missing values in 5%, 10%, 20%, 30% and 50% of the observations, integrating the missingness pattern analyzed above.

In Figure 5.5, the results are presented for missing values in 10% of the training data. The boxplots represent the performance measures calculated while executing the 100 runs. Whereas the y-axis of Figure 5.5a depicts the accuracy, the y-axis of Figure 5.5b depicts the F1-score. Each boxplot outlines the distribution of the performance measure of predicting the NACE code using a specific decision tree method in combination with a missing data handling technique. The concrete combination of missing data handling technique and decision tree is stated below each boxplot, separated by an underscore. For example, *LD_ctree* refers to listwise deletion (LD) combined with conditional inference trees implemented in the `R` function `ctree`. In total, the results of all 15 tested configurations, which are listed in Table 5.2, are shown.

Figure 5.5a demonstrates that the median accuracy for all decision tree methods is between 75% and 87%. Conditional inference trees exhibit the poorest performance with a median accuracy of 75-78% across all three missing data handling techniques - listwise deletion, missingness incorporated in attributes (MIA), and the separate class method. While the median accuracy does not significantly change accross the missing data handling technique used, MIA shows slightly higher variance. C4.5 and C5.0 decision trees perform the best with a median accuracy of 86-87%. The figure proves that both these methods exhibit similar accuracy and that the choice of the missing data handling technique does not have a significant impact on the performance. Listwise deletion, fractional cases, and



(a) Accuracy



(b) F1-score

Figure 5.5: Performance of decision tree methods with missing values in 10% of the data

the separate class method yield identical results. CART decision trees produce moderate results with a median accuracy of 78-80%. While they outperform conditional inference trees, they perform worse than C4.5 and C5.0 decision trees. The two implementations of CART, namely `rpart` and `tree`, produce slightly different outcomes: `rpart` achieves a higher accuracy than `tree` of 1-2% accross all missing data handling techniques. While the performance of `tree` deos not significantly vary with the missing data handilng technique used, it can be observed that the built-in missing data handling technique, referred to as elimination (El), yields the exact same results as listwise deletion (LD) -

the combinations *El_tree* and *LD_tree* lead to the exact same boxplots. Similarly, the median accuracy remains relatively consistent across all missing data handling techniques - listwise deletion, surrogate variables, and the separate class method - for `rpart`.

In Figure 5.5b, the results are presented similar to Figure 5.5a, but the F1-score instead of the accuracy. The figure depicts the distribution of F1-scores for different decision tree methods in combination with various missing data handling techniques for predicting the NACE code. Compared to the accuracy measure, the median F1-score is lower, and there is a greater variability in F1-scores across all combinations of decision tree methods and missing data handling techniques. Nevertheless, the figure shows that the F1-score significantly changes with the decision tree method used but it stays constant across the different missing data handling techniques for a fixed decision tree method. C4.5 and C5.0 decision tree methods consistently perform the best, regardless of the chosen missing data handling technique, achieving a median F1-score of 78-80%. These methods show not only an overall strong performance, as observed in Figure 5.5a, but also exhibit relatively good performance across all classes, as seen in the F1-score. Achieving an average F1-score of up to 80% is possible.

The results in Figure 5.5 demonstrate that, when dealing with missing values in 10% of the training data, the classification accuracy and the F1-score for predicting the NACE code are primarily influenced by the choice of the decision tree classifier rather than the specific missing data handling technique applied.

In Figure 5.6, the results are presented for missing values in 50% of the training data. Similar to previous figures, the boxplots show the accuracy obtained while executing the 100 runs, and the combination of decision tree method and missing data handling technique is stated below each boxplot. The visualization closely resembles Figure 5.5a. Conditional inference trees exhibit the poorest performance with a median accuracy of 71-74%, while C4.5 and C5.0 decision trees consistently perform the best with a median accuracy of 81-84%, depending on the missing data handling technique. CART decision trees perform moderate with a median accuracy of 72-77%. The figure shows that the choice of the decision tree method strongly influences the performance, while the impact of the different missing data handling techniques used is minimal. Compared to the results with missing values in only 10% of the training data, the accuracy is lower when dealing with missing values in 50% of the data. While conditional inference trees achieve an average accuracy between 70% and 74%, depending on the missing data handling technique, C4.5 and C5.0 decision trees achieve an average accuracy of up to 84%. Overall, also the variance of the accuracy is higher, when dealing with more missing values, particularly for conditional inference trees.

For all other amounts of missing values in the data, that were tested, the results look similar. No matter if there are missing values in 5%, 10%, 20%, 30% or 50% of the training data, the results show that the performance of the classifier used highly depends on the decision tree method and hardly on the strategy that is used to handle the missing values. Additionally, the performance decreases as the number of missing values in the data increases. This is clearly observed in Figure 5.7.

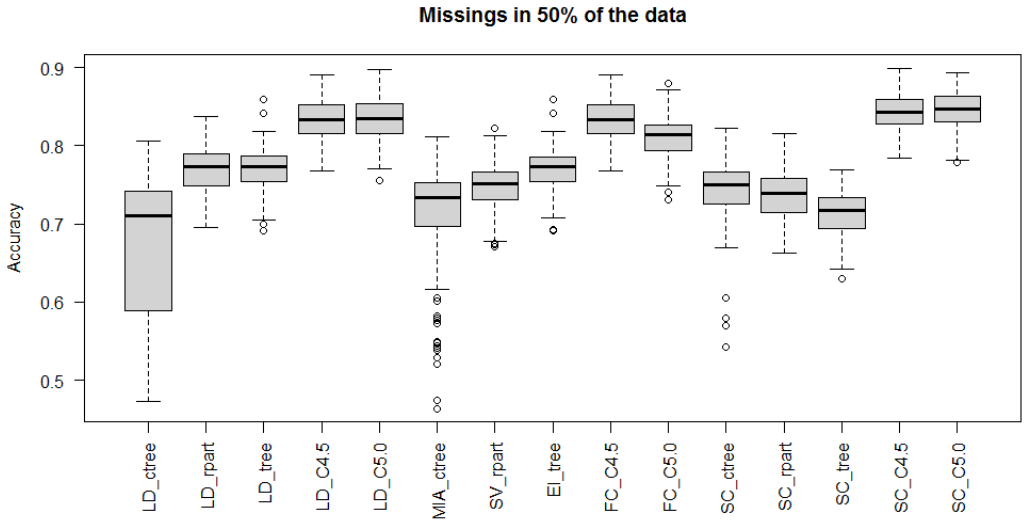


Figure 5.6: Accuracy of decision tree methods with missing values in 50% of the data

Next, the performance is evaluated of the above defined random forests in predicting the NACE code, along with the random forest missing data handling techniques specified. All tested combinations are listed in Table 5.3. The methods are evaluated for 5%, 10%, 20%, 30% and 50% of the data containing missing values, while the aforementioned missingness pattern is reflected in the data.

In Figure 5.8, the results are presented for missing values in 10% of the training data. The boxplots visualize the distribution of the performance measures evaluated during the 100 runs. Whereas in Figure 5.8a the accuracy is depicted, the F1-score is depicted in Figure 5.8b. The tested combination of random forest and missing data handling technique is indicated below each boxplot. The `R` function `randomForest` is abbreviated as "rF" and `randomForestSRC` as "rFSRC".

Figure 5.8a shows that random forests achieve a median accuracy of 79-91%, depending on the chosen random forest method and missing data handing technique. Compared to decision trees, random forests exhibit better performance in terms of accuracy. Among all tested random forests, Breiman's random forests perform the best, with a median and average accuracy of around 90%. The average accuracy is roughly equal for all missing data handling techniques and for both implementations of random forests, that are available in `R` through the functions `randomForest()` and `randomForestSRC`. Conditional random forests perform worst, with a median and average accuracy of approximately 77-79%, which is almost identical to the accuracy achieved by a single conditional inference tree. Overall, it can be observed that the performance is highly dependent on the random forest method used and remains almost unaffected by the selected missing data handling technique.

