UTRECHT UNIVERSITY

MASTER'S THESIS BUSINESS INFORMATICS

Constructing and Predicting School Advice for Academic Achievement: A Comparison of Item Response Theory and Machine Learning Techniques

Koen J. NIEMEIJER

Department of Information and Computing Sciences
Faculty of Science

July 21, 2019

Supervisors

Dr. M.J.S. Brinkhuis	Utrecht University
Dr. Habil. Ing. G.M. Krempl	Utrecht University
Dr. R.C.W. Feskens	Cito
J. Koops, MSc.	Cito

Utrecht University

Abstract

Constructing and Predicting School Advice for Academic Achievement: A Comparison of Item Response Theory and Machine Learning Techniques

by Koen J. NIEMEIJER

Introduction. In contemporary education, tests can be used to estimate students' abilities and thereby give an indication of whether their school type is suitable for them. However, tests are usually conducted for each content area separately which makes it difficult to combine these results into one single school advice. To this end — in the context of a student monitoring system — we research a series of tests that measure progress for the purpose of predicting school advice. Concretely, we examine domain-specific and domain-agnostic methods and compare their results both quantitatively and on explanations.

Method. In this research, we provide a comparison between both domain-specific and domain-agnostic methods for predicting school advice. First, we describe current approaches for measuring progress from educational tests. Next, we examine which methods are suitable to use for the purpose of combining content areas and predicting school advice. An IRT model is calibrated from which an ability score is extracted and is subsequently plugged into a multinomial log-linear regression model. Second, we train a random forest (RF) and a shallow neural network (NN) and apply case weighting to give extra attention to students who switched between school types. We compare the predictive performance, computational feasibility, and explainability of the models.

Results. When considering the performance over all students, RFs provided the most accurate predictions followed by NNs and IRT respectively. When only looking at the performance of students who switched school type, IRT performed best followed by NNs and RFs. Case weighting proved to provide a major improvement for this group. Furthermore, all models were found to be computationally feasible. Lastly, IRT was found to be much easier to explain in comparison to the other models; RFs are somewhat more interpretable than NNs.

Discussion. In practice, concept drift would occur because the recommendations made by the model leads to different school advice than the one given if there was no model; hence, this positive feedback loop seems inevitable but can be diminished by solely using the model for decision support. Ethical issues for the use of ML models revolve around differences in importance between content areas and whether this is fair. Moreover, legal considerations are posed by the GDPR's 'right to an explanation'. Future research includes choosing different class aggregations, other models, inspecting confounding variables, and testing generalisability.

Keywords. *Educational Measurement, Item Response Theory, Machine Learning, Explainable AI*

Acknowledgements

In November 2018 I started my thesis that marks the end of five years of studying at Utrecht University. When I embarked on this project, I had little affinity with tests or educational measurement, but over the past few months in which I have started to make my knowledge on this area more profound, I truly found a passion for this subject; not only for the machine learning side but also for the more traditional aspects. In order to successfully complete this project, my master's programme has given me the tools I needed. That is why I want to thank all professors of Utrecht University for their knowledge, wise lessons, and effort they put into their courses I followed over the last few years.

This study would not have been possible without the help of a number of outstanding people. First of all, I would like to thank Matthieu Brinkhuis for your passion and enthusiasm for the topic and my project in particular, and the much needed support and faith you have put in me. Secondly, Georg Krempl, thank you for the many discussions we have had on the topic of machine learning and the many ways in which you have shown me how to overcome a problem I faced. Thirdly, I would like to thank Cito for providing me with the opportunity, data, and workplace to write this thesis and the guidance you have given me throughout this project. Specifically, I would like to thank Remco Feskens — your optimism and enthusiasm have encouraged me to get past many difficult problems and will much be missed in future endeavours. Jesse Koops, thank you for the excellent feedback and the (very necessary) help you have offered me with Dexter.

Next, I would like to thank my friends and family — without their much-needed encouragement none of this would have been possible. A special word of thanks goes out to my fellow students Lars, Rowan, and Thomas who supported me throughout this venture.

Koen Niemeijer

Contents

List of Abbreviations

1PL	**O**ne-**P**arameter **L**ogistic model
2PL	**T**wo-**P**arameter **L**ogistic model
AUC	**A**rea **U**nder the **C**urve
CRISP-DM	**C**ross-**I**ndustry **S**tandard **P**rocess for **D**ata **M**ining
CTT	**C**lassical **T**est **T**heory
ECDF	**E**mpirical **C**umulative **D**istribution **F**unction
GAM	**G**eneralised **A**dditive **M**odel
GDPR	**G**eneral **D**ata **P**rotection **R**egulation
IRT	**I**tem **R**esponse **T**heory
LDA	**L**inear **D**iscriminant **A**nalysis
MCC	**M**atthews **C**orrelation **C**oefficient
MLR	**M**ultinomial **L**og-linear **R**egression
MML	**M**arginal **M**aximum **L**ikelihood
MIRT	**M**ultidimensional **I**tem **R**esponse **T**heory
ML	**M**achine **L**earning
NN	**N**eural **N**etwork
RBF	**R**adial **B**asis **F**unction
RF	**R**andom **F**orest
ROC	**R**eceiving **O**perating **C**haracteristic
SEM	**S**tructural **E**quation **M**odelling
SMS	**S**tudent **M**onitoring **S**ystem
SVM	**S**upport **V**ector **M**achine

Chapter 1

Introduction

In 1967, only 38.5% of children in the Netherlands aged 12-18 years old attended secondary education. A new law was subsequently passed requiring all children to attend some form of education. One year later, the number of children aged 12-18 attending secondary education rose to 64.4%, and even further to 81.1% in 50 years' time (CBS, 2018). Moreover, if we include children who already attend tertiary education, as many as 98.6% attend part- or full-time education. Now that nearly every teenager goes to school, secondary and tertiary education have become a substantial part of our society.

Academic achievement has become increasingly important because it largely determines admission to further education and future opportunities on the labour market. In contemporary educational systems, grades are an indication of academic achievement. Subject tests gauge an underlying (latent) skill which results in such a grade. In practice, however, important decisions of students' prospect are based on the general view that a teacher has of a student. While latent skill can be measured accurately through tests, combining different test results is less straightforward. For example, should the performance on mathematics be just as important in making decisions as English? In this research, we want to combine test results to form an overall assessment of a student to help them in making important educational decisions. After all, one of the most important decisions a pupil has to make in the final year of primary school is which school type they want to pursue.

The Netherlands' secondary educational system is characterised by division — a layered educational system ranging from vocational to theoretical school types[1]. Grades determine what type of secondary education a student is eligible for; Subsequently, the type of secondary education determines what type of tertiary education a student can go to and consequently which jobs they can apply for. To adequately guide students through the educational system, school advice plays an important part in helping students decide on which school type to pursue. For this reason, the aim of this study is to establish a method for determining school advice (i.e. advice on which school type to go to) in secondary education. We are particularly interested in finding those students who switch between school types in their first three years, since they are usually the ones relying most on this advice.

[1] See Appendix A for an explanation of the educational system in the Netherlands

Problem Statement

In the Netherlands, teachers in secondary education already give school advice to students. They do so based on the results of a series of tests that measure progress. Students take tests on different content areas (e.g. English or mathematics) and based on the score of these tests, the teacher determines which school type is most suitable for a student.

However, there are three problems with this approach:

- Latent scores are not intuitive. For instance, a score of 150 does not have any intrinsic value until you compare it to the scores of classmates.
- Latent scores from tests on different content areas are not comparable. That is, a score of 150 for mathematics and a score of 140 for Dutch does not imply this student is better at mathematics than at Dutch because the scores have different scales. Thus, scores only have value if you compare them to other scores from the same test.
- For teachers, it is difficult to combine scores from different tests when trying to establish school advice because of the previous problem. How should scores with different values be counted? Furthermore, should a teacher consider different subjects to be equally important? While we may be inclined to let each subject weigh in equally, problems arise when one content area — for instance English or mathematics — is highly beneficial to students' future success and should, therefore, attribute more to the school advice than tests from other content areas. That is to say, should, for example, English weigh as much as mathematics when deciding if a student should pass or not? How about when deciding which educational track a student should pursue?

Since our aim is to facilitate the decision making of choosing the most suitable school type, we predict students' prospective academic achievement by establishing school advice that gets to grips with their overall academic progress. To this end, this study aims to create a novel approach to constructing this school advice for students.

In order to do so, we look at both domain-specific and domain-agnostic approaches such as techniques from the fields of item response theory (IRT) and machine learning (ML) respectively[2]. Using these techniques, we attempt to produce high-quality predictions to estimate students' achievement in a few years' time based on their current performance including the implicit (and some explicit) weights that are put on these areas. To determine the validity of school advice, we compare the predictive validity of different methods. More specifically, we analyse which methods are more suitable to predict school advice in terms of accuracy, feasibility, and explainability.

Contributions

We make several scientific contributions: (i) a state-of-the-art literature review on the latest trends in e-learning, (ii) a novel way to combine test scores on a unidimensional scale, (iii) a proof-of-concept showing how domain-specific techniques compare to domain-agnostic techniques, and (iv) an analysis of the trade-off between predictive power and explainability.

[2] See the list of abbreviations, pp. ix

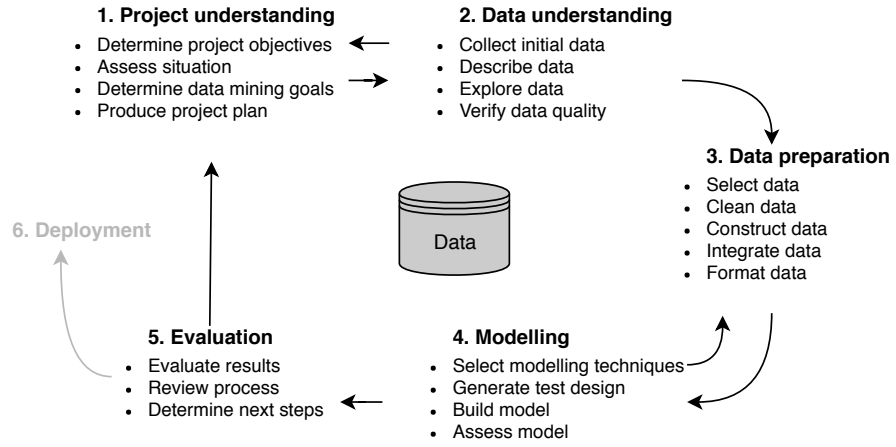

Figure 1.1: The cross-industry standard process for data mining (DM) is a general stepwise description for a data mining project.

1.1 Context

This research revolves around a student monitoring system (SMS) in the Netherlands; that is an information system widely used in Dutch secondary education to keep track of students' educational progress over multiple years (Van Til & Van Boxtel, 2015). The SMS consists of multiple tests and test versions in which multiple skills of the students are measured. Students can take tests at four moments in time: at the start of year one and at the end of year one, two, and three. The further the students progress in their education, the more difficult the tests become relative to their starting level. Moreover, there are different test versions which differ in difficulty, depending on the school type. Finally, tests aim to measure students' skills in various content areas, such as English reading skills or mathematics.

The results from these tests are used by teachers to determine the progress of students in comparison to their peers. For each content area, teachers receive a score that is consistent with other test scores for that content area. In other words, two tests taken by the *same* student at the *same* moment in time but on *different* content areas have independent scores. In this research, we attempt to establish school advice for students by combining these scores into an (implicit) composite score.

1.2 Research Approach

To ensure a logical workflow and minimise the risks of pitfalls, we adopt the *CRISP-DM* method. The cross-industry standard process for data mining (CRISP-DM) is a widely applicable method for all sorts of data mining projects (Chapman et al., 2000); it outlines a general process which minimises the chance of making critical errors. The main phases and their subphases are depicted in Figure 1.1. Phases in the model are sequential but backpedalling is usually inevitable (as is also indicated by the arrows).

In this model, a project starts from the perspective of *understanding* both the project (goals) and the data itself. When the goals are set and the data has been explored, *data preparation* is a necessary step to construct a tidy, complete data set that can be

used in the modelling phase. The modelling phase is where the action happens — various models are constructed and their parameters are tuned to an optimal value. Once the models perform sufficiently, the results are evaluated and interpreted, and subsequently gauged against the initial goals set in the project understanding phase. The lessons learnt in this cycle can then be used to trigger new, more focussed research questions. As is evident from Figure 1.1, there is a sixth phase called *deployment* which is greyed out. This phase involves presenting the acquired knowledge to the customer or some other stakeholder of the model to influence their decision making. However, this step is outside the scope of this research.

Research Questions

At the beginning of this chapter, we stated a societal and scientific need to predict school advice. The main research question is thus formulated as follows:

RQ *How can educational tests be used to predict school advice?*

To further concretise this knowledge problem, we identify several subquestions (SQs) and closely related subsubquestions:

SQ1 *What techniques can be used to predict school advice?*

1.1 *What are current approaches to predicting school advice?*

1.2 *How can domain-specific techniques be used to predict school advice?*

1.3 *How can domain-agnostic techniques be used to predict school advice?*

This subquestion focusses in particular on the exploration of both traditional and contemporary techniques that may be utilised to predict school advice. While SQ1.2 looks at the domain-specific side of aggregating items and estimating its error, SQ1.3 showcases which state-of-the-art domain-agnostic techniques can be employed to predict school advice.

SQ2 *How do different techniques perform when predicting school advice?*

2.1 *How do domain-specific and domain-agnostic techniques perform in terms of predictive accuracy?*

2.2 *How do domain-specific and domain-agnostic techniques perform in terms of computational feasibility?*

At the core of this research is constructing a number of models which attempts to predict school advice as accurately as possible. Thus, we tune their hyperparameters and see whether there any differences between domain-specific and domain-agnostic techniques in terms of accuracy. Additionally, we must take into account the computational feasibility that is likely to be a factor when constructing computationally expensive ML models.

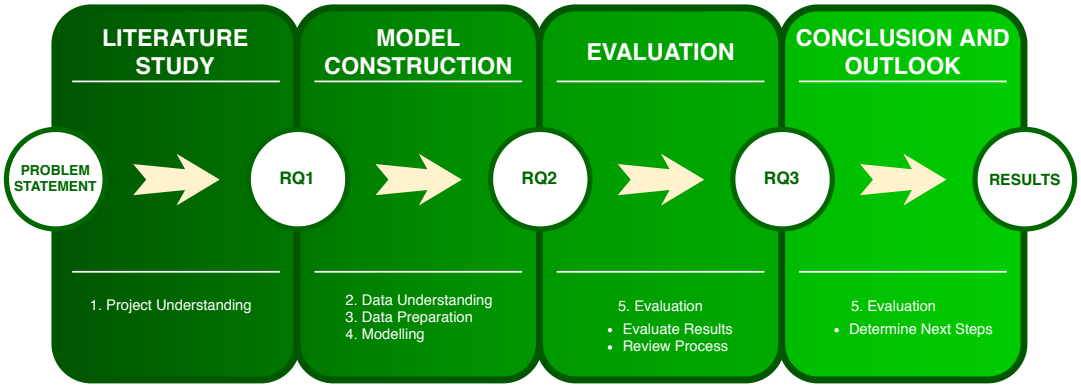

Figure 1.2: Thesis structure

SQ3 *How do different techniques perform in terms of explainability?*
In addition to comparing the predictive quality of models, we must also look at differences in explainability of domain-specific and domain-agnostic models. That is, we hypothesise that predictions from more complex approaches — such as black-box ML models — are more difficult to explain than predictions from simpler methods such as those from test theory.

We can roughly map each SQ to a phase in CRISP-DM. Concretely, SQ1 can be seen as a way to understand and aggregate the data and therefore belongs in the *data understanding*, *data preparation*, and perhaps even in the *modelling* phase since we not only look at *which* models can be used but also *how* they can be used. Next, SQ2 evaluates the predictive performance of both models and therefore belongs in the *modelling* (*assess model*) phase. Lastly, SQ3 covers the *evaluation* phase of CRISP-DM and elaborates on the models in terms of explainability.

1.3 Thesis Outline

The rest of this thesis is structured as follows. A graphical overview can be found in Figure 1.2. First, Part 1 provides an overview of current approaches for predicting school advice and scrutinises literature which aims to yield a comprehensible analysis of both domain-specific and domain-agnostic methods. This can be seen as understanding the context and literature of the project and is, therefore, part of the project understanding phase.

Then, Part 2 describes the data which is used to predict school advice and combine scores to form school advice. Moreover, it corresponds to the data understanding, data preparation, and modelling phases as defined by CRISP-DM.

Part 3 expands on both the philosophical and psychological aspect that underlies this research. That is, models created in the previous part are compared in terms of performance and explainability and we thereby evaluate their advantages and disadvantages. This part corresponds to the evaluation phase.

Finally, Part 4 concludes this thesis by summarising over the results and providing a discussion of the results by looking at its practical implications, limitations, and outlook for the years to come.

Literature Study

The previous chapter described the goals and research questions of this project and provided a method to achieve this. In this part, we explore the scientific literature to learn about existing solutions in different contexts and scrutinise both domain-specific and domain-agnostic statistical methods.

Chapter 2

Background: Current Approaches for Measuring Progress

Before commencing a literature study for potential techniques to use, we must first evaluate which approaches have already been adopted in educational and related fields. First, we look at (educational) progress tests and how they are used in practice and in other fields. Moreover, we look at growth models which is a basic form of a progress test. Finally, we examine how school advice has fared both in the SMS and in other, related educational systems.

2.1 Educational Progress Tests

There are roughly two traditions of measuring progress in SMSs: tracking progress and modelling growth. We first look at the former and see how they are applied in an educational context before moving on to growth models.

Bradley and Terry (1952) and Luce (1959) first introduced a paired comparison model known as the BTL model for comparing opponents in terms of skill. Based on this model, an influential algorithm for doing this in chess was developed by Elo (1978) and had shortly thereafter been adopted by the World Chess Federation (FIDE). In a chess match, two opponents of equal strength play a game after which their scores are updated to reflect either a victory or a defeat. This situation is analogous to item scoring in the sense that a test taker and an item are 'opponents' such that a test taker's score increases when the item has been endorsed (i.e. answered correctly) or decreases when the item is not endorsed. While the Elo-rating has much been researched in its original context (e.g. Batchelder & Bershad, 1979; Brinkhuis & Maris, 2010; Glickman, 1999), it has also been applied to other contexts such as sports (Gásquez & Royuela, 2016), Go (Coulom, 2007), and, as stated before, education (Brinkhuis, 2014; Klinkenberg, Straatemeier, & van der Maas, 2011; Pelánek, 2016).

With the advent of big data, continuously updating scores and parameters dynamically has been an important topic especially. In their research, Brinkhuis et al. (2018) describe their experience over a decade of on-the-fly parameter estimation in an online learning environment for mathematics. While they found that the Elo-rating system is well applicable to online learning, various unforeseen problems such as situations where students outsmarted the system or violations of unidimensionality

occurred. Nevertheless, Brinkhuis et al. (2018) describe their approach as a success thanks to high predictive accuracy and clear overall improvement of student performance.

Similarly, Pelánek (2016) examines the use of the Elo-rating by applying it to tests concerning knowledge of geography. They do so by comparing three methods: (i) the proportion of correct answers, (ii) joint maximum likelihood estimation, and (iii) the Elo-rating. In their findings, they describe Elo-rating to be well suitable for *"adaptive practice or low stakes testing"* (Pelánek, 2016, p. 177) since it is fairly easy to use, although it does not give statistical guarantees of estimated skills. Hence, when one desires a fine-grained model for a high-stakes test, other methods might be preferred over the Elo-rating.

Another approach of tracking progress in an education context is given by Hofman, Jansen, De Mooij, Stevenson, and Van der Maas (2018) who used time-series analysis to gain insight into developmental and learning process of mathematical skills. In contrast to the previously listed studies, Hofman et al. (2018) applied Bayesian logistic regression and a unidimensional IRT model to the data. Another example is given by Brinkhuis and Maris (2019a) who use dynamic Bayesian estimation on a general IRT model, which is able to deal with changes over time and missing data. Since IRT is used as a dedicated method in this research, we refer to Chapter 3.1 for a detailed description.

2.1.1 Growth Models

Already in 1938, Wishart developed an early growth model for the physical development of piglets. For each individual, they used a growth term based on sex, litter, and treatment, and fitted a model with both linear and quadratic terms as a function of time. Corrected for the initial weight of the piglets, Wishart (1938) uses analysis of variance to show how live weight-gain (i.e. growth) can be attributed for by a function of time containing both linear and parabolic terms.

Rao (1958) further explained how various growth models can be used on all sorts of data. Under the assumptions of normality and independence of errors, measurements of growth $y_0, y_1, \ldots, y_n$ do not need to be taken at specific points in time as long as they are somewhat spread evenly. Since they also note how growth rate is rarely uniform, a growth function $\tau = G(t)$ must be a complex function of time. By analysis of variance on the growth of rats under different treatments, they show how growth rate between groups is nearly unchanged when corrected for the initial weight (Rao, 1958, p. 6). Furthermore, they note how the growth curve can be transformed by higher order polynomials or logarithms to get an even closer fit. In the rest of their work, they outline several other tests for verifying the validity of their work.

Based on the work of Rao (1958) and Tucker (1958), McArdle (1986) developed *latent-variable structural-equation modelling (SEM)* which seeks to replace fixed higher-order terms with estimated basis functions for the coefficients. However, as Stoel, van den Wittenboer, and Hox (2004, p. 242–243) note, while this is an elegant way of estimating growth curves, it contains a number of deep-rooted pitfalls that may significantly alter conclusions. Most of these issues are due to scaling of the latent growth factor which may result in different estimations of the standard error and may even give opposite conclusions in significance tests. In the work of Stoel et al. (2004, p. 243), they explore the applicability of a *two-stage approach* (Jöreskog & Sörbom, 1989) for

a latent growth curve model (LGC) with an estimated basis function. According to T. D. Little, Schnabel, Baumert, and Schnabel (2000, p. 82), this two-stage approach *"is certainly a viable optional basis in LGC models"*.

2.2 Reliability of School Advice in Practice

In practice, an important question to ask when predicting school advice is to what extent the advice is actually being followed-up on. In other words, *what is the predictive accuracy of school advice?* To this end, we draw a comparison between the test used in this research and one that is very similar but has already been thoroughly investigated in earlier work. Concretely, we report on the results of a test taken at the end of students' primary education in the Netherlands.

After 8 years of primary education, students take a final test in order to get school advice concerning their secondary education. In total, 85% of primary schools in the Netherlands use this test to get an indication (i.e. school advice) as to what type of secondary education is suitable for students (Van Boxtel, Engelen, & De Wijs, 2011). Aside from school advice as a result of this test, students also receive school advice from their teacher *prior* to taking this test. As it turns out, school advice from teachers and this final test show great overlap, although there can be differences in the 'lower' levels of school types (Stroucken, Takkenberg, & Béguin, 2008). This is, therefore, an indication of the reliability of this test. Additionally, Van Boxtel et al. (2011) applied multiple regression as a way to predict school advice by using students' test results as predictors, $R^2 = .69$, $p < .001$. Furthermore, results did *not* improve by adding results of an intelligence test.

Finally, Van Boxtel et al. (2011) compared school advice from the final test to the actual school type students were enrolled in their first or second year of secondary education. In total, 80% of students who received school advice to go to a certain type of secondary education followed this advice, while 13% went upstream (i.e. to a 'higher' type) and 7% downstream. Notably, students who went upstream or downstream were more likely to switch school types in their first or second year of secondary education (Stroucken et al., 2008). In conclusion, while the final test in primary education's school advice is not perfect, it is being followed up on in 80% of the cases and is off by one level in 96.7% of the cases (Van Boxtel et al., 2011, p. 80).

2.3 Summary

In this chapter, we laid out past and current state-of-the-art approaches for measuring progress which contributes to answering SQ1.1. First, we described how and why the Elo-rating system (Elo, 1978) can be applied to an educational context and, more specifically, to item scoring. We scrutinised a number of researches thereby highlighting current practices in today's cases. Next, we examined the development of growth models and how they can be applied to, amongst others, measuring the growth of rats. Rao (1958) showed how growth must be a function of time and why other variance may be explained by context-specific *latent* variables. This, in turn, led to the development of *SEM* (McArdle, 1986) which, to this day, is still a widely used method.

Finally, we analysed the reliability of school advice in practice by detailing the predictive accuracy of a very similar test to the one studied in this research. In the

next section, we further explore a multitude of state-of-the-art methods which may be used to predict school advice and hence provide us with a way forward.

Chapter 3

An Overview of Statistical Methods

For the Prediction of School Advice

In this chapter, we look at various domain-specific and domain-agnostic methods and evaluate their use in this problem context. First, we examine a widely used method for evaluating both dichotomous and polytomous questions on tests, namely IRT. Secondly, we list and describe various ML methods and discuss their applicability to test data.

3.1 Item Response Theory

In the SMS, students receive one test for every school subject at four moments in time. The goal of these tests is to estimate students' performance on that subject. Tests for each subject are composed of a series of questions which can come from pre-existing test batteries. Naturally, tests are carefully constructed such that they only measure the desired skills and are not too easy or too difficult for students.

A popular method for doing this from a psychometric point of view is called *item response theory* (IRT) (Hambleton, Swaminathan, & Rogers, 1991; Lord, 1980; Lord, Novick, & Birnbaum, 1968). In contrast to *classical test theory (CTT)* (Verhelst & Verstralen, 1994), IRT focusses on the theory underlying the item and so does not have a test-level focus like CTT. Therefore, IRT is often more justifiable and more practical to solve measurement problems with than CTT (Embretson & Reise, 2000). In the context of test theory, an *item* is usually a question on a test which can have either one or multiple correct answers. Furthermore, a student's answer to an item is called a *response*. An item can be either dichotomous (i.e. two choices) or polytomous (multiple choices). A scored item is the representation of a student's response with respect to the true answer of the item.

Definition 3.1.1 — Scored item. For a response X_i on item i, a scored item has the following properties:

$$X_i = \begin{cases} 1 & \text{if the response to item } i \text{ is correct} \\ 0 & \text{if the response to item } i \text{ is incorrect} \end{cases}$$

The idea behind IRT is that a person's score is derived from the interaction of at least these two elements: (i) a person's innate skill at some subject called the *ability*, and (ii) the difficulty of an item referred to as *item difficulty*. A latent trait, or a latent variable, is a hypothetical construct that is not directly observable from the data (Bollen, 2002). In the context of IRT, the latent trait of a person is often referred to as their *ability*. Thus, the goal of IRT is essentially to make an ordering of the relative ability of respondents (Partchev, 2004).

Assumptions

To simplify the process of calculating their joint probability and thereby to estimate their ability, IRT assumes local independence of the answers given by the respondents. That is to say, the probability of a person *endorsing* (i.e. answering the item correctly) does not depend on their other answers but rather on their ability. More formally,

$$P(X_{1i}, X_{2i}, \ldots, X_I|\theta_j) = \prod_{i=1}^{I} P(X_{ij}|\theta_j). \tag{3.1}$$

Hambleton et al. (1991, p. 10–12) show that if this local independence assumption does not hold, respondents with the same ability level would give different answers. Since this property is undesirable, tests are constructed in such a way that this property will hold.

1PL. Given the ability parameter θ and the item parameter β, a person's likelihood $P(x_{ij} = 1|\beta_i)$ of endorsing an item can be modelled as a sigmoid (logistic) function which is more aptly called the *item response function (IRF)*:

$$P(x_{ij} = 1|\theta_j, \beta_i) = \frac{\exp(\theta_j - \beta_i)}{1 + \exp(\theta_j - \beta_i)}, \tag{3.2}$$

where i denotes the question and j references a person. This model is known as the **one-parameter logistic model (1PL)** since only one parameter (β) is used to estimate θ. The 1PL model is mathematically similar to the Rasch model (Rasch, 1960) although some underlying philosophical aspects differ. The sigmoid function has the properties that it forces the probability of a correct answer to always lie between 0 and 1 and is monotone. In this IRT model, when the item parameters have been estimated, a person's ability can be obtained as the maximum likelihood given the difficulty by essentially equating the ability parameter against the probability. Since the 1PL model uses the item parameters β, we clarify how these can be estimated later in this section when discussing design issues.

2PL. The 1PL model can be extended by including a *discrimination parameter* α that controls the slope of the item response function. An increase in the discrimination parameter allows for better distinguishing between persons with abilities far away from each other on the ability axis but suffers in determining ability parameters that are close to each other (Partchev, 2004, p. 25–26). This **two-parameter logistic model (2PL)** is defined as

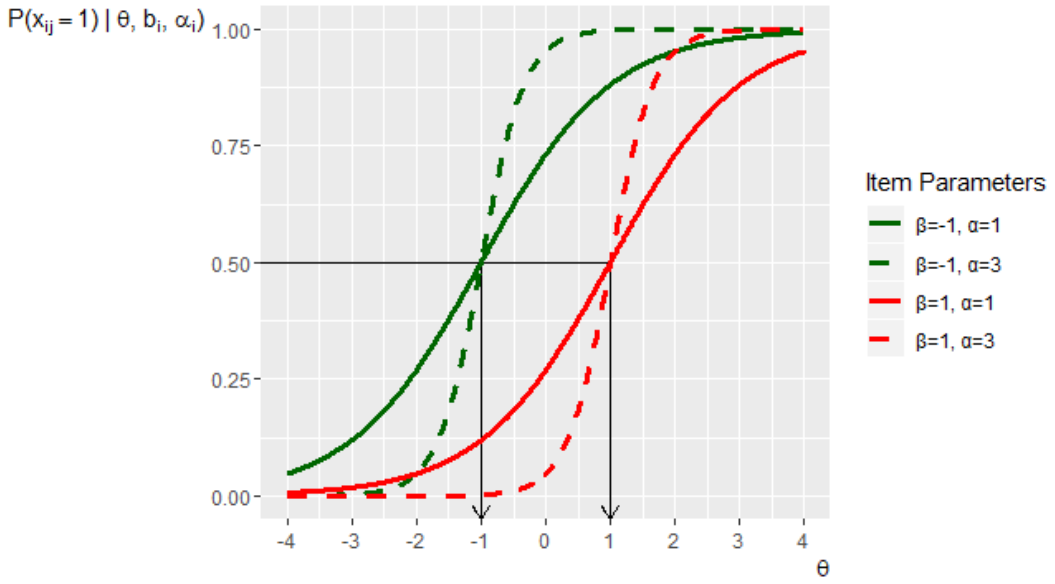

Figure 3.1: Four probability distributions with different parameter settings for the 1PL and 2PL model.

$$P(x_{ij} = 1|\theta_j, \beta_i, \alpha_i) = \frac{\exp[\alpha_i(\theta_j - \beta_i)]}{1 + \exp[\alpha_i(\theta_j - \beta_i)]} \tag{3.3}$$

A graphical representation of the 1PL and 2PL model can be found in Figure 3.1. In this figure, the curves with the same colour have the same item difficulty. As a consequence, curves with the same colour have the same ability when $P(x_{ij} = 1|\beta_i, \alpha_i) = 0.5$. For example, both green curves indicate students need, on average, an ability of -1 to have a 50% chance of endorsing the item, while the for the red curves students need an ability of 1 to have this same 50% chance on average. In fact, the difficulty parameter β controls the position of the curve along the x-axis. This is also indicated by the black line indicating the positions of the curves on the x-axis when $P(x_{ij} = 1|\beta_i, \alpha_i) = 0.5$. In other words, $P(x_{ij} = 1|\beta_i, \alpha_i) = 0.5$ if $\theta = \beta_i$.

Additionally, curves with the same line type have the same discrimination parameter α. In Figure 3.1, both dashed curves have the same slope. Since two models have a discrimination parameter of 1, they are equivalent to a 1PL model. Only the dashed curves are distinct from a 1PL and are in fact a 2PL model. Since the dotted curves have $\alpha = 3$, they are *more* discriminating for people with an ability close to 0. On the other hand, the models represented by the solid curves have $\alpha = 1$ and are therefore better at discriminating between ability far away from each other.

To summarise, there are several conclusions to be observed in these models:

- A higher ability θ gives a higher probability of endorsing the item.
- A higher β shifts the IRF to the right, which implies the item is more difficult.
- For 2PL models, the discrimination parameter α controls the slope of the curve. When α increases, the slope increases as well, making the item *more* discriminating for people with close to the same ability.

Design issues. Subsequent tests are linked through a so-called *pre-test*, which is a test containing items from two subsequent tests and is given to a sample group in order to estimate a few item parameters. To prevent students in the sample group from seeing a few items that may or may not be in their real test, the items on the pre-test are not used for at least a few years. This notion of having *anchor items* is visually represented in Figure 3.2 where level 1 is the most difficult and level 4 is the easiest. In these hypothetical tests, each version is made up of 10 items. Furthermore, each level inherits a few items (three in the example) from the more difficult levels. By doing this, comparisons can be made to estimate the difficulty of each item in relation to the level below (or above).

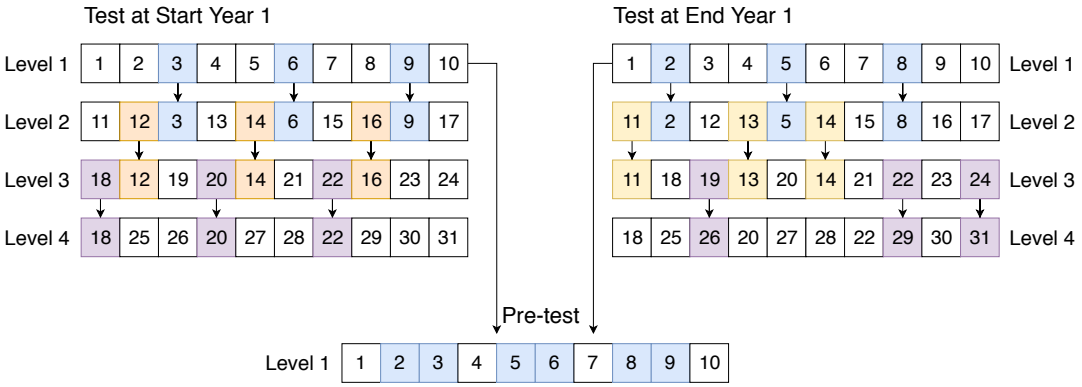

Figure 3.2: *Anchor items are items used in multiple test versions for the purpose of determining their difficulty.*

In summary, a test taken in a certain time period has multiple versions to accommodate different levels of difficulty. The items in each test are not disjoint but overlapping such that the second most difficult version inherits a few items from the most difficult test, the third most difficult version inherits a few items from the second most difficult test, and so and so forth.

3.2 Machine Learning

A *Machine learning* algorithm is a statistical learning method which takes which constructs a mathematical model of data and is able to make predictions or decisions without being explicitly programmed (Nasrabadi, 2007). The goal of ML is to model a function $\hat{f}(\cdot) = \hat{y}$ which approximates the true model $f(\cdot) = y$. In other words, ML algorithms attempt to model output data from some function that is shaped by input data. This chapter discusses several ML methods from the work of Breiman, Friedman, Olshen, and Stone (1984); James, Witten, Hastie, and Tibshirani (2013); Jurafsky and Martin (2009) amongst others.

Definition 3.2.1 — Machine learning model. A *machine learning model* is a function $\hat{f}(\cdot)$ of a number of predictors X that estimates some output values $\hat{Y}$ (James et al., 2013, p.15). The true model is given by $f(\cdot)$ which characterises the actual values Y. Since the true model is assumed to be unknown (unless artificially created), the estimated model can be seen as $\hat{f}(X) \rightarrow Y$, i.e. the model is a mapping from a number of input to a number of output values (Mohri, Rostamizadeh, & Talwalkar, 2012).

