WIRTSCHAFTSUNIVERSITÄT WIEN
Vienna University of Economics and Business

Master Thesis

Title of Master Thesis:	Estimation of the (re-)integration likelihood into the Austrian labour market: a random forest approach
Author (last name, first name):	Bliem, Christian
Student ID number:	00505247
Degree program:	Master in Information Systems
Examiner (degree, first name, last name):	Dr.rer.soc.oec. Martin Waitz

I hereby declare that:

1. I have written this Master thesis myself, independently and without the aid of unfair or unauthorized resources. Whenever content has been taken directly or indirectly from other sources, this has been indicated and the source referenced.

2. This Master Thesis has not been previously presented as an examination paper in this or any other form in Austria or abroad.

3. This Master Thesis is identical with the thesis assessed by the examiner.

4. (only applicable if the thesis was written by more than one author): this Master thesis was written together with

   The individual contributions of each writer as well as the co-written passages have been indicated.

03/08/2020
Date

Signature

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EFMD EQUIS ACCREDITED

Master's Thesis

Estimation of the (re-)integration likelihood into the Austrian labour market: a random forest approach

Christian Bliem

Date of Birth: 16.11.1982
Student ID: 00505247

Subject Area: Information Systems

Study Code: J 066 925

Supervisor: Dr.rer.soc.oec. Martin Waitz

Date of Submission: 03 August 2020

Department of Information Systems and Operations, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

Acknowledgements

I would like to thank my supervisor, Martin Waitz, for his excellent guidance, for sharing his knowledge with me, and for his continuous support throughout the many months of working on this thesis. Furthermore, I would like to thank the Austrian Public Employment Service (AMS) for giving me the opportunity to conduct this study and, in particular, for providing the infrastructure and data for this work. My special thanks goes to my colleagues Karin König (AMS federal head office) and Reinhard Pipal (AMS regional office of Lower Austria) for their professional and personal support, without which I would not have been able to realise this thesis within the intended scope. Finally, I would like to thank my wife, Gisela, for many hours of listening, the possibility to exchange thoughts, and her incredible patience with me throughout the whole process of working on this thesis. I am deeply grateful to all of you who encouraged, supported and enabled me to write this thesis.

Contents

List of Figures

List of Tables

Abstract

Understanding the different aspects of the *labour market* is key for the well-founded service provision of public employment services. In Austria, *labour market monitoring* is performed by the *Austrian Public Employment Service (AMS)*, which systematically collects and analyses *labour market* aspects. The purpose of this thesis was to support the AMS' efforts in *labour market monitoring* by providing a *supervised machine learning model* to estimate the (re-)integration likelihood of unemployed people into the Austrian labour market.

The *machine learning approach* applied was *random forest*, a widely used *decision tree*-based ensemble method introduced by Leo Breiman in 2001. At the center of the study was the concept of *sustainable employment*, a concept describing an employment relationship which lasts for at least 63 days. The data for model building was drawn from the AMS database and included administrative data (i.e. data obtained in the context of client support) and insurance data. From this data, two different data sets were created, viz. the *main set* with 94 variables (comprising additional historical client information) and the *production set* with 71 variables. The two *performance measures* used for assessing the performance quality of the models were the *area under the ROC curve (AUC)* and the *geometric mean (GM)*.

The outcome of this study were four final models of considerable prediction quality. The best-performing model, a *probability forest* created from the *main set*, exhibited a prediction quality of 92.08% (GM). As this model provides class probabilities on an individual level, it is best suited for answering this thesis' research interest of estimating the *likelihood of labour market (re-)integration*. The second *probability forest*, created from the *production set*, is more applicable to situations in which there is a lack of historical data, a current example for such a situation being the fundamental shifts in the *labour market* caused by the Corona pandemic. The remaining *classification forest* models, which produce one-dimensional class assignments, are suitable for multi-class problems, such as for estimating the most suitable AMS funding measure for an individual client.

This thesis was intended to enhance the AMS' *labour market monitoring* toolkit by providing a (modifiable) prediction model. Furthermore, as *random forest* has so far rarely been used in the context of *labour market* research, this thesis has striven to contribute to the scientific community by showcasing the method's application in the environment of a national labour market. Finally, the combination of gaining a high level of prediction quality despite facing serious hardware limitations can hopefully contribute to the practitioners' community in terms of demonstrating *random forest*'s capabilities in a computationally low-end environment.

1 Introduction

The *labour market* is an important driver for economic and social development and welfare. Because of its importance, it is desirable to understand (and ideally to anticipate) the drivers and forces in this market of job seekers (*labour supply*) and job vacancies (*labour demand*). In order to comprehend the *labour market*, and in particular its most influential factors, observation is necessary. The discipline which deals with observing occurrences in the *labour market* is called *labour market monitoring*. *Labour market monitoring* is the systematic collection and analysis of labour market data and the development and compilation of reports in order to support (internal) decision-making processes, but also to provide information to the public. In Austria, it is the *Austrian Public Employment Service (AMS)* which is legally obliged to conduct *labour market monitoring* for the Austrian *labour market*.

The AMS has a long tradition in *labour market monitoring*. In 1946, the former Labour Market Administration (*"Arbeitsmarktverwaltung"*) started with the publication of regularly created statistics tables [46]. Over the years, and with the step-wise integration of IT-systems from the 1970s onwards, *labour market monitoring* evolved from a purely descriptive task (i.e. segmenting the *labour market* along certain characteristics, such as sex or age groups) to a more exploratory endeavour. In 1994, the *Labour Market Administration* was separated from the *Ministry of Social Affairs* and its succeeding organisation, the *Austrian Public Employment Service ("Arbeitsmarktservice Österreich, AMS"*), came into existence. The year 1999 finally laid the basis for modern *labour market monitoring*. In particular, the introduction of a Data Warehouse system allowed for the creation of new observations and concepts, such as the tracking of individual *occupational careers*. Overall, the AMS has constantly striven to improve and modernise its *labour market monitoring*. The most recent addition is a (third-party developed) *logistic regression*-based tool for the prediction of labour market re-integration prospects for AMS clients [45]. Methodically, this tool can be associated with *machine learning* and, thus, for the AMS represents the introduction of advanced *labour market monitoring* methods into its *labour market monitoring* toolbox.

Currently, a growing number of national *public employment services* are utilising methods of *machine learning* to supplement and enhance their national *labour market monitoring*. A recent paper published by Desiere et al. for the OECD [31], which compares different approaches of *machine learning* adopted by different countries, shows that the variety of *machine learning methods* applied ranges from traditional and well-accepted approaches, such as *logistic regression* (e.g. Australia, Austria, Italy, the Netherlands, Sweden, and the United States of Amer-

ica), *factor analysis* (e.g. Latvia) or *probit regression*[1] (e.g. Ireland) to rather new methods, such as *random forest* (e.g. Belgium) and so-called enhanced approaches, examples of the latter being a combination of *random forest* and *gradient boosting* (e.g. New Zealand) or a *big data model* (e.g. Denmark) [31]. Similarly, there is a wide variety in terms of research interests addressed via *machine learning models*. According to Desiere et al., these research interests can be captured by the so-called *target*, meaning the variable to be estimated by a model.[2] Apart from cost-related aspects (e.g. New Zealand, USA)[3], the national *public employment services* reviewed in Desiere et al.'s study mainly focused on the estimation of the *likelihood for long-term unemployment* (e.g. Australia, Belgium), whereas the definition of long-term unemployment varied between "more than 6 weeks" and "more than 12 months". Another aspect, though less frequently pursued, was the *estimation of (re-)integration likelihood into the labour market* (e.g. Austria, Ireland) [31].

Whether or not an individual is able to participate in the *labour market* is determined by several factors. While some of these factors are context-related, such as the overall economic situation or (local or industry-related) market saturation, others are person-related, including personal characteristics, such as an individual's age and sex, but also one's qualifications or preceding career. Especially the latter, i.e. the *pre-career*, has been shown to have an impact on an individual's likelihood to be re-integrated into the *labour market* (i.e. the longer an unemployment episode, the harder re-integration [44]). Thus, *machine learning models* created for addressing (re-)integration have tended to combine several sources of data, such as a client's labour market history, hard and soft skills (e.g. education, communication skills), regional labour market information, or even job-seeker behaviour (e.g. "click-behaviour" on the *public employment service's* website) [31].

The *machine learning approach* of interest for this thesis is *random forest*. *Random forest* is a *supervised machine learning method* which was introduced by Leo Breiman in 2001 [18]. A *random forest* is an ensemble of *decision trees*, which is created by applying Breiman's ***C**lassification **A**nd **R**egression **T**rees (CART) algorithm* [19]. The *random forest* method is able to predict three different types of *targets*, which are reflected in three different types of *random forest*, viz. *regression forests* (for continuous targets), *classification forests* (for a discrete number of classes), and *probability forests* (a combination of *random forest classification* and

[1] *Probit regression* is a special type of *linear regression* with a binary target variable [12].

[2] Depending on the discipline, the *target* is also known as the *response* (e.g. social sciences) or the *dependent variable* (e.g. statistics).

[3] New Zealand's *target* was the *costs of lifetime income support* and the *respective staff costs from offering case management services*, and the USA's *target* was the *likelihood of the complete exhaustion of unemployment benefits*.

regression for binary classification problems [69]). This variety has made *random forest* a popular methodical choice in many different fields. For instance, besides *data science* applications (e.g. the classification of data streams [39]), *random forest* has also been put to use in the natural sciences (e.g. biology [94, 84], medicine [110], physics [105]) and humanities (e.g. social sciences [81, 5, 82], geography [75]), and has also fostered interdisciplinary research (e.g. population mapping by combining geo-spatial data and Census data [103]).

Random forest models also offer a wide range of utilisation in the field of economics. Apart from financial market applications, such as for the prediction of stock price index movements [91] or the development of financial trading strategies [57], *random forest* has also been used in the context of the *labour market*. Gerunov, for instance, employed *random forest* for investigating the determinants of an individual's employment status [38], whereas Boselli et al. used *random forest* in the context of Web labour market intelligence, focusing on the extraction of skill information from online job ads [14]. Khadilkar and Joshi, on the other hand, utilised the *random forest* concept to predict job hopping behaviour and to determine employability of job applicants [55]. Currently, however, apart from the noteable exception of Belgium [31], it appears as though the method of *random forest* has not been frequently used in the context of *labour market monitoring* of *national public employment services*.

Pulling all these strands together, the aim of this thesis is to create a *machine learning model* using the method *random forest* to estimate the *(re-)integration likelihood of unemployed people into the Austrian labour market* and thereby to provide the AMS with a useful addition to its *labour market monitoring* toolbox. Successful (re-)integration, in this thesis, is informed by the concept of *sustainable employment*, which is defined by the AMS as an employment relationship which lasts longer than two months [56]. The model was built with a set of all people registered with the AMS in the year 2017. The data was drawn from the AMS' database and included administrative data (i.e. person-related information and data generated in the context of client support) as well as a self-calculated *pre-career*.

The remainder of this thesis is structured as follows: Chapter 2 describes the context of this thesis, viz. the *Austrian Public Employment Service (AMS)* in general and its *labour market monitoring* in particular. Chapter 3 provides the theoretical framework for *machine learning*. In chapter 4, besides definitions for the *population*, *sample* and *target variable*, the main focus lies on the description of *data collection*, *data preparation*, and *data analysis*. Chapter 5 outlines the model generation process and describes the steps undertaken for *model training*, *tuning*, and *testing*. Chapter 6 provides a closer inspection as well as interpreta-

tions of the results. The final chapter 7 contains concluding remarks as well as recommendations for future research.

Attention should also been drawn to the fact that this thesis has several appendices (A to E), which offer additional information for each chapter. Furthermore, for the sake of readability, parts of the data visualisations, which would not have been readable in print, have been stored online and can be found under this dropbox link [11].

2 Context

The intention of this thesis is to develop a *random forest* model which can predict the (re-)integration likelihood of unemployed people registered with the *Austrian Public Employment Service* (AMS). Thus, the context of this thesis is the AMS and, in particular, its approaches towards *labour market monitoring*.

In the following, this chapter introduces the *Public Employment Service Austria*, its organisational structure, tasks, and objectives. It then presents the development of *labour market monitoring* within the AMS and concludes by narrowing in on the AMS' *occupational career monitoring system*.

2.1 Public Employment Service Austria

The *Public Employment Service Austria* (AMS) is "Austria's leading provider of labour market related services" [86]. Up to the 1990s, the so-called *Labour Market Administration* ("Arbeitsmarktverwaltung") was part of the *Federal Ministry of Labour and Social Affairs* (in short: *Ministry of Social Affairs*).[4] In the year 1994, however, the AMS was separated from the Ministry and emerged as a separately governed unit, changing into the "Arbeitsmarktservice Österreich" (AMS) [80].

The AMS has its own legal personality and is the only organisation in Austria which has the legal form of a *service enterprise of public law* ("Dienstleistungsunternehmen öffentlichen Rechts") [86]. The main characteristic of this legal form is that the AMS covers *sovereign tasks* (official tasks and services regarding unemployment insurance or the employment of foreign workers) as well as *non-sovereign tasks* (services attributed to active labour market policy, such as funding) [113]. Accordingly, the AMS is a public authority as well as a private (not-for-profit-oriented) company.

In the following, this section first presents a brief overview of the AMS' *organisational structure*. After that, it sketches the AMS' main *tasks* and *legal framework* as well as its *objectives* and fields of action. Since *labour market monitoring* is the area of interest for this thesis, this section is concluded with a short history of *labour market monitoring* as done by the AMS.

2.1.1 Organisation

The organisational model of the AMS follows the idea that the development and implementation of an effective labour market policy can only be achieved through coordinated collaboration between the government, employers, and employees and

[4] Over the years, several directorates were added or removed from the Federal Ministry dealing with labour market affairs, which often implied a change in its name. For the sake of simplicity, in this thesis the respective Ministry will be referred to as the *Federal Ministry of Social Affairs*.

the additional consideration of local differences and characteristics [80, p. 17]. Coordination between employers and employees is achieved by involving the *Austrian Social Partnership* in the AMS' *monitoring and decision-making bodies*.

The *Social Partnership* in Austria is a conglomerate consisting of a total of four labour and employer associations (the so-called *social partners*), with each side represented by two organisations. Concerning employees, the representative organisations are the *Federal Chamber of Labour* ("Bundesarbeiterkammer") and the *Austrian Trade Union Federation* ("Österreichischer Gewerkschaftsbund"). With regard to employers, the representative organisations are the *Austrian Economic Chamber* ("Wirtschaftskammer Österreich") and the *Austrian Chamber of Agriculture* ("Österreichische Landwirtschaftskammer"). However, in the context of the AMS' *monitoring and decision-making bodies*, the *Austrian Chamber of Agriculture* is substituted by the *Federation of the Austrian Industry* ("Industriellenvereinigung") [86, 89].[5] So, following this AMS particularity, and for reasons of simplicity, the notion of *social partners* in the following includes the *Federation of the Austrian Industry* and excludes the *Austrian Chamber of Agriculture*.

The organisational structure of the AMS consists of three levels. Besides the federal level, there are also organisational units for regional and local offices, the latter in particular demonstrating the operationalisation of taking into consideration local differences and characteristics. Figure 1 illustrates these different organisational levels for the AMS:

[5]The exact composition of the *monitoring and decision-making bodies* on all organisational levels of the AMS are defined in Paragraphs 5, 13, and 20 AMSG (*Federal Public Employment Service Act*).

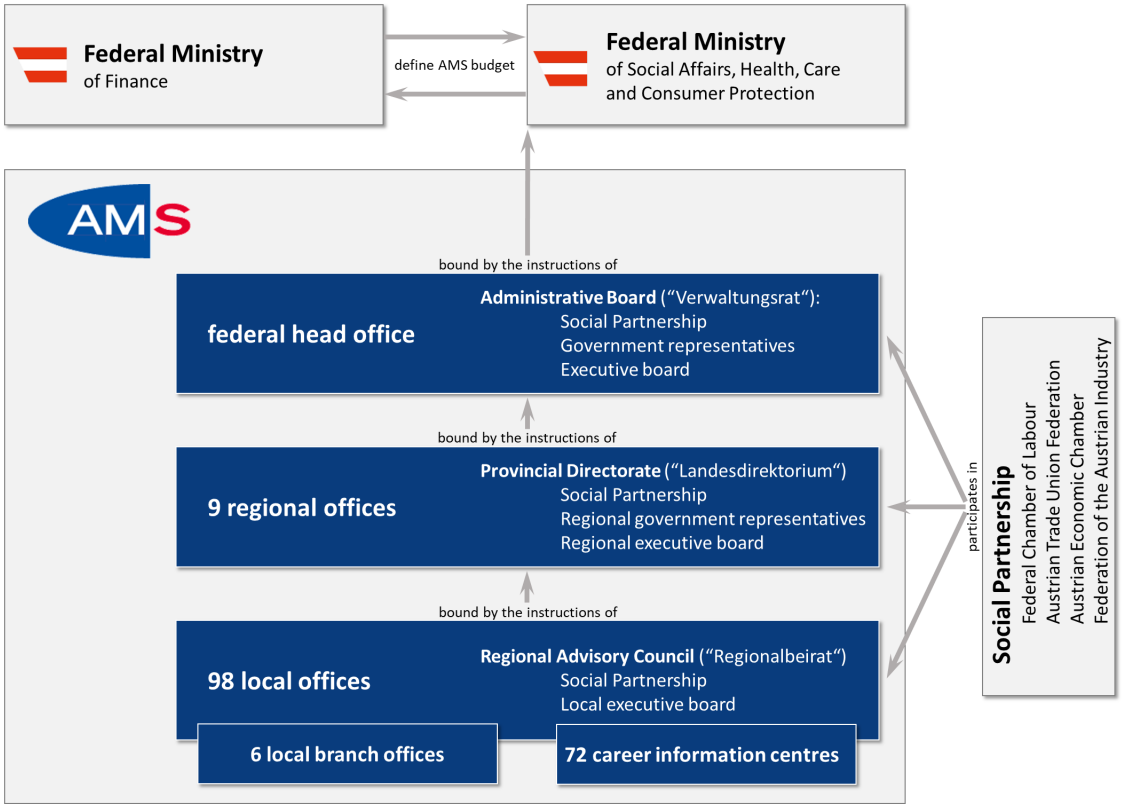

Figure 1: Organisation of the *Austrian Public Employment Service* consisting of the *federal head office*, nine *regional offices*, and 98 *local offices*. For some *local offices*, there are *local branch offices*. Furthermore, the AMS maintains 72 *career information centres* within *local offices*. Each of the *monitoring and decision-making bodies* comprises representatives of *social partners* and members of *executive boards* (and in some cases, *representatives of the government*). Each organisational level is bound by the instructions of the level above. The highest level is the *Federal Ministry of Social Affairs* which, together with the *Federal Ministry of Finance*, defines the AMS' budget for personnel and material expenses. (Adapted from [80])

The AMS consists of one *federal head office* ("Bundesgeschäftsstelle"), nine *regional offices* ("Landesgeschäftsstellen"), and 98 *local offices* ("Regionale Geschäftsstellen"). The two federal states of Styria and Vorarlberg together have six additional *local branch offices* ("Zweigstellen") for some of their local offices.[6] On top of that, the AMS maintains 72 *career information centres* ("Berufsinformationszentren").

Within the hierarchy of the AMS, each lower organisational level is bound by the instructions of the higher organisational level. At the very top, the *federal*

[6]If beneficial in terms of service provision, the *provincial directorate* ("Landesdirektorium") is permitted to outsource parts of *local offices* to *local branch offices*, which then offer these outsourced services (Paragraph 23, section 2 AMSG) [90].

head office is bound by the instructions of the *Federal Ministry of Social Affairs*. It is important to note that the AMS budget for personnel and material expenses ("Präliminarien") is defined by both the *Federal Ministry of Social Affairs* and the *Ministry of Finance*.

On all organisational levels, representatives of labour and employer organisations are involved in the AMS' monitoring and decision-making process [86]. These *monitoring and decision-making bodies* are organised as follows:

- federal level: *administrative board* ("Verwaltungsrat")
- regional level: *provincial directorate* ("Landesdirektorium")
- local level: *regional advisory council* ("Regionalbeirat")

On the federal level, the *administrative board* ("Verwaltungsrat") is tripartite, comprising *social partners*, *government representatives* and the *AMS' board of directors*. On the regional level (federal states), the *provincial directorate* ("Landesdirektorium") comprises the *social partners*, the *regional executive board*, and in some cases *representatives of the regional government*.[7] Finally, on the local level, the *regional advisory council* ("Regionalbeirat") comprises the *social partners* and the *local executive board* [80, 86].

2.1.2 Tasks

As previously mentioned, the AMS is a public authority which carries out both *sovereign* and *non-sovereign tasks* [113]. According to Bock-Schappelwein et al., the three main tasks of the AMS can be summarised as follows [13, p. 10]:

- placement of unemployed people on job offers
- protection of employment
- livelihood security

In a nutshell, for the AMS this means that all efforts are geared towards establishing and maintaining a preferably complete, economically sensible, and sustainable reconciliation of labour supply and labour demand (i.e. to provide labour force to the economy and, at the same time, to secure employment for all people available for the Austrian labour market), and to safeguard people's economic existence during phases of unemployment (by granting statutory provisions) [90,

[7] Representatives of the regional government can be called upon in a consultative capacity if the regional government financially participates (to a certain amount) in labour market policy measures, such as by funding projects (Paragraph 13, section 2 AMSG) [90].

p. 20]. In practice, this translates into matching job-seekers with job opportunities (while at the same time taking into account people's job aspirations), mitigating circumstances hindering job placements, preserving job opportunities, and alleviating quantitative and qualitative disparities between labour supply and labour demand. Further still, the AMS is tasked with ensuring the provision of occupational training opportunities for youths (by placing them on apprenticeship offers) as well as with fostering the re-employment of people with health impairments (by placing them on suitable job offers and by offering additional and preparatory measures while taking into consideration the individual's capabilities) [90, p. 20].

All tasks carried out by the AMS are laid down by law. The previously mentioned three main tasks, for instance, are made concrete in the *Federal Public Employment Service Act* ("Arbeitsmarktservicegesetz, AMSG"). The disbursement of wage replacements (e.g. unemployment benefit) is stipulated in the *Unemployment Insurance Act* ("Arbeitslosenversicherungsgesetz, AlVG"), whereas measures regarding the employment of foreigners are to be found in the *Employment of Aliens Act* ("Ausländerbeschäftigungsgesetz, AuslBG") and occupational training opportunities for youths are normalised in the *Vocational Training Act* ("Berufsausbildungsgesetz, BAG") [80]. With regard to active labour market policy issues (e.g. customer advisory services, funding), the AMS acts within the scope of Austrian private law.

2.1.3 Objectives

While the AMS' tasks are laid down by law, its objectives are determined by a number of political and strategic programmes [80]. These programmes, which are dependent on one another, are listed below:

- *Europe 2020* (EU-programme for growth and jobs, intended to strengthen Europe's competitiveness and productivity by fostering smart, sustainable, and inclusive growth [71])
- *Austrian National Reform Programme* (national strategy for implementing the objectives of the Europe 2020 programme [20])
- *labour market policy objectives*
- *AMS long-term plan*
- *Gender Equality and Women's Promotion Programme*
- *annual labour market policy objectives*

Starting from the top, *Europe 2020* defines the overall growth objectives for the European Union. The member states translate these objectives into a national strategy paper called the *National Reform Programme*. As economic growth is a cross-sectional matter, several Federal Ministries are involved in composing this programme. In terms of labour market aspects, the *Federal Ministry of Social Affairs* defines rather general *labour market policy objectives* ("Arbeitsmarktpolitische Zielvorgaben"), also known as the *Ministry objectives*. These objectives are specified for particular target groups (e.g. youths and apprentices, women, older people and people with health impairments, long-term unemployed and handicapped people, migrants and refugees), but also address management and organisational objectives (e.g. organisational development, optimisation of placement and matching processes) [22, 21].

Taking account of the *AMS' long-term plan* and the *Gender Equality and Women's Promotion Programme*, the AMS' *federal head office* translates the generic *Ministry objectives* into a proposal for specific and measurable *annual labour market policy objectives* ("Arbeitsmarktpolitische Ziele"). This proposal is then presented to the *administrative board* for its approval. After approval, the annual objectives are put into force. Examples for specific and measurable objectives are the achievement of *"sustainable employment (at least 2 months) of people under the age of 45 years with a net unemployment duration of at least 12 months"* [85] or the *"reduction of long-term unemployment of people younger than 25 years"* [85, 87].

2.2 Labour market monitoring

As a precondition for accomplishing its tasks, the AMS is legally obliged to perform *labour market monitoring and statistics* as well as to conduct *labour market* related fundamental and development work and research (Paragraph 30, section 1 AMSG) [90, p. 20]. *Labour market monitoring* is the systematic analysis of labour market data and the development and compilation of reports in order to support internal decision-making processes, but also to be able to provide information to the public. Additionally, the data and statistics generated in the context of labour market monitoring are an important source for *labour market research* (e.g. for conducting labour market projections) [6]. Overall, the precondition of any *labour market monitoring approach* is the (preferably) comprehensive collection of relevant aspects of labour supply (unemployed people and people in employment relationships) and labour demand (available vacancies).

In the following, this section first provides an overview of *central monitoring concepts*. It then continues with a brief history of the AMS' *traditional labour market monitoring approach* dating back to times before automated computer-aided data processing was in place, and concludes with the AMS' computer-aided

current labour market monitoring approach.

2.2.1 Central monitoring concepts

The idea of segmenting the labour market and describing differences between submarkets is central to *labour market monitoring* and has a long tradition within the AMS. For example, according to the *Documentation on National Statistics of the Labour Market Administration*, the *labour market administration* already in 1946 started to regularly create statistics tables featuring labour market aspects which were segmented along different characteristics, such as *sex* or *age groups* [46]. In the context of *labour market monitoring*, three main concepts are relevant:

- time
- measure
- characteristic

The first concept, *time*, can refer to specific pre-defined points in time (e.g. *report dates* at the end of any month[8]) or to a period of time (e.g. the time between two consecutive *report dates* or the time between the beginning and the end of certain events).

The second concept, *measure*, refers to data which is used for gaining a particular perspective on the labour market. In the context of *labour market monitoring*, the AMS uses time-related *stock and flow measures* as well as *ratios* or *quotas*. For instance, when the number of *unemployed people* in a certain month is put into relation with the respective *report date*, this represents a *stock measure*. In contrast, when the number of *people which became unemployed* within a month is related to the time-period between the respective *report date* of that particular month and the *report date* preceding that month, this "inflow" (into the group of unemployed people) is considered a *flow measure*. The *unemployment rate*, on the other hand, is an example of a *quota*.

The third important concept is that of *characteristic*. A *characteristic* restricts a *measure*, as it acts like a filter. Consequently, the labour market (or parts of the labour market) can be split into segments according to characteristics. Examples for commonly used characteristics in the context of unemployed people are *sex*, *age*, *last occupation*, or *highest educational attainment*.

[8] In some cases, the middle of the month can also be used as a *report date*. This is mostly relevant for internal processes.

2.2.2 Traditional approach

Until the 1970s, *labour market monitoring* within the AMS was a laborious task, as it had to be conducted manually. Due to the amount of reports, which had to be created on a regular basis, the process of *labour market monitoring* demanded for high personnel peak efforts. This was the reason why the introduction of the concept of a *report date*, i.e. a defined point in time for stocktaking, was not only an evident, but also a necessary decision.

At the end of every month (*report date*), the employees of the local offices collected the information recorded for all unemployed people registered with their local office. This was done by scrutinising the so-called *working cards* ("Arbeitskarten") of those clients that were being serviced at the time.[9] Along with basic client information and information about benefit receipt, these paper cards featured small colour-coded tabs at predefined positions at the top of the card, indicating certain client characteristics, such as *sex* or *age group*.

The information gathered by the *local office's* statistician was organised in the form of standardised tables. These standard tables were then submitted (either in person or by mail) to the statistics departments of the *regional offices*. The regional statisticians consolidated this information and reported the resulting standard tables to the *Federal Ministry*, where the respective regional standard tables were consolidated into standard tables covering the nine federal provinces of Austria. Finally, the standard tables were reported to the Minister, who would then communicate the figures to the public. Depending on the number and complexity of the reports, on average this process could take between a couple of days and a couple of weeks [92].

2.2.3 Current approach

Labour market monitoring within the *Labour Market Administration* started to change with the introduction of the first labour market related IT-systems. As of the mid 1970s, this meant a step-wise introduction of hardware and software elements. One of the first elements to herald IT-based *labour market monitoring* was the introduction of a terminal-based mainframe computer system for job vacancies administration, which was first implemented in the federal states of Vienna and Lower Austria [64, 59]. In the following years, until the mid 1980s, these terminals were rolled out step-by-step in the *local offices* of the remaining federal states [58, 60, 61].

In the year 1984, the next developmental step was the introduction of an IT-based client administration system called PST ("Personenstamm") [62]. The newly

[9] See Appendix A for an example of a *working card* from the year 1946.

implemented PST, in combination with the already existing job vacancies administration tool, fostered automated *matching* of job seekers with job vacancies [63, 50]. This merging of labour market supply and labour market demand in one single system laid the foundations for the AMS' contemporary *labour market monitoring*.

A major milestone was reached in 1987 with the realisation of the statistical labour market information system SAMIS ("Statistisches Arbeitsmarkt Informations System").[10] This, back then, future-oriented database system enabled the creation of *report date*-related statistical information in only a fraction of time. The SAMIS system featured two main applications, viz. the (pre-defined) SAMIS standard reports and a query-function called SAMFA ("Freie Abfrage") [96, 74]. In the following years, only rather minor additions were made to the existing system, such as the inclusion of aspects concerning the employment of foreigners [48] or the expansion of self-service systems [50, 48, 49].

The years 1989 to 1998 saw a large-scale modernisation of the *Labour Market Administration*'s IT infrastructure. During this time, the terminal-based system was gradually replaced by a network of OS/2-based personal computers [47, 74]. Further developments were the integration of other new hardware and software applications, such as the use of the computer mouse, the export of data and standard tables in Excel-format [96], or a change in the operating system to Windows 3.11 [74].

The next major *labour market monitoring* landmark was the introduction of a Data Warehouse system, which went online in October 1999 [74]. Firstly mentioned in the AMS' long-term plan for the years 1997 to 1999, the Data Warehouse system was intended as a management information system for creating reports that were precisely tailored (content- and time-wise) to the informational needs of decision-makers [4]. Simultaneously, the old personal computers' operating system was replaced by Windows NT [74].

In 2003, the Data Warehouse system by Cognos (in the meantime incorporated by IBM), an analytical tool for fast and simple queries of data from predefined cubes[11], allowed for the implementation of the so-called *occupational career monitoring* ("Erwerbskarrierenmonitoring, EWKM") [25]. The *occupational career monitor* combines several (AMS-)internal and external data sources and comprises

[10] The idea for SAMIS had in fact already been borne in 1975. As a large-scale project, however, it had required an extensive planning stage and was finally introduced in 1987 [96].

[11] A *Data Warehouse cube* is a package consisting of a collection of measures (i.e. stock and flow measures, quotas and ratios) and dimensions (e.g. age, sex, education). The notion of *cube* refers to the dimensionality, imagined as geometrical forms (or arranged in the form of a coordinate system). Following this reading, two dimensions would lead to a square, three dimensions to a cube, four dimensions to a hypercube, and so forth. The human imagination, however, usually ends with four spatial dimensions. From this point of view, the designation "cube" could appear slightly artificial or even misleading.

all people covered by the Austrian social security system [24]. As the *occupational career monitor* plays a central role for this thesis (in terms of both the definition of the *population* as well as the creation of the *target variable*), it will be presented in more detail in the subsequent section 2.3.

The latest *labour market monitoring*-related milestone within the AMS concerns the introduction of AMAS ("Arbeitsmarktchancen Assistenz System"), a labour market assistance system, which - at the time of writing - was foreseen to go live in 2021.[12] The main idea of the model is to split up the client base into three segments (high, medium, or low), which represent a person's prospect of (re-)integration into the labour market [45]. The main goal is to better allocate client funding. As the creators of the model chose a *persona-based approach*[13], it is possible to (roughly) determine a client's labour market re-integration prospects. Furthermore, the model is able to provide insights into aspects which might (even) hinder a higher degree of labour market success.

2.3 Occupational career monitoring

As the concept of a client's *occupational career* is central to the analytical purposes of this thesis (see section 4.1), this section is devoted to introducing this concept in more detail. As the name suggests, the *occupational career monitoring system* (EWKM) was developed within the AMS as a labour market monitoring tool which is used specifically for monitoring a client's *occupational career*. The main objectives of *occupational career monitoring* are to find industries with high chances for job seekers to find work, to identify (at an early point in time) groups of people with an increased risk of job loss, and to determine the stability of labour market segments in order to estimate their degree of demand fluctuation. Furthermore, the monitor provides information about labour force flows and is used as an information support tool for adjusting and further developing the Austrian social security system [88].

The underlying concept of the *occupational career monitoring system* is that of the *universal labour market status* (in short: *Uni-status*). There are 74 different, hierarchically ordered *labour market statuses*, which are assigned to three supercategories, viz. *AMS registration*, *employment*, and *other* (statuses). The *Uni-status* represents a person's (current) status and, thus, the person's position in the labour market.

[12] AMAS is an externally created *logistic regression segmentation model* developed on the basis of *occupational career monitoring* data (see section 2.3).

[13] The developers of the AMAS model created a list of 81,000 pre-calculated labour market re-integration prospects (i.e. personas) from combinations of nine out of 14 variables [45]. By comparing the characteristics of an actual client with these personas, it is possible to discern the influence of certain variables on a client's individual labour market success.

In order to describe an individual's complete *occupational career*, data from different sources needs to be combined. The relevant sources are AMS-data (e.g. client master data) and insurance data provided by the *Main Association of Austrian Social Security Institutions* ("Dachverband der österreichischen Sozialversicherungsträger"), data about disabilities sourced from the *Service of the Ministry of Social Affairs* ("Sozialministeriumsservice"), and enterprise data gleaned from the *Social Insurance Institution of the Industrial Economy* ("Sozialversicherungsanstalt der gewerblichen Wirtschaft"). The combination of these sources results in a series of different statuses which form the *occupational career*.[14]

Occupational careers can show overlapping statuses. This problem is tackled by following a predefined status hierarchy, meaning that in the case of colliding statuses, the status with the highest priority is preferred. The following figure 2 displays an exemplary combination of statuses from two different sources (viz. insurance data and AMS-data) and the resulting non-overlapping *occupational career* recorded by the AMS (via EWKM):

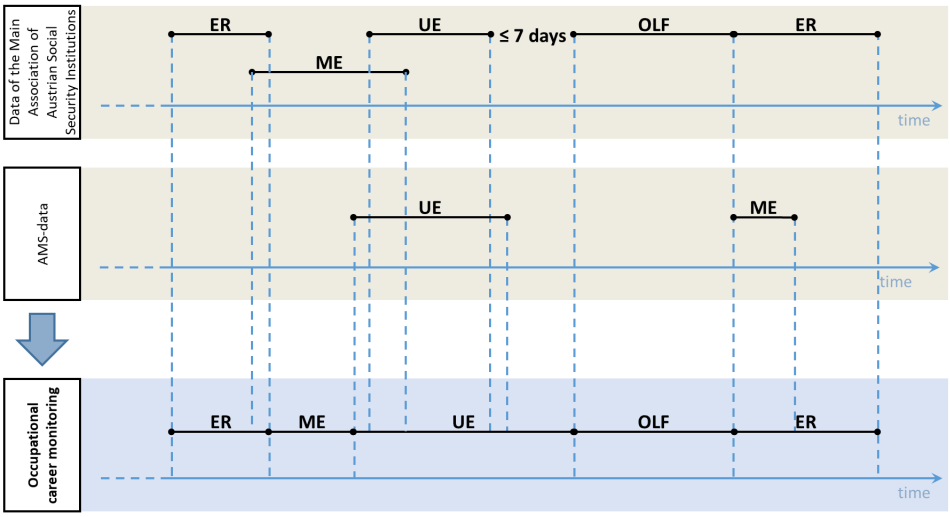

Figure 2: Exemplary non-overlapping *occupational career* composed from *Uni-statuses* of two sources. By combining information from the *Main Association of Austrian Social Security Institutions* (upper bar) with AMS-data (middle bar), a non-overlapping *occupational career* (lower bar) is created. *Employment relationships* (ER), *unemployment episodes* (UE), and *out-of-labour force* (OLF) have higher priorities than *marginal employments* (ME). Gaps between periods of insurance are assigned to the preceding status, as long as the gap is smaller or equal to seven days. Otherwise, the gap receives its own status. (Adapted from [88])

The upper brown bar in figure 2 exhibits a series of Uni-statuses, e.g. *employment relationships* (ER), *marginal employment* (ME), *unemployment episodes*

[14]Technically speaking, the *occupational career monitoring system* is a set of Data Warehouse cubes which provide blended data from these different sources.

(UE), and *out-of-labour force* (OLF).[15] The lower brown bar represents episodes recorded by the AMS, i.e. a period of *unemployment* (UE) and an episode of *marginal employment* (ME). By combining the two sources and by observing the statuses' hierarchies, the blue bar constitutes an individual client's complete, *non-overlapping occupational career*. If there are gaps between periods of insurance, these are assigned to the respective preceding status (as long as the gap is smaller or equal to seven days; otherwise, the gap receives its own status).

[15]For the sake of simplicity, the displayed status labels in figure 2 are generalised towards their respective super-category (see Appendix A, table 3, for a list of *Uni-statuses*, including their hierarchy rank and assigned *categories* and *super-categories*).

3 Theory

The aim of this chapter is to provide an overview of one of the most commonly used machine learning approaches, viz. *random forest*. Random forest is an *ensemble method*, meaning that a forest consists of numerous *decision trees* [105]. Thus, in order to understand the method of *random forest*, it is important first to understand the concept of *decision trees*.

This chapter begins with an introduction to *machine learning*. This is followed by overviews of the concept of *decision trees* and of the method of *random forest*. Finally, this chapter closes with a review of *model evaluation* strategies and measures.

3.1 Machine learning

Machine learning is a sub-field of artificial intelligence [5] and, according to Harrington, "lies at the intersection of computer science, engineering, and statistics" [42, p. 5]. The basic idea of *machine learning* is to automatically gain information from large amounts of data by handing over the learning activity to computer algorithms.

In the following, this section first presents the two common types of *machine learning*, viz. *supervised* and *unsupervised machine learning*. It then continues to present an overview of a typically applied *machine learning process*.

3.1.1 Supervised and unsupervised machine learning

Machine learning can be divided into two groups, viz. *supervised* and *unsupervised learning*. *Supervised learning* entails the definition of a qualitative or quantitative *target variable* which can be predicted by a set of input variables [26]. Well-known representatives of *supervised learning* are regression or clustering methods, such as linear and logistic regression, K-Nearest-Neighbor, Support Vector Machines, or decision tree approaches [118, 42]. *Unsupervised learning*, on the other hand, has no target variable. Instead, the machine looks for similarities in the data (*clustering*), determines the distribution of data (*density estimation*), or reduces data dimensionality down to two or three dimensions (for better data visualisation) [10]. Commonly known representatives of *unsupervised learning* approaches are, for instance, K-Means clustering, unsupervised (deep) learning neural networks, or Principal Component Analysis. Overall, while for *supervised learning* it is necessary to have knowledge about the data, this is not a necessary prerequisite for *unsupervised learning*.

This thesis focuses on *supervised learning* methods, specifically on the method of *random forest* and its foundation, *decision trees*. In general, there are two

different types of problems which *supervised machine learning* techniques can solve, viz. *classification* and *regression problems*. According to Harrington, the main difference between these two problems lies in the number of possible expressions to be gained for the *predicted variable* [42]. In the context of *random forests*, this variable is referred to as the *target* variable.[16] Another difference concerns the type of the *target*'s expressions. In terms of *classification*, for instance, the number of classes is finite and the classes are represented by nominal or ordinal values (e.g. different classes of sex, marital status, or educational level). In terms of *regression*, on the other hand, the number of expressions (in theory) is infinite and expressions are numeric (e.g. weight in kg, income in EUR).

Depending on the problem at hand, the most suitable *machine learning approach* is put to use. For *regression problems*, typical approaches are linear regression, support vector regression, or decision tree regression. For *classification problems*, possible approaches are logistic regression, Naive Bayes, or decision tree classification [42].

3.1.2 The machine learning process

Machine learning works according to a specific process. In his book *Machine Learning in Action*, Harrington [42] presents a generic list of steps which are taken in the course of a (typical) *machine learning processes*. These steps for *supervised learning* (turquoise) and *unsupervised learning* (violet) are depicted as a flow graph in figure 3 below:

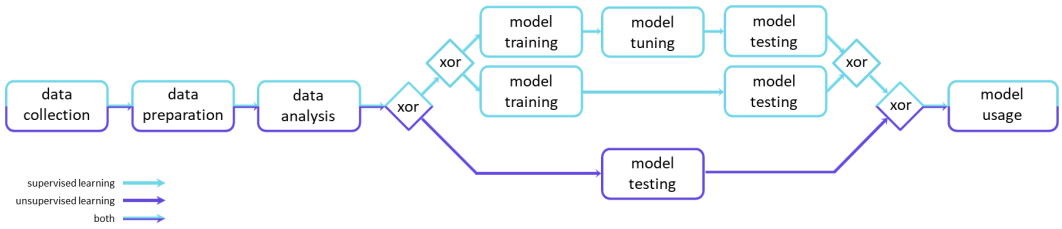

Figure 3: Generic steps of *machine learning approaches*. The steps for *data collection*, *data preparation*, and *data analysis* are typically carried out for both *supervised* and *unsupervised learning*. Concerning actual model generation, *supervised learning* can employ either three steps, viz. *model training*, *model tuning*, and *model testing* (upper turquoise route) or two steps, viz. *model training* and *model testing* (lower turquoise route). *Unsupervised learning*, on the other hand, relies on *model testing* only (violet route). After having completed *model testing*, the model can be put to use.

The process starts with *data collection* (e.g. downloading data from a database). In a next step, this data needs to be *prepared* to meet the preconditions of a certain

[16] Depending on the discipline, the *predicted variable* is also known as the *response* (e.g. social sciences) or the *dependent variable* (e.g. statistics).

algorithm and/or platform (such as the statistical software *R*, which was used in this thesis). This can mean, for instance, that dummy-variables are created from categorical data or that columns with high amounts of missing data are dropped. The next stage is *data analysis*. Here, data visualisation is of great importance, as certain patterns may already become salient (e.g. age-specific differences).

The next steps are the ones in which actual *machine learning* takes place. Each of the stages (*model training*, *model tuning*, and *model testing*) requires data (i.e. a particular data set) in order to be operationalised. For *supervised machine learning* (turquoise routes), for instance, the prepared and analysed data is split up into either two or three parts, which results in different data sets for the different model generation stages. In the case of a *two-fold split*, the data is split into a *training set* and a *test set*. As the name suggests, the *training set* is used to train the model. This set is typically between two thirds and 80% of the total data set [8, 70, 105, 81]. The remaining *test set* is used for (iterative) *tuning* and *testing*. In the case of a *three-fold split*, the *training set* is further subdivided into the *actual training set* and a *validation set* [41], whereas the latter is explicitly used for *tuning* the model. The choice of *split regime* is dependent on the *evaluation strategy* used (see section 3.4.1), meaning that the model is evaluated either with unfamiliar data by using additionally sourced data (*two-fold split*), or with familiar data by drawing on data emergent from the data set at hand (*three-fold split*). With *unsupervised learning* (violet route), no splits are performed, as model generation essentially is a continuous process of *model testing* for which the entire data set is used [42, p. 12].

Regardless of the *split regime* used, the testing process is conducted under a certain *evaluation regime* (e.g. *k-fold cross-validation*), which goes hand in hand with choosing a suitable *performance measure* (e.g. *Pearson correlation between predicted and true values* [84], *mean squared error*, *area under the ROC curve* [27]). Finally, having completed *model testing*, the model can be used for the purpose intended.

From the *machine learning approaches* presented above, this thesis opted for *supervised machine learning* allowing for a *model training*, *model tuning*, and *model testing* phase (upper turquoise route). Concerning the choice of *split regimes*, this thesis employed both a *two-fold split* and a *three-fold split*.

3.2 Decision trees

In order to understand the concept of *random forest*, it is necessary to first understand the underlying concept of *decision trees*. *Decision trees* are a method of *supervised machine learning* [42]. According to Loh [68], the beginnings of this method can be traced back to the early 1960s, whereas scholarly research and interest intensified in the 1980s, in particular as a result of Qunilan's seminal pub-

lication *Induction of Decision Trees* [95]. The method of *decision trees* is one of the most important knowledge representation approaches, which works by using a top-down model to project high-dimensional data to low-dimensional space [120].

This section is intended to present the theoretical framework for *decision trees*. It therefore first introduces *graphs* as the foundations of *decision trees*, then moves on to the process of decision tree *growing* by observing *splitting rules*, and concludes with the method of *pruning*, which is performed in order to avoid model overfitting.

3.2.1 Graphs, trees, and decision trees

The mathematical understanding of a graph, as defined by Aho, Hopcroft and Ullman is that a graph $G = (V, E)$ consists of a finite, not empty set of vertices V and a set of edges E [1, p. 50]. A graph is said to be *directed*, if the edges are ordered pairs of vertices (v, w), where v is the *tail*, and w is the *head* of an edge (v, w). A graph is said to be *undirected*, if the edges are unordered pairs.

A *path* in a graph is a sequence of graphs, whose edges are connected.[17] Furthermore, a path is called *simple*, if all edges and vertices on the path (except for the first and the last vertex) are distinct. The special case of a simple path of the length of at least 1, which begins and ends at the same vertex, is called a *cycle*. Figure 4 below provides examples of connected graphs:

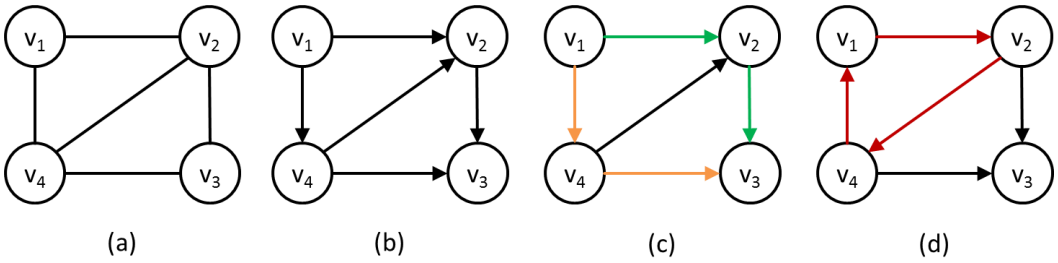

Figure 4: Connected *undirected* (a) and *directed* graphs (b, c, d). Graph (c) visualises two different paths from vertex 1 to vertex 3: $p_1 = (v_1, v_2), (v_2, v_3)$ (green) and $p_2 = (v_1, v_4), (v_4, v_3)$ (orange) (*edge representation*). There is also another path from vertex 1 to vertex 3, viz. $p_3 = v_1, v_4, v_2, v_3$ (*vertex representation*). The paths in (c) are *simple*, as all edges and vertices on the path are distinct. The highlighted path (red) in (d) represents a *cycle*. (Adapted from [1])

While graph (a) is *undirected*, graphs (b), (c), and (d) are *directed*. Graph (c) illustrates *simple paths* (green and orange), as all edges and vertices on the path are distinct. Graph (d) shows an example of a *cycle* (red).

Trees are special types of graphs. Following the definition of Aho, Hopcroft and Ullman, a tree (or *rooted tree*) is a (connected [115]) directed acyclical graph

[17] Paths can be described either by *edge representation*, e.g. $(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n)$, or by *vertex representation*, e.g. $v_1, v_2, ..., v_n$.

which satisfies the following properties [1, p. 55f]:

1. There is exactly one vertex with *no* entering edges (the *root*).
2. Every vertex (except for the root) has exactly *one* entering edge.
3. There is a *unique path* from the root to each vertex.

In other words, a *tree* begins at its *root*. This *root* does not receive any information, but already contains all the information which is subsequently used for growing the *tree*. With the *root* as the starting point, the tree grows by branching out, creating unique directed paths from the root to each vertex. Each vertex has only one entering edge, but can produce several *branches*. Figure 5 illustrates a generic tree and its basic elements, which are described in terms of the vocabulary used in the context of *random forests*:

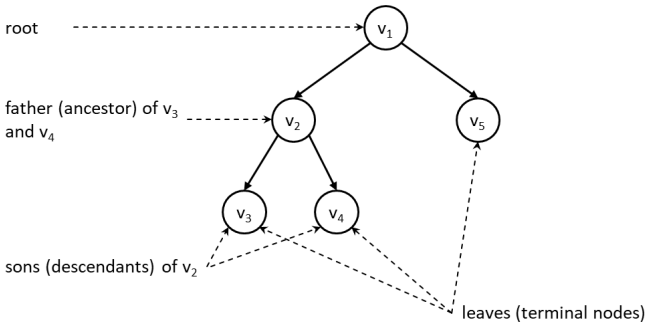

Figure 5: Generic *tree* with basic elements. The *root* is the starting point of each tree. The *node* which passes information to the next node, in a directed manner, is called the *father* (*ancestor*), the node which is informed by this (one) entering edge is called the *son* (*descendant*). A *son*-node which has descendants can, at the same time, be the root of a *subtree*. A node which has no descendants is called a *leaf* or (*terminal node*).

Starting from top to bottom, the *root* (v_1) is the starting point of each tree. The *node* which passes information to the next node, in a directed manner, is called the *father* (v_2), the node which is informed by this (one) entering edge is called the *son* (v_3 and v_4). As such, the father is the *ancestor* and the son is the *descendant*. This relationship between the ancestor and the descendant is said to be *proper* (meaning *non-cyclical*). A node with no proper descendants is called a *leaf* (or *terminal node*) (v_3, v_4, and v_5). A node and all its descendants are referred to as a *subtree*.

A *forest* is a collection of several *trees*. *Trees*, however, take on many different shapes and sizes. Consequently, efforts have been made to describe different aspects of trees in a more general manner. A useful description is provided by Aho,

Hopcroft and Ullman [1]:[18]

- *depth of a node*: the length of the path from the *root* to the *node*
- *height of a node*: the length of the longest path from the *node* to a *leaf*
- *height of a tree*: the height of the *root*
- *level of a node in a tree*: the height of the *tree* less the depth of the *node*

Another important aspect of trees is that they can be *ordered*. An *ordered tree* is a tree whose sons of each node are ordered, usually from left to right. A special case of an ordered tree is a *binary tree*, which is a tree within which a node has at most two sons, i.e. a *left son* and a *right son*. In a *complete binary tree*, each node (apart from a *leaf*) has both a left and a right son [1, p. 53]. These types of *trees* are illustrated in figure 6:

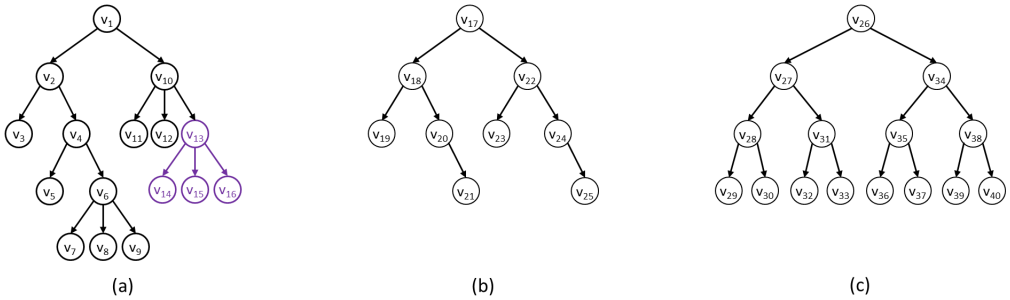

Figure 6: A *forest* as an ensemble of trees. Tree (a) is an *ordered tree*, tree (b) is a *binary tree*, and tree (c) is a *complete binary tree*. In (a), the highlighted nodes (violet) are an example of a *subtree*, with v_{13} being the subtree root, and v_{14}, v_{15}, and v_{16} its sons. In combination, trees (a), (b), and (c) constitute a *forest*. (Adapted from [1])

Decision trees are a particular type of trees and have been defined and described by several scholars. Moret [78], for instance, describes a decision tree as a rooted, ordered, vertex-labelled tree, wherein each node has either m_i children (for some i, $1 \leqslant i \leqslant n$) or no children (which makes the node a *leaf*) [78, p. 598]. With particular regard to *binary trees*, Chang and Pavlidis define a decision tree to be *binary* if all of its decision functions are double functions [28, p. 29], meaning that the result of a decision leads to exactly one element of the Boolean domain (*true* or *false*). Apart from being *binary* (distinguishing between two classes), decision trees can also be *k-ary* (distinguishing between more than two classes). A *complete*

[18] It should be noted that Aho, Hopcroft and Ullman actually use the expression *vertex* instead of *node*. In the context of *decision trees*, however, *vertices* are called *nodes*, which is why this denomination was chosen for introducing this description.