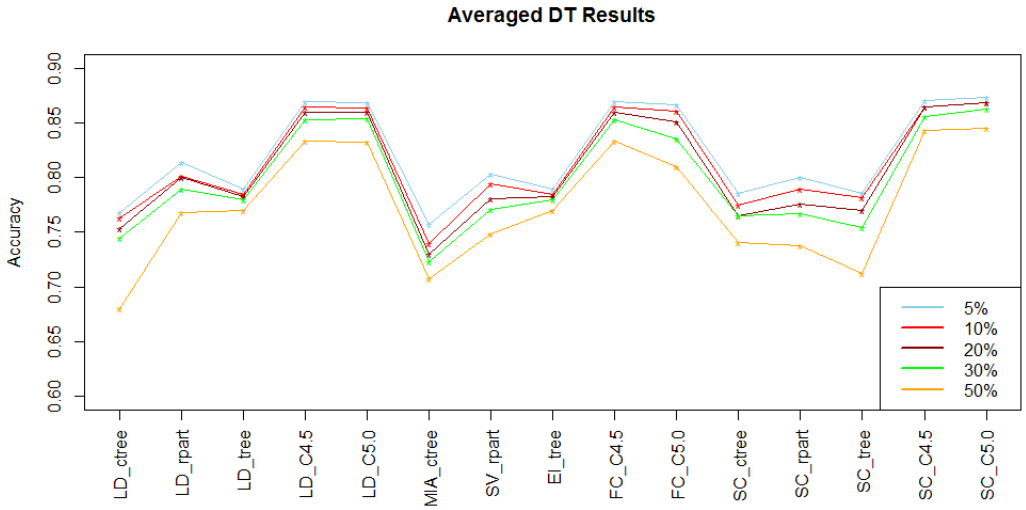
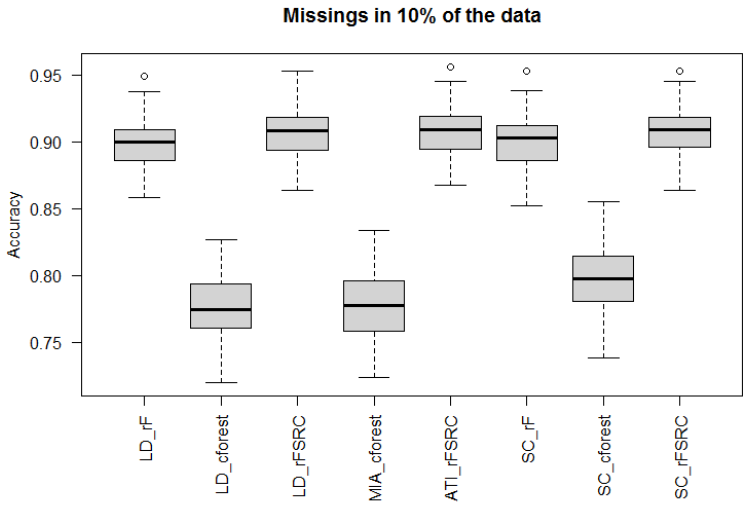


Figure 5.7: Averaged accuracy of decision tree methods for different proportions of missing data

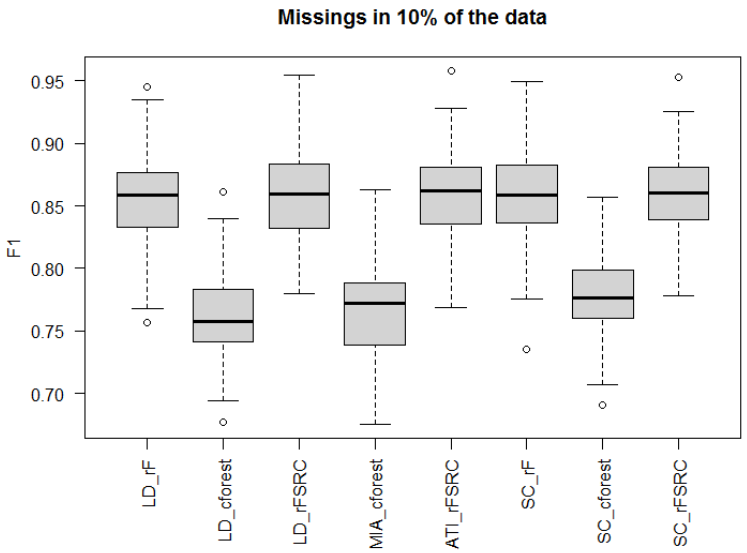
Figure 5.8b presents the distribution of F1-scores that is achieved by the random forest methods. In contrast to the accuracy, the median F1-score is lower for all specifications and the variability of the results is higher. All Breiman's random forests achieve a median and average F1-score of approximately 87%, whereas onditional random forests exhibit a median and average F1-score of 76-77%, across all missing data handling techniques. Similar as for decision trees, the figure shows that the performance on predicting the NACE code heavily relies on the random forest method applied and not on the missing data handling technique.

In Figure 5.9 the results for missing values in 50% of the data are visualized. The boxplots depict the distribution of the accuracy obtained while executing the 100 runs. Breiman's random forests achieve an accuracy of 86-88% and conditional random forests exhibit an accuracy of 70-75%. The behavior of the random forest methods, along with their missing data handling techniques, is similar to that observed when there are missing values in only 10% of the data. While the overall accuracy is of course lower, the figure shows that the choice of the random forest method has significant impact on the performance. Additionally, it is to observe that the performance hardly depends on the missing data handling technique.

For the other tested amounts of missing values in the training data, the results of the random forest methods, along with the tested missing data handling techniques, look similar. However, the performance tends to decrease as the number of missing values increases. The average accuracy of all specifications of random forest method and missing data handling technique in classifying the NACE code is displayed in Figure 5.10 for all considered amounts of missing values in the training data.



(a) Accuracy



(b) F1-score

Figure 5.8: Performance of random forest methods with missing values in 10% of the data

5.3.2 Imputation

In this section the performance of a classical decision tree and a random forest method fitted on imputed data to classify the NACE code of buildings is evaluated. The missing values in 5%, 10%, 20%, 30% and 50% of the training data, respectively, are imputed once by applying kNN imputation with $k = 5$, and once by applying MICE imputation

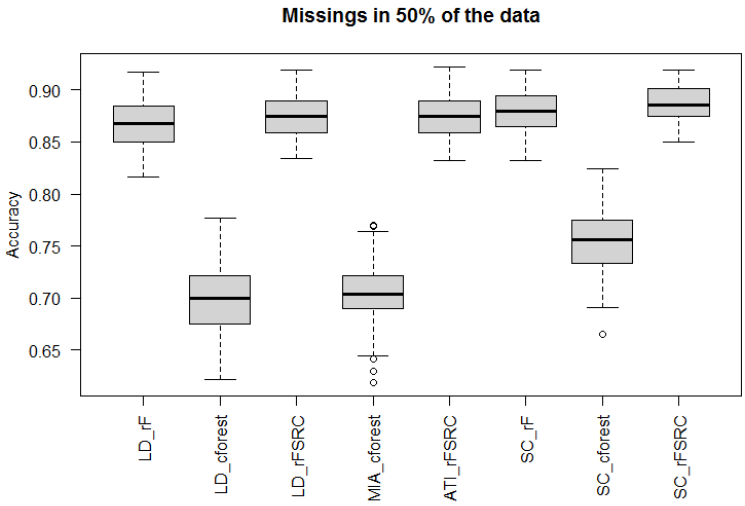


Figure 5.9: Accuracy of random forest methods with missing values in 50% of the data