3.2.1 Categories of ML

Following James et al. (2013, Chap. 2), this section distinguishes between three types of machine learning. While there are many categories in ML, the ones listed below are essential to understand before we describe various ML algorithms in the subsections thereafter.

Supervised. In supervised ML, the outcome is known and the goal is hence to create a model $\hat{f}(\cdot) \approx y$. Usually, the following procedure is used (known as the *validation-set approach* (James et al., 2013, p. 176–178):

1. Split the data set into disjoint two parts: a training set (usually 70% of the data) and a test (or dev) set.
2. Train a model on the training data using the class labels.
3. Validate the model's performance on the test set.

Semi-supervised. While in supervised ML all the class labels are known, this is often a scenario where the data is gathered before analysis. In many other practical applications, only a part of the class labels is unknown and the model seeks to find some structure in the data based on the class labels that it does have (Chapelle, Scholkopf, & Zien, 2009, p. 2). In *semi-supervised* ML, there are two approaches to determining this structure.

First, in *passive* learning, some class labels are already available and the algorithm has to find some structure in the data to construct a model. In other words, the algorithm has to make do with the data it is given.

Second, in *active* learning, the model starts with *no* class labels but the algorithm is allowed to 'buy' a number of class labels to estimate a model from the data (Settles, 2012). Each class label that is 'bought' adds to the total cost of the algorithm. The fewer labels are bought, the better the algorithm as long as the performance does not suffer.

Unsupervised. Learning methods that fall under the unsupervised category of ML try to seek interesting patterns and relationships in the data (James et al., 2013, p. 26–28). Consider, for example, clustering algorithms that separate the data into a number of groups (called *clusters*) that belong together according to some pre-specified evaluation criteria.

3.2.2 Regression Analysis

Regression analysis is a ML technique which establishes a mathematical model of some data where the output is continuous. For instance, regression analysis is useful for predicting the maximum loan of a customer or learning the selling price of a house based on prices of similar houses in the same area.

Linear regression. The most well-known example of regression analysis is *linear regression*. This method models the relationship between an *independent variable* — i.e. the cause — and a *dependent variable* — the effect. As described previously, the independent variable can be seen as the parameters X while the dependent variable

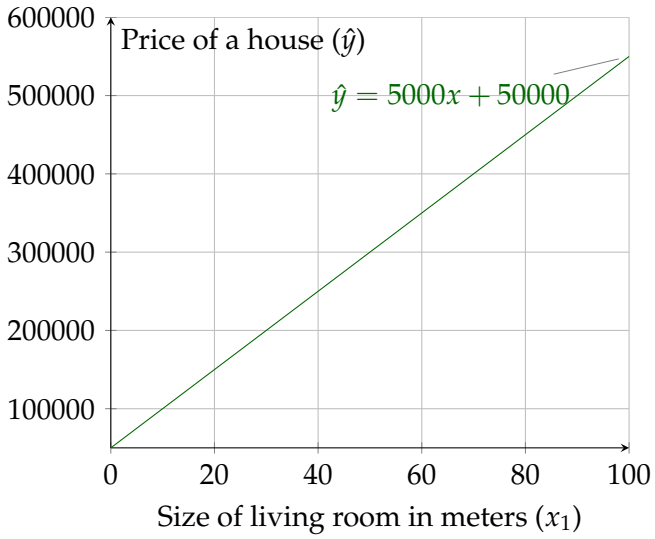

Figure 3.3: A linear plot of the interaction between the size of the living room and the price of a house. The relationship of the data is in the form $\hat{y} = \beta_0 + \beta_1 X_1$ as described earlier.

is represented by Y. If a regression has multiple parameters as input, it is known as *multiple regression*.

■ **Example 3.1 — Linear regression for housing data.** A real estate agent wants to know the future selling price of a house they intend to sell. Since the price of the house is determined by what it features (how many bedrooms, size of the living room, etc.), the agent can use these features to model the relationship they assume is linear. The model is in the form as described by James et al. (2013, Equation 3.19) only without the random error term:

$$\hat{f}(X_1, X_2, X_3) = \hat{Y} \tag{3.4}$$

or equivalently

$$\hat{Y} \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 \tag{3.5}$$

In Equation 3.4, X_1 represents the size of the living room, X_2, the number of bedrooms, and X_3 the size of the garden. The function is thus an estimation of the true model, $\hat{f} \approx f$. Note how capital letters refer to variables, whereas small letters denote an *example* x_1, i.e. a single instance of the variable X_1.

Equation 3.5 is similar to Equation 3.4 but in a different form. Here, $\hat{Y}$ is modelled as the interaction of variables (also called *predictors*) X_1, X_2, X_3 and their corresponding coefficients $\beta_1, \beta_2, \beta_3$. Furthermore, β_0 represents the *bias term*, that is the offset from the Y-axis. ■

As the name implies, linear regression is always linear and thus has a straight line through the data points. An advantage of linear regression is that, while the model has high *bias*, it has low *variance* (James et al., 2013, p. 33–36). In other words, it has a

strong bias in the sense that it assumes a linear relationship of the data while this is not necessarily the case but changes very little when a portion of the data is changed or a new sample is given. Linear regression performs especially well when the true model $f(\cdot)$ is also linear, although a perfect linear relationship rarely ever occurs. So even if the model is not perfectly linear and performs slightly worse than more complex models, it can still be the most preferred model due to Occam's Razor, that is the simplest model within a small error margin is usually the best one (Blumer, Ehrenfeucht, Haussler, & Warmuth, 1987).

Polynomial regression. An extension to linear regression is the use of *polynomials* to represent a higher order function than linear regression which is strictly linear (James et al., 2013, p. 266–288). A polynomial function based on Example 3.1 could be:

$$\hat{Y} \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2^2 + \beta_3 X_3^3$$

Rather than fitting a high-degree polynomial over the entire range of X, *piecewise polynomial regression* divides this range into separate pieces each with their own polynomial term. In this model, the region of X is split up into a number of predetermined regions of X where the points of change of the coefficients are called *knots* (James et al., 2013, p. 271).

$$y_i = \begin{cases} \beta_{01} + \beta_{11 x_i} + \beta_{21 x_i} + \beta_{31 x_i} \text{ if } & x_i < c; \\ \beta_{02} + \beta_{12 x_i} + \beta_{22 x_i} + \beta_{32 x_i} \text{ if } & x_i \geq c; \end{cases} \tag{3.6}$$

Equation 3.6 splits up the region of X into two regions: one of the subset of the observation with $x_i < c$, and one one of the subset of observations with $x_i \geq c$.

However, a consequence of piecewise polynomial regression is that the function $\hat{f}$ 'breaks' at every knot since it suddenly changes in slope and is therefore not continuous. An outcome to this problem is the use of a *degree-d spline* which takes the derivative of each piece. A further improvement can be made with *natural splines* which pose additional boundary constraints (the region where X is smaller than the smallest knot or larger than the largest knot) (James et al., 2013, p. 274) to further improve estimates at the boundary of X, but this is outside the scope of this research.

Finally, *smoothing splines* can be used to prevent $\hat{f}$ becoming too flexible and overfit the data. This process of restricting the parameters is called *penalisation*. James et al. (2013, p. 277) use the non-negative tuning parameter λ to penalise the residuals of $\hat{f}$:

$$\sum_{i=1}^{n} (y_i - g(x_i)^2) + \lambda \int g''(t)^2 dt \tag{3.7}$$

In Equation 3.7, the term $\sum_{i=1}^{n} (y_i - g(x_i)^2)$ is simply the *loss function* that encourages the function $\hat{f}$ to minimise its residuals, while $\int g''(t)^2 dt$ is a measure of the total change in the function $g\prime(t)$. Thus, if $\lambda = 0$ the function is a perfect fit for the data whereas if $\lambda \to \infty$ the line would be straight.

Generalised additive models. A useful addition to the previously mentioned linear regression methods is *generalised additive models (GAMs)* (James et al., 2013, p. 282–287). This method provides a framework for flexibly fitting Y when using multiple predictors. GAMs allows one to fit, for example, smoothing splines or least squares for multiple predictors by plotting a single predictor while holding the others constant. By doing this, multiple predictors can be fitted in turn and the method can consequently easily be updated by simply plotting another predictor. Notwithstanding this apparent ease of use, some methods, e.g. smoothing splines, are harder to fit than others. *Backfitting* (Hastie, 2017) provides an outcome to this problem by iteratively smoothing over the partial derivatives to estimate the parameters $\hat{f}(X_j)$.

3.2.3 Classification

While Section 3.2.2 focussed on ML techniques that predict *continuous* data, there are also scenarios in which ML techniques aim to predict *discrete* data. That is variables that are not quantitative but qualitative in nature so that a model can *classify a category*. Hence, this section describes a multitude of ML techniques that classify data into or multiple categories.

Notation 3.1. *We refer to predicted discrete instances of data as* classes *or* (class) labels. *We denote a set of k labels as* $\mathcal{L} \in \mathbb{R}^k$.

Logistic regression. While in theory linear regression can be used to predict categorical data, this is often a poor way of doing so (James et al., 2013, p. 129–130; Jurafsky & Martin, 2009, p. 82–87). Linear regression for categorical data with a decision boundary has two major problems:

- When there are more than two classes, using a linear regression would impose an ordering in the classes while none exists.
- In a binary setting, some values may fall outside the range $[0, 1]$ which can be problematic.

An adaptation of linear regression called *logistic regression* solves this by squeezing the values between 0 and 1 at either end of the curve. Logistic regression models the probability that Y has a particular label from the set of class labels $\mathcal{L}$. Its function is given by the *logistic function* (James et al., 2013, p. 132):

$$P(X) = \frac{e^{\beta_0+\beta^T X}}{1 + e^{\beta_0+\beta^T X}} \tag{3.8}$$

which can be rewritten as

$$\frac{P(X)}{1 - P(X)} = e^{\beta_0+\beta^T X} \tag{3.9}$$

In fact, logistic regression is mathematically similar to IRT as described in Section 3.1. Equation 3.8 is plotted in Figure 3.4 and is characterised by an S-shaped curve. The left-hand side of equation 3.9 is called the *odds*, which can take on any value between 1 and ∞. In this notation, β is a vector of parameters such that if the transpose is

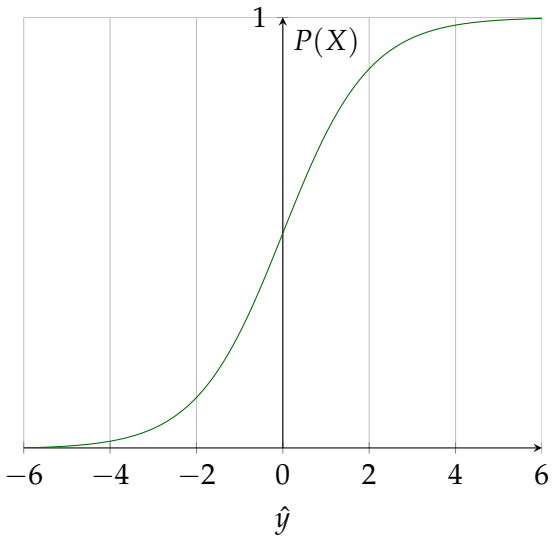

Figure 3.4: Logistic regression is characterised by its S-shaped curve.

taken, i.e. β^T, one can multiply it with the vector of parameters X to get the same results as the sum over all parameters $\sum_{j=0}^{n} \beta_j x_j$.

To assess the fit of a logistic regression model, it is possible to look at the z-statistic of a coefficient and its associated p-value. The error of the models is found by comparing the *true* labels against the *predicted* labels:

$$\frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

where $I(y_i \neq \hat{y}_i)$ is an indicator function that returns 1 if the labels are equal and 0 otherwise (James et al., 2013, p. 37).

Linear Discriminant Analysis. Similar to linear regression, it is possible to use logistic regression with multiple parameters. In practice, however, *multiple logistic regression* is hardly used due to the fact that multiple logistic regression may yield different results for the same predictors when they are highly correlated. Instead, *linear discriminant analysis (LDA)* assumes predictors are drawn from a Gaussian distribution with a common covariance matrix where the density functions overlap. James et al. (2013, p. 139) then use Bayes' theorem to estimate the probability that an observation x belongs to the Kth class:

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)},$$

where π_k is the class prior probability (i.e. the fraction of observations belonging to the Kth class) and $f_k(x)$ the posterior probability $f_k \equiv P(X = x|Y = k)$. Using the assumption that X is drawn from a Gaussian distribution, a class-specific mean vector and a common variance can be estimated and plugged into the posterior density function:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$
$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

James et al. (2013, p. 140) show that a variable is assigned to a class where the discriminant function

$$\delta_k = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

is the largest. In that case, the decision boundary for x is equal to the point

$$x = \frac{\mu_1 + \mu_2}{2}$$

When there are multiple classes, however, problems occur when data is highly collinear. This is due to the fact that a multivariate Gaussian distribution expects only some correlation between individual predictors and becomes distorted as collinearity increases. In short, LDA outperforms logistic regression when the assumption of a normal distribution approximately holds and the opposite is the case if it does not hold (James et al., 2013).

Naive Bayes. While logistic regression attempts to *discriminate* the features by directly modelling the chance $P(Y|X)$, naive Bayes instead is a *generative* classifier, that is a classifier with an underlying model of the data (Ng & Jordan, 2002).

As is apparent from the name, naive Bayes utilises the *Bayes theorem* that states

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}.$$

Obviously, one wants to maximise the chance $P(Y|X)$ which can be done by calculating the class *prior* of the feature distribution $P(X)$ by marginalising over y (Hand, Mannila, & Smyth, 2001, Chap. 4):

$$P(X = x) = \sum_y P(X, Y = y)$$
$$= \sum_y P(X|Y = y) \cdot P(Y = y).$$

However, calculating these probabilities for multivariate data is expensive, i.e. naive Bayes suffers from the *curse of dimensionality*. The algorithm thus assumes a conditional independence assumption such that X_i and X_j are assumed to be conditionally independent given the class label (Wu et al., 2008, p. 24–27)

$$P(X_i|X_j, Y) = P(X_i|Y)$$

which simplifies the posterior into

$$\begin{aligned} P(Y|X_i, X_j) =& \frac{P(X_i|Y) \cdot P(X_j|Y) \cdot P(Y)}{P(X_i, X_j)} \\ & \propto P(Y) \cdot \prod_{x \in X} P(x|Y) \end{aligned} \tag{3.10}$$

By assuming this conditional independence, Equation 3.10 no longer suffers from the curse of dimensionality. In a multiclass setting, the class with the highest probability gets selected as the correct label. The error of naive Bayes is similar to that of logistic regression as defined in Equation 3.2.3.

3.2.4 Tree-based Methods

A whole different category of models is *tree-based methods*. These models are based on *decision rules*, that is rules derived from the data to decide in which category a case belongs. More specifically, there are two types of trees: *classification trees* and *regression trees*. While in the former decision rules are used to decide which class label belongs to some input (see Example 3.2), the use of regression trees is equally possible by discretising continuous values.

■ **Example 3.2** Consider the situation of a loan officer who determines whether customers are eligible for a short-term loan. To help with their decision process, the loan officer has constructed a (very) simple decision tree as shown in Figure 3.5. In this example, a client walks into their office and asks whether they can get a loan. The loan officer subsequently interviews the client and asks for their income, savings, and age in order to determine whether this particular client is eligible for a loan. ■

The rest of this subsection describes a number of popular tree-based methods, such as decision trees, bagging, and boosting.

Decision trees. Decision trees involve segmenting the vector space into discrete subgroups that maximise intra-node homogeneity and inter-node heterogeneity (James et al., 2013). In other words, a decision tree attempts to create an upside-down tree of the data where the *leaves* are as pure as possible (up to some extent). A visual representation of a decision tree can be found in Figure 3.5.

In a decision tree, points where the tree splits are called *internal nodes*. *Branches* that do not have a split are known as *terminal nodes* or *leaf* nodes. Furthermore, the top (starting) node is termed the *root node*. As with any model, the goal of a decision tree is to model the data as accurately as possible. This is generally achieved by using *recursive binary splitting* which looks at *node impurity* and is measured by the Gini index (Breiman et al., 1984, p. 38)

$$i(t) = \sum_j p(j|t)(1 - p(j|t)),$$

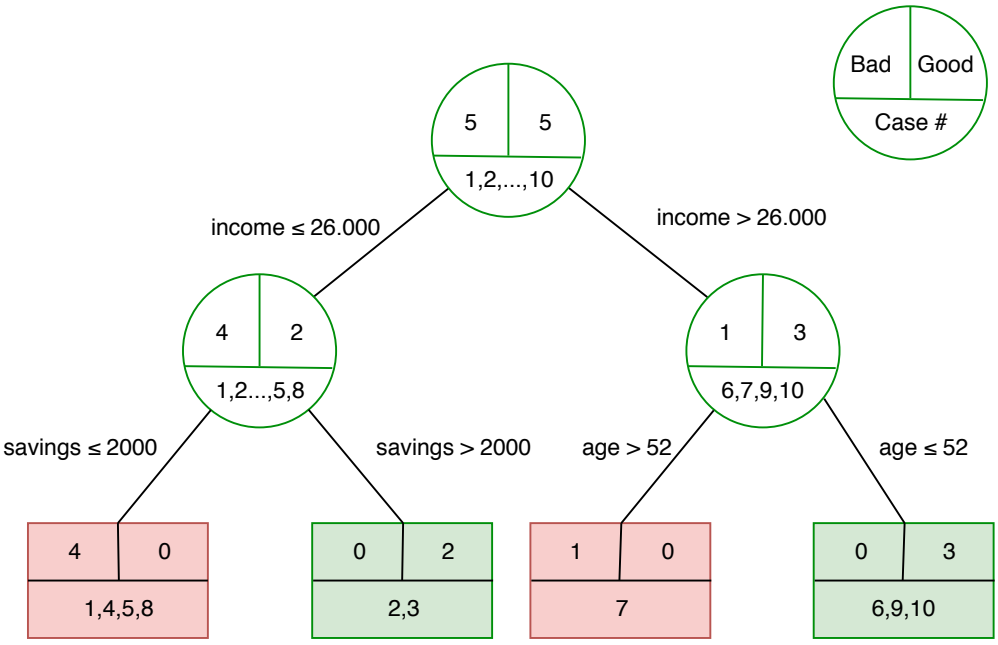

Figure 3.5: A decision tree for the purpose of determining whether someone is eligible for a short-term loan or not. A case traverses from the top of the tree until it reaches a leaf node where a red node indicates a loan has been declined and a green node that a loan has been approved.

where $p(j|t)$ is the relative frequency of class j in node t. Alternatively, entropy can be used which is similar to the Gini index:

$$i(t) = -\sum_j p(j|t) \log p(j|t).$$

However, there are two problems with a decision tree that models the data in this way:

- For a large dataset, a decision tree for every possible split grows very complex and becomes very difficult to compute.
- A fully grown tree models the data (nearly) perfectly which means the data has been overfitted.

For this reason, cost-complexity pruning (Breiman et al., 1984, p. 66–71) with tuning parameter α can be used to merge back the tree. However, even with this approach decision trees are known to be a weak classifier with a high variance but low bias (Breiman, 1996; Dietterich, 2000; Freund & Schapire, 1997, p. 62). To accommodate their weaknesses, various *ensemble* methods have been developed that combine multiple trees in order to make more accurate predictions.

Bagging. As stated before, decision trees are generally not a very good method since they are prone to overfitting and cannot handle very complex cases. Consequently, **b**ootstrap **agg**regat**ing** (abbreviated as bagging) is an *ensemble* method which merges multiple bootstrapped decision trees to significantly reduce the error rate. Breiman

(1996) found that for classification trees the error rate was reduced by 6% to 77% by using bagging instead of regular trees, and 21% to 46% for bagged regression trees.

In bagging, each time a tree is grown, the dataset is sampled with replacement (known as *bootstrapping*) so that each tree is different from the rest. Trees are grown deep, which means they have high variance but low bias. In a classification setting, the majority vote can be used to determine to which class a case belongs.

Random forests. *Random forests* is a method closely related to bagging. In essence, the algorithm is the same except that each time a split occurs at a node, the predictors are sampled so that only a subset is considered for that split. Breiman (2001) proposed this tweak to bagging by stating that bagged trees with very strong predictors often have high correlation among themselves. By sampling the predictors at each split, the algorithm is only allowed to consider a few of the total number of predictors which thereby decorrelates the trees. From a set of m predictors, $\sqrt{m}$ are usually considered best for classification and $m/3$ for regression trees (Breiman, 2001).

Boosting. *Boosting* is a general method that aims to make a 'weak' classifier strong by slowly increasing its performance. While this method can be applied to any learning algorithm, it is commonplace to do so for decision trees since these are weak learners in itself (Dietterich, 2000; Freund & Schapire, 1997, p. 119–120). The algorithm works by training a (weak) base classifier on various distributions of the data and subsequently combining these predictions into a single composite classifier. More specifically, the algorithm works by sequentially growing trees where each tree uses information gained from the previously grown tree. Thus, in contrast to bagging, trees learn slowly over time to prevent overfitting. Instead of fitting to the outcome Y, boosting fits and updates the residuals with each new tree. Boosting has three tuning parameters (James et al., 2013, p. 316–319):

- The number of trees $\mathcal{B}$.
- The shrinkage parameter λ.
- The interaction depth d which determines the number of splits. If $d = 1$ the tree is called a *stump*.

Various implementations of the algorithm have been developed, such as AdaBoost (Freund & Schapire, 1997), stochastic gradient boosting (Friedman, 2002), and a more efficient and scalable implementation called eXtreme Gradient Boosting (XGBoost) (Chen, He, Benesty, Khotilovich, & Tang, 2015).

3.2.5 Support Vector Methods

An entirely different approach than the previously listed methods are *kernel-based methods*. While kernels could in theory be applied to any ML technique, they are generally applied to *support vector machines*.

Maximal margin classifier. Suppose a dataset of n training observations $x_1, \ldots, x_n \in \mathbb{R}^p$ with two observation classes $y_1, \ldots, y_n \in \{-1, 1\}$ that is completely linearly separable. According to James et al. (2013, p. 340), a *separating hyperplane* has the

properties that

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} > 0 \text{ if } y_i = 1,$$
$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} < 0 \text{ if } y_i = -1,$$

and

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}\right) > 0$$

for all $i = 1, ..., n$. A test observation $x_1^*, \ldots, x_p^*$ can thus be classified by looking at the sign $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \ldots + \beta_p x_p^*$ with respect to the hyperplane.

Moreover, a *maximal margin classifier* seeks to maximise the distance (or *margin*) between the hyperplane and the nearest training observation. The observations that are equidistant from the maximal margin hyperplane are known as *support vectors*. The maximal margin classifier is the solution to the optimisation problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximise}} M$$
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{3.11}$$
$$y_i \left(\beta_0 + \beta_1 x_{i1} = \beta_2 x_{i2} + \ldots + \beta_p x_{ip}\right) \geq M \; \forall i = 1, \ldots, n. \tag{3.12}$$

While Equation 3.12 guarantees that an observation is on the correct side of the hyperplane, 3.11 gives the perpendicular distance from the ith observation to the hyperplane given by Equation 3.12 (James et al., 2013, p. 343). Together these constraints ensure than an observation is on the correct side of the hyperplane keeping at least M distance to the hyperplane. The optimisation problem then chooses $\beta_0, \beta_1, \ldots, \beta_p$ to maximise the margin M.

Support vector classifier. Naturally, a dataset is almost never completely separable. Instead, we may want to allow some observations to cross the margin to create a more robust and flexible method. A *support vector classifier* (sometimes called *soft margin classifier* has a nonnegative tuning parameter C which acts as a sort of budget for the amount that the margin can be violated. It is the solution to the optimisation problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximise}} M$$
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$
$$\left(\beta_0 + \beta_1 x_{i1} = \beta_2 x_{i2} + \ldots + \beta_p x_{ip}\right) \geq M\left(1 - \epsilon_i\right),$$
$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C, \tag{3.13}$$

where $\epsilon_1, \ldots, \epsilon_i$ are *slack variables* which take on the value $\epsilon_i = 0$ when x_i is on the correct side of the margin, $\epsilon_i > 0$ when x_i has violated the margin, and $\epsilon_i > 1$ when x_i is on the wrong side of the hyperplane (James et al., 2013, p. 346). If C in Equation 3.13 is 0, it must be that $\epsilon_1 = \ldots = \epsilon_n = 0$ which means there is no budget for error and amounts to a maximal margin hyperplane. Similar to the maximal margin classifier, as James et al. (2013, p. 348–349) note, a support vector classifier solely relies on the

support vectors to determine the margin which is distinct from other approaches, such as LDA, which rely on all observations.

Support vector machines. Both the maximal margin classifier and the support vector classifier perform poorly with non-linear class boundaries. Similar to linear regression, James et al. (2013, p. 350) extend the feature space with polynomials to allow for fitting non-linear boundaries. The algorithm then becomes

$$\begin{aligned} &\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximise}} M \\ &\text{subject to } y_i \left(\beta_0 \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M \left(1 - \epsilon_i\right), \\ &\sum_{i=1}^{n} \epsilon_i \leq C, \epsilon_i \geq 0, \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1. \end{aligned} \tag{3.14}$$

However, with a large number of features computing this algorithm quickly becomes infeasible. To this end, *support vector machines (SVMs)* employ kernels to efficiently execute this idea.

As it turns out, only the inner products are needed between observations to solve the problem presented by Equation 3.14 so that $\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$ which can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle.$$

Furthermore, only $\binom{n}{2}$ inner products are needed between the pairs to estimate α which is nonzero only for the support vectors. Given a set of support vectors $\mathcal{S}$, James et al. (2013, p. 351) then rewrite a linear support vector as

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle. \tag{3.15}$$

Next, James et al. (2013, p. 351) replace Equation 3.15 with a generalisation:

$$K\left(x_i, x_{i'}\right),$$

where K is a function referred to as a *kernel*. The *kernel trick* allows us to quantify distances in feature space, usually in non-linear data such that we have a mapping function of the input space to feature space $\phi : \mathcal{X} \to \mathcal{F}$. This generalised distance measure is more efficiently than calculating inner products (Schölkopf, 2001). Combining a non-linear kernel with a support vector classifier is known as a *support vector machine*.

There are a few considerations to be made when picking a kernel. Naturally, the choice for a kernel depends on the data, that is a *linear kernel* will perform well on data that is (nearly) linearly separable while, for instance, a polynomial kernel would perform poorly. In fact, an SVM with a linear kernel is equal to a support vector classifier. Similar to other methods, the optimal kernel is estimated by looking at the error via cross-validation. Adequate hyperparameters can be tuned via grid search (Chu, Marron, & others, 1991) or adaptive feature selection (Somol, Pudil, Novovičová, & Paclık, 1999). Next, we discuss a number of popular kernels and their general use.

Kernels. As stated before, an SVM with a linear kernel is equal to a support vector classifier (James et al., 2013, p. 352). It uses Pearson (standard) correlation to measure distances between inner products. It works especially well on data with a strong linear relation. Because of this constraint, a linear kernel is seldom used for real-world data that tends to have more complex connections.

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij}, x_{i'j}. \qquad \text{(Linear kernel)}$$

Alternatively, the feature space $\mathcal{F}$ may be extended and replace the generalised quantity measure of the linear kernel with a polynomial (James et al., 2013, p. 352). Essentially, a *polynomial kernel* extends $\mathcal{F}$ in order to fit a support vector classifier in a higher-dimensional space with degree d. Therefore, it allows for more complex relations to be fitted than a linear kernel.

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij}, x_{i'j}\right)^d. \qquad \text{(Polynomial kernel)}$$

Another popular kernel is the *Gaussian radial basis function (RBF) kernel* (James et al., 2013, p. 352). As the name implies, this kernel uses the *Gaussian function* to estimate the Euclidean distance between training and test observations. Thus, if the distance between two observations is large, $\exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right)$, will be tiny and will therefore be irrelevant in determining $f(x^*)$.

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right), \qquad \text{(Gaussian RBF kernel)}$$

where γ is a positive constant.

Finally, we briefly discuss the multi-class classification setting for SVMs. First, in the one-against rest method, Bottou et al. (1994) fit each class K to the remaining $K-1$ classes. They choose the class K for which $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \ldots + \beta_{pk}x_p^*$ is the largest since this amounts to the highest level of confidence. Second, the one-against-one method (Krebel, 1999) fits each class K against another class K' such that $K \neq K'$. As a consequence, this *all-pairs* approach means having to fit $\binom{K}{2}$ number of SVMs. The observation is assigned to the most frequently assigned class from the $\binom{K}{2}$ pairs.

3.2.6 Neural Networks

Neural networks (NNs) are a connectionist system of nodes that are inspired by the biological structure of the brain (Goodfellow, Bengio, & Courville, 2016, p. 13–17). Similar to the brain, the idea in *artificial neural networks* is to find some universal learning function. They are essentially a set of nodes with specialised functions: there is always a set of input and output nodes, and frequently there is also one or multiple sets of *hidden* nodes.

Since the concept of NNs comes from the human brain, the idea has been around for a long time. For example, McCulloch and Pitts (1943) developed the concept of threshold logical units (TLUs) which can be seen as an early form of an activation

function. Since then, advances have been made by Hebb (1949) (development of Hebbian learning), Rosenblatt (1958) (development of perceptrons), and Werbos (1974) who first applied backpropagation to NNs. While extensive usage has been limited in the past, (deep) NNs have seen a large increase in usage in recent years thanks to advancements in computing technology.

We now briefly describe a number of popular NN families for the purpose of exploring which NN architecture is most suitable for the purpose of predicting school advice.

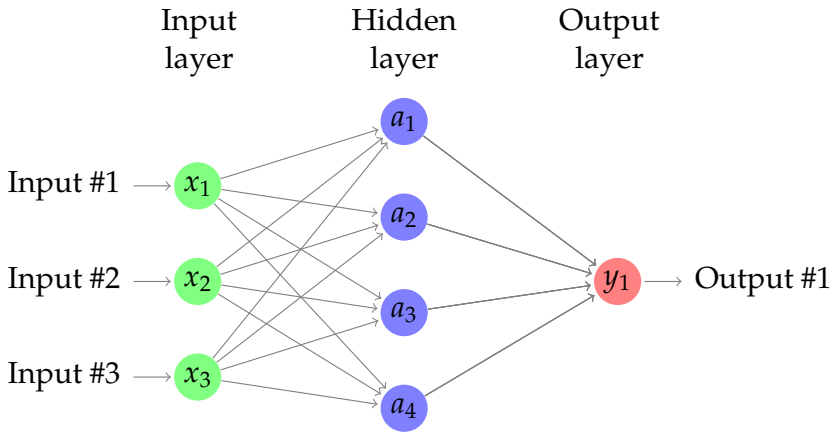

Figure 3.6: A feedforward neural network with three input nodes, four activation functions, and one output node.

Feedforward. *Deep feedforward neural networks*, also called *feedforward neural networks* or *multilayer perceptrons* are the most commonplace type of NNs. A feedforward model seeks to define a mapping $f(x;\theta) = y$ and learn the value of parameter θ through a series of *activation functions* (Goodfellow et al., 2016, p. 169). They are called feedforward NNs because they receive no feedback from a subsequent node. If this is the case, they are known as recurrent neural networks. An example feedforward NN is depicted in Figure 3.6. The displayed model has a *depth* of three (layers) and a *width* of four (nodes). The model is driven to approximate $f(x)$ with $f^*(x)$. Each training example is accompanied by some label $f^*(x) \approx y$ which defines the desired output of the node, but this does not define how each layer should behave. For this reason, all layers — except the input and output layers — are known as *hidden layers*.

The universal learning strategy can be thought of as defining a mapping $\phi(x) \to Y$. As it turns out, ϕ can be any family of mapping functions, such as RBF kernel or a manually engineered mapping. Thus, NNs seeks to learn this mapping ϕ by having a parameter θ (not related to θ in IRT) and a set of weights w that maps from $\phi(x)$ to the desired output. Following Goodfellow et al. (2016, p. 170), the model then becomes

$$y = f(x;\theta,w) = \phi(x;\theta)^\top w. \tag{3.16}$$

Other considerations to be made are that of the *activation function*. For example, we may choose to use a softmax, sigmoid, or hyperbolic tangent function to compute the hidden layer values. To determine the gradients of these functions, *backpropagation* and its modern variants are used.

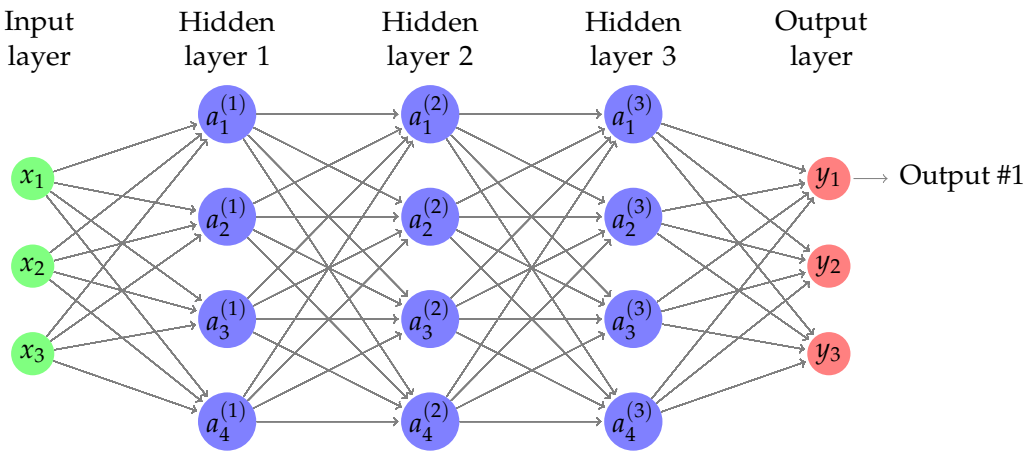

Figure 3.7: A deep neural network with three input nodes, four activation functions three layers deep, and three output nodes.

Deep neural network. A feedforward NN with a single hidden layer is called a *shallow NN*, while a feedforward NN with multiple hidden layers (shown in Figure 3.7) is known as a *deep NN*. The *universal approximation theory* (Hornik, Stinchcombe, & White, 1989) states that a shallow NN with a 'squashing' activation function (such as the sigmoid function, see also the paragraph on logistic regression in Section 3.2.3) can *represent* any non-linear function. However, just because a shallow NN can *represent* this universal learning function, does not mean it can actually *learn* it (Goodfellow et al., 2016, p. 198–199).

The universal approximation theory also states that a shallow NN with enough units can approximate this function to any extent desired. However, Barron (1993) shows that in a binary case, the number of possible binary functions on vectors $\in \{0,1\}$ is 2^{2^n} which means it becomes infeasibly large to train. By subdividing this problem into a number of tasks, they can reduce the number of units; this thus introduces the notion of *deep neural networks* which contains multiple layers. There have been a number of successes indicating that a greater depth corresponds to better generalisation (Bengio, Louradour, Collobert, & Weston, 2009; Goodfellow, Bulatov, Ibarz, Arnoud, & Shet, 2014)

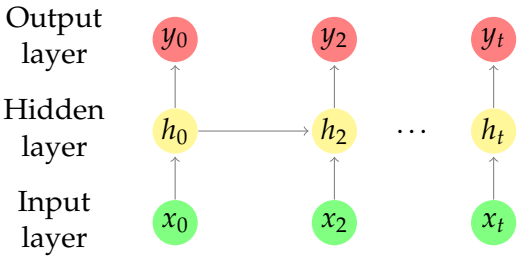

Figure 3.8: A recurrent neural network of width t.