(or *completely balanced*) *k-ary* decision tree is a decision tree in which the decision function at each non-leaf node is *k-ary* and each path from the *root* to a *leaf* has the same length [28].

The purpose of *decision trees*, as special types of trees, is not simply to organise information, but to gain more knowledge from the data. Essentially, a decision tree approach is a *multi-stage approach*, meaning that decisions are refined at multiple stages. In the words of Safavian and Landgrebe [97], the basic idea of multi-stage approaches is "to break up a complex decision into a union of several simpler decisions, hoping the final solution obtained this way [will] resemble the intended solution" [97, p. 660]. A *decision tree* represents the complex decision, *nodes* within the tree represent the simpler decisions [28]. From a different angle, Hauska and Swain [43] describe a decision tree as a maximum likelihood classifier using multi-stage decision logic. Such a classifier is able to classify an unknown sample into a class using one or several decision functions in a successive manner [43, p. 4]. In that context, a *leaf node* (also called an *answer node* or a *terminal node*) contains a *class name* (or *class label*, e.g. "1", "0"), whereas a *non-leaf node* (or *decision node*) runs an attribute test for each possible value of the attribute [112]. Similarly, Moret [78] describes a *decision tree* as a model of the evaluation of a discrete function, within which the value of a variable determines the next action to be taken. Such an action can either be to choose another variable to be evaluated or to output the value of the function.

As previously described for (ordinary) *trees*, for *decision trees* too, the *decision path* is the path from node x to node y [28, p. 29]. Also, the top node (the *root*) contains the entire sample, whereas the remaining nodes contain subsets of their respective ancestor nodes [115]. In other words, each interior node in a *decision tree* represents a decision [1] which splits up the sample and drops down from the *root* to a *leaf*. One way to illustrate the outcome of how *decision trees* make decisions is via *partitioned feature space*, an example of which is depicted in figure 7:

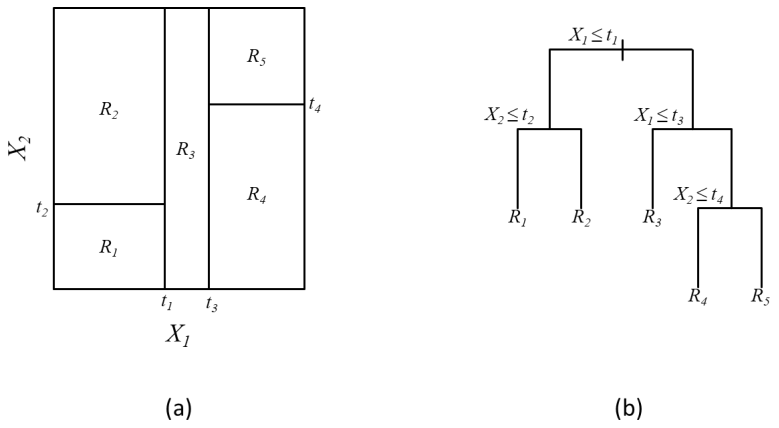

Figure 7: *Decision tree* and corresponding *feature space*. Illustration (a) is a *partitioned feature space* and (b) is the corresponding decision tree. The *feature space* comprises features X_1 and X_2, and is partitioned into $R_1, ..., R_5$ classes. The corresponding *decision tree* shows left-hand and right-hand splits, which depend on whether the variable is lower (left-hand split) or higher (right-hand split) than the *splitting threshold* t_k. (Adapted from [52])

As sketched above, the *feature space* (or *predictor space*) (a) comprises the set of predicting variables X. By means of decision tree classification, this space is partitioned into R classes. The corresponding *decision tree* shows left-hand and right-hand splits, which emerge depending on whether the variable returns as lower (left-hand split) or higher (right-hand split) value than the *splitting threshold* t_k (see section 3.2.3).

Essentially, the purpose of creating a *random forest model* is to be able to make a prediction concerning a particular feature. In *decision trees*, *nodes* are referred to as *predictors* (or *feature variables*) and *leaves* are referred to as the *response* (or *target*) [115]. The *random forest* method is able to predict three different types of *targets*, which is also reflected in three different types of *random forest* which can be created, viz. *regression forests* (for continuous targets), *classification forests* (for a discrete number of classes), and *probability forests* (a combination of *random forest classification* and *regression* for binary classification problems [69]). Furthermore, the decisions taken in a *decision tree* follow a predefined algorithm (e.g. ID3, C4.5, CART). In a way, the algorithm puts the "decision" into *decision trees*. More precisely, this algorithm comprises three essential aspects, viz. the *tree growing direction*, the *splitting rule* applied, and the type of *pruning* performed. These aspects are presented in more detail in the following sections.

In this thesis, the two types of *random forest* chosen to address the research interest were *classification forests* and *probability forests*. Furthermore, the algorithm applied was Breiman's CART algorithm.

3.2.2 Tree growing

The first aspect of a decision tree algorithm is the *tree growing direction*. Safavian and Landgrebe [97] describe four basic tree-growing approaches:

- *bottom-up*: The process of growing starts with the information classes and continues to combine classes until only one class is left. The *root* contains all the classes.
- *top-down*: The process starts from the *root* and uses *splitting rules* to divide the classes up until a stopping criterion is reached. The main aspects of this approach are the choice of splitting criteria, stopping rules, and the labelling of the terminal nodes (i.e. the assignment of a class label to each of the *leaves*).
- *hybrid*: A bottom-up procedure which directs and assists a top-down growing procedure.
- *tree growing-pruning*: The tree is grown to its maximum size and is selectively pruned afterwards.

In addition to the aforementioned approaches, there is also the approach of *growing a tree to its maximum size without pruning*. This is primarily used when applying Breiman's *Forest-RI algorithm* for *random forest* [18] (see section 3.3), which is also the approach chosen for this thesis.

3.2.3 Splitting rules

The second aspect of a decision tree algorithm concerns *splitting rules* (or *branching* [95]). According to Chiew [30], splitting rules are "essentially the heart of the transformation process from Data to Information" [30, p. 166]. He further elaborates that the splitting algorithm determines the shape of a decision tree, that the shape of a *decision tree* in turn determines the rules extracted from the data set, and that, eventually, the rules extracted from the data set determine the value of the information derived [30, p. 166].

In principle, splits are realised by the definition of cut points (*thresholds*) on a scale [115]. These cut points determine a left-hand or a right-hand split. According to Safavian [97], the aim of splitting an internal node in a decision tree is to make the data in its *son*-nodes "purer" [97, p. 664]. A node is said to be *pure*, if it contains observations from only one class [97]. To achieve *pure* nodes, an *impurity function* is defined for each internal node of a decision tree. By selecting feature variables as so-called *candidate splits*, the observations are split up into *left sons* and *right sons* of a node. More precisely, a proportion of the observations is

handed over to the *left son node* and the remainder of the observations is handed over to the *right son node* [97, p. 664]. According to Breiman [17], the best split is achieved by maximising the *goodness-of-split function*, as the goodness of a split can be defined as the *decrease in impurity* [97] (see Appendix B).

There are many different splitting rules (e.g. *Pearson's chi-squared* [77, 115], *Marshall correction* [77, 23], *mean square error node impurity* [53, 66], or *random attribute selection* [23]). Four of the probably most commonly used splitting criteria are:

- entropy-based information gain
- gain ratio
- gini impurity
- twoing

In very simple terms, all these splitting rules are an attempt to address *disorder* in a data set. *Entropy-based information gain*, for instance, as used with Quinlan's ID3 decision tree generation algorithm [95], is a splitting criterion which uses a metric based on Shannon's entropy concept[19] [112, 99, 17] and which seeks to find the best splitting point for the *root node* in a *binary tree* [95]. Quinlan, however, also reports about a bias with *entropy-based information gain*, in that features with more expressions (i.e. more different values) tend to be preferred over attributes with fewer expressions [95]. This led to the introduction of the *gain ratio*, a criterion which selects from among all attributes the one that has the highest *information value*.

Similar to entropy-based information gain, *gini impurity* (or the *gini index of diversity*) is a measure of disorder in a data set [42]. As the name suggests, it measures the impurity of an attribute with respect to the classes [77]. The best split is found with the attribute that maximises the expected decrease in terms of *gini impurity* [119] (which is equal to selecting the attribute with the lowest *gini index* as its splitting attribute). The information gain (or *increase in impurity*) is the total impurity of a data set lesser the weighted average of the *gini index* [77]. Lastly, while *twoing* is based on the *gini impurity* measure, it is a split criterion used solely for binary classification problems [17].

In overall regard of the *splitting rules* mentioned above, it is important to note that while the best *gini* splits try to create *pure* nodes, *twoing* and *entropy-based*

[19] *Entropy* can be understood as a *measure of disorder* [42] or as *uncertainty* [99]. According to Shannon, for instance, "[w]ith equally likely events there is more choice, or uncertainty, when there are more possible events." [99, p. 10]. In the context of information theory, *entropy* is a measure for the rate of information produced by a stochastic process (i.e. a mathematical model that produces a sequence of symbols governed by a set of probabilities) [99, p. 5].

information gain try to equalise the sample sizes in a node's right or left branch [17, p. 42]. Furthermore, each decision tree algorithm comes along with a certain set of *splitting rules*. Breiman's *CART* algorithm, for instance, uses either *gini impurity* or *twoing* [115, 17, 100], the former which was used in this thesis.

3.2.4 Tree pruning

A major issue in the context of *machine learning* is the problem of *model overfitting*. *Overfitting* arises from the way machine learning is realised. A model is trained with a subset of the total observation data (*training set*). The knowledge gained from the *training set* is then applied to make predictions on a new set of data (*test set*). Dietterich [32] describes the phenomenon of *overfitting* as follows: "The goal is to maximize [a model's] predictive accuracy on the new data points - not necessarily its accuracy on the training data. Indeed, if we work too hard to find the very best fit to the training data, there is a risk that we will fit the noise in the data by memorizing various peculiarities of the training data rather than finding a general predictive rule" [32, p. 1].

In terms of decision tree growing, the method used to tackle *model overfitting* is called *pruning*. There are two types of pruning, viz. *prepruning* and *postpruning*. *Prepruning* (or *stopping*) restricts the tree growing process in the learning phase [42]. *Prepruning* is a rule-driven tree growing restriction-based stopping criterion, which is used to find a balance between too complex, overfitted trees on the one hand, and too simple trees that smooth over important details on the other hand. This balance is also referred to as *bias-variance trade-off* [53, p.8]. There are several methods used for deciding where to stop the splitting procedure:

- the definition of the *node size* (i.e. a threshold for a *terminal node* to contain a certain percentage of all observations for a given class, e.g. 60% to 80%) [30, 7, 53, 82, 117, 66]
- the stepwise performance of *statistical tests* (e.g. chi-square, F-test, reduction in the sum of squared residuals) during the splitting process, and stopping the tree growing process as soon as a *branch* fails to meet the test criterion [115, 5]
- a threshold based on a *node's impurity* [7, 97]
- the *depth of a tree* (the number of split levels) [53]
- the *homogeneity* of class distribution in the *terminal nodes* [53]

An advantage of *prepruning* is that it is computationally cost saving, as the trees are not grown to their maximum size. A drawback of *prepruning*, however,

is that not all of the data information may be accounted for and that useful information may be overlooked by stopping the tree-growing process at a too early stage. Hence, this may result in detrimental knowledge generation [30, p. 169f]. Therefore, it is generally recommended to grow a maximum tree by continuing the splitting process until all *terminal nodes* are *pure* (i.e. that each terminal node contains observations of one single class) or nearly pure, and to selectively *prune* the resulting large tree afterwards to achieve *optimum cost* trees [97, 30, 115, 118].

Postpruning (or *pruning*) is done in a separate step after an unstopped tree has been generated [42], i.e. after a tree has been grown to its maximum size. In a subsequent step, the least reliable *branches* are identified and removed. Although the number of classification errors on the *training set* increases after *pruning*, the error rate on the *test set* decreases [76]. This can be measured by calculating the *misclassification error* [10, p. 39f]. Misclassification happens, when an object of class C_1 is erroneously assigned to class C_2 and vice versa. The probability of such a misclassification can be mathematically calculated and, in turn, the error detected and reduced. An equally important measure in the context of *postpruning* is the so-called *cost-complexity measure* [97, p. 665].[20] The complexity of a tree is defined as the number of *terminal nodes* in a tree. The *cost-complexity measure* is a calculation which takes into account the estimated misclassification rate and the complexity cost of the tree. In turn, the objective of *pruning* is to find the *subtree* which minimises the *cost-complexity measure*.

According to Wu and Kumar [118], a well-known example of such *cost-complexity pruning* (or *error-complexity pruning* [77, 23]) is Breiman's CART algorithm. Based on the *cost-complexity measure*, this *pruning* algorithm already determines whether or not a *subtree* is replaced by a *leaf*-node, which means that *tree pruning* in this case is not necessary. The reason for this lies in the fact that *random forest* is an *ensemble method*, which Breiman explains by stating that the "[u]se of the Strong Law of Large Numbers shows that [trees] always converge so that overfitting is not a problem" [18, p. 6]. The *number of trees*, however, has to be sufficiently high (see section 3.3.3).

3.3 Random Forest

This section introduces the *machine learning algorithm* known as *random forest*. *Random forest* is a concept which was introduced in 2001 by Leo Breiman in his seminal paper *Random Forests* [18]. Svetnik et al. [105], for instance, describe random forests as an *ensemble of unpruned regression or classification trees*, created

[20] Besides *cost-complexity pruning*, there is also a number of other *pruning* methods, such as *critical-value pruning* [76], *minimum-error pruning* [76], *reduced-error pruning* [76, 118], and *pessimistic error pruning* [76, 33, 118].

by using bootstrap samples from training data and random feature selection in tree induction. What is essential is that, in contrast to *decision trees* in which relationships are perfectly identifiable, this is not the case with a *random forest*. In other words, a *random forest* is a *black box* [18, 105]. As an *ensemble* method, *random forest* is impenetrable in terms of simple interpretations of internal mechanisms [18].

In the following, this section first introduces Breiman's *Forest-RI* algorithm. It then continues with the assessment of *variable importance* and closes with an overview of *(hyper-)parameters* and their *tuning* abilities.

3.3.1 Forest-RI

As defined by Breiman [18], *random forests* are "a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest".[21] Breiman also developed the so-called *Forest-RI algorithm* for *random forest*. Forest-RI describes a process for creating a *random forest* and essentially consists of three steps:[22]

- sampling
- tree growing
- prediction

Sampling means that a number of n_{tree} *bootstrap samples* are drawn from the original training data [67, p. 18]. *Bootstrapping* refers to repeatedly drawing random samples (with replacement [105, 8]) from the same *training set* (see section 3.4.1).

Tree growing, in order to create a *forest* (an *ensemble of trees*), is done by running the so-called *classification and regression trees algorithm*, better known as the *CART* algorithm [19]. As the name suggests, CART is an algorithm which can be used for both *classification* and *regression problems*. The CART algorithm has particular choices for *tree growing*, *splitting*, and *pruning*. For instance, with CART, *trees* are first grown to their maximum size, without the use of stopping rules (see section 3.2.4), and are only stopped when no further splits are possible (as no more data is available) [118]. As already previously mentioned, the splitting rules used by CART for tree growing are either *gini impurity* or *twoing* [115, 17,

[21] In more technical terms, Breiman [18] defines a random forest as "a classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta_k), k = 1, ...\}$ where Θ_k are independent identically distributed random vectors, and each tree casts a unit vote for the most popular class at input x." [18, p. 6]. In this context, Θ_k refers to a number k of *bootstrapping samples*.

[22] For a detailed overview see Liaw and Wiener [67, p. 18].

100]. More precisely, at each node, rather than choosing the best split among all predictors, a random sample m_{try} of the available predictors p (i.e. features) is drawn, and the best split from among those variables is chosen [67, p. 18]. In the end, fully grown trees are *pruned* by applying the *cost-complexity* algorithm [118].

The last step of the generic *Forest-RI algorithm* concerns the *prediction* of new, unfamiliar data (i.e. data which was not used for growing and tuning the trees). By aggregating the predictions of the n_{tree} trees (either by aggregation via averaging for a *regression problem*, or by majority vote for a *classification problem*), new data is predicted [67, p. 18]. In order to increase prediction accuracy, several training sets are generated from the training sample by the use of *bagging* [16] (see section 3.4.1). *Prediction quality* is measured in terms of the *generalisation error* [18] [9].

Essentially, what makes the random forest *random* is a combination of *bagging* and a *random selection of feature subsets* [8, 84, 2]. These aspects are described in more detail in sections 3.4.1 and 3.3.3, respectively.

3.3.2 Variable importance

As previously mentioned, in a *random forest*, the relationships between variables are hidden, which is the reason why a random forest is considered a *black box* [18, 105]. In his original publication, Breiman [18] introduced measuring *variable importance* (or *feature importance*) as a means for tackling this shortcoming. There are two approaches for measuring variable importance [18]:

- (permutation) variable importance (VI)
- gini importance

(Permutation) variable importance (VI) is a simple but effective method for determining a variable's contribution to a model's prediction accuracy [18]. After growing the trees, the values of the m^{th} variable in the *OOB samples* (the samples which are not part of the tree growing process, see section 3.4.1) are *randomly permuted* and are "dropped down" the corresponding tree m times. As a last step, the plurality class vote with the *noised up variable* m (meaning that m is randomly permuted) is compared to the true class label in order to receive the misclassification rate [18]. The more the classification error increases after permutation, the higher a variable's importance [81, p. 140], and vice versa. Nau et al. [81], for instance, describe the application of *permutation* as *mimicking the absence of this variable* [81, p. 140].

Gini importance is a more sophisticated measure. It is split-based and measures the level of *impurity* (see section 3.2.3) of the samples assigned to a node, based on a split at the parent node. Every time a feature is used to split a node, the *gini value* for the two descendant nodes is smaller than the parent's value. The sum of

the *gini values* decreases for each feature over all trees, from parents to children (or *fathers* to *sons* [1]), and provides a simple estimate of *feature importance*. In general, the smaller the *gini value*, the *purer* the respective node [94, p. 497].

In many applications, *variable importance* and *gini importance* are sensible means for determining important variables. However, in cases in which predictor variables show high differences in the number of categories or strongly vary in scale, these measures are not reliable [104, p. 2], as both measures are biased in favour of variables with more categories [104, 3, 93].

In their efforts to tackle this issue, Altmann et al. [3] introduced *permutation importance (PIMP)*. PIMP is a heuristic for correcting biased measures of feature importance, such as *gini importance*, and is a "method [which] normalizes the biased measure based on a permutation test and returns significant P-values for each feature" [3, p. 1341] (see Appendix B). Furthermore, according to Altmann et al. [3], the advantages of PIMP are that the dependence between predictor variables stays the same, that the number of permutations can be much smaller than the number of predictor variables, and that PIMP can be used together with any method that generates (biased or unbiased) measures for feature importance [3, p. 1342]. Unfortunately, due to computational restrictions, PIMP was not an option for this thesis, which is why *(permutation) variable importance* was selected for measuring *variable importance*.

3.3.3 (Hyper-)parameters

There are several *(hyper-)parameters* that, on the one hand, determine a *forest*'s randomness and, on the other hand, are crucial in terms of finding an optimum between low correlation and reasonable strength of the trees within a *forest* [93]. The process of influencing a model's parameters (in order to find this optimum) is referred to as *tuning* [93]. For some parameters, recommendations for suitable tuning have been put forward (e.g. [93]). For others, however, scholars have (had) to make their own (best) decisions, taking into account the peculiarities of their respective data set. The examples of *(hyper-)parameters* listed below, and as put forward by Probst, Wright, and Boulesteix, are introduced in the following, whereas special attention is given to their *tuning possibilities* [93, p. 3ff]:

- number of randomly drawn candidate variables (or *mtry*)
- sample size and replacement
- node size
- number of trees

- splitting rule

Mtry. According to Probst et al., *mtry* (or m_{try}) is one of the central (hyper-) parameters in *random forest* [93]. *Mtry* defines the number of randomly drawn candidate variables for each split. That is, each new split starts a new random selection of *mtry* candidates from the data set. If *mtry* is small, the trees grown are more diverse, less correlated and more *stable*. *Instability*, in this context, means that small variations in the *training set* can lead to different trees and different predictions for the same validation examples [65, p. 145].

In forests, a low *mtry* leads to a better exploitation of variables, with only a moderate effect on the *target*, which otherwise would be masked by splitting candidates with a strong effect on the *target*. Forests created with a low *mtry*, however, on average tend to show a lower performance, as they are based on suboptimal variables selected from a small set of randomly drawn candidates. On the other hand, a large *mtry* increases the likelihood of at least one strong variable being part of the set of splitting candidates. In short, *mtry* unveils a trade-off between stability and accuracy of the individual trees.

For *mtry* there are different default values, which depend on whether the problem at hand is one of *classification* or *regression*. For *classification problems*, the default value for *mtry* is $\sqrt{p}$ (regarding *error rate*), for *regression problems*, the default value for *mtry* is $\frac{p}{3}$ (regarding *mean squared error*). For both, p is the number of predictor variables [93]. Although these default settings have proven reasonable for many random forest applications, they can in some cases be improved by *tuning* [93].

Sample size and replacement. The number of samples chosen for training the trees (i.e. the *sample size*) has an effect on the diversity, correlation, and stability of the trees. A small number of training observations leads to more diverse, and therefore less correlated trees, with an individual tree having a lower prediction accuracy [93]. On the other hand, low correlation between trees increases prediction accuracy after forest *aggregation* (see section 3.4.1). As is the case with *mtry*, the *sample size* unveils a trade-off between stability and accuracy.

Sample size also has an effect on computational performance, in that the smaller the *sample size*, the shorter the runtime [93]. When comparing *sample drawing without replacement* with *sample drawing with replacement* (see section 3.4.1), there is no substantial difference in performance. However, in the case of varying numbers of category expressions among categorical variables, *sample drawing with replacement* may lead to a decrease in performance, which can be due to a slight variable selection bias [93]. The reason for this lies in the fact that *bootstrap sampling* artificially induces an association between the variables. Although this effect is always present when statistical inference is corrected out

of *bootstrap samples*, it is of special interest in the case of variables with more categories. Samples, even if they are drawn from independent distributions, always show minor variations from the *hypothesis of independence*, and these minor variations are aggravated by *bootstrap sampling* [104, p. 18].

The common default value for the parameter *sample replacement* is TRUE [93]. Further options for tuning are limited, as the only possibility is to choose between *sample drawing with replacement* and *sample drawing without replacement.*

Node size. This parameter specifies the minimum number of observations in a terminal node [93]. The balancing-out of this number has an effect on the shape of the trees, in that if the *node size* is small, trees are grown with greater depth and they contain more splits.

The common default values for node size are "10" for *probability forests*, "5" for *regression forests*, and "1" for *classification forests* [116]. Although common default values in general provide good results, further tuning can lead to even better results. If the number of noise variables is increased, the optimal *node size* increases. According to Probst et al., for large data sets, it may be helpful to set the *node size* higher than the default setting, as runtime substantially decreases on higher values (but appears to have little influence on performance) [93, p. 3].

Number of trees. The number of trees in a forest is not a typical tuning parameter, but it has an effect on the model's *prediction quality.* As the *generalisation error* decreases with the increase in the *number of trees*, this parameter has to be sufficiently high. This is because the *generalisation error* of a *random forest* model almost surely convergences to a limit with the increasing *number of trees* [18, p. 5].

As the optimal *number of trees* depends on the data set used, there is no general default value. However, it is not only the *number of trees* which influences the convergence rate, but also a lower *sample size*, a larger *node size*, and a smaller *mtry*. What can be said, however, is that the biggest gain in performance is achieved within the first 100 trees, which suggests that it is a good idea to grow a large number of trees (e.g. Probst, Wright and Boulesteix recommended 500 or 1,000 trees [93]). On the other hand, it should be kept in mind that the *number of trees* linearly affects computation time [93].

Splitting rule. The *splitting rule* is also not a typical tuning parameter. Breiman's [18] original random forest for *classification problems* used *gini impurity* to select the split that minimises gini impurity (see section 3.2.3) and *weighted variance* for *regression problems.* The problem with this selection, however, is that variables with many possible splits are favoured over variables with a small

number of possible splits (e.g. *continuous variables* or *categorical variables* with many expressions) [93, p. 4].

Probst, Wright, and Boulesteix [93] recommend an alternative, viz. *p-value approximations* (as used with *R*'s *ranger* package). This alternative *splitting rule*, however, is not available for *random forests* addressing *classification problems*, which is why the *splitting rule* applied in this thesis was *gini impurity*.

3.4 Model evaluation

Model evaluation is an essential part of each *machine learning* process. It is an iterative process which begins after the initial tree has been built, is repeated in the course of *model tuning*, and ends with the *final evaluation* of the model. The purpose of *model evaluation* is to identify the *prediction error* of a classifier.

Model evaluation is not always straightforward, but depends on the *evaluation strategy* and the *evaluation measure* chosen. *Evaluation strategies* determine the framework for model evaluation, common examples being *bootstrap sampling* and *k-fold cross-validation*. *Evaluation measures* (or *performance assessment measures*), on the other hand, are the actual application of the strategy and allow for comparing classifier performance. Evaluation measures can either be *metric* (e.g. *accuracy*, *error rate*, *true positive rate*) or a *graphical representation* (e.g. *ROC curve*).

In the following, this section first introduces two common *evaluation strategies*. It then moves on to present different *evaluation measures*.

3.4.1 Evaluation strategies

While *evaluation strategies* can differ in some respects, a general process of model evaluation starts with the evaluation of the initial model, which is followed by an alternating tuning and evaluation process. This process is repeated as long as a gain in performance can be achieved (i.e. as long as a trade-off between tuning effort and performance gain is feasible). The process is terminated with the final model evaluation. The two following strategies are both used for *random forest*:

- bootstrap sampling (or bootstrapping)
- k-fold cross-validation

Bootstrap sampling. *Bootstrapping* refers to repeatedly drawing random samples of independent observations from the same training set *with replacement* [16, 105, 8], meaning that a certain observation can be drawn from this training set more than once [42]. Furthermore, each *bootstrap sample* grows into a separate tree

and all observations which are not chosen for this tree are used for *validation* [105, p. 1948]. These samples are referred to as *out-of-bag (OOB) samples* (see section 3.4.2). If there is only one training set, several training sets can be "imitated" via repeated *bootstrap sampling*. Several bootstrap learning sets emerge from this procedure, from which the set predictors are created.

An advancement of bootstrap sampling is *bagging*. *Bagging* was introduced by Breiman [16] in order to increase a classifier's prediction accuracy. The term "bagging" is a portmanteau term of *bootstrap aggregating* [16], whereby *aggregation*, in this context, means that all classifiers from a set of decision tree classifiers are aggregated, producing *one aggregated predictor* which is better than one single classifier [16, p. 1] (see Appendix B).

K-fold cross-validation. In order to conduct *k-fold cross-validation*, the original data set is divided into groups of observations (or *folds*) of approximately equal size, whereas the first *fold* is treated as a *validation set* (i.e. the set that is *held out*), while the remainder constitutes the *training set* [52] (see section 3.1.2). The prediction error is estimated on the held-out *folds*.[23] Figure 8 shows two examples of *k-fold cross-validation*:

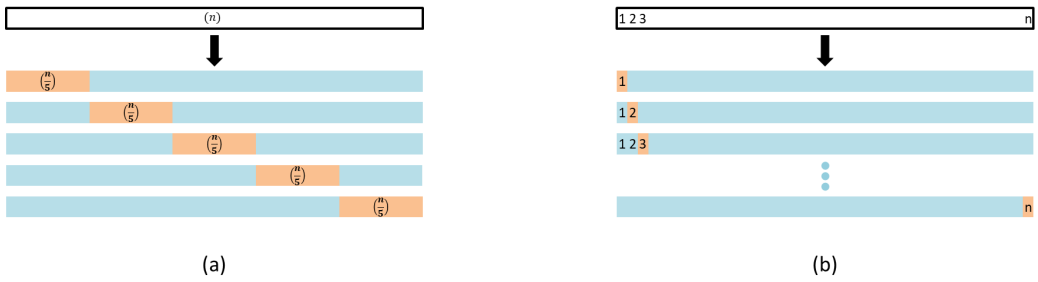

Figure 8: K-fold cross-validation. Illustration (a) shows an example of *5-fold cross-validation*, and illustration (b) shows the special case of *leave-one-out cross-validation* (LOOCV). The white bar represents the total data set, the blue bars are the *training sets*, and the orange elements are the validation *folds* of equal size ($\frac{n}{5}$). Each of the orange folds is used as a *validation set* and is *held-out* from the tree growing process, while the remainder (blue) is used to train the model. The test error is calculated by averaging the k resulting error rates. (Adapted from [52])

In figure 8, illustration (a) shows the original data set with n observations, randomly separated into 5 non-overlapping *folds* of approximately equal size ($\frac{n}{5}$). The white bar above represents the original sample with observations $1, 2, 3, ..., n$. The blue bars with orange elements illustrate the sampling process. Blue elements

[23]For instance, if $k = 10$ (*ten-fold cross-validation*), this means that the process of model training is repeated ten times. In order to calculate the total error, the average of the errors of each of the ten validation sets is calculated.

represent training observations, orange elements are (*held-out*) validation observations. Each of the orange fifths is used as a *validation set* and is *held-out* from the tree growing process, while the remainder (blue) is used to train the model. The test error is calculated by averaging the five resulting error rates [52]. Illustration (b) depicts a special case of *k-fold cross-validation* in which the number of splits k is set to the number of observations n. This special case is called *leave-one-out cross-validation (LOOCV)* [52].

In this thesis, the *evaluation strategy* employed was *bootstrap sampling*. All in all, this strategy appeared easier to implement, in particular given that different self-defined *training set sizes* were used.

3.4.2 Evaluation measures

Evaluation measures are used to assess the prediction quality of a model, which is done by comparing the classifications of *true* and *predicted values*. The type of *evaluation measure* chosen (from among the many available options) depends on the type of problem addressed. For *classification problems*, examples are *error rate*, *geometric mean*, and *F-measure* [109]. For *regression problems*, examples are *mean squared error* [16, 18, 66], *root mean squared error* [27, 121], R^2*-score* [121], and the *index of agreement (IOA)* measure [121].

Evaluation measures can roughly be divided into *metric* and *graphical measures*, meaning that they either output absolute (e.g. *Naive Bayes*) or relative scores (e.g. *area under the ROC curve (AUC)* [35]), or diagrams (e.g. *receiver operating characteristics (ROC) curve*, *precision-recall (PR) curve*, *detection error trade-off curve (DET)* [109, 106]). While there is an abundance of evaluation measures, the intention in this section is to provide an overview of metric measures and a common example for a graphical measure:

- metric measure: measures derived from a *confusion matrix*
- graphical measure: the *ROC curve*

Confusion matrix. A *confusion matrix* is a matrix which is represented by a contingency table and which displays, for a set of labelled examples, the differences between *true* and *predicted classes* [15, p. 1145]. Figure 9 depicts two confusion matrices:

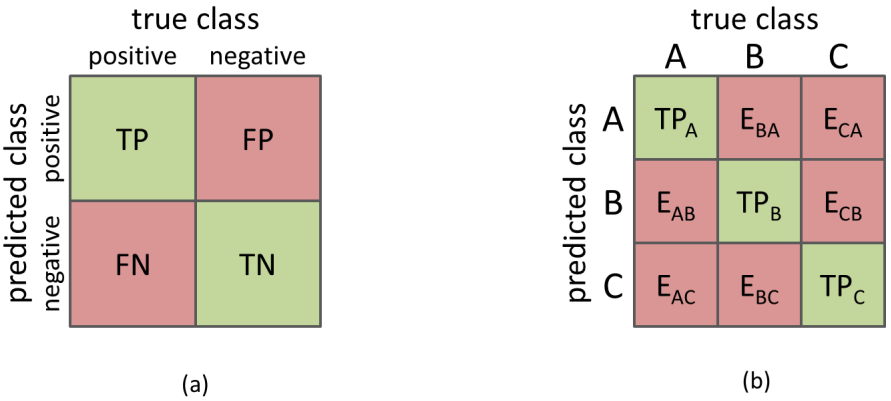

Figure 9: *Confusion matrix.* Illustration (a) is a *confusion matrix* for binary classification (*2 x 2*) and illustration (b) is a confusion matrix for multi-class classification (*3 x 3*). Confusion matrices show the correct or erroneous classification of samples in a contingency table with two types of classes, viz. *true* and *predicted*. (Adapted from [109])

In figure 9, confusion matrix (a) represents an example for *binary classification*. It contains *2 x 2* elements and has the class labels *positive* and *negative* (or *true/false*, *1/0*). The *true classes* are displayed on the x-axis, the *predicted classes* are displayed on the y-axis. This matrix allows for the differentiation of four classifications:

- *true positive (TP)*: sample and classification are positive
- *false positive (FP)*: sample is negative, classification is positive
- *false negative (FN)*: sample is positive, classification is negative
- *true negative (TN)*: sample and classification are negative

Illustration (b) represents an example of a *multi-class classification problem* with three classes A, B, and C, which produce three possible *true positive* classifications, viz. TP_A, TP_B, and TP_C (one for each class).[24] All other classifications are such that one class is correctly classified, but the other is not. These elements are generically represented as E_{XY}, with the subscript XY being samples from class X erroneously classified as class Y. In this confusion matrix, the classification for a class is the row-wise sum of errors.[25]

[24] *True negative*, which is a possibility in *binary classification problems*, is not applicable to *multi-class problems* [109].

[25] In general, an m *x* m confusion matrix contains m correct classifications, and $m^2 - m$ errors [109, p. 2]. In matrix (b), for instance, *false negative* for class A (FN_A) is the sum of all erroneously classified samples of A (i.e. $FN_A = E_{AB} + E_{AC}$). On the other hand, *false positives* with respect to class A (FP_A) are all samples from class B and C incorrectly classified as A (i.e. $FP_A = E_{BA} + E_{CA}$).

Many different scalar evaluation measures (or *performance metrics*) can be derived from a *confusion matrix*. Table 1 provides an overview and definitions of these scalar metrics [109]. The table also indicates whether or not a certain metric is *sensitive* or *insensitive to imbalanced class distribution*. Whether or not a measure is sensitive depends (with few exceptions, e.g. *geometric mean*, *Youden's Index*) on whether or not a measure uses elements of both columns of a *confusion matrix* [109]. If a measure uses elements of both columns, these metrics will, in the case of changes in data distribution, change as well (even if the classifier performance remains unchanged). In the case of *binary classification*, the ratio between the number of positive (P) and negative (N) class samples ($\frac{P}{N}$) determines the *sensitivity to imbalanced class distribution*.

One of the most commonly applied performance metrics for classification problems is *(classification) error rate* [93]. It is also the foundation for the evaluation measure *out-of-bag (OOB) error estimate* (see Appendix B), which is used with *random forest*. *Out-of-bag* is a term used in combination with *bootstrap sampling* (see section 3.4.1). The purpose of OOB is to estimate a model's *generalisation error* [18]. As previously mentioned in section 3.4.1, *bootstrap sampling* means that only some observations from the *training set* are used to grow a tree. The unused remainder of samples constitutes the OOB validation sample. For each tree, the data that was not used in the growing phase (OOB-data) is "dropped down" the tree and, on its way down, is subjected to a majority vote conducted across all trees, which eventually results in the *out-of-bag classifier*. According to Breiman [18], the *out-of-bag* estimate for the *generalisation error* is the *error rate* of the *out-of-bag classifiers* on the *training set* [18, p. 11]. The *error rate* decreases when the number of input combination increases, which leads to the OOB estimate tending to overestimate the current *error rate* [18, p. 11]. However, if the number of trees is sufficiently high, the OOB estimation of the *error rate* is relatively accurate [67] or even unbiased [18]. A practical aspect of the *OOB-error* is that it is automatically calculated whenever a model is generated.

Further commonly used performance metrics are *sensitivity* (or *true positive rate, TPR*) and *specificity* (or *true negative rate, TNR*) [15, 52, 109, 110]. While *sensitivity* represents the share of correctly classified positives on all observations, *specificity* is the share of correctly classified negatives [109].[26] According to Tharwat, "the main goal of all classifiers is to improve the sensitivity, without sacrificing the specificity" [109, p. 5]. In the case of imbalanced data (i.e. uneven class distribution), however, the aims of the two metrics may contradict one another. One way to tackle this conflicting behaviour of *sensitivity* and *specificity* is to apply

[26] Given a *classification problem* with two classes P and N, *positives* are observations that are assigned to a class P (1 or *TRUE*), while *negatives* are observations that are assigned to a class N (0 or *FALSE*).

measure	definition	imbalanced data
accuracy (ACC)	$ACC = \frac{TP+TN}{TP+TN+FP+FN}$	sensitive
error rate (ERR) (misclassification rate)	$ERR = \frac{FP+FN}{TP+TN+FP+FN} = 1 - Acc$	sensitive
sensitivity (true positive rate (TPR), hit rate, recall) *[P is the number of positive class samples]*	$TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$	**insensitive**
specificity (true negative rate (TNR), inverse recall) *[N is the number of negative class samples]*	$TNR = \frac{TN}{FP+TN} = \frac{TN}{N}$	**insensitive**
false positive rate (FPR) (false alarm rate (FAR), fallout)	$FPR = \frac{FP}{FP+TN} = \frac{FP}{N} = 1 - TNR$	**insensitive**
false negative rate (FNR) (miss rate)	$FNR = \frac{FN}{FN+TP} = \frac{FN}{P} = 1 - TPR$	insensitive
positive prediction value (PPV) (precision)	$PPV = \frac{TP}{FP+TP} = 1 - FDR$	sensitive
false discovery rate (FDR)	$FDR = 1 - \frac{TP}{FP+TP} = 1 - PPV$	sensitive
negative predictive value (NPV) (inverse precision, true negative accuracy (TNA))	$NPV = \frac{TN}{FN+TN} = 1 - FOR$	sensitive
false omission rate (FOR)	$FOR = 1 - \frac{TN}{FN+TN} = 1 - NPV$	sensitive
positive likelihood ($LR+$)	$LR+ = \frac{TPR}{1-TNR} = \frac{TPR}{FPR}$	insensitive
negative likelihood ($LR-$)	$LR- = \frac{1-TPR}{TNR}$	insensitive
diagnostic odds ratio (DOR)	$DOR = \frac{TPR}{1-TNR} * \frac{TNR}{1-TPR} = \frac{TP*TN}{FP*FN} = \frac{LR+}{LR-}$	insensitive
Youden's index (YI) (Bookmaker Informedness (BM)	$YI = TPR + TNR - 1$	insensitive
Matthews correlation coefficient (MCC)	$MCC = \frac{\frac{TP}{N} - TPR*PPV}{\sqrt{PPV*TPR(1-TPR)(1-PPV)}}$	sensitive
discriminant power (DP)	$DP = \frac{\sqrt{3}}{\pi}(\log(\frac{TPR}{1-TNR}) + log(\frac{TNR}{1-TPR}))$	insensitive
F-measure (F_1-score)	$F\text{-}measure = \frac{2PPV*TPR}{PPV+TPR} = \frac{2TP}{2TP+FP+FN}$	sensitive
F_α-measure *[if negative class samples are increased by α times]*	$F_\alpha\text{-}measure = \frac{2TP}{2TP+\alpha FP+\alpha FN}$	sensitive
F_β-measure *[β is to apply weight either to precision ($0 < \beta < 1$), or to recall ($1 < \beta < +^\infty$) [36]]*	$F_\beta\text{-}measure = (1+\beta^2)\frac{PPV*TPR}{\beta^2 PPV+TPR}$	sensitive
adjusted F-measure (AGF) *[F_2 is the F-measure with $\beta = 2$, and $InvF_{0.5}$ is calculated by building a new confusion with inverted class labels]*	$AGF = \sqrt{F_2 * InvF_{0.5}}$	sensitive
markedness (MK)	$MK = PPV + NPV - 1$	sensitive
balanced classification rate (BCR) (balanced accuracy)	$BCR = \frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$	insensitive
balance error rate (BER) (half total error (HTER))	$BER = 1 - BCR$	insensitive
geometric mean (GM)	$GM = \sqrt{TPR*TNR} = \sqrt{\frac{TP}{TP+FN} * \frac{TN}{TN+FP}}$	**insensitive**
adjusted geometric mean (AGM) *[negative class samples are increased by α times]*	$AGM = \frac{GM+TNR(\alpha FP+\alpha TN)}{1+\alpha FP+\alpha TN}$	sensitive
optimization precision (OP)	$OP = Acc - \frac{\lvert TPR-TNR\rvert}{TPR+TNR}$	sensitive
Jaccard metric (Tanimoto similarity coefficient)	$Jaccard = \frac{TP}{TP+FP+FN}$	sensitive

Table 1: Performance metrics based on the *confusion matrix*. Metrics are built from *true positive (TP)*, *false positive (FP)*, *false negative (FN)*, and *true negative (TN)* classifications. P and N are the numbers of *positive* and *negative* class samples, α and β are weights [109, p. 3ff].

the *geometric mean (GM)*. The *geometric mean* aggregates the two metrics into a one-dimensional performance measure which can then be used for assessing a model's prediction quality [109, 110].

Receiver operating characteristics (ROC) curve. The *ROC curve* is a two-dimensional graph which puts the *true positive rate (TPR)* in relation to the *false positive rate (FPR)* [94, 42, 5, 109] (see table 1). Figure 10 shows a schematic ROC curve (a) and an exemplary ROC curve (b), with the x-axis representing the FPR and the y-axis representing the TPR:

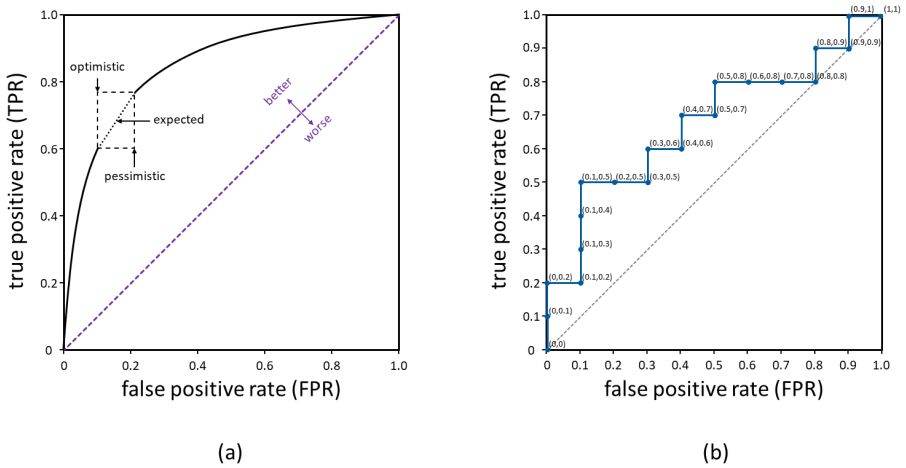

Figure 10: Schematic representation (a) and exemplary ROC curve (b). The *receiver operations characteristics (ROC) curve* depicts the relationship between *true positive rate (TPR)* and *false positive rate (FPR)*. Whithin this *ROC space*, the diagonal line represents the border for *random guessing*. In (a), the continuous line represents the ROC curve, with stylised expected, pessimistic, and optimistic classifier performance. In (b), the step-like line is the ROC curve. Each point on the ROC curve can be calculated via elements of the *confusion matrix* and represents a decision tree. The calculated coordinates in (brackets) represent the points of the form (FPR, TPR). (Adapted from [109, 35])

The space confined by the x- and y-axes is called the *ROC space*. In general, one point within the *ROC space* is better than another, if its position lies to the northwest of the first point [35, p. 862]. Furthermore, in both illustration (a) and (b), the dashed diagonal line indicates the threshold for *random guessing* [42], meaning that every classifier above that line performs better than a random guess. The corner points of the ROC space hold important information [109]:

- *lower left corner*: no positive samples are correctly classified, all negative samples are correctly classified ($TPR = 0$, $FPR = 0$)

- *upper right corner*: all positive samples are correctly classified, no negative samples are correctly classified ($TPR = 1$, $FPR = 1$)
- *upper left corner*: all positive and negative samples are correctly classified ($TPR = 1$, $FPR = 0$), thus representing *perfect classification*
- *lower right corner*: no positive and negative samples are correctly classified ($TPR = 0$, $FPR = 1$)

In illustration (a), the continuous line indicates the ROC curve. The dotted line inside the rectangle (on the ROC curve) represents expected sample positions (or *ROC segments*), with the upper left corner indicating optimistic, and the lower right corner indicating pessimistic *ROC segments* [109]. Illustration (b) shows an exemplary ROC curve for a test set with 20 samples. For this *classification problem*, each of these classes (i.e. *positive* and *negative*) consists of 10 samples.

The actual creation of a ROC curve depends on the classifier used. *Ranking* or *scoring classifiers* (i.e. classification methods that naturally yield *instance probabilities* or *scores*, such as *Naive Bayes classifiers* or *neural networks*) use a threshold for drawing the ROC curve [35].[27] *Decision trees* and *random forests*, however, use the *confusion matrix*. When applied to a *test set*, each decision tree in a *random forest* creates a *confusion matrix* which generates one point in the *ROC space*. In other words, in illustration (b), each point on the ROC curve represents a decision tree [35]. The derivation of a point on the ROC curve from the *confusion matrix* is accomplished by calculating the TPR and the FPR (see table 1).[28]

Fawcett [35] draws attention to an attractive property of a ROC curve, viz. the *insensitivity* to changes in class distribution when only one column of the *confusion matrix* is used. In other words, as ROC graphs are based on the TPR and the FPR, each of these two dimensions is a strict columnar ratio and, therefore, does not depend on class distribution [35, p. 864].

There is another interesting feature which can be applied to ROC curve problems, viz. the *area under the ROC curve (AUC)*, which is a means to compare ROC curves via one-dimensional scalar values. According to Norton and Uryasev, the "AUC does not give a direct measure of a classifier's ability to properly classify a single randomly chosen sample, but instead is concerned with a classifiers ability to properly rank two samples that are presumed to be in different classes" [83, p. 3]. An important property of the AUC is that it is equivalent to the probability

[27] *Instance probability*, in this context, is the probability of an observation being a member of a certain class. In illustration (b), this would mean that each point on the ROC curve represents a single observation.

[28] In figure 10, the numbers in brackets next to the points constitute the calculated coordinates of the form (X, Y) or (FPR, TPR).

of a classifier which is able to rank a randomly chosen positive instance higher than a randomly chosen negative instance.

Overall, three *performance measures* were used in this thesis for evaluating the models' prediction quality. For one, this was the *OOB-error* (for all the models), in particular as this error is automatically calculate whenever a model is generated. On top of that, the performance of *classification forests* was evaluated via *geometric mean (GM)*, and the performance of *probability forests* was tested via the *area under the ROC curve (AUC)*.

4 Data

Generally speaking, data used for building a *random forest* model does not have to meet specific criteria [69], which tends to make such analyses a rather straightforward matter. In the case of this thesis, however, *data collection*, *preparation*, and *analysis* required a lot of attention. The main reasons for this were the complex AMS database from which the data was retrieved, the observance of and compliance with data protection and security regulations[29], and hardware and software restrictions (see Appendix E).

In the following, this chapter first provides definitions of the *population*, *sample*, and *target variable* as determined for this study. Subsequently, it structurally follows the process of *supervised machine learning* as described in section 3.1.2. More precisely, it presents the steps taken for the stages *data collection*, *data preparation*, and *data analysis*, and the actual processes which were developed in order to deal with the constrictions mentioned above.

4.1 Population, sample, and target variable

As previously mentioned, the purpose of this thesis is to provide a measure of the probability of (re-)integration into the labour market of people registered with the AMS. Thus, the *population* was defined as registered AMS clients. In turn, the *sample* drawn from this population was defined as *all people registered with the AMS for at least one day within the year 2017*.

The *target variable*, i.e. the variable for determining (re-)integration and which all data was tested against, was determined as *sustainable employment*, whereas an employment was considered *sustainable* if it lasted longer than two months [56].[30] The notion of *employment*, in this context of sustainable employment, referred to two *Uni-status* forms of employment, viz. *employment relationships* and *self-employment* (see section 2.3)[31], whereas it was possible to have one single

[29] The sensitivity of the data did not allow for loading the data into an environment other than that of the AMS, which essentially excluded all cloud services for big data applications (e.g. Amazon Web Services (AWS), Microsoft Azure, or IBM Cloud).

[30] The AMS-internal definition specifies *sustainable employment* ("nachhaltige Arbeitsaufnahme") as an employment relationship which lasts longer than two months ($\geq$ 62 days). However, within a year, there are two combinations of months (viz. December and January, July and August) for which the condition "longer than two months" cannot be satisfied with a duration of 62 days. As a consequence, and in order to fully satisfy this condition, for this thesis *sustainable employment* was understood to be satisfied after at least 63 days (i.e. $\geq$ 63 days) of uninterrupted employment.

[31] *Marginal employments* are never considered an *employment*, as they are not subject to complete statutory insurance, i.e. statutory health, accident, retirement, and unemployment insurance (see appendix A, table 3).

employment or several consecutive employments, as long as these were seamlessly connected (i.e. without interruptions greater than 0 days).

4.2 Data collection

The sample was collected from the AMS database. This database consists of a number of different tables which contain variables suitable for estimating *sustainable employment*.

In the following, this section first describes the decisions made for selecting the appropriate tables and variables from the AMS database. This is followed by a section describing the necessary preparatory steps required before being able to download the data.

4.2.1 Table and variable selection

The AMS database features different tables, which are created by bundling data from different applications. These so-called *production tables* are classified into different *table types*:

- *basic tables (BAS)* contain unprocessed information derived from source applications
- *business tables (BUS)* contain information for a specific report date
- *intermediary tables (INT)* are used to transform BAS-tables into BUS-tables
- *temporary tables (TEMP)* are created for temporary use
- *dimension tables (DIM)* contain pre-defined dimensions for variables

These *table types* feature as abbreviations in the table names. The table names also reveal the tables' *functional areas*, thereby hinting towards the kind of data to be expected in the table. For instance, according to its label, the table `MON_UNI_STATUS_INT` is an **int**ermediary table which contains **mon**itoring information concerning a client's **Uni-status**.

The overall criterion for table selection was that the final selection had to be small enough to be processible, but large enough to contain a sufficient amount of information for the model to be reliable. The starting point for the table selection process was to gather the names of 1,370 active *production tables*. While the initial idea was to select the tables according to their *type* and *function*, it turned out that table names did not always reflect the tables' contents. For instance, according to the table name, the table `MON_UNI_STATUS_INT` is an intermediary table, yet going by the data it contained, it should actually have been classified as

a BAS- or BUS-table.[32] As several *criteria-led* table selection attempts failed, the only possibility to select a number of reliable *production tables* was to fall back on *experience-based* table selection. This meant drawing on the knowledge of one of the two AMS database expert users, who was able to point out those tables with most informational value for this thesis.

The result of the *experience-based* table selection process was a set of eight different tables. These tables covered five different content areas, viz. *client basic information* (2 tables), *occupational career monitoring* (2 tables), *unemployment insurance and livelihood security* (2 tables), *labour market funding and support* (1 table), and *client business case data* (1 table). These tables contained 564 variables, a number which, however, was too large for the available hardware to process (see Appendix E). Consequently, the number of variables had to be reduced. This process involved in-depth research of AMS documentation in order to discern the meaning and function of each variable. This variable inspection and determination process led to the discarding of 376 variables (including one complete table). Variables were discarded either due to irrelevant information (e.g. meta information or administrative information), information subject to data protection (e.g. date of birth or social security number), or because it was not possible to precisely identify their content. The remaining variables constituted a set of 188 potentially usable variables.

4.2.2 Data download

Prior to downloading the data, two important steps had to be performed, viz. *pseudonymisation* and the creation of *views in SQL*. The first step was to use *pseudonyms* for client information which could have been traceable back to actual individuals. This was done by exploiting the fact that some of the tables contained several different types of client IDs, among others the so-called `PENR` ("Personen-Nummer"), which is an AMS-generated client ID used for gaining uniquely assignable, yet pseudonymised observations. By joining observations from different tables, all other client IDs were substituted by the `PENR` key. As a consequence, the data no longer revealed personally identifiable information.

The next step was to create *views in SQL*.[33] These views offer the advantage of selecting variables from tables while preserving pseudonymisation by download-

[32] The classification scheme of the AMS database tables was developed in the 1980s. Over the last 40 years, however, this scheme has been compromised. For instance, former INT-tables have been used as BUS-tables, or different versions of the same table have been created, yet never updated. Consequently, the indication of a table name (as a stand-alone criterion) turned out not to be reliable.

[33] According to Oracle's user guide, "[v]iews are virtual tables (analogous to queries in some database products) that select data from one or more underlying tables" [79, p. 1-23].

ing non-compromising variables only (i.e. skipping personally identifiable information). Another major advantage of views is that it is possible to define the relevant time range for the tables prior to download. Data download, however, was dictated by runtime restrictions, meaning that further steps had to be taken in order to make the download manageable. At the same time, the data had to be kept in its chronological order. For this purpose, consecutive numbers were assigned to the observations in the individual tables. Numbering the observations allowed for splitting up a table into several blocks of n observations, without spoiling the chronological order within the data.