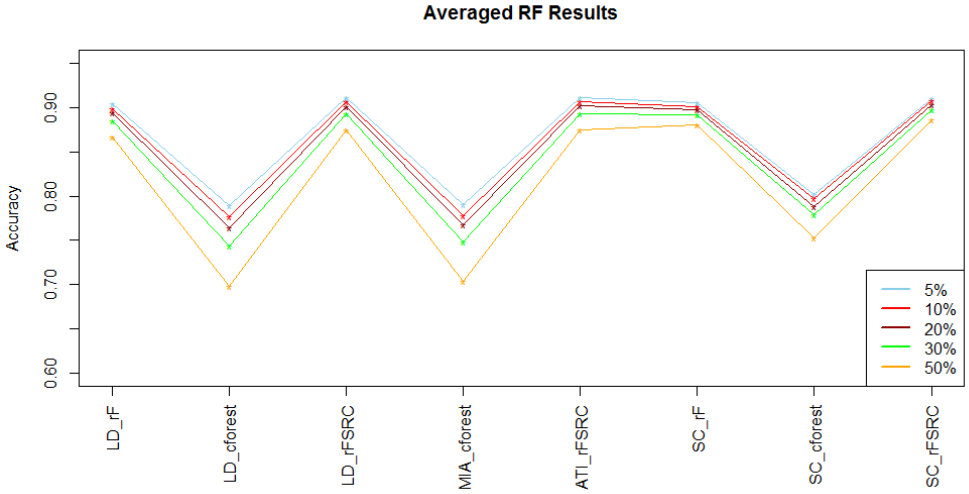


Figure 5.10: Averaged accuracy of random forest methods for different proportions of missing data

with $m = 5$. A CART decision tree and a Breiman's random forest are then built with the imputed training data sets. Their performance is compared to CART decision trees and Breiman's random forests along with strategies to handle the missing values directly without an explicit imputation step. All tested combinations are listed in Table 5.4.

In Figure 5.11, the accuracies of the tested CART decision trees in classifying the NACE code, obtained while executing the 100 runs, are presented as boxplots. Figure 5.11a

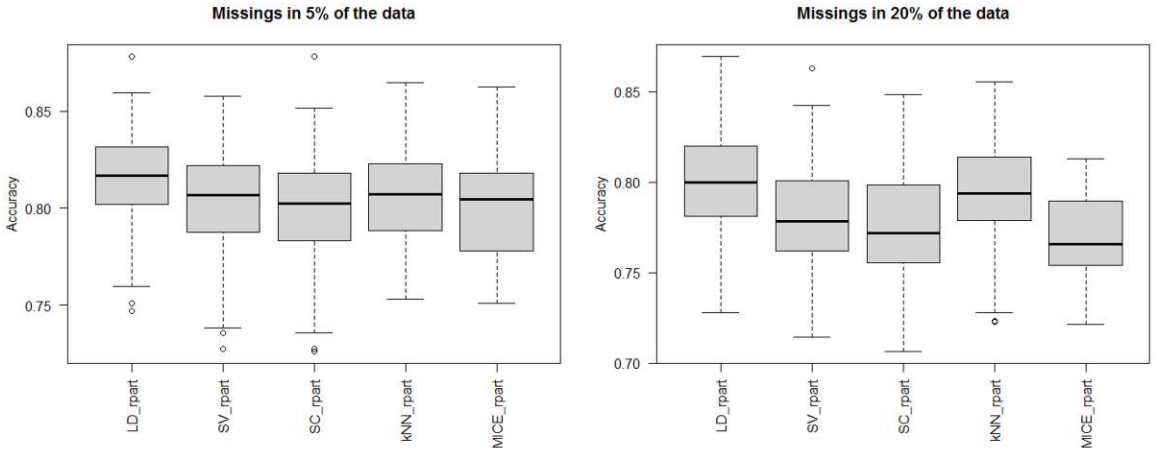
presents the results for having missing values in 5% of the data, Figure 5.11b for having missing values in 20% of the data and Figure 5.11c shows the results for having missing values in 50% of the data. To train the CART decision trees, the `R` function `rpart` was used. The concrete missing data handling technique that was used in combination with CART is stated below each boxplot. The first three techniques shown in each figure, which are listwise deletion, surrogate variables, and separate class method, are strategies of decision trees to handle the missing values directly during training the model, while kNN and MICE are the two imputation methods tested. For missing values in 5% of the data, the median accuracy is roughly the same for all techniques, as seen in Figure 5.11a. It ranges from 80-82%. Therefore, the strategies of decision trees to handle missing values directly during training the model are competitive to kNN imputation, where an explicit additional imputation step is needed before training the model. Even decision trees trained on imputed data resulting from more complex imputation methods like MICE do not show an improvement in the performance.

In Figure 5.11b and 5.11c, it can be seen that the accuracy decreases with increasing number of missing values in the data. Also with higher amount of missing values, it is observed that CART decision trees perform similarly independent of the missing data handling technique used. CART trained on kNN imputed data performs slightly better than CART trained on MICE imputed data, with a median accuracy of 79% compared to 77% for missings in 20% of the training data. Additionally, even for more missing values in the data, imputation does not outperform strategies of decision trees to deal with the missing values directly.

Figure 5.12 presents the accuracies of the tested Breiman's random forests in classifying the NACE code, obtained while executing the 100 runs, as boxplots. The individual subfigures show the results for having 5%, 20% and 50% missing values in the training data, respectively. The missing values in the training data were imputed by using a single and a multiple imputation method, namely by kNN and MICE imputation, and a classical random forest was trained on these imputed data sets. For the random forest classifiers the `R` function `randomForest` was used. Their accuracy was evaluated and compared to the accuracy of random forests that use strategies to handle the missing values directly. Therefore, for each of the three subfigures, the first two boxplots represent the performance of random forest missing data handling techniques, namely listwise deletion and separate class, and the last two boxplots represent the performance of random forests trained on imputed data.

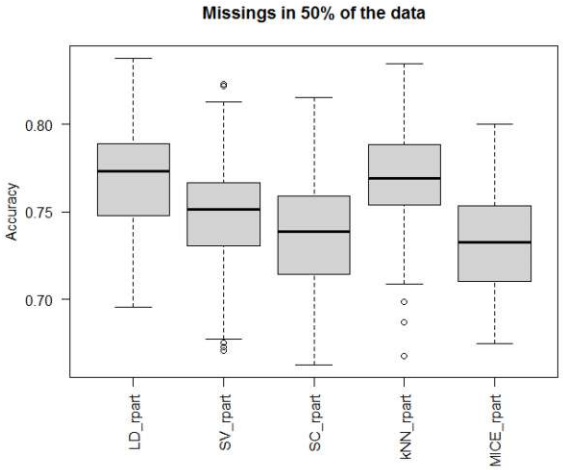
When having missing values in 5% of the data, the performance of all tested specifications of random forests and missing data handling techniques perform equally well, as seen in Figure 5.12a. The strategies of random forests to handle the missing values directly during training them, are competitive to single imputation methods like kNN and to even more complex multiple imputation methods like MICE. All techniques reach an average accuracy of more than 90%.