Recurrent neural network. A *recurrent neural network (RNN)* is a family of NNs suitable for handling a sequence of values $x^{(1)}, \ldots, x^{(t)}$. This idea is visually represented in Figure 3.8. This computational graph has been *unfolded* from a graph with a repetitive structure, such that the system refers back to itself t times. The advantage of this is that RNN can process much longer sequences than networks

without sequence-based optimisations. The RNN can also be extended with other operations where only a part is recurrent. Goodfellow et al. (2016, p. 376) rewrite Equation 3.16 to represent the recurrent state h of the hidden units in the network:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta). \tag{3.17}$$

Due to the temporal nature of the RNN, practical applications include unsegmented tasks such as handwriting and speech recognition; recent success has also been made with RNN in determining the re-hospitalisation rate after heart failure (Valk, 2018). Further extensions are possible by applying gated states which (partly) reset the progress and are used in, for example, long short-term memory networks (LSTMs), while challenges include that of the vanishing and exploding gradient problem (Goodfellow et al., 2016, p. 290).

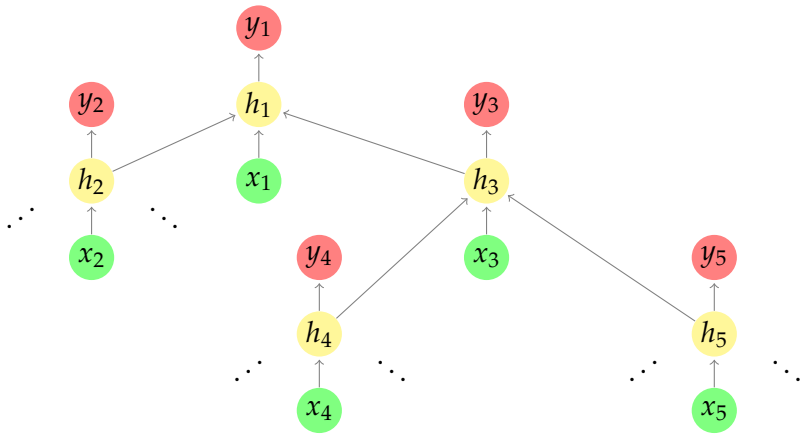

Figure 3.9: A recursive neural network has a deep tree-like structure.

Recursive neural network. A generalisation of a RNN can be made such that it is a *recursive neural network*. Figure 3.9 depicts a simple example of the tree-like structure in this model. While this structure resembles that of a decision tree, the inner workings are vastly different. Instead of applying the unfolding principle as in RNNs, operations are applied recursively to produce a *structured prediction*.

One clear advantage of a recursive neural network over a RNN is that the depth of the model for a sequence $\mathcal{T}$ is drastically reduced, going from $\mathcal{T}$ to $O(\log \mathcal{T})$ (Goodfellow et al., 2016, p. 401). Practical applications of the tree are mostly in natural language processing (NLP), such as Stanford's CoreNLP (Manning et al., 2014).

3.3 Summary

In this chapter, we gave an overview of both domain-specific and domain-agnostic statistical methods from the domains of IRT and ML. We found that IRT is most suitable as a domain-specific method since it relies on the underlying theory of a test and therefore has an edge over other methods which do not utilise this knowledge. Moreover, tests in the SMS are designed in such a way that items overlap which allows IRT to estimate certain parameters.

As for domain-agnostic methods, we looked at the different categories of ML and described both regression and classification models. Since our goal is to predict school

advice from both continuous and discrete variables, we must use methods which can handle these values. Furthermore, the problem of predicting school advice is a supervised learning problem, since the actual school type of students at T_3 is known.

II

Model Construction

The previous part discussed approaches for estimating test scores and explored possible models for predicting school advice. In this part, we describe data exploration and model construction of the approaches laid out in Part 1 for the sake of reproducibility and insightfulness

Chapter 4

Data Description

First, we describe the data set given as output by the SMS. As stated previously, the SMS consists of four measurement moments:

1. T_0 at the start of year 1.
2. T_1 at the end of year 1.
3. T_2 at the end of year 2.
4. T_3 at the end of year 3.

Since each student takes (at maximum) one test at each of these measurement moments, the data can essentially be subdivided into four regions. As also explained previously, there are multiple test versions depending on the school type. In this section, we scrutinise these intricacies of the data set.

In total, the data set contains 71 564 rows and 22 columns. Each row contains one student per test moment. Since there are four test moments, there are 17 891 students in total. Appendix B lists all columns in the data set and provides a brief description of their contents.

Student ID

Arguably the most important variable in this data set is the student ID which connects students across test moments. For this reason, it is paramount that student IDs are indeed unique to students and link those (same) students to each other across measurement moments. As it turns out, this is not a trivial task. While it is possible to do matching based on attributes such as name, school, date of birth et cetera, it is even better to ask the school directly to confirm that a student is indeed said student. Cases where this was not possible have not been included in this data set.

Test Moment and Test Version

As stated previously, progress tests in the SMS are divided into four *test moments*: T_0, T_1, T_2, and T_3.

The variable *test_id* indicates:

1. which school type, and
2. which measurement moment.

Table 4.1: The most frequently occurring report dates of test scores per test version.

Test version	Test date	Frequency
T0		
515	2015-10-21	513
516	2015-10-21	526
517	2015-10-21	850
518	2015-10-21	631
T1		
523	2016-06-08	306
524	2016-05-31	598
525	2016-05-31	611
526	2016-06-06	517
T2		
531	2017-04-19	118
532	2017-04-19	189
533	2017-04-26	965
534	2017-04-25	425
535	2017-04-20	465
T3		
541	2018-06-06	132
542	2018-07-05	155
543	2018-06-06	387
544	2018-06-06	552
545	2018-05-02	456

The following test versions exist: 515, 516, 517, 518, 523, 524, 525, 526, 531, 532, 533, 534, 535, 541, 542, 543, 544, and 545. However, this specifies neither school type nor test moment. One way to find which test version belongs to which test moment is to simply look at the date the scores were reported. Because these dates are day-specific, we simply look at the most frequently occurring one. As indicated by Table 4.1, test moments are grouped by test version chunks, i.e. 515 to 518 are tests for measurement moment T_0, 523 to 526 for T_1, and so forth. To simplify matters, we create a new column 'test moment' that contains the values T0, T1, T2, and T3.

As for the school type, we can similarly look at the most occurring school advice given by the mentor (see Table 4.2. If we assume these to be correct, we see that each test corresponds to school advice levels BB+, BB, KB, GL, HAVO, and VWO[1].

School Advice

Another important variable in this data set is *school_advice* which encodes school advice given by the student's mentor. Concretely, before a student takes a test, the student's mentor is asked which school type they think is most suitable for the student. Mentors are asked for either a specific school type or a mix of school types, i.e. a so-called mixed bridge year. Since this advice is usually the current school type and is therefore fairly accurate, the school advice by the mentor is used as a gold

[1]See Appendix A for details of distinct school types.

Table 4.2: Most frequently occurring school advice per test version.

Test version	School advice	Frequency	Percentage	Total number of students
515	KB	1596	51.73	3085
516	GL	3873	89.80	4313
517	HAVO	4751	78.99	6015
518	VWO	3449	77.02	4478
523	KB	1625	53.10	3060
524	GL	3897	91.82	4244
525	HAVO	4894	80.57	6074
526	VWO	3425	75.89	4513
531	BB+	689	53.37	1291
532	KB	1731	97.58	1774
533	GL	4089	97.10	4211
534	HAVO	5482	93.01	5894
535	VWO	3476	73.63	4721
541	BB	822	61.71	1332
542	KB	1802	99.23	1816
543	GL	4340	99.56	4359
544	HAVO	5573	98.29	5670
545	VWO	3702	78.53	4714

Table 4.3: Number and percentages of students by school advice.

School advice	Frequency	Percentage
VMBO	29908	41.79
HAVO	23568	32.93
VWO	18088	25.28

standard (or ground truth) in this research. That is, we assume that the school advice given by the mentor is correct since this advice is based on a whole year of experience whereas a test is a measure of only a single moment in time.

As can be seen in Figure 4.1, assigned classes are usually either BB+, BB, KB, GL, HAVO, VWO, or Gymnasium. Mixed classes do occur but much more infrequently. To simplify matters, we may group school types BB+, BB, BB/KB, and GL/HAVO as VMBO, HAVO and HAVO/VWO as HAVO, and VWO and Gymnasium as VWO. The result can be found at the bottom half of Figure 4.1. Disregarding incomplete or incorrectly filled in values, Table 4.3 shows that approximately 42% of the students attends VMBO, 33% attends HAVO, and 25% attends VWO. However, this distribution can vary depending on the measurement moment.

In Figure 4.2, one can observe that school advice does not vary much between years. However, as many as 19% of students switches school type (e.g. from VMBO to HAVO) between the first and third year. From this 19% of students, 42% went to a more vocational school type (downstream) and 58% went to more academic school type (upstream). Hence, school advice between years does not vary much since almost as many students go upstream as downstream.

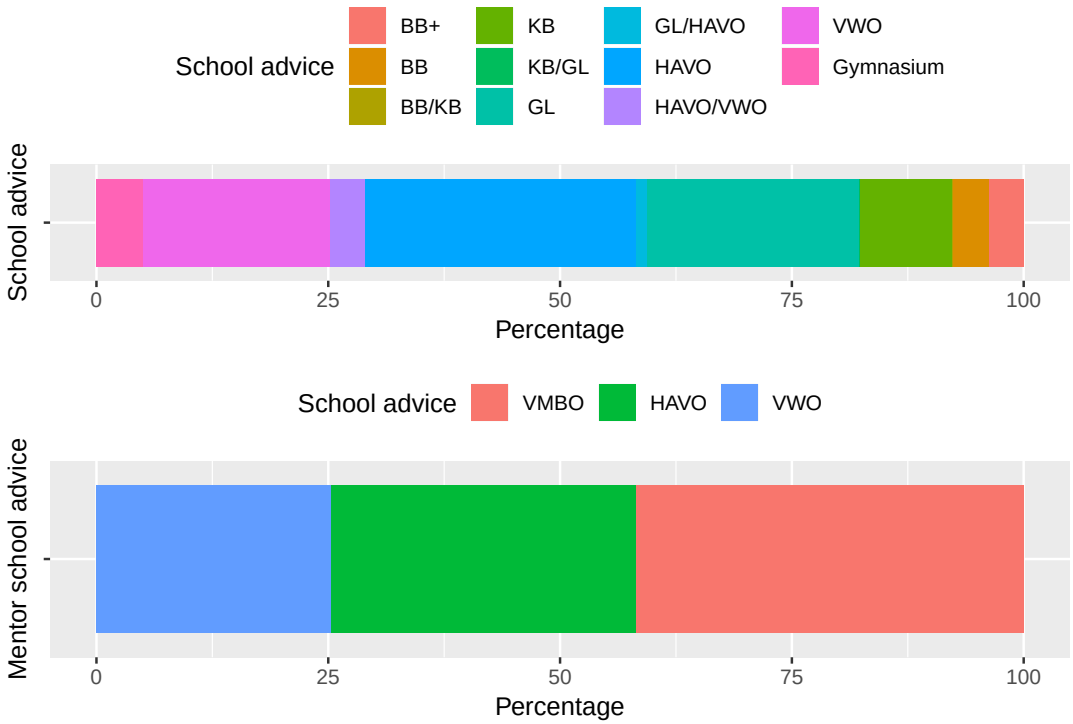

Figure 4.1: Distribution of both aggregated and non-aggregated school advice.

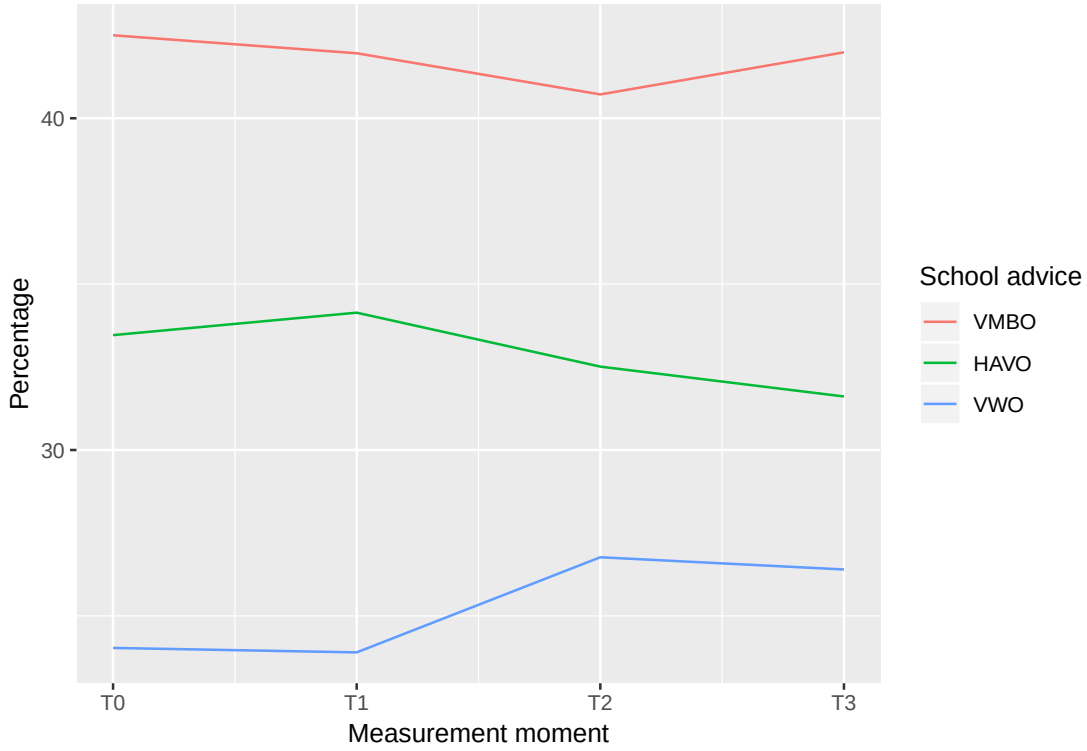

Figure 4.2: Balance between school types remains stable over test moments.

Table 4.4: Number of items per test version.

Test version	Number of items
515	290
516	293
517	292
518	298
523	348
524	349
525	346
526	357
531	349
532	350
533	349
534	357
535	357
541	354
542	357
543	357
544	357
545	360

Table 4.5: Descriptive statistics for the end test of primary education.

school_advice	N	Mean	SD	Q25	Q75
VMBO	517	531.42	6.76	528	536
HAVO	917	540.64	4.73	538	544
VWO	959	546.08	3.54	544	549
Total	2393	540.83	7.32	537	546

Scored Response String (Dichotomous)

Data from the *response* column is a string representing a candidate's responses to the item. Table 4.4 lists the number of items per test version. From this table, it is clear that tests on test moment T_0 have fewer questions than those in other test moments.

Figure 4.3 shows the density distribution of the sum score per test moment. That is the sum of the endorsed items per test moment. While one may expect that tests become more difficult as students progress, students also become smarter and are thus more able to provide the correct answers. On the other hand, tests are designed such that easier school types get easier questions. This approach of test design is also reflected in IRT as described in previous chapters.

Standardised Score on the End Test of Primary Education

At the end of primary education, most primary schools in the Netherlands conduct an end test of primary education (see Appendix A). In short, the end test of primary

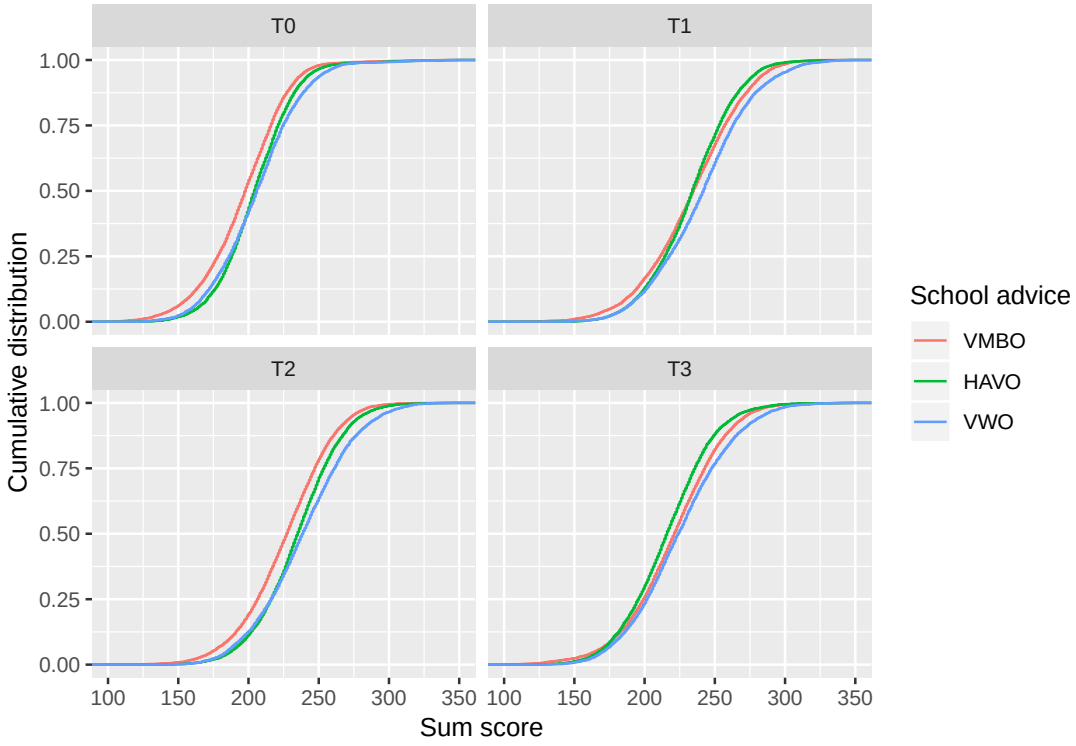

Figure 4.3: Sum scores by school advice per test moment.

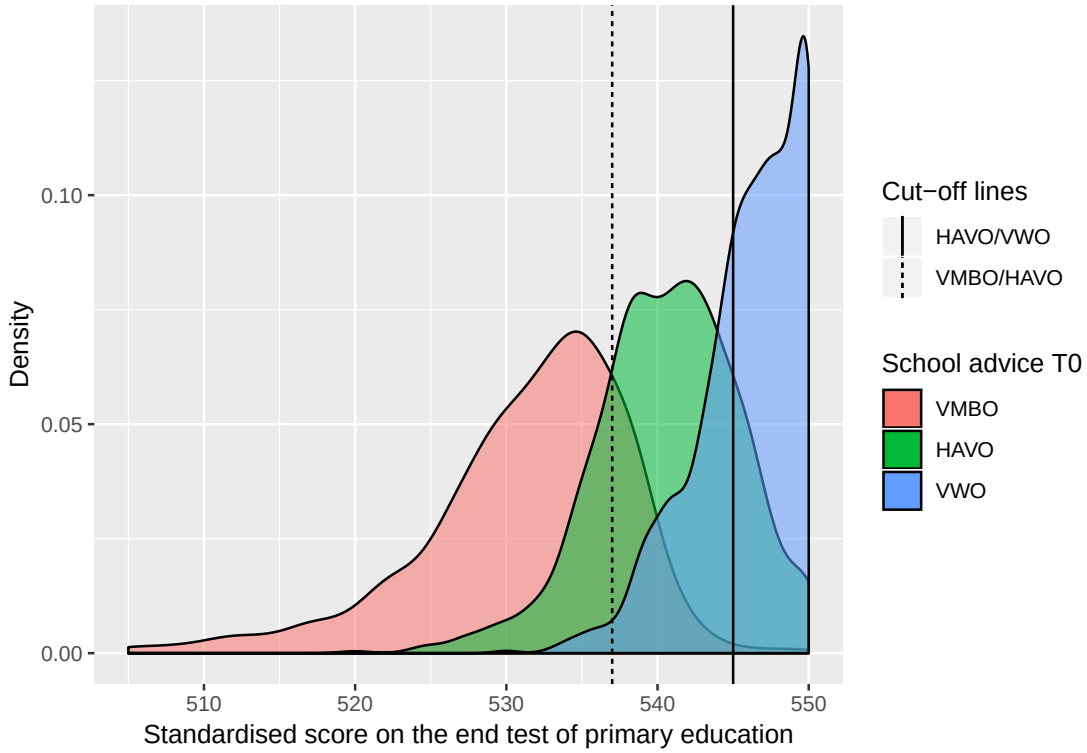

Figure 4.4: Scores on the end test of primary education by school advice. Higher values correspond to a more theoretical school type.

education helps determine what is the most suitable school type for students and is thus very similar to the tests used in this research. Until 2015, Cito was the only company authorised to develop the end test of primary education. In 2018, only 56% of all schools used the end test developed by Cito which has led to a multitude of problems such as incomparable school advice, different levels of difficulties, and schools switching between tests to seemingly achieve the 'best' advice (Swart, Van den Berge, & Visser, 2019, p. 6–7). For this reason (and possibly other reasons as well), as little as 13% of scores on the end test of primary education are *not* missing.

In total, we are left with 2 393 students whose score on the end test is *not* missing. Figure 4.4 presents density functions per school type. The score ranges from 500 to 550. The dotted borders represent the cut-off points for determining school advice as used in the end test of primary education (i.e. 537 and 545). These cut-off points have been determined by Cito (Van Boxtel et al., 2011, p. 55). While the cut-off point between VMBO and HAVO seems accurate, the line dividing HAVO and VWO is shifted too far to the right.

As concluded in Chapter 2.2, 80% of all students follow the advice given by the end test of primary education (Van Boxtel et al., 2011, p. 88). In this data set, however, only 67% of students followed their end test school advice.

Year of Test

The variable *test_year* indicates the year the test was taken in. Of course, this should be similar to the year the result was reported in (i.e. the variable *test_time*). The years corresponding to these tests are 2015/2016, 2016/2017, and 2017/2018. Of course, this corresponds to T_0/T_1, T_2, and T_3 respectively since students must progress over at least three years in order to be able to take all tests.

Specific Date of Test

Figure 4.5 depicts the number of measurements over the past few years. Additionally, dotted lines have been plotted to show the separation of test moments. Notably, tests do not need to be taken at a specific date so schools have some freedom to choose regarding which month suits them best.

Computerised Test

The variable *digital* refers to the medium that was used to take the test, i.e. paper or digital. For this data set, none of the tests were computerised.

Date of Birth

In addition to data about the tests themselves, there is of course also data about students. The column *birth_date* lists the date of birth for each student. Naturally, some of the data is incorrect. For example, the youngest person in the data set was born on 2015-01-10 while the oldest person was born on 1899-12-30. These extreme dates are, however, a bit arbitrary since the data was trimmed to ensure that there are no dates of birth such as 1899-01-01. Furthermore, there are 292 empty values. While

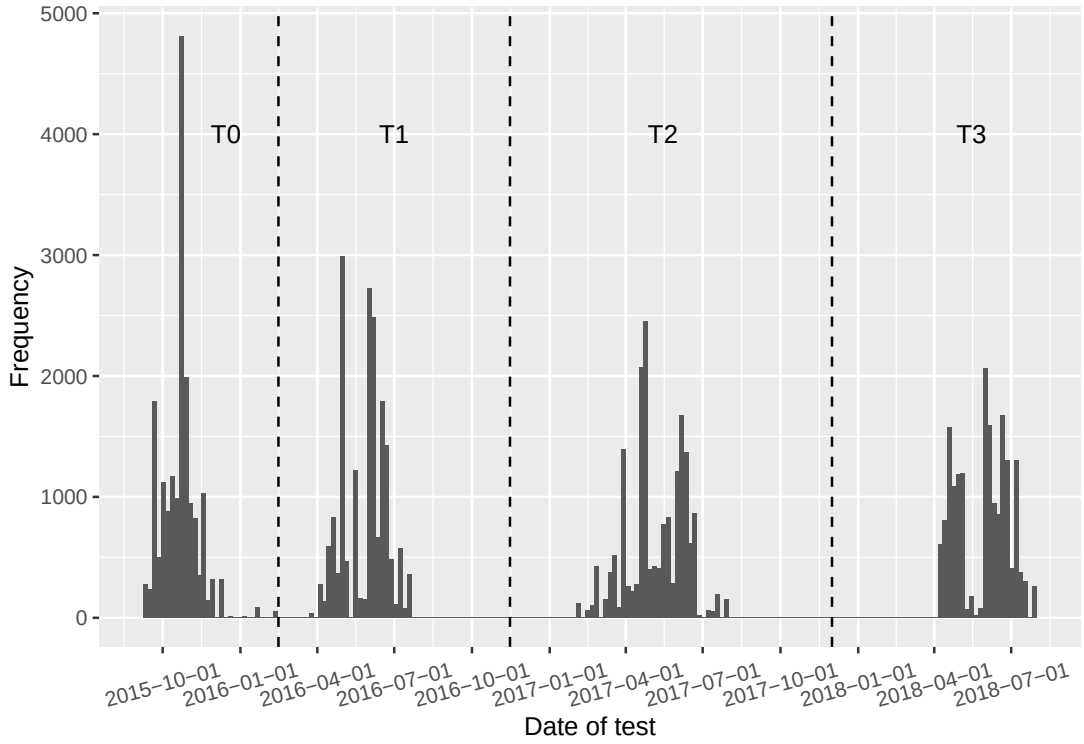

Figure 4.5: Tests registered over time. Text with test moment and dotted line indicating a new test moment have been added for visual aid.

dates of birth are not used in IRT, we must ensure empty and incorrect values are not used when applying ML techniques.

Sex

The column *sex* can take on the values female, male, and not filled in. In total, there are 34 014 boys, 37 235 girls, and 315 cases where the sex was not filled in. There are no missing values.

Primary Language spoken at Home

Students may also fill in which language they speak at home. In total, there are 60 566 students (85% of total) who speak Dutch at home, 5 860 students (8.2% of total) who speak a language other than Dutch at home, and 5 138 students (7.2% of total) who did not fill this in. Thus, not counting those whose language was not filled in, 9.7% did not speak Dutch as their primary language at home.

Sector of the Student.

In the second year of VMBO education, students can choose a *sector* they want to specialise in. They can choose between the profiles *'engineering and technology'*, *'care and welfare'*, *'business'*, and *'agriculture'*.

As can be observed from Figure 4.6, care and welfare (30% of students) is the most popular sector to specialise in while agriculture (15% of students) is the least popular

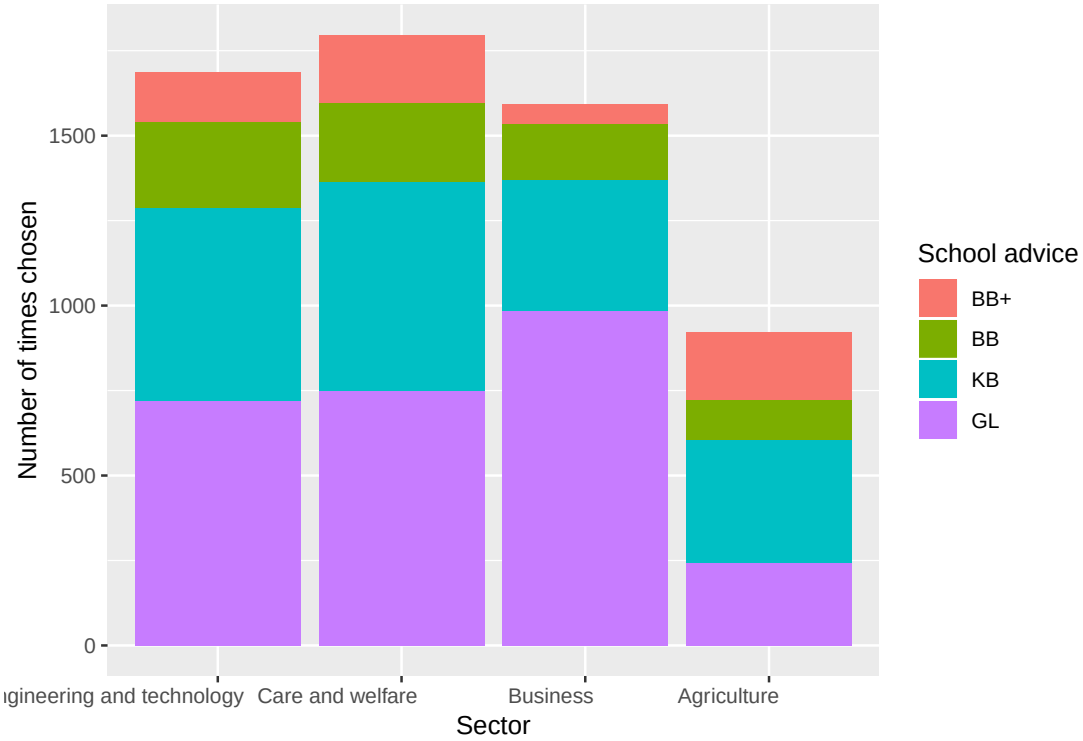

Figure 4.6: At the VMBO level, students can choose which sector to specialise in during their last three years.

sector. Interestingly, agriculture is the most popular sector for BB+ students (33%) followed by agriculture (33%). For BB students, both agriculture (15%) and business (21%) are relatively unpopular while both care and welfare (30%) and engineering and technology (33%) seem to attract the most students. On the contrary, sectors are similarly popular with KB students with the exception of care and welfare (32%) which is slightly more popular. Finally, GL students massively choose the sector of business (37%) while agriculture (19%) is by far the least popular. Note that mixed classes are not shown in Figure 4.6 due to a very limited amount of data.

Educational Track

At the HAVO level, students can choose between various educational tracks students will follow in year 4. Students can choose between the profiles *'Culture and Society (CS)' 'Economics and Society (ES)'*, *'Nature and Health (NH)'*, and *'Nature and Technology (NT)'*. The culture and society profile and the economy and society profile can be seen as preparation for humanities or economics studies, while nature and health provide access to health sciences studies, among others. Nature and technology includes subjects such as chemistry and physics and therefore prepares for a technical (STEM) studies.

The number of students choosing a certain educational track is depicted in Figure 4.7. From this figure, we see that economics and society is by far the most popular profile to choose (41% of total) while culture and society is the least popular profile (10% of total). Interestingly, as many as 46% of HAVO students chose the profile of economics and society, while only 34% of VWO students chose that same profile. Conversely, 26% of VWO students chose for science and technology, while only 15 opted for

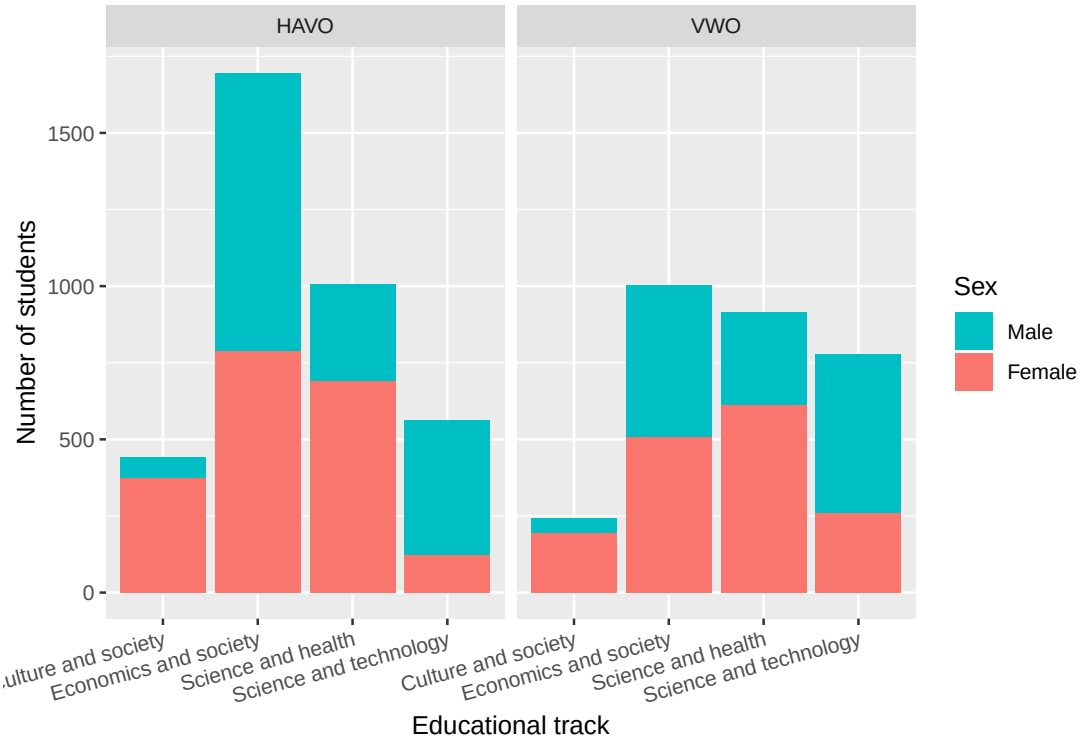

Figure 4.7: Educational profile by sex and test moment.

Table 4.6: Descriptive statistics on number of students per class.

School type	Mean	Median	SD	Max	Min
VMBO	8.77	7	6.63	59	1
HAVO	13.12	14	7.98	36	1
VWO	14.73	16	9.34	64	1

this technical side. Thus, it appears that VWO students tend to favour a technical (i.e. STEM) track, while HAVO students are more likely to follow a business-oriented profile.

Furthermore, different sexes seem to prefer different profiles as well. For instance, only 3.7% of boys chose to pursue a culture and society profile. On the other hand, 45% of boys wanted to have an economics and society profile. Girls follow the same trend, although there are different when it comes to science and health, and science and technology. That is, 37% of girls versus 20% of boys chose for the science and health profile. Conversely, 31% of boys versus 11% of girls opt for the science and technology track.