After these preparatory steps had been taken, the download was conducted. The final download resulted in 35.7 GB of data, distributed over 675 table fragments. The download also revealed that the *sample* comprised 1,005,167 individual observations, which equals *the number of clients registered with the AMS for at least one day in the year 2017.*

4.3 Data preparation

The objective of data preparation was to create one single table containing all the data. Due, in particular, to the computational restrictions of this study, it was necessary to undertake certain preparatory steps in order to be able to produce one *master table* containing the *basic set* of variables for this thesis.

In the following, this section presents these data preparation steps. These include the classification of *variable types* as well as the creation of the final *master table* containing all the downloaded variables.

4.3.1 Variable types

Prior to being able to create a *master table*, it was necessary to organise the *basic set* of variables. Overall, there were three different types of variables (see Appendix C, table 4):

- production variables
- additionally created variables
- auxiliary variables

Production variables. As the name suggests, *production variables* are variables which are produced by different client support applications. They represent the value recorded at the last *report date*, examples being sex, education or German language skills. In this analysis, these variables were fed directly into the

random forest model.

Additionally created variables. *Additionally created variables* are derivatives of *production variables*, i.e. they are (self-)calculated variables. The most important *additionally created variable* was the *target variable*. The *target variable* was calculated by counting the number of days in (uninterrupted) employment relationships immediately following the *last report date* for the last *Uni-status* recorded by the AMS.[34] The observation time-span for this *post-career* was January 2017 to December 2018. For running statuses, January 2019 was also included, as this was the last month available at the time of data downloading. The *post-career* data showed that there were great variations in the lengths of observed uninterrupted employments, ranging from 0 to 760 days.[35]

While the purpose of this study was to receive an estimation of *sustainable employment*, it also appeared useful to calculate a *pre-career* for each client. The benefit of the *pre-career* is to have more information on a client's occupational career history. The *pre-career* was defined as the time-span of two years preceding the status *report date* last recorded with the AMS.[36] For youths or people only recently having arrived in the Austrian labour market, *pre-careers* were sometimes shorter than two years.[37] The notion of the *pre-career* opened up opportunities for calculating additional variables related to the *pre-career*. Examples are the number of days in different statuses or the number of unemployment episodes within the *pre-career*.

Auxiliary variables. *Auxiliary variables* were solely used as intermediaries, which were necessary for calculating other variables. Overall, there were only two auxiliary variables, viz. `UNINTERRUPTED_EMPLOYMENT _DAYS` and `CAREER_START`. `UNINTERRUPTED_EMPLOYMENT_DAYS` represented the auxiliary variable necessary for calculating the target variable `SUSTAINABLE_EMPLOY MENT`, as it counted the number of days in uninterrupted employment relationships within the *post-career*. The second auxiliary variable, `CAREER_START`, represented the beginning of the *pre-career*, dating back two years from the last *report date*. The purpose for creating this variable was to set the mark for the start-date of the individual client's *pre-*

[34]The *last report date* was calculated from the table `MON_UNI_STATUS_INT`. This table comprised five variables, viz. unique client ID, unique company ID, the Uni-status, and the begin and end date of the respective Uni-status.

[35]In the case of 0 days, the client had had a stable *Uni-status* (e.g. unemployment) which dated back beyond the year 2018. In the case of 760 days, the client had started to work on 2 January 2017 and was still working on 31 January 2019.

[36]The start date for the *pre-career* (i.e. the variable `CAREER_START`) was also calculated from the table `MON_UNI_STATUS_INT`.

[37]The shortest *pre-career* recorded in the *basic set* had a length of four days.

career.

4.3.2 Master table

The goal of *data preparation* was to create one *master table* which contained all the variables used in this thesis. An important choice concerning the design of the *master table* was to decide whether to produce a table containing *one* line of observation or n lines of observations per client (with n equal to the maximum number of observations recorded for the same client ID). Given the computational restrictions, the choice was made to create one line of observation per client and that this line would be identifiable by `PENR`.

In practice, creating the *master table* meant rebuilding the seven tables (more precisely, the 675 downloaded table fragments) in such a manner that the *master table* eventually contained one line of observation per client, which featured the client history (i.e. the client's *pre-career*) as it appeared for the last *report date*, but which also included as much client history information as possible. Figure 11 illustrates the process of aggregating client histories by `PENR` number and the additional client history information created in order to provide a richer picture of a client's *pre-career*:

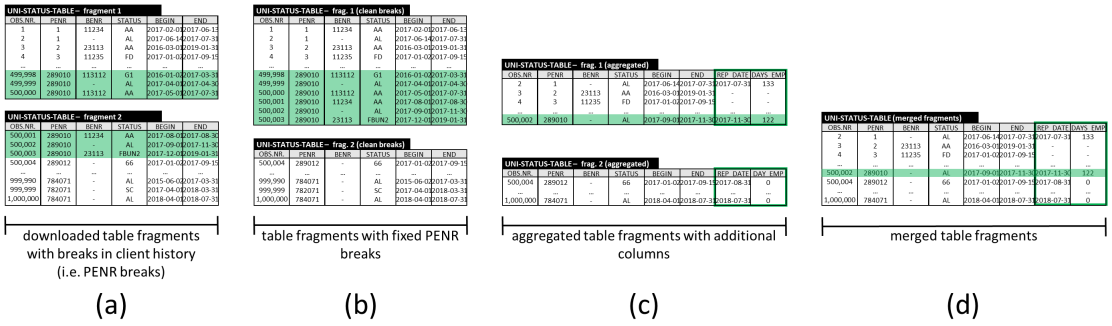

Figure 11: Process of data aggregation and merging of table fragments. The two table fragments in (a) contain lines with the same `PENR` (green). This break in the client's history was fixed by transferring all affected rows (green rows in lower table) to the preceding table fragment (b). After that, the client histories were aggregated into one line per client and additional variables were created (c). The last step involved merging the table fragments (d).

In a first step, breaks in client history needed to be fixated. In order to do so, observations affected by breaks in client history were transferred to the preceding table fragment. Subsequently, each client history was aggregated into one line. While it is likely that data aggregation at this level could lead to information loss, in the context of this thesis, aggregation was necessary due to computational restrictions. Yet, in order to nonetheless provide substance, in a next step, additional *pre-career* variables, which appeared to contain promising informational

value for the estimation of *sustainable employment*, were calculated. This was done primarily by combining status *report dates* from the `MON_UNI_STATUS_INT` table with more specific information from the other six tables. Examples are the number of days in employment or unemployment within the *pre-career*. The last step was to merge the aggregated table fragments. These steps were performed for all tables.

The final joining of all tables was conducted based on the individual client's `PENR`, the unique identifier across all tables. The result was a *master table* of 691 MB containing the *basic set* of 195 variables for 1,005,167 observations.

4.4 Data analysis

Generally speaking, *random forest* does not call for extensive data analysis or data cleaning in advance [69]. However, given the computational restrictions for this thesis, it was necessary to reduce the data to a manageable amount. Thus, the data (the *master table*) was analysed in order receive a better understanding of the nature of the variables and to be able to make an informed decision on which ones to retain (and which ones to omit).

At this point, it appears useful to draw attention to a side-track interest which emerged during *data preparation* and *data analysis*. Initially, the intention was to generate the *random forest* models using one *main set* of variables. However, driven by a particular interest (stemming from an AMS practitioner's view), there was a growing curiosity to find out whether or not the *additionally created variables* made a noticeable difference on the overall outcome or if in fact a set consisting only of data drawn from the AMS database would suffice. In order to be able to assess the influence of the *additionally created variables*, a second set, viz. the *production set* (containing no additionally created variables), was included in the analysis.

In the following, this section begins with the description and visualisation of the *basic set*. Based on these insights, the *basic set* underwent data cleaning, the outcome of which was the *main set*. Lastly, the *main set* was further stripped of the *additionally calculated variables*, which resulted in the *production set*.

4.4.1 Data description and visualisation

The *master table* represented the *basic set* comprising 195 variables. On a superficial level of data analysis, this data was organised for different (rather general aspects), viz. *data (super-)category*, *variable type*, and *data type*. These general categories provided a quick overall impression of the nature of the data (see Appendix C, table 4).

Concerning the *data (super-)category*, the *basic set* contained 156 *AMS-generated* variables, and 39 variables with *person-bound information*.[38] In terms of the previously created *variable types*, the *basic set* contained 146 *production variables*, 47 *additionally created variables*, and two *auxiliary variables*. *Data type* referred to a variable containing either *numerical data* or *categorical data*. The *basic set* contained 99 variables representing *numerical data* and 96 variables representing *categorical data*.

The intention was also to provide a deeper-level understanding of the material at hand, so that such a quantitative and qualitative assessment would lay the foundation for subsequently deciding which variables to retain and which to omit. In the literature, there are several options for visual data description for *numerical* and *categorical data* (e.g. [37, 98, 29]). This thesis employed commonly used visualisation modes, in particular such for illustrating density, distribution, frequency, and relationships between variables. For both *numerical* and *categorical variables*, the final visualisations include a *basic plot* for each variable as well as additional visualisations which were deemed suitable for receiving a richer picture (i.e. for gaining a different perspective on the same observed quality).

Numerical data visualisations. *Numerical data* was visualised in three different ways:

- basic plots
- a correlation matrix and an r^2 matrix
- scatter plots

The *basic plots* for the numerical data consisted of a table presenting basic statistical information, including details on the number of entries, the arithmetic mean, and the median, among others.[39] The *basic plots* also featured a set of five different frequency, density, and distribution visualisations (viz. a *bar plot*, a *histogram* including a *KDE line*, a *box plot*, a *normal q-q plot*, and an *ECDF diagram*), which were all interrelated to some extent, meaning that certain aspects, such as density or skew (i.e. an imbalance within the data distribution), were discernible in several plots. The *basic plots* were scrutinised for similarities and

[38] *AMS-generated data* are variables created in the course of client support. *Person-bound information* refers to variables created through client information.

[39] These statistics were calculated using the function *describe()* provided in *R*'s package *psych*, which offers statistics for the number of entries, arithmetic mean, standard deviation, median, trimmed mean, median absolute deviation, minimum and maximum values, range, skew, kurtosis, and standard error.

differences. Figure 12 depicts the five main types of *basic plots* identified for numerical data:[40]

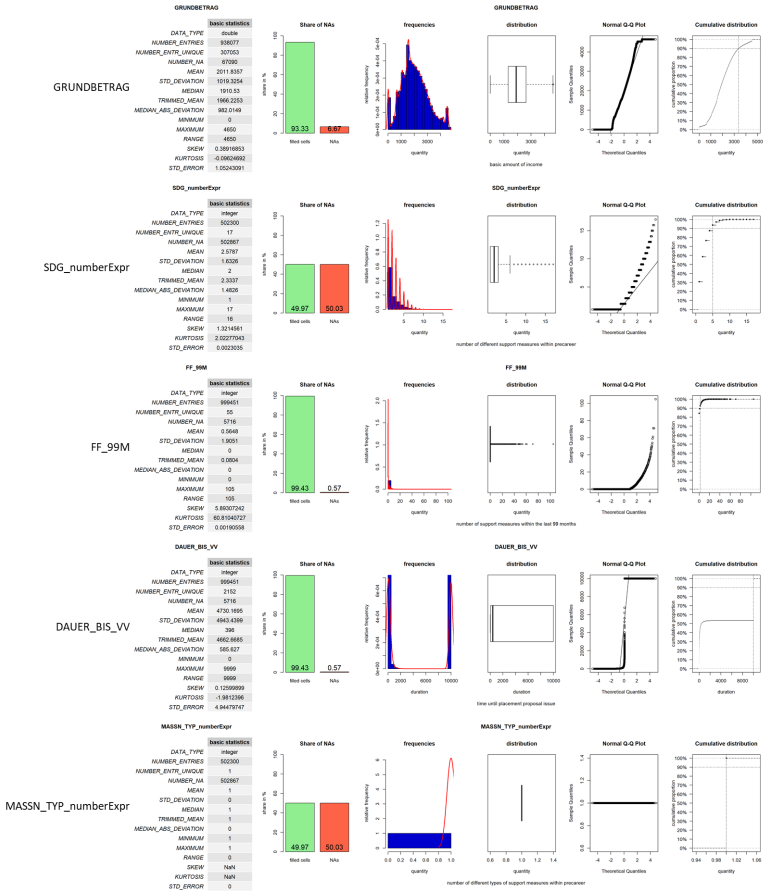

Figure 12: Five main types of *basic plots* for numerical variables. From left to right, a *basic plot* comprises basic statistical information, a *bar plot*, a *histogram (with a KDE line)*, a *box plot*, a *normal q-q plot*, and an *ECDF diagram*. From top to bottom, the main types of basic plots represent variables with *normal distribution*, *right skewed data (discrete expressions)*, *right skewed data (outliers)*, *bimodal data*, and *data with only one expression*.

Starting from the top, the first type identified refers to data which was more or less *normally distributed* (e.g. `GRUNDBETRAG`). The second and third types represented *right skewed data*, whereas a difference was made between data showing a number of discrete variable expressions (e.g. `SDG_numberExpr`), and data showing a large number of outliers (e.g. `FF_99M`). The fourth type constituted an example of *bimodal distribution* (e.g. `DAUER_BIS_VV`), and the fifth type was an example for a variable containing only *one expression* (e.g. `MASSN_TYP_numberExpr`).

[40]The total set of *basic plots* for numerical variables is available under this dropbox link [11].

The *correlation matrices* were based on Pearson's r and r^2 *matrix* (a derivative of Pearson's r). Figure 40 (see Appendix C) displays the *correlation matrix* for the 99 numerical variables of the *basic set*, and figure 42 displays the corresponding r^2 values.[41] The purpose of this specific analysis, on the one hand, was to identify variables with strong relationships (i.e. correlations) and, on the other hand, to assess the amount of variance in one variable explained by another, in order to subsequently reduce the data set by discarding redundant variables (see section 4.4.2).

The third type of visualisation was done via *scatter plots*.[42] Scatter plots are bivariate plots which are used to plot data points of one variable against another in order to visualise the relationship between the variables. In total, 9,702 scatter plots were created[43], and inspected for typical generic patterns. Figure 13 shows a selection of different *scatter plots* which emerged in the *basic set* and examples of the six main generic patterns identified:

[41] Similarly, figure 41 and figure 43 (see Appendix C) show the *correlation matrix* and the r^2 *matrix* for numerical variables for the *main set*.

[42] The scatter plots were created using the function *smoothScatter* provided in *R*'s *graphics* package.

[43] The complete list of *scatter plots* for numerical variables is available under this dropbox link [11].

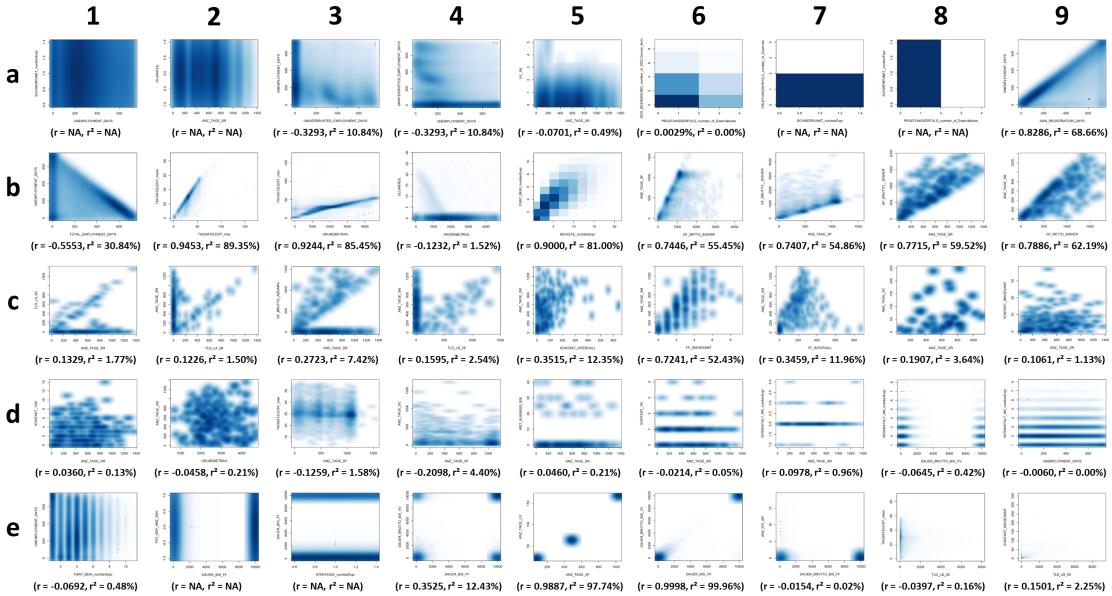

Figure 13: Selection of different *scatter plot* patterns identified in the *basic set*. The six main patterns are *blocks and bars*, *stripes*, *triangles and trapezoids*, *lines*, *clouds*, and *spots*. Each scatter plot also features its respective r and r^2 value (a measure of correlation and an explanation of variance, respectively).

The first pattern identified was that of *blocks and bars*. This pattern featured in scatter plots in which data points were more or less equally distributed, meaning that there was little or no correlation (e.g. a3, a4, and a5). Another reason for block-like patterns were variables with only one or very few expressions (a1, a2, a7, and a8). Similarly, a check-like pattern emerged for cases in which both variables showed only a few expressions (e.g. a6 and b5). *Bars* emerged from variables with only one expression, and for which this one expression represented a numeric extreme, such as "9999" (e.g. e2 and e3).

The second pattern was *stripes*, which occurred in the context of variables with few (discrete) variable expressions. Dependent on the second variable, different types of stripe patterns arose (e.g. d5, d6, d7, d8, d9, and e1).

The third pattern was recognised as *triangles and trapezoids*. *Triangle* shapes emerged when two variables shared the same maximum value and data points below this threshold were more or less equally distributed (e.g. a9 and b1). *Trapezoids* represented the opposite case, i.e. variables sharing a minimum value (e.g. b6, b7, b8, and b9). In some plots, there was an additional zero line along the axes. This occurred in cases with many clients sharing a low value for a particular variable (e.g. c3 and c4).

Next, straight *lines* were identified to represent linear relationships between variables. A line's angle (from left to right) determined the (positive or negative)

orientation of a relationship between variables (e.g. b2 and b3). Some variables also featured zero lines (e.g. c1 and c2).

The pattern *clouds* came in different shapes and sizes. Examples are sharply outlined clouds which often represented non-linear functions (e.g. c6 and c7) or irregular point clouds (e.g. c5, c8, c9, d1, d2, d3, and d4). While not visible at a first glance, also the plots b4, e8, and b9 can be classified as *clouds*.

The last salient pattern was identified as *spots*. *Spots* were understood as accumulations of data points whose location in the *scatter plots* typically hinted towards numerical extremes. Scatter plot e4, for instance, depicts the case in which both variables had an extreme value (i.e. "9999"), but in which most clients featured a low value. The resulting plot pattern showed data point accumulations in all four corners. *Spots* in opposite corners showed the case in which there was a linear relationship between the variables with a small number of extremely high values and a large number of low values (e.g. e6). Accumulations at both extremes, on the other hand, represented the combination of one variable including extremely high values and the other variable representing information that only affected a few clients (e.g. e7). Finally, plot e5 is a special case, as the *spots* in fact represented single data points. This was due to a *smoothScatter* particularity (i.e. that colour densities are created relative to the number of data points plotted).

Categorical data visualisations. *Categorical data* was also visualised in three different ways:

- basic plots
- a p-value matrix (as the result of a chi-squared test of independence)
- a Cramér's V value matrix

In the same vein as for numerical data, the *basic plots* for the categorical data featured a table presenting basic statistical information. However, as only very few operations are sensible in the context of *categorical data*, only four statistical measures were displayed. These statistics are the data type, the total number of entries, the number of unique entries, and the number of missing values. This statistical information was accompanied by two *bar plots* for visualising frequencies. More precisely, these two *bar plots* displayed the relative share of filled cells as opposed to missing values (NAs) and an overview of expression frequencies. Figure 14 depicts three typical *basic plots* for categorical variables:

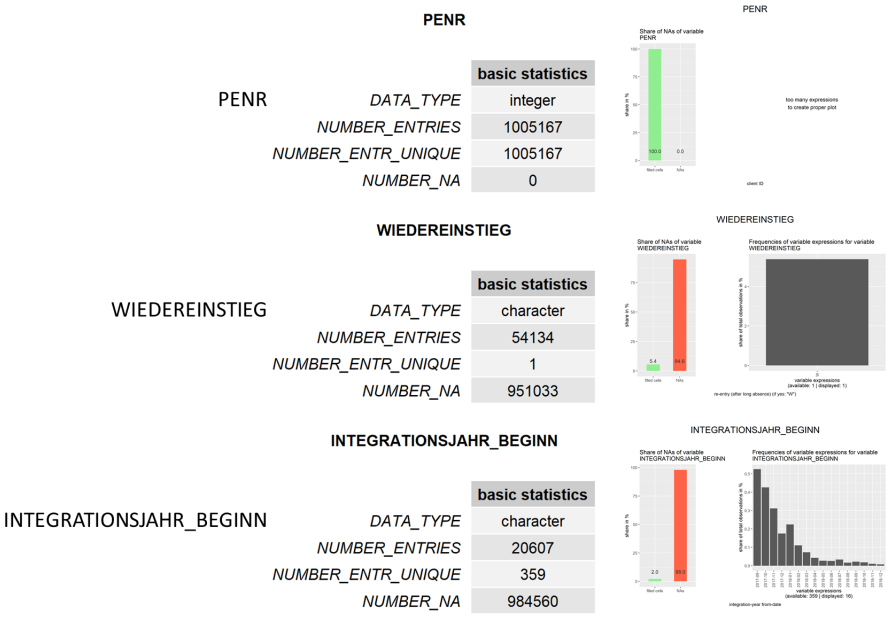

Figure 14: Three typical *basic plots* for categorical variables. From left to right, a *basic plot* comprises *basic statistical information* and two *bar plots*. The three *basic plots* represent variables with *too large a number of expressions to be displayed*, variables with *one expression*, and variables with *aggregated expressions*.

From top to bottom, the first *basic plot* depicts a case in which the number of expressions was simply *too large to be displayed* (e.g. `PENR`). The second *basic plot*, in contrast, shows a variable with only *one expression* (e.g. `WIEDEREINSTIEG`). The last *basic plot* showcases a variable with a *large number of expressions*, which were aggregated in order to be depictable (e.g. `INTEGRATIONSJAHR_BEGINN`).

Similar to the visualisations of the *numerical data*, the intention was to investigate and present the *categorical data* in terms of relationships between variables. For the *categorical data*, however, the measure was one of *independence* (as opposed to a measure of *correlation*). The two methods employed were *chi-squared test of independence* and a *Cramér's V* calculation (a derivative of *chi-squared test*) [54]. Figure 44 (see Appendix C) illustrates the *p-value* heatmap for the 96 categorical variables in the *basic set*, and figure 46 displays the corresponding *Cramér's V* value heatmap for the *basic set*.[44]

4.4.2 Data cleaning

The main driver for cleaning the data was the limited amount of computational capacity. Thus, cleaning the data essentially meant omitting variables, yet in such

[44]Similarly, figure 45 and figure 47 (see Appendix C) show the *p-value heatmap* and the *Cramér's V value heatmap* for categorical variables for the *main set*.

a way that as much information as possible could be retained. The previously conducted in-depth analysis of the variables (see section 4.4.1) served as a basis for decision-making in terms of final variable selection.

In sum, nine *omission categories* were identified.[45] An overview of these categories as well as examples and the number of variables omitted, are provided in table 2:

omission category	examples	variables omitted
auxiliary variable	UNINTERRUPTED_EMPLOYMENT_DAYS, CAREER_START	2
indeterminable content	TAGE_IN_BE_VOR_FDG, TLD_LA_28	4
ID variable	PENR, LAST_BENR_WITHIN_PRECAREER	4
inconclusive information	BESCHAEFTIGUNG_AB, BMS	12
irrelevant for model	KRANKENVSTR	18
no informational value	SPJU, KRE_EINSTCODE	10
redundant variable	PLZ, BEHINDERUNG	12
replaced by FUNDING_RECEIVED	AMF_ID, FDG_ID	16
strong correlation	TOTAL_EMPLOYMENT_DAYS	19

Table 2: Variable *omission categories*.

As previously mentioned, *auxiliary variables* were not part of the model building process as they were proxy variables (see section 4.3.1). In the data set, there were two auxiliary variables, viz. UNINTERRUPTED_ EMPLOYMENT_DAYS and CAREER_START.

For some variables, their content was simply *indeterminable*. On the one hand, this became apparent for abbreviations or descriptions in variable names, such as the abbreviation "BE", which can mean *support* ("Betreuung"), *receipt* ("Bezug"), or *employment* ("Beschäftigung"). In a similar vein, the abbreviation "LA", representing "Leistungsart" (i.e. the *type of benefit receipt*) included no further specification as to the actual type of benefit received.

ID variables were not part of the model building process, as there were too many different expressions. The variable PENR, in particular, had to be excluded, as *R*'s *ranger* package would otherwise have treated it as a feature variable.

Other variables were dropped because of *inconclusive information*, i.e. incomplete, inaccurate, or erroneous information. For instance, the variable BESCHAEFTIGUNG_AB represents the expected start date of an employment relationship. The data collected included start dates ranging from the year 1983 to 2412, the latter which is not possible. Similarly, the variable BMS represented the receipt of demand-oriented minimum income. According to the data, in 2017, only 3,776 clients received BMS, whereas official AMS figures speak of more than 69,000.

[45] The identification of *omission categories* is a good example for the iterative nature of this analysis. In some cases (e.g. *auxiliary variables*), the grounds for omitting such variables only presented themselves after the first round of model generation had been completed. In table 4 (see appendix C), these variables are highlighted with an asterisk (*) in the column *reason for omission*.

Some variables were considered *irrelevant for the model*, as they were either AMS-internal target measures, variables with little informational value, or not exploitable in the context of labour market analysis. The variable `KRANKENVSTR`, for instance, merely denotes the name of a client's health insurance agency.

Variables which contained either no expressions or only one unspecified expression were considered as having *no informational value* for this study. This was the case, for instance, for the variable `SPJU` for which there was only one expression recorded as "*" (meaning "not specified"). For the variable `KRE_EINSTCODE`, there were no entries.

Redundant variables were variables which were implicitly or explicitly covered by other variables. For instance, the variable `GKZ` (community code) implicitly covered the information stored in the variable `PLZ` (postal code). Furthermore, while the variables `BEGUENSTIGUNG` and `BEHINDERUNG` contained information on the same granularity level, `BEGUENSTIGUNG` was better maintained.

Another means to reduce the number of variables was to have them *replaced by the variable* `FUNDING_RECEIVED`. For example, the variable `AMF_ID` stood for a certain support project which covers one or several `FDG_ID`s (which are certain funding and support measures). These were replaced by the binary variable `FUNDING_RECEIVED`.

Finally, certain numerical variables were dropped on the grounds of *strong correlations*.[46] The variable `TOTAL_EMPLOYMENT_DAYS`, for instance, was strongly correlated with the variable `UNFUNDED_EMPLOYMENT_DAYS`, which is why the former was dropped in favour of the latter.

The next step in *data cleaning* involved *replacing missing values* (NAs).[47] In terms of *numerical variables*, NAs were replaced by the value 0, in terms of *categorical variables*, missing values were replaced by placeholders pre-defined by the AMS (and stored in the respective DIM table for each variable, see section 4.2.1).

The process of *data cleaning* resulted in a *main set* containing 98 variables and a *production set* containing 73 variables. As mentioned in the beginning of this section, the difference between these two sets is that the *main set* contains production variables *and* artificially created variables, whereas the *production set* contains *only* production variables (apart from the variable `LAST_REPORT_DATE`).[48]

[46] Although, in general, multi-colinearity is not an issue in terms of a *random forest*'s prediction accuracy [53], highly correlated feature variables negatively affect *variable importance* ranking [40, 111].

[47] An alternative (for replacing missing values) proposed in the literature is *value imputation* (e.g. [102]). However, in the context of the data at hand, the imputation of variables would have created virtual values which may have affected prediction performance. For instance, if a client is not eligible for unemployment benefit, the respective variable value is empty. In the case of *value imputation*, an imputed entry could have led to a false entry, which would not have reflected reality.

[48] The variable `LAST_REPORT_DATE` is necessary for calculating the *target variable*.

5 Results

This chapter, like chapter 4, structurally follows the process of *supervised machine learning* as described in section 3.1.2. Thus, the stages of interest are the remaining stages of *model training*, *model tuning*, and *model testing*.

This chapter is also important, in that it is the first time that the entire study design becomes apparent. Throughout this thesis, hints have been made concerning the elements to be included in the study design (e.g. two different types of *random forest*, two different *split regimes*, two different *sets*). In order to bring these strands together, it appeared useful to provide a visualisation of the study design, which (incidentally) also serves as an overview of the *model generation process*.

In the following, this chapter first presents a general *overview of the model generation process*. It then continues by going into detail for the results produced in the course of the *model training*, *model tuning*, and *model testing* stages.

5.1 Model generation process

Strictly speaking, the *model generation process* started with the *model training* phase - and not with *data analysis*, as depicted in figure 15. However, as briefly mentioned above, figure 15 also serves as a visualisation of the (basic) study design, and since two of the basic study design elements, viz. the *main set* and the *production set*, were the result of *data analysis*, this previous analytical stage was included in the depiction:

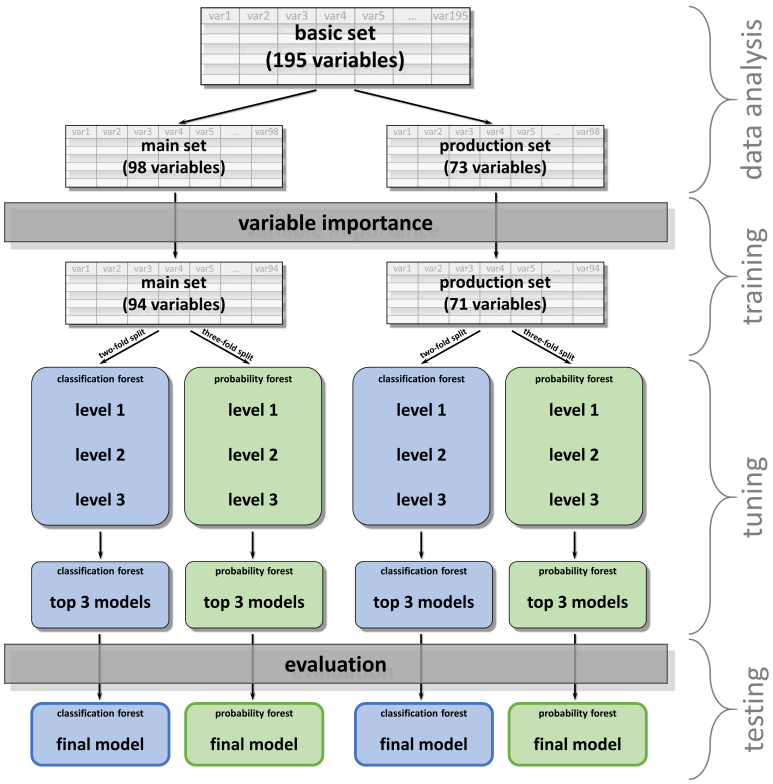

Figure 15: Overview of the *model generation process* (and the basic study design). In the stage *data analysis*, the *main set* (98 variables) and the *production set* (73 variables) were derived from the *basic set* (195 variables). The *training stage* was initiated with the processing of *variable importance*, which led to some variables being omitted, ultimately resulting in a *main set* with 94 variables and a *production set* with 71 variables. Both sets were subjected to a particular split regime, i.e. a *two-fold split* for *classification forests* (blue *threads*) and a *three-fold split* for *probability forests* (green *threads*). The next stage, *model tuning*, comprised three consecutive *tuning levels*, each building on the previous level and intended to refine the models' quality. In the last stage, *model testing*, the previously established top 3 models for each *thread* were evaluated. This stage allowed for determining a final model for each combination of *set* and *model type*.

Starting from top to bottom, and to briefly refresh, the *basic table* containing 195 variables underwent *data cleaning*, which eventually resulted in a *main set* with 98 variables and a *production set* with 73 variables. The larger number of variables for the *main set* was due to *additionally created variables* (see section 4.3.1).

The actual *model generation process* started with the *training stage*. This stage involved the determination of important and negligible variables and is presented in section 5.2. The next stage, *tuning*, was the stage in which the two *split regimes* came into play. While the *classification forests* (in each set) were processed with a *two-fold split* (blue *threads*), the *probability forests* (in each set) were executed with a *three-fold split* (green *threads*). In total, this practice produced four *threads*.

These, subsequently, underwent three consecutive *tuning levels*, which allowed for determining the three top-performing models for each *thread*. The final *testing stage* was the stage in which all models (which had made it so far) were evaluated. The outcome of this stage were four final models, i.e. the best-performing model for each *thread*.

While creating both *model types* for both *sets* and both *split regimes* in principle would have been perfectly possible, this would have doubled the time and effort needed for *model tuning*. Thus, the decision was made to focus on the combinations presented in figure 15.

5.2 Model training

The first stage in any *supervised machine learning process* is the *training phase* [42]. Essentially, this stage involves producing a series of random forests, which are the groundwork for subsequently determining a suitable *entry point* (i.e. the best-performing forests) for the ensuing *tuning phase*.

This series of *random forests* is the result of combining different parameters, which in this case were the *number of trees*, the *number of variables*, and the *number of split candidates (mtry)*.[49] This training step was performed for both the *main set* and the *production set* and in total covered 80 different forest configurations. These first insights revealed that a combination of more variables (*main set*), a larger forest (20 trees or more), and an *mtry* value of 10 showed a better performance than other combinations (see Appendix D, figures 48 and 49).

The next step in this *training phase* involved checking the data for *variable importance*. The *variable importance* method used was Breiman's *permutation importance* provided in *R*'s *ranger* package [18, 117].[50] Based on the results of the initial random forest series, *variable importance* was performed for both sets, whereas hardware restrictions forced a compromise in terms of the maximum *number of trees*.[51] *Permutation importance* revealed negative importance values for four variables of the *main set* (viz. `ANZ_EPI_SR`, `ANZ_TAGE_SR`, `PRUEFUNGSERFOLG`, and

[49] The following parameter configurations were used: Twenty different values between 1 and 1,000 for the *number of trees*, two different sets for the *number of variables* (98 variables for the *main set* and 73 variables for the *production set*), and two different values for *mtry* (5 and 10).

[50] The fact that the variables comprised in the sets varied greatly in terms of their numbers of expressions would have justified using Altmann et al.'s corrected *permutation importance (PIMP)* [3]. This, however, was not possible due to hardware limitations.

[51] Although the *number of trees* in the initial random forest series (figures 48 and 49) goes up to 1,000 trees, the rather laborious task (in terms of memory consumption) of calculating *permutation importance* only allowed for running forests with a maximum of 750 trees. Due to that, yet keeping in mind the result of the initial random forest series, the configurations chosen for the calculation of *permutation importance* for both the *main set* and the *production set* were as follows: *number of trees* = 750, *mtry* = 10, *splitting rule* = gini.

PRUEFUNGSERFOLG_number_of_Exam.terminations, see Appendix D, figure 50) and for two variables of the *production set* (viz. ANZ_EPI_SR and PRUEFUNGSERFOLG, see Appendix D, figure 51). Dropping these variables led to slightly improved performances for all *threads*. Thus, the outcome of *model training* was a (new) *main set* with 94 variables and a (new) *production set* with 71 variables.

The *model training* steps described above are also visualised below in figure 16 (*main set*) and figure 17 (*production set*). Each of the plots displays a series of random forests for the two *mtry* values 5 and 10 *before* and *after* having dropped the above mentioned variables. While the left plots in each figure show continuous x-axes, the right plots display the actual forests grown with the *number of trees* displayed on the discrete x-axes. Moreover, the upper plots of each figure show *classification forests* assessed with the performance measure *geometric mean (GM)*[52] and the lower plots display *probability forests* tested with the *area under the ROC curve (AUC)*. In cases in which the forests were beyond a size of 300 trees, the reduced sets of both the *main set* and the *production set* perform slightly better than their respective initial sets. Consequently, the reduced sets (viz. the *main set* with 94 variables and the *production set* with 71 variables) were used for the subsequent *tuning stage*.

[52]What is measured here is the *geometric mean* of the two performance measures *sensitivity* (or *true positive rate*) and *specificity* (or *true negative rate*).

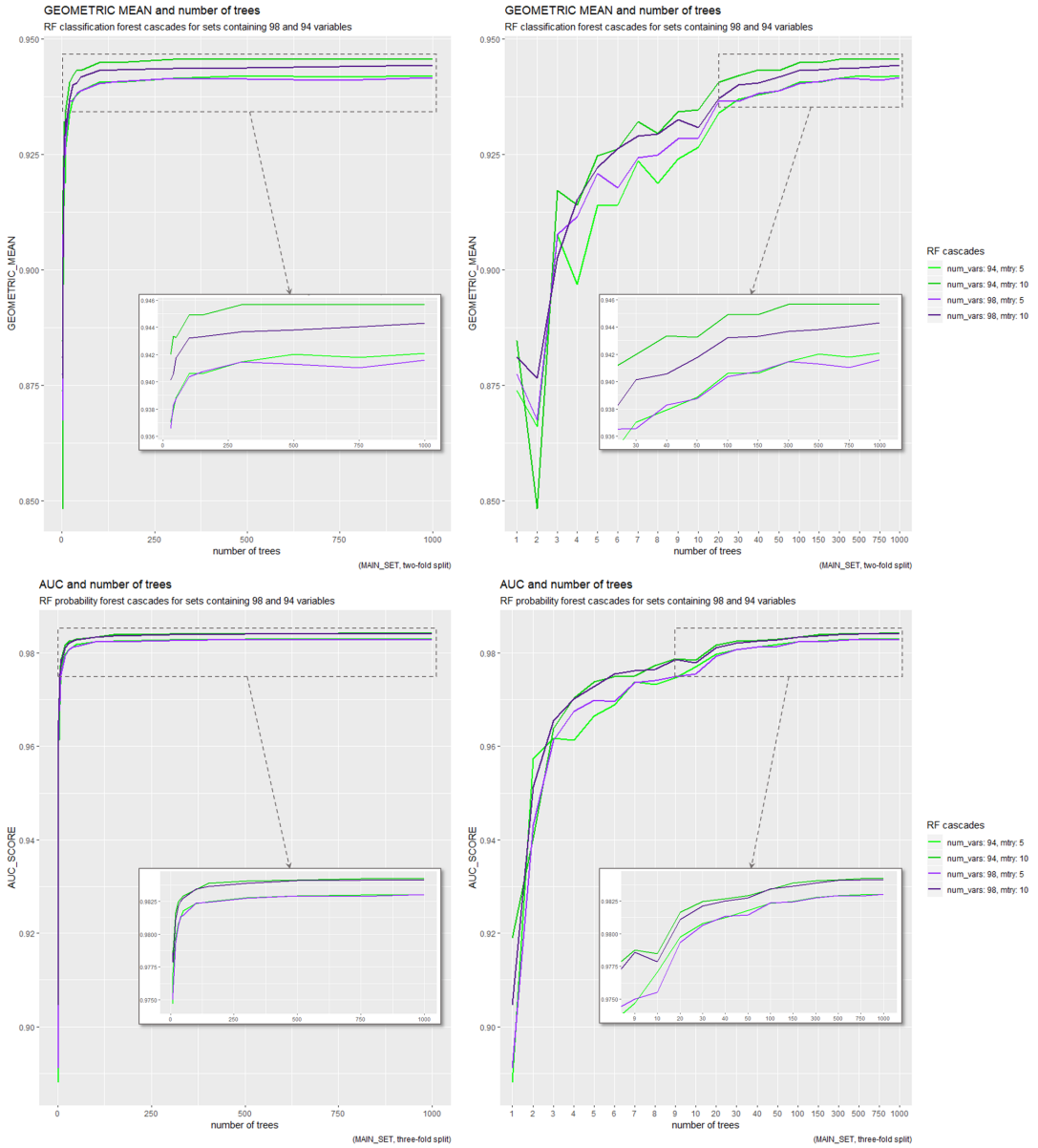

Figure 16: Random forest series for the *main set* before (98 variables) and after (94 variables) dropping variables with negative importance values. While the left plots show continuous x-axes, the right plots display discrete x-axes with the random forests' actual *number of trees*. The upper plots constitute *classification forests* with the *geometric mean* as their performance measure, the lower plots represent *probability forests* with the AUC as their performance measure. If the number of trees was sufficiently high (300+ trees), the reduced *main set* with 94 variables exhibited slightly better performance values than the original set with 98 variables.

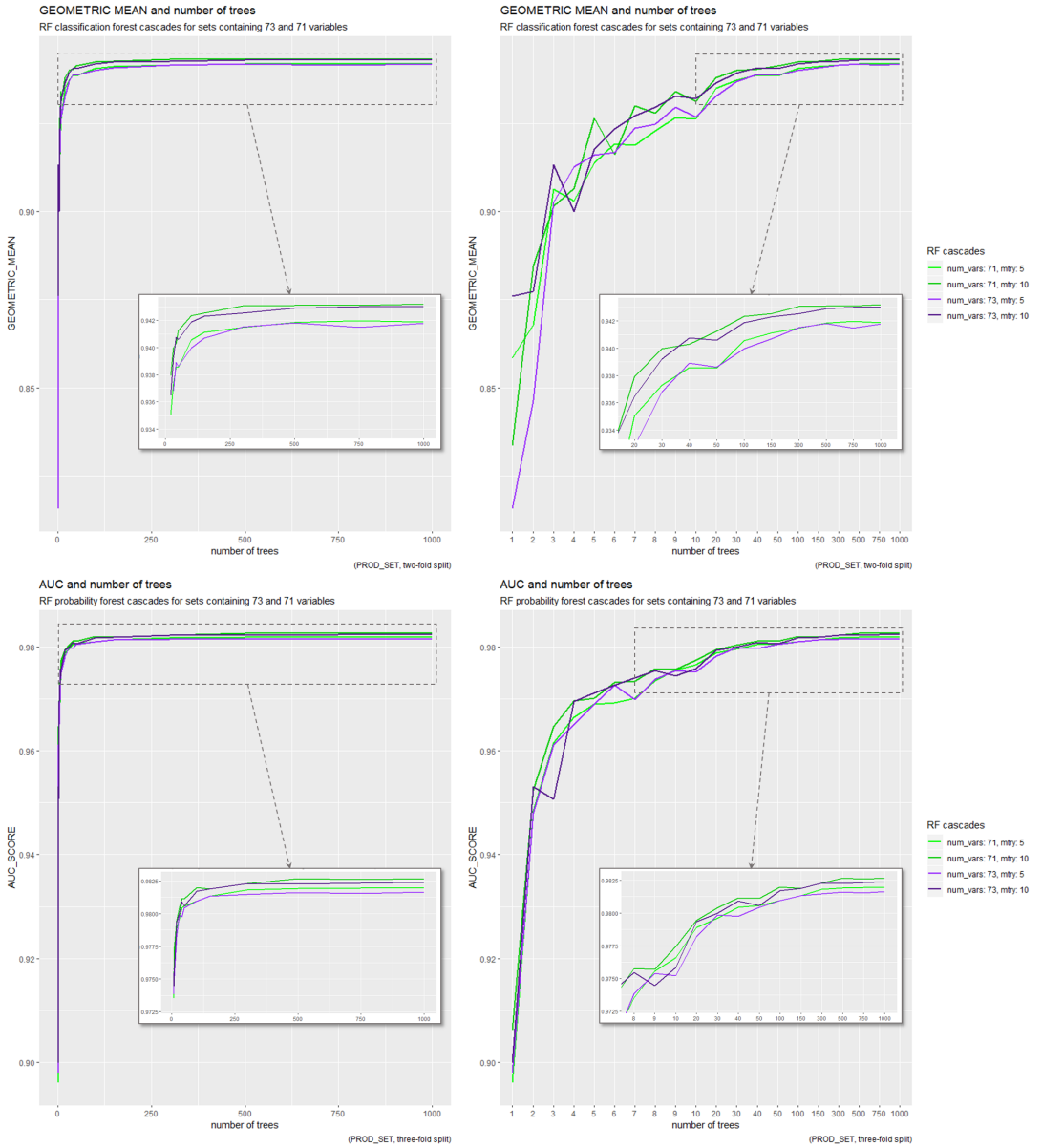

Figure 17: Random forest series for the *production set* before (73 variables) and after (71 variables) dropping variables with negative importance values. While the left plots show continuous x-axes, the right plots display discrete x-axes with the random forests' actual *number of trees*. The upper plots constitute *classification forests* with the *geometric mean* as their performance measure, the lower plots represent *probability forests* with the *AUC* as their performance measure. If the number of trees was sufficiently high (300+ trees), the reduced *production set* with 71 variables exhibited slightly better performance values than the original set with 73 variables.

5.3 Model tuning

The purpose of *model tuning* was to identify the best-performing model within each *thread*. The process involved increasing the performance of a random forest model by tweaking different (hyper-)parameters. For this particular *model tuning* phase, all (hyper-)parameters introduced in section 3.3.3 were employed.[53] To recall, these parameters were the *number of trees*, the *replacement strategy*, the *number of randomly drawn candidate variables (mtry)*, the *sample size*, and the *node size*.

The configurations for the *number of trees* and the configurations for the *sample replacement strategy* were applied throughout the entire tuning process. In terms of the *number of trees*, the three different configurations used were 750, 1,000, and 1,500 trees. Regarding the *sample replacement strategy*, both *sample drawing with replacement* (or *bootstrap sampling*) and *sample drawing without replacement* were applied. For the remaining three parameters, however, a three-tier tuning approach was developed. The intention was to give special attention to one particular parameter on each *tuning level*, while the other parameters remained unchanged. Tuning was performed in the following order:

- *tuning level 1*: mtry
- *tuning level 2*: training set size
- *tuning level 3*: minimum node size

The best-performing *mtry* values of each tuning level were passed on to the next tuning stage. From the final *tuning level 3*, the three best-performing models of each *thread* were passed on for final *model evaluation* (see section 5.4).

The reason for employing this *three-tier tuning process* was the high number of possible parameter combinations. Breaking the tuning process into smaller units (i.e. *tuning levels*) allowed for a more structured approach, which appeared suitable for dealing with the sheer amount of possible parameter combinations. However, this approach also implies the assumption that the best results on one tuning level would still be the best results on the subsequent tuning level. For instance, on *tuning level 1*, the five best-performing *mtry* values were selected for further tuning on the next tuning level. On *tuning level 2*, however, the object of consideration was the *training set size*. If some of the *mtry* values "left behind" suddenly performed better in the changed environment of different *training set sizes*, this would never be detected. Although no indication of such behaviour was observed in the course of *model tuning*, this uncertainty remains.

[53] With the exception of the *splitting rule*, as *gini impurity* was the overall choice for the entire analysis.

The remainder of this section presents the three tuning levels and their respective results (which for *tuning levels 1 and 2* constitute the inputs for the subsequent tuning level). This section concludes with presenting the three best-performing models for each of the four *threads*.

5.3.1 Tuning level 1: mtry

The (hyper-)parameter under observation on this tuning level was the *number of split candidates (mtry)*. Each of the four *threads* was tested for eight different *mtry* values. While the first six *mtry* values were the same for both sets (10, 15, 20, 30, 40, and 50), the larger *main set* (94 variables) was additionally tested for the *mtry* values 70 and 90, and the smaller *production set* (71 variables) was additionally configured for the *mtry* values 60 and 70. The choice of the highest *mtry* value for each of the sets was made on the grounds that this value equalled the number of predictors in the set (which essentially constitutes *bagging*, see section 3.3.1). Apart from these *mtry* settings and the recurring parameter configurations for the *number of trees* and *sample replacement* (see section 5.3), the *training set size* was set at 75% for *classification forests* and at 52.2% for *probability forests*.[54]

The eight different values for *mtry* in combination with three different configurations for the *number of trees* and two options for the *sample replacement strategy* resulted in 48 models for each *thread*. Accordingly, *tuning level 1* output a total of 192 models (for all four *threads*). From the eight different values for *mtry* tested in *tuning level 1*, the five best-performing values were passed on to *tuning level 2*. The performances of the different models are also displayed in figure 18:

[54]The *training*, *validation* and *test sets* for the *probability forests* were created (from the *main* and *production sets*) as follows: In a first step, the respective sets were split into a temporary *training set* (covering 75% of all observations) and a *test set*. The temporary *training set* was further split into the actual *training set* (i.e. 70% of the temporary *training set* or 52.2% of the overall set) and a *validation set* (i.e. 30% of the temporary *training set* or 22.5% of the overall set).

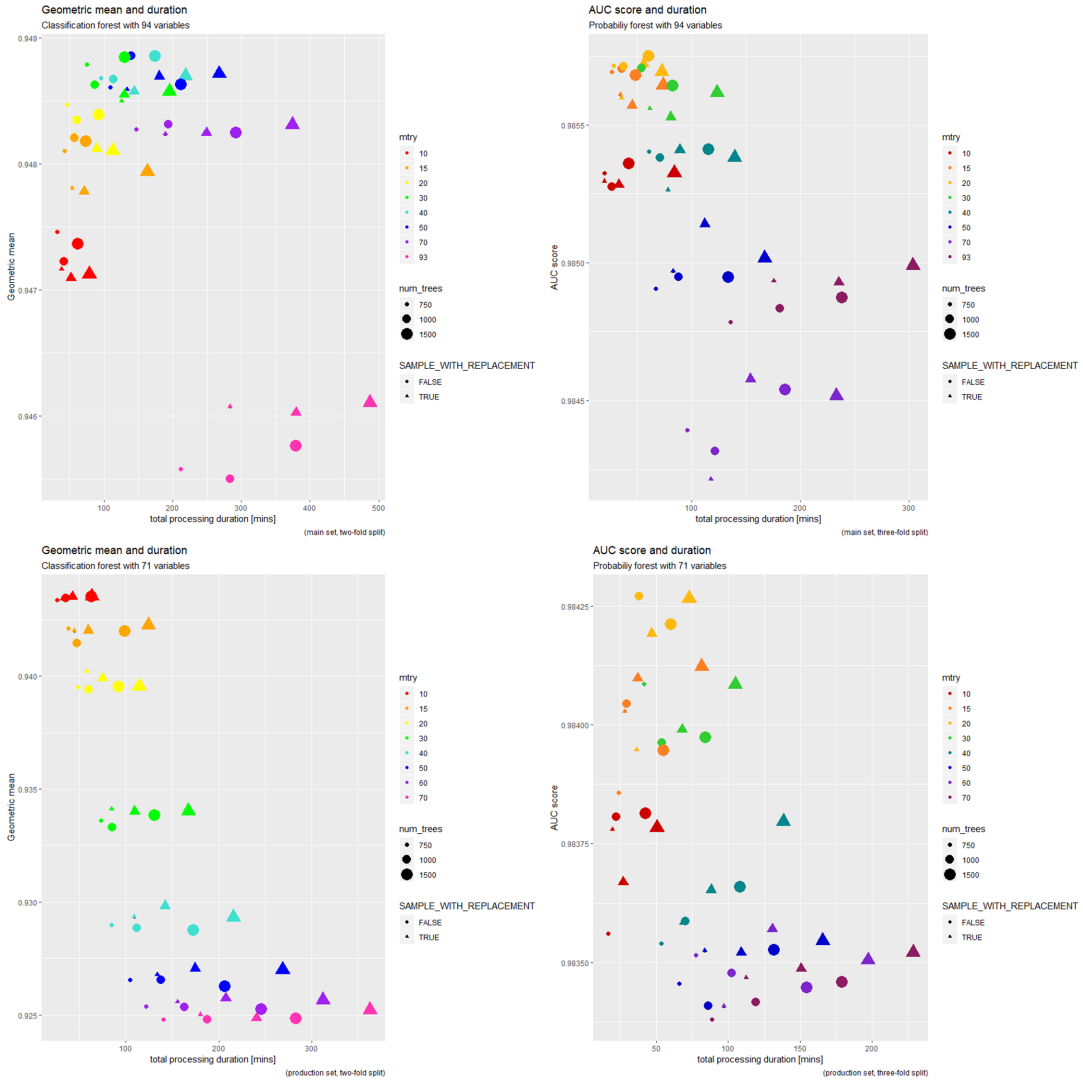

Figure 18: Tuning level 1: Performance of models created from the *main set* (above) and the *production set* (below). The two plots on the left-hand side represent *classification forests* and the two plots on the right-hand side constitute *probability forests*. On their axes, the plots show the duration for the creation of the models (x-axes) and their respective performance measures (y-axes), viz. *geometric mean* for *classification forests* and the *area under the ROC curve* for *probability forests*. Further parameters displayed are *mtry* (colour-coded), the *number of trees* (size of data points), and the *sample replacement strategy* (shape of data points). The five best-performing *mtry* values of each plot were passed on to the next tuning level.