For more missing values in the data, the accuracy decreases for all combinations of random forests and missing data handling techniques. Figure 5.12b shows the results for



(a) Missing values in 5% of the data

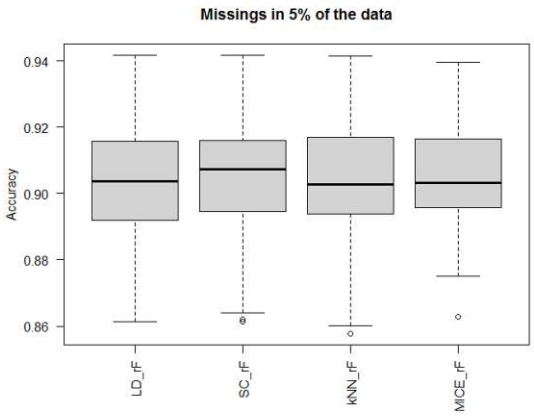
(b) Missing values in 20% of the data



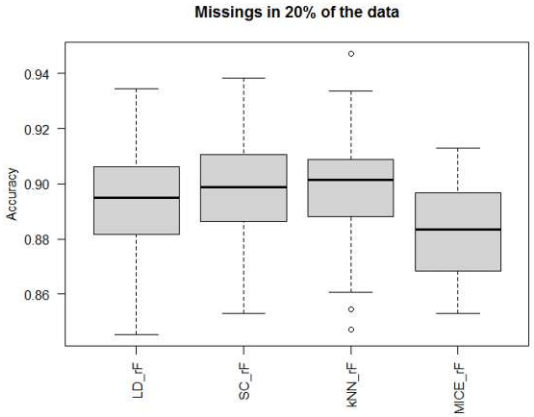
(c) Missing values in 50% of the data

Figure 5.11: Decision tree methods with imputed data

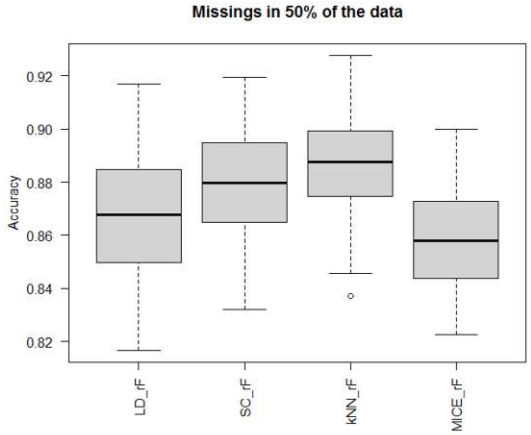
missing values in 20% of the data. While listwise deletion or the separate class method along with random forests are competitive to random forests trained on kNN imputed data, achieving a median accuracy of 90%, random forests trained on MICE imputed data performs slightly worse with 89%. The results for having missing values in 50% of the data are shown in Figure 5.12c. Small variations in the performance of the different techniques can be observed. Keeping in mind the spread of the results, these deviations are rather small and all of the techniques perform rather equally well. The results for having missing values in 10% or 30% of the data lead to the same observations.



(a) Missing values in 5% of the data



(b) Missing values in 20% of the data



(c) Missing values in 50% of the data

Figure 5.12: Random forests with imputed data

5.4 Discussion

The simulation study examined the performance of decision trees and random forests in classifying the NACE code for buildings in the data set provided by UNIQA, along with different strategies to handle the missing values in the data directly during training the model or with a single or multiple imputation step before training the model. The aim was to find appropriate missing data handling techniques along with a decision tree or random forest classifier for different proportions of missing data while considering the specific missingness pattern explained above.

Considering various factors such as the size and structure of the data set, the missingness pattern and the impact of misclassifications, the threshold for the accuracy for considering results as satisfactory within the simulation study was defined as 80% and the threshold

for the F1-score was defined as 75% for missing values in 10% of the training data, which matches the median performance of a classical CART decision tree combined with the simple approach of listwise deletion for handling the missing values in the data. It is a threshold that is feasible to achieve, but one that ensures reliable results at the same time. The thresholds were decreased for higher amounts of missing values in the training data. Therefore, for missing values in 50% of the data the results are considered as satisfactory if an accuracy of more than 75% and an F1-score of more than 70% is achieved.

Among the tested decision tree methods, sufficient good results in the context of this study were achieved. For missing values in 10% of the training data, C4.5 and C5.0 decision trees exhibit a median accuracy of 86-87% and median F1-score of 78-80%. These values clearly exceed the chosen threshold values. Conditional inference trees exhibit poor performance with a median accuracy of 75-78% and CART decision trees show sufficient good results with a median accuracy up to 80%.

One possible reason is that C4.5 and C5.0 decision trees use the information gain ratio to evaluate the quality of splitting attributes during training the model compared to CART decision trees which use the Gini index or information index. The information gain ratio is especially suitable when dealing with mixed, continuous, and categorical variables in the data. Furthermore, it considers the intrinsic information of a variable. This prevents bias towards variables with many distinct values. [Qui93] In order to prove and understand the influence of the information gain ratio comparison tests would need to further be performed. By using the exact same implementation of decision trees once with the information gain ratio and once with the Gini or information index as splitting criterion and by using cross-validation the performance could be directly compared based on several performance metrics. Afterwards statistical tests could be applied to test the significance of the difference in the performance. On the other hand, a possible reason for the poor performance of conditional inference trees is the presence of imbalanced classes in the data, as the permutation tests performed when building a conditional inference tree are affected. Tests with balanced datasets, achieved by applying resampling techniques, would need to be performed to prove this. Another reason may be nonlinear relationships in the data. An assumption of the statistical tests in conditional inference trees are linear relationships between the explanatory variables and the output variable. In case of non-linear relationships, the performance may suffer. [May] To give evidence for this, statistical tests would need to further be performed to test the linearity of the relationships between the explanatory variables and the output variable.

C4.5 and C5.0 decision trees show similar performance since they rely on the same key concepts as explained in Chapter 4.3. Similarly, the `R` functions `rpart` and `tree` are both implementations of CART decision trees and lead therefore to a similar performance. The main differences of these two implementations are the built-in strategies for handling missing values. As noted in Section 4.1.2, the built-in missing data handling technique of the function `rpart` handles missing values in a more ambitious way by using surrogate variables, as proposed by the theorie of CART decision trees, than the built-in missing data handling technique of the function `tree`. Therefore, `rpart` comprehensibly leads to

a slightly higher median accuracy of 1-2% than `tree`, when using the built-in techniques. The built-in missing data handling technique of the `R` function `tree`, referred to as elimination (El), simply eliminates or ignores observations with missing values. This is equivalent to listwise deletion (LD) and this is the reason why the combinations *LD_tree* and *El_tree* yield the exact same results in previous figures, such as Figure 5.5.

Among the tested random forests, Breiman's random forests exhibit high performance in predicting the NACE code for the buildings with a median accuracy of 90% while conditional random forests perform with a median accuracy of only 77-79%. The explanation for the poor performance of conditional random forests may be the same as for conditional inference trees since the forest consists of an ensemble of conditional inference trees.

For each decision tree and random forest method, no significant difference in the performance across the techniques for handling the missing values in the data could be observed. All strategies of decision trees and random forests to handle the missing values directly during training the model led to the same performance as simple listwise deletion, where the observations containing missing values are just dropped. The special missingness pattern reflected in the data is an explanation, as addressed in detail again below. Techniques of decision trees and random forests to handle missing values improve the performance compared to considering only the complete observations, if values are only occasionally missing. If for observations many values are missing and especially many of the most important ones as in the study data set and as evaluated in Chapter 6.3, the performance is not improved when considering them. In case of CART decision trees, for example, surrogate variables are used to handle the missing values in the data. When deciding on a split variable at one node of the tree, only the observations with known values of the specific considered attribute are used in the calculation of the splitting criterion. Since observations with missing values contain missing values for almost all variables worth to consider for the split variable due to the missingness pattern, they only have minor influence on deciding on the primary split variable. The same applies to the choice of the surrogate variables. Furthermore, as soon as the primary split variable and all surrogate variables are selected at one node, the observations with the missing values are assigned to a child node based only on surrogate variables ranked further down the list, since these observations have missing values for many important surrogate variables due to the missingness pattern. Therefore, these observations might end up in child nodes, where they don't fit.