Class

The variable *class* refers to the school class a student is in. Usually, this is a combination of numbers and letters to refer to the year a student is in (e.g. 1, 2, or 3) and to which class since there are frequently too many students to fit in one class. For instance, if there are 80 students in year 1, it makes sense to divide this into groups of 20 and labelling them as 1A, 1B, 1C, and 1D. Table 4.6 shows the number of

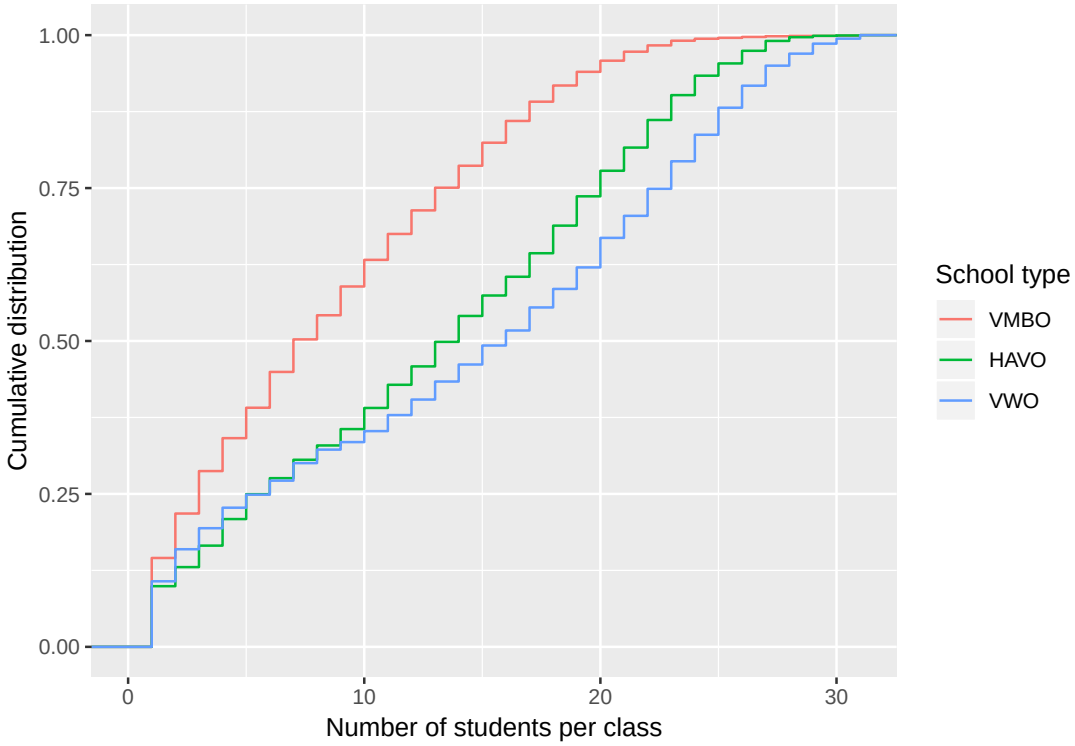

Figure 4.8: These distribution functions show how VMBO classes are typically smaller, whereas both HAVO and VWO classes have around 20-25 students.

students per class per school type. From this table, we see that a typical VMBO class contains fewer students than a HAVO class and even fewer students than a VWO class. Notably, the standard deviation and the maximum number of students per class is very high.

Figure 4.8 depicts a possible explanation for this high dispersion. That is, the density function for the number of students per class seems to be bimodal for both HAVO and VWO classes. While this does not explain why there are classes with fewer than 5 students, it does shed some light on what the 'true' number of students in a HAVO or VWO class is. As for VMBO, classes are usually smaller since educational is vocational and therefore requires more active guidance for students. On top of that, classes with mixed school type in VMBO education (e.g. BB/KB or KB/GL) have fewer students as well. Classes with more than 40 students are likely to be an anomaly.

School ID

The variable *school_id* refers to a school. In total, there are 239 schools in the data set which means there are on average of 299 students per school.

Chapter 5

IRT Analysis

This chapter describes the modelling of response data using different IRT models, a process that is sometimes referred to as item calibration. Once item calibration has been completed successfully, the IRT model can estimate ability scores or person parameters. These ability scores will later be used for predicting the future educational track of students.

5.1 Data preprocessing

Since data is in a compressed format, we must first do some additional cleaning (and add some information in the process). Table 5.1 lists the different subjects that each test measures. Note that at test moment T_0, the subjects *mathematics 1* and *mathematics 2* are not measured yet since this is usually not taught in primary school. For the purpose of calibrating items, it is imperative to know which items overlap between tests. To this end, we add item codes to responses to link different tests so that it can be used to compare students over different test versions.

To calibrate items on a unidimensional scale, we use the *Dexter* package in R (Maris, Bechger, Koops, & Partchev, 2019). This package requires data in a long format which means it has to be transformed. Starting with a long format, we use the `melt` function from the *reshape2* package (Wickham, 2007) to convert the list of items and item codes to a long format and to combine this with the responses in the data set.

5.1.1 Creating Scoring Rules for Dexter

Dexter also requires scoring rules for the items. These rules indicate what type of response is valid for an item and how this response should be scored. Since items in the tests are dichotomous, these are simply the categories of responses (i.e. 0 or 1) per item. Items not filled in by a student are encoded as 9 but are be scored as 0 here.

The output of the scoring rules is displayed in Table 5.2 and the first six rows of the input for *Dexter* are given in Table 5.3. The terminology is a bit different since *Dexter* expects these exact column names, but it should be obvious that a `person_id` is a `student_id` and a `booklet_id` is a `test_id`. In total, the data in long format has 24 231 337 rows and 4 columns. Finally, the data is loaded into an SQLite database which serves as a backend to *Dexter*.

Table 5.1: There are 10 distinct subjects in this data set. To identify subjects, we add a prefix to the item codes.

Subject	Abbreviation
Dutch reading skills 1	NL1
Dutch reading skills 2	NL2
Dutch vocabulary	NLWA
Dutch language skills	NLTA
English 1	EN1
English 2	EN2
Arithmetic 1	R1
Arithmetic 2	R2
Mathematics 1	WI1
Mathematics 2	WI2

Table 5.2: Structure of the scoring rules used in dexter. The column response refers to the response given by the student while the item_score is the score used by dexter to assess the ability.

item_id	item_score	response
NL197531	1	1
NL197531	0	0
NL197531	0	9
NL197532	0	0
NL197532	1	1
NL197532	0	9

5.2 Estimating Item Parameters

As described in 3.1, person parameters can be estimated by using maximum likelihood once the item parameters have been estimated. Thus, we first create models that estimate these item parematers, review these parameters, and then move on to assessing person ability in the next section. A multidimensional approach has also been attempted and is described in Appendix C.

5.2.1 CTT

Usually before any IRT analyses are performed, some classical test theoretical (CTT) item and test statistics are computed. Figure 5.1 depicts an empirical cumulative distribution function (ECDF) of the p-values per test moment. In this context, a p-value is defined as the proportion of correct answers in relation to the total number of given answers. From this figure, we see that especially the subject of mathematics was a more difficult test for students since the p-values are much lower than for other subjects. Additionally, we can look at the test statistics given in Table 5.4. Because of the high number of items per test version, it is not surprising to find a high level of Cronbach's alpha, indicating that the tests have very high internal consistency.

Table 5.3: This table gives the first few rows of the data that is actually loaded into a database to be used by *Dexter*. Here, the response column corresponds to the one in the rules.

person_id	booklet_id	item_id	response
729870901	515	NL197531	1
729870975	515	NL197531	1
729870899	515	NL197531	1
729870935	515	NL197531	0
729870908	515	NL197531	0
729870932	515	NL197531	1

Table 5.4: Test statistics for the different test versions showing, amongst others, Cronbach's alpha, the mean proportion of correct answers across items, and various correlations between tests, items, and responses. The high alpha levels show that there is a high consistency in the different test versions.

Test version	# of items	Alpha	Mean p-value	Mean item-test correlation	Mean item-response correlation	Max test score	# of students
515	290	0.952	0.640	0.262	0.249	290	3085
516	293	0.911	0.660	0.193	0.176	293	4313
517	292	0.926	0.678	0.213	0.197	292	6015
518	298	0.933	0.664	0.220	0.206	298	4478
523	348	0.960	0.667	0.264	0.254	348	3060
524	349	0.931	0.642	0.201	0.187	349	4244
525	346	0.938	0.653	0.211	0.198	346	6074
526	356	0.944	0.644	0.219	0.207	356	4513
531	349	0.950	0.620	0.236	0.224	349	1291
532	350	0.936	0.640	0.209	0.196	350	1774
533	349	0.950	0.620	0.236	0.225	349	4211
534	357	0.952	0.633	0.239	0.228	357	5894
535	357	0.955	0.640	0.245	0.234	357	4721
541	354	0.961	0.577	0.262	0.251	354	1332
542	357	0.947	0.613	0.228	0.217	357	1816
543	357	0.952	0.586	0.240	0.228	357	4359
544	357	0.965	0.557	0.273	0.264	357	5670
545	360	0.961	0.586	0.260	0.250	360	4714

5.2.2 Estimate Item Parameters in Dexter

First, let us take a look at the test design in *Dexter*. This design is displayed as a network of nodes in Figure 5.2. For calibration purposes, nodes are ideally connected either directly or indirectly, this is not the case with this data set. In the actual SMS, a *pre-test* is used to connect the test moments but this data is not at our disposal (see Section 3.1 for an explanation of a pre-test). Since there are very few or none overlapping items, the design is not connected and hence *Dexter* is unable to concurrently estimate item parameters over all test moments. Thus, we have to estimate one model per test moment (T_0, T_1, T_2, and T_3). To do this, we use the `fit_enorm` function from the *Dexter* package.

Results can be found in Figure 5.3. Analogous to the results found in Figure 5.1, the subjects *mathematics 1* and *mathematics 2* have been experienced as more difficult test subjects. Item difficulties for the remaining item subjects seem to be comparable. Naturally, this in line with our expectations since Figure 5.1 already shows that the p-values follow approximately the same trend as the item difficulties.

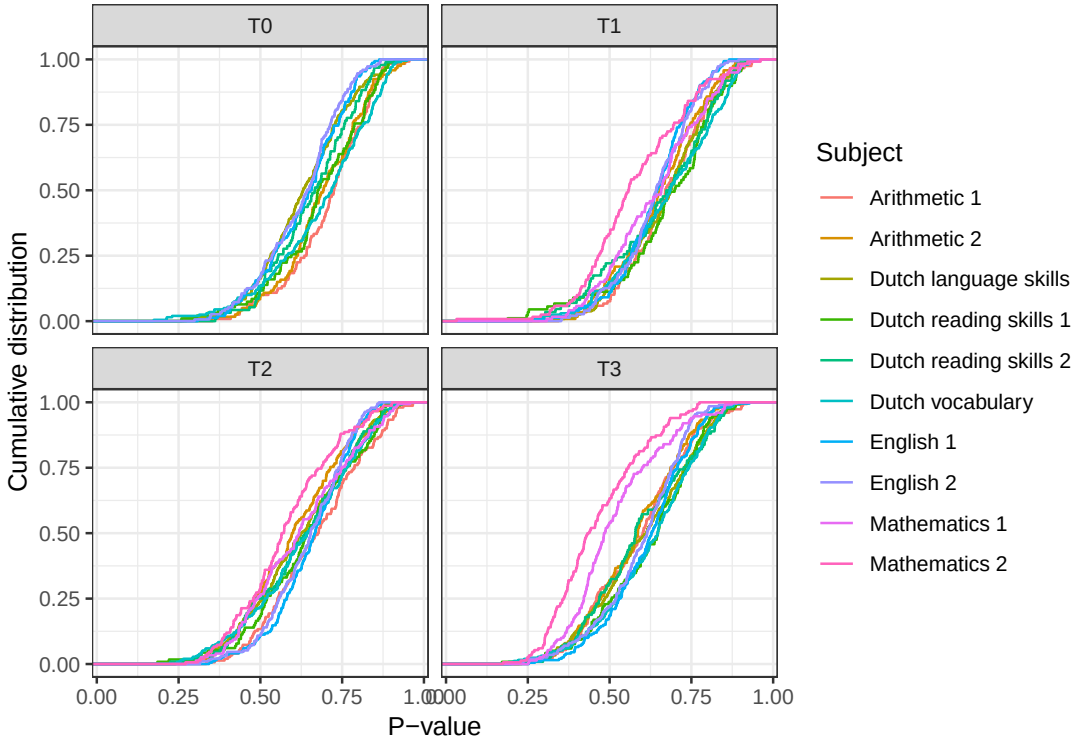

Figure 5.1: This empirical cumulative distribution function (ECDF) shows the proportion of correct answers (p-values) per test moment and coloured by subject.

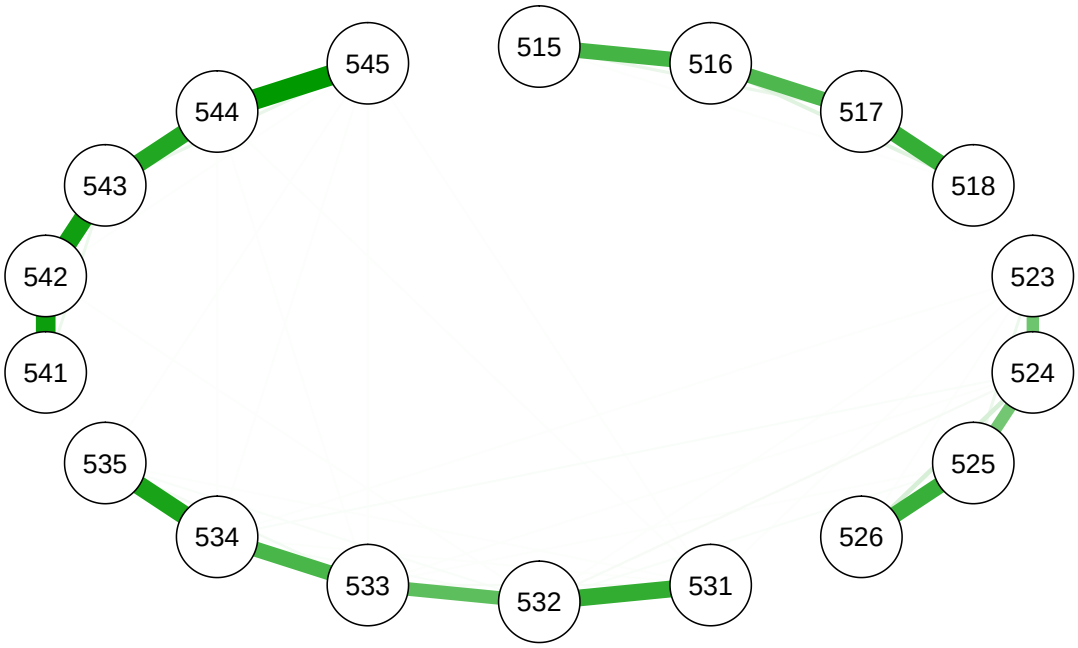

Figure 5.2: By presenting the tests as nodes and connecting overlapping items, we see that the design is unconnected.

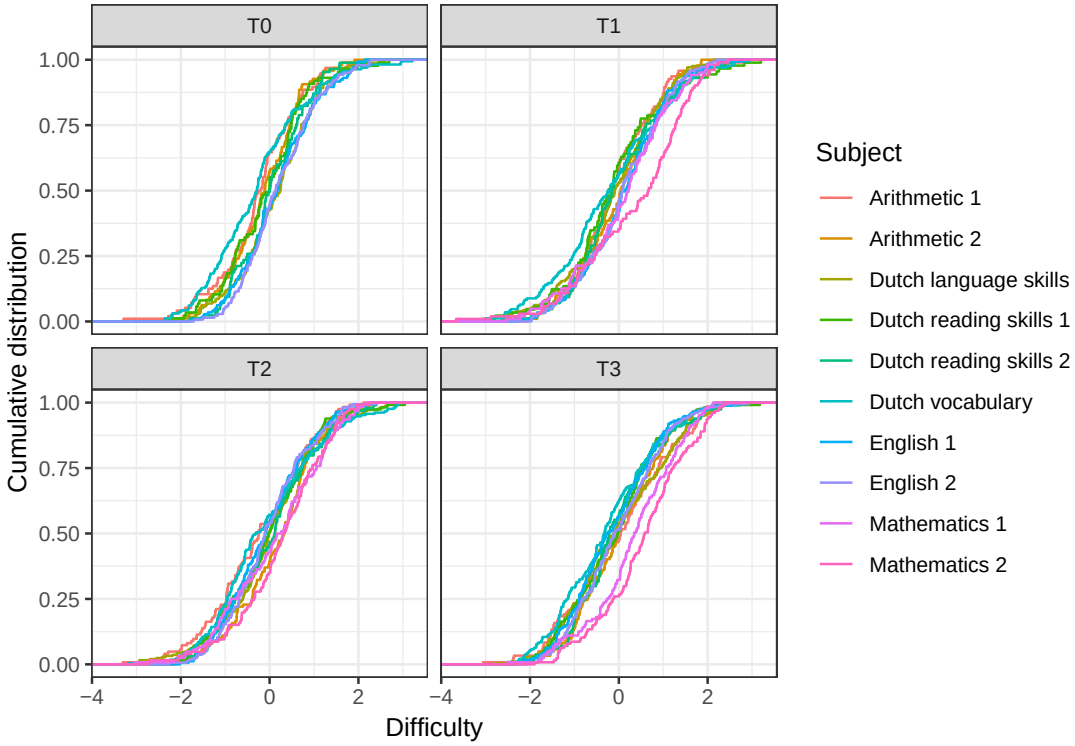

Figure 5.3: Empirical cumulative distribution functions (ECDF) for difficulty parameters per test moment.

5.3 Estimating Person Parameters

Besides estimating item parameters, the goal of IRT is to estimate latent trait scores (in this case students' ability). First, we estimate person parameters on a unidimensional scale using *Dexter* and expected a posteriori.

Estimating person parameters in *Dexter* is straightforward. Using the function `ability`, we estimate the corresponding person ability and visualise them together with the observed score in Figure 5.4. Unsurprisingly, a higher observed score leads to a higher ability estimate. Additionally, each facet in the plot has either four or five lines which correspond to the number of test versions for that test moment. We can thus show how each test version has been designed for a specific school type (although they can be administered to students of other school types as well).

Since abilities have been calibrated on a unidimensional scale per test moment, the ordering of students' abilities only applies within a test moment. In other words, the ability of students does not necessarily increase or decrease over test moments and stays the same if their ability progresses just as much as other students; if they stay in the same ability quantile, their ability score stays approximately the same as well in relation to other students. We may, therefore, report on the correlation between test moments to show to what extent abilities stayed the same across students. We find there to be a strong correlation between abilities on test moment T_0 and T_1, $\rho = 0.91$. While there is still a strong correlation between T_0 and T_2, $\rho = 0.83$, this correlation gets weaker across test moments — between T_0 and T_3 the correlation is mediocre, $\rho = 0.76$. Between contiguous test moments, however, the correlation stays around $\rho = 0.89$.

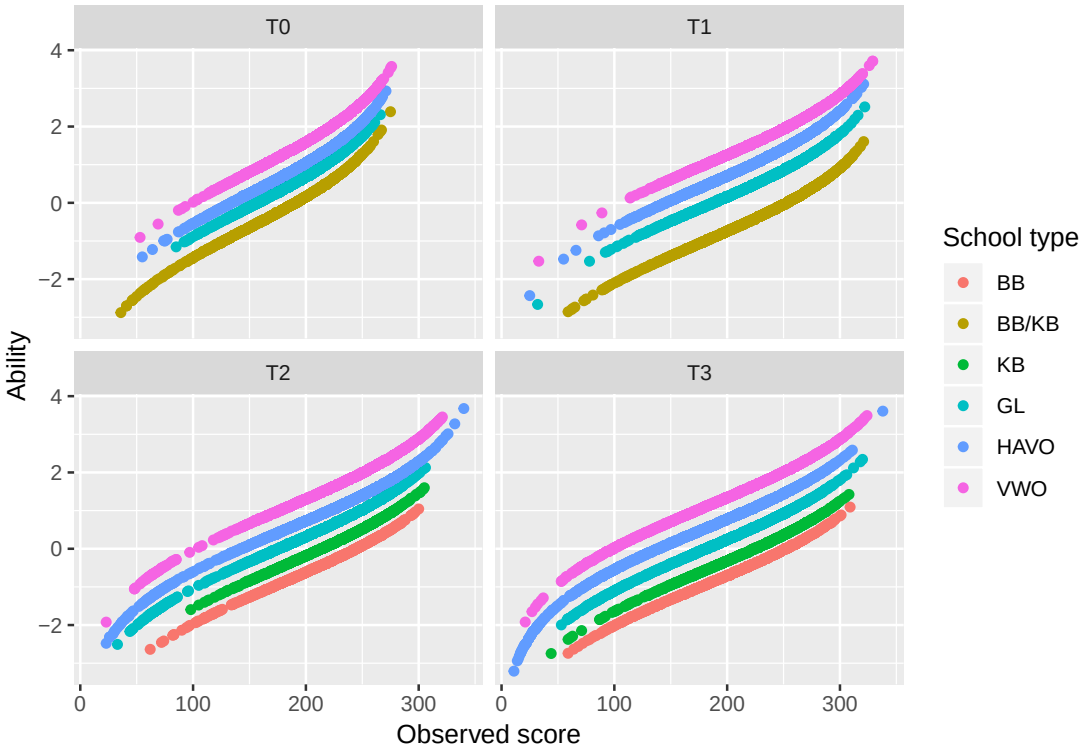

Figure 5.4: As expected, a higher score corresponds to a higher ability and a more theoretical school type corresponds to a higher ability.

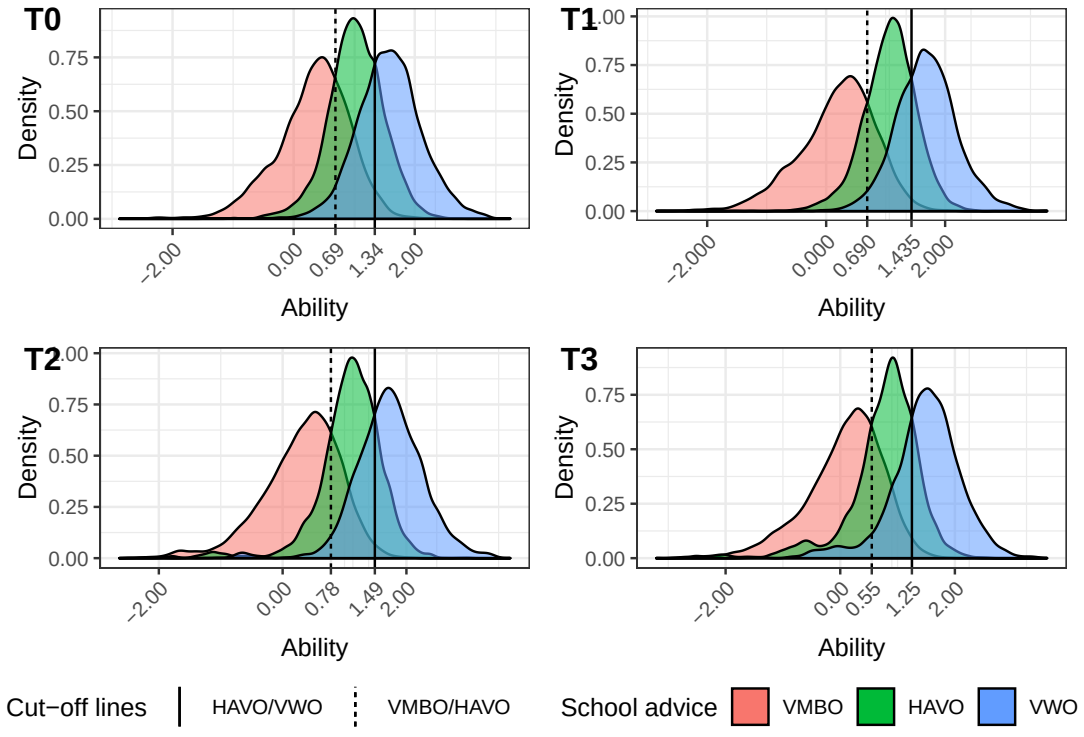

Figure 5.5: Cut-off lines have been drawn between the intersections of the density functions by visual inspection. By doing this, we can determine what ability is needed to get certain school advice.

5.4 Classifying Students

However, the goal of this research is not to estimate ability but to determine school advice. To this end, we can create density functions and colour them by the class labels of T_3, thereby effectively inspecting the fit of the ability at some test moment in relation to school advice at T_3. We can then draw borders by visual inspection to determine the cut-off lines. This process is displayed in Figure 5.5. The plot shows that the borders lie on different points. One reason for this is the fact that the mean is slightly different across different test moments — case in point, the mean ability at T_0 ($M = 0.88, SD = 0.75$) is larger than the mean ability at T_3 ($M = 0.67, SD = 0.86$). Therefore, assuming that abilities stay the same (numerically) across test moments would be incorrect. A simple paired two-tailed t-test indeed proves the distributions are *not* similar, $t(17\,890) = 50$, $p < .001$. Of course, this problem could be easily overcome by standardising the distributions where a t-test would then point out the two samples *are* similar, $t(17\,890) = 0$, $p = 1$.

Table 5.5: Confusion matrices per test moment for a Rasch model made with *Dexter*.

(a) T_0

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.724**	0.201	0.042
HAVO	0.247	**0.536**	0.3
VWO	0.029	0.263	**0.658**

(b) T_1

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.785**	0.172	0.026
HAVO	0.203	**0.598**	0.334
VWO	0.012	0.229	**0.639**

(c) T_2

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.782**	0.191	0.028
HAVO	0.207	**0.606**	0.295
VWO	0.012	0.203	**0.677**

(d) T_3

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.761**	0.266	0.064
HAVO	0.224	**0.557**	0.238
VWO	0.015	0.177	**0.699**

We can look at how accurate these borders are by means of looking at the F_1-score, precision, and recall. This is visually represented for each class in Figure 5.6 and the corresponding confusion matrices are found in Table 5.5. Moreover, the non-proportional confusion matrices are found in Appendix D.1. We see that VMBO is the best performing class, followed closely by VWO. The worst performing class by far is HAVO which can be explained by the fact that this class gets cut-off at both sides of the curve (see Figure 5.5). Interestingly, the fit is not for the class labels at T_3 is not the based when using the ability at T_3 but at T_2, having an overall F_1-score of 0.69 while the ability at T_3 has an F_1-score of only 0.68 over all classes. Granted, differences in F_1-score between the test moments are slim, ranging from 0.69 (T_2) to 0.64 (T_0) and therefore may be explained by inaccuracy when determining the cut-off lines by visual inspection.

Finally, we note how the precision and recall are similar for school types at different test moments. For example, differences in precision and recall at T_0 are at most 0.08 (precision at recall for VMBO) but is much smaller for any other classes at any other

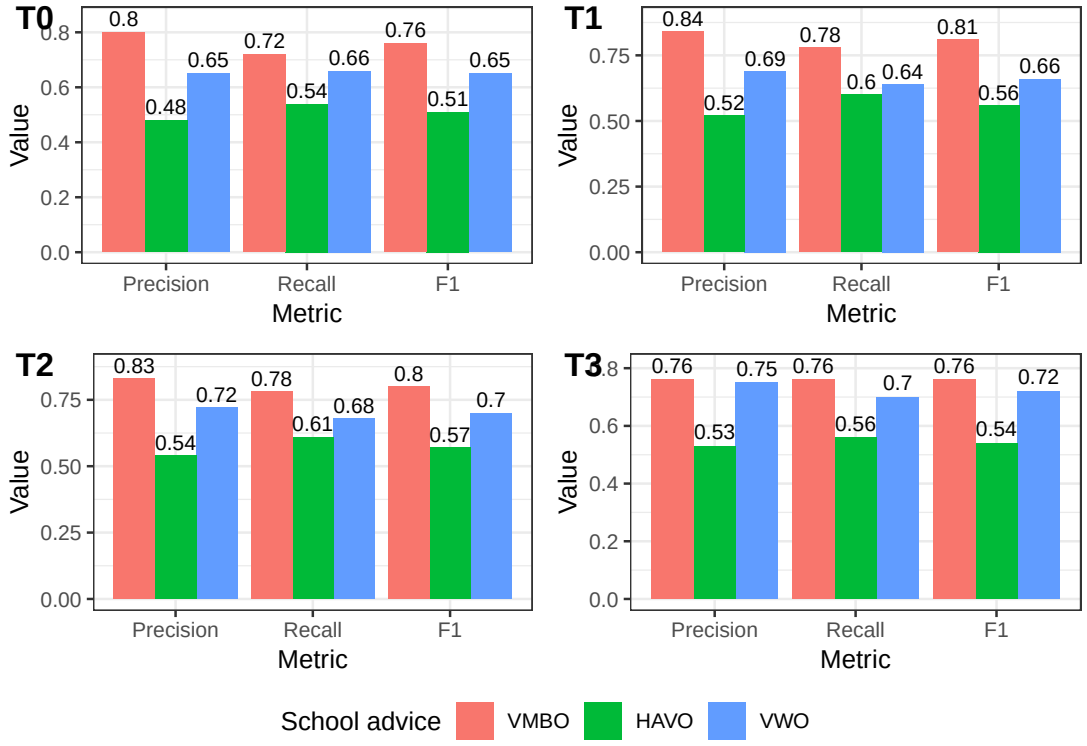

Figure 5.6: Three performance measures show how VMBO is the best performing class followed by VWO and HAVO. The latter is likely to perform worse because it is a middle class and is therefore cut off at both sides of the curve.

test moment. Thus, we can say that the models are balanced in the sense that they favour neither precision nor recall. Of course, that also raises the question of what is more important in this situation: *being sure that the students (for any school advice) are correctly predicted (precision)* or *being sure that the students that should have a specific school advice have been correctly captured*? In multi-class classification, this may be hard to interpret and can best be understood from the viewpoint of one specific class. For instance, we could maximise recall for the class VMBO by predicting VMBO for all students. As a result, precision will be low since many students that have been classified as VMBO will actually be HAVO or VWO. Then for the other classes, recall will automatically be 0 since there are no students left to be put in this class. The takeaway from this is that it makes sense that precision and recall is balanced among classes, since balancing one metric for one class will lower it for the others.

5.5 Predicting School Advice

In addition to drawing borders between the classes' density functions, we can also fit a multinomial log-linear regression (MLR) with the ability at T_0 as a predictor and school advice at T_3 as an outcome variable. We do so by using the function `multinom` from the *nnet* package (Venables & Ripley, 2002) in R. It should be noted that contrary to ML procedures, this model does not need to be trained first and then tested on a separate (unseen) part of the data set since the ability distribution has already been fixed by *Dexter* when the IRT Rasch model was applied to the data. The only difference which would result from following a validation-set approach is due

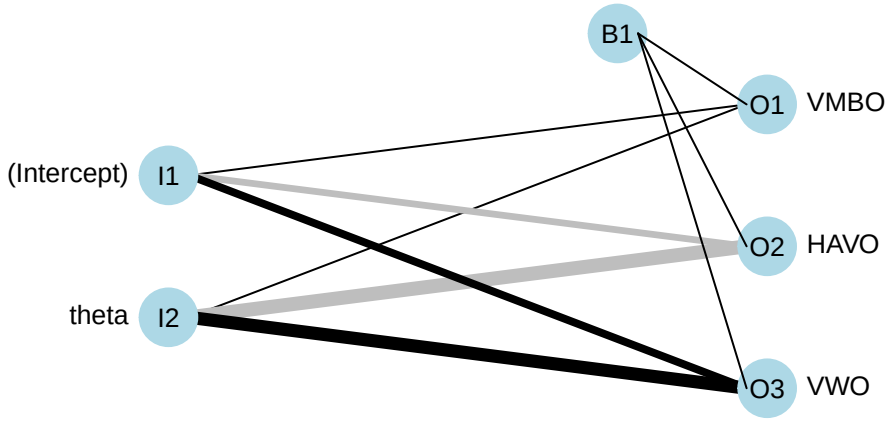

Figure 5.7: By displaying the multinomial log-linear model as a set of nodes and arcs, we see that this loosely resembles a neural network.

Table 5.6: Coefficients for the multinomial log-linear model describing predictions from ability at T_0 to school advice T_3.

Class	(Intercept)	theta
HAVO	-2.136206	2.627102
VWO	-5.408948	5.008078

to sampling and does therefore not improve the model. Hence, we can train on the full data set and evaluate the fitted values directly.

Just like the other functions from the *nnet* package, `multinom` is essentially a small neural network without a hidden layer. For our function, this idea is demonstrated in Figure 5.7. The model has two input nodes, i.e. (i) an intercept term and (ii) the theta (ability) itself, and four output nodes, i.e. one node for each class plus a bias term. The coefficients for these nodes are provided in Table 5.6.

We can visualise the distributions created by the MLR in a similar fashion as previously. As can be seen in Figure 5.8, we can draw cut-off borders by visual inspection although in contrast to earlier in this chapter this is not necessary since the model automatically calculates the class probability for every data instance (i.e. student).

We then compare Figure 5.8 to Figure 5.5 in terms of performance. Note that in this section we have only predicted T_3 from T_0 and not from any other test moments since we also do to this for ML in the next chapter. The reason for this is trivial, namely that predicting from the earliest to the latest test moment is the most difficult, although one might also argue that T_1 is a good candidate as a predictor since students have to make a choice at this test moment regarding which school type they want to pursue next year.

Various performance measures have been plotted along with their exact value in Figure 5.9. The proportional confusion matrix for this model is given in Table 5.7. When comparing these values to the one in Figure 5.6, we see that they are more or less the same for most metrics across most school types. Overall, the model has actually gone a bit *down* in performance, i.e. precision is now 0.65 (was 0.64), recall is now 0.65 (was 0.64), and F_1-score is now 0.65 (was 0.64). The only noticeable improvement seems to be for the class VMBO in terms of recall, which is now 0.8 (was 0.724). Indeed,

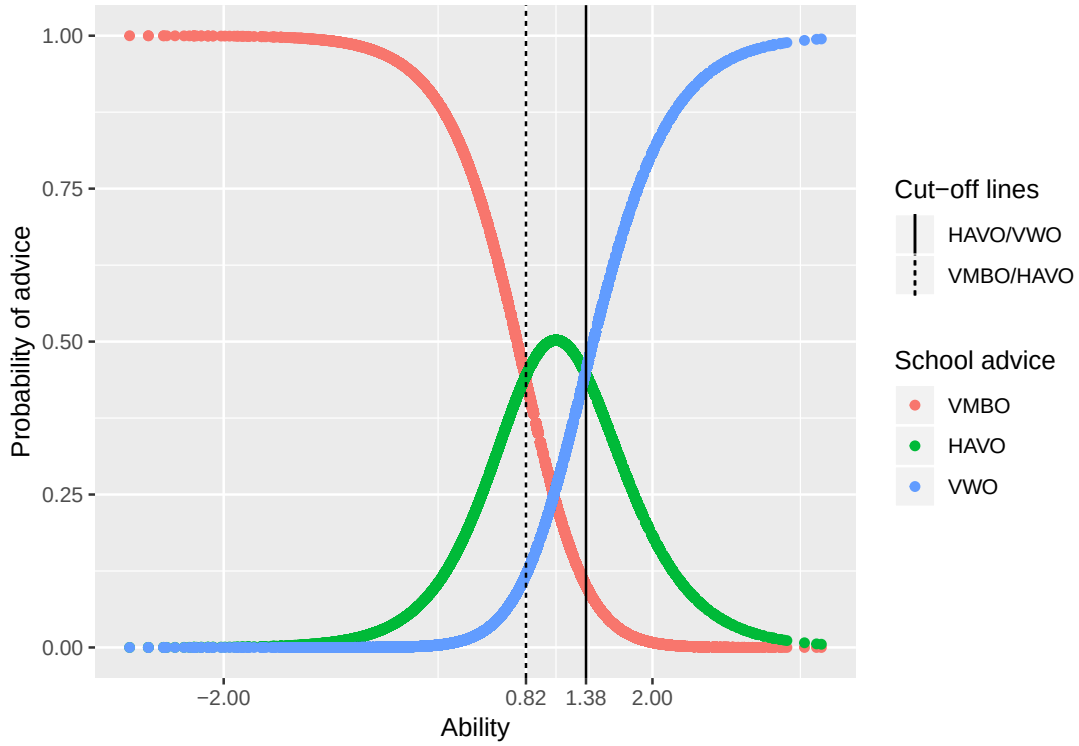

Figure 5.8: This plot shows the fitted values of the multinomial log-linear model. Just like before, HAVO seems the be the worst performing class.