The two plots on the left-hand side represent *classification forests* and the plots on the right-hand side constitute *probability forests*. The x-axis shows the overall duration for the creation and the testing of the models. The y-axis indicates the performance measures used, viz. *geometric mean* for *classification forests* and the

area under the ROC curve for *probability forests*. The remaining three parameters visualised in the plots are the colour-coded *mtry* values, the *number of trees* (represented by the size of the data points), and the *sample replacement strategy* used (indicated by the shape of the data points).

5.3.2 Tuning level 2: training set size

The (hyper-)parameter of interest in *tuning level 2* was the *training set size*. In total, four configurations were chosen, viz. 60%, 70%, 80%, and 90% of the overall observations. This consequently resulted in four different versions of set splits for each *data set* and *forest type*. These were stored and, from then on, repeatedly used whenever the respective *training set size* became relevant (e.g. in *tuning level 3*).

It appears suitable at this point also to draw attention to implications for *probability forests* when creating different *training set sizes*. In this thesis, *probability forests* were paired with the three-fold split regime, which resulted in a *training*, a *validation*, and a *test set*. As a rule, if a *training set* is very large, the *validation* and *test sets* are rather small. However, as has previously been mentioned, in order to maintain a high standard in terms of a model's final evaluation, the size of the *test set* needs to be sufficiently high. Thus, for this thesis, a particular requirement was introduced, viz. that the *test set* be twice the size of the *validation set*.[55]

Overall, the five different *mtry* values in combination with the three different configurations for the *number of trees*, both options of the *sample replacement strategy* and four different *training set sizes* resulted in 393 models.[56] In the same

[55]The creation of the three sets was performed as follows: In a first step, the *training set size* was calculated with this formula: $trainSize + (1 - trainSize) * 1/3)$, where the $trainSize$ is the desired size of the *training set*. The share of the observations resulting from this formula represented the *temporary training set*. The remaining observations constituted the *test set*. The *validation set* was drawn from the *intermediate training set*. This was done with the following formula: $1 - ((1 - trainSize * 1/3)/(trainSize + (1 - trainSize) * 1/3))$. While applying the resulting share on the *temporary training set* created the *actual training set*, the remainder constituted the *validation set*. For a desired *training set size* of 60%, the first formula resulted in a *temporary training set* of size 73.33% (of all observations). The remaining 26.67% represent the *test set*. The second formula resulted in a value of 0.8181. Multiplying this value with the size of the *temporary training set* led to the desired (actual) *training set size* of 60%. The remainder of 13.33% constituted the *validation set*.

[56]In theory, the combination of parameters should have allowed for 120 models per *thread* (5 *mtry* values * 3 *number of trees* configurations * 2 *sample replacement strategies* * 4 *training set sizes* = 120 models per *thread* and 480 models in total). However, due to the hardware limitations, some parameter combinations were simply not calculable. For instance, with a *training set size* of 80% or more, forests with more than 1,000 trees were not computable. Furthermore, small values for *mtry*, such as 10 or 15, created more complex trees, leading to higher memory consumption, which resulted in memory overflows. The distribution of models across the four *threads* is as follows: *main set* = 100 *classification forests* and 95 *probability forests*, *production set* = 100

vein as for *tuning level 1*, the best-performing *mtry* values were passed on to the next and final *tuning level 3*. However, in order to decrease the possible variable combinations (in particular with regard to the time needed to compute the models), not all four *training set size* configurations were considered for further analysis. Instead, the two best-performing *mtry* values of the two best-performing *training set sizes* per *thread* were chosen to be processed in the last tuning level.

The following figures 19 to 22 display the performances of the five best-performing *mtry* values of *tuning level 1* for the four different *training set sizes* queried in *tuning level 2*. The representations of the plots follow the same rules as those introduced for figure 18:

classification forests and 98 *probability forests*.

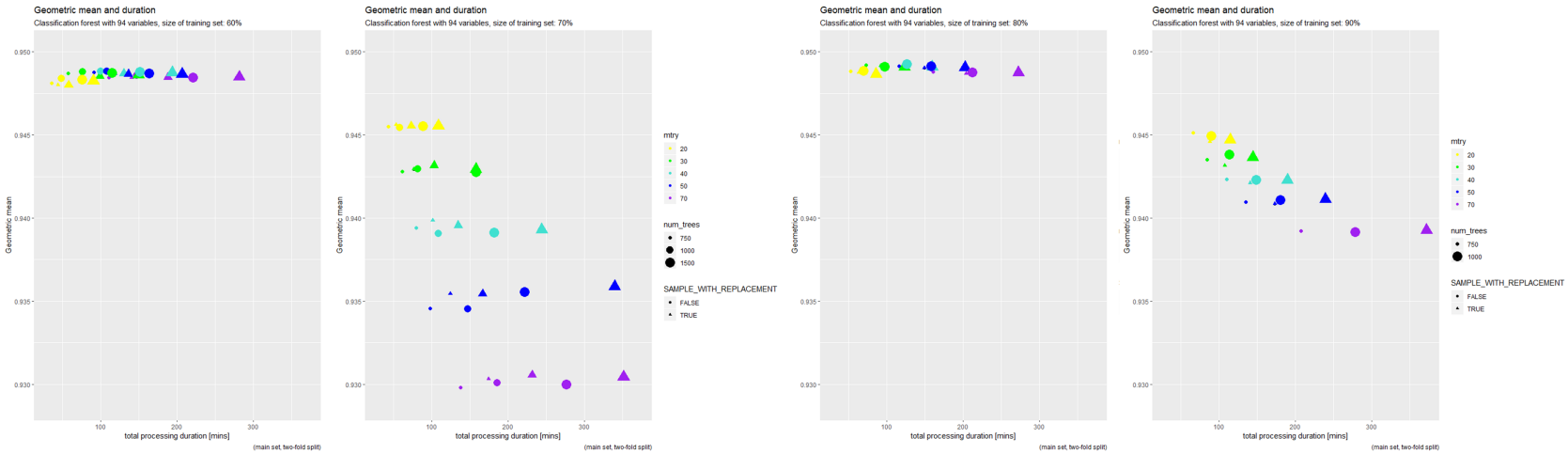

Figure 19: Tuning level 2: Performance of *classification forest* models created from the *main set*. The five best-performing *mtry* values from the preceding *tuning level 1* were calculated for four different *training set sizes* (60%, 70%, 80%, and 90%). A more detailed version of these plots (with zoomed-in y-axes) is presented in figure 52 (see Appendix D).

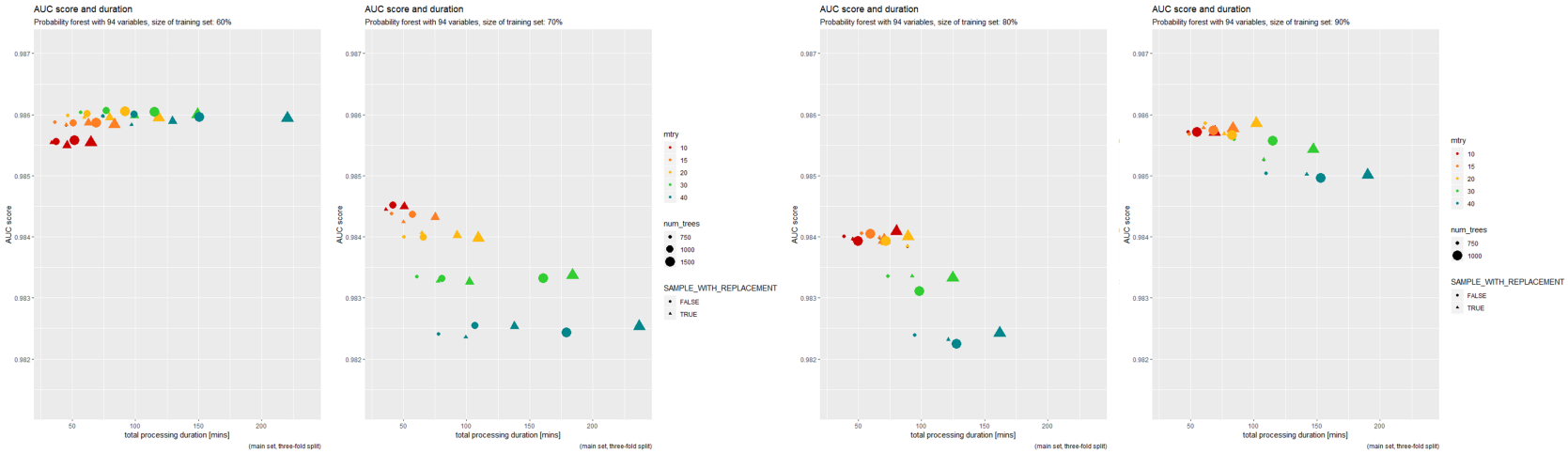

Figure 20: Tuning level 2: Performance of *probability forest* models created from the *main set*. The five best-performing *mtry* values from the preceding *tuning level 1* were calculated for four different *training set sizes* (60%, 70%, 80%, and 90%). A more detailed version of these plots (with zoomed-in y-axes) is presented in figure 53 (see Appendix D).

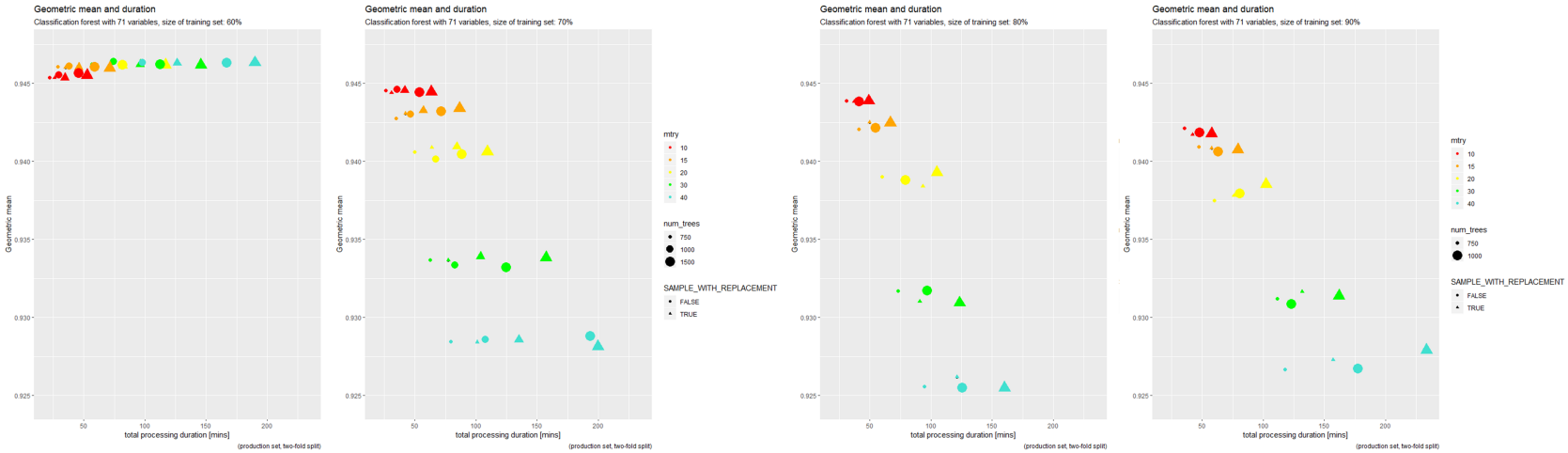

Figure 21: Tuning level 2: Performance of *classification forest* models created from the *production set*. The five best-performing *mtry* values from the preceding *tuning level 1* were calculated for four different *training set sizes* (60%, 70%, 80%, and 90%). A more detailed version of these plots (with zoomed-in y-axes) is presented in figure 54 (see Appendix D).

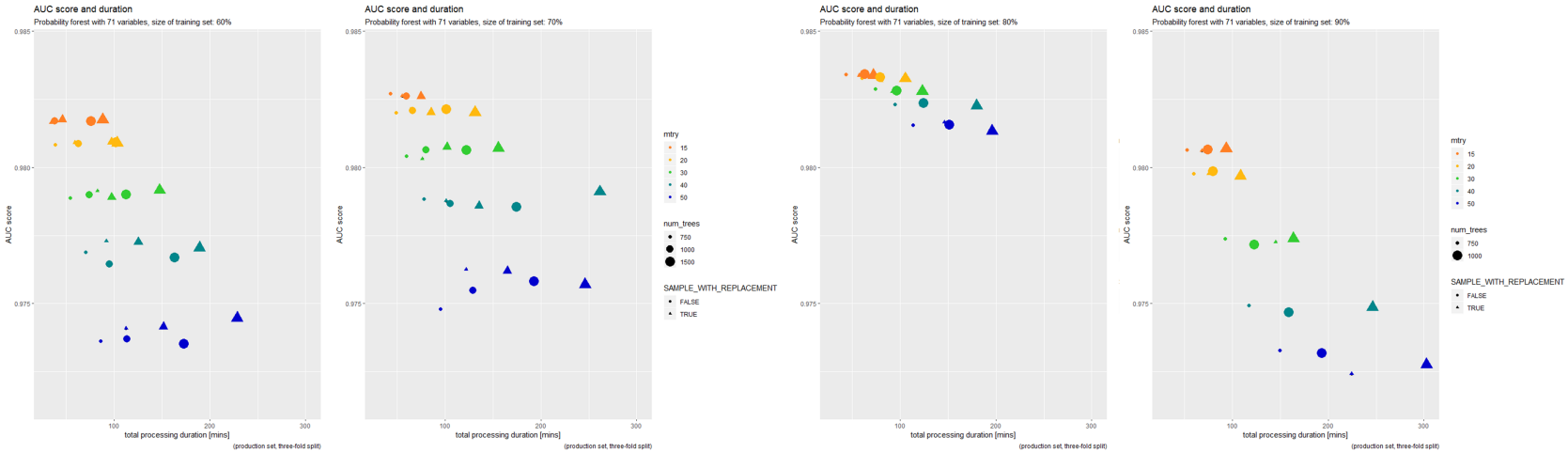

Figure 22: Tuning level 2: Performance of *probability forest* models created from the *production set*. The five best-performing *mtry* values from the preceding *tuning level 1* were calculated for four different *training set sizes* (60%, 70%, 80%, and 90%). A more detailed version of these plots (with zoomed-in y-axes) is presented in figure 55 (see Appendix D).

5.3.3 Tuning level 3: minimum node size

The final *tuning level 3* had a special interest in the *minimum node size*. The *minimum node size* refers to the number of observations in a terminal node (or *leaf*) [93]. For this parameter, apart from the default values[57], two additional values were chosen, so that all *threads* in total were scrutinised for three different *minimum node size* values, viz. 1, 5, and 10. As previously mentioned, the settings for this tuning stage also included input from *tuning level 2* (viz. the two best-performing *mtry* values[58] of the two best-performing *training set sizes*).

Similar to difficulties encountered with the *training set size*, the *minimum node size* also affected computational performance: the smaller the *minimum node size*, the deeper the trees and the higher memory consumption. In terms of *classification forests*, this was not an issue, as the default value equalled 1, meaning that this setting already produced the most complex trees. However, in the context of *probability forests*, where the default value was 10, this was an issue. The remaining lesser *node size* values (1 and 5) produced more complex models, which exceeded the computational boundaries. Consequently, some of the models were not calculable. This was exacerbated in particular in cases were a low *minimum node size* encountered a high *training set size* and a low *mtry* value.

The three different values for the *minimum node size* for the two best-performing *mtry* values of the two best-performing *training set sizes* in combination with three configurations for the *number of trees* and both options of the *sample replacement strategy* led to a total of 264 models.[59] From the 753 models generated within the three tuning stages[60], the top 3 models of each *thread* were identified and subsequently passed on for their final *evaluation*.

[57] In *R*'s *ranger* package, the default values for the *minimum node size* are 1 for *classification forests* and 10 for *probability forests* [116].

[58] In some cases, three different *mtry* values had to be passed over to this tuning stage due to bad separability of the two top-performing *mtry* values from the next best *mtry* value. This issue exclusively affected the *main set*, but both model types: While in terms of *classification forests*, both *training set sizes* (60% and 80%) were concerned, with *probability forests* only the larger *training set size* (90%) was affected.

[59] Similar to *tuning level 2*, it was not possible to calculate certain parameter combinations. In theory, it should have been possible to calculate 288 models (3 *node sizes* * 2 *mtry* values * 2 *training set sizes* * 3 *number of trees* configurations * 2 *sample replacement strategies* = 72 models per *thread* and 288 models in total). However, only 264 were calculable (*main set*: 90 *classification forests* and 56 *probability forests*, *production set*: 72 *classification forests* and 46 *probability forests*).

[60] The distribution of models per tuning level is as follows: *tuning level 1* = 192 models, *tuning level 2* = 393, *tuning level 3* = 168. In fact, in this chapter, 849 models were displayed. The reason for this is that 96 models, which were generated in *tuning level 2* where also displayed in *tuning level 3*. These models are such cases in which the *minimum node size* represented the default settings, and therefore did not have to be re-calculated.

Below, the figures 23 to 26 display the performance of the two best-performing *mtry* values of the two best-performing *training set sizes* for three different *minimum node sizes*. The plots are presented in line with the rules applied for the *tuning level* figures presented so far:

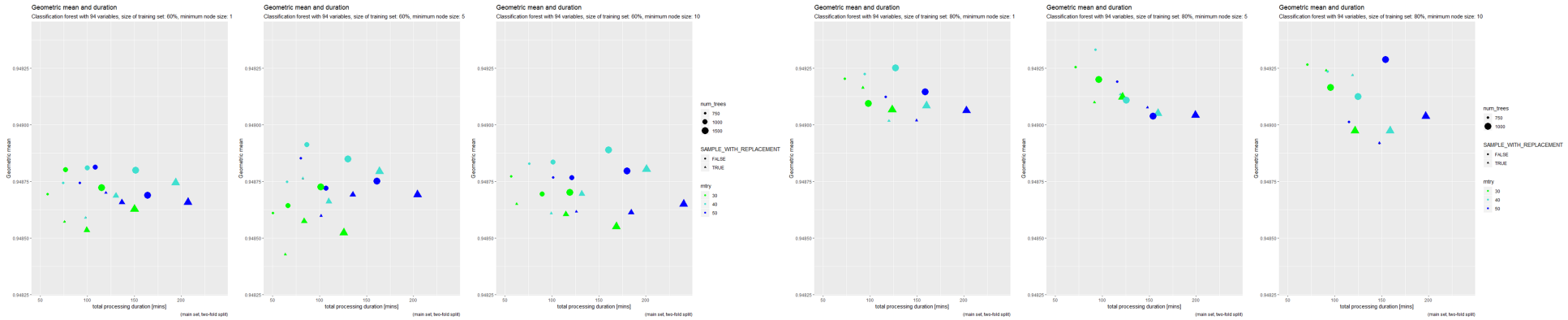

Figure 23: Tuning level 3: Performance of *classification forest* models created from the *main set*. For the two best-performing *mtry* values from the two best-performing *training set sizes* of the preceding stage, three different *minimum node sizes* were calculated (1 = default value, 5 and 10). The three plots on the left represent the three different *node size* values for a *training set size* of 60%, the three plots on the right display the three *node sizes* for a *training set size* of 80%. A more detailed version of these plots (with zoomed-in y-axes) is presented in figure 56 (see Appendix D).

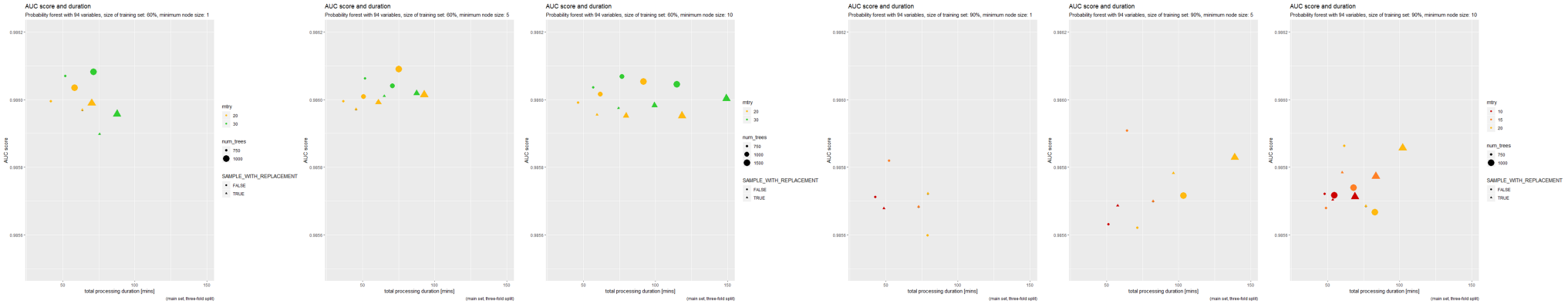

Figure 24: Tuning level 3: Performance of *probability forest* models created from the *main set*. For the two best-performing *mtry* values from the two best-performing *training set sizes* of the preceding stage, three different *minimum node sizes* were calculated (1, 5, and 10 = default value). The three plots on the left represent the three different *node size* values for a *training set size* of 60%, the three plots on the right display the three *node sizes* for a *training set size* of 90%. A more detailed version of these plots (with zoomed-in y-axes) is presented in figure 57 (see Appendix D).

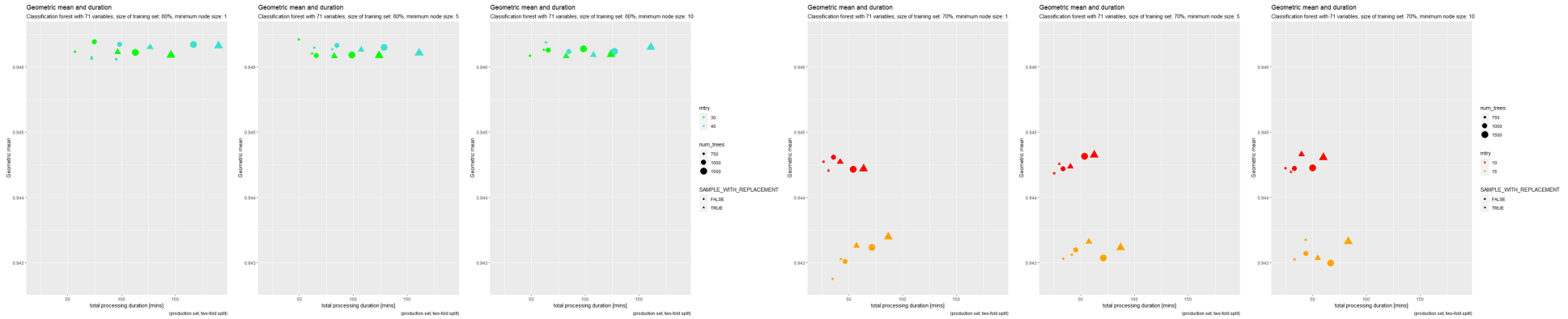

Figure 25: Tuning level 3: Performance of *classification forest* models created from the *production set*. For the two best-performing *mtry* values from the two best-performing *training set sizes* of the preceding stage, three different *minimum node sizes* were calculated (1 = default value), 5 and 10). The three plots on the left represent the three different *node size* values for a *training set size* of 60%, the three plots on the right display the three *node sizes* for a *training set size* of 70%. A more detailed version of these plots (with zoomed-in y-axes) is presented in figure 58 (see Appendix D).

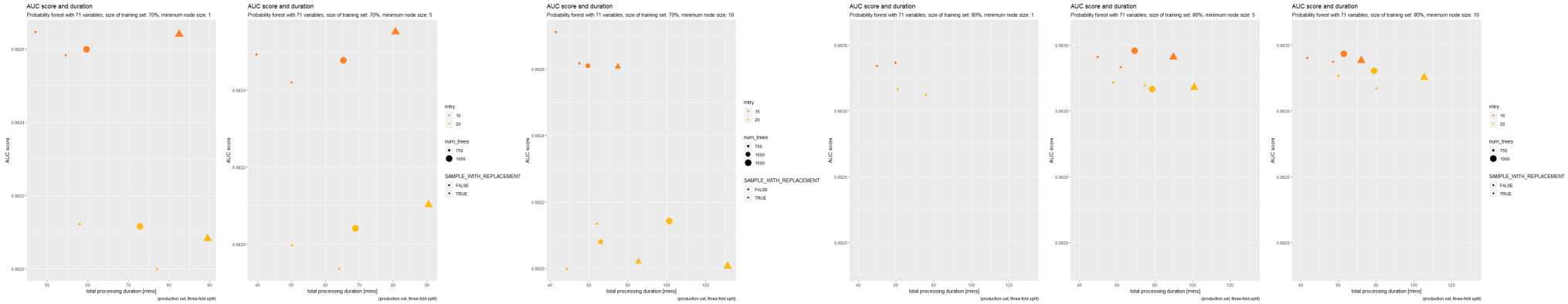

Figure 26: Tuning level 3: Performance of *probability forest* models created from the *production set*. For the two best-performing *mtry* values from the two best-performing *training set sizes* of the preceding stage, three different *minimum node sizes* were calculated (1, 5, and 10 = default value). The three plots on the left represent the three different *node size* values for a *training set size* of 70%, the three plots on the right display the three *node sizes* for a *training set size* of 80%. A more detailed version of these plots (with zoomed-in y-axes) is presented in figure 59 (see Appendix D).

5.4 Model testing

The final stage in a *supervised machine learning process* is the *testing phase* [42]. In this stage, the final models are *evaluated* with data that has neither been used in the process of *model training* nor in the process of *model tuning*. In order to accomplish this requirement, more observations had to be collected. In order to do so, the processes of *data collection* and *data preparation* (see sections 4.2 and 4.3) were repeated for *people registered with the AMS for at least one day within the year 2018*.[61] The final *evaluation data* comprised two sets, i.e. one for the *main set* (94 variables) and one for the *production set* (71 variables).[62] Each of the two sets contained 969,978 observations.

The three best-performing models for each *thread* (which had emerged from *model tuning*) were tested against unknown evaluation data.[63] In order for the results to be comparable, both the *geometric mean (GM)* and the *area under the ROC curve (AUC)* were calculated for all models. An additional benefit of having both performance measures at one's disposal was that their direct comparison allowed for observing the performance measures' behaviour regarding the order of the top 3 models.

Figures 27 and 28 display the evaluation results for the top 3 models of each *thread*. While the plots in figure 27 display the *AUC* value for all four *threads*, figure 28 shows the *geometric mean*:

[61] Similar to the data gathered for *model training* and *tuning* (where the base year was 2017), the *evaluation data* covered a *pre-career* of a length of two years (starting with the year 2016) and a *post-career* of a length of one year (covering the year 2019).

[62] In fact, in theory it would have been possible to additionally evaluate the *probability forests* with data from the year 2017, as the performed *three-fold split* has an inherent *test set* for evaluation purposes. However, due to insufficient comparability (between sets) or even lacking comparability (concerning *classification forests*), *model evaluation* was performed only for evaluation data of the year 2018. Nonetheless, visualisations of model performance for *probability forests* evaluated with data from the year 2017 are available in Appendix D.

[63] In practice, the models had to be re-calculated. The reason for this was that the number of models created up to this point was very high and that it was not possible to store temporary models on a larger scale.

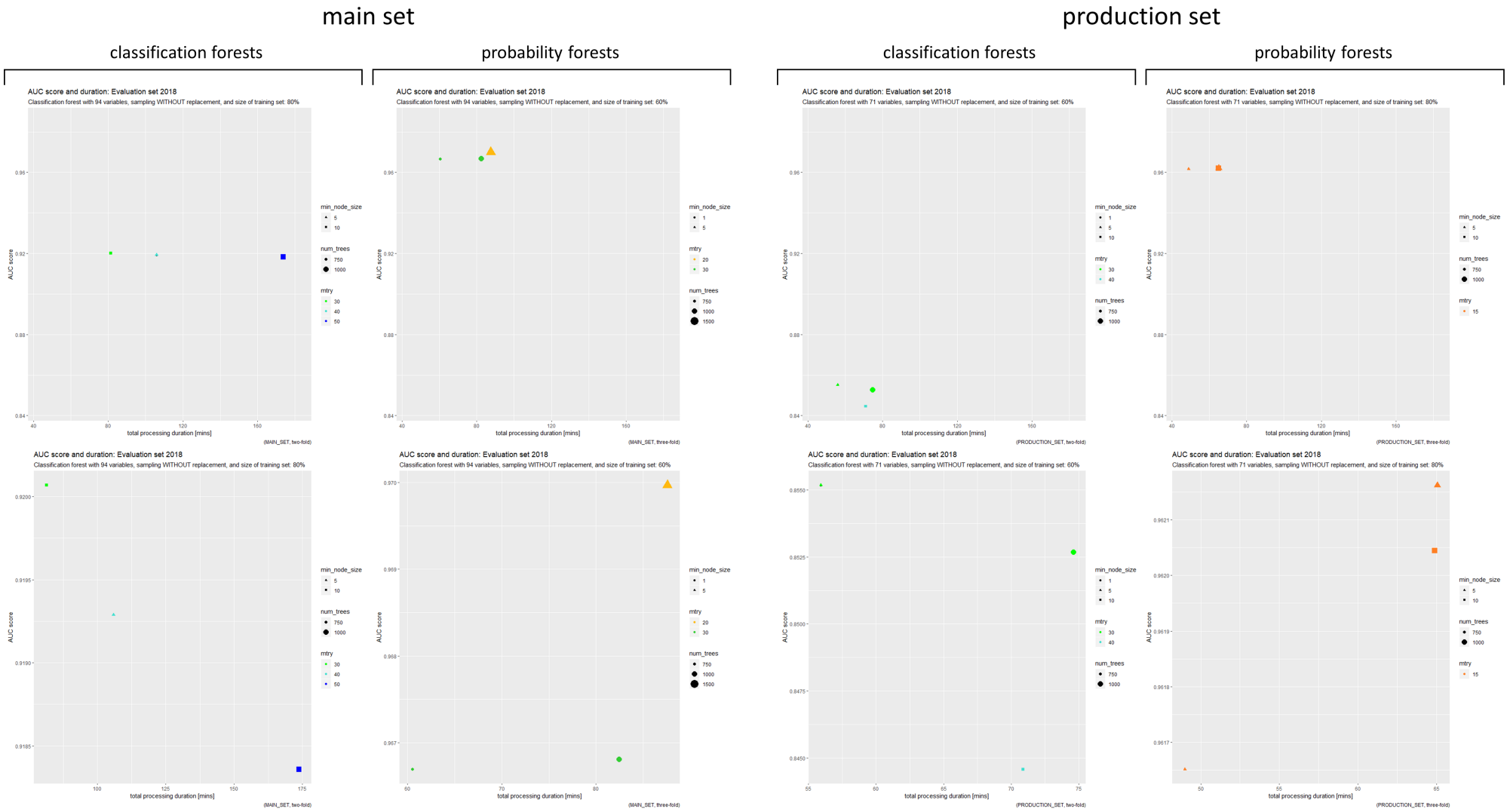

Figure 27: Evaluation of the top 3 models of all four *threads* (performance measure: *AUC*). The plots in the upper row share the same y-axes scale. The lower plots display the same models as the plots above, but with individually adjusted y-axes (i.e. the four plots are zoomed-in). The four plots on the left represent the *main set*, the four plots on the right represent the *production set*. The left-hand side of each block presents the performance of the *classification forests*, the right-hand side that of the *probability forests*.

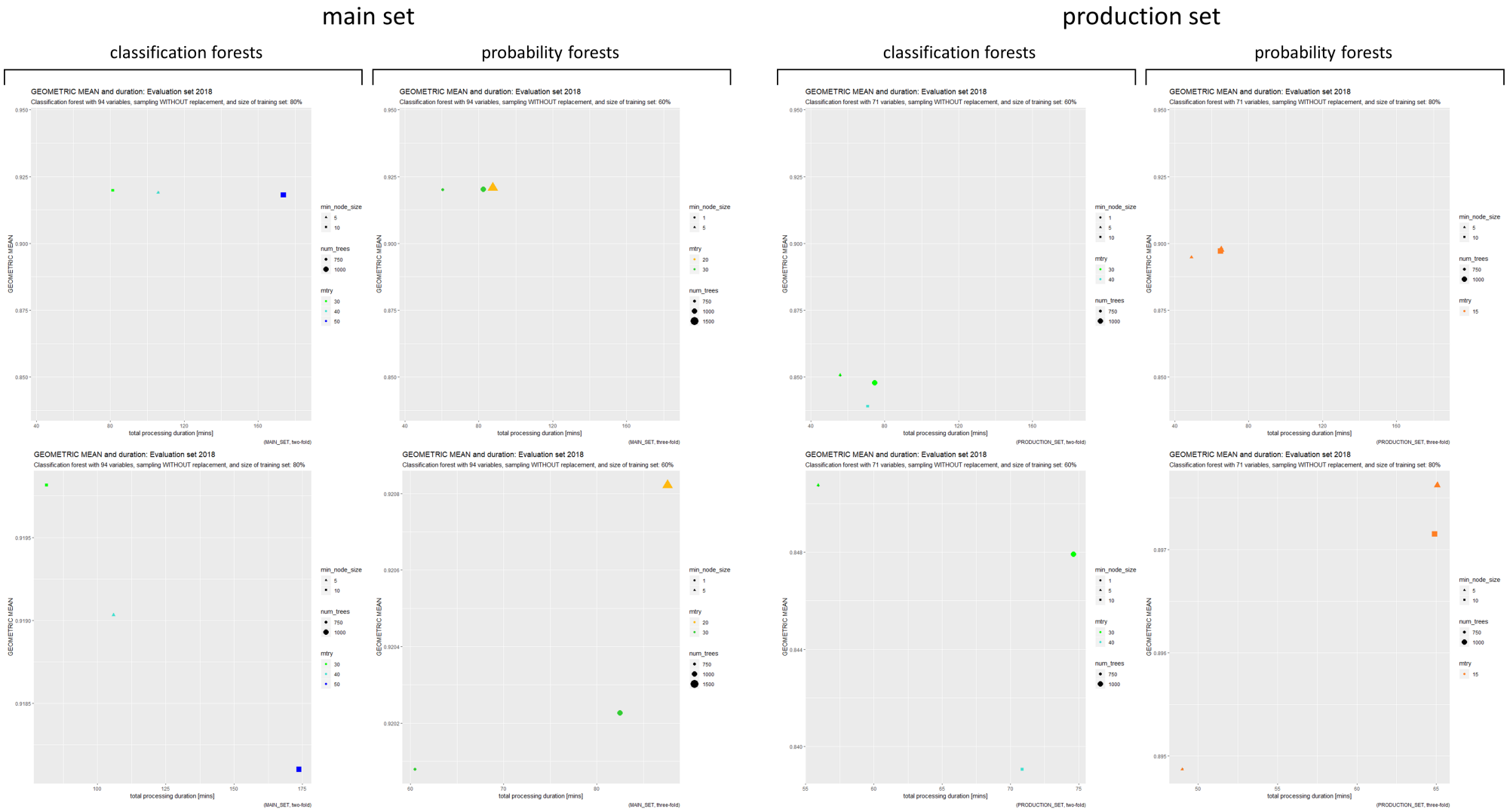

Figure 28: Evaluation of the top 3 models of all four *threads* (performance measure: *GM*). The plots in the upper row share the same y-axes scale. The lower plots display the same models as the plots above, but with individually adjusted y-axes (i.e. the four plots are zoomed-in). The four plots on the left represent the *main set*, the four plots on the right represent the *production set*. The left-hand side of each block presents the performance of the *classification forests*, the right-hand side that of the *probability forests*.

A quick glance back to figure 15 confirms that the *model generation process* has been completed. All the stages produced (intermediate or final) results. Figure 29 below presents the more detailed results obtained in the *model tuning* and *model testing* stages. From top to bottom, figure 29 shows the three *tuning levels* (see section 5.3) performed for all four *threads*. The configurations of (hyper-)parameters as well as the number of models generated are indicated by the infographics and the gear-icons, respectively. The three best-performing models per *thread* were chosen for final *evaluation*.

For both the *main set* and the *production set*, the best-performing models had *set sizes* of either 60% or 80%. Furthermore, in all cases *sampling without replacement* performed best. All other parameters, at a first glance, showed fewer obvious patterns. These results were passed on for their final *evaluation*, meaning their testing against data from the year 2018.

Depending on the performance measure used, two different rankings of the four final models emerged. In both cases, however, the *probability forest* created from the *main set* performed best, demonstrating a prediction quality of 97.0% (AUC) and 92.08% (GM). Furthermore, going by AUC, the *probability forests* outperformed the *classification forests*, and in terms of *geometric mean*, the *main set* outperformed the *production set*.

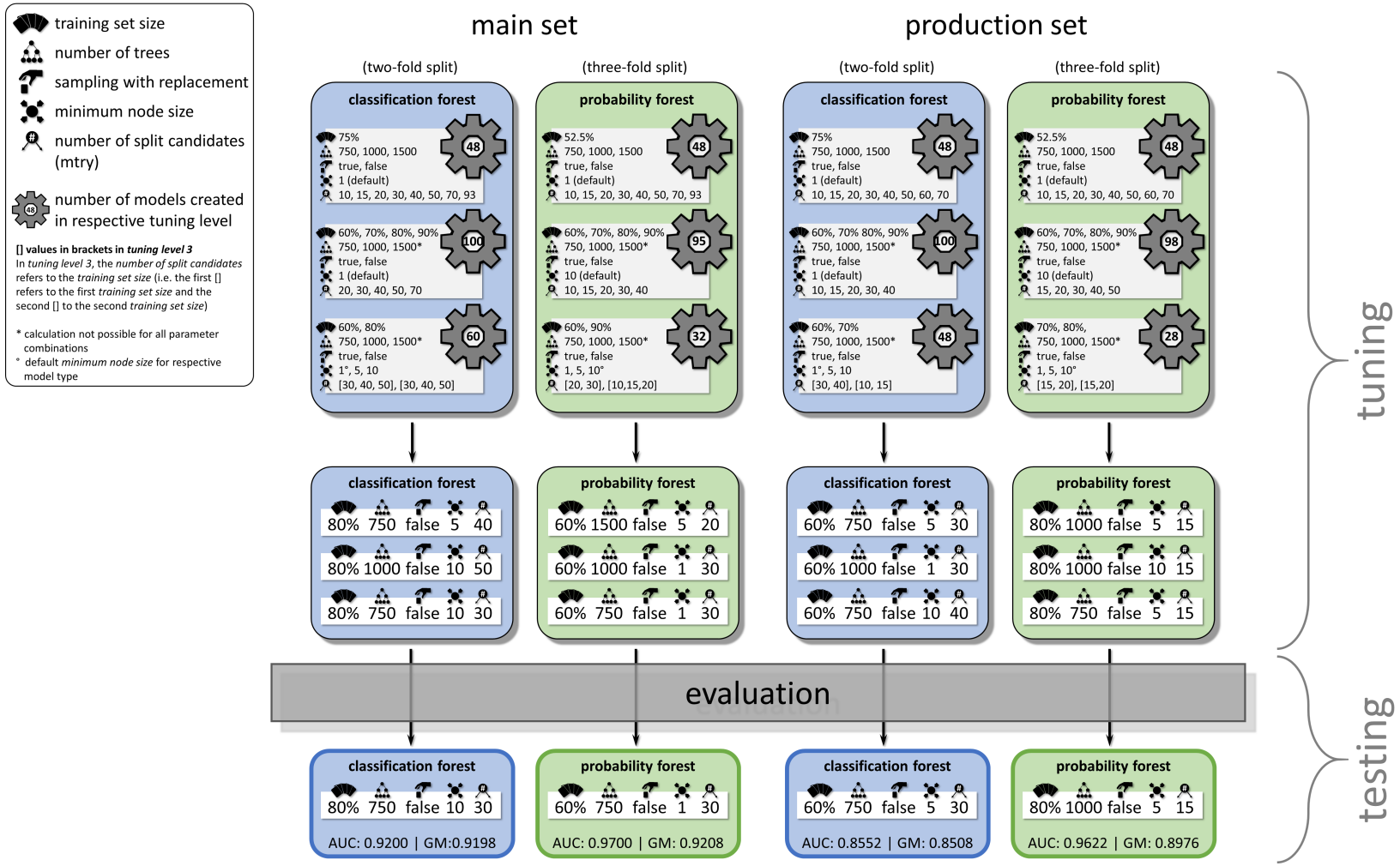

Figure 29: Overview of the results obtained in the *model tuning* and *model testing* stages. The blue and green boxes correspond with the four *threads* introduced in figure 15 (see section 5.1). In the *tuning stage*, each of the four *threads* underwent three *tuning levels* (indicated by the info-graphics and gear-icons). The numbers displayed in the gear-icons shows the number of models created in the respective *tuning level*. The three best-performing models were then chosen for *evaluation*. *Model testing* involved the comparison of all models' performance measures (*geometric mean*, AUC) with data from the year 2018. This resulted in one final model per *thread*.

6 Discussion

As stated in the introduction, the aim of this thesis was to create a *random forest machine learning model* to estimate the (re-)integration likelihood of unemployed people into the Austrian *labour market*. The data used for model building was drawn from the AMS database and the statistical software *R* was used as the environment for *data analysis* and *model building*. *Model training*, *tuning*, and *testing* were conducted with due regard to existing knowledge on generating a *random forest* model. The final outcome were four models which were each the best of their kind, i.e. the best-performing model for each combination of *data set* and *model type*.

Overall, the intended objective of building a useful model was achieved, however, not without compromises made along the way. These compromises primarily had to do with hard- and software limitations, which had an impact on specific aspects of the *data set* and *study design*.

In the following, this chapter first reflects on the *actual results* obtained for the models. It then goes into detail for noteworthy aspects and limitations concerning the *data set* and the *study design*.

6.1 Model results

The total data set for this study comprised 1,005,167 unique client observations and 195 variables. From this *basic set*, two separate sets were created, viz. the *main set* (94 variables) and the *production set* (71 variables), whereas the larger *main set* included additionally created variables. For each set, two different types of *random forest* were pursued, viz. *classification forests* and *probability forests*. Forest growing entailed *training*, *tuning*, and *testing* the models in order to be able to determine the best-performing model for each combination of *data set* and *model type*. In total, 765 models[64] (or 772,500 individual trees) were generated in a net time of 1,315.4 hours (or 54.8 days).[65]

In the following, this section first takes a closer look at the *overall performance* of the four final models. It then goes into further detail for the performance of the individual *(hyper-)parameters* used.

[64] 753 models were created within the stage of *model tuning* and an additional 12 models were (re-)calculated within the *testing* stage.

[65] The *net time* refers to successfully generated models only, meaning that besides the 765 models mentioned above, in fact many more were created. These models, however, were not documented. This is because their generation process, for reasons of memory overflow, usually failed at the time of calculating the *OOB-error*.

6.1.1 Overall performance

As previously mentioned, the study design allowed for receiving four final models. These models are illustrated in figure 30:

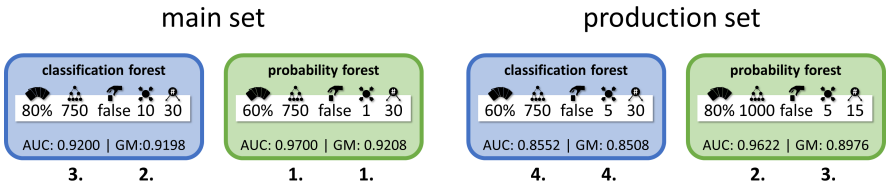

Figure 30: The four final models for both sets (*main set* and *production set*) and both model types (*classification forests* and *probability forests*). The models' performance ranking (1.-4.), which was derived from the respective *AUC* and *GM* values, is indicated below the model specifications.

Overall, a rather gratifying result is that all the final models showed a remarkably high level of performance quality (derived from the performance measures *AUC* and *GM*). The different performance values received were an invitation to rank the models, the result of which in figure 30 is marked below each model. While ranks 1 and 4 are the same for both performance measures, ranks 2 and 3 are reversed. The overall best-performing model, both in terms of *AUC* and *GM*, was the result of generating a *probability forest* from the *main set*. For this model, the prediction qualities resulted in 97.0% for the *AUC* and 92.08% for the *GM*.

Going by the ranking, two general statements can be made. First, the *main set* produced (slightly) better overall results than the *production set*, implying that the *additionally self-created variables* did indeed have a positive influence on model performance. Second, *probability forests* performed better than *classification forests*. As mentioned above, it was a probability model which took first place in terms of both *AUC* and *GM*, but also overall, the *probability forests* demonstrated a higher quality level.

The ranking also warrants a brief consideration from a practical perspective. As a rule, *probability forests* and forests created from the *production set* were less time-consuming than *classification forests* and forests created from the *main set*. In terms of the *model types*, the reason might be traceable back to the type of allocation decision the model is forced to perform, meaning that while *classification forests* had to make a strict binary allocation decision, *probability forests* had more freedom and allocated likelihoods to each class. Concerning the different sets, a very likely reason for this outcome is that because the *production set* contained fewer variables, less data had to be processed.

One of the most important features of this study was the choice of using two performance measures. The *AUC* and the *GM* are not comparable, as they dif-

fer in terms of how they determine a model's performance, meaning that while the AUC offers the result of the overall performance of every single decision tree within a *random forest* [35], the GM indicates the overall performance for an entire *random forest* model. As both approaches are valid, the question arises as to which performance measure is best used to communicate a model's final prediction quality. As shown in figure 30, while the higher AUC values are more attractive, the GM seems to be the more sensible choice, as it represents the percentage of correct classifications for the whole model.

Another aspect affirming the GM as the performance measure of choice becomes apparent when taking into account the range of the values exhibited for the GM and the AUC. While the GM values for the final models range from 85.08% to 92.08%, the AUC values range from 85.52% to 97.0%, which is a difference of 7% and 11.48%, respectively. Essentially, the GM is more suitable for the purpose of this study, as it indicates an *absolute* performance value, whereas the AUC would be favoured for studies interested in measuring the *relative* performance of two (or more) models [27, 34].

6.1.2 (Hyper-)parameter performance

(Hyper-)parameters were a central element of this analysis. Their particular combinations informed the *tuning stage* and, thus, which models would eventually be identified as the best-performing models. Overall, the *(hyper-)parameters* also revealed (more or less obvious) patterns. The remainder of this section takes a closer look at these patterns and seeks to offer interpretations and possible explanations, in particular since the performance of the *(hyper-)parameters* has to be interpreted and understood in the context of this thesis. The *(hyper-)parameters* listed below are discussed in the following:

- mtry
- sample replacement
- minimum node size
- training set size
- number of trees

Mtry. The *number of split candidates* (in short: *mtry*) was the central *(hyper-) parameter* for tuning the models. To recall, *mtry* was the prime interest in *tuning level 1* and acted as a gatekeeper for defining which model configurations would be considered for further analysis. The *mtry* settings for this analysis were 10, 15, 20,

30, 40, 50, 60, 70, and 93.[66] As depicted in figure 31, *mtry* 30 featured in three out of the four final models. Also, when taking into consideration the runners-up, the *classification forests* showed an incline towards higher *mtry* values (30, 40, and 50) while the *probability forests* slanted towards lower *mtry* values (15, 20, and 30):

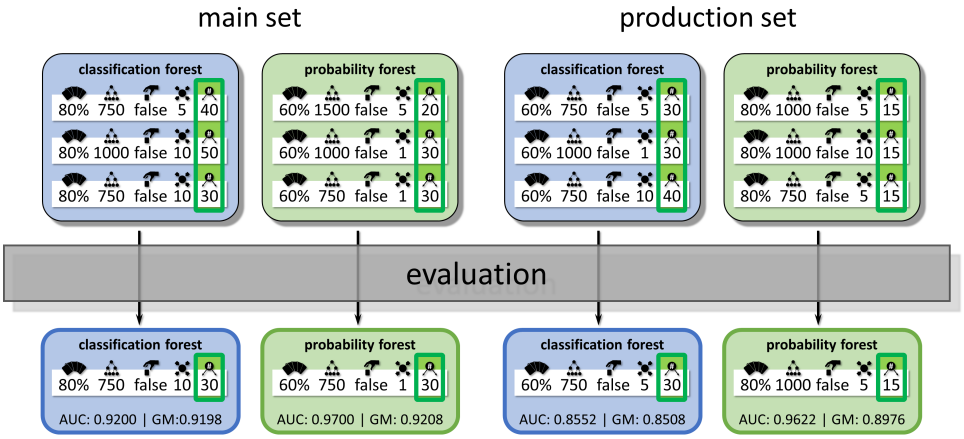

Figure 31: Best-performing models: *mtry*. *Classification forests* showed an incline towards higher *mtry* values (30, 40, and 50), while *probability forests* slanted towards lower *mtry* values (15, 20, and 30).

The exact reasons for these outcomes are difficult to pinpoint, as *mtry* is also dependent on other *(hyper-)parameter* settings. Yet, a general observation was that parameter settings which brought about higher complexity seemed to influence *mtry* performance (e.g. the influence of the *training set size* on the selection of the best-performing *mtry* values).

Another interesting observation was made when comparing the results with default settings proposed in the literature. Going by Probst et al., the default values would be 10 (*main set*) and 8 (*production set*) for *classification forests*, and 31 (*main set*) and 23 (*production set*) for *probability forests* [93]. When taking into account the best-performing models, only one model's *mtry* setting (viz. 30 for the *probability forest* in the *main set*) was similar to the proposed default setting of 31, while the others showed greater deviance. In fact, when taking into consideration the runners-up, the results rather suggest the opposite direction for promising *mtry* values (e.g. *classification forest* in the *main set*: default setting of 10 vs. results of 30, 40, 50). On the other hand, this study did not test specifically for the default settings, as the ambition was to investigate a wider range of *mtry* values. Consequently, the results obtained here might not be entirely comparable

[66]Not all *mtry* settings were applied to both *data sets*. While the values 10, 15, 20, 30, 40, 50, and 70 were used for both sets in *tuning level 1*, 60 was used only for the *production set* and 93 was used only for the *main set*.

with Probst et al.'s findings. Overall, however, these insights might nonetheless encourage tuning for this *(hyper-)parameter* in order to gain more insight in this matter.

Another noteworthy phenomenon observed had to do with how prone specific *mtry* values were to being aborted mid-way. For instance, when comparing different *mtry* values, as a rule, the higher the value, the longer the processing time. In terms of complexity (observed via memory overflows), however, the rule went: the lower the *mtry* value, the higher the likelihood of memory overflows. Interestingly, the greatest difficulties did not arise for the smallest value of 10, but for *mtry* 15. For instance, while models using *mtry* 10 and 20 rarely caused for memory exhaustion, *mtry* 15 often aborted model calculation (especially in combination with larger *training set sizes* and smaller *minimum node sizes*). One possible explanation might be that somewhere between the range of 10 to 20, there is a particularly fragile area within which variables with many expressions are over-preferred, causing for the creation of highly complex trees, which are simply not processable. This interpretation, however, would have to be verified by further research.

Sample replacement. The (hyper-)parameter *sample replacement* had two possible settings, viz. TRUE or FALSE. As illustrated in figure 32, there was a strong preference for the FALSE setting, which performed best in all the models generated:

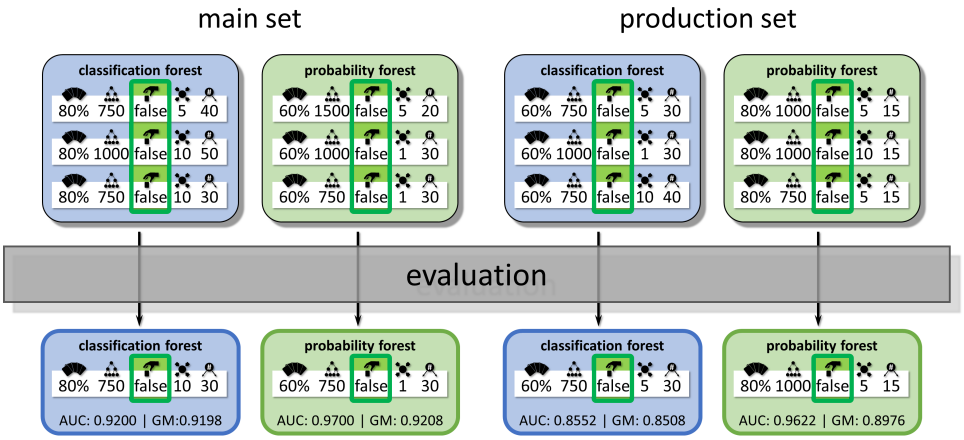

Figure 32: Best-performing models: *sample replacement*. *Sample replacement* = FALSE was the preferred setting for all the (best-performing) models.

A possible explanation for *non-replacement* performing better than *replacement* might be the varying number of category expressions for categorical variables [93], which might cause for the induction of artificial associations between variables [104]. Another explanation might be that, in this mode, particularities within a

training set were reinforced, for the simple reason that there was no possibility for an observation to be re-chosen. This, in turn, could have strengthened the model with regard to certain aspects, leading to a more consistent (and possibly less complex) forest showing a better overall performance.