Neither single imputation, using kNN, nor multiple imputation, using MICE, could improve the classification compared to strategies of decision trees and random forests to handle the missing values directly. One possible reason is the quality of the imputed data. Due to the special missingness pattern, the imputation is very difficult as further explained in the next paragraph. In order to prove this, further tests about the quality of the imputed data might be performed in the future, by comparing the distribution and descriptive statistics for the observed and imputed data or by performing graphical checks.

Overall, these insights shows that observations containing missing values don't improve the classifier in the study data set. Nevertheless, several results and findings from other studies in the literature show that appropriate handling of missing values can indeed improve the classification. However, it is very difficult to impute or to handle the missing values in the data set consisting of buildings in another meaningful and appropriate manner due to the missingness pattern in the data. For two-thirds of all the observations containing missing values both variables, industry category and type of business, and consequently all values of their embeddings are not known. This results in missing values for 47 columns of the prepared data set. For the remaining one-third of all the observations containing missing values both of the variables, state and land use, and as a consequence all values of their embeddings are not known, resulting in missing value for 30 columns in the prepared data set. Therefore, if an observation has a missing value, it has missing values for multiple variables and less information remains that can be used for imputation or handling the missing values in another way. Furthermore, the variables containing missing values, like industry category and type of business, belong to the most important variables for predicting the NACE code and it is difficult to impute them based on the other information given for an observation. The imbalanced classes in the data add another challenge for the classification.

All in all, Breiman's random forests exhibit high performance in classifying the NACE code, with an average accuracy of up to 90%, while missing values are present in the training data with the specific missingness pattern described above. They achieve an accuracy and F1-score that is clearly above the threshold and therefore considered to be highly acceptable within the context of this study. They are worthwile to be taken into account to predict the missing NACE code attributes in the data set and consequently improve the data quality. Both `R` implementations of random forests, `randomForest` and `randomForestSRC`, achieve almost identical results, since both of them are based on the same key concepts of Breiman's random forests.

While the performance decreases with increasing amount of missing values, the accuracy and F1-score are higher than 86% for missing values in 5%, 10%, 20%, 30% or 50% of the training data. The average accuracy ranges from 91% for missing values in 5% of the data to 87% for missing values in 50% of the data. The explanation is that the higher the amount of missing values in the data the higher the loss of information and consequently the lower the available number of observations in case of the listwise deletion method. Therefore, the more missing values the less information is available that the model can use for the classification. As discussed above, the handling of the missing values such as imputation is very difficult due to the missingness pattern and therefore observations containing missing values don't improve the classifier in this study data set. This means that imputed values cannot compensate the loss of information of missing values in the data set.

The performance of the random forest hardly depends on the technique that is used to handle the missing values in the data. All strategies of random forests to handle the missing values directly during training them lead to the same results. So, simple listwise

deletion and the separate class method achieve the same accuracy as more complex adaptive tree imputation. Imputation of the missing values before training the random forest does not exhibit improvement compared to strategies of random forests to handle missing values directly, neither kNN imputation nor more complex MICE imputation.

Therefore, for the UNIQA data set with the specific missingness pattern reflected in especially less than 20% of the data, it is advisable to proceed arbitrarily in the choice of the missing data handling technique, as long as a well performing classification model like random forest is used.

CHAPTER 6

Case Study

In the following case study a final classification model to predict the NACE code for buildings, that is, the economic area attributed to a building such as agriculture, manufacturing, education, etc., was trained and evaluated by using the data set provided by UNIQA.

6.1 Classification Model

The aim of the case study was to train and evaluate a final classification model that predicts the NACE code by using the original data set with the original missing values provided by UNIQA. For this purpose, an appropriate decision tree or random forest classifier along with an appropriate strategy to handle the missing values in the data had to be chosen.

For this case study, the full provided data set was used. As examined in Chapter 5.1, the full data consists of 1620 observations and 136 explanatory variables after applying one hot encoding embeddings or manual embeddings to all categorical variables. The data set contains a specific missingness pattern, shown in Figures 5.3 and 5.4. For most of the observations, the values are either known for all columns of the embeddings of type of business and industry category or missing for all of them. The same relation exists between the variables state and land use, meaning that the values of the columns of the embeddings of state and land use are either known or missing for all of them.

In total, 283 observations contain missing values, that it, around 17.5% of all observations of the data contain missing values. From these, around 27% contain missing values for all columns of the embeddings of both variables state and land use and around 60% contain missing values for all columns of the embeddings of both variables industry category and type of business.

According to the simulation study in Chapter 5, Breiman's random forest classification models achieve an accuracy of more than 86% on the provided data for missing values in any percentage of the observations of the data between 0 and 50. And the performance hardly depends on the missing data handling strategy that is used in combination with a random forest classifier. Simple listwise deletion or the separate class method exhibit similar performance as the built-in random forest adaptive tree imputation. Applying imputation to the missing values in the data in a separate step before fitting the random forest doesn't improve the performance compared to the mentioned strategies of random forests to handle the missing values directly during fitting the model.

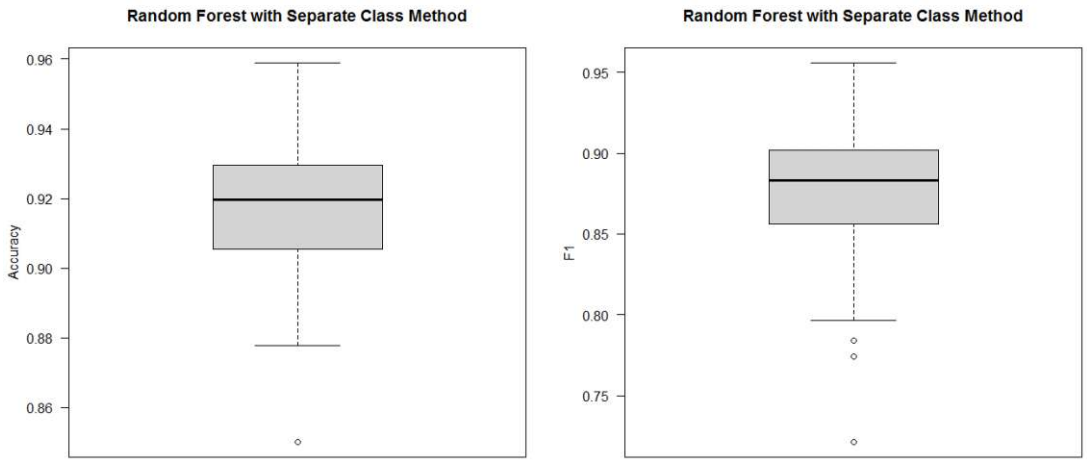
Therefore, an arbitrary strategy among the analyzed ones to handle the missing values in the data can be chosen in combination with a random forest classifier. The chosen option to classify the NACE code for buildings is a random forest classifier along with the separate class method to handle the missing values directly during fitting the model because the separate class method is an easy technique where no explicit imputation step is needed before.

6.2 Study

A random forest classifier along with the separate class method to handle the missing values in the data was used to predict the NACE code. The case study was conducted with the `R` software for statistical computing while a train-test split with a ratio of 80:20 was used. That is, random 80% of the observations of the full data set, including missing values, were used to train the model and the remaining 20% of the observations of the full data set were used to evaluate its performance. Making a random train-test split with a ratio of 80:20, fitting the random forest on the training data and evaluating its performance on the test data was repeated 100 times to obtain an indication about the stability of the results. For the random forest the `R` function `randomForestSRC` was used and the separate class method to handle the missing values in the data as a separate class in the random forest was implemented manually. The accuracy and the F1-score were used as performance criteria.

Figure 6.1 presents the performance of the random forest predicting the NACE code of buildings while using the separate class method to handle the missing values in the data directly during training the model. Figure 6.1a shows the accuracy and Figure 6.1b shows the F1-score evaluated while executing the 100 runs as boxplots. The model performance has an average accuracy of 0.918 over the 100 runs and an average F1-score of 0.875. Except for some outliers, all measures of the 100 runs are close to the average. Therefore, not only the overall performance is good but also the average performance on all different classes.