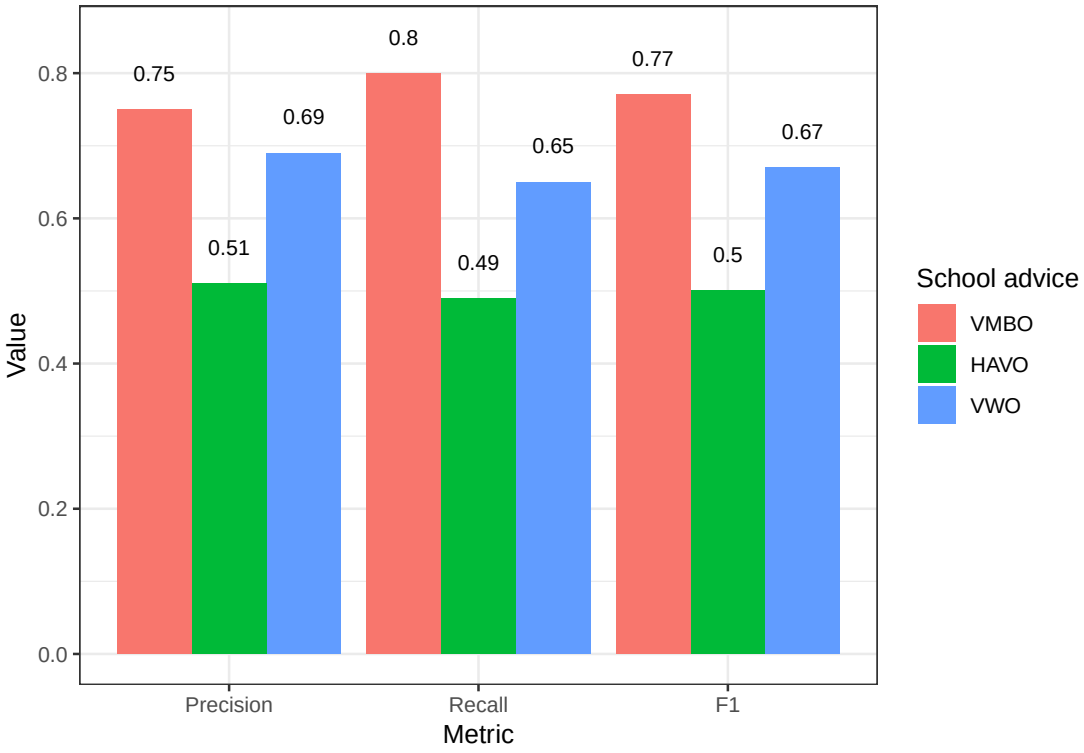

Figure 5.9: Precision, recall, and F_1-score for a multinomial log-linear regression using the ability at T_0 as a prediction for the school type at T_3.

Table 5.7: Confusion matrix for the three classes when applying a multinomial log-linear regression. Numbers in the table are given as a proportion of the reference counts.

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.795**	0.3	0.065
HAVO	0.181	**0.489**	0.282
VWO	0.023	0.211	**0.652**

Table 5.8: Sensitivity, specificity, and AUC for the multinomial log-linear model for classifying students from ability scores.

School advice	Sensitivity	Specificity	AUC
VMBO	0.781	0.891	0.744
HAVO	0.494	0.747	0.679
VWO	0.679	0.839	0.719
Overall	0.661	0.830	0.826

a McNemar's Chi-squared test with continuity correction (Agresti, 1990, p. 415) points out that predictions are significantly different, $\mathcal{X}^2(1, N = 17\,891) = 7, p = 0.008$.

Additionally, ROC analysis is also performed to give insight regarding the sensitivity/specificity trade-off which is visualised in Figure 5.10. The specific numbers per class are given in Table 5.8. As described previously, VMBO is the best performing class followed closely by VWO with HAVO lagging much behind. Overall, the area under the curve (AUC) is 0.83. Additionally, note how the specificity is much higher than the sensitivity, although this can be changed by adjusting the (probability) thresholds. These metrics will be used in Chapter 7 to draw a comparison between the performance of IRT and ML.

5.5.1 Evaluating Switching of School Type

At the beginning of this research, we stated how there is not only a need to accurately predict those who stay in the same school type throughout test moments but to provide special attention to predicting those who do switch. To this end, we provide a brief overview of how well IRT has performed in terms of predicting switchers. We do so based on the MLR model since this is overall a cleaner way to predict school advice. First of all, we perform a similar analysis as earlier in this section but this time only for students that actually switched. Secondly, we reshape the data so that our target variable changes from *school advice* to *switched*, simply indicating whether a student has made a switch or not. This is then evaluated by using measures that can handle imbalanced classes more appropriately.

Figure 5.11 visualises sensitivity, specificity, and AUC for the three school types but this time only for students who actually switched. The classes have high specificity but low sensitivity, indicating that not many students are classified as switching from this class but if it does predict so, the model is fairly certain this is actually the case. Interestingly, the AUC of HAVO is much higher than for the other class. One

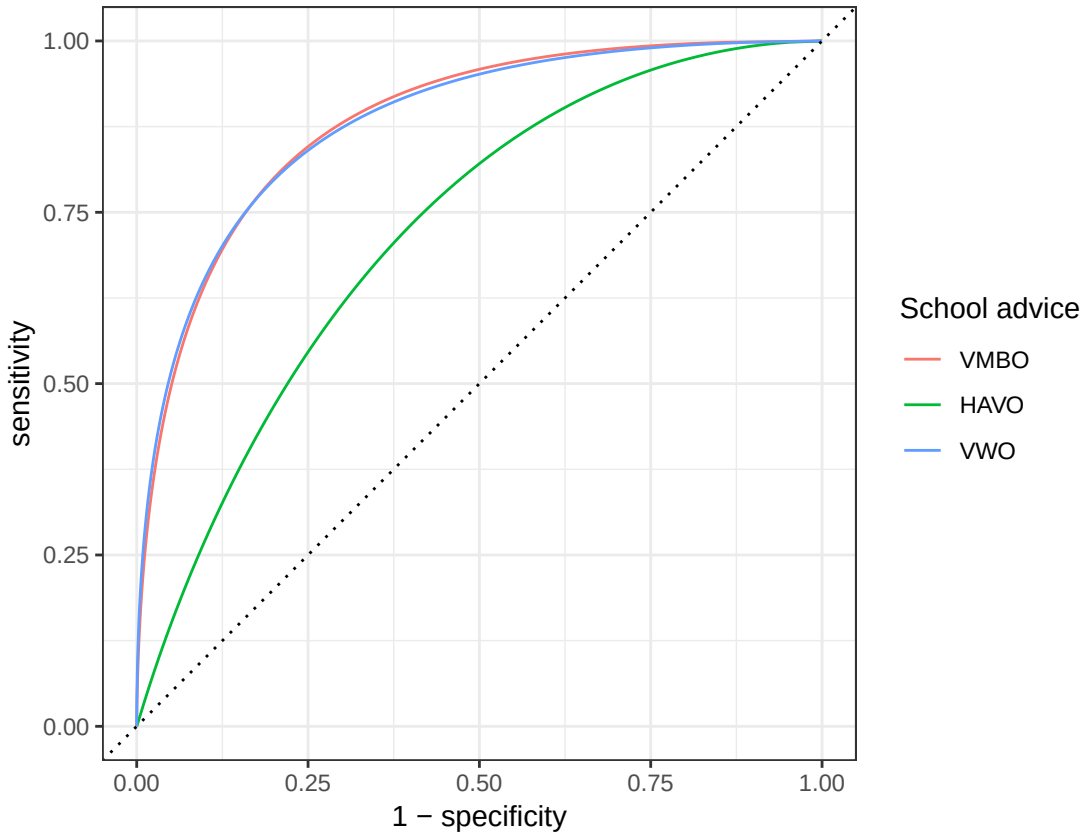

Figure 5.10: As an alternative to precision and recall, we can also draw a multi-class ROC curve to evaluate the sensitivity/specificity trade-off. As expected, HAVO is the worst performing class.

Table 5.9: Confusion matrix for students that actually switched.

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.503**	0.32	0.131
HAVO	0.4	**0.444**	0.428
VWO	0.097	0.236	**0.44**

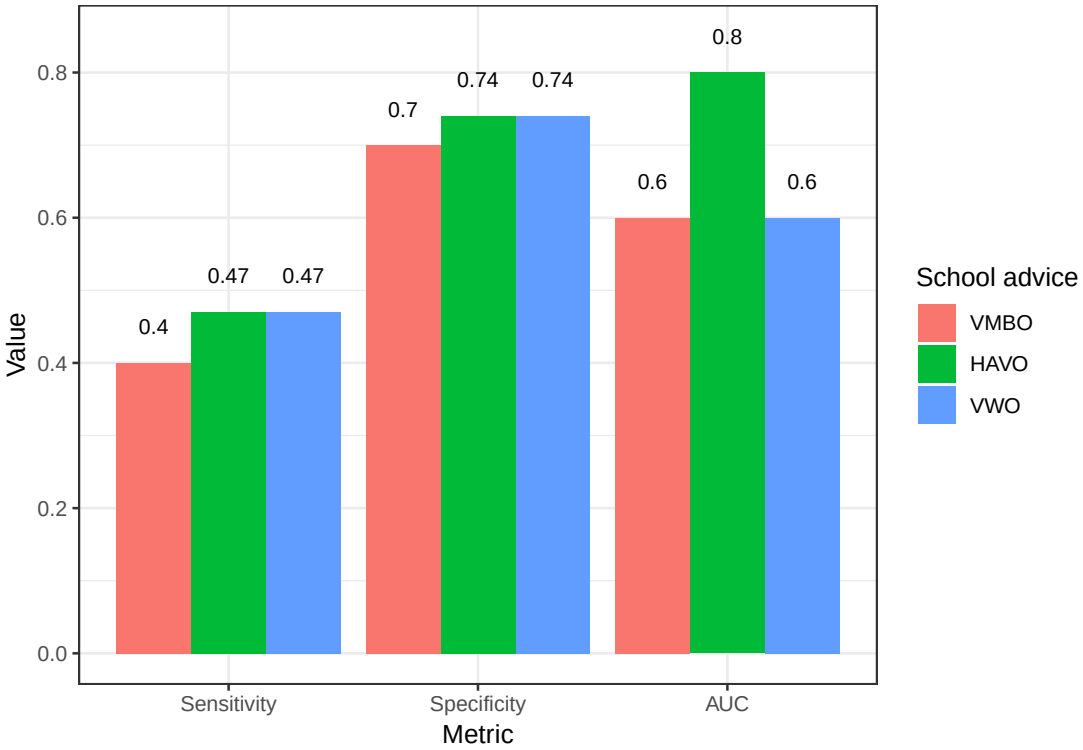

Figure 5.11: Performance metrics for students who switched school type between T_0 and T_3. Overall, the model performed worse for students who switched school type than for those who did not switch.

explanation may be the result of the 'squashing' that occurs in Figure 5.8. That is, only a few pupils are classified as HAVO (while there are much more) which culminates into predicting correctly many HAVO students who switched school type.

Overall, the model treats students who switch much worse than students who do not, compare the sensitivity (switchers 0.45; all students 0.661), specificity (switchers 0.73; all students 0.830), and AUC (switchers 0.66; all students 0.826). So for students who switch, the model performs much worse. Notwithstanding this bad performance, a binomial test points out the predictions are better than the no information rate, $p = 0.04$ (1-sided).

As an alternative to looking at the performance of looking at switchers in isolation, we can also look at a binary view (switch / no switch) and use a performance metric which is more robust to class imbalance.

To evaluate this, we only look at whether someone switched school type or not in comparison to test moment T_0. All sensible combinations (while keeping T_0 fixed

Table 5.10: Table of combinations between school advice at T_0, T_3, and predictions where school advice at T_0 is fixed at VMBO. Labels 0 and 1 indicate incorrect and correct respectively.

	T_0	T_3	Prediction	Truth label	Prediction label
TN	VMBO	VMBO	VMBO	0	0
FN	VMBO	HAVO	VMBO	1	0
FP	VMBO	VMBO	HAVO	0	1
TP	VMBO	HAVO	HAVO	1	1

Table 5.11: Disagreement between school advice at T_0, T_3, and predictions for T_3. Misclassification type indicates the severity of the prediction in relation to actual school advice at T_0 and T_3. In total, there are 231 underestimates, 122 overestimates, and 35 cases where the prediction was in between the school advice at T_0 and T_3.

School advice at T_0	School advice at T_3	Predicted label	# of cases	Misclassification type
VMBO	HAVO	VWO	61	Overestimate
VMBO	VWO	HAVO	15	Average
HAVO	VMBO	VWO	61	Overestimate
HAVO	VWO	VMBO	144	Underestimate
VWO	VMBO	HAVO	20	Average
VWO	HAVO	VMBO	87	Underestimate

at VMBO) are reflected in Table 5.10. The cases that are not shown, are for instance when school advice at T_0 is VMBO, at T_3 HAVO, and predicted is VWO. In that case, the student indeed switched school type and although this is also predicted, the prediction is for the wrong class. In order to make a fair comparison, we throw these cases out. In total, there are 388 students where the school advice at T_0, T_3, and the prediction were all different school types. A more in-depth analysis is given in Table 5.11.

Since there are only 2 965 students who switched school type between T_0 and T_3 (out of 17 503 students, leaving out the 388 students mentioned before), there is a vast class imbalance. From the work of Parker (2011), we know that traditional performance measures such as F_1 or AUC give a misleading view when there is a large class imbalance. When classes are very imbalanced, both the H measure (Hand, 2009) and Matthews correlation coefficient (MCC) (Matthews, 1975) have the least disagreement with other measures and are therefore most suitable (Boughorbel, Jarray, & El-Anbari, 2017; Parker, 2011). However, the H measure depends on knowing the posterior of the classes. While the posterior for school advice is provided by MLR, we do not have this for the classes switch / not switch. Thus, we resort to MCC which can be calculated from a contingency table. It is known as the ϕ coefficient and is related to the Chi-square statistic

$$|MCC| = \sqrt{\frac{\chi^2}{n}}$$

or based on a 2x2 contingency table

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Essentially, it is the correlation between the true and predicted class labels such that 1 represents total agreement, -1 total disagreement, and 0 no better than a random prediction. For the data as described previously, we found there to be a ϕ statistic of 0.16. Thus, the MLR model is only slightly better than a random guess when it comes to predicting whether a student switched or not.

5.6 Discussion and Conclusion

In this chapter, we described the process of predicting one's school advice at T_3 by using test results from T_0. To this end, we used a Rasch model to estimate the item and person parameters. We converted ability to school advice in two ways: (i) by drawing borders between the school advice density functions by visual inspection and (ii) by applying a MLR to compute the class probabilities for each student. We found the results to be similar for both models. For the log-linear regression model, the overall F_1-score is 0.65 and the overall AUC is 0.83. Interestingly, HAVO is by far the worst performing class which is to be explained by the fact that it is cut-off at both sides of the curve.

Since students who switch are of extra importance, we also looked at how MLR performs on those who actually switched school type. Overall, we found that the model treats switchers much worse than non-switchers, having an overall AUC of versus 0.826 for all students. Furthermore, an analysis was performed on the binary case of a switch (1) or no switch (0) where we employed the measure of Matthews Correlation Coefficient to battle heavy class imbalance. This indicated that the model is only slightly better at predicting whether someone switched school type than a random guess, $\phi = 0.16$.

A multidimensional approach was also attempted and is described in Appendix C. In brief, the (multidimensional) 2PL model gave misleading results due to population misspecification and the results were therefore unusable. Hence, we solely focussed on the unidimensional model in this chapter. Nevertheless, we noted that it is possible to overcome this problem by specifying the actual school type for each student. Further research will have to show whether a multidimensional 2PL model will improve results.

Chapter 6

Machine Learning

This chapter demonstrates how ML techniques can be used with data from educational progress tests. Specifically, we utilise ML to predict school advice from a set of dichotomous response data. This chapter is structured following two CRISP-DM phases: (i) data preparation and (ii) modelling. Hence, we first briefly outline data preprocessing and some challenges we faced before training the models in the next section. Subsequently, we evaluate the performance so that we can compare them with IRT as described in Chapter 5.

6.1 Data Preparation

In Chapter 5 we outlined the conversion of the original data to a long format. In Appendix C, we described how data can be transformed to a wide format using the `spread` function from the *tidyr* package (Wickham & Henry, 2019). However, this means data is then in a transactional-like format having four rows per student (i.e. for every test moment) instead of a flat-file format which is the de-facto standard in ML (i.e. one row per student). However, reducing four rows per student to one row is not an effortless task.

This is caused by, for example, the features `test_moment`, `test_id`, or `sex`. For `test_moment` it may be obvious that this cannot be brought back to one row, but for `test_id` we can create four columns `test_id_T0`, `test_id_T1`, `test_id_T2`, and `test_id_T3` which denote the test versions a student was administered. For `sex`, this should be the same value[1] as previous years so we can simply take the majority value — for example, if *'male'* was filled in three times and *'female'* one time, we assume the student to be a male. One final issue we had to overcome was the fact that due to a clerical error, there are 12 overlapping items between test *moments*, which we heretofore considered to be non-existent. This is problematic since every student now has one row with items in the columns. If a student was administered an item twice, we can either take the sum, mean, or take the value we happen to find first. All of these options are nonsensical so we remove these 12 items which will likely have a very minimal impact given that there are 5 183 items in total. When we throw these out, we can finally reduce the data set to one row per student.

[1] Since 2014 it is possible to change your sex at birth in the Netherlands. For the sake of simplicity, we assume this did not occur, especially since people need to be at least 16 years of age in order to apply for this change.

6.2 Modelling

In this section, we outline the usage of two very different types of models: (i) a relatively transparent model and (ii) a *'black-box'* model. By doing this, we showcase what the trade-off is between the complexity and performance of a model. We hypothesise that a more complex (i.e. black-box) model performs better, while a transparent model, despite given less accurate results, may be more valuable thanks to a higher degree of explainability. In this section, we only focus on the performance of the model while explainability is explored in Chapter 7. To keep a structured approach, we follow the various tasks laid out by CRISP-DM in the *modelling* phase.

6.2.1 Select Modelling Technique

As mentioned before, we select two models at either end of the complexity spectrum. Concretely, we look at a random forest (RF) and a single hidden layer (i.e. shallow) neural network (NN). RFs (Breiman, 2001) are fairly transparent since they are built on decision trees which are easy to interpret. In contrast, a NN (Venables & Ripley, 2002) is much more complex due to the hidden layer which essentially forms a model of other models.

While there are many other models available (see Chapter 3 for examples), we limit ourselves to only these two since our goal is not to try every model available, but rather to examine two well-known models that are very different from each other and thereby get an idea of what trade-off exists between complexity and performance.

6.2.2 Generate Test Design

One of the key challenges in training ML models is choosing a suitable test design. To make a fair comparison between IRT and ML, we opt to limit the ML models to the item responses. That is, we do not use 'metadata' about students such as their sex, birth date, or language spoken at home (see Chapter 4 for other attributes of students). Thus, the predictions for the school types are based on a matrix with only 0's (incorrect), 1's (correct), 9's (not filled in), and missing values (item was not administered). The latter option has to do with the fact that the tests follow a design where the difficulty of the items has been adapted to the ability of students and has been described in Section 4. In essence, the data used for prediction is a sparse matrix with values *not* missing at random (NMAR) (R. J. A. Little & Rubin, 2002).

Since most classifiers cannot deal with large amounts of missing value (especially if they are NMAR), we consequently train one classifier per test version. Because our goal is to predict school advice at T_3 from item responses at T_0, we have four test version to deal with: 515, 516, 517, and 518 (see Section 4 for a description). The procedure for generating our test design is as follows:

1. Select the items from T_0;
2. Split the data into a training (70%) and test (30%) set;
3. Split the training data into four parts corresponding to their test version;
4. Train the models on each of the parts (continued in Section 6.2.3).

We end up with two models for each test version, so eight models in total.

In this research, there is a special interest in those who switch school type. However, we have already established that there are many more people who do not switch

than those who do switch. A classifier is therefore prone to predicting *no one* (or very few) as having switched to a different school type. To remedy this, we follow a cost-sensitive learning approach (Elkan, 2001) and assign *case weights* to observations. According to this approach, a case weight determines the *misclassification cost* that occurs when an observation is incorrectly classified; the higher the weight, the more costly the misclassification. To gauge the impact of these weights, we train two variations of a model: one with case weights and one without case weights. The weights are chosen such that the group of switchers has the same sum of weights as the groups who did not switch.

For instance, if there are 20 people who switched and 100 people who did not switch between school types, we assign a weight of five to switchers and a weight of one to non-switchers so that the sum of the weights for both groups is 100. As a consequence, misclassifying one (entire) group is just as costly as misclassifying the other. However, this is an extreme case as it is in practice unlikely to find misclassifying five people just as bad as misclassifying one. As stated before, this is merely done to showcase the potential of predicting switching of school type. To test the effect of case weighing, we train every model with two variations: one having weights and one having no weights. This increases the total number of models we need to train from 8 to 16.

6.2.3 Build Model

In the previous subsections we argued the need to train 16 different models: two techniques, namely a RF and a NN; one model per test version (4 in total); and a variation of every model with weights and no weights. Training these models and tuning their hyperparameters in a loop can be done effortlessly with the `train` function from the *caret* package (Kuhn, 2008) in R. For the RF we use the *Ranger* package (Wright & Ziegler, 2017) (which is just a faster reimplementation in C++ of the original Fortran code by Breiman (2001)) and for the NN we use the *nnet* package (Venables & Ripley, 2002).

The *caret* package tries different combinations of hyperparameters, automatically performs cross-validation (CV), and then calculates a performance metric. To speed up this process, we perform *futility analysis* (Kuhn, 2014) to adaptively resample hyperparameters such that we discard combinations which are clearly sub-optimal. When tuning hyperparameters in the *caret* package, there are two types of performance measures which can be optimised for in a classification setting, that is (i) accuracy and (ii) Cohen's kappa. While the latter is more suitable for class imbalance (as is the case for school type), we instead use accuracy because, as it turns out, kappa cannot always be computed and thus results in a value of 0. For each hyperparameter, we try 20 different combinations (i.e. tune length).

While it is possible to increase certain settings which allow more hyperparameters to be tested, this is most likely not necessary and will only increase training time which is already high. For example, the maximum number of weights of the NN is restricted to 10 000 and increasing this to, for example, 20 000 allows for more complex fits but also *greatly* increases training time. In total, training all the models took 254 minutes.

6.2.4 Assess Model

Now that the models are trained, we can evaluate their performance on the test set. Similar to Chapter 5, we do this by looking at the metrics sensitivity, specificity, and

Table 6.1: Performance metrics for 12 models. For models computed for test version 515, some metrics could not be computed due to division by zero and is therefore not shown in this table.

Test	Model	Weighted	Sensitivity	Specificity	AUC
516	NN	Unweighted	0.740	0.870	0.691
516	NN	Weighted	0.742	0.871	0.719
516	RF	Unweighted	0.863	0.932	0.698
516	RF	Weighted	0.861	0.931	0.634
517	NN	Unweighted	0.692	0.846	0.529
517	NN	Weighted	0.567	0.784	0.638
517	RF	Unweighted	0.697	0.848	0.685
517	RF	Weighted	0.700	0.850	0.713
518	NN	Unweighted	0.790	0.895	0.715
518	NN	Weighted	0.772	0.886	0.659
518	RF	Unweighted	0.795	0.897	0.788
518	RF	Weighted	0.790	0.895	0.792

area under the curve (AUC). The results are presented in Table 6.1, but we discuss them in greater detail in this section. A confusion matrix for each table specifying the class counts between the predicted and reference model is given in Table D.2. Because we trained a separate classifier per test version, we will evaluate them per classifier as well. In the next subsection, we take a closer look at how the models treat students who switch — analogously to Section 6.2.5.

Test version 515. As described in Section 4, test version 515 roughly corresponds to VMBO-BB and VMBO-KB which is one of the vocational school subtypes. Consequently, it rarely occurs that people go 'upstream' multiple levels (see Appendix A for an explanation of the different subtypes). In fact, out of 3 085, only 6 people switched to HAVO. While case weighing has been applied to accommodate for class imbalance, this situation is so severe that the model has essentially become worthless. To understand why, consider that the performance of hyperparameters is tested by means of CV. If there are only 6 people who switched (of which 2 are in the test set), it stands to reason that in some iterations there is only one output class (i.e. VMBO). These iterations then result in an error (because there is only one response class) which means this particular value of the hyperparameter will not be considered. Thus, the model is ill-trained and if it was not, the results would hardly be generalisable.

In contrast to the other test version, results for test version 515 are *not* included in Table 6.1. We find there to be a specificity of 0.999 for all four models (i.e. NN and RF, one with weights and one without) while the sensitivity cannot be computed. Consequently, the AUC cannot be computed either. As stated before, however, these results are nonsensical due to having only a few people who switch school type.

Test version 516. In the previous paragraph, we found that only a few students switched to a different school type. This was due to the different subtypes in VMBO education. For test version 516, recall that this roughly corresponds to school advice of VMBO-GT. Thus, there are now fewer subtypes separating VMBO and HAVO.

Indeed, students who were administered test version 516 switched to a different school type 687 times while in 3 626 cases their school advice remained the same. Notwithstanding this increase in switchers, there is still a large imbalance in both switchers and school type. Case in point, 3 640 students followed the VMBO school type at T_3 versus 647 students pursuing HAVO and 26 students in VWO. These numbers do not exactly correspond with the previous number because a test version can also be distributed to students *not* in the corresponding school type.

We begin our performance evaluation by looking at the performance of the NN. Performance on the test set yields a sensitivity of 0.740 and a specificity 0.870 for the unweighted model. Furthermore, AUC is 0.691. One characteristic of performance that is true for all models in Table 6.1 is that the specificity is higher than the sensitivity. Of course, one could adjust the threshold t such that sensitivity and specificity are balanced. Another characteristic of these models we note is that there is a minimal difference in terms of performance between the weighted and unweighted model. In fact, the weighted NN performs a bit better in terms of sensitivity and specificity, 0.742 and 0.871 respectively. This discrepancy is likely caused by the random search algorithm used in *caret* — it tries random values of hyperparameters and then out of this subset selects the best ones available. It just might have happened that a different set of random hyperparameter values was tried which resulted in a slightly better performing value to be selected for the weighted model.

In addition to a NN, we also trained a RF on data for test version 516. For the unweighted model, we found a sensitivity of 0.863 and a specificity of 0.932. The overall AUC is 0.698. Interestingly, this model performs much better than the NN described previously in terms of sensitivity and specificity but not so much on AUC. This indicates that, at one point, a RF performs much better than a NN but on average a RF is only slightly better. Finally, we note that the sensitivity and specificity of the *weighted* model are similar to that of the *unweighted* model — 0.861 and 0.931 respectively. Again, this minor difference is likely due to random chance when selecting hyperparameters.

Test version 517. In the previous two test versions, we saw people who switched between school types were under-represented. While in test version 516 only 16% of students switched school types between T_0 and T_3, this is already 30% in this test version (517). Consequently, we expect the models to perform worse for this test version. Furthermore, in contrast to the previous test version, students can now go upstream (to VWO) or downstream (to VMBO), whereas previously they could only go upstream. Between T_0 and T_3, 4 175 students stayed at the HAVO school type, 756 went to VMBO, and 1 084 went to VWO.

As expected, both sensitivity and specificity are much lower than for the previous test versions. For the unweighted NN, we find a sensitivity of 0.692 and a specificity of 0.846. The overall AUC is 0.529. Because many more students who were administered test version 517 switched school type (because this is a middle class), we expect weighing to play a larger role. A weighted NN has a sensitivity of 0.567 and a specificity of 0.784. Although both the sensitivity and specificity are lower for a weighted than for an unweighted NN, the AUC is *higher*, i.e. 0.638 (weighted) versus 0.529 (unweighted). Thus, while an *unweighted* NN performs better for these thresholds, a *weighted* NN performs better *on average*.

The RF models perform worse for this test version as well. An unweighted RF model achieves a sensitivity of 0.697 and a specificity of 0.848. Nevertheless, the AUC has not decreased by much; while an unweighted RF in test version 516 had an average AUC of 0.698, the AUC in test version 517 was 0.685.

As stated before, we expected case weights to play a larger role in this test version because more people switched school type. In contrast to NNs, a weighted RF model performs slightly better than an unweighted RF — having a sensitivity of 0.700, a specificity of 0.850, and an average AUC of 0.713. However, it is unclear whether this is actually caused by case weights or if it is the effects of the random search pattern when tuning hyperparameters.

Test version 518. The final test version we discuss is test version 518 which roughly corresponds to VWO. Consequently, we expect most people to stay in VWO and some going downstream to HAVO or even VMBO. The number of switchers is approximately equal to test version 516, only for this test version the primary class is VWO. Between T_0 and T_3, 3 613 stayed at the VWO school type, 832 went to HAVO, and 33 went to VMBO.

We begin our performance evaluation by looking at the NN. For the unweighted model, we find a sensitivity of 0.790, a specificity of 0.895, and an average AUC of 0.715. Coincidentally, not only are the number of switchers approximately equal between test version 516 and 518, but the models also perform approximately the same. A weighted NN has a sensitivity of 0.772 and a specificity of 0.886 which means a weighted NN slightly underperforms in comparison to an unweighted NN. The average AUC is 0.659. In the next section, we take a closer look at how the models treat students who switch instead of all students as in this section.

A RF performs similar to a NN. Concretely, an unweighted RF trained with the *ranger* package achieves a sensitivity of 0.795, a specificity of 0.897, and an average AUC of 0.788. A weighted RF performs slightly worse in terms of sensitivity and specificity (0.790 and 0.895 respectively) but a bit better on average AUC (0.792).

6.2.5 Evaluating Switching of School Type

In this section, we evaluate the impact of the model on students who have *actually* switched school type. This section is laid out analogously to Section 5.5.1 with a few changes:

- In Section 5.5.1 there was only a log-linear model; in this section, we have two different models: a NN and a RF. We can evaluate the impact of both these models separately.
- Instead of training a single classifier over all test versions, we now have one classifier per test version. While this complicates the process of comparing the models, this was ultimately necessary to be able to train the models.
- In Section 5.5.1 we had only an unweighted model. In this section, we have both a weighted and unweighted model. In Section 6.2.4 we hypothesised that an unweighted model would do better overall; Likewise, for students who *actually* switched we assume that a model which assigns a higher weight to those students will outperform a model who does not give this group special attention.

Figure 6.1: This figure depicts the sensitivity, specificity, and AUC of 12 models (NN and RF; weighted and unweighted; one model per test version) for students who switched school types.

The sensitivity, specificity, and AUC of the 12 models for students who switched school type are visually displayed in Figure 6.1. While we will not discuss these results in as much detail as in Section 6.2.4, we note a number of key observations that characterise model performance of switchers.

The first observation is straightforward; that is students who switch are likely harder to predict that those who do not (and are more scarce) so it makes sense that the models perform (much) worse on this group. Averaging over all models, the sensitivity has decreased by 0.55, the specificity by 0.28, and the AUC by 0.028 in comparison to the unweighted models. The decrease in specificity and most notably sensitivity appears to be an enormous cutback but may be misleading given the fact that the AUC has only gone down slightly in performance. This may indicate that the threshold t along which we can move the ROC curve is chosen sub-optimally. In other words, there are thresholds which are favourable to students who switched, but perhaps less favourable to those who do not switch.

A second observation is that, in contrast to the performance metrics shown in Table 6.1, *weighted* models perform better than *unweighted* models in terms of AUC. This is an important observation as it confirms our intention that we want to weigh students who switch school types higher than students who do not switch. Thus, Figure 6.1 effectively shows that the poor performance for switchers can be mitigated at the cost of performance of non-switchers by employing a cost-sensitive learning approach. Of course, the question then becomes: *how much is the performance of non-switchers allowed to decrease in order for the performance of switchers to increase?* Comparing the AUC of the *weighted* models in Figure 6.1 to the AUC of the *unweighted* models in Table 6.1,

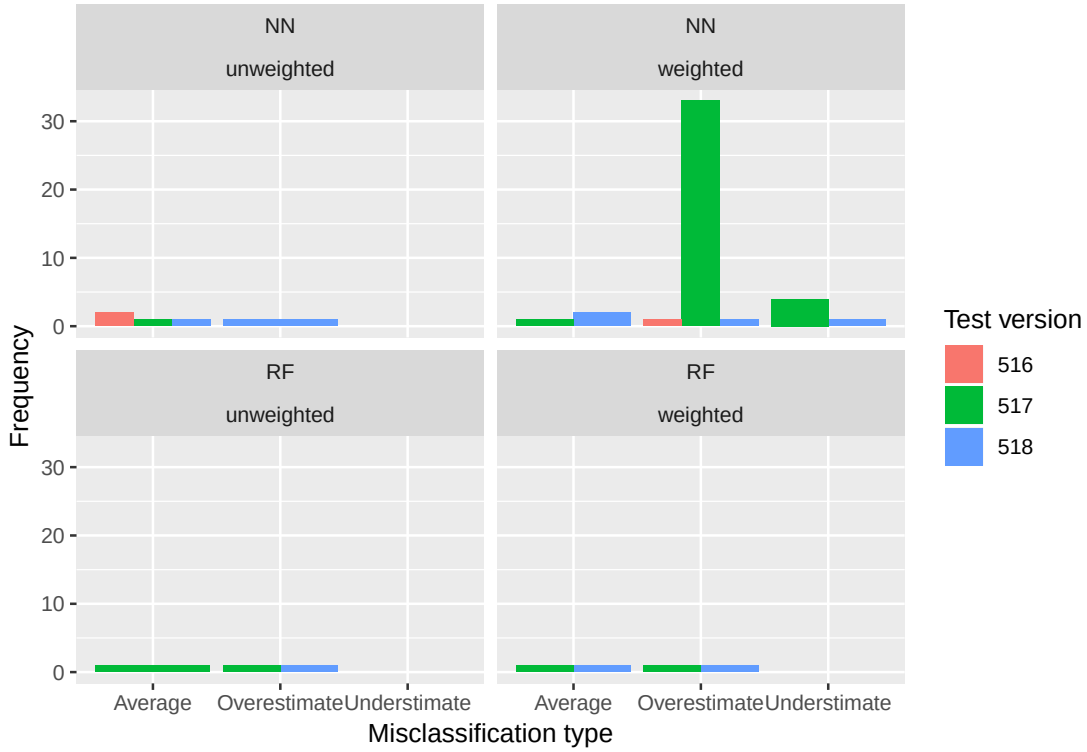

Figure 6.2: This figure shows the number of mismatches between school advice at T_0, T_3, and the predictions output by ML models. These results have previously been established in Table 5.10.

we see that former generally performs better, indicating that the weights of the model (for switchers) are set too high if the goal is to treat the two groups equally.

Finally, we note that — similar to the performance over all students — specificity is much higher than sensitivity. While it is unclear what exactly is causing this, one explanation may be that it is only favourable to classify that a student has switched when the model's confidence about this is high. Consequently, only few students who *actually* switched school type have been classified as such, but when this did occur the confidence of this event is relatively high. If we compare the accuracy of the models on switchers, we find that the accuracy is significantly *worse* than the no information rate, $p < .001$. In practice, however, this test is nonsensical since it implies that if we know that a student will switch, it is better to predict the majority class (HAVO).

To conclude this Section in a similar fashion as Section 5.5.1, we briefly look at how the models perform in terms of predicting *whether* a student has switched instead of which school type. Rules for transforming the data are given by Table 5.10. In total, there are 66 mismatches over the different models and test versions. A more detailed specification is shown in Figure 6.2. One interesting pattern is that a weighted NN in test version 517 (corresponding to HAVO) tends to *overestimate* students — that is it predicts they will switch from HAVO to VWO but they actually went from HAVO to VMBO. The other mismatches occur relatively infrequently (between 1–4).