Minimum node size. The *minimum node sizes* chosen for this analysis were 1, 5, and 10, whereas the smallest and the highest values represented default settings for certain *model types* (1: *classification forests*, 10: *probability forests* [116]). As depicted in figure 33, while no single *minimum node size* stood out, there were preferences for node sizes *within* the four *threads*:

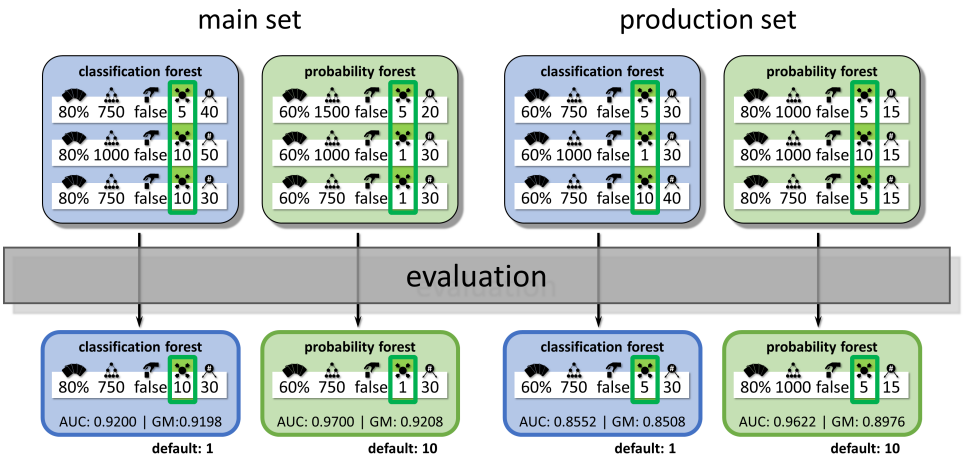

Figure 33: Best-performing models: *minimum node size*. None of the best-performing models and only a few of the runners-up exhibited the default setting for the *minimum node size*. The default setting for each forest type is indicated below each final model.

For instance, when taking a closer look at the top 3 models pertaining to the *main set*, the *minimum node sizes* tended to be lower for the *probability forests* (1, 1, 5) and higher for the *classification forests* (5, 10, 10). In contrast, concerning the *production set*, the *minimum node sizes* for the *probability forests* showed higher values (5, 5, 10), whereas the *classification forests* revealed a mixed bag of all possible *node sizes* (1, 5, 10).

A deeper understanding of these manifestations is only possible after more in-depth analysis. What can be said at this point, however, is that none of the best-performing models (and the majority of the runners-up, i.e. the second and third best-performing models for each *thread*) had been generated using the *minimum node size* default setting, which suggests that the modification of this parameter did have an influence on fostering better performance.

Training set size. The *training set sizes* determined for this thesis were 60%, 70%, 80%, and 90% (of the entire set size). As with all the other parameters, the intention of pursuing different settings was to find out if a particular *training set size* would perform better than the others. As can be seen in figure 34, the results show that *training set sizes* of 60% and 80% performed best for the final (and runner-up) models:

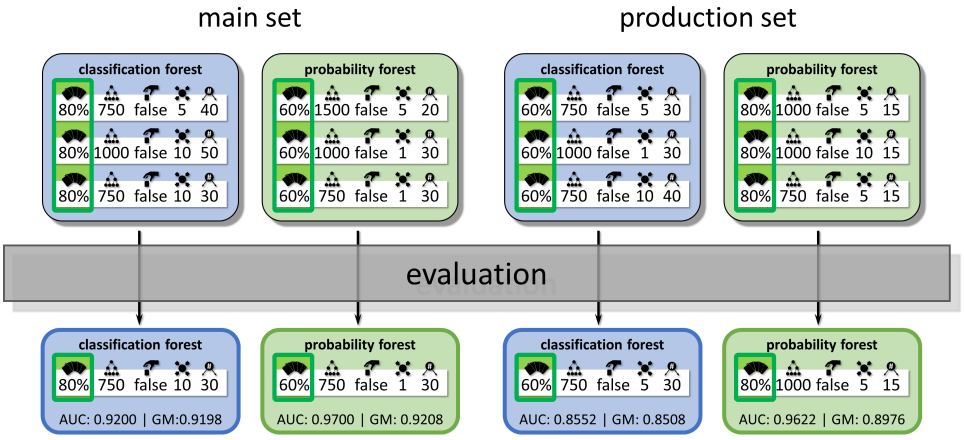

Figure 34: Best-performing models: *training set size*. *Training set sizes* of 60% and 80% performed best for the final models and the runners-up.

It can, however, also be seen that there is neither evidence that a certain *model type* favoured a particular *training set size* nor that either of the two settings (on the whole) outperformed the other (i.e. *training set size* of 60%: ranks 1 and 4; *training set size* of 80%: ranks 2 and 3).

Another thought emerges when taking into consideration the association between *training set size* and *model performance*. Probst et al. proposed that a smaller number of training observations leads to less correlated trees which, in turn, increases the overall prediction quality [93]. This might be interpreted as a linear relationship between an increase in performance quality and a decrease in *training set size*. In terms of this study, however, such a linear relationship was not observed. Instead, when taking a closer (albeit still superficial) look at the behaviours of the models for the different *training set sizes* (e.g. compare figures 19 and 20), a discernible pattern is that of a "wave" (with ups and downs) rather than a linear progression, which would imply that there are particular *training set sizes* which are more strongly influenced by other parameters than others. The quality of this hunch, however, can only be assessed after deeper analysis has been conducted.

Number of trees. The *number of trees* defines a forest's size and has an

influence on the forest's complexity. In this thesis, the three settings chosen for the *number of trees* were 750, 1,000, and 1,500.[67] These dimensions were determined in section 5.2, owing to the prerequisite that the *number of trees* has to be sufficiently high in order to be able to produce reliable results [18]. As shown in figure 35, the output for the four final models indicates a rather strong preference for the 750 setting, which locates this result within Probst et al.'s recommended range of 500 to 1,000 trees [93]:

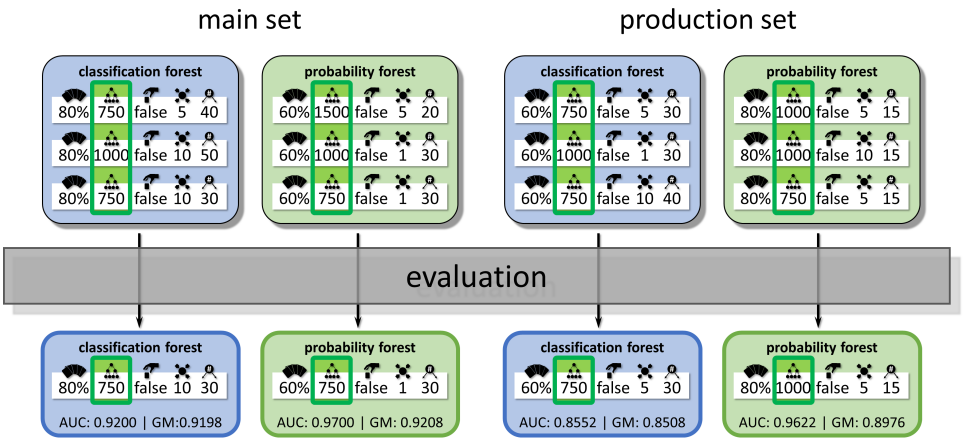

Figure 35: Best-performing models: *number of trees*. The lowest setting for the *number of trees* seemed to perform best. However, due to memory overflows, a large amount of *random forests* of a size of 1,500 trees was not computable.

However, it might not be wise to attach too much weight to this outcome, as in many cases, models with a very large *number of trees* were not computable. In total, 207 models were not processable, many of which pertained to the sample set sizes of 80% and 90% (in fact, not a single model with the 1,500 setting was computable for these set sizes). On the other hand, it may also be fair to say that, as observed during *model training* (see figures 16 and 17), when surpassing a certain *number of trees*, there was hardly any gain in prediction quality. This, in turn, would allow the outcome of this study to pass as reasonable.

6.2 Data set

From the onset of this thesis, it was of the utmost importance to work with a quality data set, as such a data set would allow for better understanding and

[67] The choice for these values was driven by the intention to gain the best results possible given the hardware available. It can, however, be assumed that a smaller *number of trees* (e.g. 500 trees) may also would have led to sufficient results.

interpreting the results. The final *data set* used for generating the *random forest* models consisted of 5 tables, 94 variables, and 1,005,167 observations.

The *data set* was also the result of a rather lengthy process. The reason for this was that obstacles occurred in all the common data-related stages (i.e. *data collection*, *preparation* and *analysis*). While some obstacles were to be reckoned with (e.g. variable omittance), others were less predictable (e.g. the complexity of the AMS database). Overall, the most noteworthy aspects concerning the *data set* are listed below and are presented in more detail in the following, whereas particular address is given to limitations (where applicable):

- table selection
- information omittance and aggregation
- missing values replacement
- "perfect" results

Table selection. The data used for this thesis originated from the AMS database, and although all the relevant variables were stored within this database, the selection process was a major venture. This was mainly due to the almost impenetrable nature of this database. The reason for this was the vast amount of data tables, which (more often than not) contained (intentional) redundancies. Furthermore, some of the tables were mere copies of *production tables*, which simply had not been maintained after duplication (or, as encountered on several occasions, triplication). In addition, conventions for table naming and table types were not consistently applied.

As stated in section 4.2.1, the initial idea for selecting a set of tables containing relevant variables for this study was a *criteria-led* table selection process (including several iterative rounds of table examination). This approach, however, failed for different reasons (e.g. inconsistent table naming, lack of information on data lineage or master data management) and was, after several months and countless hours of scrutinous table examinations, abandoned and replaced by an *experience-based* selection process (based on AMS practitioner knowledge).

While table selection was tedious, a highly gratifying fact is that the AMS has developed both awareness and acknowledgement of these database issues. As a consequence, incidentally, the AMS is currently investing resources in revising its database.

Information omittance and aggregation. Due to hardware and software

restrictions[68], large parts of the efforts taken in the course of this thesis dealt with reducing the amount of available information. One aspect in this context was the diminution of the set of variables used for model building. The exclusion of variables occurred at several stages throughout this thesis (e.g. during *data collection*, *data analysis*, and even *model training*). Reasonable omission of data, however, demanded for knowledge about the variables, which led to a rather in-depth and complex investigation of the data. In some cases, however, the contents of variables, or even of whole data tables, were simply not determinable. Indeterminate information, as a rule, was omitted, as data revealing no information (neither on its origin nor on its content) was considered futile. On top of that, other reasons for omittance were, for instance, information irrelevant in the context of this thesis (e.g. meta data), data subject to data protection rules (e.g. a client's date of birth or social security number), or inconsistent information (e.g. the variable name not reflecting its actual content).

Another aspect related to reducing the amount of variables concerned the aggregation of variables. Aggregation of this kind involved the creation of the variable `FUNDING_RECEIVED`, which served as a container for 16 funding-related variables. The creation of the variable `FUNDING_RECEIVED` was a direct consequence of the memory issues caused by the hardware limitations. Regrettably, this aggregation represents a considerable compromise in terms of not being able to include more detailed funding-related data into the model as, essentially, a wide range of funding-related variables were reduced to a single binary variable which (merely) indicated whether or not a client had received a funding measure within his or her *pre-career*. So, while this aggregation was inevitable, it also has to be said that it prevented from eliciting a more fine-grained differentiation on the effectiveness of certain funding measures, an area which is of particular interest to the AMS.

The third and final noteworthy aspect in this context is that of observation aggregation. Observation aggregation was performed in the course of creating individual *pre-careers*. To recall, *pre-careers* represent historical data, i.e. data observations for different variables dating back two years prior to the client's last *report date* (see section 4.1). However, instead of retaining all available observations recorded within the *pre-career*, memory restrictions only allowed for using

[68]Hardware limitations were the result of data protection regulations. Since the process of client support largely depends on person-bound information (data directly linked to individual clients), the AMS' data protection officer prohibited the migration of data to other environments. Consequently, the undertakings of this thesis were largely determined by the hardware performance provided by the AMS (see Appendix E, table 5). Software limitations had to do with particularities concerning *R* in general, and *R*'s *ranger* package in particular. A commonly known issue in the context of processing large data sets with *R* is the fact that *R* needs all data to be loaded into memory in order for it to be calculated (e.g. [108, 51]). Moreover, *R*'s *ranger* package requires the data for model building to be stored in a single variable (e.g. a data frame).

aggregated historical information. In practice, this meant that for each client, only the observations recorded at the last *report date* were extracted and that each of these observations were then supplemented with historical information in the form of aggregated aspects (such as the number of days in different employment statuses, or the number of job aspirations expressed by a client and recorded by the AMS). Although parts of the historical data were preserved via aggregation, the choice of the variables to be aggregated was highly selective and limited to only a subset of the available information (as only the most promising variables were aggregated).

While reducing the number of observations per client to one single line meant that there would be a certain information loss, there was also an advantageous side-effect in that it was not necessary to balance out the number of observations per client and, hence, in the model, each client was considered with the exact same weight. Furthermore, the reduction to one observation per client implied a substantial reduction of redundancies within the *data set*. More precisely, while up to 24 observations per client were possible (i.e. one observation for each month of the *pre-career*), the majority of the client information did not (or did only slightly) change over time (e.g. person-bound information, such as a client's sex or education). Retaining all observations for each client would have led to vast amounts of redundant information.

In sum, it is to be expected that every omittance or aggregation of information comes with a certain loss of information. However, the nature of the data (e.g. indeterminate information), but also the circumstances of this study (e.g. hard- and software limitations) necessitated making such choices, the documentation of which is fully provided in this thesis (see Appendix C, table 4). Overall, it was possible to reduce the initial amount of data to a *data set* processable with the hardware provided. A special case of necessary variable omittance is discussed under the aspect of "perfect" results.

Missing values replacement. Missing data is a problem with many *machine learning methods* [102, 107], because data sets containing missing values (in most cases) can not be processed. For this reason, *missing values replacement*, which is typically performed with *value imputation* (e.g. [102, 82]), has become an important preparatory step in the context of *data preparation*. In the data used for this thesis, missing values emerged for different reasons. One reason was traceable back to the method of data recording during the AMS' client support process. The software applications the AMS uses for client support contain a number of fields for obligatory client information, such as a client's name, date of birth, or residence. Missing values for these entries are not allowed [72] and are in fact technically impossible, as a blank is prevented by the application's design. Other

information, however, which is not mandatory, does not have to be provided and therefore can be the cause for missing values.

Another reason for missing values occurred during *data preparation*, more precisely in the course of joining the information of several data tables. To recall, this step was performed because, in the AMS database, different types of information are stored in different tables (see section 4.2.1). The emergence of missing values can be exemplified well when taking a closer look at the joining of funding data (stored in the table `FOERDERUNG_INT`) and the study sample data (stored in the table `MON_UNI_STATUS_INT`). As it appeared, not all clients were granted funding measures. In fact, only every other client (about 50%) received extra support of this kind (see dropbox link [11]). Consequently, by joining the funding data table with the study sample table, approximately every second observation returned a missing value.

Regardless of the reason, *random forest* does not allow for missing values in the data used for model building. Although there are several approaches for *missing value imputation* (e.g. [102, 107, 82]), *automated value imputing* was not an option for this thesis. The reason for this lay in the fact that the absence of information itself may have carried information. Concerning the AMS data, occurrences of this phenomenon were found for both *numerical* as well as for *categorical variables*. In terms of *numerical variables*, for instance, missing data in the context of a client's benefit receipt can indicate a client's lack of entitlement for benefit receipt. In this context, *automated value imputation* would most likely have produced a wrong value, especially if the imputed value had depended on comparable observations within the *data set* (as would be the case with *Strawman imputation* or *Proximity imputation* [107]).[69] In fact, any value other than the variable's blank setting in this case can be considered wrong and the only correct replacement for a missing value is 0 (as, on the one hand, any other value would represent a monetary benefit received and, on the other hand, character-based placeholders would not be a viable option for this type of data). Consequently, this proxy value ("0") was used for all *numerical variables*.

In terms of *categorical variables*, the situation was slightly different, as the AMS provides not only category codes and labels (i.e. expression names) for variables, but also definitive default values.[70] Thus, these default values were used to replace

[69] Tang and Ishwaran describe different value imputation methods, such as *Strawman imputation* or *Proximity imputation*. In terms of *Strawman imputation*, missing values for numerical variables are replaced by the median value of the non-missing values and missing values for categorical variables are replaced by the most frequently occurring non-missing value [107, p. 2]. *Proximity imputation*, on the other hand, builds upon the values imputed by *Strawman* and refines these values [107]. While these two methods are quick to use, they tend to be imprecise.

[70] This information is stored in so-called *dimension (DIM) tables*. DIM tables serve as deposits of codes for *categorical variables* which feature in the context of the AMS' Data Warehouse.

the missing values. For instance, for the variable `WIEDEREINSTIEG`, which indicates if a client is re-entering the *labour market* after a longer absence (e.g. after parental leave), the AMS defines the two possible values "W" (for re-entry) and "-" (other). The reason why missing values occur is because the variable `WIEDEREINSTIEG` is not a mandatory variable in the AMS' software application, which is why it often is left blank (unless a person is a returnee). The correct code for "non-returnees" would be "-", which, however, never occurred in the data. Again, this example demonstrates that *automated variable imputation* would never have been able to guess the correct code, regardless of the level of sophistication of the imputation process. Consequently, as correct values could not be obtained via *automated value imputation*, missing values were replaced manually.[71]

"Perfect" results. The issue of overly perfect results occurred when running the first series of *random forests* with the first set of variables (which was small enough to be processable with the hardware at hand).[72] This first series of *random forests* exhibited an almost perfect *OOB-error* rate of approximately 0.0031 as well as an AUC score of 1 for *random forests* comprising only 10 trees.[73] In the light of existing benchmarks, the results obtained were too good to be true. Consequently, this outcome demanded for deeper analysis of the interrelationships between the variables.

In order to gain information on these relationships, the *numerical variables* underwent additional scrutiny. This meant creating a *logistic regression* "mini-model" for each *numerical variable*, which comprised only one *numerical variable* and the categorical *target variable* (`SUSTAINABLE_EMPLOYMENT`). The intention was to find the model that minimised the AIC value[74] and, hence, revealed the *numerical variable* featuring the strongest relationship with the *target variable*. The variable identified (`UNINTERRUPTED_EMPLOYMENT_DAYS`) was originally created to determine whether a client had entered into a sustainable employment relationship within the *post-career*. As such, it constituted an illegitimate proxy variable to the *target variable*.

[71] Initially, *automated value imputation* was thought to be an option. The tool used was *R*'s *missRanger*-package [73]. While *missRanger* was easy to implement, it turned out that *automated value imputation* was simply not an option for the data at hand.

[72] This first set comprised 115 variables. Also, at the time of calculating this first series of *random forest* models, only machines (a) and (b) (see Appendix E, table 5) were available, whereas only machine (b) was capable of calculating the *random forests*.

[73] According to Matty, models showing AUC scores between 0.8 and 0.9 are considered "excellent" in terms of class discrimination, and AUC scores of 0.9 and more are considered "outstanding" [70, p. 30]. Concerning the *OOB-error*, the lower the value, the better.

[74] *Akaike's Information Criterion (AIC)* is a measure to compare the parsimonious fits of several models [101]. According to Chen et al., the AIC "selects the model that minimizes a sum of the residual deviance plus a penalty term times a measure of model complexity" [29].

Apart from the illegitimate proxy, two other future-related variables (`FIRST_BENR_AFTER_REG` and `SAME_EMPLOYER_AFTER_REG`), which both dealt with *post-career* employment relationships, became manifest. After excluding the proxy and the future-related variables, the models' scores decreased to more realistic values of approximately 0.98 (AUC) and 0.04 ($OOB\text{-}error$). Satisfyingly, these "new" results could still be classified as having "outstanding" discrimination capabilities [70].

Overall, while strong results are typically a reason for celebration, the aforementioned extra lap which was taken in the course of this thesis might serve as a good example for cases in which results are overly perfect and, therefore, require closer inspection as to the reasons of this outcome. At least for this thesis, it can be said that this on-top procedure was indicative of revealing how a handful of well-intended, self-created variables did not serve, but in fact counteracted model building. It should also be pointed out that the AIC proved to be a useful tool for tackling the issue of uncovering relationships between variables of different types.

6.3 Study design

Random forest is a method which offers a variety of opportunities for its implementation. It is possible, for instance, to create different *types of forests*, to apply different *split regimes*, and to choose from among different *evaluation strategies*. The ambition for this thesis was to create an interesting study design, which combines the most suitable and promising *random forest* elements for addressing the research interest at hand. These decisions also had to be made bearing in mind the aforementioned hard- and software limitations.

All in all, while the study design met the intended goal, it also reflects a constant trade-off between model complexity, processability, and comparability. The most important aspects in this context are listed below and are described in more detail in the following, again drawing attention to limitations (where applicable):

- classification forests and probability forests
- fixed assignments of split regimes and evaluation strategies
- three-tier tuning process
- (hyper-)parameter constrictions
- variable importance computation
- overfitting

Classification forests and probability forests. The *random forest* method offers the possibility to create *classification forests*, *regression forests*, or *probability forests*. Given that the research interest of this thesis was to estimate the (re-)integration likelihood, the natural first choice was to build a *probability forest* (since a *probability forest* predicts the degree of an observation's class affiliation in the form of class probabilities). This thesis, however, also pursued the creation of *classification forests*, meaning forests which estimate an observation's class assignment. Although in the context of *labour market monitoring*, the mere classification of clients without providing the respective class probabilities is of little avail, the expectation towards *classification forests* was that they would be a means for a fast preliminary classification of observations.

Unfortunately, these expectations had to be dismissed as, in general, the calculation of the *classification forests* took longer than the calculation of *probability forests* under mostly similar *(hyper-)parameter* configurations. To illustrate this, while the generation and evaluation of the average *probability forest* took 90.3 minutes (or 5.57 seconds for an individual tree), it took the average *classification forest* 114.3 minutes (or 6.68 seconds for an individual tree). Nonetheless, as *classification forests* were part of the study from the beginning, they remained part of it throughout the entire process (and essentially also contributed towards showing interesting results).

Fixed assignments of split regimes and evaluation strategies. Another facet in terms of the design of this study stemmed from the deliberation to pursue variation also in terms of *split regimes* and *evaluation strategies*. Consequently, the idea was to use a fixed combination of elements which would then be assigned to the *model types*. In short, the *two-fold split* and the *geometric mean (GM)* were assigned to *classification forests*, the *three-fold split* and the *area under the ROC curve (AUC)* were assigned to *probability forests*. Although additional assignments (e.g. *geometric mean* for *probability forests*) would have (in theory and technically) been perfectly possible, the time and effort needed to tune and describe the different models (in particular in terms of the iterative model generation process) would have blown this thesis out of proportion. For instance, the entire net calculation time (excluding re-runs, etc.) for all models was 1,315.4 hours (or 54.8 days). Had all possible combinations been pursued, calculation time would have at least doubled.

The use of these fixed assignments led to a lack of comparability in the *model training* and *tuning* stages. Yet, since the three best-performing models of all four *threads* (see figures 15 and 29) had to be re-calculated anyway (as limited hard disc space did not allow for saving the models for later use), this circumstance was used as an opportunity to calculate both *evaluation measures* (*GM* and *AUC*) for both

classification forests and *probability forests*. Consequently, in the *model testing* stage (which produced the final results), the models (in the end) were comparable.

Another issue regarding comparability arose in terms of the usage of the *three-fold split*. In general, a *three-fold split* appears suitable in cases in which data for evaluating the model is limited. The advantage of the *three-fold split* is that a subset of the data collected (the *test set*) is used for *model evaluation* so that additional testing data is not necessary. From the onset of this thesis, the intention was to use a separate (additional) data set for *model evaluation* - a choice which, strictly speaking, renders the concept of the *three-fold split* redundant. However, the ambition was also to learn about the application of the *three-fold split* in this context and (possibly) to receive interesting insights. These, however, were limited, as the *test sets* of the different models were not comparable (due to the different *training set sizes*).

Nonetheless, although the initial plan of using different *evaluation strategies* for the two different *model types* did not work out as intended, it can be said that both *evaluation strategies* led to conclusive results in terms of prediction quality. Thus, both performance measures have their raison d'être.

Overfitting. As mentioned above, model performance in this study was assessed with the *GM* and the *AUC*, whereas the direct comparison of all models was only possible in the last *tuning level*. Initially, the ideal study design foresaw direct comparability on all *tuning levels*. This would have been achievable by using the *OOB-error*, a performance measure which (conveniently) is automatically calculated each time a *random forest* is grown. However, it can be assumed that when relying solely on the *OOB-error*, overfitting is likely, given that overfitting is understood as the over-adjustment of a model towards its *training data*, the result being that while prediction quality increases for familiar *training data*, it decreases for unfamiliar *testing data* [32].

In this thesis, while the *OOB-error* was not used for the actual analysis, its values were, nonetheless, readily available. Thus, out of curiosity, these values were compared to the *GM* and *AUC* scores of the first *tuning level*. The outcome of this comparison is visualised in figure 36:

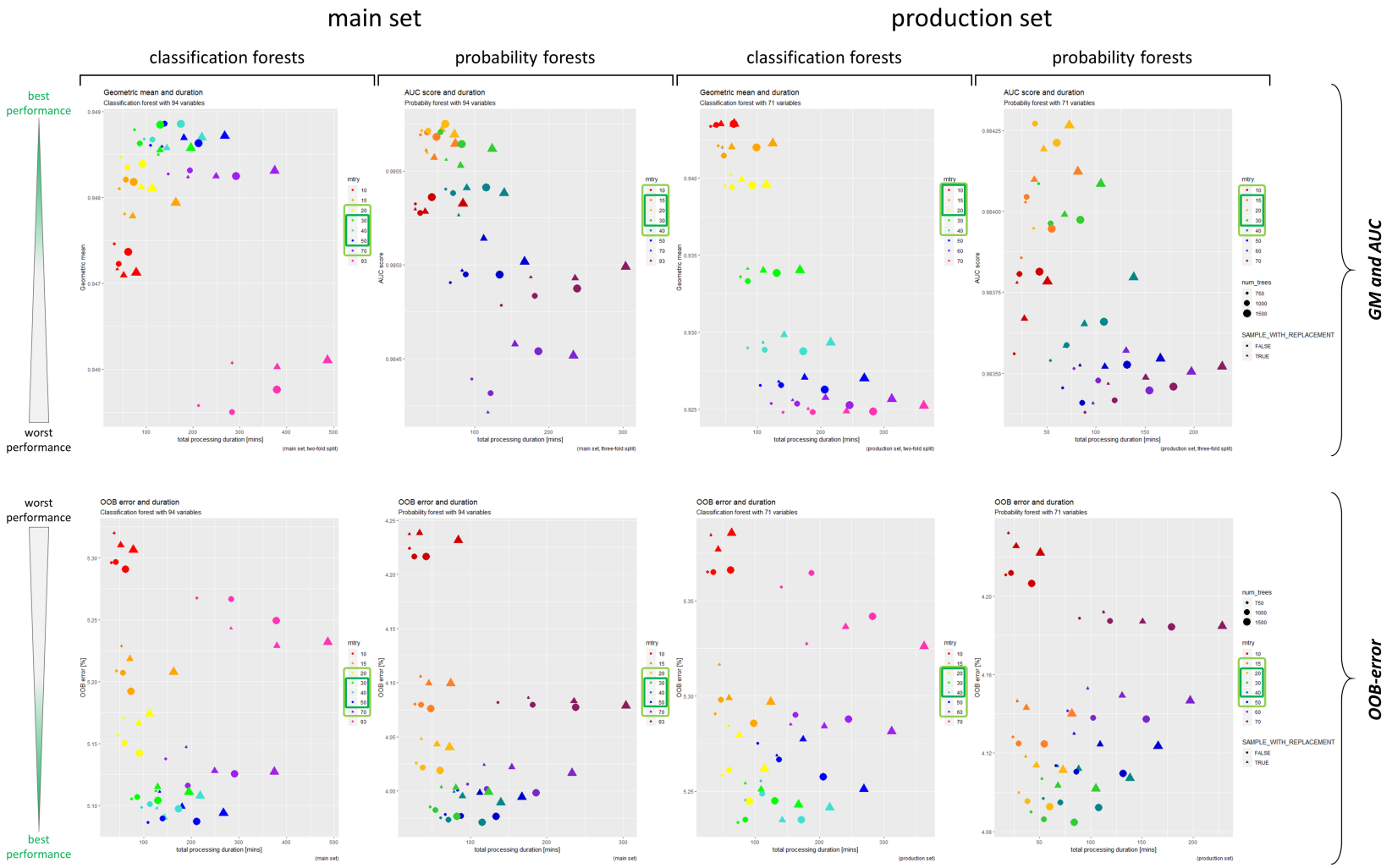

Figure 36: Comparison of the performance measures *GM*, *AUC*, and *OOB-error*. The plots show the results of *tuning level 1* for all four *threads* (see figure 15). While the upper four plots represent the results for *mtry* values obtained via *GM* (for *classification forests*) and *AUC* (for *probability forests*), the lower plots show the same models queried for the *OOB-error*. In the upper row, the best-performing models are the ones located at the very top of the y-axes, in the bottom row, the best-performing models are the ones situated at the lower end of the y-axes. The top 5 and the top 3 *mtry* values are highlighted by a light green and a dark green box, respectively. A direct comparison shows that the *OOB-error* tends to favour higher *mtry* values.

While the upper four plots show the results for *tuning level 1* obtained via *GM* and *AUC*, the lower four plots show the results for *tuning level 1* from an *OOB-error*'s perspective. As a reminder, the intention on *tuning level 1* was to discern the best *mtry* values in order to pass them on to the next *tuning level*. The top 5 and the top 3 *mtry* values are highlighted by a light green and a dark green box, respectively. The comparison is always made for the plot above with the plot below. Overall then, apart from the left-most plots (for which the choice of performance measure did not seem to make a difference), in all other cases the *OOB-error* tended to favour higher *mtry* values. For instance, while the *probability forest* created for the *main set* and using the *AUC* score led to *mtry* values of 15, 20, and 30, the *OOB-error* would have led to *mtry* choices of 30, 40, and 50.

All in all, while the *OOB-error* would have been the practical and straightforward choice, it was decided to use performance measures which appear less prone to overfitting (as these performance measures are generated with unfamiliar data). Also, on a side note, while the mini-analysis presented above cannot be considered evidence for overfitting, it does show that the choice of performance measure (at least in the course of this study) could have made a difference.

(Hyper-)parameter constrictions. One of the study design components directly affected by hardware limitations were the *(hyper-)parameters* used for model building. To recall, these *(hyper-)parameters* were *mtry*, the *training set size*, the *sample replacement strategy*, the *minimum node size*, and the *number of trees*. Overall, it was not possible to run all parameters at their maximum capacity. In other words, some configurations of parameters, in particular such which had an influence on the complexity of the *random forest*, were only possible in a reduced manner. The *number of trees* in a *random forest*, for instance, is a driver for the model's complexity and at the same time is limited by available main memory. Consequently, the maximum value for this parameter used for model generation was restricted to a maximum of 1,500 trees. Furthermore, additional *(hyper-) parameter* configurations which fostered a model's complexity (e.g. a small value for the *minimum node size*) caused for the *number of trees* having to be reduced even further in order for the models to be processable. For instance, although it was perfectly possible to generate *random forests* with 1,500 trees at a *minimum node size* of 10, it was not possible to create a *random forest* of equal size at a *minimum node size* of 1. In conclusion, then, the more a parameter affected the *random forest*'s complexity, the smaller the maximum *number of trees*.

Possibly, these parameter constrictions might not have materialised (to this extent) if, for instance, the final *set sizes* for the *main set* and the *production set* had been smaller (i.e. if they had contained fewer variables) and thus would have freed up memory space. However, the objective was to make use of as much

relevant data as possible, which meant including as many variables as possible. Consequently, the final sets used for model building were created to meet the joint goal of exploiting both the given data as well as hardware potential.

Variable importance computation. Another constraint caused by limited hardware had to be faced during *model training* when seeking to determine *variable importance*. *Variable importance* is a measure which reveals a variable's contribution to a model's prediction accuracy [18]. It is useful for increasing the quality of a data set, as less relevant variables can be dropped (which is a particularly beneficial side-effect when faced with hardware limitations).

Since memory restrictions did not allow for calculating Altmann's *permutation importance (PIMP)* [3], the method employed was Breiman's *permutation importance*, provided in *R*'s *ranger* package [18, 117].[75] This method, however, tends to overrate the importance of variables with more expressions [104, 3, 93]. Due to this bias, although the ranking of the variables (according to their *permutation importance* values) appeared reasonable (see figures 50 and 51), the actual ranking might not be fully trustworthy. Nonetheless, the exclusion of variables with negative importance values led to a slight increase in model performance (see figures 16 and 17). Thus, to a certain extent, this improvement in model performance can be interpreted as a rationalisation for the application of this *variable importance* method.

Three-tier tuning process. *Model tuning*, in the course of this study, was performed in a step-wise manner. The hardware limitations meant that instead of generating models with all possible combinations of *(hyper-)parameter* configurations and then picking out the best-performing model for final evaluation, tuning was a piecemeal task, resulting in the development of a *three-tier tuning process*. In the course of this tuning process, the focus of each level lay on a specific *(hyper-) parameter*, while the other parameters remained fixed (see section 5.3).

Although this *three-tier tuning process* produced satisfactory outcomes (in terms of prediction qualities), this approach bears the risk of producing non-optimal models. As already stated in section 5.3, the reason for this lies in the assumption that the best-performing models on one level would still be the best-performing models on the subsequent level which, however, constituted a completely different interplay of parameter configurations. For instance, the results for *tuning level 2* (see figures 52 to 55) show that the *training set size* had a major influence on the order of the best-performing *mtry* values. Considering this, the question arises whether or not *mtry* values that had not been passed over to *tuning*

[75] PIMP builds upon the results generated with Breiman's *permutation importance* and corrects the importance values of over-favoured variables by calculating corrected p-values [3].

level 2 might actually have perform better than they had in the preceding *tuning level 1*.

For the models created in this thesis, no evidence for this phenomenon was found. Figures 37 and 38 below serve in support of this assessment, as they show that an additional *mtry* value of 50, which initially had not been considered for *tuning level 2*, when passed over to *tuning level 2* did not produce better results than the initially selected *mtry* values. It is also important to note that this finding has no effect on the quality of the decisions made for selecting the two best-performing *mtry* values as input for *tuning level 3*.

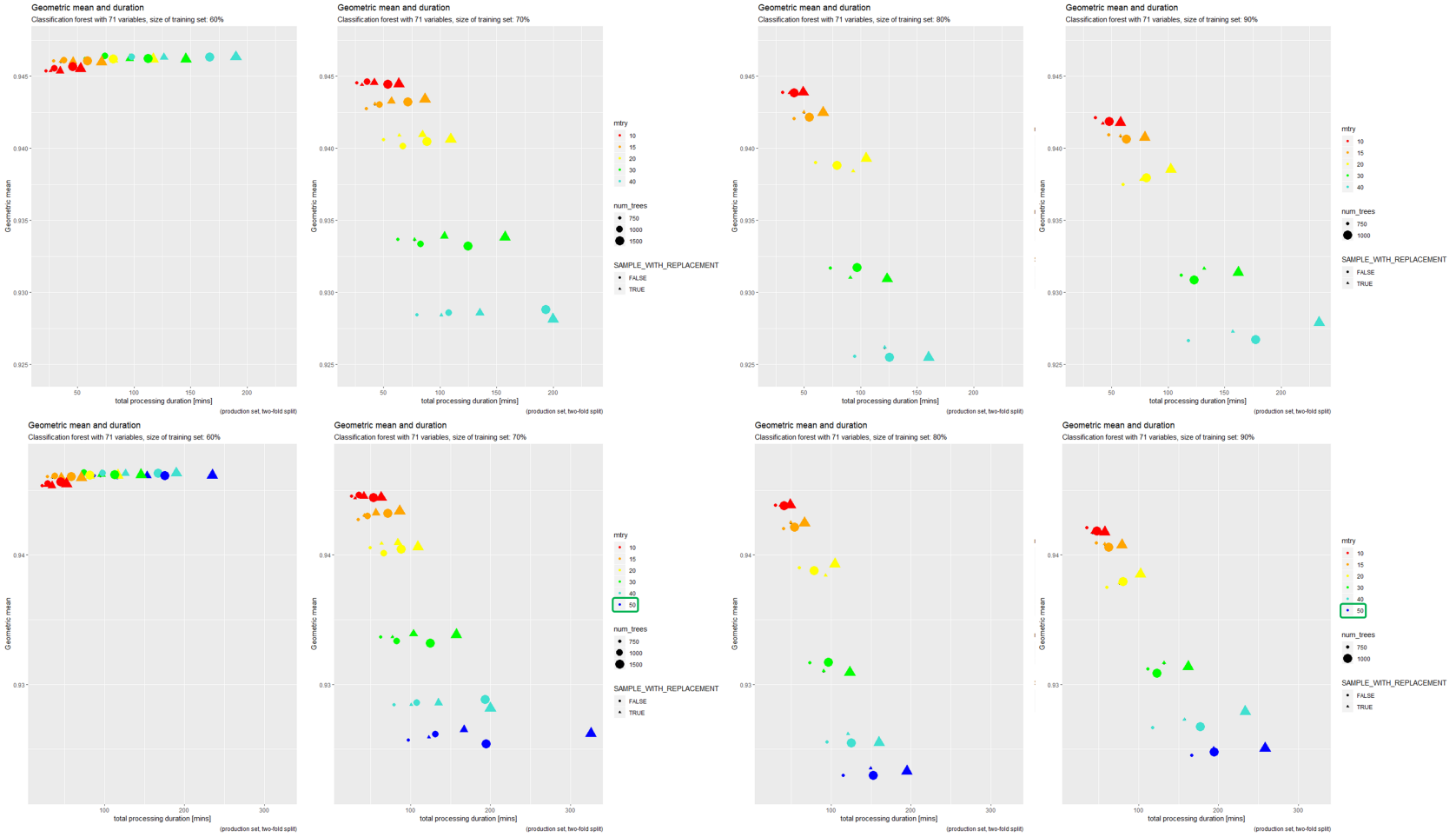

Figure 37: Comparison of results of *tuning level 2* for all *training set sizes* with five (above) and six (below) *mtry* values for *classification forests* created from the *production set*. The additional *mtry* value 50 (dark blue) initially was not selected for *tuning level 2*. For the *training set sizes* 70%, 80%, and 90%, the additional *mtry* value performed least well, for the *training set size* 60% its performance was average. However, only the two best-performing mtry values were passed on to the next level, which in the cases displayed above would not have included *mtry* 50. Zoomed-in variants of the lower plots with six *mtry* values are presented in figure 62 (see Appendix E).

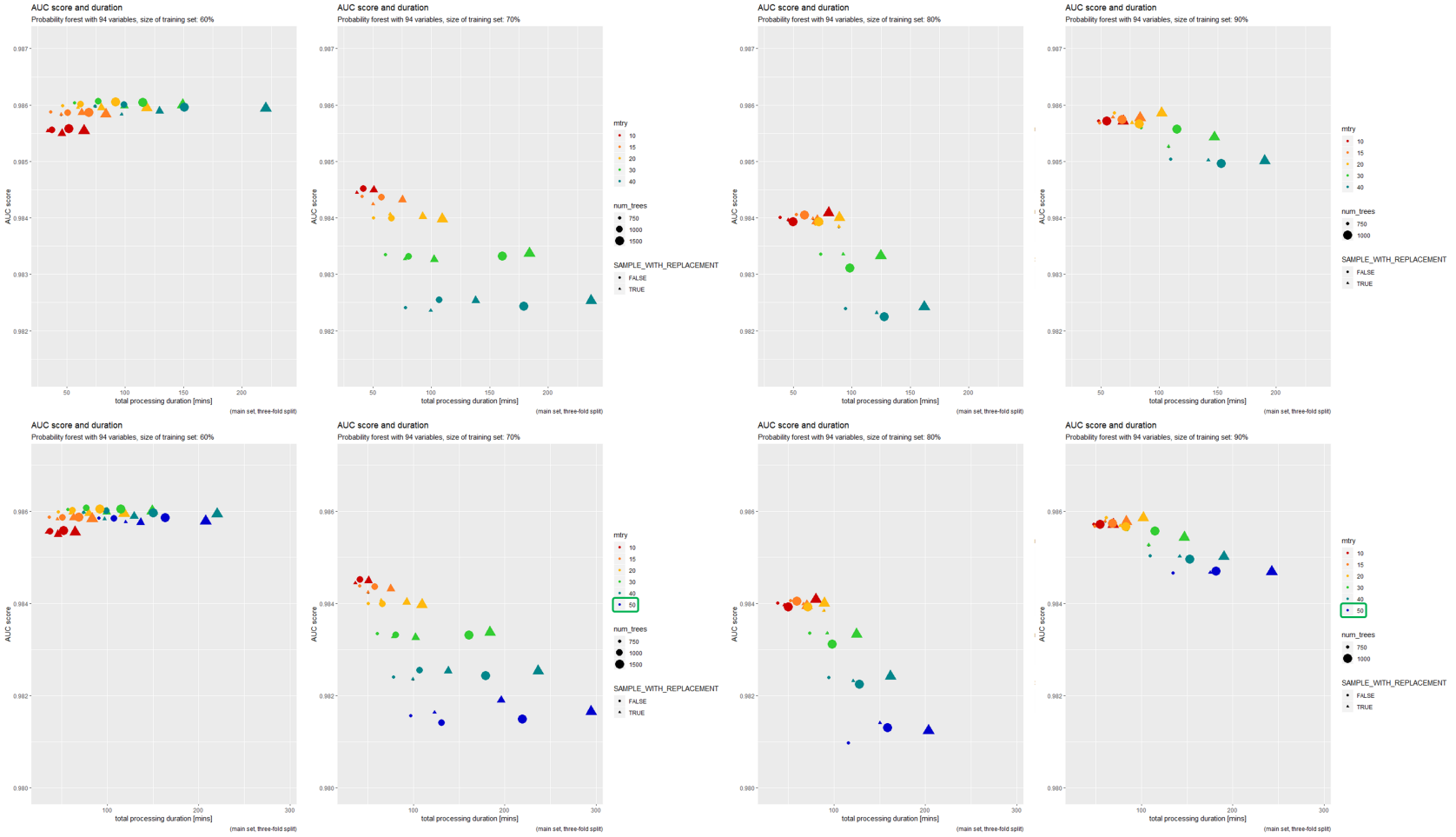

Figure 38: Comparison of results of *tuning level 2* for all *training set sizes* with five (above) and six (below) *mtry* values for *probability forests* created from the *main set*. The additional *mtry* value 50 (dark blue) initially was not selected for *tuning level 2*. For the *training set sizes* 70%, 80%, and 90%, the additional *mtry* value performed least well, for the *training set size* 60% its performance was average. However, only the two best-performing mtry values were passed on to the next level, which in the cases displayed above would not have included *mtry* 50. Zoomed-in variants of the lower plots with six *mtry* values are presented in figure 63 (see Appendix E).

7 Conclusion

The intention of this thesis was to create a *random forest* model which is able to estimate the *(re-)integration likelihood* of unemployed people into the Austrian labour market. A *random forest* is a *decision tree*-based ensemble method of *supervised machine learning* which was introduced by Leo Breiman in 2001 [18] and which can be used to predict different types of *target variables* [93, 69]. An interesting characteristic of *random forest* is its ensemble nature: The use of a set of *decision trees* enables for averaging out possible data set particularities and thereby makes the model relatively resistant to overfitting. At the same time, the ensemble nature is responsible for *random forest*'s impenetrability in terms of the straightforward determination of a variable's importance [18]. Although *random forests* are frequently used in a wide range of fields, the method has so far rarely been used in the context of *labour market* applications (compare [31]).

The data used for model creation and evaluation was drawn from the AMS' internal database and included administrative data as well as insurance data gathered in the context of client support. From an initial *basic set* of variables, which included 1,005,167 individual client observations and featured 195 variables, two different data sets were derived, viz. he *main set* with 94 variables and the *production set* with 71 variables, whereas the former comprised additional historical client information. This historical information was included by means of the *pre-career*, which was defined as a time-span of two years prior to a client's last status (i.e. a client's last *report date* recorded with the AMS). As retaining and processing the entire *pre-career* information was not possible (due to hardware limitations), parts of the historical information were preserved in the form of aggregated data (e.g. as the sum total of unemployment episodes).

Two different types of *random forest* were created from the two *data sets*, viz. *classification forests* and *probability forests*. The *split regimes* used were a *two-fold split* for *classification forests* and a *three-fold split* for *probability forests*. Model performance was assessed via the two evaluation measures *geometric mean (GM)* and *area under the ROC curve (AUC)*, and similar to the *split regimes*, these performance measures were fixedly assigned to the *model types* (*GM* for *classification forests* and *AUC* for *probability forests*). The reason for the fixed assignments was entirely pragmatic and was based on the hardware limitations encountered during the entire process. Although a complete coverage of both *split regimes* and both *performance measures* for both *model types* would have been perfectly possible, the time and effort needed to create and maintain the resulting amount of models would have gone far beyond the scope of this thesis.

The main result of this study were four final *random forest* models (the best-performing model for each combination of *data set* and *model type*) of considerable

predictive quality. The best model, and thus the model considered most suitable for estimating the (re-)integration likelihood into the Austrian labour market, was a *probability forest* created from the *main set*, which exhibited a precision score of 92.08% in terms of *GM* and 97.0% in terms of *AUC*. Although both performance measures are perfectly valid in terms of assessing a model's predictive quality, the *AUC* is recommended for querying *relative* comparisons of model performance (as opposed to *absolute* model performance) [27, 34], which consequently means that, in the context of this thesis, the *GM* appears to be more suitable for communicating the degree of predictive quality.

The remaining three best-performing models also appear to have their benefits. The *probability forest* created from the *production set*, for instance, can be put to use in situations in which no historical information is available. An example for such a situation is the Corona-virus pandemic, which incidentally emerged at the time of writing this thesis and which severely impacted *labour markets* all over the world [114]. From a data perspective, however, this event does not have an impact on the basic set-up of the *production set*. In fact, taking into consideration a time lag (which is necessary for defining the *target variable* and for retrieving *report date*-related data), the model could be put to use within five months.[76]

The *classification forests*, on the other hand, which although not suitable for estimating the likelihood of (re-)integration into the labour market, are usable in a multi-class context. An interesting area of application for the AMS would be for estimating the most suitable funding measure for an individual client. In this thesis, funding-related variables had to be bundled into a single variable (`FUNDING_RECEIVED`). The unravelling of this variable would allow for more fine-grained insights into funding measures. Furthermore, the exploitation of *classification forests* for multi-class problems, which can be put to use for measuring the effectiveness of funding measures, is likely to allow for an increased differentiation in terms of the allocation of monetary resources.

Overall, the results revealed that the models created from the *main set* generally performed better than the models created from the *production set*. As the *main set* comprised additionally created historical information, this is a strong indication that the self-calculated additional *pre-career* indeed had a positive impact on performance levels. In terms of model types, *probability forests* generally performed better than *classification forests*. Essentially, these insights are also reflected by the overall best-performing model, in that this model was a *probability forest* created from the *main set*.

[76] For instance, if an employment relationship started on the first day after the lock-down ended (e.g. 1 May 2020), this employment would be considered sustainable on the 63^{rd} day (i.e. 2 July 2020). This data would be available within the AMS database in the middle of September, at the earliest (which is due to *report date* and latency aspects).

Limited hardware played a major role throughout this thesis. The *tuning process*, in particular, was subject to significant hardware limitations, which led to the development of a *three-tier model tuning process*. In each of these three levels, another *(hyper-)parameter* was tuned, with the *mtry* value functioning as the gatekeeper for passing on the best-performing model to the subsequent tuning level. The *(hyper-)parameters* exhibited interdependencies, as the performance of a model created with a certain parameter setting was dependent on the settings of the remaining *(hyper-)parameters*. Different settings for the *training set size*, for instance, led to different performances of *mtry* values. Overall, what can be said is that some *(hyper-)parameters* performed as expected (i.e. as described in the literature), while others did not. The *sample replacement strategy*, for instance, was perfectly in line with theory, as sampling *without replacement* consistently performed better than sampling *with replacement*. The reason for this was a high variability in the number of variable expressions [93]. The same is true for the *number of trees*, for which a further increase in the *number of trees* resulted in ever-decreasing performance gains, while the computation time increased [93]. In terms of the remaining *(hyper-)parameters*, viz. *mtry*, the *training set size*, and the *minimum node size*, tuning is highly recommended, in particular for such cases in which default settings are available (e.g. *minimum node size* and *mtry* [93, 116]), since - at least in this thesis - the default values rarely formed part of the best-performing models' *(hyper-)parameter* settings.

Overall, these results ought to be treated as tentative until more research has been conducted to identify the individual parameter contributions. In this thesis, it was mainly the problem of limited hardware which hindered such queries. What was observed, however, is that hardware issues became particularly salient for cases in which parameter settings fostered highly complex trees, meaning a combination of a large *number of trees* with a high *training set size* and small values for *node size* and *mtry*. Thus, if more memory were available, more parameter combinations would be computable, which in turn would provide for more in-depth knowledge on parameter relationships. Furthermore, higher computational performance would allow for the integration of further information (such as non-standardised data stored in the AMS software applications' text boxes, a client's click-behaviour in the context of using the AMS' vacancies-related self-service tools, or information on a regional *labour market*'s structure). Such additions could make for a richer model and could be utilised for providing tailor-made client support.

The use of *machine learning methods* for the allocation of AMS resources (regardless of the resources being of a monetary- or of a service-nature) implies discrimination of clients and bears the risk of poor decision-making. As decisions in the AMS context of providing benefits and services directly and substantially affect

the lives of individual clients, the quality and correctness of the data and models used is of the utmost importance. Although providing monetary and support services on an individual client level is the AMS' daily business, *machine learning tools*, such as the application presented here, are to be understood as a means for supporting the decision-making process - and not as a means for replacing the well-founded decision-making abilities of an experienced consultant.

Finally, this thesis was intended to contribute to the AMS' current efforts in advancing its *labour market monitoring* toolbox by providing an (adaptable) prediction model. On a wider scale, this thesis also sought to contribute to the scientific community in terms of showcasing the application of *random forest* for the *target* of *(re-)integration likelihood* in the context of a national *labour market*. Finally, by providing an example of employing *random forest* in a computationally low-end environment, within which it was nonetheless possible to maintain high levels of prediction quality, this thesis is hopefully a useful contribution to the practitioners' community in terms of demonstrating the capabilities of the methods and tools used.

References

[1] Aho, Alfred V. and Hopcroft, John E., and Ullman, Jeffrey D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing, Boston, MA, USA, 1st edition, 1974.

[2] Jehad Ali, Rehanullah Khan, Nasir Ahmad, and Imran Maqsood. Random Forests and Decision Trees. *International Journal of Computer Science Issues*, 5(3):272–278, September 2012.

[3] André Altmann, Laura Tolosi, Oliver Sander, and Thomas Lengauer. Permutation Importance: A Corrected Feature Importance Measure. *Bioinformatics*, 26(10):1340–1347, April 2010.

[4] Arbeitsmarktservice Österreich, Bundesgeschäftsstelle. Längerfristiger Plan des Arbeitsmarktservice Österreich für die Planungsperiode 1997-1999, 1997. Internal document.

[5] Bruno Arpino, Marco Le Moglie, and Letizia Mencarini. Machine-Learning Techniques for Family Demography: An Application of Random Forests to the Analysis of Divorce Determinants in Germany. Technical report, Universitat Pompeu Fabra, Barcelona, Spain, April 2018.

[6] Eva Auer, Nadine Grieger, and Iris Wach. Arbeitslosigkeit & Arbeitsmarktbeobachtung, January 2018. Internal document.

[7] Yael Ben-Haim and Elad Tom-Tov. A Streaming Parallel Decision Tree Algorithm. *Journal of Machine Learning Research*, 11:849–872, March 2010.

[8] Simon Bernard, Laurent Heutte, and Sébastien Adam. On the Selection of Decision Trees in Random Forests. In *2009 International Joint Conference on Neural Networks*, pages 302–307, June 2009.

[9] Simon Bernard, Laurent Heutte, and Sébastien Adam. A Study of Strength and Correlation in Random Forests. In *International Conference on Intelligent Computing*, pages 186–191, August 2010.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[11] Christian Bliem. Dropbox link for figures in section 4.4.1 *Data description and visualisation*: https://www.dropbox.com/sh/7w6o3q0205tp6gn/AABB1Xb6goy4Bw08iZTnFNr4a?dl=0, 2020.

[12] Chester I. Bliss. The method of PROBITS. *Science*, 79(2037):38–39, January 1934.

[13] Julia Bock-Schappelwein, Stefan Fuchs, Ulrike Huemer, Regina Konle-Seidl, Helmut Mahringer, and Thomas Rhein. Aktive und passive Arbeitsmarktpolitik in Österreich und Deutschland - Aufkommen und Verwendung der Mittel im Vergleich. Technical report, Arbeitsmarktservice Österreich, Vienna, Austria, 2014. Study conducted by the Austrian Institute of Economic Research (WIFO) on behalf of the Public Employment Service Austria.