Figure 6.2 presents the results of the random forest per class, and therefore per NACE code. The F1-scores calculated in the 100 runs per class are shown as boxplots. Since there are no observations with NACE code A or S in any of the 100 test sets, there is no boxplot shown for these classes. While the median F1-score for many of the NACE



(a) Accuracy (b) F1-score

Figure 6.1: Performance of random forest with separate class method

codes is above 0.8, the median F1-score is above 0.6 for all the classes. There is no single class where the performance is extremely poor.

Comparing this visualization with Figure 5.2 of the number of observations per class, a dependence can be observed. The median F1-score is lower and the variance of the F1-scores is higher the smaller the class, that is, the less representatives of this class are included in the training data. This can be observed especially for the classes F, N and R. The number of observations in these classes is small and this is reflected in the F1-score. Compared to the other classes, the median F1-score for F, N and R is below 0.8, respectively. On the other side, the F1-score is high and stable especially for the classes C, G and H, as these classes are large and therefore have many representatives in the training data.

It was also analysed if the performance of a class is related to an extreme missingness pattern in the observations of that specific class by doing manual graphical checks. However, all classes exhibit a similar missingness pattern as the total data set and the classes with a worse performance don't show a different, more extreme missingness pattern.

A comprehensive view of the performance and the behavior of the model is delivered by the confusion matrix of one of the 100 random forests fitted along with the separate class method, shown in Figure 6.3. That specific random forest produced an accuracy of 95%, that is, 95% of the observations are in the diagonal of the confusion matrix. Only 5% of the observations were misclassified and are therefore off the diagonal of the confusion matrix. It is to observe that 3 out of 21 observations with NACE code L were classified

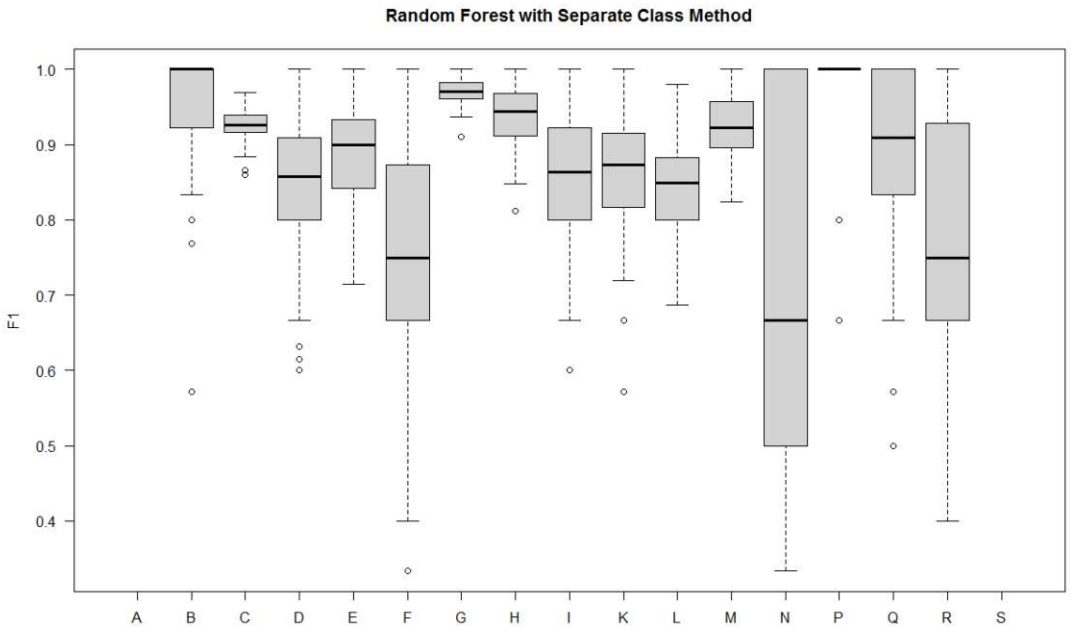


Figure 6.2: F1-score per class of random forest with separate class method

to have NACE code C. This indicates that the model has difficulties to decide between NACE code L and C for observations having actually NACE code L.

6.3 Variable Importance

The variable importance (VIMP) for a variable x of the data is defined by the difference between the prediction error calculated using the original ensemble and the prediction error calculated for the ensemble where all values of variable x were randomly permuted. It often indicates the same as the change in prediction error for a random forest trained with and without variable x. Large values for the importance of variables specify variables with high predictive ability, while low values specify variables with no predictive ability. [IKBL08]

The importance of the 25 most important variables evaluated by a random forest along with the separate class method are depicted in Figure 6.4. Every bar refers to the importance of one variable, while the variables are ordered from top to bottom according to their importance.

The visualization shows that especially the variables of the embeddings of type of business and industry category are very predictive. 15 out of the 25 most important variables are from the embeddings of these variables, while the most important variable is one column

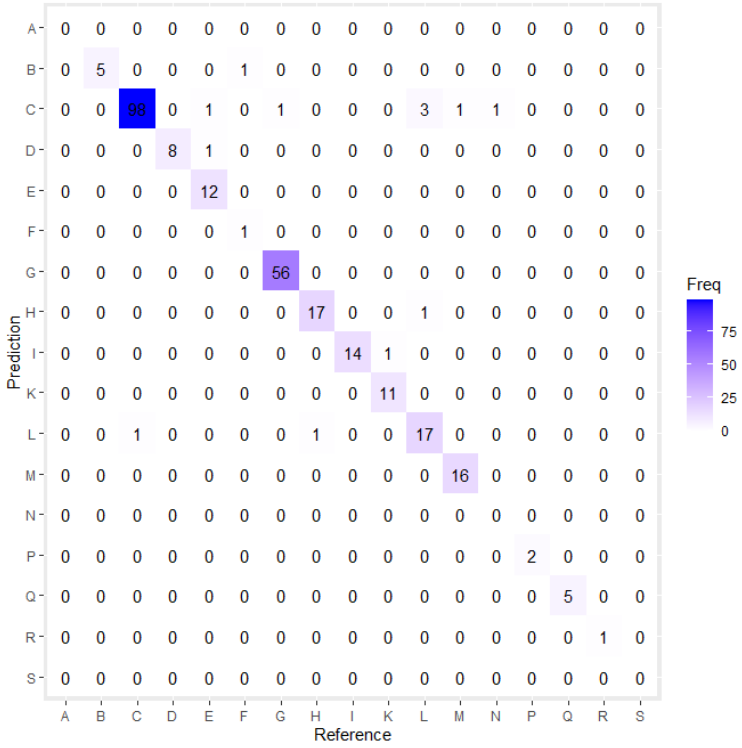


Figure 6.3: Confusion matrix of random forest with separate class method

of the embedding of the variable industry category. Further discussion is given in Section 6.4.

6.4 Discussion

The aim of the case study was to train and evaluate a final classification model that predicts the NACE code by using the original data set with the original missing values provided by UNIQA. For this purpose, a random forest classifier along with the separate class method to handle the missing values directly during fitting the model was chosen.