So, without considering the mismatches between school advice at T_0, actual school type at T_3, and predictions by ML, we can look at Matthews correlation coefficient (MCC, denoted as ϕ) to discover the relationship between the actual school type at T_3

and the predictions by ML. This is done analogously to Section 5.5.1 and is further explained there. For an unweighted NN, we find there to be a strong correlation between predicted switches by ML and actual switch at T_3, $\phi = 0.7$. For the weighted variant, $\phi = 0.39$ which indicates a weak to moderate correlation between predicted and actual class labels. RF models perform better in predicting *whether* a student has switched; both an unweighted and weighted RF have a very strong correlation between predicted switching of school type and actual switches at T_3, $\phi = 0.88$ and $\phi = 0.81$ for an unweighted and weighted model respectively.

6.3 Discussion and Conclusion

In this chapter, we described how two types of ML models can be applied to predicting school advice from educational tests. Concretely, we have trained a RF and a NN on different test versions. This was necessary because of the sparse matrix resulting from the multiple test versions. Furthermore, cost-sensitivity learning (Elkan, 2001) was applied to assign extra weights to students who switched school type between T_0 and T_3. Thus, we trained 16 models in total.

We first established that results from test version 515 are inoperable since the model simply predicts that all students stay at the same school type. Next, we found that weighted models sometimes exceeded and sometimes fell behind unweighted models in terms of performance; this also depends on how many students switched. Another observation was that all models seemed to favour specificity over sensitivity, although it must be noted that this can be adjusted by moving the threshold along the ROC curve.

To evaluate how the ML models treat students who switch school type, we highlighted their performance by only looking at the performance of this group. In contrast to previous results, weighted models now outperformed unweighted models. In other words, when students are assigned a higher weight, ML tends to treat them better. One caveat that must be noted is that, if we know that a student has switched, it is better to predict the majority class (VMBO), $p < .001$. Finally, we looked at the performance of *whether* ML predicted a student has switched (instead of to which school type). Overall, we found that weighted NN on test version 517 systematically overestimates school type. Class label correlation ranged from $\phi = 0.39$ (weighted NN) to $\phi = 0.88$ (unweighted RF).

III

Evaluation

In the final part of the CRISP-DM cycle, we perform an evaluation of the models. In Chapter 7, we take a closer look at the explainability of the model and how this compares to their compared performance. Thus, we essentially seek to answer SQ2: *How do different techniques perform when predicting school advice?* and SQ3: *How do different techniques perform in terms of explainability?*.

Chapter 7

Comparing IRT and ML

The primary goal of this research is investigating how IRT and ML differ in terms of performance and explainability. To this end, we evaluate two key aspects of performance in this subsection: (i) predictive accuracy and (ii) computational feasibility after which we discuss differences in explainability by utilising the explainability framework laid out by Lipton (2016) and is further described by Robeer (2018) and extended by Waa, Robeer, Diggelen, Brinkhuis, and Neerincx (2018) in the form of *foil trees*.

7.1 Performance Evaluation

In this section, we compare the two methods in terms of predictive accuracy and computational feasibility as specified in research questions SQ2.1 and SQ2.2. Already in Chapter 5 and Chapter 6, we have described the performance of IRT and ML respectively in detail. In this section, we discuss the comparison of both methods in more detail. Moreover, we compare the two models in two different ways: (i) in terms of predictive accuracy and (ii) in terms of sensitivity, specificity, and AUC. This is because different performance metrics may come to different conclusions.

7.1.1 Predictive Accuracy

First, we look at the predictive accuracy over all students. To compare IRT and ML in terms of performance, we have computed a McNemar's chi-squared test (Agresti, 1990, p. 415) to compare the *accuracy* of IRT[1] and the four ML models. This is achieved by aggregating the class labels of each model over all test versions (i.e. pretending there is only one test version) and then tabulating a confusion matrix of the two methods from which the diagonals can be compared against each other. This chi-squared test is used since it can be applied to a 2x2 contingency matrix where the *McNemar test statistic* is defined as $\mathcal{X}^2 = \frac{(b-c)^2}{b+c}$ (Agresti, 1990, p. 415). Note that this is only performed on the test set of ML since using all data would overfit the ML models. Results are presented in Table 7.1. The tests point out that all ML models are *significantly different* from IRT, $p < .001$. However, this does not imply ML models are performing *better* than IRT, but judging from the higher sensitivity, specificity, and occasional higher AUC it is easy to imagine that ML techniques indeed perform

[1] Although we used MLR to make predictions from the IRT procedure, we use the term IRT to refer to the general process that involves both IRT and MLR.

Table 7.1: McNemar's chi-squared test for four ML models. The labels from the different test versions have been aggregated. The columns *Model 1 and 2 acc* indicate the number of correct labels alongside the diagonal (i.e. the true positives).

Model 1	Model 2	Model 1 acc	Model 2 acc	$\mathcal{X}^2$	df	p-value
Weighted RF	IRT	0.813	0.657	316.49	1	<.001
Weighted NN	IRT	0.735	0.657	74.73	1	<.001
Unweighted RF	IRT	0.814	0.657	315.76	1	<.001
Unweighted NN	IRT	0.781	0.657	194.38	1	<.001
Weighted RF	Weighted NN	0.813	0.735	180.57	1	<.001
Weighted RF	Unweighted RF	0.813	0.814	0.07	1	.79
Weighted RF	Unweighted NN	0.813	0.781	72.25	1	<.001
Weighted NN	Unweighted RF	0.735	0.814	174.31	1	<.001
Weighted NN	Unweighted NN	0.735	0.781	58.27	1	<.001
Unweighted RF	Unweighted NN	0.814	0.781	79.81	1	<.001

better than IRT. Additionally, the higher accuracy of the ML models also indicates they indeed perform better than IRT.

Furthermore, we have compared every ML model with every other ML model in a similar fashion. One way to appreciate these models is to rank them by their accuracy:

1. Unweighted RF (accuracy of 0.81)
2. Weighted RF (accuracy of 0.81)
3. Unweighted NN (accuracy of 0.78)
4. Weighted NN (accuracy of 0.73)
5. IRT (accuracy of 0.66)

From Table 7.1, we see that the predictions of a *weighted* RF and an *unweighted* RF are similar, $\mathcal{X}^2(1, N = 5\,364) = 0.07, p = 0.79$. This pattern has previously been observed — albeit without a significance value — in Table 6.1 where the performance metrics of both models were consistently similar. Other inferences we made from this table carry over to the aforementioned ranking. For example, from the performance metrics, we noticed how a weighted NN usually performed the worst; hence, this model is in the last place.

Comparing Sensitivity, Specificity, and AUC

In addition to looking at the predictive accuracy of the models, we can compare the other performance measures of IRT given in Table 5.8 against the performance measures of ML given in Table 6.1. One use of doing this is that it serves as a validation of the previously described results. In essence, the metrics describe hereafter are *robustness measures*. Note that the performance measures of IRT are computed using all data, whereas those for ML are only on the test set. Furthermore, this comparison is convoluted because classifiers are trained per test version and can therefore not easily be compared to the IRT. In other words, the aggregated class (VMBO, HAVO, VWO) do not directly correspond to a single test version so these cannot simply be compared. Additionally, test versions can also be administered to students *not* corresponding to that school type. We can, however, describe the general patterns we have seen between the two approaches. We do so by following a mapping between test versions and school type and hence we describe these results per test version.

Table 7.2: McNemar's chi-squared test for all models of the group of students that switched.

Model 1	Model 2	Model 1 acc	Model 2 acc	$\mathcal{X}^2$	df	p-value
Weighted RF	IRT	0.147	0.693	464.83	1	<.001
Weighted NN	IRT	0.354	0.693	202.21	1	<.001
Unweighted RF	IRT	0.091	0.693	542.82	1	<.001
Unweighted NN	IRT	0.167	0.693	425.53	1	<.001
Weighted RF	Weighted NN	0.147	0.354	154.13	1	<.001
Weighted RF	Unweighted RF	0.147	0.091	49.59	1	<.001
Weighted RF	Unweighted NN	0.147	0.167	3.33	1	.07
Weighted NN	Unweighted RF	0.354	0.091	235.37	1	<.001
Weighted NN	Unweighted NN	0.354	0.167	125.39	1	<.001
Unweighted RF	Unweighted NN	0.091	0.167	51.61	1	<.001

Starting with the class VMBO, we can roughly map this to test version 515 and 516 (see Section 4). As stated before, however, test version 515 has been rendered inoperable due to only very few people switching school type and thus being left with only one class. For this reason, we leave test version 515 out of the equation and continue to compare VMBO to test version 516. In terms of sensitivity and specificity, a (un)weighted NN slightly underperforms and a (un)weighted RF slightly outperforms IRT. When it comes to AUC, however, IRT is the better model. This may be relevant if one decides sensitivity should be more favoured instead of specificity (or vice versa), but at the current threshold, it means RF performs slightly better and a NN performs slightly worse than IRT.

The HAVO class is interesting because it features many more switchers than other classes. This is precisely where ML may have an edge; ML models trained on test version 517 all outperform IRT in terms of sensitivity and specificity. The differences are also much greater than in the previous test version. When it comes to AUC, NNs perform much worse while RFs perform slightly better. Thus, we can say that IRT is, on average, a slightly better model than NNs in the HAVO class, but NNs still perform better at the current threshold. RFs outperform IRT on every metric.

Finally, we look at the VWO class which we can roughly map to test version 518. Again, in terms of sensitivity and specificity, ML outperforms IRT. NNs are slightly behind IRT when it comes to AUC, a weighted NN more so than an unweighted NN. On the other hand, RF also outperforms IRT in terms of AUC; consequently, we can say that a RF is a better model if a student's school advice at T_0 is VWO or their administered test version is 518.

Switching of School Type. Similar to Chapter 5 and Chapter 6, we direct some of our attention towards those who switch school type. Hence, it is imperative we also evaluate which model performs best in terms of predictive accuracy on the group of students who switched school types. Similar to before, we have also computed McNemar's chi-squared tests (Agresti, 1990, p. 415) to compare the predictive accuracy of IRT and various ML models. Results are displayed in Table 7.2. Looking at the comparisons of each ML model with respect to IRT, we see that all are significantly different. Moreover, the predictive accuracy of IRT is much higher than for any ML model. Being provided with the results described above and the much higher

accuracy, we can conclude IRT performs better at predicting school advice of students who switch.

We can also draw some conclusions about ML models. For instance, we can create a ranking based on their predictive accuracy:

1. IRT (accuracy of 0.69)
2. Weighted NN (accuracy of 0.35)
3. Unweighted NN (accuracy of 0.17)
4. Weighted RF (accuracy of 0.15)
5. Unweighted RF (accuracy of 0.091)

This ordering is the complete opposite of the one we found when looking at all students. Furthermore, from Table 7.2 we see that an unweighted NN and a weighted RF perform similar, $\mathcal{X}^2(1, N = 987) = 3.3, p = 0.07$. This reversed ranking can have a number of causes. First of all, weighted models now perform better than unweighted models. This is a pattern we noted earlier in Section 6.2.5 where we hypothesised and found that weighted models perform better as more people switch school type. A second reason may have to do with the complexity of the models. While previously we found that a RF performed better than a NN, it is the other way around when we only focus on those who switch school type. Because a NN is generally more flexible than a RF (i.e. a RF is more *robust*), NNs can more easily adapt itself to 'unlikely' data patterns such as students who switch school type and therefore deviate from the majority of students.

Comparing Sensitivity, Specificity, and AUC

Like Section 6.2.5, we also compare the sensitivity, specificity, and AUC for the group of students that switched school type. Thus, we compare Figure 5.11 (IRT) to Figure 6.1. Again, the mapping between school types and test version is unclear, but we assume the structure 515/516–VMBO, 517–HAVO, and 518–VWO to provide a reasonable comparison.

From Figure 5.11 we can see that for the VMBO school type the sensitivity is 0.4, the specificity 0.7, and the AUC 0.6. Then, from Figure 6.1 we learn that the sensitivity and specificity of every ML model trained on test version 516 is lower than for IRT (again, we leave test version 515 out of the equation). That said, the AUC for every ML model is higher than for IRT which indicates that ML models are better on average, while at the current threshold IRT performs better.

For the HAVO class, IRT achieved a sensitivity of 0.47, a specificity of 0.74, and an AUC of 0.8. None of the ML models performed better on any measure. Hence, we can conclude IRT is clearly the better model for the HAVO school type.

Finally, we compare the performance of the VWO school type. IRT has a sensitivity of 0.47, a specificity of 0.7, and an AUC of 0.6. These results are comparable to that of the VMBO school type; the same is true for ML models. All ML models performed worse in terms of sensitivity and specificity, but better in terms of AUC.

Binary case. We concluded Chapter 5 and Chapter 6 by looking at they performed in terms of *whether* a student switched instead of which school type (i.e. converting the problem to a binary case). When comparing the results of IRT provided in Table 5.11 against ML in Figure 6.2, we see there are dramatically fewer mismatches in ML than

Table 7.3: Overall training time in minutes for calibrating the Rasch model (IRT) and training each ML model at T_0.

Model	Weighted	Training time
NN	Yes	115.67
NN	No	88.51
RF	No	25.10
RF	Yes	24.54
IRT	No	4.51

IRT. This becomes a bit better provided that Table 5.11 considers all 17 891 students whereas Figure 6.2 only covers the test set of ML, i.e. 5 364 students (of whom 970 switched). In terms of performance, we can compare Matthews correlation coefficient (MCC) of IRT and ML to make an ordering of the models ranging from better to worse:

1. Unweighted RF ($\phi = 0.88$)
2. Weighted RF ($\phi = 0.81$)
3. Unweighted NN ($\phi = 0.7$)
4. Unweighted NN ($\phi = 0.39$)
5. IRT ($\phi = 0.16$)

In conclusion, all ML models outperform IRT in predicting *whether* a student has switched in terms of MCC.

7.1.2 Computational Feasibility

At the beginning of this research (in Section 1.2), we hypothesised that computational feasibility would be an issue when constructing (computationally) expensive ML models. This time is of course depending on the machine used. For both IRT and ML, we used an R server running Ubuntu 14.04.6 LTS with 16GB RAM and an Intel(R) Xeon(R) E5620 @2.40GHz CPU.

Already in Section 6.2.3, we have commented on the total time to train the models but in this section, we dissect this even further. Overall, we found that the total training time for all ML models was 254 minutes. To get an idea of the differences between IRT and ML we can look at the training time for each model. The number of minutes needed to train (or calibrate in the case of IRT) is provided in Table 7.3. From this table, it becomes clear that a weighted NN requires the most time to train (almost 2 hours) while calibrating a Rasch model takes only four and a half minutes. Thus, all ML models require much more time to be trained than an IRT (i.e. Rasch) model.

We can further dive into the training time to gain a deeper understanding of the underlying patterns. A subdivision is visually represented for each model per test version in Figure 7.1. IRT is not included since this is trained over all test versions. There are a couple of key observations that stand out:

- A NN can take twice as much time to train than a RF. This is likely due to the fact that NNs are much more complex than RFs or that a NN *needed* to be more complex in order to achieve the same performance as a RF.

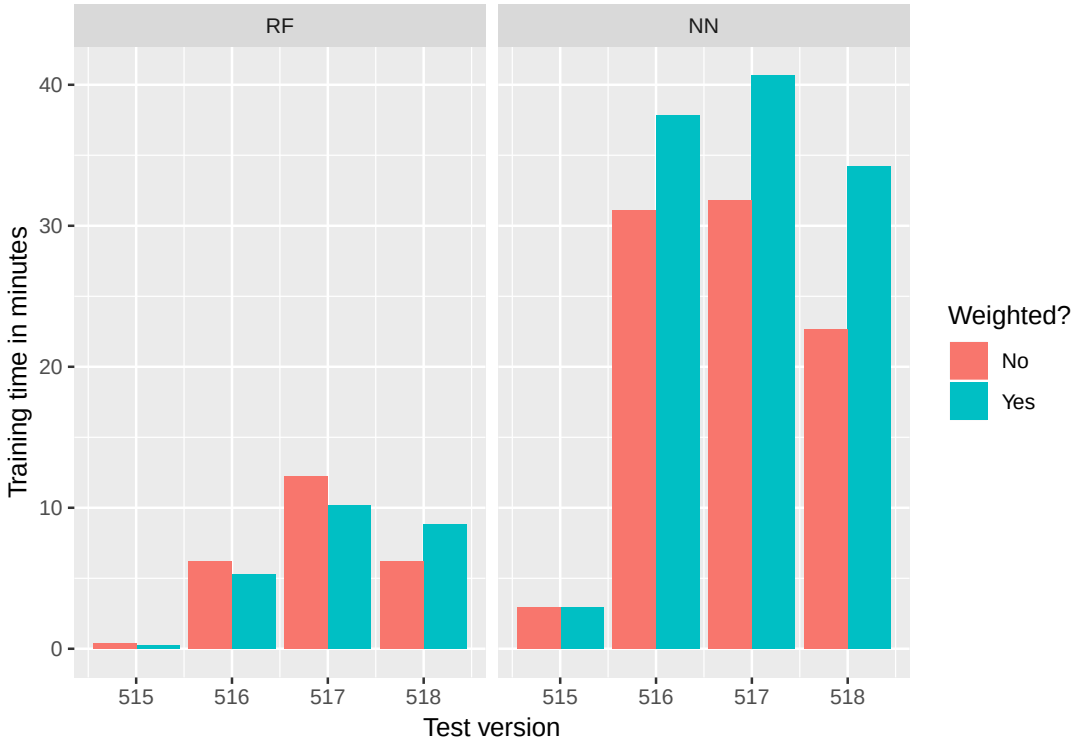

Figure 7.1: Training time in minutes for the ML models.

- Training models for test version 515 is much faster than for any other test version. A simple explanation is the one we noted in Section 6.2.4, namely that only 6 people switched school type (of which none was predicted correctly). Hence, we end up with a straightforward model in which all students stay in the same school type.
- Training a model with case weights requires more training time than a model without case weights. This is especially true for the NN but can also be observed in test version 518 for the RF. Apparently, following a cost-sensitivity learning approach requires more computational power because it (in this research) requires the model to be more complex.

In addition to these inferences, there are a couple of other intricacies worth pointing out. For example, note that for IRT we have only displayed the time needed to calibrate the Rasch model but *not* the time needed to determine the expected a posteriori (EAP) estimates for the abilities. This is significant because calculating the EAP values take much more time than making predictions for ML models (using the `predict` function in R). Compare 25 seconds for IRT to a tenth of a second for ML. This is closely related to some effects that occur as more students get added to the database.

So far we have seen that training a ML model costs much more time than calibrating a Rasch model. As more students get added (say twice as many), ML models will require much more time to be trained because new patterns occur that may influence the complexity of the model (likely make it more complex). In contrast, in IRT (using conditional maximum likelihood) the time to calibrate stops increasing once the probability of all response patterns have been calculated, so adding many more students will not increase calibration time by much. Conversely, more students will

not cause any problems for ML when *predicting* their values school advice but this might cause IRT to take longer. That said, the difference in training/calibration time between IRT and ML is much greater than the difference in their prediction time. One solution for ML to this problem is to take a reasonably large sample to train the model until the desired performance is reached, and then use the model from there on only to make predictions. For IRT, this problem is much less severe.

Finally, it should be noted that the training time of ML is greatly influenced by the settings chosen in Section 6.2.3. In other words, by allowing more or fewer hyperparameters to be tested by *caret*, the overall training time can be adjusted. This may possibly influence the performance of the model.

In conclusion, we can say that calibrating an IRT model is much faster than either a RF or a NN. To put these results into perspective, training even the most expensive model (weighted NN, 116 minutes) does still not take so much time since this can easily be run over night. Even when adding many more students, ML is still unlikely to run into any problems. Otherwise, a more powerful computer, fewer hyperparameters, or a sample is required to remedy this problem. Thus, while ML models take much longer to be trained than IRT models, all of them remain computationally *feasible*.

7.2 Explainability Evaluation

At the beginning of this research, we stated how there is a need to have *explainable* models. This necessity has been captured in SQ3. While in the previous section we have looked at the performance of the models, the *interpretability* has been left undiscussed. In other words, if a student was to ask why they were given certain school advice, to what extent can we explain why this advice was given?

One of the properties of interpretable models is that a model is transparent. In a sense, a *transparent* model is the antithesis of a *black-box* model. Lipton (2016) shows how transparency can be subdivided into three distinct viewpoints: (i) simulatability, (ii) decomposability, and (iii) algorithmic transparency. We visit each of these three notions, explain what they entail, and to what extent this applies to IRT, RFs, and NNs.

7.2.1 Simulatability

One way to evaluate whether a model is transparent is to see if a person can grasp the workings of a model in its entirety. That is, *simulatability* (Lipton, 2016) refers to what extent a model can be fully understood by a human in a *reasonable* amount of time. This may include time needed to perform the calculations of the model by hand, but may also refer to making inferences based on the model — for example in a decision tree following a data point from root to leaf. However, simulatability is not necessarily intrinsic to a model. Consider a linear regression with three input features; this can easily be calculated by hand. If we were to increase the number of features to 1000, this suddenly becomes infeasible. On the other hand, a shallow NN with two input nodes and two hidden nodes can actually be simulated as well.

An IRT model (Rasch in this research) with only a few items can be easily calculated by hand. However, if the number of items increases to 290–360 (as is the case in this research), this process suddenly becomes cumbersome. This is much different for a RF, although, admittedly, this depends on the definition of what simulatability

entails. As Lipton (2016) notes, this can be either the time needed to compute the model by hand *or* the time needed to make inferences. While calculating a RF for even a small number of features is time-consuming, walking through a RF from root to leaf tree-by-tree is viable. The model can then be understood as a series of textual explanations after which the final verdict results from a majority vote. Finally, it should be evident that computing a single hidden layer NN with 290–360 input features is rather infeasible.

However, simulatability does not necessarily refer to the time it takes to *compute* a model, but to the time it takes to *understand* it on a conceptual level. While both RFs and certainly NNs are difficult to conceptually grasp for non-expert, this is not the case for IRT. In this model, the difficulty parameter is (as the name implies) the difficulty in relation to the other items on the test and is therefore very easy to understand. Moreover, the ability parameter is less intuitive but can also be described by looking at the sum score of the scored item responses since this is a *sufficient statistic*. In other words, by looking at the number of endorsed items, a student can easily understand how this affects their ability which makes IRT the best explainable model in terms of simulatability.

7.2.2 Decomposability

The concept of *decomposability* refers to the extent that each input, parameter, and calculation can be intuitively explained. This is, of course, much easier if the inputs, parameters, and calculations can be related to the output of the model and its effect can be described. For this reason, models with (inherent) feature selection score poorly on this transparency property. For ML models, we evaluate their decomposability based on their hyperparameters and their feature importance (FI). Although it is not even possible to do this for IRT (since it does not have hyperparameter or FI), in a way we have already provided some insight into the model by describing the distribution of the difficulty parameters and how they relate to ability in Chapter 5.

These distributions provide valuable insight in terms of decomposability. In the Rasch model, there is one parameter that is to be estimated (difficulty) and one output value (ability). The difficulty is easy to interpret and understand since it has real-world value in that it can be linked to items itself. The ability is either useful when comparing it to peers or when looking at sum score. The decomposability is even greater in a 2PL model since this also estimates a *discrimination parameter* which can be interpreted as the importance of an item.

Hyperparameters. One way of interpreting a model through decomposability is to look at its (hyper)parameters. The hyperparameters of each RF and NN model are given in Table 7.4. Starting with the NNs, we see that the models for test version 515 appear to be identical. Having only two units in the hidden layer and a very low weight decay, they only predict a single class (VMBO). In a NN, the *size* of the model refers to the number of nodes that are in the hidden layer (see Section 3.2.6 for a more detailed explanation) while the weight decay functions as an early stopping mechanism for gradient descent. In fact, the weight decay is similar to ℓ_2 penalisation as used in, for example, the ridge or lasso regression. Other than this, there are two other observations that stand out for the hyperparameters of a NN:

Table 7.4: Hyperparameters selected by *caret* based on adaptive cross-validation.

(a) Hyperparameters for a neural network with a single hidden layer. The size refers to the number of nodes in the hidden layer. The decay function is penalisation of the model.

Size	Decay	Test version	Weighted?
2	0.00006	515	yes
2	0.00006	515	no
20	0.00337	516	yes
7	0.00026	516	no
4	0.00144	517	yes
1	0.06917	517	no
20	1.15205	518	yes
16	8.08174	518	no

(b) Hyperparameters for a random forest model. The number of trees is the number of trees used in aggregating the model. The split rule is the algorithm used to determine on which value to split. The minimum node size is the minimum number of observations used to make a leaf node.

Number of trees	Split rule	Min node size	Test version	Weighted?
180	Extra trees	1	515	yes
169	Extra trees	2	515	no
27	Gini	9	516	yes
269	Extra trees	18	516	no
38	Gini	20	517	yes
179	Extra trees	3	517	no
237	Extra trees	19	518	yes
245	Gini	5	518	no

- The number of nodes in the hidden layers is higher in a weighted model than in an unweighted model. From this observation, it appears that weighed models are more complex than their unweighted counterparts.
- The weight decay of the unweighted models is higher than the weighted models. Recall that the weight decay parameter is essentially a form of regularisation for the gradient descent optimisation algorithm to prevent it from overshooting its minimum value. Thus, models with a *higher* weight decay are *less* complex because they are more restricted by the higher penalisation term. Thus, not only are unweighted models less complex because they have fewer nodes in the hidden layer, but they are also more restricted. Consequently, this may also imply that the models with a higher weight decay generalise better to other, previously unseen data (Krogh & Hertz, 1992).

We also note some observations for the hyperparameters of the RF models (see Table 7.4). In a RF model, the number of trees is the number of trees that leads to a majority vote. This has previously been laid out in Section 3.2.4. The split rule is the cost function used to determine the optimal split. This can either be Gini (see Section 3.2.4) or extremely randomised trees (extra trees) (Geurts, Ernst, & Wehenkel, 2006), which aims to further randomise random forests by also selecting random split points from random variables. By doing this, the time-consuming Gini index needs no longer be calculated while performance decrease is often kept at a minimum. Finally, the 'min node size' is the interaction depth of the model — i.e. the minimum number of observations needed to perform a split; a higher 'min node size' leads to a *less* complex tree.

For the RF models trained for test version 515, we see that they consist of 180 and 169 trees. Furthermore, the trees are grown deep having a minimum node size of 1 and 2. The trees are randomised even further by applying the extra trees split rule. These results are unexpected given that they predict only one class (VMBO). A possible explanation may be given by the fact that the CV accuracy for all attempted hyperparameters on all folds was 1. For the other hyperparameters, there seems to be no obvious pattern as to why they were selected. While there is no clear explanation for this, one hint may be given by the fact that there is only little variance in the accuracy when cross-validating different hyperparameters set-ups. For instance, the difference between the best and the worst performing selection (i.e. the range) of hyperparameters for an unweighted RF forest model is only as little as 0.0033; the average range over all RF models is only 0.0052. Thus, it hardly seems to matter what value of the hyperparameters is selected since all attempted combinations perform about the same.

Feature Importance. A second way of looking at the decomposability of a model is by looking at FI. We define this concept as the relative importance of the most important feature in comparison to all the other features. Concretely, the most important feature is defined as having a FI of 100(%) while the others have a score relative to this number. In a RF, the feature importance is computed as the total decrease in node impurities from splitting on the variable, averaged over all trees. The FI in NNs is computed using Garson weights (Garson, 1991). Because each test version has 290–360 input features (i.e. items), we have increased readability by summing over the subjects the features belong to and is displayed in Figure 7.2. From this figure, it is easy to see that Dutch language skills are viewed as most important

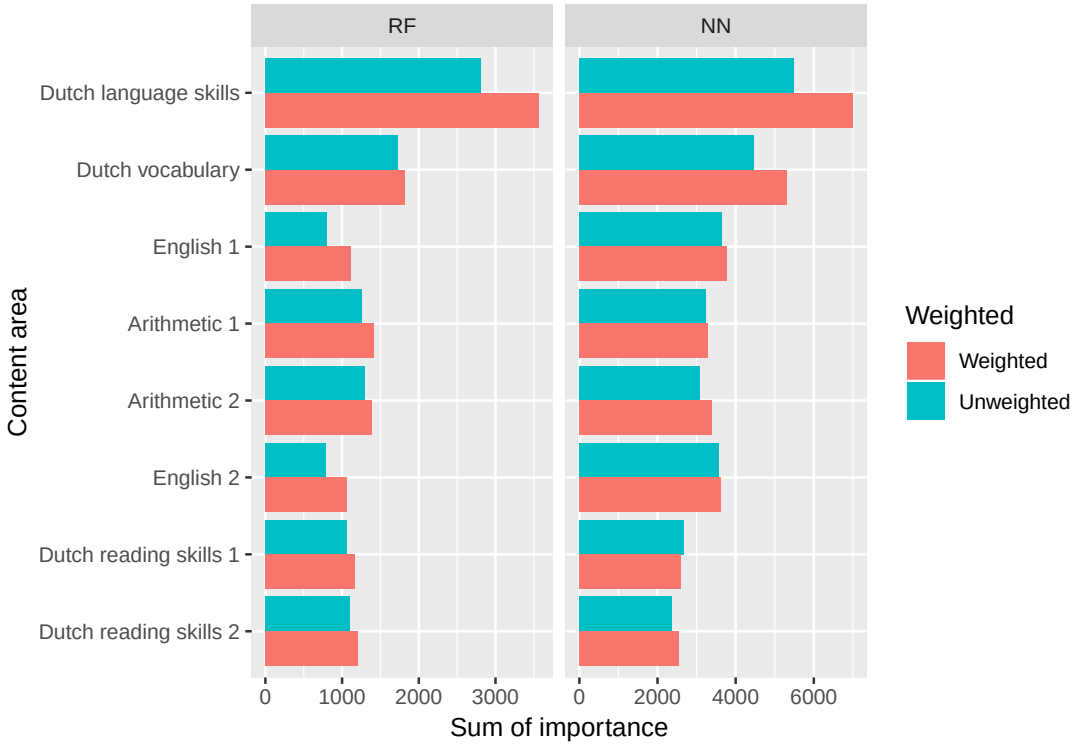

Figure 7.2: Relative feature importance grouped by subject. This figure shows that Dutch language skills is the most important subject followed by Dutch vocabulary. Note that in this figure the models over different test versions have been aggregated.

in both models. In the RF model, arithmetic 1 and 2 seem to be a little bit more important than the other content areas, but in general, the other content areas are similarly influential. In a NN, only Dutch reading skills 1 and 2 appear to be less important than the other content areas.

One interesting pattern in Figure 7.2 is that the sum of importance over all content areas is much greater for the NN (60 079) than for the RF (23 576). This is remarkable because FI has been scaled so that the most important feature has a score of 100. An explanation for this phenomenon can be found in Figure 7.3. Here we see the distribution of the FI per feature and is therefore much more detailed than Figure 7.2. What stands out is that the box plots for the NN are much more stretched out than for the RF. In other words, many features in NNs are more or less important, with some being highly important. In RFs, most features do *not* contribute much (or even anything) to the model while there is a small number of highly informative 'super' features that greatly determine the outcome. In terms of fairness, having a model where most features contribute approximately equally is much more favourable than a model with a small number of super features.

7.2.3 Algorithmic Transparency

The final property of a transparent model concerns the algorithm's transparency. In contrast to simulatability, this is intrinsic to the learning algorithm itself. For example, since we know the shape of the error surface of a linear regression model, we can give guarantees it will perform well even on unseen data. On the other hand, the gradient descent algorithm used for NNs does not give such guarantees and thus

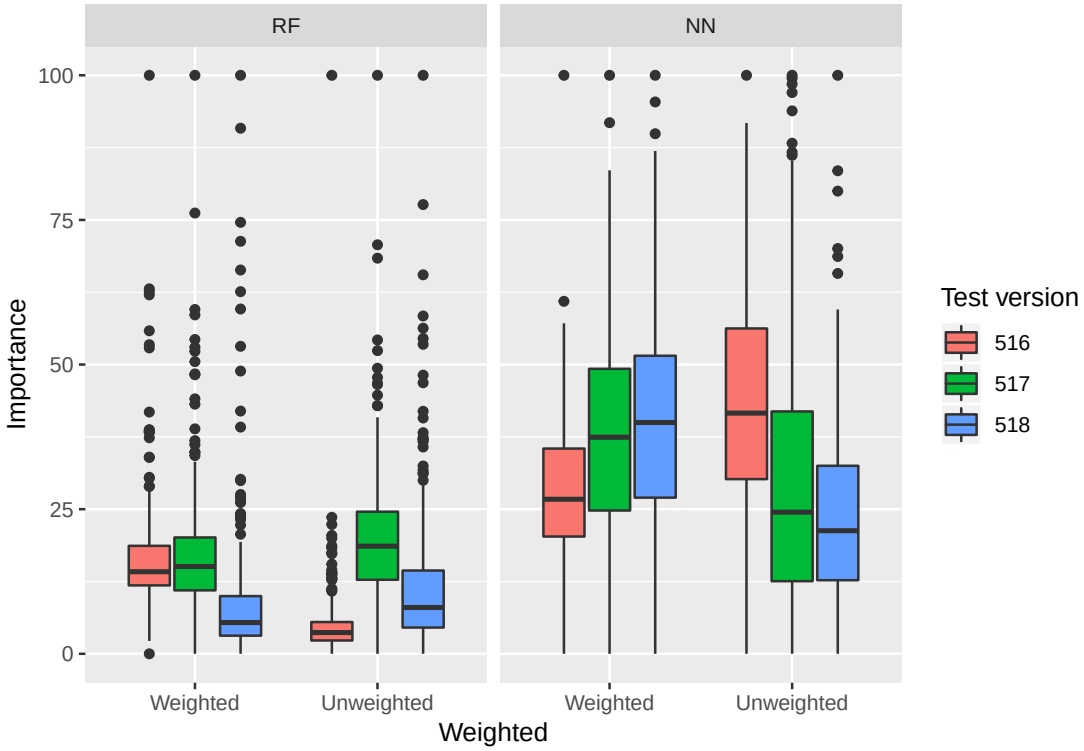

Figure 7.3: The box plots in this figure show the distribution of feature importance between a RF and a NN. For a RF most features are of little importance with many outliers greatly determining one's odds of achieving certain school advice. On the other hand, the feature importance of a NN is much more spread out, indicating that, while some are more difficult than others, there are only few that hardly matter or that are highly informative.

may need retraining. For both RFs and NNs we cannot a priori guarantee whether they will work well on previously unseen data and their algorithmic transparency is hence low. However, for IRT we can guarantee that the sum score of items is directly related to the ability itself; the higher the sum score, the higher the ability since this function is monotone and the sum score is a *sufficient statistic* of the model.

7.2.4 Post-hoc Interpretability

As Lipton (2016) notes, post-hoc interpretability involves explaining models *after the fact*. While this has not been implemented in this research, we provide some directions on how post-hoc interpretability can contribute to understanding the results.

Text Explanations. Consider a situation where a student asks their mentor why they were given certain school advice. The mentor may then describe why they think this is the case. Text explanations for ML follow a similar approach; by giving verbal explanations of the model's strategy, we can inform a student why they were given certain school advice. For example, we can elucidate a student's school advice by (textually) listing some of the items they answered incorrectly but that proved to be important for the model. Another approach is to train a second model (for example a RNN language model) to generate an explanation (e.g. Krening et al., 2017).