[14] Roberto Boselli, Mirko Cesarini, Fabio Mercorio, and Mario Mezzanzanica. Labour Market Intelligence for Supporting Decision Making. In *Proceedings of the 25th Italian Symposium on Advanced Database Systems*, June 2017.

[15] Andrew P. Bradley. The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognition*, 30(7):1145–1159, July 1997.

[16] Leo Breiman. Bagging Predictors. Technical report, Department of Statistics, University of California, Berkely, CA, September 1994.

[17] Leo Breiman. Technical Note: Some Properties of Splitting Criteria. *Machine Learning*, 24(1):41–47, July 1996.

[18] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[19] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[20] Bundeskanzleramt. Nationales Reformprogramm 2019 und länderspezifische Empfehlungen der Europäischen Kommission. Technical report, Federal Chancellery of Austria, Vienna, Austria, June 2019.

[21] Bundesministerium für Arbeit, Soziales, Gesundheit und Konsumentenschutz. Arbeitsmarktpolitische Zielvorgaben, Stand: Februar 2019. Technical report, Federal Ministry of Labor, Social Affairs, Health and Consumer Protection, Vienna, Austria, February 2019.

[22] Bundesministerium für Arbeit, Soziales und Konsumentenschutz. Arbeitsmarktpolitische Zielvorgaben 2010, Stand: Dezember 2017. Technical report, Federal Ministry of Labor, Social Affairs, and Consumer Protection, Vienna, Austria, December 2017.

[23] Wray Buntine and Tim Niblett. A Further Comparison of Splitting Rules for Decision-Tree Induction. *Machine Learning*, 8(1):75–85, January 1992.

[24] Andreas Buzek, Hannes Edlinger, Claudia Friedenthal, Klaus Hochrainer, Franz Schmitzberger, Brigitte Tauer, and Manfred Zauner. Arbeitsmarkt Monitoring mit dem Data Warehouse des Arbeitsmarktservice. Technical report, Federal Ministry of Economic Affairs and Labor, Vienna, Austria, October 2003.

[25] Andreas Buzek, Hannes Edlinger, Claudia Friedenthal, J. Ernst Oberklammer, Franz Schmitzberger, Barbara Zajic, and Manfred Zauner. Arbeitsmarktmonitoring 2004 mit dem Data Warehouse des Arbeitsmarktservice. Technical report, Federal Ministry of Economic Affairs and Labor, Vienna, Austria, December 2004.

[26] Danilo Bzdok, Martin Krzywinski, and Naomi Altman. Machine Learning: Supervised Methods. *Nature Methods*, 15:5–6, January 2018.

[27] Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An Empirical Evaluation of Supervised Learning in High Dimensions. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*, pages 96–103, New York, NY, USA, 2008.

[28] Robin L. P. Chang and Theodosios Pavlidis. Fuzzy Decision Tree Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(1):28–35, January 1977.

[29] Chun-houh Chen, Wolfgang Karl Härdle, and Antony Unwin. *Handbook of Data Visualization*. Springer-Verlag Berlin, Heidelberg, January 2008.

[30] Vincent Chiew. A Software Engineering Cognitive Knowledge Discovery Framework. In *Proceedings of the 1st IEEE International Conference on Cognitive Informatics (ICCI 02)*, pages 163–172, February 2002.

[31] Sam Desiere, Kristine Langenbucher, and Ludo Struyven. Statistical Profiling in Public Employment Services. *OECD Social, Employment and Migration Working Papers*, 2019.

[32] Thomas Dietterich. Overfitting and Undercomputing in Machine Learning. *ACM Computing Surveys*, 27(3):326–327, September 1995.

[33] Thomas Dietterich and Eun Bae Kong. Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms. Technical report, Department of Computer Science, Oregon State University, 1995.

[34] Rogério Espíndola and Nelson Ebecken. On Extending f-measure and g-mean Metrics to Multi-class Problems. *WIT Transactions on Information and Communication Technologies*, 35:25–34, January 2005.

[35] Tom Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[36] Marcelo Fernandes. F-Beta Score: http://www.marcelonet.com/snippets/machine-learning/evaluation-metrix/f-beta-score, December 2017. Accessed: 24 July 2019.

[37] Michael Friendly. Visualizing Categorical Data: Data, Stories and Pictures. In *Proceedings of the SAS User's Group International Conference*, pages 889–897, 2000.

[38] Anton Gerunov. Big Data Approaches to Modeling the Labor Market. Technical report, Munich Personal PepEc Archive, November 2014.

[39] Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabricio Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive Random Forests for Evolving Data Stream Classification. *Machine Learning*, 106(9):1469–1495, October 2017.

[40] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. Correlation and Variable Importance in Random Forests. *Statistics and Computing*, 27:659–678, October 2013.

[41] Isabelle Guyon. A Scaling Law for the Validation-Set Training-Set Size Ratio. Technical report, AT & T Bell Laboratories, Berkeley, CA, 1997.

[42] Peter Harrington. *Machine Learning in Action*. Manning Publications Co., Greenwich, CT, USA, 2012.

[43] Hans Hauska and Philip Swain. The Decision Tree Classifier - Design and Potential. Technical report, Lulea University of Technology and Purdue University, Lulea, Sweden and West Lafayette, Indiana, February 1975.

[44] Eva Heckl, Daniela Hosner, and Karin Petzlberger. Unterstützungsleistungen für langzeitbeschäftigungslose Personen in ausgewählten europäischen Ländern. Technical report, KMU Forschung Austria, Vienna, Austria, December 2018.

[45] Jürgen Holl, Günter Kernbeiß, and Michael Wagner-Pinter. Das AMS-Arbeitsmarktchancen-Modell. Technical report, Synthesis Forschung, Vienna, Austria, October 2018. Internal document.

[46] Waltraute Hopfgartner. Dokumentation über bundeseinheitliche Statistiken der AMV, October 1988. Internal document.

[47] Waltraute Hopfgartner. Jahresbericht '89 und arbeitsmarktpolitisches Schwerpunktprogramm '90, April 1990. Internal document.

[48] Waltraute Hopfgartner. Jahresbericht '90 und arbeitsmarktpolitisches Schwerpunktprogramm '91, April 1991. Internal document.

[49] Waltraute Hopfgartner. Jahresbericht '91 und arbeitsmarktpolitisches Schwerpunktprogramm '92, May 1992. Internal document.

[50] Waltraute Hopfgartner, Reinhard Pipal, and Renate Hahn. Jahresbericht '88 und arbeitsmarktpolitisches Schwerpunktprogramm '89, April 1989. Internal document.

[51] IntelliPaat. SAS versus R: https://intellipaat.com/blog/sas-versus-r/, July 2016. Accessed: 18 May 2020.

[52] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, New York, 2014.

[53] Zachary Glen Jones and Fridolin Linder. Exploratory Data Analysis Using Random Forests. Technical report, 73rd Annual MPSA Conference, April 2015.

[54] Michael W. Kearney. Cramer's v. In M. R. Allen, editor, *The SAGE Encyclopedia of Communication Research Methods*. Sage, Thousand Oaks, CA, 2017.

[55] Neeraj Khadilkar and Deepali Joshi. Predictive Model on Employability of Applicants and Job Hopping Using Machine Learning. *International Journal of Computer Applications*, 171(1):37–41, August 2017.

[56] Isabella Klingsbichl and Katarina Pisskova. AMP Ziele 2019, March 2019. Internal document.

[57] Christopher Krauss, Xuan Anh Do, and Nicolas Huck. Deep Neural Networks, Gradient-boosted Trees, Random Forests: Statistical Arbitrage on the S&P 500. Discussion papers in economics, Friedrich-Alexander University Erlangen-Nürnberg, Institute for Economics, March 2016.

[58] Ottilie Kritsch. Jahresbericht 1981, May 1982. Internal document.

[59] Ottilie Kritsch and Norbert Auzinger. Jahresbericht 1980, May 1981. Internal document.

[60] Ottilie Kritsch and Norbert Auzinger. Jahresbericht 1982 und arbeitsmarktpolitisches Schwerpunktprogramm 1983 des Landesarbeitsamtes Niederösterreich, May 1983. Internal document.

[61] Ottilie Kritsch and Norbert Auzinger. Jahresbericht 1983 und arbeitsmarktpolitisches Schwerpunktprogramm 1984 des Landesarbeitsamtes Niederösterreich, May 1984. Internal document.

[62] Ottilie Kritsch and Norbert Auzinger. Jahresbericht '84 und arbeitsmarktpolitisches Schwerpunktprogramm '85, June 1985. Internal document.

[63] Ottilie Kritsch and Norbert Auzinger. Jahresbericht '86 und arbeitsmarktpolitisches Schwerpunktprogramm '87, May 1987. Internal document.

[64] Ottilie Kritsch and Hans-Werner Wieland. Jahresbericht 1979, May 1980. Internal document.

[65] Mark Last, Oded Maimon, and Einat Minkov. Improving Stability of Decision Trees. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 16:145–159, March 2002.

[66] Michael Lechner. Modified Causal Forests for Estimating Heterogeneous Causal Effects. Technical report, IZA Institute of Labour Economics, December 2018.

[67] Andy Liaw and Matthew Wiener. Classification and Regression by Random Forest. *R News*, 2(3):18–22, December 2002.

[68] Wei-Yin Loh. Fifty Years of Classification and Regression Trees. *International Statistical Review*, 82:329–367, June 2014.

[69] James D. Malley, József Kruppa, Abhijit Dasgupta, Karen G. Malley, and Adrian Ziegler. Probability Machines: Consistent Probability Estimation Using Nonparametric Learning. *Methods of Information in Medicine*, 51(1):74–81, 2012.

[70] Simon Matty. Predicting Likelihood of Long-term Unemployment: The Development of a UK Jobseekers' Classification Instrument. Technical report, Department for Work and Pensions, London, United Kingdom, 2013.

[71] Christine Mayer, Markus Hemetner, Mariia Kostetckaia, Renate Ruech, Asya Dimitrova, Ariane De Rocchi, Eva Gschwend, Nick Evans, and Andreas Prahl. *Smarter, Greener, More inclusive? Indicators to Support the Europe 2020 Strategy: 2018 Edition.* European Union, December 2019.

[72] Manuela Mayer and Manfred Trott. EDV-Handlungsanleitung mit verfahrensanweisungen zur AMS Applikation PST, June 2019. Internal document.

[73] Michael Mayer. *Package 'missRanger': https://cran.r-project.org/web/packages/missRanger/.* CRAN repository, 2019.

[74] Thomas Michlik, Herbert Kubicek, and Walter Fortunat. 25 Jahre AMS EDV. CD ROM publication, May 2001. Internal document.

[75] Koreen Millard and Murray Richardson. On the Importance of Training Data Sample Selection in Random Forest Image Classification: A Case Study in Peatland Ecosystem Mapping. *Remote Sensing*, 7:8489–8515, July 2015.

[76] John Mingers. An Empirical Comparison of Pruning Methods for Decision Tree Induction. *Machine Learning*, 4(2):227–243, November 1989.

[77] John Mingers. An Empirical Comparison of Selection Measures for Decision Tree Induction. *Machine Learning*, 3(4):319–342, March 1989.

[78] Bernard M. E. Moret. Decision Trees and Diagrams. *ACM Computing Surveys*, 14(4):593–623, December 1982.

[79] Chuck Murray. *Oracle SQL Developer User's Guide (Release 4.0).* Oracle, 2014.

[80] Ingrid Nagl and Tanja Jandl-Gartner. *Basisinformationsbericht. Arbeitsmarktpolitik - Institutionen, gesetzlicher Rahmen und Verfahren, Maßnahmen: Berichtsjahr 2016/2017.* Federal Ministry of Labor, Social Affairs, Health and Consumer Protection, April 2018.

[81] Claudia Nau, Hugh Ellis, Hongtai Huang, Brian S. Schwartz, Annemarie Hirsch, Lisa Bailey-Davis, Amii M. Kress, Jonathan Pollak, and Thomas A. Glass. Exploring the Forest Instead of the Trees: An Innovative Method for Defining Obesogenic and Obesoprotective Environments. *Health & Place*, 35:136–146, August 2015.

[82] Christina Neuwirth. Predicting Educational Attainment of the Austrian Population Using Data from the Austrian Social Security Institutions. Technical Report 2016-01, The Christian Doppler (CD) Laboratory - Aging, Health, and the Labor Market, Johannes Kepler University Linz, Austria, 2016.

[83] Matthew Norton and Stan Uryasev. Maximization of AUC and Buffered AUC in Classification. Technical report, Risk Management and Financial Engineering Lab, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, January 2015.

[84] Joseph Ogutu, Hans-Peter Piepho, and Torben Schulz-Streeck. A Comparison of Random Forests, Boosting and Support Vector Machines for Genomic Selection. In *Proceedings of the 14th European Workshop on QTL Mapping and Marker Assisted Selection (QTL-MAS)*, May 2010.

[85] Arbeitsmarktservice Österreich. Arbeitsmarktpolitische Ziele 2017. Technical report, Public Employment Service Austria, Vienna, Austria, July 2016. Internal document.

[86] Arbeitsmarktservice Österreich. About the AMS: https://www.ams.at/organisation/public-employment-service-austria/about-ams, September 2018. Accessed: 17 February 2020.

[87] Arbeitsmarktservice Österreich. Arbeitsmarktpolitische Ziele 2020. Technical report, Public Employment Service Austria, Vienna, Austria, February 2019. Internal document.

[88] Arbeitsmarktservice Österreich. Erwerbskarrierenmonitoring, April 2019. DWH Erwerbskarrierenmonitoring (Infopunkt "Allgemein"). Internal document.

[89] Die Sozialpartner Österreich. Die Österreichische Sozialpartnerschaft: https://www.sozialpartner.at, 2015. Accessed: 17 February 2020.

[90] Republik Österreich. Arbeitsmarktservicegesetz (AMSG), 1994. BGBl. Nr. 313/1994. Accessed: 11 February 2020.

[91] Jigar Patel, Sahil Sha, Priyank Thakkar, and Ketan Kotecha. Predicting Stock and Stock Price Index Movement Using Trend Deterministic Data Preparation and Machine Learning Techniques. *Expert Systems with Applications*, 42:259–268, January 2015.

[92] Reinhard Pipal. Personal account by the Statistican in Chief of the AMS Niederösterreich, February 2020. Personal account.

[93] Philipp Probst, Marvin N. Wright, and Anne-Laure Boulesteix. Hyperparameters and Tuning Strategies for Random Forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):1301–1319, 2019.

[94] Yanjun Qi, Ziv Bar-Joseph, and Judith Klein-Seetharaman. Evaluation of Different Biological Data and Computational Classification Methods for Use in Protein Interaction. *Proteins: Structure, Function, and Bioinformatics*, 63:490–500, May 2006.

[95] J. Ross Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, March 1986.

[96] Roland Ruepp and Heimo Flink. 63. EDV-Information, April 1998. Internal document.

[97] S. Rasoul Safavian and David A. Landgrebe. A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:660–674, 1991.

[98] David W. Scott. Multivariate Density Estimation and Visualization. Technical paper, Humboldt-Universität Berlin, Center for Applied Statistics and Economics (CASE), 2004.

[99] Claude E. Shannon. A Mathematical Theory of Communication (Reprint for the Bell System Technical Journal, with corrections). *Mobile Computing and Communications Review*, 5(1):3–55, January 2001.

[100] Yu-Shan Shih. Families of Splitting Criteria for Classification Trees. *Statistics and Computing*, 9(4):309–315, November 1999.

[101] Michael Snipes and D. Christopher Taylor. Model Selection and Akaike Information Criteria: An Example from Wine Ratings and Prices. *Wine Economics and Policy*, 3(1):3–9, 2014.

[102] Daniel J. Stekhoven and Peter Bühlmann. MissForest - Non-parametric Missing Value Imputation for Mixed-type Data. *Bioinformatics*, 28(1):112–118, October 2011.

[103] Forrest Stevens, Andrea Gaughan, Catherine Linard, and Andrew Tatem. Disaggregating Census Data for Population Mapping Using Random Forests with Remotely-Sensed and Ancillary Data. *Public Library of Science ONE*, 10(2):online, February 2015.

[104] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution. *BMC bioinformatics*, 8(1):25, February 2007.

[105] Vladimir Svetnik, Andy Liaw, Christopher Tong, John Culberson, Robert P. Sheridan, and Bradley Feuston. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. *Journal of Chemical Information and Computer Sciences*, 43:1947–1958, November 2003.

[106] John A. Swets. Measuring the Accuracy of Diagnostic Systems. *Science*, 240(4857):1285–1293, 1988.

[107] Fei Tang and Hemant Ishwaran. Random Forest Missing Data Algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(6):363–377, 2017.

[108] Dataflair Team. Pros and Cons of R Programming Language: Unveil the Essential Aspects: https://data-flair.training/blogs/pros-and-cons-of-r-programming-language/, December 2019. Accessed: 18 May 2020.

[109] Alaa Tharwat. Classification Assessment Methods: A Detailed Tutorial. *Applied Computing and Informatics*, online:1–74, 2018.

[110] Alaa Tharwat, Yasmine S. Moemen, and Aboul Ella Hassanien. Classification of Toxicity Effects of Biotransformed Hepatic Drugs Using Whale Optimized Support Vector Machines. *Journal of Biomedical Informatics*, 68:132–149, 2017.

[111] Fabian Tomaschek, Peter Hendrix, and R. Harald Baayen. Strategies for Addressing Collinearity in Multivariate Linguistic Data. *Journal of Phonetics*, 71:249–267, 2018.

[112] Paul E. Utgoff. Incremental Induction of Decision Trees. *Machine Learning*, 4(2):161–186, November 1989.

[113] Georg Waller and Erika Jaklitsch-Schmitt. Die Organisation des AMS, March 2017. Internal document.

[114] WHO. Rolling updates on coronavirus disease (COVID-19): https://www.who.int/emergencies/diseases/novel-coronavirus-2019/events-as-they-happen, 2020. Accessed: 21 May 2020.

[115] Leland Wilkinson. Tree Structured Data Analysis: AID, CHAID and CART. In *Proceedings of the Sawtooth/SYSTAT Joint Software Conference*, 1992.

[116] Marvin N. Wright, Stefan Wagner, and Philipp Probst. *Package 'ranger': https://github.com/imbs-hl/ranger*. CRAN repository, 2019.

[117] Marvin N. Wright and Andreas Ziegler. Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017.

[118] Xindong Wu and Vipin Kumar. *The Top Ten Algorithms in Data Mining*. Chapman & Hall, CRC Press, 1st edition, 2009.

[119] Fen Xia, Wensheng Zhang, Fuxin Li, and Yanwu Yang. Ranking with Decision Tree. *Knowledge and Information Systems*, 17(3):381–395, December 2008.

[120] Hadi Sadoghi Yazdi and Nima Salehi-Moghaddami. Multi Branch Decision Tree: A New Splitting Criterion. *International Journal of Advanced Science and Technology*, 45:91–106, August 2012.

[121] Weiran Yuchi, Enkhjargal Gombojav, Buyantushig Boldbaatar, Jargalsaikhan Galsuren, Sarangerel Enkhmaa, Bolor Beejin, Gerel Naidan, Chimedsuren Ochir, Bayarkhuu Laagan, Byambaa Tsogtbaatar, Prabjit Barn, Sarah Henderson, Craig Janes, Bruce Lanphear, Lawrence C. McCandless, Tim Takaro, Scott A. Venners, Glenys Webster, and Ryan W. Allen. Evaluation of Random Forest Regression and Multiple Linear Regression for Predicting Indoor Fine Particulate Matter Concentrations in a Highly Polluted City. *Environmental Pollution*, 245:746–753, November 2018.

A Appendix: Context

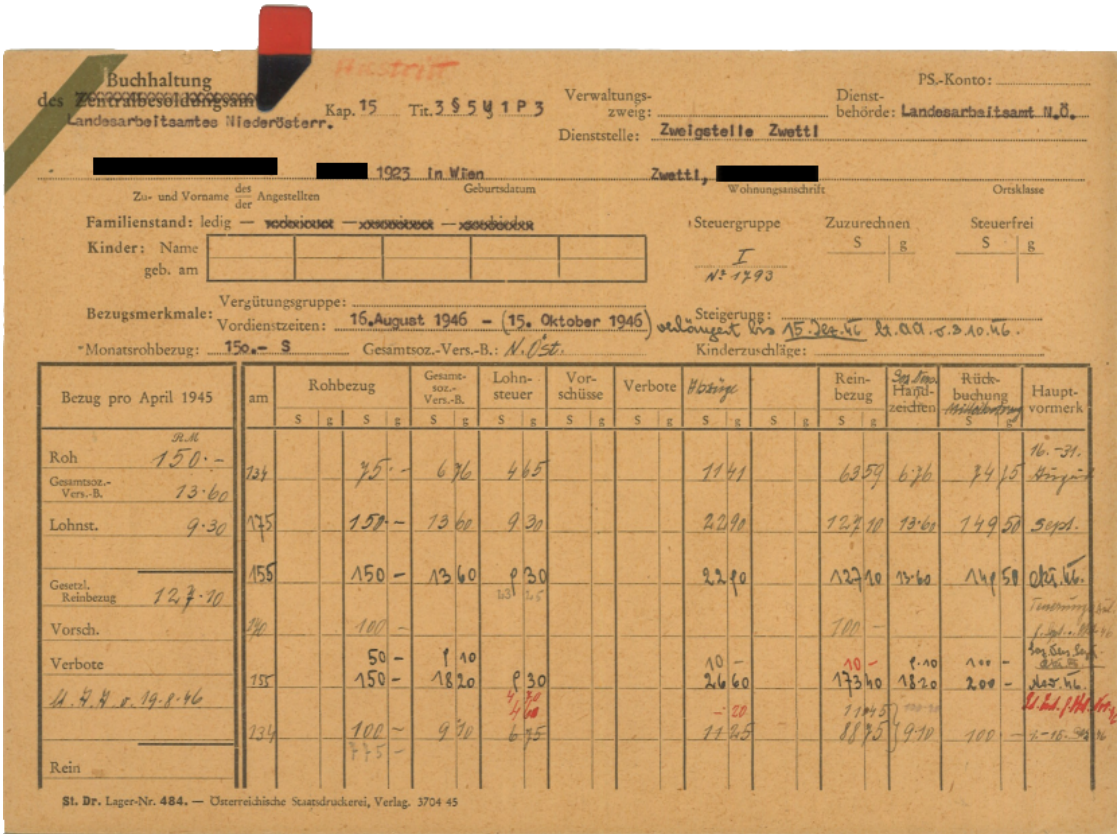

Figure 39: *Working card* from 1946. The *working card* ("Arbeitskarte") covered basic client information, such as the client's name, sex, address, and date of birth. Furthermore, it covered information about the client's insurance and benefit receipt. The red tab at the top of the paper card indicates that the client is female (blue tab: male). Further tabs were used in order to indicate a client's respective age group, former occupation, or industry (source: archive of the AMS Lower Austria).

priority	status code	status label	category	super-category
1	FU	Fragmentierte UB	unfunded employment	employment
2	LFJAS	AMS gef. Lehre JASG	funded employment	employment
3	LFIBA	AMS gef. Lehre IBA	funded employment	employment
4	LFUBA	AMS gef. Lehre UBA	funded employment	employment
5	LFLST	AMS gef. Lehre LST	funded employment	employment
6	LFP30	AMS gef. Lehre Paragr. 30	funded employment	employment
7	LFVOL	AMS gef. Lehre VOL	funded employment	employment
8	FBEB	AMS gef. unselbst. Besch. EB	funded employment	employment
9	FBBEB	AMS gef. unselbst. Besch. BEB	funded employment	employment
10	FBBS2	AMS gef. unselbst. Besch. BS2	funded employment	employment
11	FBBS3	AMS gef. unselbst. Besch. B3	funded employment	employment
12	FBEB1	AMS gef. unselbst. Besch. EB1	funded employment	employment
13	FBEB2	AMS gef. unselbst. Besch. EB2	funded employment	employment
14	FBGEB	AMS gef. unselbst. Besch. GEB	funded employment	employment
15	FBSOL	AMS gef. unselbst. Besch. SOL	funded employment	employment
17	FBKOM	AMS gef. unselbst. Besch. Kombilohn	funded employment	employment
18	FBSOB	AMS gef. unselbst. Besch. SÖB	funded employment	employment
19	FBBS1	AMS gef. unselbst. Besch. BS1	funded employment	employment
20	FBGBP	AMS gef. unselbst. Besch. GBP	funded employment	employment
21	FBEPU	AMS gef. unselbst. Besch. EPU	funded employment	employment
22	LFTEL	BSB gef. Lehre TEIL	funded employment	employment
23	LFVRL	BSB gef. Lehre VERL	funded employment	employment
24	LFNRM	BSB gef. Lehre NORM	funded employment	employment
25	FBES	BSB gef. unselbst. Besch. ES	funded employment	employment
26	FBES1	BSB gef. unselbst. Besch. ES 2525	funded employment	employment
27	FBARB	BSB gef. unselbst. Besch. ARBS	funded employment	employment
28	FBENG	BSB gef. unselbst. Besch. ENTG	funded employment	employment
29	FBUN1	BSB gef. unselbst. Besch. UNT1	funded employment	employment
30	FBUN2	BSB gef. unselbst. Besch. UNT2	funded employment	employment
31	FBUN3	BSB gef. unselbst. Besch. UNT3	funded employment	employment
32	FBBP	BSB gef. unselbst. Besch. BP	funded employment	employment
33	BE	Beamte	unfunded employment	employment
34	LE	Lehre	unfunded employment	employment
35	AA	Arbeiter/Angestellte	unfunded employment	employment
36	FD	Freie Dienstverträge	unfunded employment	employment
37	SO	Sonstige UB	unfunded employment	employment
38	S1S2	selbständige Besch. lt. HV ohne LW	self-employment	employment
39	AL	AL	unemployment	AMS registration
40	D2	DLU aktiv	AMS qualification	AMS registration
41	SC	Sonstige SC	AMS qualification	AMS registration
42	LS	Lehrstellensuchend	other AMS registration	AMS registration
43	SF	Fachkräftestipendium	AMS qualification	AMS registration
44	SR	REHA-Schulung	AMS qualification	AMS registration
45	SBSVA	selbständige Besch. lt. SVA ohne LW	self-employment	employment
46	66	Übergangsgeldbezug	AMS qualification	AMS registration
47	LW	Landwirte	self-employment	employment
48	W1	Wochengeld aus DV	out-of-labour force	other
49	W2	Wochengeld ohne DV	out-of-labour force	other
50	ED	Karenz aus aufr. DV	out-of-labour force	other
51	EO	Karenz ohne aufr. DV	out-of-labour force	other
52	KG	Kinderbetreuungsgeld aus aufr. DV	out-of-labour force	other
53	KO	Kinderbetreuungsgeld ohne aufr. DV	out-of-labour force	other
54	PZ	Präsenzdienst / Zivildienst	out-of-labour force	other
55	RE	Erwerbspension / Rente	out-of-labour force	other
56	SG	Sonst.ges.erwerbsferne Pos.	out-of-labour force	other
57	AG	Klärung Arbeitsfähigkeit	other AMS registration	AMS registration
58	AS	Arbeitsuchend	other AMS registration	AMS registration
59	AM	Leistungsbezug aufgrund ausländischer Versicherungszeiten	other AMS registration	AMS registration
60	AF	Frühzeitige Arbeitssuche	other AMS registration	AMS registration
61	LF	Frühzeitige Lehrstellensuche	other AMS registration	AMS registration
62	TA	Teilintegrierte AusländerInnen	other AMS registration	AMS registration
63	VM	Vormerkung	other AMS registration	AMS registration
64	AO	Arbeitslosigkeit laut HV	HV registration	other
65	G1	Unselbständige GB	marginal employment	other
66	AU	Ausbildung	other position far from labour market	other
67	MK	Mitversichertes Kind	other position far from labour market	other
68	MP	MitversicherteR PartnerIn	other position far from labour market	other
69	MS	Sonstige Mitversicherung	other position far from labour market	other
70	SV	Sonst. Versicherungszeiten	other position far from labour market	other
71	TO	Tod / Keine Daten	undefined	other
72	BA	Vermutete Auslandsbeschäftigung	unfunded employment	employment
73	LL	Versicherungslücken	other position far from labour market	other
74	KD	keine Daten	undefined	other

Table 3: List of *Uni-statuses* including the corresponding *priority code*, *category*, and *super-category*. Statuses relevant for population definition are highlighted in blue.

B Appendix: Theory

Goodness-of-split function. Let J be classes numbered $1, ..., J$ and $p = (p_1, ..., p_J)$ are proportions of the classes in a node t. If $p = (p_1, ..., p_J)$ are the node proportions, then $\phi(p)$ is an *impurity function* if it is convex in p, has a maximum when all p_J are equal, and is a minimum when one of the $p_J = 1$. For an impurity function $\phi(p)$ the associated goodness-of-split is defined as

$$\theta(s,t) = \phi(p) - P_L\phi(p_L) - P_R\phi(p_R),$$

where P_L is the proportion of node t sent to the left branch after split s, and $P_R = 1 - P_L$ is the proportion sent to the right branch. Furthermore, $p_L = (p_{1,L}, ..., p_{J,L})$ is the proportion of the J classes in the left node t_L and p_R is the proportion of the J classes in the right node t_R [17, p. 41f].

Permutation importance (PIMP). In contrast to Breiman's original *(permutation) variable importance*, which permutes the OOB examples of a certain feature, the PIMP algorithm permutes the *response vector* s times. The relevance for all features is assessed for each permutation. The resulting vector of s importance measures for every variable is called *null importance*.

PIMP draws upon three different *probability distributions*, viz. *Gaussian distribution*, *log-normal distribution*, and *gamma distribution*, which show different skews. This allows the PIMP algorithm to fit the population of *null importances* to the chosen distribution.

The PIMP algorithm uses the *Kolmogorov-Smirnov test* to automatically identify the most suitable distribution (the choice of probability distribution, however, can also be made manually). After that, maximum likelihood estimators for the selected distribution are calculated and the *PIMP P-value* (i.e. the probability of observing a relevance v or higher by using the true response vector) is computed. If the tests show little resemblance to any of the three proposed distributions, a non-parametric estimation of PIMP P-values is used, which is determined by the fraction of *null importances* that are more extreme than the true importance v. Since the variance of *null importances* may be very small in practical applications (which leads to artificially boosted PIMP P-values), variances that are smaller than the mean variance of all importances are set to the mean variance [3, p. 1341f].

Aggregation. Let $\mathcal{L}$ be a learning set (*training set*) that consists of data $\{(y_n, x_n), n = 1, ..., N\}$. If there is only one learning set, the set predictor $\varphi(x, \mathcal{L})$ is based on this leaning set $\mathcal{L}$. Instead, if there is a number of k learning sets $\mathcal{L}_k$, with each of them holding N observations, the set predictor is of the form $\varphi(x, \mathcal{L}_k)$. In the case of several learning sets, aggregation depends on the response y being either numerical or class labels.

If y is numerical (regression problem), $\varphi(x, \mathcal{L}_k)$ is the average over k, meaning that the *aggregated* learning set predictor $\varphi_A(x) = E_{\mathcal{L}\varphi}(x, \mathcal{L})$, where $E_{\mathcal{L}}$ is the expectation over $\mathcal{L}$. If y is a class (classification problem), $j \in \{1, ..., J\}$, the method for receiving the aggregated set predictor $\varphi_A(x)$ is voting: $\varphi_A(x) = argmax_j N_j$, where $N_j = \#\{k; \varphi(x, \mathcal{L}_k) = j\}$ is the number k of set predictors for a class label j [16, p. 1].

Out-of-bag error estimation. For random forest classification problems, OOB-error estimation uses *error rate (ERR)*:

$$ERR \approx ERR^{OOB} = n^{-1} \sum_{i=1}^{n} I(\hat{Y}^{OOB}(X_i) \neq Y_i),$$

where $I(\cdot)$ is the indicator function.

Analogous to the *accuracy (ACC)* and *error rate (ERR)* as defined in table 1, the OOB accuracy is $Acc^{OOB} = 1 - ERR^{OOB}$ [105] and the OOB-error rate is:

$$ERR^{OOB} = 1 - Acc^{OOB} = \frac{FP^{OOB} + FN^{OOB}}{TP^{OOB} + TN^{OOB} + FP^{OOB} + FN^{OOB}}$$

For random forest regression problems, OOB-error estimation applies *mean squared error (MSE)*:

$$MSE \approx MSE^{OOB} = n^{-1} \sum_{i_1}^{n} \{\hat{Y}^{OOB}(X_i) - Y_i\}^2.$$

C Appendix: Data

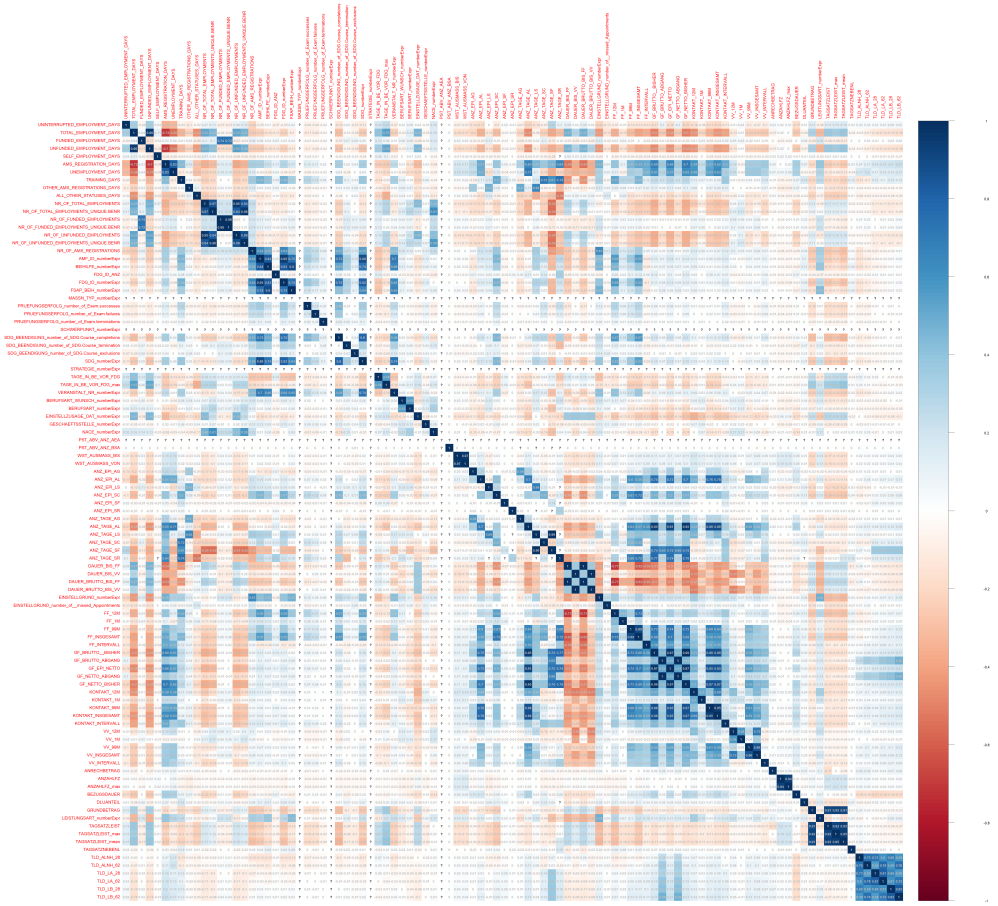

Figure 40: *Correlation matrix* of numerical variables for the *basic set*. The correlation coefficient (Pearson's r) is a measure of linear relationship between two variables. The correlation coefficient ranges from −1 for negative correlations (red) to +1 for positive correlations (blue). White squares with question marks (?) indicate missing values, mostly originating from variables containing only one expression.

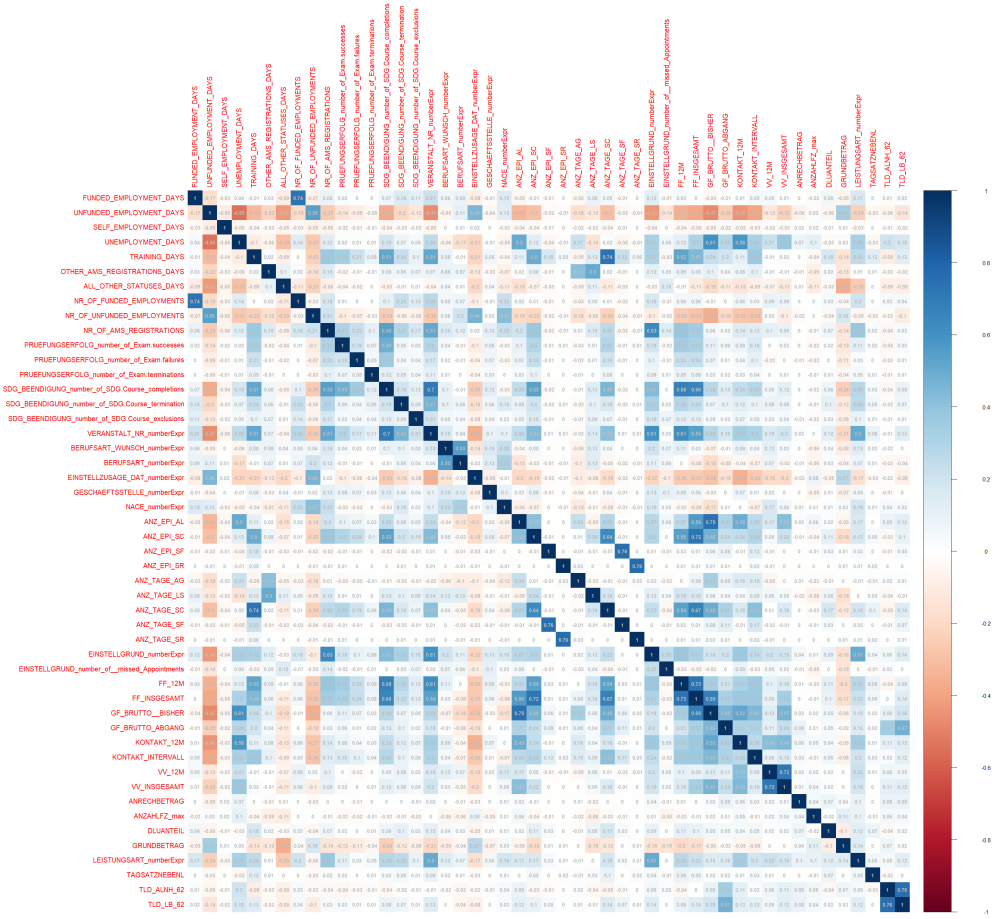

Figure 41: *Correlation matrix* of numerical variables for the *main set*. The correlation coefficient (Pearson's r) is a measure of linear relationship between two variables. The correlation coefficient ranges from −1 for negative correlations (red) to +1 for positive correlations (blue). White squares with question marks (?) indicate missing values, mostly originating from variables containing only one expression.

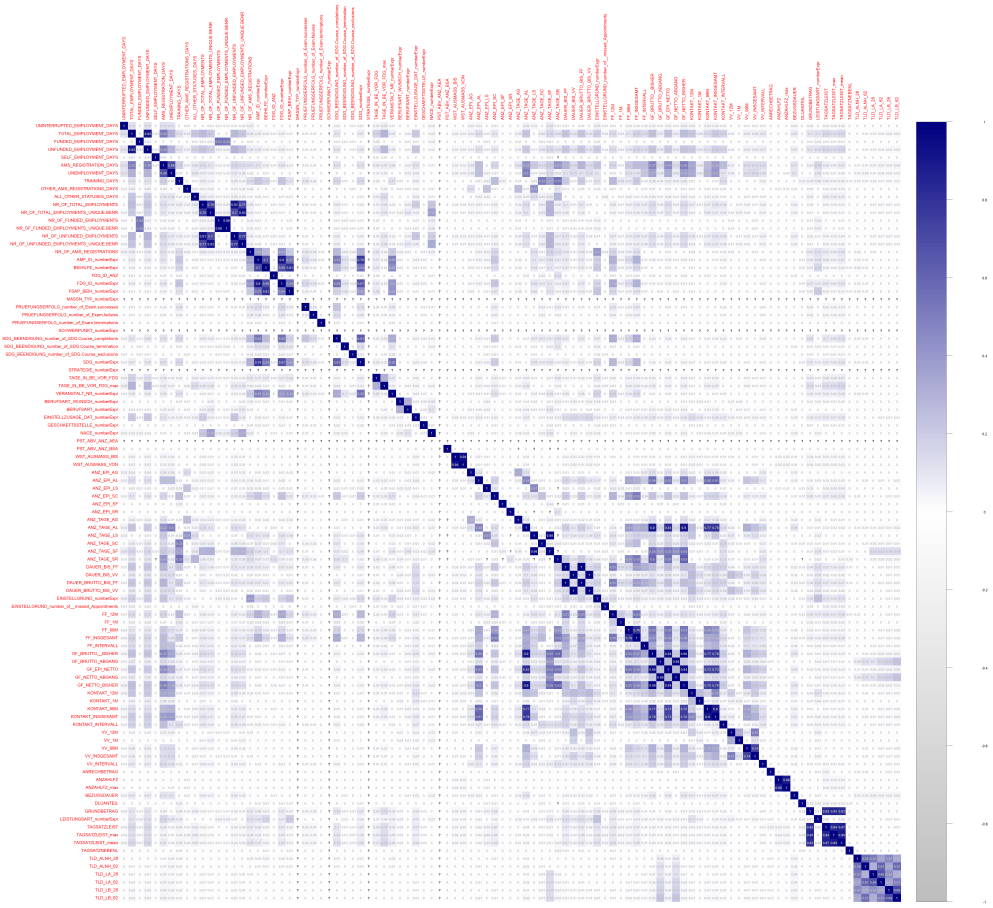

Figure 42: *Coefficient of determination (r^2)* of numerical variables for the *basic set*. The coefficient of determination is a measure of the variance in either variable that can be explained by the other variable. The coefficient of determination ranges from 0 (grey) to 1 (blue). White squares with question marks (?) indicate missing values, mostly originating from variables containing only one expression.

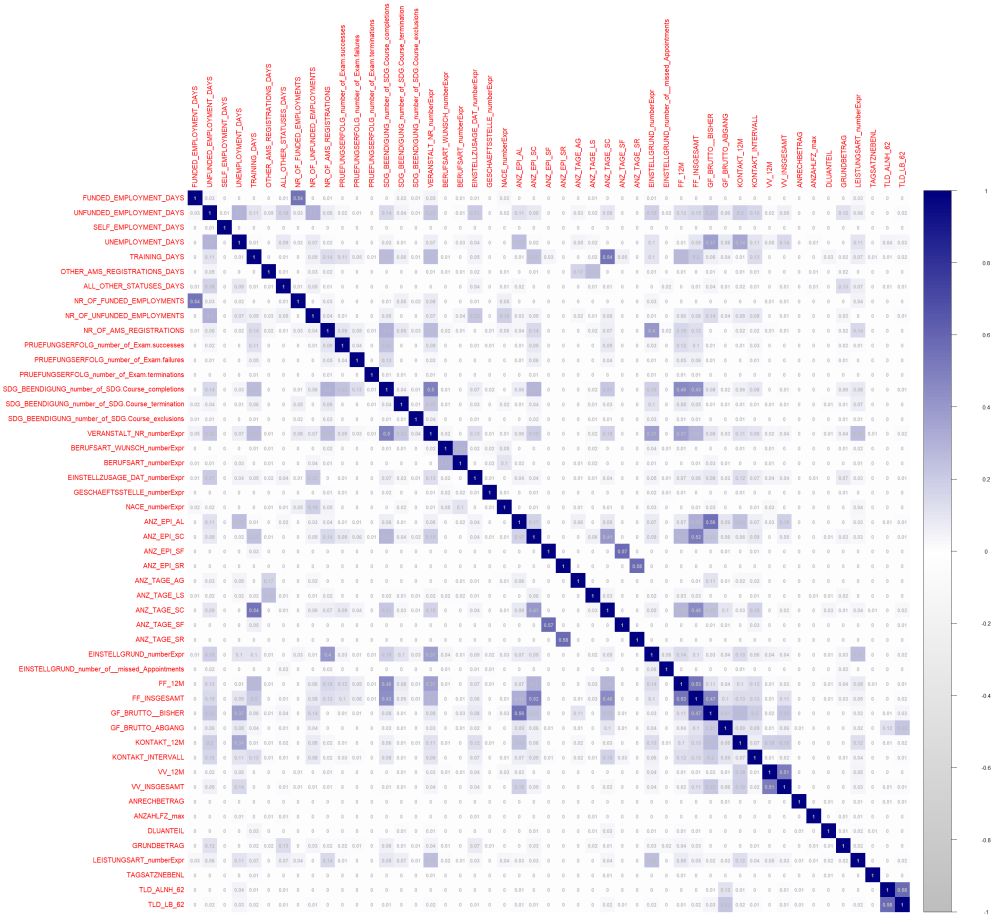

Figure 43: *Coefficient of determination (r^2)* of numerical variables for the *main set*. The coefficient of determination is a measure of the variance in either variable that can be explained by the other variable. The coefficient of determination ranges from 0 (grey) to 1 (blue). White squares with question marks (?) indicate missing values, mostly originating from variables containing only one expression.

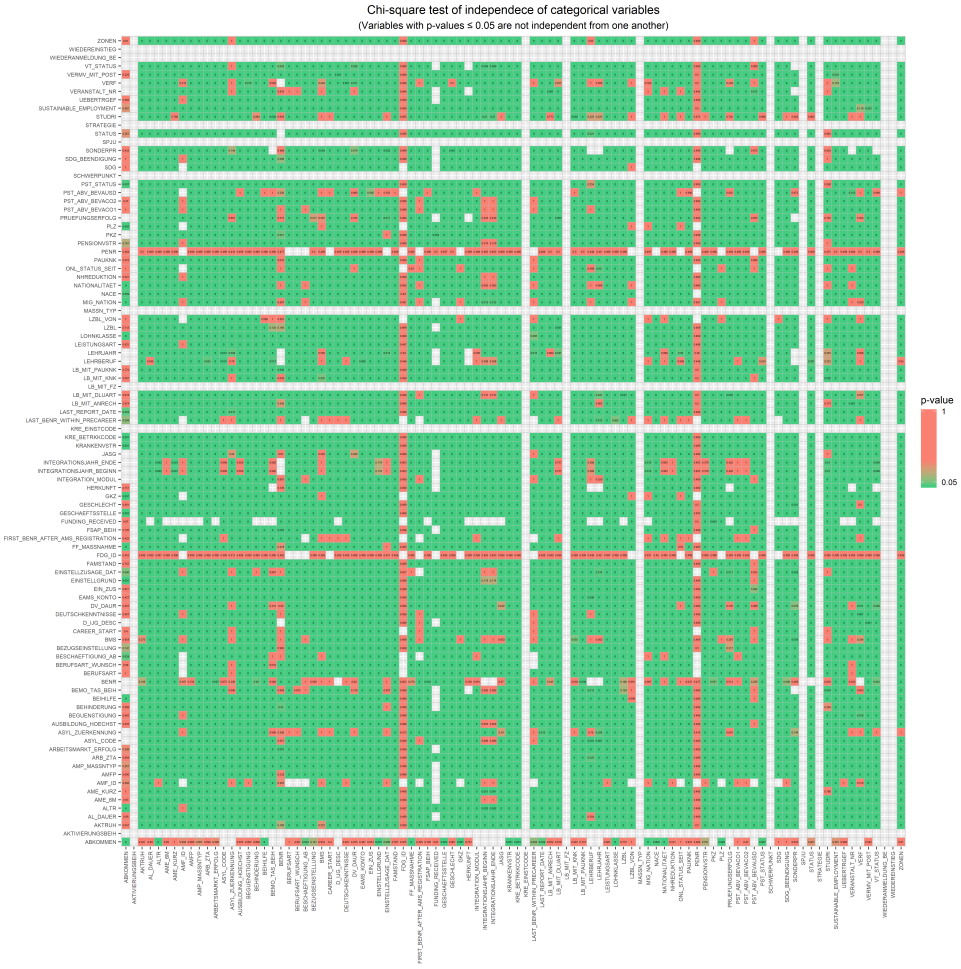

Figure 44: *P-value heatmap* of categorical variables for the *basic set*. The *p-values* are the result of the pairwise performance of the chi-squared test of independence. If the *p-value* for two values is smaller or equal to .05, H_0 (*"The variables are independent from one another."*) is rejected in favour of H_A (*"The variables are not independent from one another."*). Green squares represent p-values smaller or equal to .05, red squares represent p-values greater than .05, and white squares represent NAs.

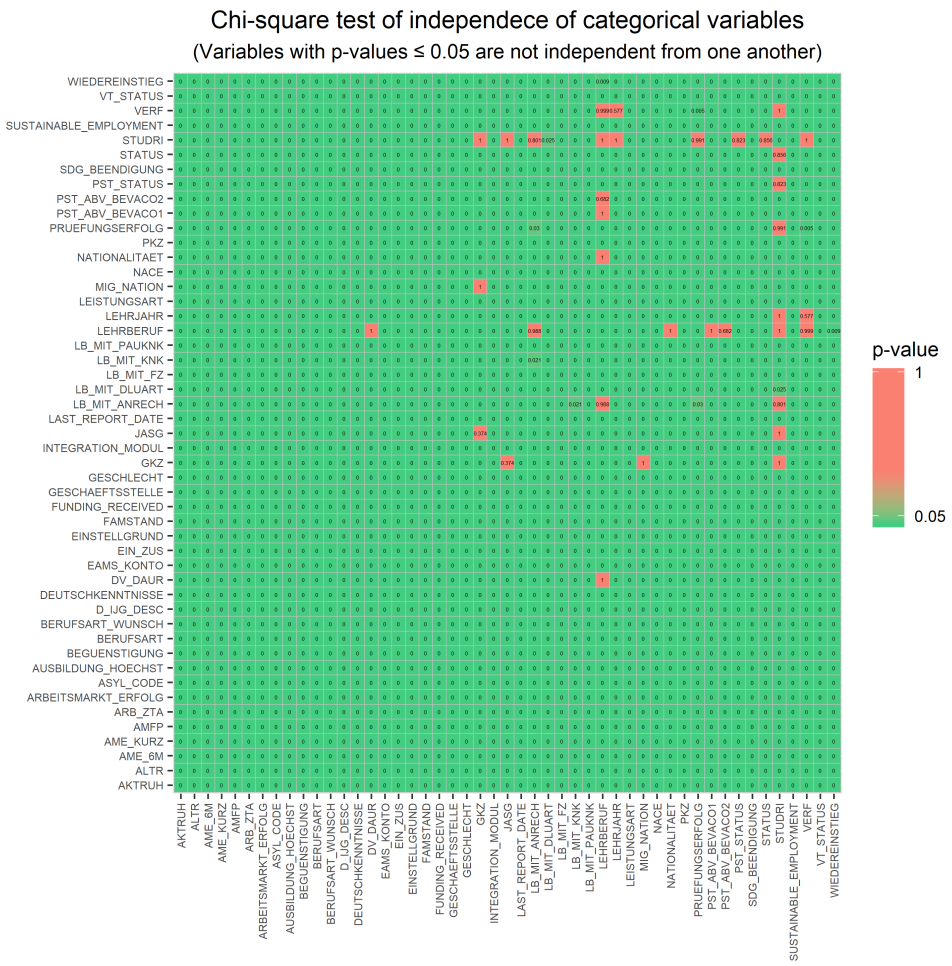

Figure 45: *P-value heatmap* of categorical variables for the *main set*. The *p-values* are the result of the pairwise performance of the chi-squared test of independence. If the *p-value* for two values is smaller or equal to .05, H_0 (*"The variables are independent from one another."*) is rejected in favour of H_A (*"The variables are not independent from one another."*). Green squares represent p-values smaller or equal to .05, red squares represent p-values greater than .05, and white squares represent NAs.