The model achieved an average accuracy of 0.918 over the 100 runs and an average F1-score of 0.875. In the context of this study, these are highly satisfactory results, as these values clearly exceed the threshold values defined within the simulation study. Compared to the data with artificially created missing values, the performance is 1-2% higher with the full data set and the originally included missing values, since more observations are available in the full data set to train the model. While considering the aim, the data size, the data structure, the missingness pattern and the impact of miscalssifications, these are highly acceptable results for the given project. The model does not only yield an overall good performance, as seen in the accuracy, but also performs well on all different classes, as seen in the F1-score. Based on these results the model is recommended to

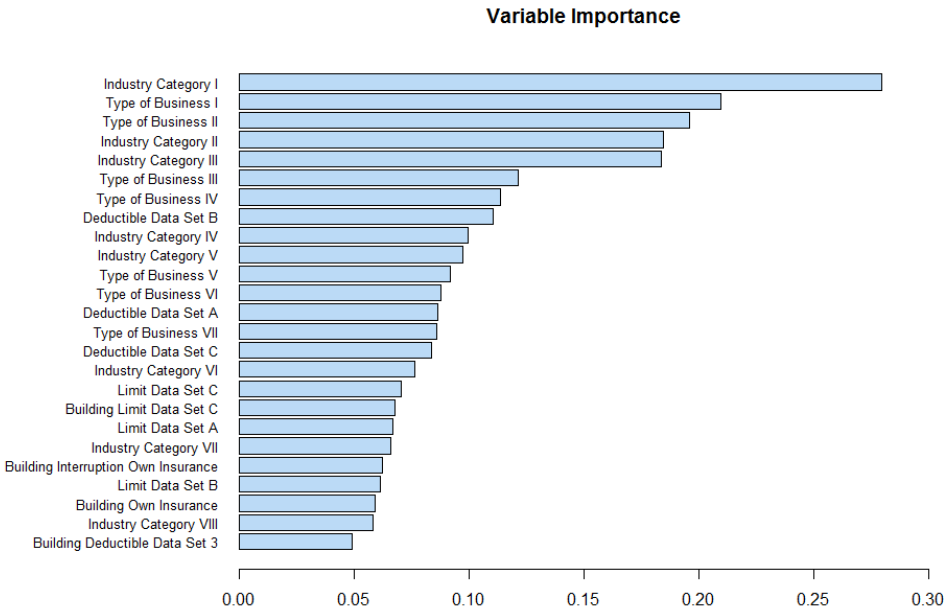


Figure 6.4: Variable importance of random forest with separate class method

be used in practice to predict the NACE code for the buildings to improve the quality of the data set. Reliable analysis can be done on the improved data set in the future, since the impact of the misclassifications, which concerns only a small number, is minor proven by the confusion matrix.

While an average F1-score of 0.875 is achieved, a dependence between the size of the classes and the classification performance can still be observed, when comparing Figure 5.2 and Figure 6.2. The median F1-score is lower and the variance of the F1-scores is higher the smaller the class, that is, the less representatives of this class are included in the training data. This is plausible since the larger the sample size and the larger the number of observations of a specific class available for training the model, the better the model performs for that class.

A detailed view of the performance is delivered by the confusion matrix of one trained random forest, with that the impact of misclassifications can be better understood. Figure 6.3 shows that 3 out of 21 observations with NACE code L were classified to have NACE code C and this indicates that the model struggels to distinguish between these classes. Since this only affects 14.3% of the observations with NACE code L, it highlights only minor weaknesses of the model's performance for class L. To generalize this statement, further test by considering the performance of multiple models, e.g. by analysing the confusion matrix of all 100 runs, would need to be performed.

As seen in Figure 6.4, the variables that are most important and have the greatest

predictive ability, are these of all variables that contain most missing values. Additionally, if an observation has a missing value for one of the columns of the embeddings of industry category and type of business, then it has missing values for all columns of the embeddings without exception, as the missingness pattern shows. Therefore, many of the predictive variables are not available for assigning such observations with missing values for the variables industry category and type of business to nodes of the trees and only less important variables can be used.

This is also related to the outcome of the simulation study that simple techniques for handling the missing values in the data, like listwise deletion or the separate class method, perform as good as more complex approaches or as single and multiple imputation methods. When the most important variables, like all variables of the embeddings of industry category and type of business, are missing, there is only little or not important information left to replace the missing values with reasonable values or to improve the classification model with the help of these observations while handling the missing values directly during training the model.

CHAPTER 7

Conclusion

The thesis examines a single imputation method, namely k-nearest neighbor (kNN) imputation, and a multiple imputation method, namely multivariate imputation by chained equations (MICE). It analyzes 15 combinations of decision trees and missing data handling techniques and 8 combinations of random forests and missing data handling techniques. These missing data handling techniques are strategies that are applied by decision trees or random forests to handle missing data directly at training time without the need of an explicit imputation step.

For the data set provided by UNIQA Insurance Group, outcomes of the simulation study reveal similar performance across all tested missing data handling techniques, while using a specific decision tree or random forest classifier, due to the special missingness pattern reflected in the data. However, the choice of the decision tree or random forest method has significant impact on the performance. Among all tested decision tree methods, C4.5 and C5.0 decision trees exhibit an average accuracy of 81-87%, whereas conditional inference trees yield an average accuracy of 71-78%, depending on the number of artifically created missing values in the training data. CART decision trees lead to an accuracy of 72-80%, meaning they achieve a higher accuracy than conditional inference trees but a lower accuracy than C4.5 and C5.0 decision trees. Among all tested random forest methods, Breiman's random forests exhibit excellent performance with an average accuracy of 86-90% compared to conditional random forests, which achieve an average accuracy of 70-79%. Breiman's random forests also outperform all tested decision tree methods. The study shows that the higher the amount of missing values in the data, the lower the prediction accuracy. However, even with missing values in 50% of the training data, all methods exhibit an accuracy of more than 70%.

A Breiman random forest in combination with the separate class method to handle the missing values in the training data is chosen as final model and fitted with the full provided data set to classify the NACE code for buildings within a case study. An average accuracy of 91% and average F1-score of 86% are achieved, when fitting and

testing the model 100 times with different train-test splits. The performance differs for the individual classes. Depending on the number of observations of a specific class in the training data, the performance is higher the higher the number of observations. Furthermore, the study outlines the importance of all variables. Especially the variables, that contain the information of the type of the business and the industry category of the buildings, are the most predictive variables with the highest importance.

The findings of the thesis have successfully addressed all research questions formulated in Chapter 1.2 of the thesis: By using the data set provided containing a special missingness pattern, imputation of missing values and strategies for handling missing values directly in decision trees and random forests does not significantly improve the classification accuracy compared to classification taking only the complete instances into account. Simple strategies of decision trees and random forests that directly handle missing values are competitive to single imputation methods and even to complex multiple imputation approaches. The NACE code of buildings, as grouping of the occupancy, can be accurately predicted with an accuracy of 91% by using a Breiman's random forest along with the separate class method for handling the missing values in the training data.

The thesis provides insights into decision tree and random forest classification and into missing values, as well as their intersection. Previously unexplored comparisons between various combinations of decision trees and random forests, respectively, and missing data handling techniques are outlined. A focus is put on decision tree and random forest strategies to handle missing values directly at training time with no explicit imputation step before fitting the model. The importance of analyzing the missingness pattern of the data is highlighted, as this significantly influences the performance of missing data handling techniques. In this sense, new perspectives for future research and applications are provided.

Beyond the theoretical implications, the outcomes of the thesis provide an example of a practical application in the industry with a real-world data set, while dealing with one of the most relevant inherent challenges, which data incompleteness. Real-world data sets are more often than not incomplete, not large enough and immense effort needs to be put in data cleaning and preparing. Only after proper data pre-processing, reliable and robust results can be achieved. This is seen within this study where a major part of the work was put on data preparation and exploration. The thesis also shows the importance of understanding and of adequate handling the limitations of real-world data sets, such as missing values and their patterns. The thesis additionally demonstrates that the existing machine learning models can be successfully applied to make predictions to improve the quality of real-world data and consequently to improve the accuracy of the results of the analyses performed with these data. Within this study, a random forest classifier combined with the separate class method to handle the missing values in the data was successfully trained to classify the NACE code of buildings. This classifier is used to improve the data quality of the industry data set by predicting missing NACE codes.