Visualisation. Since humans are experts at quickly grasping visual patterns, post-hoc interpretation through visualisation is a popular technique of qualitatively showcasing what a model has learnt. In the context of educational tests, however, we do not immediately see any form of visualisation which can help explain students' school advice.

Local Explanations. Whereas the previous types of post-hoc explanations aim to explain the entire model, it may be useful to explain only a single decision if the former is unattainable. For example, in computer vision applications we may create a *saliency map* by asking what would happen if a single pixel were changed. The gradient descent search pattern can then be superimposed on an image to show what the NN has focussed on. As Edwards and Veale (2017, p. 56) note, a decision may explain both before and after it has occurred as long as one has access to the model. In turn, this allows students to create mental models of how an algorithm came to a decision. However, in the context of educational tests, this may not be applicable since a single changed value is unlikely to impact the model significantly.

Explanation by Example. If a student is to ask a mentor on why they were given certain school advice, the mentor is likely to bring up the results from other students to draw an analogy to theirs. Given enough students, every response pattern has been filled and we can thus find (highly) similar response patterns where the school advice was *different*. By doing this, the student may get a better understanding of why their school advice was different from others who gave slightly different answers. Note that this only works with students who were administered the same test version.

7.3 Discussion and Conclusion

In this chapter, we have described a comparison of IRT and ML in terms of performance and explainability. Before we draw conclusions based on these findings, we first note a couple of limitations and considerations arising from the results.

One limitation of comparing the predictive performance is that we only looked at predictive accuracy. While sensitivity, specificity, and AUC were also discussed, we did not draw any conclusions on this and this was not shown in the ranking. The reason why this was not showed is that we had to train a separate classifier per test version. Consequently, this test version could not be compared to the results of IRT subdivided by class since this does not completely correspond (see Section \ref{{test-moment-and-test-version}). Because AUC was generally higher for IRT, this leads to the question of whether it is better to have a better model *on average* or at this threshold.

When it comes to *decomposability* of the models, we found that Dutch language skills were deemed to be more important than the other content areas. While this simply arose from the data, we can ask ourselves whether such a property is desirable to have in an educational test. In other words, should it be the case that one content area is more important than others, or should all content areas weigh equally? If the latter is considered to be more desirable, tests should be accommodated such that this shows in the data.

In conclusion, we found that all ML models performed better than IRT when we look at the predictive accuracy over all students. More specifically, RFs performed better than NNs even though this was not hypothesised. There was little difference between weighted and unweighted models. When we zoomed in on students who switched, we found the ranking is now suddenly in reverse order; IRT performs by far best followed by NNs and then RFs. Moreover, weighted models proved to provide a substantial improvement to unweighted models. However, when only looking at *whether* a model predicted a switch accurately, we found Matthews correlation coefficient between the predicted switch and ground truth to be highest for RFs, followed by NNs and IRT lagging much behind.

As a second component of performance, we evaluated computational feasibility. Even though ML models took much longer to train than IRT (especially NNs), we concluded both are computational feasible since these models can still easily be trained overnight. Of more data were to be added, this would not pose any problem for IRT but may — at some point — start to become cumbersome for training ML models. To reduce training time, one can either upgrade to a more powerful computer or use a sample to train the model and use that to predict the rest. After all, when it comes to *predicting* school advice, times for both IRT and ML are negligible.

Finally, we utilised the explainability framework by Lipton (2016) to evaluate the explainability of both IRT and ML models. In terms of *simulatability*, IRT is more transparent since a model with a small number of items can be calculated by hand. Furthermore, it is conceptually easy to understand given the real-world value attached to both the difficulty and ability parameters. RFs are somewhat interpretable because one can walk through every decision tree root to leaf to gain a deeper understanding of the model. NNs are least transparent due to the nature of the hidden layer. In terms of *decomposability*, we were able to gain a deeper understanding using the hyperparameters of the models. From this we found weighted models to be complex

than unweighted models. By looking at the feature importance, we found that the content area of Dutch language skills was more important in determining school advice than others. For IRT, we can look at the difficulty parameter distribution and ability distribution to gain a better understanding of how a student compares to others. Although this does not tell us which items were more important for this decision, it does give an idea of which items were more difficult. Thirdly, in terms of *algorithmic transparency*, neither RFs nor NNs can give any guarantee that they will hold for new data. IRT does this to some extent, namely that we know that the test response function is monotone such that a higher sum score relates to a higher ability. Finally, we gave some directions to provide post-hoc interpretability such as providing textual explanations or providing similar response patterns with different school advice.

IV

Conclusion and Discussion

In the last part of this research, we attempt to answer the (sub)research questions. We conclude this part in Chapter 9 by having a brief discussion on the implications, limitation, and recommendations following this research.

Chapter 8

Conclusion

This study investigates how progress tests can be used to establish school advice by comparing methods from the fields of IRT and ML. To this end, we have defined the main research question as follows:

RQ *How can educational tests be used to predict school advice?*

In this chapter, we look back at the previous chapters and try to answer the main research question by first reviewing its subquestions.

Subquestions

In order to answer the main research questions, we look at its subquestions respectively.

SQ1 *What techniques can be used to predict school advice?*
- 1.1 *What are current approaches to predicting school advice?*
- 1.2 *How can domain-specific techniques be used to predict school advice?*
- 1.3 *How can domain-agnostic techniques be used to predict school advice?*

In Chapter 2, we provided an overview of currently used methods to measure educational progress. Concretely, we found that both SEM and IRT in conjunction with an Elo-like rating were used in both adaptive and non-adaptive tests. Then in Chapter 3, we gave an overview of domain-specific and domain-agnostic methods that may be used for predicting school advice. Concretely, we found that IRT is likely the most suitable domain-specific method because of its underlying focus on the theory behind the test. Given that school advice is a discrete outcome variable, some potent domain-agnostic methods from ML include RFs, SVMs, and various types of NNs. As to *how* these techniques can be used, we laid out the process of calibrating both a unidimensional and multidimensional IRT model in Chapter 5, although the latter was rendered inoperable due to population misspecification. Furthermore, we outlined the process of constructing a RF and a NN, the process of tuning the hyperparameters, and the intricacies resulting from having multiple test versions which caused us to train 16 models in Chapter 6.

SQ2 *How do different techniques perform when predicting school advice?*
2.1 *How do domain-specific and domain-agnostic techniques perform in terms of predictive accuracy?*
2.2 *How do domain-specific and domain-agnostic techniques perform in terms of computational feasibility?*

In terms of overall predictive accuracy, we found that ML outperformed IRT with a RF giving more accurate results than a NN. When zooming in on the group of students who switched school type, however, IRT severely outperformed all other models with NNs being more accurate than RFs. Moreover, using case weights substantially improved the ML models for the group of switchers. When looking at *whether* a switch was predicted, ML yielded better results than IRT in terms of Matthews correlation coefficient. Related to computational feasibility (SQ2.2), we found that IRT is much faster than ML (with RFs being faster than NNs) but this may only be an issue if the data set is considerably larger. Even then, alternatives exist to handle this problem and we thus conclude all models are computationally feasible.

SQ3 *How do different techniques perform in terms of explainability?*

In Chapter 7 we evaluated the explainability of both IRT and ML. Overall, IRT is easier to explain since its parameters (difficulty and ability) have real-world value and the overall ability can be explained through the sum score which is a *sufficient statistic*. In terms of *decomposability*, IRT is also easier to break apart since its difficulty and ability parameters provide valuable insight when combined with students' item responses. ML models now perform slightly better since we can look at their hyperparameters and feature importance to get a better understanding of how a model came to a decision. Furthermore, none of the ML models provides decent inherent *algorithmic transparency* while IRT provides some assurances since it is a monotone function where a higher sum score corresponds to a higher ability. Finally, *post-hoc interpretability* allows even greater explainability by providing textual or visual explanations, although this has not been implemented in this research.

Main Research Question

Provided with the insight and inferences from the subquestions, we can integrate this in order to answer our main research question:

RQ *How can educational tests be used to predict school advice?*

School advice is used for determining which school type is most suitable for a student. Two disparate ways to predict school advice is to use either IRT or ML techniques. For IRT, we calibrated a Rasch model on all test versions and extracted an ability estimate using expected a posteriori which is then plugged into a multinomial log-linear regression to calculate students' probabilities for every distinct school advice. For ML we trained one classifier (a RF and a NN) per test version due to a sparse matrix. Furthermore, we followed a cost-sensitive learning approach and applied higher case weights to students who switched between school type. When looking at the performance of all students, RFs proved to perform substantially better than any other model. When concentrating on only those students who switched school type,

IRT outperformed all ML models. Furthermore, IRT was usually the better model on average (in terms of AUC) while ML often performed better on the tested thresholds (sensitivity and specificity). We found all models to be computationally feasible. In terms of explainability, the models are a close match; IRT is more straightforward to understand while ML is more decomposable.

Chapter 9

Discussion

In this chapter, we expand on a number of choices made in this research by first discussing some practical implications, stating a number of practical implications, limitations, and finally providing pointers for future research. Because different test scores on different content areas are difficult to combine and interpret for teachers, this study investigated how to predict school advice for students. To this end, we have compared a number of domain-specific and domain-agnostic techniques and compared them in terms of performance and explainability. In this process, a number of decisions have been made that influence the outcome but also provide us with future work which we reflect on in this chapter.

9.1 Practical Implications

One important aspect of discussing the use of the proposed solution is to see how it can be applied in practice. To this end, we note some practical implications and limitations the models may run into. Additionally, we also note ethical considerations that arise when using algorithms to automatically classify a student for a school type. We focus on the implications of ML although we occasionally make a comparison with IRT to see how the current situation would change.

If the proposed ML models were to be implemented into practice, there are few key aspects that should be taken into consideration. First of all, *concept drift* is likely to occur. Concept drift is best explained by considering an alternative reality where the proposed solution is already implemented (whether this is RFs or NNs is irrelevant) and where students always follow the advice given by the system. Thus, if the model advises a student to go to a different school type than they are currently, they will do so. This means the model then becomes a *positive feedback loop* (Brinkhuis, Bakker, & Maris, 2015). That is, a student will switch school type because the model said so; the model is correct in that the student switched because the model *recommended* the student to do so. To combat this problem, the use of *trackers* has been proposed to deal with such dynamically changing data (e.g. Brinkhuis & Maris, 2019b). As stated before, it is therefore of paramount importance that the model remains for decision support only. While this does not solve the feedback loop, it does diminish it. An indirect effect is the way it may influence teachers. For example, from the ML models it turned out that Dutch reading skills was among the less important content areas. Consequently, teachers may decide to spend less time on teaching this skill because it is deemed 'unimportant'. Of course, in practice this effect would be undesirable.

Earlier we already commented on the fact that the feature importance for NNs is much more equally divided over all items than for RFs. In other words, RFs has a number of 'super' features whereas NN's features contribute more or less equally. While this is a scientifically interesting result, in practice this may be undesirable. After all, one the requirements of educational tests is that it is *fair* to students in the sense that they all have an equal opportunities. Although it may be the case that Dutch language skills are more important than other content areas, it does not mean this should be reflected in the model since students who are lacking on this subject may still have room for improvement. Concretely, imagine a student who — for whatever reason — has been lagging behind on this particular content area; their school advice is now significantly lower than it would have been if all content areas were to weigh in equally despite that they could easily improve their results by one or two weeks of training. To make a comparison, school advice from a teacher does not suffer from this because teachers look at how the students progress week by week instead of at a single (measurement) moment.

One final practical implication is related to the explainability of the models. As we have shown, 'black-box' ML models show a lot of promise in making accurate predictions. With this high performance also comes the cost of complexity; to what extent are we capable to explain how an algorithm comes to a decision? Especially because of the 'right to an explanation' from the General Data Protection Regulation (GDPR) (Council of the European Union & European Parliament, 2016), we pose the question whether it is ethical (or even legal) to use algorithms to influence an important choice in the lives of teenagers. While this is largely mitigated if the model is instead used as decision support, future implementations (with better performance) may play a larger role thereby also imposing greater ethical concerns.

9.2 Limitations

With this project, there are also a few other limitations that need to be taken into account. First of all, this is a study aimed at showcasing differences between IRT and ML within an educational context. Consequently, the created models can by no means be readily implemented into practice. This is also the reason why the *deployment phase* from the CRISP-DM cycle is not implemented. In addition to the practical implications noted before, there are also a number of other limitations that impact the validity of this study. We evaluate this by using the validity framework laid out by Wohlin et al. (2012, p. 102–112).

Internal validity refers to the extent the models influence the outcome. The concerns for this type of validity is two-fold: (1) the underlying validity of the tests itself and (2) the internal validity of the models. Starting with the former, the threat to **instrumentation** is mitigated by having a pre-test to detect any deviating items which are then dropped from the test. **Selection** is only partly mitigated since schools can choose whether they can participate in the SMS or not. If their students performed poorly last year, they are less likely to participate next year. Furthermore, **mortality** occurs since students may change schools, repeat a year, or some other event occurs which causes us to lose track of them. **Diffusion or imitation of treatments** may pose a threat since students in a higher year may pass some answers to a year below them. This is possible because tests are changed only every few years. Lastly, **resentful demoralisation** may be present because some students do not *want* to participate in the tests. As a consequence, they may not put in as much effort and thereby get

different school advice. Internal validity of the model may also be at risk due to the *black-boxness* of ML which leads to a decreased understanding of how the models came to a decision. Another internal validity threat for ML is posed by the seemingly contradictory results found in Table 7.1 and Table 7.2 where the ranking of best performing models based on accuracy reverses. While a possible explanation is that this is caused by the different degrees of flexibility of the model, this is not further explored and is therefore a threat.

External validity is concerned with conditions that limit our ability to generalise the results outside the context of this research. Most notably, **subject population** is a threat because certain groups are not included in the data. Students who switched schools, skipped or repeated a year, or just any student who had an anomaly in their data have been left out and therefore the results may be misleading.

Construction validity involves the relation between theory and observation. First of all, **Inadequate preoperational explications of constructs** were mitigated because we beforehand specified that the 'best' model is the one with the highest predictive accuracy. However, **confounding constructs and levels of constructs** were not accounted for but can be evaluated by future research.

Conclusion validity is concerned with the relationship between the models and the outcome. Although **low statistical power** has been mitigated because the data set consists of 17 891 students, ML could likely profit from having even more data. Because we trained one classifier per test version and only 19% of students switched school type, the number of switchers per school type is still relatively low for training ML models. Adding (many) more observations to the data set is likely to further improve performance for ML but might not have a similar effect on IRT. Furthermore, drawing conclusions was convoluted because of the mismatch in the number of test versions and the number of outcome classes. This led to a comparison of aggregated predictive accuracy with only a loose comparison of sensitivity, specificity, and AUC.

9.3 Future Directions

Based on these results and limitations we also provide some directions for future research.

Different Models, Metrics, and Classes. In this project, we made a number of choices that highly influenced the outcomes and the results to be found. For example, comparing other ML models (i.e. not RFs and NNs) may lead to vastly different results and insights. Similarly, comparing other metrics or choosing different thresholds along the ROC curve also affect which model performs 'best'. Additionally, choosing different classes will highly influence the outcome. That is, by choosing different aggregations of school type there is not only any longer a mismatch between the data and outcome, but it also causes the model to give different attention to different classes. For instance, we have seen how the predictions for HAVO are worse because students can go either upstream or downstream; if more classes are added (i.e. more fine-grained classes), this class discrepancy is much more spread out.

Case Weights. An open question in this research is the one we posed in Section 6.2.5 related to case weights: *how much is the performance of non-switchers allowed to decrease in order for the performance of switchers to increase?* In other words, how much do we

value correctly predicting switchers at the cost of non-switchers? In this research, we have used two extremes, namely one where switchers did not receive any special attention and one where the group of switcher was just as important as non-switchers. While there is no 'optimal' value, a desired value is likely somewhere in the middle which can be evaluated in further research.

Confounding Variables. To make the models as fair as possible (and to provide a fair comparison with IRT), we have opted to only use the item responses in ML to predict school advice. This is a deliberate choice because — although adding more information will improve the model — it may lead to biases against certain groups. For example, the model may give better predictions for boys than girls, or assign a vocational school type more frequently to those students who do not speak Dutch. For the sake of fairness, we want to avoid this. However, even though we only used item responses, it may still be the case that there are *confounding variables* which led to a bias in the model. This can be inspected by either reviewing the predictions by group to see if there are any differences or by adding those variables we think are confounding to the model to see if they are of any importance; if they are not of importance, they are not confounding. Future research may look into this matter.

Integrating IRT and ML. To even further improve the performance of the models, one might look into integrating IRT and ML into a new framework. For example, Pliakos et al. (2019) have already integrated these methods for the prediction of personalised items' recommendation in e-learning. They found their method to perform better than either IRT or ML by itself. This result may carry over to this research, although it remains to be seen how this combined approach will perform on students who switch. Furthermore, multidimensional IRT models could be integrated with ML to provide even further improvement.

References

Agresti, A. (1990). *Categorical data analysis* (3rd ed.). New York: Wiley-Interscience.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, *39*(3), 930–945.

Batchelder, W. H., & Bershad, N. J. (1979, February). The statistical analysis of a thurstonian model for rating chess players. *Journal of Mathematical Psychology*, *19*(1), 39–60. Retrieved 2019-03-07, from https://linkinghub.elsevier.com/retrieve/pii/002224967990004X doi: 10.1016/0022-2496(79)90004-X

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09* (pp. 41–48). Montreal, Quebec, Canada: ACM Press. Retrieved 2019-06-14, from http://portal.acm.org/citation.cfm?doid=1553374.1553380 doi: 10.1145/1553374.1553380

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1987). Occam's razor. *Information processing letters*, *24*(6), 377–380.

Bollen, K. A. (2002). Latent variables in psychology and the social sciences. *Annual review of psychology*, *53*(1), 605–634.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., ... others (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International* (Vol. 2, pp. 77–82). Jerusalem, Israel: IEEE.

Boughorbel, S., Jarray, F., & El-Anbari, M. (2017, June). Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLOS ONE*, *12*(6), 1–17. Retrieved 2019-06-05, from https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0177678 doi: 10.1371/journal.pone.0177678

Bradley, R. A., & Terry, M. E. (1952). Rank Analysis of Incomplete Block Designs: The Method of Paired Comparisons. *Biometrika*, *39*(3/4), 324–345. Retrieved 2019-02-28, from https://www.jstor.org/stable/2334029 doi: 10.2307/2334029

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R., & Stone, C. J. (1984). *Classification and regression trees* (1st ed.). New York: Routledge. Retrieved from https://www.taylorfrancis.com/books/9781315139470

Brinkhuis, M. J. S. (2014). *Tracking educational progress* (PhD Thesis, University of Amsterdam, Amsterdam, The Netherlands). Retrieved from http://hdl.handle.net/11245/1.433219

Brinkhuis, M. J. S., Bakker, M., & Maris, G. (2015). Filtering data for detecting differential development. *Journal of Educational Measurement*, *52*(3), 319–338. doi: 10.1111/jedm.12078

Brinkhuis, M. J. S., & Maris, G. (2010). *Adaptive estimation: how to hit a moving*

target (Measurement and Research Department Reports No. 10-01). Arnhem: Cito. Retrieved from https://www.cito.nl/-/media/files/kennisbank/psychometrie/measurement-and-rd-reports/cito-adaptive-estimation-how-to-hit-a-moving-target-brinkhuis-maris-2010.pdf

Brinkhuis, M. J. S., & Maris, G. (2019a, March). Dynamic estimation in the extended marginal Rasch model with an application to mathematical computer-adaptive practice. *British Journal of Mathematical and Statistical Psychology*, 1–16. doi: 10.1111/bmsp.12157

Brinkhuis, M. J. S., & Maris, G. (2019b, July). Tracking Ability: Defining Trackers for Measuring Educational Progress. In B. P. Veldkamp & C. Sluijter (Eds.), *Theoretical and Practical Advances in Computer-based Educational Measurement* (pp. 161–173). Springer International Publishing. doi: 10.1007/978-3-030-18480-3_8

Brinkhuis, M. J. S., Savi, A., Hofman, A., Coomans, F., van der Maas, H., & Maris, G. (2018, August). Learning As It Happens: A Decade of Analyzing and Shaping a Large-Scale Online Learning System. *Journal of Learning Analytics*, *5*(2), 29–46. doi: 10.18608/jla.2018.52.3

CBS. (2018). *50 jaar Mammoetwet: bijna iedereen gaat nu naar school.* Retrieved 2018-11-21, from https://www.cbs.nl/nl-nl/nieuws/2018/40/50-jaar-mammoetwet-bijna-iedereen-gaat-nu-naar-school

Chalmers, R. P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software*, *48*(6), 1–29. doi: 10.18637/jss.v048.i06

Chapelle, O., Scholkopf, B., & Zien, A. (2009, February). Semi-supervised learning. *IEEE*, *20*(3), 542–542. doi: 10.1109/TNN.2009.2015974

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0 Step-by-step data mining guide* (Tech. Rep.). SPSS.

Chen, T., He, T., Benesty, M., Khotilovich, V., & Tang, Y. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1–4.

Chu, C.-K., Marron, J. S., & others. (1991). Choosing a kernel regression estimator. *Statistical Science*, *6*(4), 404–419.

Coulom, R. (2007, December). Computing "Elo Ratings" of Move Patterns in the Game of Go. *ICGA Journal*, *30*(4), 198–208. Retrieved 2019-03-07, from http://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/ICG-2007-30403 doi: 10.3233/ICG-2007-30403

Council of the European Union, & European Parliament. (2016, May). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, *119*(1), 1–88.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, *40*(2), 139–157.

Edwards, L., & Veale, M. (2017). Slave to the Algorithm? Why a 'right to an explanation' is probably not the remedy you are looking for. *Duke Law & Technology Review*, *16*(01), 18–84. Retrieved 2019-06-19, from https://osf.io/97upg doi: 10.31228/osf.io/97upg

Eggen, T. J., & Sanders, P. (1993). Psychometrie in de praktijk. In *Psychometrie in de praktijk* (pp. 267–271). Arnhem: Cito.

Elkan, C. (2001). The Foundations of Cost-Sensitive Learning. *Proceedings of the*

Seventeenth International Joint Conference on Artificial Intelligence, *17*(1), 973–978.

Elo, A. E. (1978). *The rating of chessplayers, past and present*. London: B.T. Batsford, Ltd.

Embretson, S. E., & Reise, S. P. (2000). *Item response theory for psychologists*. Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119–139.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.

Garson, G. D. (1991, April). Interpreting Neural-network Connection Weights. *AI Expert*, *6*(4), 46–51. Retrieved 2019-07-02, from http://dl.acm.org/citation.cfm?id=129449.129452

Geurts, P., Ernst, D., & Wehenkel, L. (2006, April). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42. Retrieved 2019-07-01, from http://link.springer.com/10.1007/s10994-006-6226-1 doi: 10.1007/s10994-006-6226-1

Glickman, M. E. (1999, August). Parameter Estimation in Large Dynamic Paired Comparison Experiments. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *48*(3), 377–394. Retrieved 2019-03-07, from http://doi.wiley.com/10.1111/1467-9876.00159 doi: 10.1111/1467-9876.00159

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks. *International Conference on Learning Representations*.

Gásquez, R., & Royuela, V. (2016, June). The Determinants of International Football Success: A Panel Data Analysis of the Elo Rating: Determinants of International Football Success. *Social Science Quarterly*, *97*(2), 125–141. Retrieved 2019-03-07, from http://doi.wiley.com/10.1111/ssqu.12262 doi: 10.1111/ssqu.12262

Hambleton, R. K., Swaminathan, H., & Rogers, H. J. (1991). *Fundamentals of item response theory* (Vol. 2). Amherst, USA: Sage.

Hand, D. J. (2009). Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, *77*(1), 103–123.

Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of data mining*. Cambridge, Mass: MIT Press.

Hastie, T. J. (2017). Generalized additive models. In *Statistical models in S* (pp. 249–307). Routledge.

Hebb, D. O. (1949). The organization of behavior; a neuropsychological theory. *A Wiley Book in Clinical Psychology.*, 62–78.

Hofman, A. D., Jansen, B. R. J., De Mooij, S. M. M., Stevenson, C. E., & Van der Maas, H. L. J. (2018, March). A Solution to the Measurement Problem in the Idiographic Approach Using Computer Adaptive Practicing. *Journal of Intelligence*, *6*(1), 14. Retrieved 2019-02-28, from https://www.mdpi.com/2079-3200/6/1/14 doi: 10.3390/jintelligence6010014

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (1st ed.). New York, NY: Springer-Verlag New York.

Jöreskog, K. G., & Sörbom, D. (1989). *LISREL 7: A guide to the program and applications*. Chicago, Illinois: SPSS.

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: An introduction*

to natural language processing, computational linguistics, and speech recognition. Prentice Hall, Pearson Education International.

Klinkenberg, S., Straatemeier, M., & van der Maas, H. (2011, September). Computer adaptive practice of Maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, *57*(2), 1813–1824. Retrieved 2019-03-07, from https://linkinghub.elsevier.com/retrieve/pii/S0360131511000418 doi: 10.1016/j.compedu.2011.02.003

Krebel, U.-G. (1999). Pairwise classification and support vector machines. *Advances in kernel methods: support vector learning*, 255–268.

Krening, S., Harrison, B., Feigh, K. M., Isbell, C. L., Riedl, M., & Thomaz, A. (2017, March). Learning From Explanations Using Sentiment and Advice in RL. *IEEE Transactions on Cognitive and Developmental Systems*, *9*(1), 44–55. Retrieved 2019-07-14, from http://ieeexplore.ieee.org/document/7742965/ doi: 10.1109/TCDS.2016.2628365

Krogh, A., & Hertz, J. A. (1992). A Simple Weight Decay Can Improve Generalization. In J. E. Moody, S. J. Hanson, & R. P. Lippmann (Eds.), *NIPS'91 Proceedings of the 4th International Conference on Neural Information Processing Systems* (pp. 950–957). Denver, CO, USA: Morgan-Kaufmann. Retrieved 2019-07-01, from http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf

Kuhn, M. (2008, November). Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, *28*(5), 1–26. Retrieved 2019-06-11, from http://www.jstatsoft.org/v28/i05/ doi: 10.18637/jss.v028.i05

Kuhn, M. (2014, May). *Futility Analysis in the Cross-Validation of Machine Learning Models* (Tech. Rep.). Groton, USA: Pfizer Global R&D. Retrieved 2019-06-11, from http://arxiv.org/abs/1405.6974 (arXiv: 1405.6974)

Lipton, Z. C. (2016, June). The Mythos of Model Interpretability. In K. Been, D. M. Malioutov, & K. R. Varshney (Eds.), *Proceedings of the 2016 ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)* (pp. 96–100). New York, NY: arXiv.org[stat.ml]. Retrieved 2019-06-19, from http://arxiv.org/abs/1606.03490 (arXiv: 1606.03490)

Little, R. J. A., & Rubin, D. B. (2002). *Statistical Analysis with Missing Data* (2nd ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.

Little, T. D., Schnabel, K. U., Baumert, J., & Schnabel, K. U. (2000, January). *An Introduction to Latent Growth Models for Developmental Data Analysis.* Retrieved 2019-03-10, from https://www-taylorfrancis-com.proxy.library.uu.nl/ doi: 10.4324/9781410601940-9

Lord, F. M. (1980). *Applications of item response theory to practical testing problems*. Mahwah, NJ, US: Lawrence Erlbaum Associates.

Lord, F. M., Novick, M., & Birnbaum, A. (1968). *Statistical theories of mental test scores*. Oxford, England: Addison-Wesley.

Luce, R. D. (1959). *Individual choice behavior*. Oxford, England: John Wiley.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55–60). Baltimore, Maryland: Association for Computational Linguistics. Retrieved 2019-06-13, from https://www.aclweb.org/anthology/P14-5010 doi: 10.3115/v1/P14-5010

Maris, G., Bechger, T., Koops, J., & Partchev, I. (2019). *dexter: Data Management and Analysis of Tests* [Package Readme]. Retrieved from https://CRAN.R-project

.org/package=dexter

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, *405*(2), 442–451.

McArdle, J. J. (1986, January). Latent variable growth within behavior genetic models. *Behavior Genetics*, *16*(1), 163–200. Retrieved 2019-03-06, from http://link.springer.com/10.1007/BF01065485 doi: 10.1007/BF01065485

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of machine learning*. Cambridge, MA: MIT Press.

Nasrabadi, N. M. (2007). Pattern recognition and machine learning. *Journal of electronic imaging*, *16*(4), 049901.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems* (pp. 841–848).

Parker, C. (2011). An Analysis of Performance Measures For Binary Classifiers. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM2011)* (pp. 517 – 526). IEEE. Retrieved from http://www.clparker.org/parker-measure.pdf

Partchev, I. (2004). *A visual guide to item response theory*. Friedrich-Schiller-Universitat Jena.

Pelánek, R. (2016, July). Applications of the Elo rating system in adaptive educational systems. *Computers & Education*, *98*, 169–179. Retrieved 2019-03-07, from https://linkinghub.elsevier.com/retrieve/pii/S036013151630080X doi: 10.1016/j.compedu.2016.03.017

Pliakos, K., Joo, S.-H., Park, J. Y., Cornillie, F., Vens, C., & Van den Noortgate, W. (2019, August). Integrating machine learning into item response theory for addressing the cold start problem in adaptive learning systems. *Computers & Education*, *137*, 91–103. Retrieved 2019-07-05, from https://linkinghub.elsevier.com/retrieve/pii/S036013151930096X doi: 10.1016/j.compedu.2019.04.009

Rao, C. R. (1958). Some Statistical Methods for Comparison of Growth Curves. *Biometrics*, *14*(1), 1–17. Retrieved 2019-02-28, from https://www.jstor.org/stable/2527726 doi: 10.2307/2527726

Rasch, G. (1960). *Studies in mathematical psychology: I. Probabilistic models for some intelligence and attainment tests*. Oxford, England: Nielsen & Lydiche.

Robeer, M. (2018). *Contrastive explanation for machine learning* (Unpublished master's thesis). Utrecht University, Utrecht.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Schölkopf, B. (2001). The kernel trick for distances. In *Advances in neural information processing systems* (pp. 301–307).

Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *6*(1), 1–114.

Somol, P., Pudil, P., Novovičová, J., & Paclık, P. (1999). Adaptive floating search methods in feature selection. *Pattern recognition letters*, *20*(11-13), 1157–1163.

Stoel, R. D., van den Wittenboer, G., & Hox, J. (2004). Methodological Issues in the Application of the Latent Growth Curve Model. In K. van Montfort, J. Oud, & A. Satorra (Eds.), *Recent Developments on Structural Equation Models: Theory and*

Applications (pp. 241–261). Dordrecht: Springer Netherlands. Retrieved 2019-02-28, from https://doi.org/10.1007/978-1-4020-1958-6_13 doi: 10.1007/978-1-4020-1958-6_13

Stroucken, L., Takkenberg, D., & Béguin, A. (2008). Citotoets en de overgang van basisonderwijs naar voortgezet onderwijs. *Sociaaleconomische trends*, *2*, 7–16. Retrieved 2019-03-08, from https://www.cbs.nl/-/media/imported/documents/2008/22/2008-k2v-4p07art.pdf

Swart, L., Van den Berge, W., & Visser, D. (2019, March). De waarde van eindtoetsen in het primair onderwijs. *Policy Brief*, 14. Retrieved 2019-04-02, from https://www.cpb.nl/sites/default/files/omnidownload/CPB-policy-brief-2019-03-de-waarde-van-eindtoetsen.pdf

Tucker, L. R. (1958, March). Determination of parameters of a functional relation by factor analysis. *Psychometrika*, *23*(1), 19–23. Retrieved 2019-03-06, from https://doi.org/10.1007/BF02288975 doi: 10.1007/BF02288975

Valk, M. (2018). *Interpretable Recurrent Neural Networks for Heart Failure Rehospitalisation Prediction* (Master's thesis, Utrecht University, Utrecht). Retrieved from https://dspace.library.uu.nl/bitstream/handle/1874/365568/M.C.Valk_Thesis_3830810.pdf?sequence=2

Van Boxtel, H., Engelen, R., & De Wijs, A. (2011). *Wetenschappelijke verantwoording van de Eindtoets Basisonderwijs 2010*. Arnhem: Cito. Retrieved 2019-08-03, from https://formulieren.cito.nl/~/media/cito_nl/files/onderzoek%20en%20wetenschap/cito_wetenschappelijke_verantwoording_eindtoets.ashx

Van Til, A., & Van Boxtel, H. (2015). *Wetenschappelijke verantwoording Toets 0 t/m 3, tweede generatie*. Arnhem: Cito. Retrieved 2018-11-27, from https://www.cito.nl/-/media/Files/kennisbank/cito-bv/96_wetenschappelijke-verantwoording-volgsyteemvo-gen2.pdf?la=nl-NL

Venables, W. N., & Ripley, B. D. (2002). *Modern Applied Statistics with S* (Fourth ed.). New York: Springer. Retrieved from http://www.stats.ox.ac.uk/pub/MASS4

Verhelst, N., & Verstralen, H. (1994). *The one parameter logistic model: computer program and manual*. Arnhem: Universiteit Twente, CITO. Retrieved from https://research.utwente.nl/en/publications/the-one-parameter-logistic-model-computer-program-and-manual

Waa, J. v. d., Robeer, M., Diggelen, J. v., Brinkhuis, M. J. S., & Neerincx, M. (2018, June). Contrastive Explanations with Local Foil Trees. In *2018 Workshop on Human Interpretability in Machine Learning (WHI)*. Stockholm, Sweden. Retrieved from https://arxiv.org/pdf/1806.07470.pdf&sa=D&ust=1537878080177000.pdf

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* (PhD.). Harvard University.

Wickham, H. (2007). Reshaping Data with the reshape Package. *Journal of Statistical Software*, *21*(12), 1–20. Retrieved from http://www.jstatsoft.org/v21/i12/

Wickham, H., & Henry, L. (2019). *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions*. Retrieved from https://CRAN.R-project.org/package=tidyr

Wishart, J. (1938). Growth-Rate Determinations in Nutrition Studies with the Bacon Pig, and Their Analysis. *Biometrika*, *30*(1/2), 16–28. Retrieved 2019-03-06, from https://www.jstor.org/stable/2332221 doi: 10.2307/2332221

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Berlin, Heidelberg: Springer.

Wright, M. N., & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, *77*(1), 1–17. doi: 10.18637/jss.v077.i01

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., … Steinberg, D. (2008, January). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1–37. Retrieved 2019-07-21, from http://link.springer.com/10.1007/s10115-007-0114-2 doi: 10.1007/s10115-007-0114-2

Appendix A

Education in the Netherlands

This appendix explains the various school types and tracks a student must go through to complete their education. First, we outline the different parts of the Dutch educational system and how they contribute to their academic achievement. Next, we take a closer look at the various components in Dutch education and the choices a student can make along the way. Since this research is in the context of secondary education, we scrutinise the different secondary educational tracks and how they lead up to choosing a specific type of tertiary education.