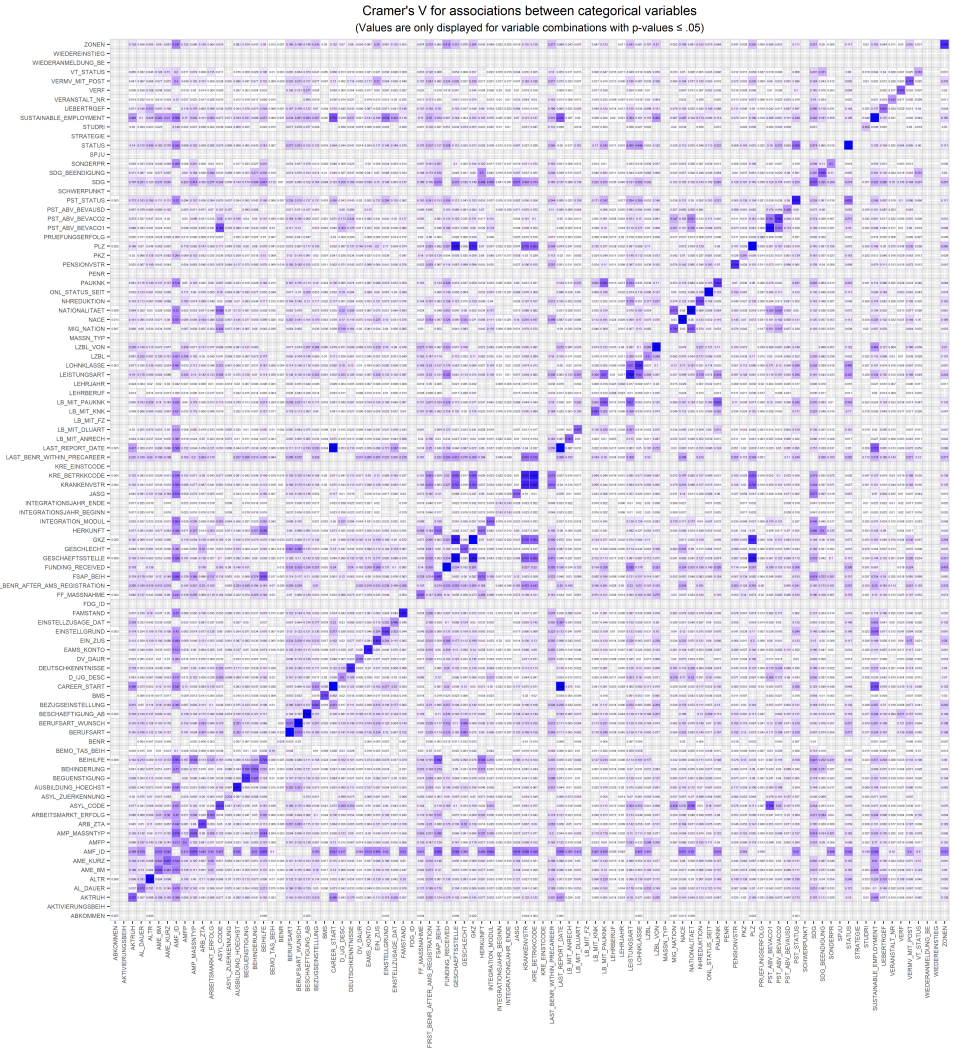

Figure 46: *Cramér's V value heatmap* for the *basic set*. While chi-squared test of independence solely indicates whether two variables are related, *Cramér's V* statistic informs about the strength of the relation. *Cramér's V* value ranges from 0 (white) to 1 (blue), with higher values indicating a higher effect size (i.e. a stronger association between the variables) [54].

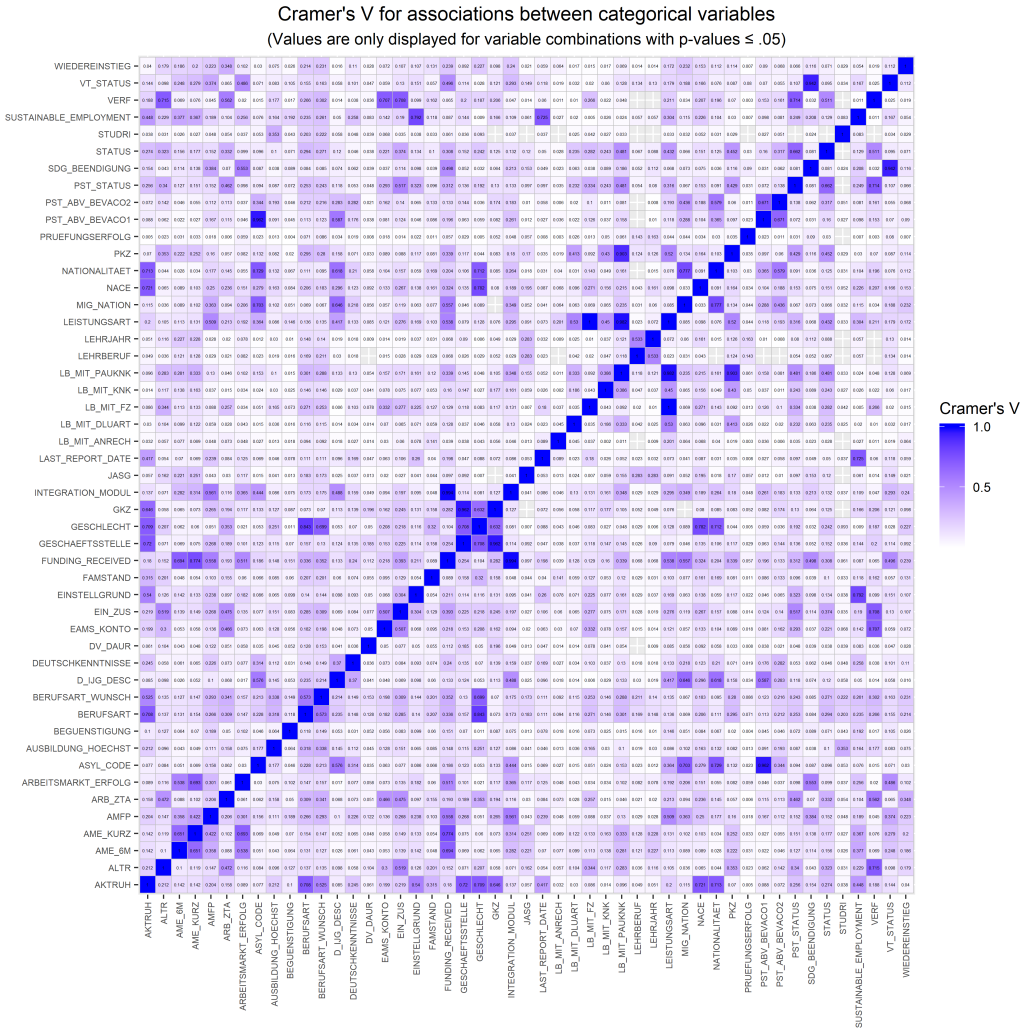

Figure 47: Cramér's V value heatmap for the *main set*. While chi-squared test of independence solely indicates whether two variables are related, *Cramér's V* statistic informs about the strength of the relation. *Cramér's V* value ranges from 0 (white) to 1 (blue), with higher values indicating a higher effect size (i.e. a stronger association between the variables) [54].

no.	variable name	description	type of variable	type of data	data category	data super-category	keep column	reason for omission	part of main set	part of production set
1	PENR	The client's ID. A unique pseudonymised observation identifier.	production variable	nominal categorical	person basic data	Person-bound information	no	ID variable*	no	no
2	BENR	A company's ID. A unique identifier for companies, known as "contribution account number" ("Beitragskontonummer"), defined by the Main Association of Austrian Social Security Institutions ("Hauptverband der österreichischen Sozialversicherungs-träger"). The number of company IDs pertraining to this variable is very small. This is because only AMS-registration statuses were considered (i.e. 14 out of 74 Uni-statuses). It can be assumed that the companies pertaining to this variable are socio-economic enterprises.	production variable	nominal categorical	employment career data	AMS generated data	no	ID variable	no	no
3	STATUS	Universal Labour Market Status ("Uni-Status") expresses a person's position in the labour market. The Uni-status is created by blending AMS client data with insurance data from the Main Association of Austrian Social Security Institutions ("Hauptverband der österreichischen Sozialversicherungsträger"), data about disabled people from the "Service of the Ministry of Social Affairs" ("Sozialministeriums-service"), and enterprise data from the "Social Insurance Institution of the Industrial Economy" ("Sozialversicherungsanstalt der gewerblichen Wirtschaft"). The pool of Uni-statuses comprises 74 different hierarchically ordered statuses associated to three different super-categories (i.e. "AMS registration", "employment", and "other (statuses)").	production variable	nominal categorical	labour market policy data	AMS generated data	yes		yes	yes
4	LAST_REPORT_DATE	The report date is the last day of a month. From the AMS' perspective, this variable represents the last known status of a client.	additionally created variable	ordinal categorical	labour market policy data	AMS generated data	yes		yes	yes
5	CAREER_START	The career start represents the start of the pre-career. The pre-career is a two-year timespan, dating back from the last report date. This is an artificially created auxiliary variable.	auxiliary variable	ordinal categorical	employment career data	AMS generated data	no	auxiliary variable	no	no
6	UNINTERRUPTED_ EMPLOYMENT_DAYS	The number of days of uninterrupted employment in one's "post-carreer". There can either be one single employment or several consecutive employ-ments, as long as these are seamlessly connected (without interruptions greater than 0 days). This is an artificially created auxiliary variable. It is the basis for generating the variable SUSTAINABLE_EMPLOYMENT.	auxiliary variable	numerical	employment career data	AMS generated data	no	auxiliary variable*	no	no
7	SUSTAINABLE_EMPLOYMENT	This is the target variable. An employment is considered sustainable if it lasts longer than 62 days. This variable is calculated based on the values stored in the variable UNINTERRUPTED_EMPLOYMENT_DAYS.	additionally created variable	dichotomous categorical	employment career data	AMS generated data	yes		yes	yes
8	FIRST_BENR_AFTER_AMS_ REGISTRATION	A company's ID. These company identifiers (contribution account numbers) represent a client's first employment relationship after AMS-registration. They are taken from a filtered set of 41 employment statuses (out of 74 Uni-tatuses in total).	additionally created variable	nominal categorical	employment career data	AMS generated data	no	ID variable*	no	no
9	LAST_BENR_WITHIN_ PRECAREER	A company's ID. These company identifiers (contribution account numbers) represent a client's last employment relationship within the precareer. They are taken from a filtered set of 41 employment statuses (out of 74 Uni-statuses in total).	additionally created variable	nominal categorical	employment career data	AMS generated data	no	ID variable	no	no
10	TOTAL_EMPLOYMENT_DAYS	This is the total number of days in employment within the pre-career. This variable comprises 41 different Uni-statuses. Furthermore, this variable represents the total sum of the variables FUNDED_EMPLOYMENT_DAYS, UNFUNDED_EMPLOYMENT _DAYS and SELF_EMPLOYMENT_DAYS, and constitutes their superset.	additionally created variable	numerical	employment career data	AMS generated data	no	strong correlation*	no	no
11	FUNDED_EMPLOYMENT_DAYS	This is the total number of days in funded employment relationships within the pre-career, comprising 31 different Uni-statuses.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
12	UNFUNDED_EMPLOYMENT_ DAYS	This is the total number of days in unfunded employment relationships within the pre-career, comprising 7 different Uni-statuses.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
13	SELF_EMPLOYMENT_DAYS	This is the total number of days in self-employment within the pre-career, comprising 3 different Uni-statuses.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
14	AMS_REGISTRATION_DAYS	This is the total number of days in AMS-registration within the pre-career, comprising 14 different Uni-statuses. Furthermore, this variable represents the total sum of the variables UNEMPLOYMENT_DAYS, TRAINING_DAYS and OTHER_AMS_ REGISTRATION_DAYS, and constitutes their superset.	additionally created variable	numerical	employment career data	AMS generated data	no	strong correlation	no	no
15	UNEMPLOYMENT_DAYS	This is the total number of days of unemployment witin the pre-career, comprising 1 Uni-status.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
16	TRAINING_DAYS	This is the total number of days of training within the pre-career, comprising 5 different Uni-statuses. Although clients in training are basically unemployed, there are specific statuses for AMS-induced qualification measures.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
17	OTHER_AMS_ REGISTRATIONS_DAYS	This is the total number of days in other AMS-registrations within the pre-career. This variable covers a set of 8 different Uni-statuses that are neither used for encoding unemployment episodes nor for qualification measures.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
18	ALL_OTHER_STATUSES_ DAYS	This is the total number of days in all other Uni-statuses different from employment and AMS-registration within the pre-career, comprising 19 different Uni-statuses.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
19	NR_OF_TOTAL_ EMPLOYMENTS	This is the total number of employment relationships within the pre-career. Multiple countings of BENRs for each client are possible.	additionally created variable	numerical	employment career data	AMS generated data	no	strong correlation*	no	no
20	NR_OF_TOTAL_ EMPLOYMENTS_UNIQUE.BENR	This is the total number of employment relationships with unique companies within the pre-career (unique countings of BENRs).	additionally created variable	numerical	employment career data	AMS generated data	no	strong correlation*	no	no
21	NR_OF_FUNDED_ EMPLOYMENTS	This is the total number of funded employment relationships within the pre-career. Multiple countings of BENRs for each client are possible.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
22	NR_OF_FUNDED_ EMPLOYMENTS_UNIQUE.BENR	This is the total number of funded employment relationships with unique companies within the pre-career (unique countings of BENRs).	additionally created variable	numerical	employment career data	AMS generated data	no	strong correlation*	no	no
23	NR_OF_UNFUNDED_ EMPLOYMENTS	This is the total number of unfunded employment relationships within the pre-career. Multiple countings of BENRs for each client are possible.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
24	NR_OF_UNFUNDED_ EMPLOYMENTS_UNIQUE.BENR	This is the total number of unfunded employment relationships with unique companies within the pre-career (unique countings of BENRs).	additionally created variable	numerical	employment career data	AMS generated data	no	strong correlation*	no	no
25	NR_OF_AMS_REGISTRATIONS	This is the total number of AMS-registration episodes within the pre-career.	additionally created variable	numerical	employment career data	AMS generated data	yes		yes	no
26	AKTIVIERUNGSBEIH	Activation benefit ("Aktivierungsbeihhilfe"), a special support measure for people participating in the second labour market.	production variable	dichotomous categorical	funding/ promoting data	AMS generated data	no	no information value	no	no
27	AL_DAUER	Duration of the current unemployment episode at the last report date.	production variable	nominal categorical	labour market policy data	AMS generated data	no	redundant variable	no	no
28	AME_6M	Indication of labour market success ("Arbeitsmarkterfolg") after 6 months (e.g. employment relationship 6 months after support measure).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
29	AME_KURZ	Indication of labour market success ("Arbeitsmarkterfolg") after 92 days (e.g. employment relationship 92 days after support measure).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
30	AMF_ID	Support project ID. An AMF_ID represents a unique project. Within a project, several support measures (identifiable via FDG_ID) can be applied to a client (e.g. a certain qualification programme's course costs and ancilliary course costs, whereas both have a separate FDG_ID, but are assigned to one project with one AMF_ID).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
31	AMF_ID_numberExpr	This is the total number of different support project IDs within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
32	AMFP	Indication whether or not a client is considered a "person far from the labour market". Clients are considered "far from the labour market", if, within the last 12 months, they did not achieve at least 2 months of (interrupted or uninterrupted) employment relationships and show at least 4 months of (interrupted or uninterrupted) AMS-registration. Female returnees ("Wieder-einsteigerinnen") and young people under the age of 25 are never considered "far from the labour market".	production variable	dichotomous categorical	labour market policy data	AMS generated data	yes		yes	yes
33	AMP_MASSNTYP	Type of labour market policy measure. This variable's codes depend on the variable HERKUNFT, which defines the source application from which these codes emerge. As a consequence, this variable needs to be considered in the light of the variable HERKUNFT in order to be able to differentiate from one another measures which are stored under the same code.	production variable	nominal categorical	labour market policy data	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
34	ARBEITSMARKT_ERFOLG	Indication of labour market succes (binary variable).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
35	BEHINDERUNG	Indication whether a client is disabled or not (either classified by law or, in terms of placement restrictions due to health problems, by the AMS).	production variable	nominal categorical	person basic data	Person-bound information	no	redundant variable	no	no
36	BEIHILFE	Type of allowance (e.g. course costs, ancilliary course costs, livelyhood coverage, etc.).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
37	BEIHILFE_numberExpr	Total number of allowances within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
38	BEMO_TAS_BEIH	Type of "occupational mobility" measure. A special programme within the regime of basic indivdual support ("Basis Individualförderung").	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
39	D_IJG_DESC	Indicates whether or not a client with residental entitlement ("Asylbe-rechtigter") is within his or her integration year.	production variable	dichotomous categorical	labour market policy data	AMS generated data	yes		yes	yes

no.	variable name	description	type of variable	type of data	data category	data super-category	keep column	reason for omission	part of main set	part of production set
40	FDG_ID	A unique support measure identifier. One support project (with a distinct AMF_ID) can hold one or several support measures (with distinct FDG_IDs). For instance, a support project holds two different FDG_IDs, one for the course costs of the qualification measure itself (e.g. computer course) and one for ancilliary course costs (e.g. tickets for public transport).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
41	FDG_ID_ANZ	Total number of support measure IDs.	production variable	numerical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
42	FDG_ID_numberExpr	Total number of support measures IDs within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
43	FSAP_BEIH	Support measure ID within the regime of the enterprise resource planing software SAP. With the FSAP system, funding budgets are calculated and monitored.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
44	FSAP_BEIH_numberExpr	Total number of support measures in the course of the SAP regime within the pre-career.	production variable	numerical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
45	HERKUNFT	Indicates the source-application of codes stored in the variable AMP_MASSNTYP.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
46	INTEGRATION_MODUL	Type of integration module (i.e. special course measure for clients with residential entitlement).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
47	INTEGRATIONSJAHR_BEGINN	Begin date of the integration year for clients with a residential entitlement.	production variable	ordinal categorical	"funding/ promoting data"	AMS generated data	no	irrelevant for model	no	no
48	INTEGRATIONSJAHR_ENDE	End date of the integration year for clients with a residential entitlement.	production variable	ordinal categorical	"funding/ promoting data"	AMS generated data	no	irrelevant for model	no	no
49	JASG	Youth-employment programme in the course of basic executing agency support ("Basis Trägerförderung").	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
50	LEHRBERUF	Apprenticeship occupation.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
51	LEHRJAHR	Apprenticeship year.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
52	LZBL	Indication of long-term unemployement.	production variable	dichotomous categorical	labour market policy data	AMS generated data	no	redundant variable	no	no
53	MASSN_TYP	Type of support measure. Examples are job orientation ("Berufsorientierung"), qualification, and work training ("Arbeitstraining").	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	no information value	no	no
54	MASSN_TYP_numberExpr	Total number of different support measures within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	no	no information value	no	no
55	MIG_NATION	Country of origin for clients with a migration background (code "1" for first generation migrant, code "2" for second generation migrant).	production variable	nominal categorical	person basic data	Person-bound information	yes		yes	yes
56	PRUEFUNGSERFOLG	Exam success for qualification measures.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
57	PRUEFUNGSERFOLG_ number_of_Exam.successes	Total number of exam successes for qualification measures within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	no
58	PRUEFUNGSERFOLG_ number_of_Exam.failures	Total number of exam failures for qualification measures within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	no
59	PRUEFUNGSERFOLG_number_of_ Exam.terminations	Total number of exam terminations for qualification measures within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	no
60	SCHWERPUNKT	Support focus. The purpose of this variabel was not determinable. (One assuption is that the support focus refers to different client groups, such as elderly people, youths, disabled people. This, however, was never confirmed.)	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	inconclusive information	no	no
61	SCHWERPUNKT_numberExpr	Total number of support foci within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	no	no information value	no	no
62	SDG	Initially, this variable held IDs of projects in the context of a system called "Schulungsauftrag Dienstgeber (SDG)" (training order/training projects employer). Today, this variable holds codes from the SDG application's successor, viz. "Teilnahmen-Administrations-System (TAS)" (participation administration system). Although the names of the administration systems have changed over time, the content is the same. Measures subsumed under the TAS regime are, for instance, measures for integration of clients into the second labour market or self-employment support (founding programme: "Unternehmensgründungsprogramm (UGP)").	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
63	SDG_BEENDIGUNG	Indicates how a project in the course of the TAS regime (formerly called SDG) was ended (e.g. completed, terminated etc.).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
64	SDG_BEENDIGUNG_number_ of_SDG.Course_completions	Total number of TAS course completions within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	no
65	SDG_BEENDIGUNG_number_ of_SDG.Course_termination	Total number of TAS course terminations within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	no
66	SDG_BEENDIGUNG_number_ of_SDG.Course_exclusions	Total number of TAS course exclusions within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	no
67	SDG_numberExpr	Total number of TAS courses within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
68	SONDERPR	Special programme (e.g. special programme for short-time work, youths, etc.).	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	inconclusive information	no	no
69	SPJU	Special programme for youths.	production variable	dichotomous categorical	"funding/ promoting data"	AMS generated data	no	no information value	no	no
70	STRATEGIE	Measure strategy. The purpose of this variable was not determinable.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	no information value	no	no
71	STRATEGIE_numberExpr	Total number of different measure strategies within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	no	no information value	no	no
72	TAGE_IN_BE_VOR_FDG	Days in support/councelling/employment prior to the support measure. The actual meaning of "BE" was not determinable. (Assumptions are support ("Betreuung") /receipt ("Bezug")/employment ("Beschäftigung"). This, however, was never confirmed.)	production variable	numerical	"funding/ promoting data"	AMS generated data	no	content not determinable	no	no
73	TAGE_IN_BE_VOR_FDG_max	Maximum days in support/councelling/employment prior to the support measure. The actual meaning of "BE" was not determinable. (Assumptions are support ("Betreuung")/receipt ("Bezug")/employment ("Beschäftigung"). This, however, was never confirmed.)	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	no	content not determinable	no	no
74	UEBERTRGEF	Indicates a client's risk of transitioning into the "long-term unemployed" status.	production variable	nominal categorical	labour market policy data	AMS generated data	no	redundant variable	no	no
75	VERANSTALT_NR	Course number. A unique identifier also used for encoding different instances of the same course.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
76	VERANSTALT_NR_numberExpr	Total number of different course numbers within the pre-career.	additionally created variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	no
77	VT_STATUS	Status of client/course participant with regard to a course.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes
78	WIEDERANMELDUNG_BE	Re-registration of an employment relationship with the same employer at the Main Association of Austrian Social Security Institutions ("Hauptverband der öster-reichischen Sozialversicherungsträger").	production variable	dichotomous categorical	labour market policy data	AMS generated data	no	no information value	no	no
79	WIEDEREINSTIEG	Re-entry into the labour market after a long absence. Returnees ("Wieder-einsteigerinnen") are women with no employment relationship (above the marginal wages threshold) lasting longer than 365 days between the receipt of childcare allowance ("Kinderbetreuungsgeld") and the report date + 5 days of the month this status was first assigned.	production variable	dichotomous categorical	labour market policy data	AMS generated data	yes		yes	yes
80	AKTRUH	Indicates whether a client's account ("Personenstamm") is idle or not (at the last report date).	production variable	nominal categorical	mediation activity data	AMS generated data	yes		yes	yes
81	AUSBILDUNG_HOECHST	A client's highest level of education.	production variable	nominal categorical	person qualification data	Person-bound information	yes		yes	yes
82	BEGUENSTIGUNG	Indicates whether or not a client is disabled and, therefore, privileged for a certain group of support measures.	production variable	nominal categorical	person basic data	Person-bound information	yes		yes	yes
83	BERUFSART	Type of a client's last-known occupation (stored as a 6-digit job title code).	production variable	nominal categorical	person qualification data	Person-bound information	yes		yes	yes
84	BERUFSART_WUNSCH	Career aspiration, i.e. the type of occupation as wished by the client (stored as a 6-digit job title code).	production variable	nominal categorical	person qualification data	Person-bound information	yes		yes	yes
85	BERUFSART_WUNSCH_ numberExpr	Number of differend career aspirations within the pre-career.	additionally created variable	numerical	person qualification data	Person-bound information	yes		yes	no
86	BERUFSART_numberExpr	Number of different types of occupations within the pre-career.	additionally created variable	numerical	person qualification data	Person-bound information	yes		yes	no

no.	variable name	description	type of variable	type of data	data category	data super-category	keep column	reason for omission	part of main set	part of production set
87	BESCHAEFTIGUNG_AB	Date on which a client starts an employment relationship.	production variable	ordinal categorical	mediation activity data	AMS generated data	no	inconclusive information	no	no
88	BMS	Indicates whether or not and to which extent a client receives demand-oriented minimum income ("Bedarfsorientierte Mindessicherung").	production variable	nominal categorical	transfer payment data	AMS generated data	no	inconclusive information	no	no
89	DEUTSCHKENNTNISSE	German language skills based on CEFR (Common European Framework of Reference for Languages), extended by supergroups A, B, and C.	production variable	nominal categorical	person qualification data	Person-bound information	yes		yes	yes
90	EINSTELLZUSAGE_DAT	Start date of a job offer with a deferred start date.	production variable	ordinal categorical	mediation activity data	AMS generated data	no	inconclusive information	no	no
91	EINSTELLZUSAGE_DAT_ numberExpr	Total number of different start dates of job offers with deferred start dates within the pre-career.	additionally created variable	numerical	mediation activity data	AMS generated data	yes		yes	no
92	FAMSTAND	A client's marital status.	production variable	nominal categorical	person basic data	Person-bound information	yes		yes	yes
93	GESCHAEFTSSTELLE	Local branch of the Austrian Employment Service (AMS).	production variable	nominal categorical	person basic data	Person-bound information	yes		yes	yes
94	GESCHAEFTSSTELLE_ numberExpr	Total number of a client's allocations to a local branch of the Austrian Employment Serivce (AMS).	additionally created variable	numerical	person basic data	Person-bound information	yes		yes	no
95	GESCHLECHT	Client's sex.	production variable	nominal categorical	person basic data	Person-bound information	yes		yes	yes
96	GKZ	District code of a client's residence.	production variable	nominal categorical	person basic data	Person-bound information	yes		yes	yes
97	NACE	Industry identifer based on the Austrian ÖNACE classification, which is based on NACE ("Nomenclature statistique des activités économiques dans la Communauté européenne"). A client's NACE classification is inhereted by his/her employer's NACE allocation.	production variable	nominal categorical	job-related person data	Person-bound information	yes		yes	yes
98	NACE_numberExpr	Total number of a client's different NACE assignments within the pre-career.	additionally created variable	numerical	job-related person data	Person-bound information	yes		yes	no
99	NATIONALITAET	Country code. Country of origin of clients with non-Austrian citizenship.	production variable	nominal categorical	person basic data	Person-bound information	yes		yes	yes
100	PLZ	Postal code of client's residence.	production variable	nominal categorical	person basic data	Person-bound information	no	redundant variable	no	no
101	PST_ABV_ANZ_AEA	Number of work permits ("Arbeitserlaubnis") for employed foreigners.	production variable	numerical	job-related person data	Person-bound information	no	no information value	no	no
102	PST_ABV_ANZ_BSA	Number of exception certificates ("Befreiungsschein") for employed foreigners.	production variable	numerical	job-related person data	Person-bound information	no	inconclusive information	no	no
103	PST_ABV_BEVACO1	Indication of preferred foreigners ("bevorzugte Ausländer").	production variable	nominal categorical	job-related person data	Person-bound information	yes		yes	yes
104	PST_ABV_BEVACO2	Degree of intregration of preferred foreigners ("bevorzugte Ausländer").	production variable	nominal categorical	job-related person data	Person-bound information	yes		yes	yes
105	PST_ABV_BEVAUSD	Date of adjudication of a client's status as a preferred foreigner ("bevorzugter Ausländer").	production variable	nominal categorical	job-related person data	Person-bound information	no	inconclusive information	no	no
106	VERMV_MIT_POST	Indication whether or not placement proposals ("Vermittlungsvorschläge") can be transmitted to a client by mail.	production variable	nominal categorical	mediation activity data	AMS generated data	no	irrelevant for model	no	no
107	WST_AUSMASS_BIS	Maximum extent of possible working hours per week.	production variable	numerical	job-related person data	Person-bound information	no	inconclusive information	no	no
108	WST_AUSMASS_VON	Minimum extent of possible working hours per week.	production variable	numerical	job-related person data	Person-bound information	no	inconclusive information	no	no
109	ZONEN	Allocation of a client to a department of a local branch of the Austrian Employment Service (AMS).	production variable	nominal categorical	mediation activity data	AMS generated data	no	irrelevant for model	no	no
110	ALTR	Client's age (in years).	production variable	ordinal categorical	person basic data	Person-bound information	yes		yes	yes
111	ANZ_EPI_AG	Number of episodes in the labour market policy status "AG" (i.e. clarification of working ability), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	no	strong correlation*	no	no
112	ANZ_EPI_AL	Number of episodes in the labour market policy status "AL" (i.e. unemployed), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
113	ANZ_EPI_LS	Number of episodes in the labour market policy status "LS" (i.e. clients looking for an apprenticeship), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	no	strong correlation*	no	no
114	ANZ_EPI_SC	Number of episodes in the labour market policy status "SC" (i.e. qualification measures initiated by the AMS), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
115	ANZ_EPI_SF	Number of episodes in the labour market policy status "SF" (i.e. specialist's scholarship: "Fachkräftestipendium"), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
116	ANZ_EPI_SR	Number of episodes in the labour market policy status "SR" (i.e. qualification with re-education in the context of rehabilitation), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
117	ANZ_TAGE_AG	Number of days in the labour market policy status "AG" (i.e. clarification of one's working ability), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
118	ANZ_TAGE_AL	Number of days in the labour market policy status "AL" (i.e. unemployed), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	no	strong correlation*	no	no
119	ANZ_TAGE_LS	Number of days in the labour market policy status "LS" (i.e. clients looking for an apprenticeship), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
120	ANZ_TAGE_SC	Number of days in the labour market policy status "SC" (i.e. qualification measures initiated by the AMS), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
121	ANZ_TAGE_SF	Number of days in labour market policy status "SF" (i.e. specialist's scholarship - "Fachkräftestipendium") at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
122	ANZ_TAGE_SR	Number of days in the labour market policy status "SR" (i.e. qualification with re-education in the context of rehabilitation), at the last report date.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
123	ARB_ZTA	Type of the employment relationship (i.e. full-time, part-time, or both).	production variable	nominal categorical	job-related person data	Person-bound information	yes		yes	yes
124	ASYL_CODE	Type of residential entitlement (i.e. convention refugee or subsidiary protection).	production variable	nominal categorical	job-related person data	Person-bound information	yes		yes	yes
125	ASYL_ZUERKENNUNG	Adjudication date of a client's residential entitlement.	production variable	ordinal categorical	job-related person data	Person-bound information	no	irrelevant for model	no	no
126	DAUER_BIS_FF	Mean duration in days until receipt of support measure.	production variable	numerical	"funding/ promoting data"	AMS generated data	no	irrelevant for model	no	no
127	DAUER_BIS_VV	Mean duration in days until receipt of placement proposal ("Vermittlungsvorschlag").	production variable	numerical	mediation activity data	AMS generated data	no	irrelevant for model	no	no
128	DAUER_BRUTTO_BIS_FF	Mean gross duration in days until receipt of support measure.	production variable	numerical	"funding/ promoting data"	AMS generated data	no	irrelevant for model	no	no
129	DAUER_BRUTTO_BIS_VV	Mean gross duration in days until receipt of placement proposal ("Vermittlungs-vorschlag").	production variable	numerical	mediation activity data	AMS generated data	no	irrelevant for model	no	no
130	DV_DAUR	Extent of employment relationship (e.g. temporary, permanent, seasonal, etc.).	production variable	nominal categorical	job-related person data	Person-bound information	yes		yes	yes
131	EAMS_KONTO	Status of a client's electronic AMS ("eAMS") account.	production variable	nominal categorical	mediation activity data	AMS generated data	yes		yes	yes
132	EIN_ZUS	Indication of a job offer with a deferred start date.	production variable	nominal categorical	job-related person data	Person-bound information	yes		yes	yes
133	EINSTELLGRUND	Reason for termination of monetary benefit and/or support.	production variable	nominal categorical	transfer payment data	AMS generated data	yes		yes	yes
134	EINSTELLGRUND_numberExpr	Total number of different reasons for the termination of monetary benefit and/or support within the pre-career.	additionally created variable	numerical	transfer payment data	AMS generated data	yes		yes	no
135	EINSTELLGRUND_number_of_ _missed_Appointments	Total number of missed appointments with the AMS within the pre-career.	additionally created variable	numerical	transfer payment data	AMS generated data	yes		yes	no
136	FF_12M	Number of support measures within the last 12 months (from last report date).	production variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	yes
137	FF_1M	Number of support measures within the last month (from last report date).	production variable	numerical	"funding/ promoting data"	AMS generated data	no	redundant variable	no	no
138	FF_99M	Number of support measures within the last 99 months (from last report date).	production variable	numerical	"funding/ promoting data"	AMS generated data	no	strong correlation	no	no
139	FF_INSGESAMT	Total number of support measures (from last report date).	production variable	numerical	"funding/ promoting data"	AMS generated data	yes		yes	yes
140	FF_INTERVALL	Interval between support measures applied to a client (in days).	production variable	numerical	"funding/ promoting data"	AMS generated data	no	irrelevant for model	no	no
141	FF_MASSNAHME	Type of support measure.	production variable	nominal categorical	"funding/ promoting data"	AMS generated data	no	replaced by FUNDINGS_ RECEIVED	no	no
142	GF_BRUTTO__BISHER	Gross duration of a business case (at the last report date). A business case starts with an AL-, AG-, LS-, SC-, SF- or SR-episode and ends with inter-ruptions greater than 62 days. Consequently, the end of a business case can only be determined 3 months later. In terms of gross duration of a business case, interruptions of up to 62 days remain unconsidered. Interruptions are, for instance, short unemployment relationships, cases of illness, etc.	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
143	GF_BRUTTO_ABGANG	Gross duration of a business case until a client's exit from AMS services (e.g. in the case of finding a job).	production variable	numerical	labour market policy data	AMS generated data	yes		yes	yes
144	GF_EPI_NETTO	Net duration of a current business case episode (at last report date).	production variable	numerical	labour market policy data	AMS generated data	no	redundant variable	no	no
145	GF_NETTO_ABGANG	Net duration of a business case until a client's exit from AMS services (e.g. in the case of finding a job).	production variable	numerical	labour market policy data	AMS generated data	no	strong correlation*	no	no

no.	variable name	description	type of variable	type of data	data category	data super-category	keep column	reason for omission	part of main set	part of production set
146	GF_NETTO_BISHER	Net duration of a business case (at the last report date). A business case starts with an AL-, AG-, LS-, SC-, SF- or SR-episode and ends with inter-ruptions greater than 62 days. Concequently, the end of a business case can only be determined 3 months later. In terms of net duration of a business case, interruptions of up to 62 days are deducted from the total duration of the business case. Interruptions are, for instance, short unemployment relationships, cases of illness etc.	production variable	numerical	labour market policy data	AMS generated data	no	strong correlation*	no	no
147	KONTAKT_12M	Number of client contacts within the last 12 months (from last report date).	production variable	numerical	mediation activity data	AMS generated data	yes		yes	yes
148	KONTAKT_1M	Number of client contacts within last month (from last report date).	production variable	numerical	mediation activity data	AMS generated data	no	redundant variable	no	no
149	KONTAKT_99M	Number of client contacts within the last 99 months (from last report date).	production variable	numerical	mediation activity data	AMS generated data	no	strong correlation	no	no
150	KONTAKT_INSGESAMT	Total number of client contacts (at last report date).	production variable	numerical	mediation activity data	AMS generated data	no	strong correlation*	no	no
151	KONTAKT_INTERVALL	Interval of client contacts (in days).	production variable	numerical	mediation activity data	AMS generated data	yes		yes	yes
152	LZBL_VON	Date of a client's status of being "long-term unemployed".	production variable	ordinal categorical	labour market policy data	AMS generated data	no	redundant variable	no	no
153	ONL_STATUS_SEIT	Date of a client's status of being supported online.	production variable	ordinal categorical	mediation activity data	AMS generated data	no	inconclusive information	no	no
154	PST_STATUS	Labour market policy status according to the Austrian Employment Service (AMS) status directive. In contrast to the Universal Labour Market Statuses ("Uni-Statuses"), which combine data of several sources, these (AMS) labour market policy statuses solely cover AMS generated data and distinguishes 13 different statuses.	production variable	nominal categorical	labour market policy data	AMS generated data	yes		yes	yes
155	STUDRI	Client's field of study.	production variable	nominal categorical	person qualification data	Person-bound information	yes		yes	yes
156	VERF	Client's availabiliy (i.e. immediately available or not immediately available).	production variable	nominal categorical	person basic data	Person-bound information	yes		yes	yes
157	VV_12M	Number of placement proposals ("Vermittlungsvorschläge") within the last 12 months (from last report date).	production variable	numerical	mediation activity data	AMS generated data	yes		yes	yes
158	VV_1M	Number of placement proposals ("Vermittlungsvorschläge") within the last month (from last report date).	production variable	numerical	mediation activity data	AMS generated data	no	redundant variable	no	no
159	VV_99M	Number of placement proposals ("Vermittlungsvorschläge") within the last 99 months (from last report date).	production variable	numerical	mediation activity data	AMS generated data	no	strong correlation	no	no
160	VV_INSGESAMT	Total number of placement proposals ("Vermittlungsvorschläge") (at last report date).	production variable	numerical	mediation activity data	AMS generated data	yes		yes	yes
161	VV_INTERVALL	Interval of placement proposals ("Vermittlungsvorschläge") (at last report date, in days).	production variable	numerical	mediation activity data	AMS generated data	no	irrelevant for model	no	no
162	ABKOMMEN	Cross-national agreements concering the transfer of unemployment benefits from other countries to Austria. This applies to cases in which a client with a non-Austrian citizenship cannot fulfill the entitlement for unemployment benefits in Austria.	production variable	nominal categorical	job-related person data	Person-bound information	no	inconclusive information	no	no
163	ANRECHBETRAG	Chargeable amount of a partner's income, creditable against a client's monetary benefit.	production variable	numerical	transfer payment data	AMS generated data	yes		yes	yes
164	ANZAHLFZ	Number of family allowances ("Familienzuschläge") (at last report date). This allowance enhances the daily benefit and is granted for children, stepchildren and foster children (provided there are care responsibilites).	production variable	numerical	person basic data	Person-bound information	no	strong correlation*	no	yes
165	ANZAHLFZ_max	Maximum number of family allowances ("Familienzuschläge") within the pre-career.	additionally created variable	numerical	person basic data	Person-bound information	yes		yes	no
166	BEZUGSDAUER	Duration of benefit receipt (at last report date).	production variable	numerical	transfer payment data	AMS generated data	no	irrelevant for model	no	no
167	BEZUGSEINSTELLUNG	Indicates state of benefit receipt (at report date).	production variable	nominal categorical	transfer payment data	AMS generated data	no	redundant variable	no	no
168	DLUANTEIL	Amount of livelyhood coverage ("Deckung des Lebensunterhaltes"). This is a special monetary benefit during qualification measures for clients with small or no unemployment benefit ("Arbeitslosengeld") or unemployment assistance ("Notstands-hilfe").	production variable	numerical	transfer payment data	AMS generated data	yes		yes	yes
169	GRUNDBETRAG	Basic amount of a client's income, used as the foundation for calculating benefit receipt.	production variable	numerical	person basic data	Person-bound information	yes		yes	yes
170	KRANKENVSTR	A client's health insurance agency.	production variable	nominal categorical	transfer payment data	AMS generated data	no	irrelevant for model	no	no
171	LB_MIT_ANRECH	Indicates whether or not a client's benefit receipt is reduced due to a chargeable amount of a partner's income.	production variable	dichotomous categorical	transfer payment data	AMS generated data	yes		yes	yes
172	LB_MIT_DLUART	Indicates whether or not a client's benefit receipt is enhanced by lifelyhood coverage ("Deckung des Lebensunterhaltes").	production variable	dichotomous categorical	transfer payment data	AMS generated data	yes		yes	yes
173	LB_MIT_FZ	Indicates whether or not a client's benefit receipt is enhanced by family allowances ("Familienzuschläge").	production variable	dichotomous categorical	transfer payment data	AMS generated data	yes		yes	yes
174	LB_MIT_KNK	Indicates whether or not a client's benefit receipt is enhanced by ancilliary course costs ("Kursnebenkosten").	production variable	dichotomous categorical	transfer payment data	AMS generated data	yes		yes	yes
175	LB_MIT_PAUKNK	Indicates whether or not a client's benefit receipt is enhanced by fixed-rate ancilliary course costs ("pauschale Kursnebenkosten").	production variable	dichotomous categorical	transfer payment data	AMS generated data	yes		yes	yes
176	LEISTUNGSART	Indicates the type of a client's monetary benefit receipt.	production variable	nominal categorical	transfer payment data	AMS generated data	yes		yes	yes
177	LEISTUNGSART_numberExpr	Total number of different types of a client's monetary benefit receipt.	additionally created variable	numerical	transfer payment data	AMS generated data	yes		yes	no
178	LOHNKLASSE	The wage category a client is assigned to.	production variable	nominal categorical	person basic data	Person-bound information	no	inconclusive information	no	no
179	NHREDUKTION	Reduction of a client's monetary benefit "unemployment assistance" ("Notstands-hilfe") relative to the former receipt of unemployment benefit ("Arbeitslosengeld"). Dependent on the amount of a client's unemployment benefit receipt, the subsequent unemployment assistance is either 95% or 92% of the unemployment benefit.	production variable	nominal categorical	transfer payment data	AMS generated data	no	redundant variable	no	no
180	PAUKNK	Amount of fixed-rate ancilliary course costs ("pauschale Kursnebenkosten").	production variable	nominal categorical	transfer payment data	AMS generated data	no	irrelevant for model	no	no
181	PENSIONVSTR	A client's retirement insurance agency.	production variable	nominal categorical	transfer payment data	AMS generated data	no	irrelevant for model	no	no
182	PKZ	Indicates the category of livelyhood coverage ("Deckung des Lebensunter-haltes") a client is entitled to.	production variable	nominal categorical	transfer payment data	AMS generated data	yes		yes	yes
183	TAGSATZLEIST	A client's daily benefit rate.	production variable	numerical	transfer payment data	AMS generated data	no	strong correlation*	no	no
184	TAGSATZLEIST_max	A client's maximum amount of daily benefit rate within the pre-career.	additionally created variable	numerical	transfer payment data	AMS generated data	no	strong correlation	no	no
185	TAGSATZLEIST_mean	A client's mean amount of the daily benefit rate within the pre-career.	additionally created variable	numerical	transfer payment data	AMS generated data	no	strong correlation	no	no
186	TAGSATZNEBENL	A client's daily ancilliary benefit rate.	production variable	numerical	transfer payment data	AMS generated data	yes		yes	yes
187	TLD_ALNH_28	Total number of days of benefit receipt (at report date), i.e. unemployment benefit ("Arbeitslosengeld") or unemployment assistance ("Notstandshilfe"), without interruptions of benefit receipt greater than 28 days.	production variable	numerical	transfer payment data	AMS generated data	no	irrelevant for model	no	no
188	TLD_ALNH_62	Total number of days of benefit receipt (at report date), i.e. unemployment benefit ("Arbeitslosengeld") or unemployment assistance ("Notstandshilfe"), without interruptions of benefit receipt greater than 62 days.	production variable	numerical	transfer payment data	AMS generated data	yes		yes	yes
189	TLD_LA_28	Total number of days of benefit receipt (at report date) for a certain type of benefit, without interruptions of benefit receipt greater than 28 days.	production variable	numerical	transfer payment data	AMS generated data	no	content not determinable	no	no
190	TLD_LA_62	Total number of days of benefit receipt (at report date) for a certain type of benefit, without interruptions of benefit receipt greater than 62 days.	production variable	numerical	transfer payment data	AMS generated data	no	content not determinable	no	no
191	TLD_LB_28	Total number of days of benefit receipt (at report date) for all types of benefit, without interruptions of benefit receipt greater than 28 days.	production variable	numerical	transfer payment data	AMS generated data	no	irrelevant for model	no	no
192	TLD_LB_62	Total number of days of benefit receipt (at report date) for all types of benefit, without interruptions of benefit receipt greater than 62 days.	production variable	numerical	transfer payment data	AMS generated data	yes		yes	yes
193	KRE_BETRKKCODE	A client's company insurance agency ("Betriebskrankenkasse").	production variable	nominal categorical	transfer payment data	AMS generated data	no	irrelevant for model	no	no
194	KRE_EINSTCODE	Reason for termination of monetary benefit and/or support (similar to EINSTELLGRUND, but without any entries).	production variable	nominal categorical	transfer payment data	AMS generated data	no	no information value	no	no
195	FUNDING_RECEIVED	Indication whether or not a funding/support measure was applied to a client.	additionally created variable	dichotomous categorical	"funding/ promoting data"	AMS generated data	yes		yes	yes

Table 4: *Basic set* of variables.

D Appendix: Results

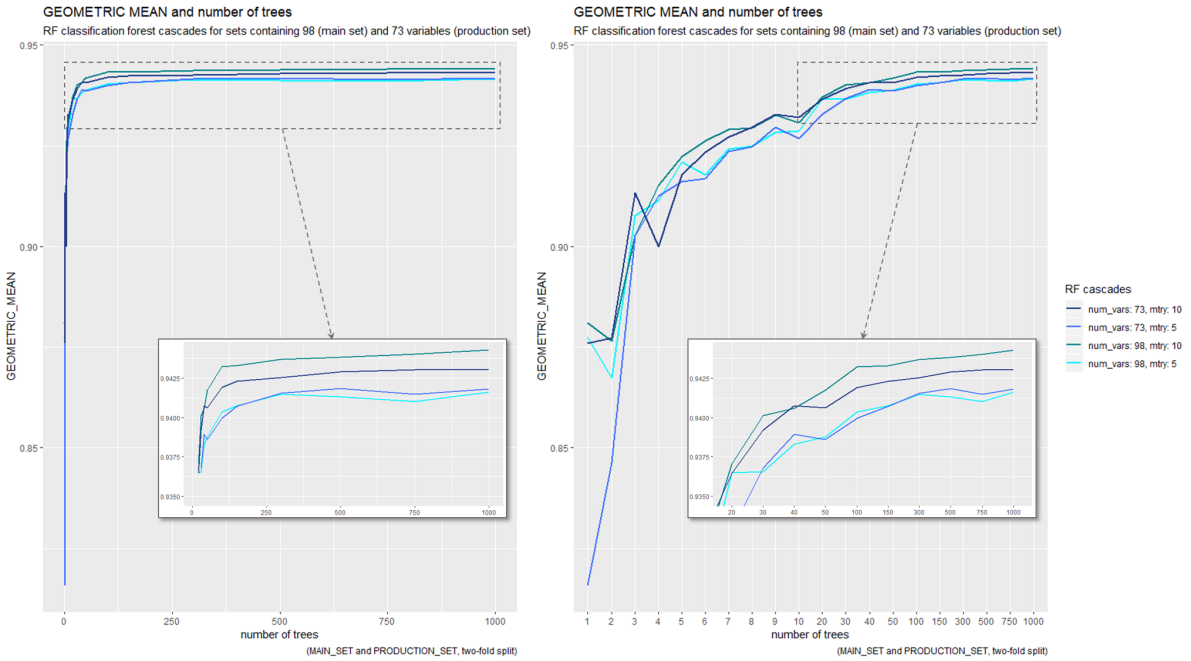

Figure 48: Initial series of *classification forests* for the *main set* (98 variables) and the *production set* (73 variables). While the left plot on the x-axis shows a continuous number of trees, the right plot displays the actual discrete values. The y-axis displays the performance measure *geometric mean (GM)*. For random forests of a size of 20 trees or more, the larger *main set* performed better than the smaller *production set*. Moreover, from this forest size onwards, *mtry* 10 performed better than *mtry* 5.

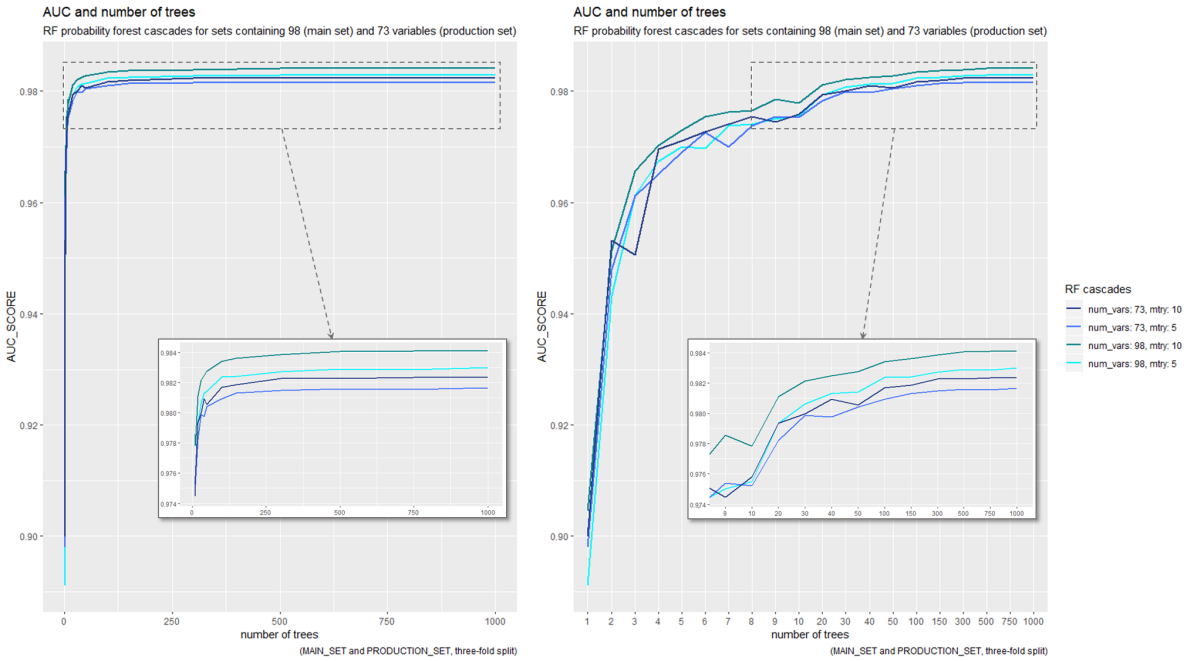

Figure 49: Initial series of *probability forests* for the *main set* (98 variables) and the *production set* (73 variables). While the left plot on the x-axis shows a continuous number of trees, the right plot displays the actual discrete values. The y-axis displays the performance measure *area under the ROC curve (AUC)*. For random forests of a size of 20 trees or more, the larger *main set* performed better than the smaller *production set*. Moreover, from this forest size onwards, *mtry* 10 performed better than *mtry* 5.

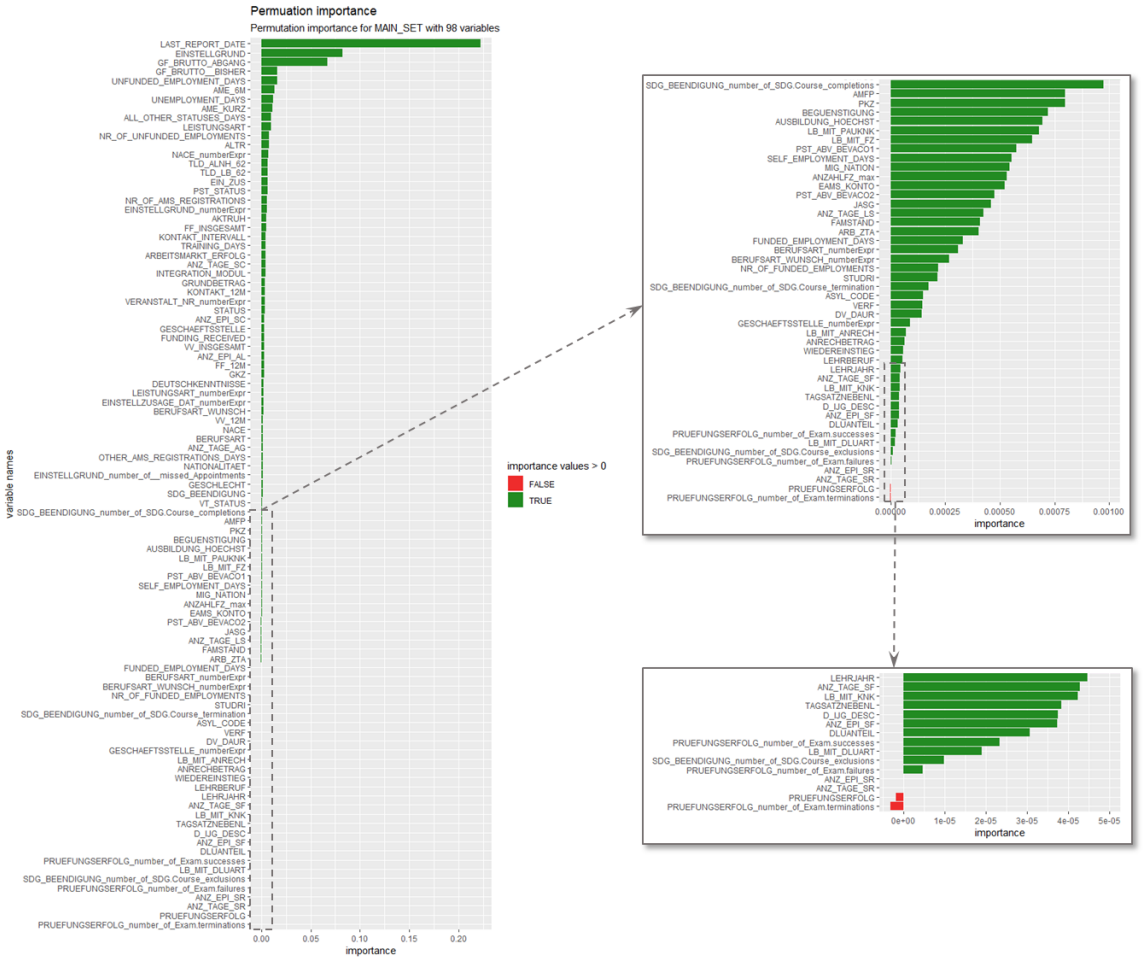

Figure 50: *Variable importance* for variables of the *main set* (98 variables). Breiman's *permutation importance* [18] revealed four variables exhibiting negative importance values, viz. `ANZ_EPI_SR`, `ANZ_TAGE_SR`, `PRUEFUNGSERFOLG`, and `PRUEFUNGSERFOLG_number_of_Exam.terminations`. After removing these variables from the data set, the performance of random forests with 300+ trees slightly increased (see figure 16, section 5.2).