In future work, a greater focus should be put on the understanding of the performance of the tested methods and the correlation between the algorithms and the data. The

used data set within this study contains a specific missingness pattern, which is the basis for the explanation of the outcomes, reflecting special features and limitations as every real-world data set. It is therefore adviced to conduct controlled simulation studies with multiple different real-world data sets. This would lead to more generalized results, which would contribute to the understanding of the algorithms and their relation to the data. Furthermore, the focus should be extended to missing values in the test data as well. While some of the tested approaches, and especially the one that is used for the final fitted model - Breiman random forest along with the separate class method -, can also handle missing values in the test data, it is appropriate to conduct a further study in this direction. Moreover, in the context of the data provided, missing values could theoretically appear for new observations.

List of Figures

List of Tables

Bibliography

[Agg23] Sangeet Aggarwal. Data science in pharmaceutical industry [use cases + examples], 2023. https://www.knowledgehut.com/blog/data-science/data-science-in-pharmaceutical-industry (accessed 15.07.2023).

[AKAM12] Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5):272, 2012.

[BC11] Leo Breiman and Adele Cutler. Manual–setting up, using, and understanding random forests v4.0, 2011. https://www.stat.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf (accessed 25.04.2023).

[BFSO84] Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[Bre17] Leo Breiman. *Classification and Regression Trees*. Routledge, 2017.

[Cao17] Longbing Cao. Data science: A comprehensive overview. *ACM Computing Surveys (CSUR)*, 50(3):1–42, 2017.

[CCS12] Adele Cutler, D. Richard Cutler, and John R. Stevens. Random forests. *Machine Learning*, 45(1):157–175, 2012.

[Dec23] Decision tree advantages and disadvantages, 2023. https://www.educba.com/decision-tree-advantages-and-disadvantages/ (accessed 15.08.2023).

[D'O21] Marcello D'Orazio. Distances with mixed-type variables, some modified gower's coefficients. *arXiv preprint arXiv:2101.02481*, 2021.

[DS10] Yufeng Ding and Jeffrey S. Simonoff. An investigation of missing data methods for classification trees applied to binary response data. *Journal of Machine Learning Research*, 11(1), 2010.

[Fee99] Ad Feelders. Handling Missing Data in Trees: Surrogate Splits or Statistical Imputation? In J.M. Zytkow and J. Rauch, editors, *Principles of Data Mining and Knowledge Discovery*, pages 329–334. Springer, 1999.

[Fid84] Raya Fidel. The case study method: A case study. *Library and Information Science Research*, 6(3):273–288, 1984.

[GBV20] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.

[GS15] Sachin Gavankar and Sudhirkumar Sawarkar. Decision Tree: Review of Techniques for Missing Values at Training, Testing and Compatibility. In *2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS 2015)*, pages 122–126. Institute of Electrical and Electronics Engineers (IEEE), 2015.

[GS16] Sachin Gavankar and Sudhir Sawarkar. Decision tree: Compatibility of techniques for handling missing values at training and testing. *International Journal of Simulation: Systems, Science and Technology*, 17(34), 2016.

[HBH+23] Kurt Hornik, Christian Buchta, Torsten Hothorn, Alexandros Karatzoglou, David Meyer, Achim Zeileis, and Maintainer Kurt Hornik. Package 'RWeka', 2023. `https://cran.r-project.org/web/packages/RWeka/RWeka.pdf` (accessed 27.04.2023).

[HHZ06] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

[HHZ15] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Ctree: Conditional inference trees. *The Comprehensive R Archive Network*, 8, 2015.

[HZH23] Torsten Hothorn, Achim Zeileis, and Maintainer Torsten Hothorn. Package 'partykit', 2023. `https://cran.r-project.org/web/packages/partykit/partykit.pdf` (accessed 04.05.2023).

[IKBL08] Hemant Ishwaran, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3):841 – 860, 2008.

[IKK23] Hemant Ishwaran, Udaya B. Kogalur, and Maintainer Udaya B. Kogalur. Package 'randomForestSRC', 2023. `https://cran.r-project.org/web/packages/randomForestSRC/randomForestSRC.pdf` (accessed 01.06.2023).

[Kai14] Jiří Kaiser. Dealing with missing values in data. *Journal of Systems Integration*, 5(1):42–51, 2014.

[Kau23] Chirag Kaura. Data-driven decision-making in government: Improving policy formulation and public services, 2023. https://www.linkedin.com/pulse/data-driven-decision-making-government-improving-/policy-chirag-kaura (accessed 04.06.2023).

[KT16] Alexander Kowarik and Matthias Templ. Imputation with the r package vim. *Journal of Statistical Software*, 74:1–16, 2016.

[Kul97] Solomon Kullback. *Information Theory and Statistics*. Courier Corporation, 1997.

[KWC+23] Max Kuhn, Steve Weston, Mark Culp, Nathan Coulter, Ross Quinlan, et al. Package 'C50', 2023. https://cran.r-project.org/web/packages/C50/C50.pdf (accessed 16.04.2023).

[LR19] Roderick J.A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 2019.

[May] K. Mayank. Bxd primer series: Conditional inference decision trees and explanation of permutation test with p-value. https://www.linkedin.com/pulse/bxd-primer-series-conditional-inference-decision-trees-mayank-k- (accessed 20.05.2023).

[MWC19] Tim P. Morris, Ian R. White, and Michael J. Crowther. Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, 38(11):2074–2102, 2019.

[Nac] Glossary:Statistical classification of economic activities in the European Community (NACE). https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:Statistical_classification_of_economic_activities_in_the_European_Community_(NACE)#:~:text=The%20Statistical%20classification%20of%20economic,%C3%A9conomiques%20dans%20la%20Communaut%C3%A9%20europ%C3%A9enne. (accessed 01.03.2023).

[PT13] Preeti Patidar and Anshu Tiwari. Handling missing value in decision tree algorithm. *International Journal of Computer Applications*, 70(13), 2013.

[Pyl99] Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, 1999.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., 1993.

[Ran23] What are the advantages and disadvantages of random forest?, 2023. https://www.rebellionresearch.com/what-are-the-advantages-and-disadvantages-of-random-forest (accessed 17.08.2023).

[RR16] Brian Ripley and Maintainer Brian Ripley. Package 'tree', 2016. https://cran.r-project.org/web/packages/tree/tree.pdf (accessed 05.04.2023).

[Rub96] Donald B. Rubin. Multiple imputation after 18+ years. *Journal of the American Statistical Association*, 91(434):473–489, 1996.

[SG02] Joseph L. Schafer and John W. Graham. Missing data: our view of the state of the art. *Psychological Methods*, 7(2):147, 2002.

[Sny19] Hannah Snyder. Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, 104:333–339, 2019.

[SW99] Helmut Strasser and Christian Weber. On the asymptotic theory of permutation statistics. *Mathematical Methods of Statistics*, 8:220—250, 1999.

[TA+97] Terry M. Therneau, Elizabeth J. Atkinson, et al. An introduction to recursive partitioning using the rpart routines. Technical report, Technical Report Mayo Foundation, 1997.

[TARR22] Terry Therneau, Beth Atkinson, Brian Ripley, and Maintainer Brian Ripley. Package 'rpart', 2022. https://cran.r-project.org/web/packages/rpart/rpart.pdf (accessed 20.04.2023).

[TJH08] Bheki E.T.H. Twala, M.C. Jones, and David J. Hand. Good methods for coping with missing data in decision trees. *Pattern Recognition Letters*, 29(7):950–956, 2008.

[Twa09] Bhekisipho Twala. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, 23(5):373–405, 2009.

[VB18] Stef Van Buuren. *Flexible Imputation of Missing Data*. CRC press, 2018.

[VBGO11] Stef Van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45:1–67, 2011.

[ZZ08] Yongheng Zhao and Yanxia Zhang. Comparison of decision tree methods for finding active objects. *Advances in Space Research*, 41(12):1955–1959, 2008.