A.1 Composition of the Educational System

Education in the Netherlands is divided into three parts: (i) primary education, (ii) secondary education, and (iii) tertiary education. Primary education (also called *elementary* education) starts at the age of 5 although children of 4 years old can usually be admitted. After the completion of primary education, children go to *secondary* education at the age of 12 until the age of 16, 17, or 18 depending on the school type. Section A.3 identifies and explains the various types of secondary education students may choose between. Both primary and secondary education are compulsory to attend until students reach the age of 16. Then, education is only partially compulsory to attend for two days a week. This is also depicted in Figure A.1. Finally, tertiary education consists of both further and higher education (FE and HE respectively). Concretely, FE is a type of tertiary vocational education known as MBO (middle-level applied education or *"middelbaar beroepsonderwijs"* in Dutch) which is subdivided into four more levels. HE consists of both HBO (higher-level applied education or *"hoger beroepsonderwijs"* in Dutch) and WO (scientific education or *"wetenschappelijk onderwijs"* in Dutch, colloquially referred to as university) which focusses on more theory demanding jobs and scientific education. Their relation is portrayed in Figure A.1.

A.2 From Birth to Job

When children have just been born, they usually stay at home with one of their parents until they are old enough to go to daycare. From the ages of 2 to 4, children can go to kindergarten which is a type of pre-school education. While kindergarten is non-compulsory, it is popular because it allows both parents to get a full-time job. Children of 5 years old have to be admitted to primary school which is compulsory

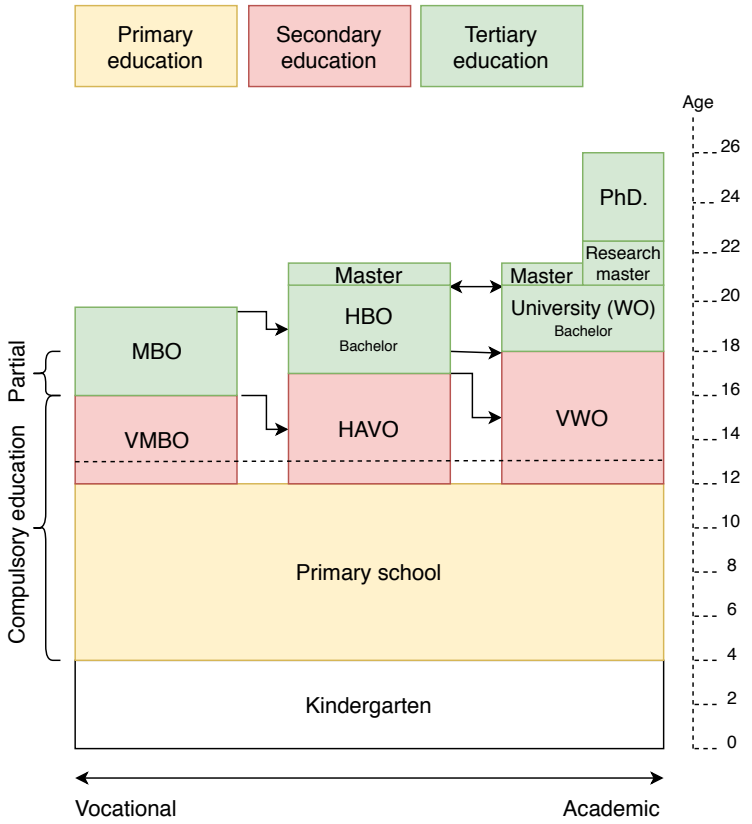

Figure A.1: The different levels of education in the Netherlands.

education, although 4-year-olds are usually also accepted. After 8 years at the age of 12, students choose a school type to go to as part of their secondary education. Which school types exist and how school advice is established is discussed in Section A.3.

Students' first year (or first two years) of secondary education is called the *bridge year* during which most students make a definitive choice for which school type to follow. This is possible because most schools offer classes with mixed school types, e.g. HAVO/VWO or VMBO GL/TL. This makes it easier for students to find out which school type suits them best and switch accordingly without too much of an impact. When finishing the VMBO school type, students can go to either MBO or complete HAVO in one more year. HAVO students can go to MBO, HBO, or complete VWO with one more of study. Additionally, HAVO students can take a full-time job even if they are under the age of 18. This is possible because, even though students must follow education for at least two days a week under the age of 18, obtaining a so-called *initial qualification* releases them from this obligation. Finally, VWO graduates may attend any form of tertiary education (i.e. MBO, HBO, or WO) or start a full-time job.

Transfers from secondary to tertiary education are depicted in Figure A.1. Since VMBO students still have to attend some form of education for two days a week, most students transfer to MBO after which they pursue a full-time job. Additionally, they can transfer to the HBO level. In contrast, HAVO students have the option to take on a full-time job since they no longer have to attend compulsory education. They may also go to the MBO or HBO tertiary education or spend one more year in secondary education to complete VWO. After graduating from VWO, students can go

to MBO or HBO, although the majority tends to go to WO (referred to as university) to attain a bachelor's degree. Subsequently, students who complete their bachelor's programme may pursue a (research) master and even a PhD.

A.3 School Types in Secondary Education

This section describes the various school types that exist in Dutch secondary education. When students graduate from primary school, they must choose which school type to go to. This choice does not happen arbitrarily; *school advice* is typically established from two components: advice the teacher gives and an extensive final test at the end of primary education (although the latter not required by law). The advice by the teacher is a teacher's own opinion of which school type they think a student fits best. Even though this is not an objective measure, it is usually highly appreciated (and thus usually more important than the results of the end test of primary education) because this advice encapsulates the experiences of a whole year (or multiple years) rather than one to three measurement moments (as is the case for the end test of primary education). In other words, school advice given by the teacher is valuable since a teacher also looks at the personal side of a student and has more time to establish advice. On the other hand, the end test of primary education is an objective, national measure to compare a student relative to their peers.

We now list the various school types a student may choose considering the school advice provided by both the teacher and the end test of primary education.

- **VMBO** — Preparatory middle-level applied education (*"voorbereidend middelbaar beroepsonderwijs"* in Dutch) is a type of pre-vocational education which aims to prepare students for furthered (vocational) education. This school type spans the ages of 12-16 years old and can roughly be divided as having vocational and theoretical tracks. All tracks prepare students for the MBO level in tertiary education (albeit different types of MBO).
  - Pro — Practical education (*"praktijkonderswijs"* in Dutch) is an educational track for those who would otherwise not be able to attain a VMBO diploma. Practical education is not included in this research.
  - BB — Basic vocational programme (*"basisberoepsgerichte leerweg"* in Dutch) aims to train students for a specific profession and is thus more practical than the other programmes.
    - BB+ — The same as BB, but with extra support for students who have difficulty with the regular BB programme.
  - KB — Middle management vocational programme (*"kaderberoepsgerichte leerweg"* in Dutch) balances both vocational and theoretical education and prepares students for either middle management or vocational training at the MBO level.
  - GL — Mixed programme (*gemengde leerweg"* in Dutch) is a mixture of KB and TL.
  - TL — Theoretical programme (*"theoretische leerweg"* in Dutch) is the highest (i.e. most theoretical) programme in the VMBO school type and prepares students for the most difficult type of MBO education. Moreover, after completing this track it is possible to go to the HAVO school type. Since

TL does not differ much from GL, we use a combined type GT (mixed theoretical or *" gemengd theoretisch"* in Dutch) to denote both these tracks.

- **HAVO** — Higher general continued education (*"hoger algemeen voortgezet onderwijs"* in Dutch) is followed by approximately 30% of students and typically lasts five years from the ages of 12 to 17. It prepares students for the HBO (polytechnic) level. After graduating, it is also possible to go to the VWO level (year 5).
- **VWO** — Preparatory scholarly education or pre-academic education (*"voorbereidend wetenschappelijk onderwijs"* in Dutch) is a school type focussed on theory which prepares students for going to university. It is followed by approximately 20% of all students and lasts from the ages of 12 to 18. There are two types of VWO.
  - Atheneum — The 'default' type of VWO.
  - Gymnasium — Replaces one language subject (not English or Dutch) with Greek or Latin. Hence, the Gymnasium school type provides students with a classical background which can be helpful in medical studies or to broaden students' common knowledge.

Appendix B

Data Description Table

Table B.1: This table lists all variables in the data set and provides a brief description of what they contain.

Column name	Type	Description
student_id	Numeric	ID unique to a single student
test_moment	Factor	Which test moment, e.g. T_0 or T_1
test_id	Factor	Which test version was used
school_advice	Factor	School advice for a student
response	Character	Scored response string (i.e. dichotomous)
ss_ftpe	Numeric	**S**tandardised **S**core on the **F**inal **T**est **P**rimary **E**ducation
school_type	Factor	School type corresponding to test version
test_year	Factor	Year in which the test was taken
test_time	Date	Specific date of when the test was taken
digital	Logical	Whether it is a computerised test or not
first_letters	Character	First letter(s) of a student's first name
articles	Character	Prepositions or articles between first name and surname, e.g. 'van' or 'van der'
surname	Character	Surname of the student
birth_date	Date	Date of birth
sex	Factor	Sex at birth
language_home	Factor	Primary language spoken at home
sector	Factor	Sector of the student (only applies to VMBO). E.g. business or agriculture
profile	Factor	Educational track students intend to follow in year 4. E.g. Culture and Society or Nature and Health
class	Character	ID code of students' school class, e.g. 1A or 2B
school_id	Numeric	ID code of students' school
unscored_response	Character	Unscored response string (i.e. polytomous)
school_advice_alt	Factor	Non-aggregated school advice

Appendix C

Multidimensional IRT Analysis

In order to calibrate a multidimensional scale, the *MIRT* (**M**ultidimensional **I**tem **R**esponse **T**heory) package is used in R (Chalmers, 2012). In contrast to *Dexter*, this package requires data in a wide format. So, we use the data set in long format as described in Section 5.1 as a stepping stone in order to transform it into a wide format. We convert long to wide format by utilising the `gather` function from the *tidyr* package (Wickham & Henry, 2019).

C.1 Estimating Item Parameters

The procedure for estimating parameters using the *MIRT* package is similar to that of *Dexter* — i.e. we create one model per test moment.

The difficulty parameters from MIRT are plotted as an empirical cumulative distribution function (ECDF) in Figure C.1. At a glance, the item parameters seem relatively similar to those estimated by *Dexter.* However, the high standard error points out something is wrong with the estimates. Where *Dexter* uses conditional maximum likelihood (CML), *MIRT* uses marginal maximum likelihood (MML) which is a different approach but does not explain the large differences between the two packages. Figure C.2 shows the relationship between *MIRT* and *Dexter* at test moment T_0. The relation appears to be a strong linear correlation, $\rho = 0.98$. However, Figure C.2 also points out a striking pattern in the item parameters: there is a clear separation between the different test versions. More specifically, although the relation between estimated item parameters by *Dexter* and *MIRT* is linear, the parameters returned by *Dexter* are much higher than those reported by *MIRT*. For example, consider the point where *Dexter* estimates parameters to have a difficulty of 0. In contrast, *MIRT* estimates these items to have a difficulty between -1.1 and -0.5 depending on the test version.

The reason why this happens is because of population misspecification. In targeted testing design, MML assumes students have been sampled from the same population. In this case, however, there seem to be multiple subpopulations. Consider two situations for the background variable $\mathcal{Y}$ which provides some information about students:

1. $\mathcal{Y}$ plays no role in sampling students, only in determining which items will be administered.

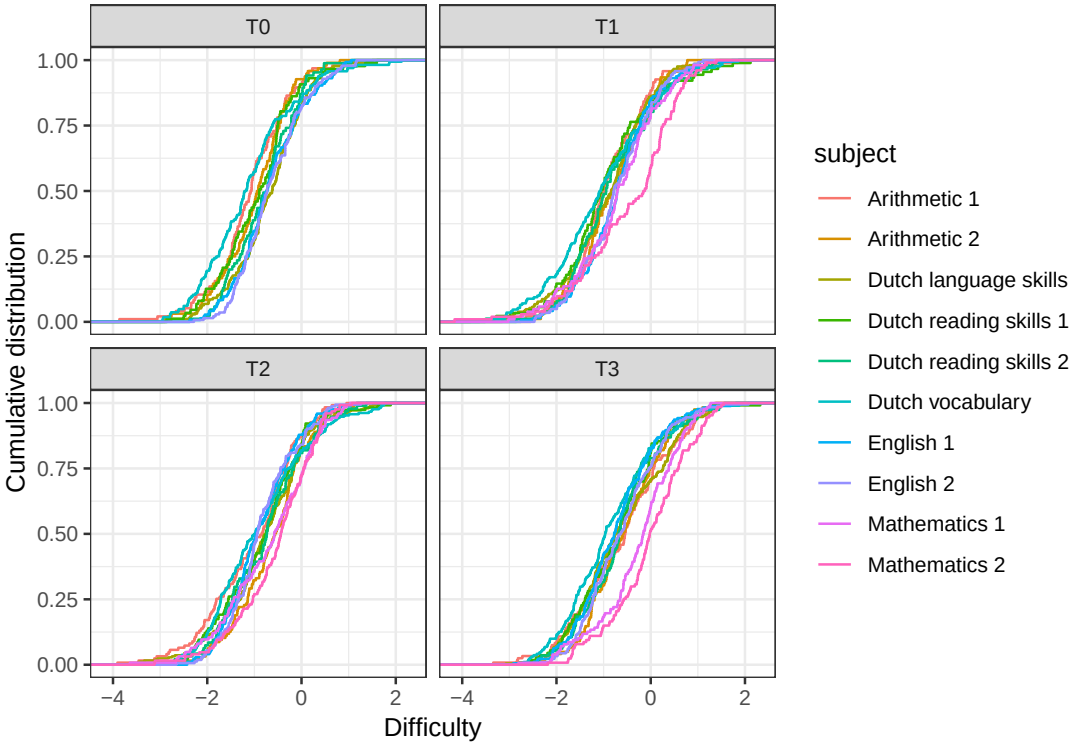

Figure C.1: Empirical cumulative distribution functions (ECDF) of the item difficulties estimated by MIRT. Mathematics 1 and 2 seem to be amongst the harder topics for students.

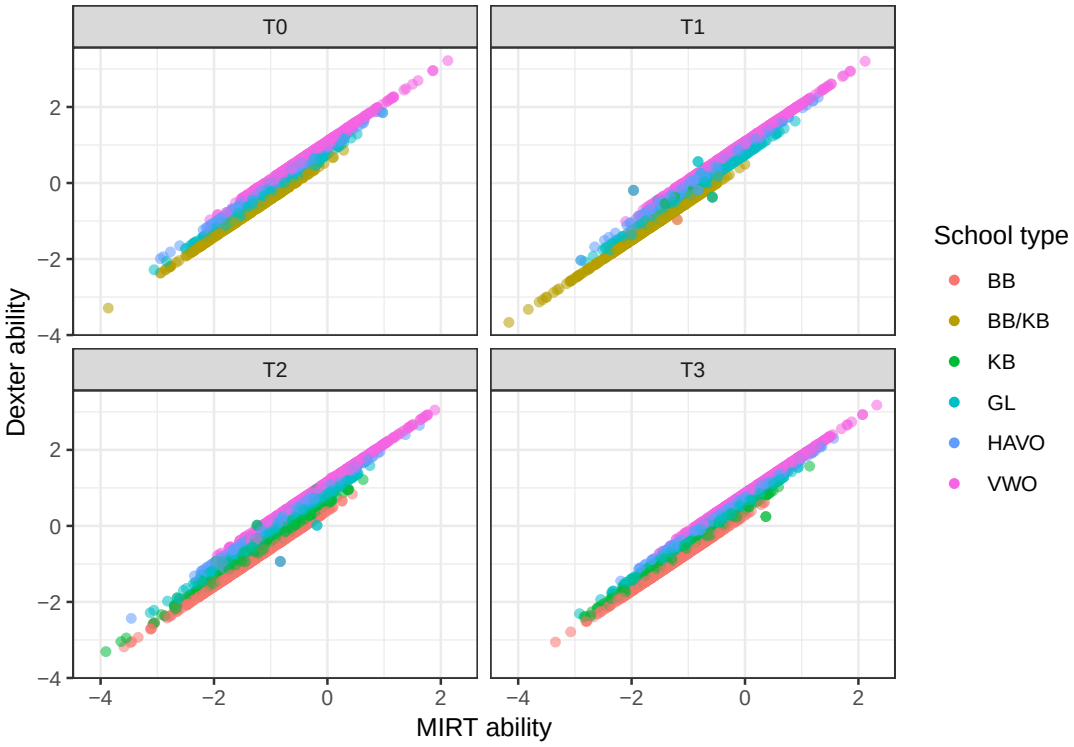

Figure C.2: Comparison of item parameters estimated by Dexter and MIRT at test moment T_0. The distinct lines indicating school advice seem to hint at population misspecification.

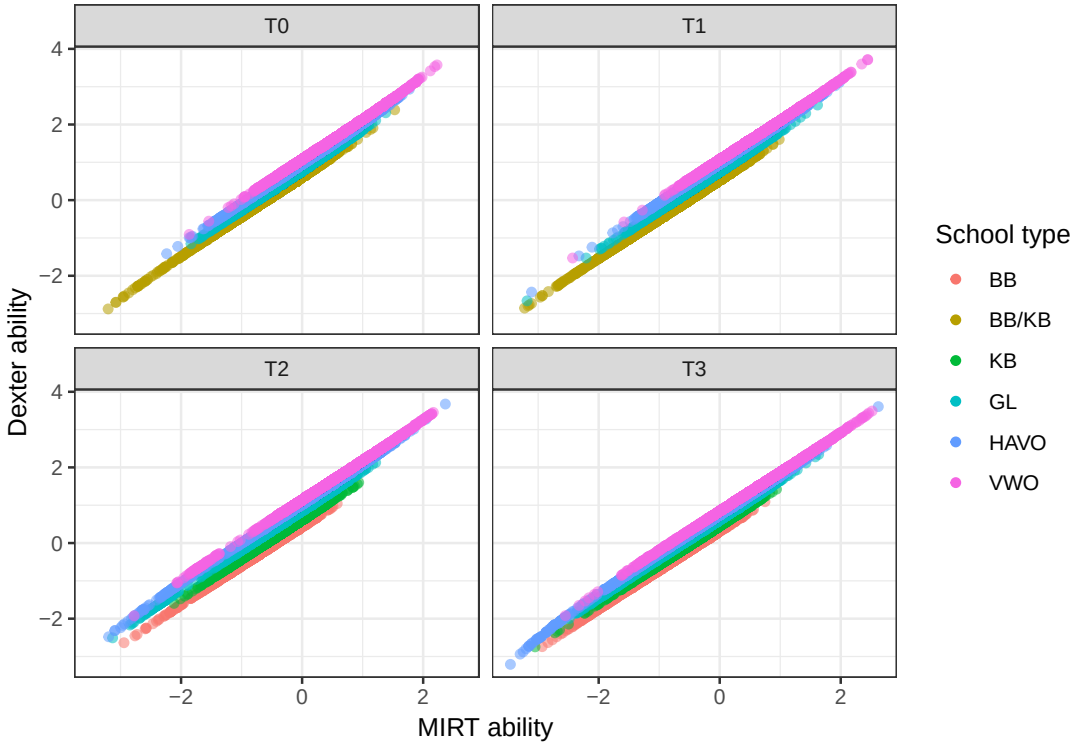

Figure C.3: This figure shows that there is a clear separation between school types when it comes to ability when this ability is estimated by *Dexter* or *MIRT*.

2. $\mathcal{Y}$ plays both a role in sampling students and in determining which items will be administered.

In the second, situation we are interested in finding both the item and person parameters from each of the ability distributions corresponding to levels of the background variable $\mathcal{Y}$ (Eggen & Sanders, 1993, p. 269–271). It may now be easy to see that $\mathcal{Y}$ is, in fact, the school type of a student. In this testing design, we have not sampled students independently of their school type but rather applied stratified sampling, so leaving out that information when performing MML analysis will lead to incorrect results since we then ignore the missing at random (MAR) condition of MML (Eggen & Sanders, 1993).

As Eggen and Sanders (1993, p. 271) note, we can overcome this problem by specifying $\mathcal{Y}$ in the MML analysis. This is certainly possible in *MIRT* by using the `mixedmirt` function to specify which school type belongs to which student. However, this is exactly what we like to be determined by IRT and by specifying this beforehand the predictions become self-fulfilling. For completeness, we evaluate the effect of population misspecification on the person parameters.

C.2 Estimating Person Parameters

Similar to the item parameters, the person parameters estimated by *MIRT* also suffer from population misspecification. A comparison can be found in Figure C.3. Similar to Figure C.2, there is a clear distinction between the ability of different school types when they are estimated by either *MIRT* or *Dexter*.

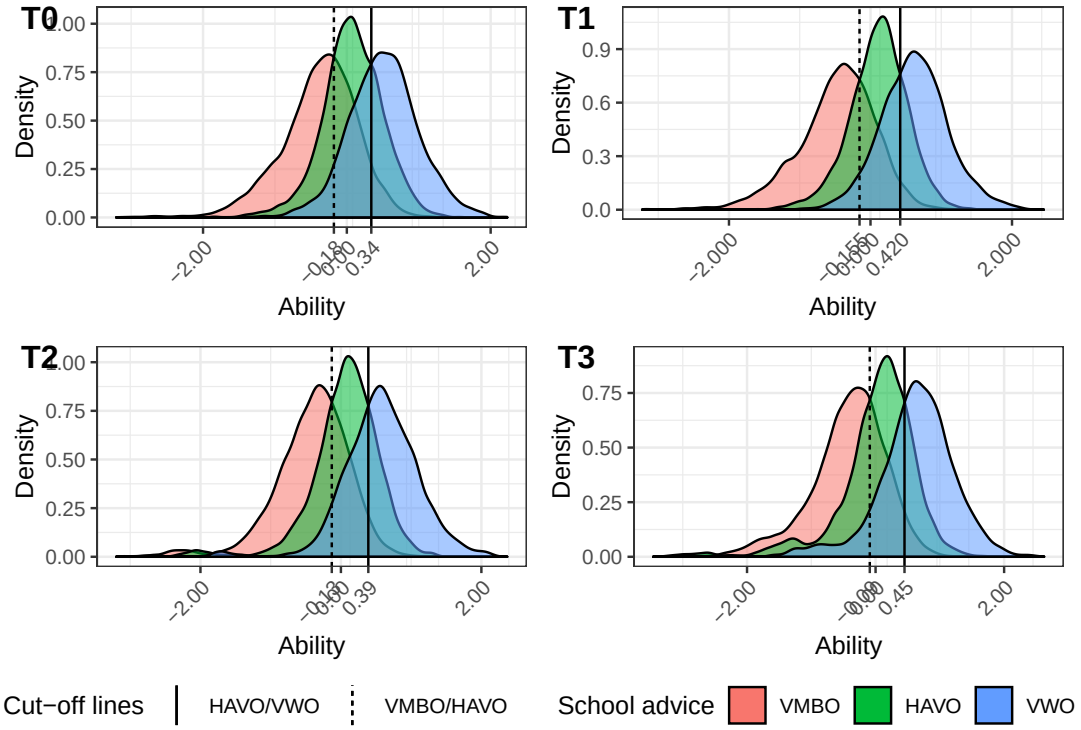

Figure C.4: In comparison to Figure 5.5, the density functions of the school types are much closer together and are therefore guaranteed to have a higher error rate. This occurs because of population misspecification.

To make this problem concrete, compare Figure C.4 to Figure 5.5. Between these charts, it stands out that the distributions per school type are closer together in the models estimated by *MIRT* than in those generated by *Dexter*. Hence, the misclassification rate will be much higher for these models since they contain more overlap. To estimate the impact of this problem, we look at various performance measures in a similar fashion as for the *Dexter* models. The result is visualised in Figure C.5.

When comparing the results in Figure C.5 (confusion matrices given in Table C.1) to those in Figure 5.6, we see that nearly all metrics are approximately 0.1 lower for all school types. The overall AUC is 0.55, 0.59, 0.6, and 0.61 for T_0 to T_3 respectively which is much lower than for the Rasch model constructed by *Dexter*. Thus, because of this violation of the missing data, the fitted school advice is significantly worse for models estimated by *MIRT*.

Finally, we briefly provide a comparison of a Rasch, 2PL and multidimensional (2 factors) model estimated by *MIRT*. This is also intended to emphasise the importance of population misspecification. Figure C.6 shows four ability distributions for three models, (1) for a Rasch model, (2) for a 2PL model, and (3) and (4) for a 2PL model with two underlying factors. The fit gets worse as the model becomes more complex. In fact, the cut-off lines for VMBO/HAVO and HAVO/VWO in Figure C.6 in (3) and (4) are swapped, indicating that students with a *higher* ability are assigned to a *more vocational* school type. Furthermore, a one-way ANOVA was conducted to compare the distributions of a Rasch and 2PL model estimated by *MIRT*. There was a significant difference between the two models, $p < .001$ (Table C.2).

Table C.1: Confusion matrices per test moment for a Rasch model made with *MIRT*.

(a) T_0

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.646**	0.286	0.105
HAVO	0.277	**0.413**	0.287
VWO	0.077	0.301	**0.607**

(b) T_1

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.718**	0.288	0.085
HAVO	0.237	**0.452**	0.308
VWO	0.045	0.26	**0.607**

(c) T_2

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.687**	0.286	0.083
HAVO	0.269	**0.471**	0.283
VWO	0.044	0.243	**0.634**

(d) T_3

	Reference		
Predictions	VMBO	HAVO	VWO
VMBO	**0.709**	0.336	0.092
HAVO	0.249	**0.447**	0.235
VWO	0.041	0.217	**0.673**

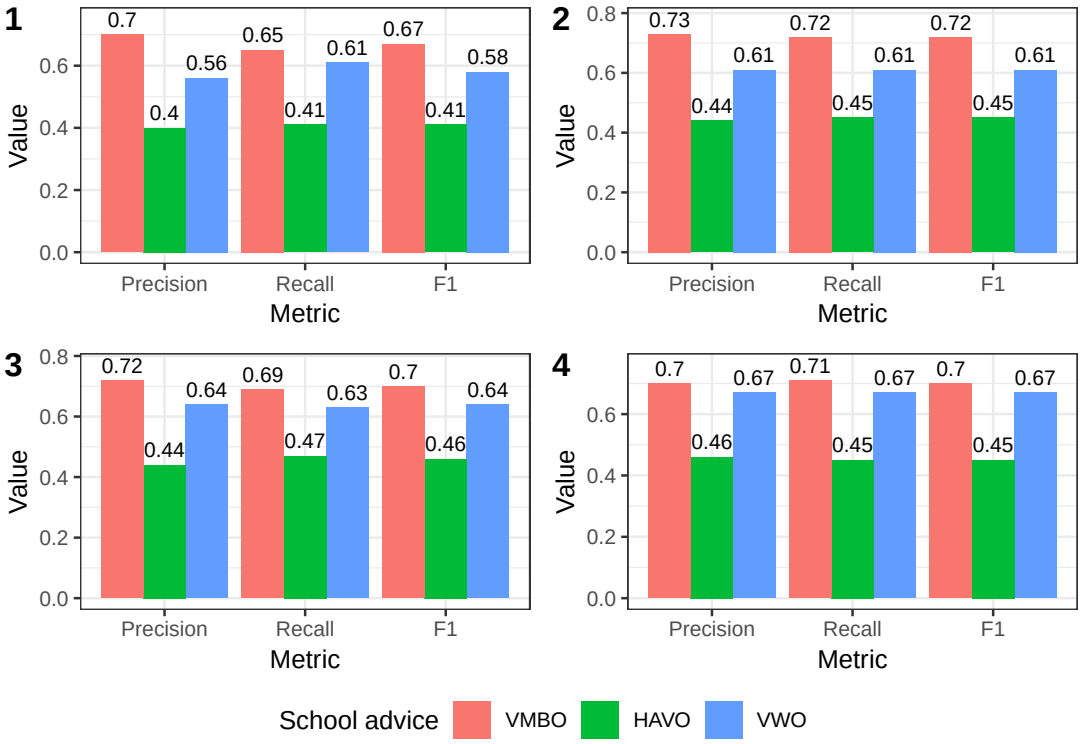

Figure C.5: Similar to Figure 5.6, these plots show how various cut-off points affect the ROC curve. In contrast to the person parameters estimated by Dexter, these ROCs have much less AUC.

Table C.2: Analysis of variance between a Rasch and a 2PL model showing Akaike's information criterion (AIC), Bayes information criterion (BIC), log likelihood, $\mathcal{X}^2$ statistic, degrees of freedom (df), and the significance value (p).

	AIC	BIC	Log likelihood	$\mathcal{X}^2$	df	p
Rasch	6071870	6079569	-3034947	-	-	-
2PL	6019226	6034607	-3007639	54616.52	986	0

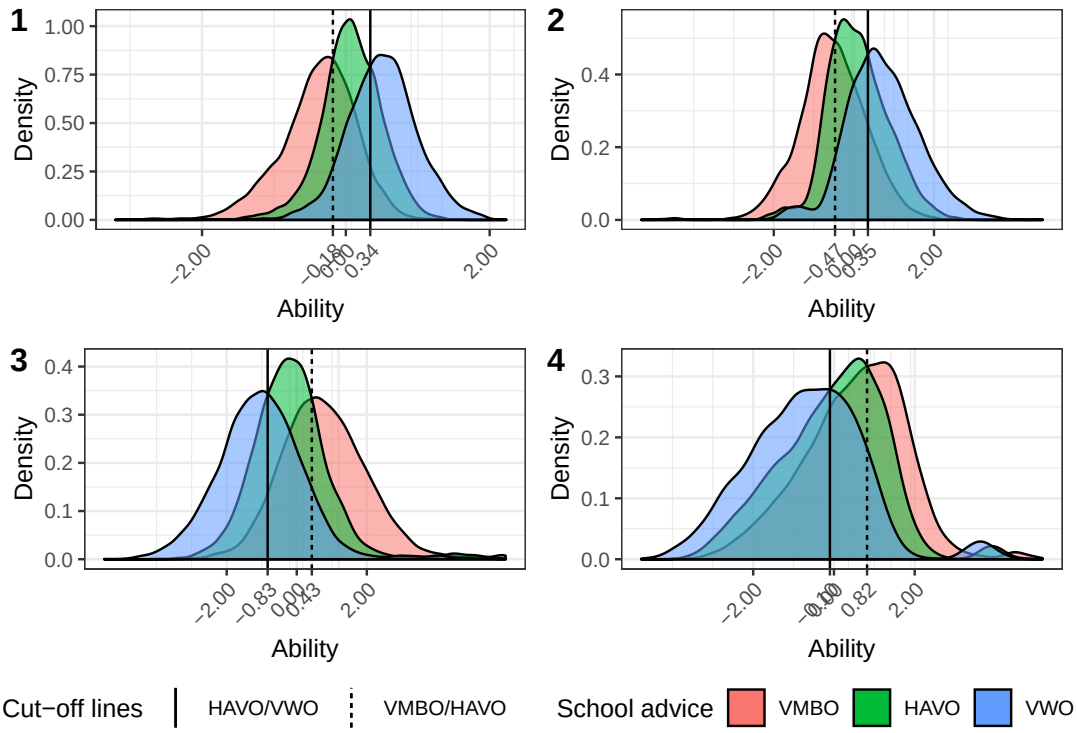

Figure C.6: This figure depicts four ability distributions from different models at test moment T_0; in (1), the ability distribution for a Rasch model is shown. Next to that, (2) shows the ability distribution under a 2PL model. Finally, (3) and (4) depict the ability distribution for a multidimensional 2PL model with two factors.

Appendix D

Confusion Matrices

D.1 Confusion Matrices for Cut-off Density Functions

Table D.1: Non-proportional confusion matrices corresponding to Table 5.5.

(a) T_0

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**5439**	1135	198	6772
HAVO	1857	**3031**	1417	6305
VWO	216	1490	**3108**	4814
Total	7512	5656	4723	**17891**

(b) T_1

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**5895**	973	125	6993
HAVO	1528	**3385**	1579	6492
VWO	89	1298	**3019**	4406
Total	7512	5656	4723	**17891**

(c) T_2

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**5872**	1078	132	7082
HAVO	1553	**3428**	1392	6373
VWO	87	1150	**3199**	4436
Total	7512	5656	4723	**17891**

(d) T_3

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**5714**	1502	301	7517
HAVO	1683	**3152**	1122	5957
VWO	115	1002	**3300**	4417
Total	7512	5656	4723	**17891**

D.2 Confusion Matrices for ML Models

Table D.2: Confusion matrices for the 16 ML models described in Chapter 6.

(a) Confusion matrix for a weighted RF model in test version 515

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**935**	2	0	937
HAVO	0	**0**	0	0
VWO	0	0	**0**	0
Total	935	2	0	**937**

(b) Confusion matrix for a weighted NN model in test version 515

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**935**	2	0	937
HAVO	0	**0**	0	0
VWO	0	0	**0**	0
Total	935	2	0	**937**

(c) Confusion matrix for an unweighted RF model in test version 515

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**935**	2	0	937
HAVO	0	**0**	0	0
VWO	0	0	**0**	0
Total	935	2	0	**937**

(d) Confusion matrix for an unweighted NN model in test version 515

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**935**	2	0	937
HAVO	0	**0**	0	0
VWO	0	0	**0**	0
Total	935	2	0	**937**

(e) Confusion matrix for a weighted RF model in test version 516

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**1094**	159	3	1256
HAVO	16	**13**	0	29
VWO	0	0	**0**	0
Total	1110	172	3	**1285**

(f) Confusion matrix for a weighted NN model in test version 516

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**1094**	159	3	1256
HAVO	16	**13**	0	29
VWO	0	0	**0**	0
Total	1110	172	3	**1285**

(g) Confusion matrix for an unweighted RF model in test version 516

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**1094**	159	3	1256
HAVO	16	**13**	0	29
VWO	0	0	**0**	0
Total	1110	172	3	**1285**

(h) Confusion matrix for an unweighted NN model in test version 516

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**1094**	159	3	1256
HAVO	16	**13**	0	29
VWO	0	0	**0**	0
Total	1110	172	3	**1285**

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**5**	3	0	8
HAVO	224	**1229**	295	1748
VWO	1	21	**33**	55
Total	230	1253	328	**1811**

(i) Confusion matrix for a weighted RF model in test version 517

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**5**	3	0	8
HAVO	224	**1229**	295	1748
VWO	1	21	**33**	55
Total	230	1253	328	**1811**

(j) Confusion matrix for a weighted NN model in test version 517

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**5**	3	0	8
HAVO	224	**1229**	295	1748
VWO	1	21	**33**	55
Total	230	1253	328	**1811**

(k) Confusion matrix for an unweighted RF model in test version 517

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**5**	3	0	8
HAVO	224	**1229**	295	1748
VWO	1	21	**33**	55
Total	230	1253	328	**1811**

(l) Confusion matrix for an unweighted NN model in test version 517

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**0**	0	0	0
HAVO	1	**29**	33	63
VWO	4	241	**1023**	1268
Total	5	270	1056	**1331**

(m) Confusion matrix for a weighted RF model in test version 518

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**0**	0	0	0
HAVO	1	**29**	33	63
VWO	4	241	**1023**	1268
Total	5	270	1056	**1331**

(n) Confusion matrix for a weighted NN model in test version 518

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**0**	0	0	0
HAVO	1	**29**	33	63
VWO	4	241	**1023**	1268
Total	5	270	1056	**1331**

(o) Confusion matrix for an unweighted RF model in test version 518

	Reference			
Predictions	VMBO	HAVO	VWO	Total
VMBO	**0**	0	0	0
HAVO	1	**29**	33	63
VWO	4	241	**1023**	1268
Total	5	270	1056	**1331**

(p) Confusion matrix for an unweighted NN model in test version 518