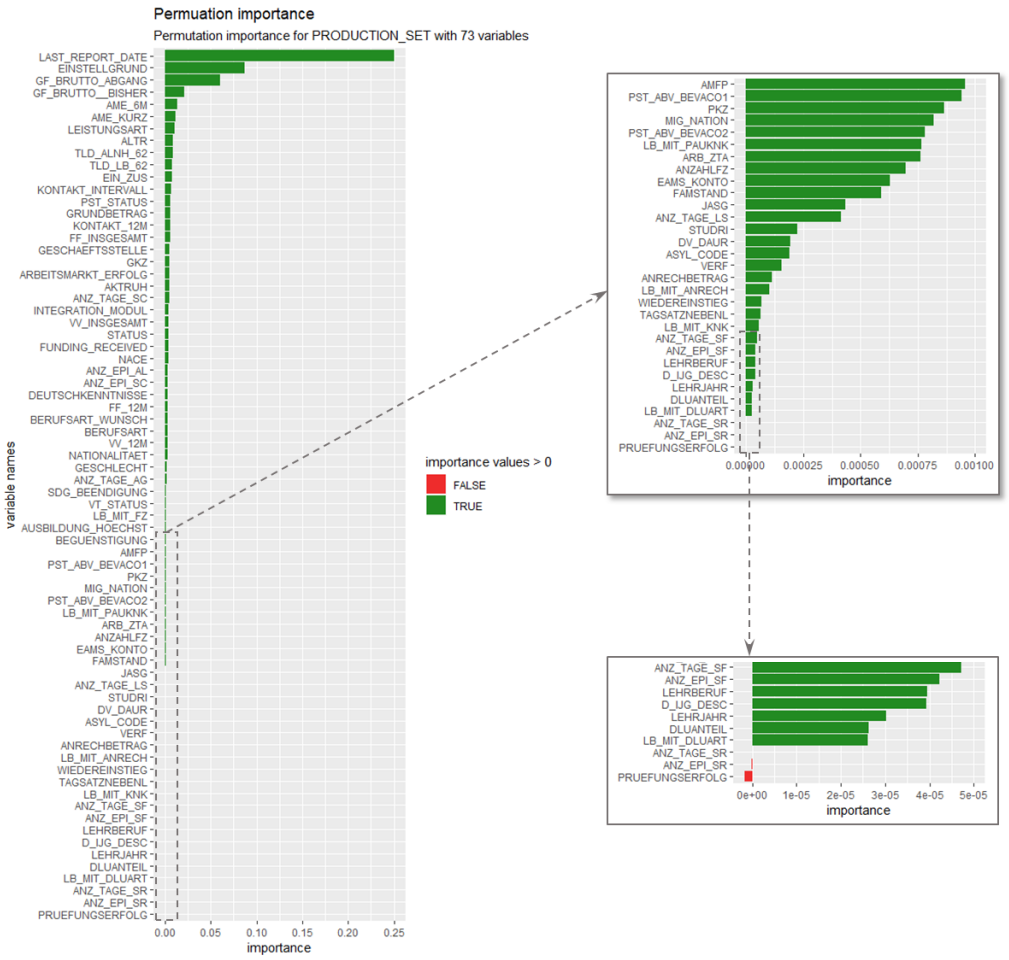

Figure 51: *Variable importance* for variables of the *production set* (73 variables). Breiman's *permutation importance* [18] revealed two variables exhibiting negative importance values, viz. `ANZ_EPI_SR` and `PRUEFUNGSERFOLG`. After removing these variables from the data set, the performance of random forests with 300+ trees slightly increased (see figure 17, section 5.2).

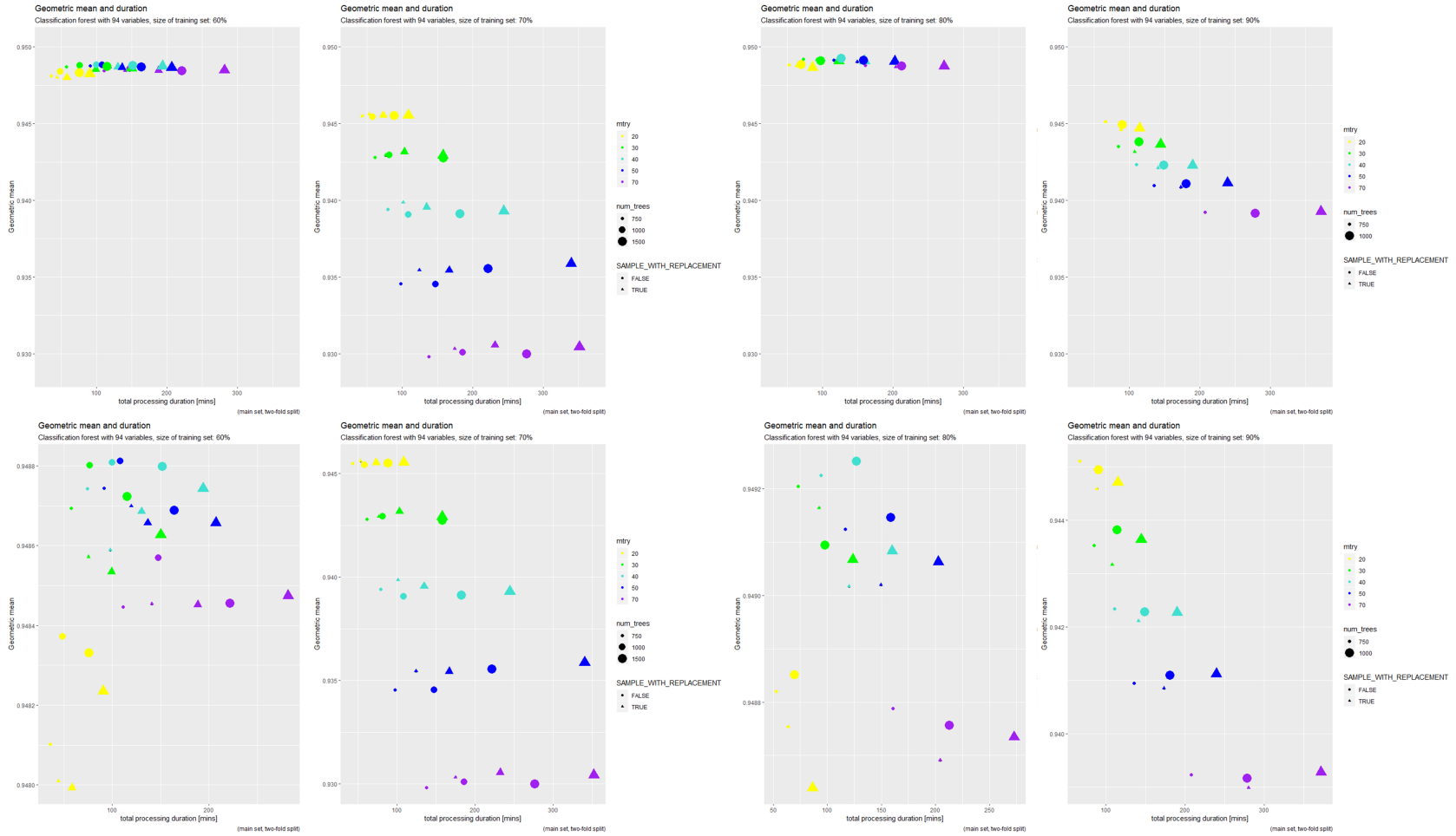

Figure 52: *Tuning level 2*: Performance of *classification forest* models created from the *main set*. The upper four plots are the same as those displayed in figure 19. They show the five best-performing *mtry* values from the preceding stage, calculated for four different *training set sizes* (60%, 70%, 80%, and 90%). While the upper four plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

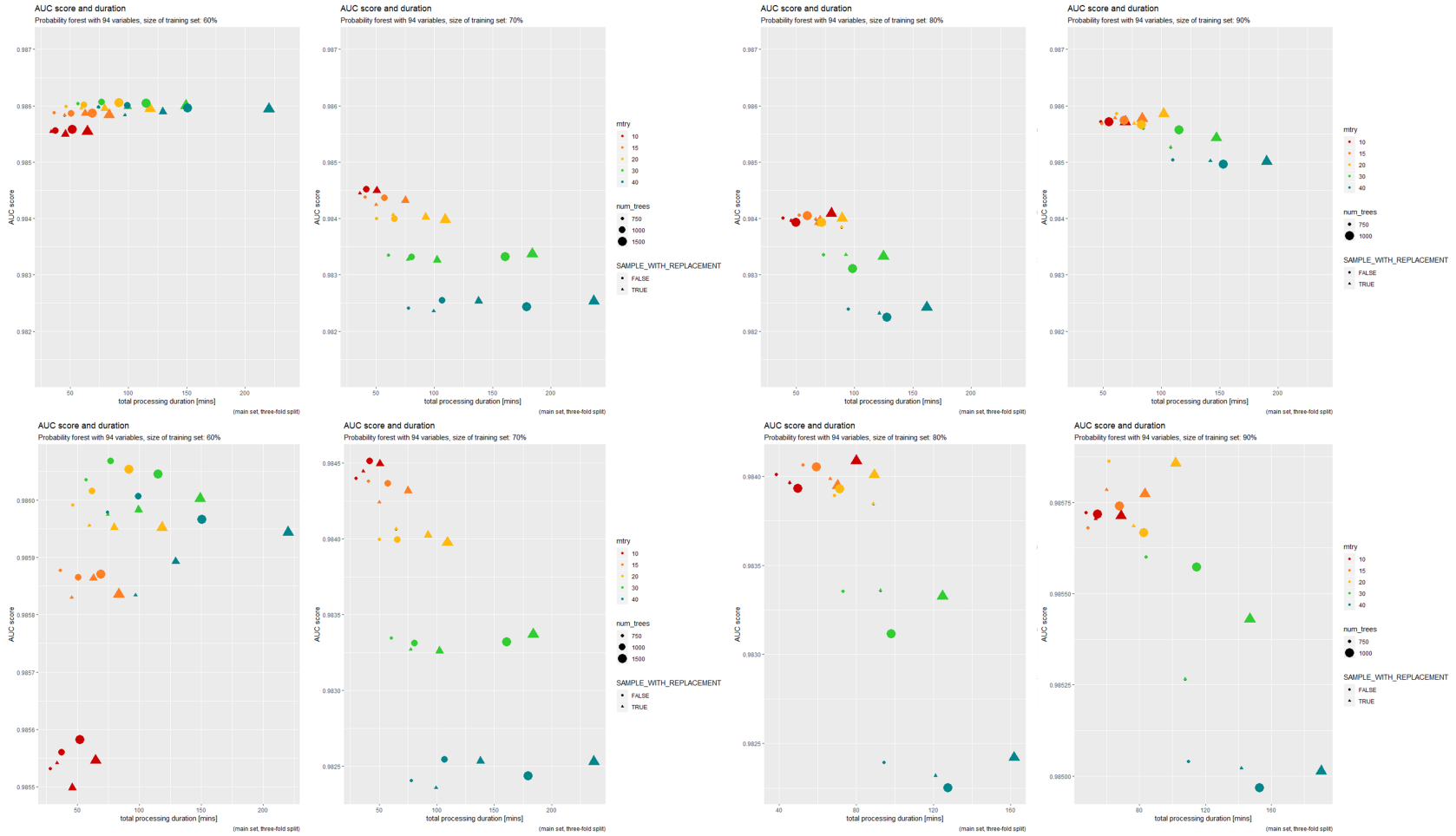

Figure 53: *Tuning level 2*: Performance of *probability forest* models created from the *main set*. The upper four plots are the same as those displayed in figure 20. They show the five best-performing *mtry* values from the preceding stage, calculated for four different *training set sizes* (60%, 70%, 80%, and 90%). While the upper four plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

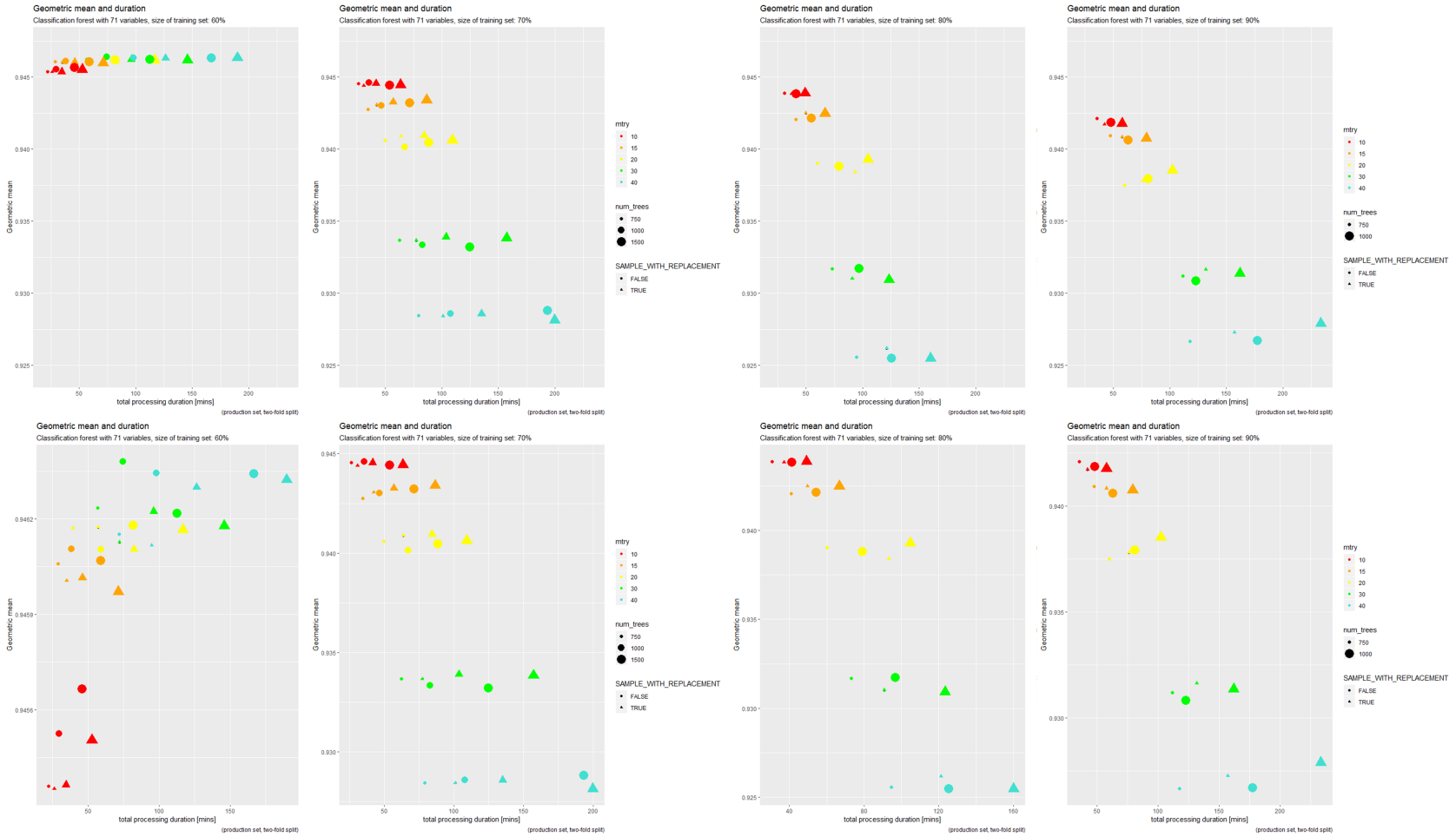

Figure 54: *Tuning level 2*: Performance of *classification forest* models created from the *production set*. The upper four plots are the same as those displayed in figure 21. They show the five best-performing *mtry* values from the preceding stage, calculated for four different *training set sizes* (60%, 70%, 80%, and 90%). While the upper four plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

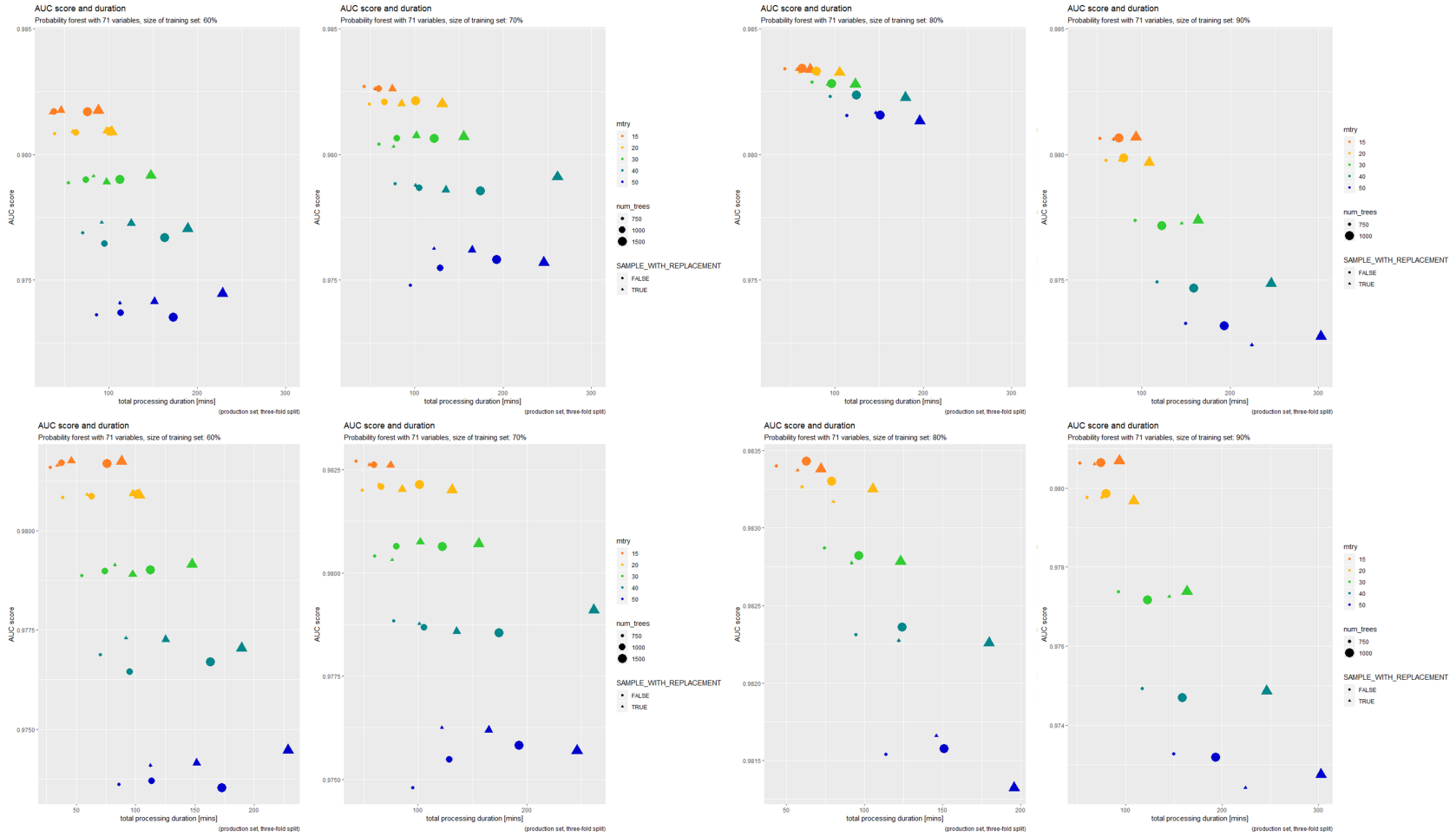

Figure 55: *Tuning level 2*: Performance of *probability forest* models created from the *production set*. The upper four plots are the same as those displayed in figure 22. They show the five best-performing *mtry* values from the preceding stage, calculated for four different *training set sizes* (60%, 70%, 80%, and 90%). While the upper four plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

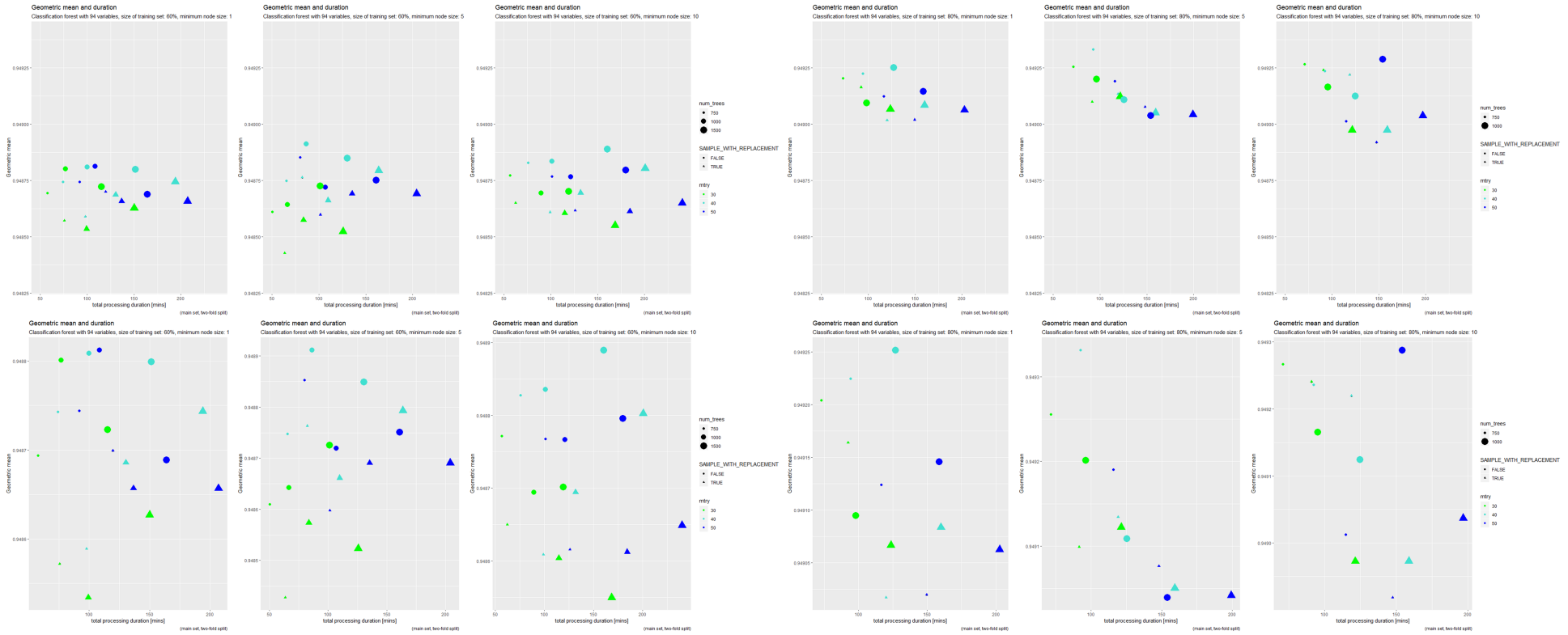

Figure 56: *Tuning level 3*: Performance of *classification forest* models created from the *main set*. For the two best-performing *mtry* values from the two best-performing *training set sizes* of the preceding stage, three different *minimum node sizes* were calculated (1 = default value, 5 and 10). The model performances for the default *node size* were taken from the former *tuning level 2*. The upper six plots are the same as those displayed in figure 23 and represent the three different *node size* values for a *training set size* of 60% (plots on the left) and 80% (plots on the right). While the upper six plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

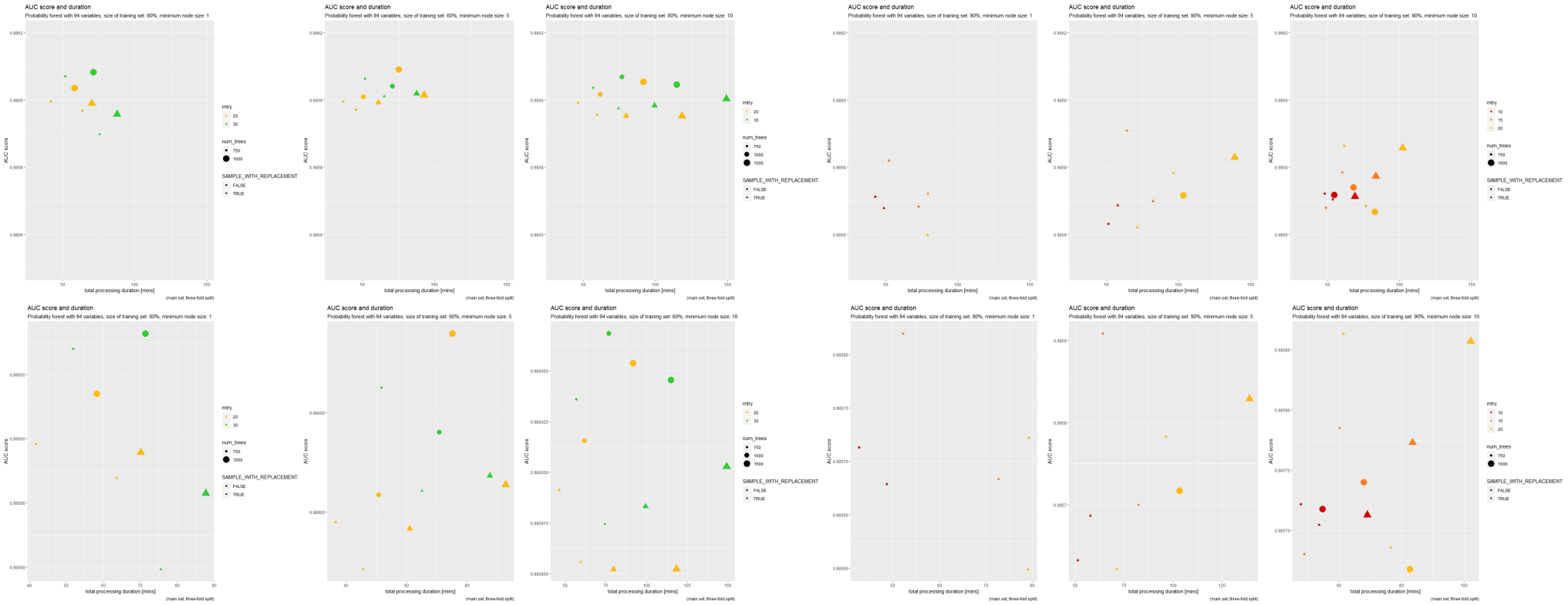

Figure 57: *Tuning level 3*: Performance of *probability forest* models created from the *main set*. For the two best-performing *mtry* values from the two best-performing *training set sizes* of the preceding stage, three different *minimum node sizes* were calculated (1, 5, and 10 = default value). The model performances for the default *node size* were taken from the former *tuning level 2*. The upper six plots are the same as those displayed in figure 24 and represent the three different *node size* values for a *training set size* of 60% (plots on the left) and 90% (plots on the right). While the upper six plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

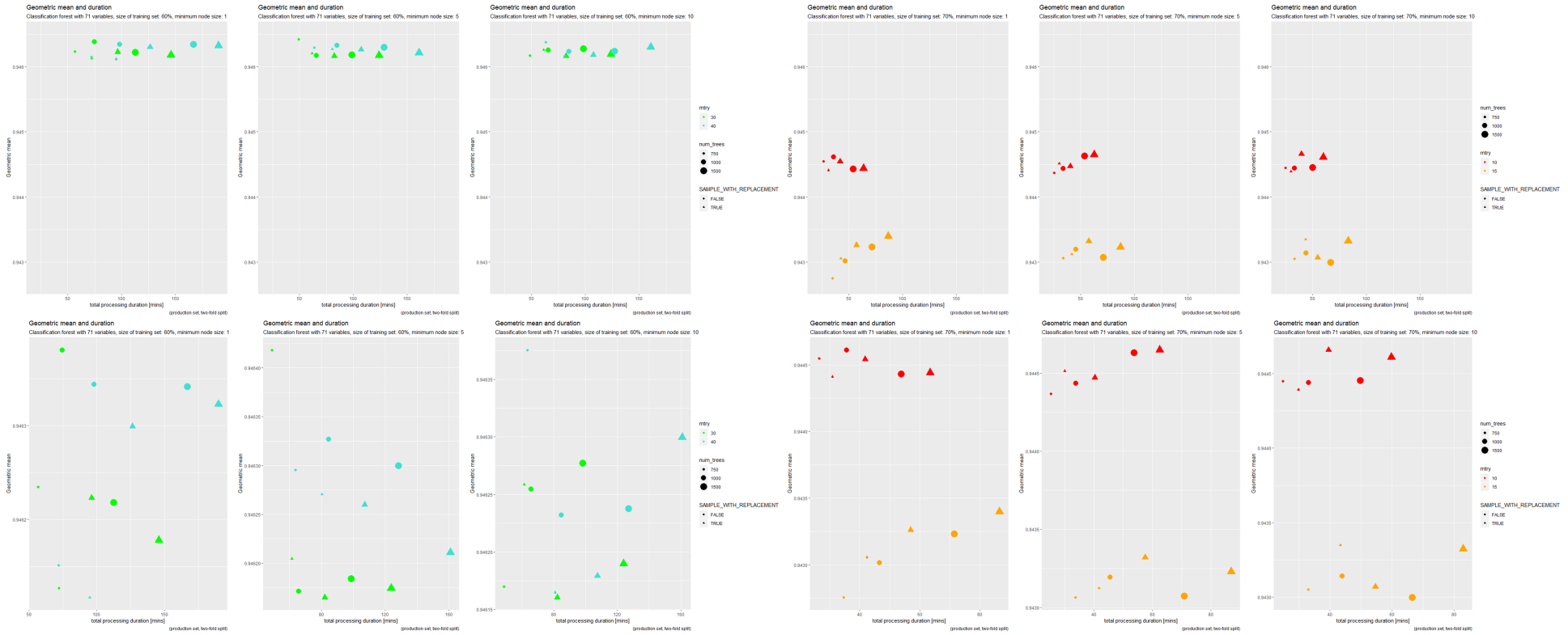

Figure 58: *Tuning level 3*: Performance of *classification forest* models created from the *production set*. For the two best-performing *mtry* values from the two best-performing *training set sizes* of the preceding stage, three different *minimum node sizes* were calculated (1 = default value, 5 and 10). The model performances for the default *node size* were taken from the former *tuning level 2*. The upper six plots are the same as those displayed in figure 25 and represent the three different *node size* values for a *training set size* of 60% (plots on the left) and 70% (plots on the right). While the upper six plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

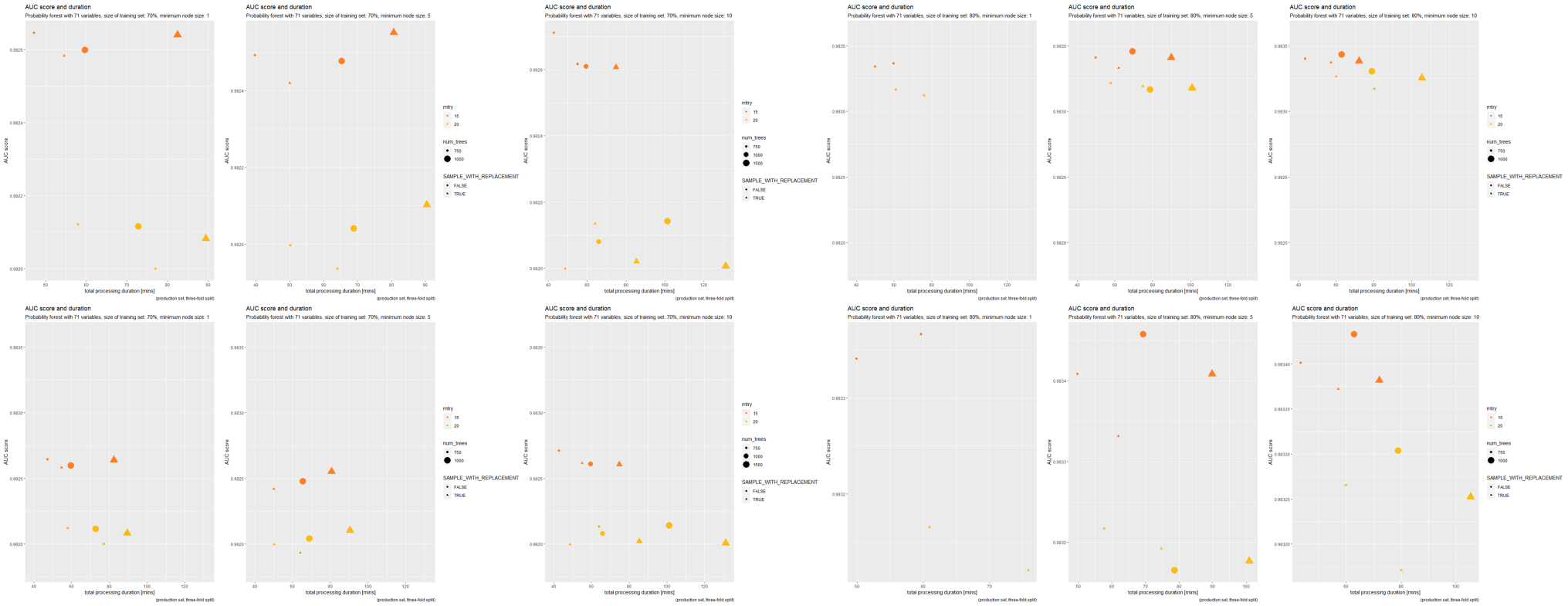

Figure 59: *Tuning level 3*: Performance of *probability forest* models created from the *production set*. For the two best-performing *mtry* values from the two best-performing *training set sizes* of the preceding stage, three different *minimum node sizes* were calculated (1, 5, and 10 = default value). The model performances for the default *node size* were taken from the former *tuning level 2*. The upper six plots are the same as those displayed in figure 26 and represent the three different *node size* values for a *training set size* of 70% (plots on the left) and 80% (plots on the right). While the upper six plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

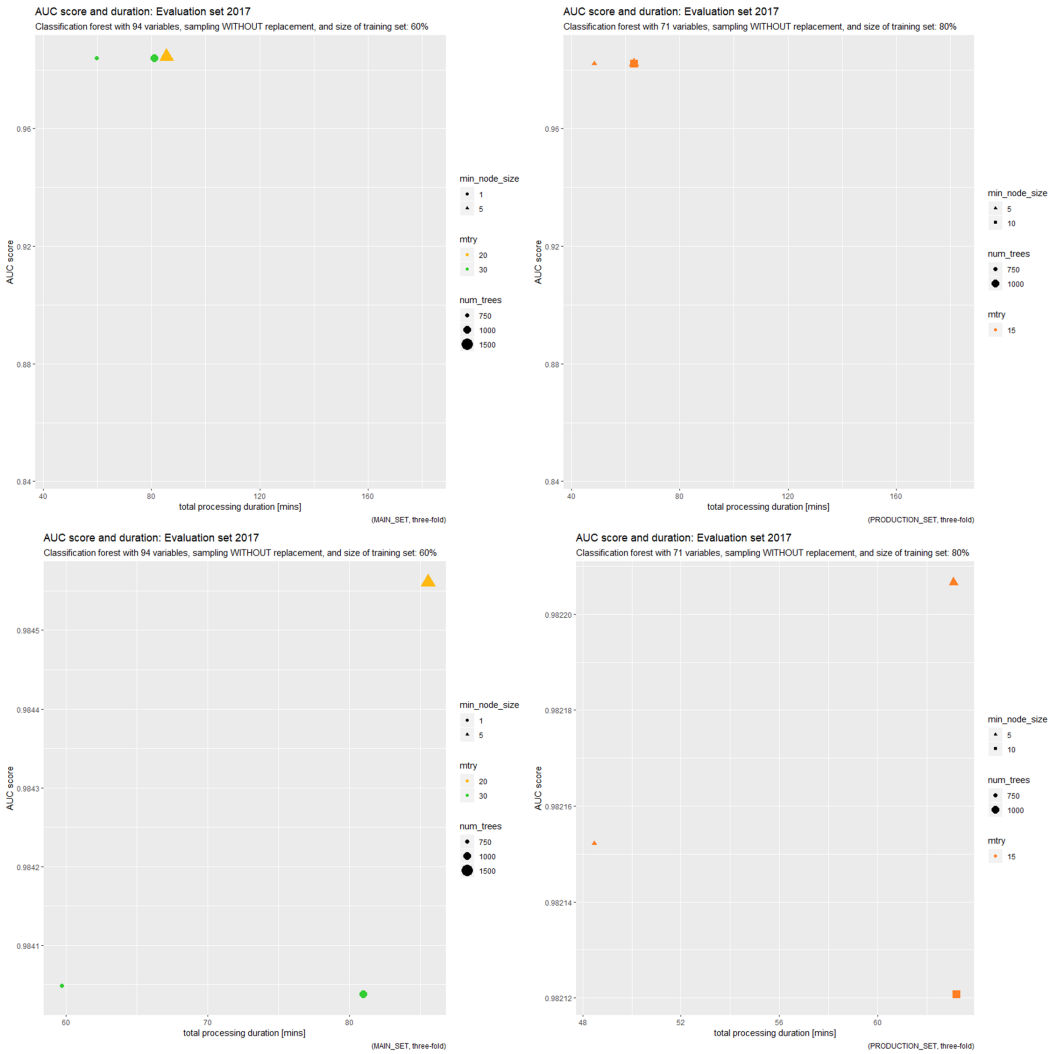

Figure 60: Evaluation of the top 3 *probability forest* models with evaluation data from 2017 and assessed via *AUC*. The evaluation was performed with the models' *test sets* created from data of the year 2017. The two plots on the left represent the *main set*, the two plots on the right constitute the *production set*. While the upper plots share the same y-axes scale, it has been adjusted ("zoomed-in") for the lower plots. As the *test sets* used for this evaluation originated from *three-fold splits*, evaluation data for 2017 was only available for *probability forests*.

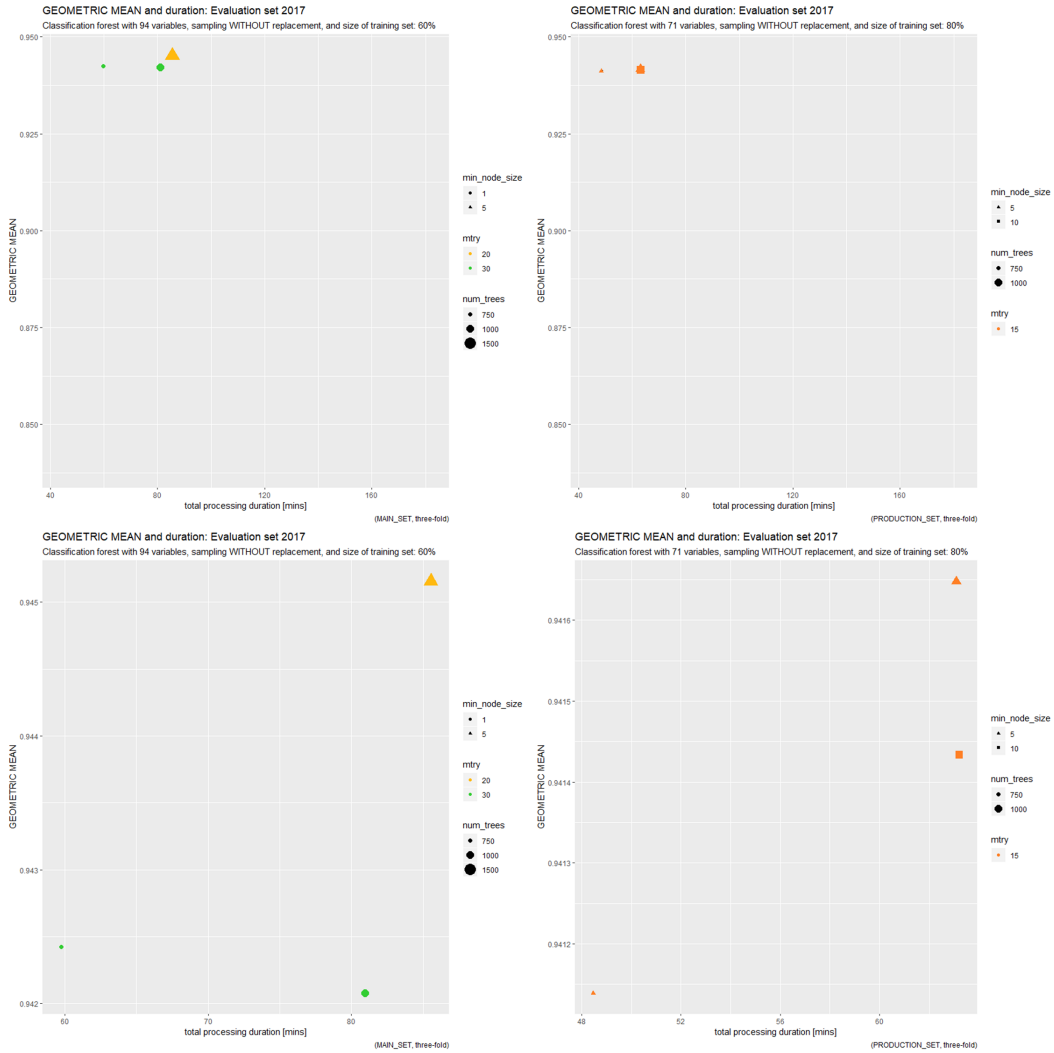

Figure 61: Evaluation of the top 3 *probability forest* models with evaluation data from 2017 and assessed via *GM*. The evaluation was performed with the models' *test sets* created from data of the year 2017. The two plots on the left represent the *main set*, the two plots on the right constitute the *production set*. While the upper plots share the same y-axes scale, it has been adjusted ("zoomed-in") for the lower plots. As the *test sets* used for this evaluation originated from *three-fold splits*, evaluation data for 2017 was only available for *probability forests*.

E Appendix: Discussion

(a)	**Type**	**HP**
	Model	**ProDesk 600 G1 Desktop Mini Business PC**
	Operating System	Windows 7 Professional (64-bit)
	Processor	2.2 GHz Intel Core i5-4590T
	Cache	6 MB
	Processor cores	4
	RAM	4 GB DDR3 SDRAM
	Graphics processor	Intel HD Graphics 4600
	Storage	500 GB SATA HDD (7,200 rpm)

(b)	**Type**	**HP**
	Model	**EliteBook 720 G1 Notebook-PC**
	Operating System	Windows 7 Professional (32-bit)
	Processor	1.7 GHz Intel Core i5-4210U
	Cache	3 MB
	Processor cores	2
	RAM	4 GB DDR2 SDRAM
	Graphics processor	Intel HD Graphics 4400
	Storage	500 GB SATA HDD (5,400 rpm)

(c)	**Type**	**Lenovo**
	Model	**ThinkCentre M720q Tiny**
	Operating System	Windows 10 Enterprise
	Processor	1.7 GHz Intel i5-8400T
	Cache	9 MB
	Processor cores	6
	RAM	8 GB DDR4
	Graphics processor	Intel UHD Graphics 630
	Storage	256GB SSD

(d)	**Type**	**HP**
	Model	**EliteBook 840 G5 Notebook-PC**
	Operating System	Windows 10 Enterprise
	Processor	1.8 GHz Intel i5-8250U
	Cache	8 MB
	Processor cores	4
	RAM	8 GB
	Graphics processor	Intel UHD Graphics 620
	Storage	256 GB SSD

Table 5: List of hardware available. In the course of an organisation-wide replacement of hardware on 2 October 2019, machines (a) and (b) were replaced by machines (c) and (d), respectively.

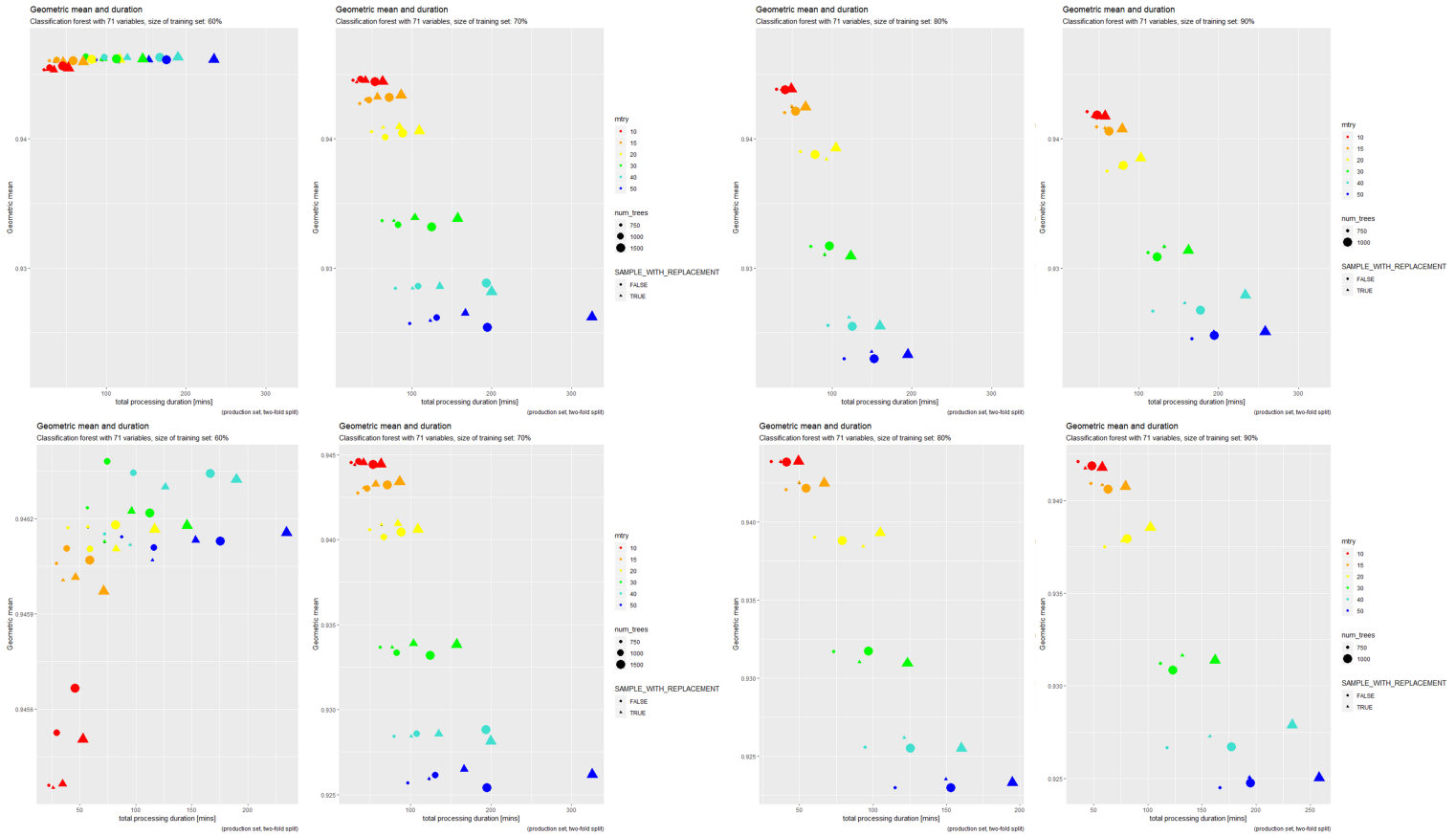

Figure 62: Comparison of *tuning level 2* results for all *training set sizes* with five (above) and six (below) *mtry* values for *classification forests* created from the *production set*. The upper four plots are the same as those displayed in figure 37. The additional *mtry* value 50 (dark blue) initially was not selected for *tuning level 2*. For the *training set sizes* 70%, 80%, and 90%, the additional *mtry* value performed least well, for the *training set size* 60% its performance was average. However, only the two best-performing *mtry* values were passed on to the next level, which in the cases displayed above would not have included *mtry* 50. While the upper four plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.

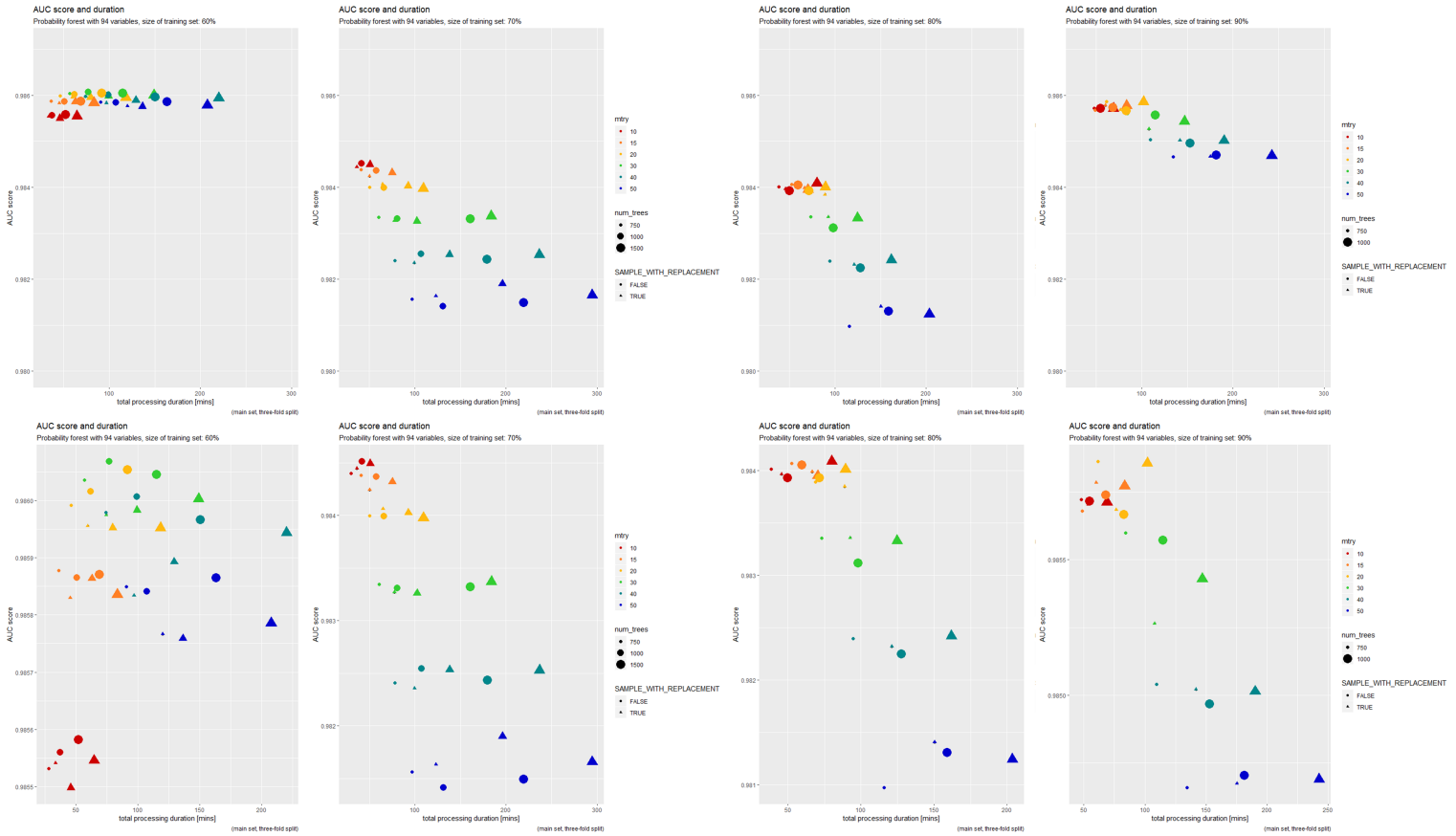

Figure 63: Comparison of *tuning level 2* results for all *training set sizes* with five (above) and six (below) *mtry* values for *probability forests* created from the *main set*. The upper four plots are the same as those displayed in figure 38. The additional *mtry* value 50 (dark blue) initially was not selected for *tuning level two*. For the *training set sizes* 70%, 80%, and 90%, the additional *mtry* value performed least well, for the *training set size* 60% its performance was average. However, only the two best-performing *mtry* values were passed on to the next level, which in the cases displayed above would not have included *mtry* 50. While the upper four plots share the same y-axis scale, it has been adjusted ("zoomed-in") for the lower plots.