Computational detection of a high-risk DLBCL group

Lukas Geßl

A thesis presented for the degree of
Master of Science

First Supervisor: Prof. Dr. Harald Garcke
Second Supervisor: Prof. Dr. Rainer Spang

Department of Mathematics
University of Regensburg

August 17, 2024

Abstract

Chemotherapy with R-CHOP is the standard treatment for diffuse large B-cell lymphoma, the most common type of non-Hodgkin lymphoma, achieving a cure for about two thirds of patients. Survival for the remaining third with refractory or relapsed disease, however, remains poor. Pharma-sponsored randomized trials in the whole DLBCL population to date have failed to improve R-CHOP. The International Prognostic Index (IPI), the only widely accepted risk-assessment tool for DLBCL and an easy clinical test, fails to identify a high-risk DLBCL subpopulation that is large and precise enough to trigger research and enable clinical trials for new treatments that outperform R-CHOP on this subpopulation.

This thesis aims to develop a computational method that identifies DLBCL patients with progression-free survival (PFS) below two years with higher prevalence and significantly higher precision than the IPI and to show this on independent data. It also deals with the question under which circumstances we can do so reliably.

After introducing into DLBCL, the IPI together with its shortcomings and MMML-Predict, the project that this thesis is part of, in the first chapter, we describe the statistical frame and models as well as our software in the second chapter. The third chapter applies the method to three different data sets and one big data set comprised of these three data sets and shows that we can indeed deliver a model with the desired properties. Analysis after freezing the models and unlocking the test data suggest that, for a reliable internal validation and high test performance, (a) data sets with a large number of samples, even if they result from combining somewhat different, partly non-prospective data sets, (b) relying on already-existing molecular signatures rather than fitting new ones and (c) deploying simple, generalized linear models that can handle batch effects play a key role. The fourth chapter discusses how we can develop even better models in the future, especially for MMML-Predict.

We conclude that with currently available data and statistical and computational methods, we can identify a DLBCL subpopulation with poor survival that is larger and holds significantly more high-risk patients than that identified by the IPI. With more data and new and more accurately measured features in the future, we expect to be able to further improve the performance of such models.

Contents

Chapter 1

Introduction

1.1 Diffuse large B-cell lymphoma: treatment and state-of-the-art risk prediction

Diffuse large B-cell lymphoma (DLBCL) is the most common type of B-cell lymphoma in adults, representing almost 30% of all diagnoses of non-Hodgkin's lymphoma. This aggressive and heterogeneous group of lymphoid neoplasms exhibits diverse phenotypic, genetic, and clinical characteristics. The clinical presentation of DLBCL varies significantly, with differences in tumor load and patient performance status, leading to varied outcomes [23].

Despite being an aggressive and fatal disease, DLBCL is highly curable with the application of intensive immunochemotherapy even in the elderly population. The standard treatment for DLBCL has long been immunochemotherapy in the R-CHOP (rituximab, cyclophosphamide, doxorubicin, vincristine, and prednisone) regimen. R-CHOP-like therapies have significantly improved survival rates, with approximately two-thirds of patients achieving a cure. However, the remaining one-third of patients, especially those with relapsed or refractory disease, continue to face poor outcomes and in most cases finally succumb to their disease [12]. This underscores the clinical need for an accurate, robust, affordable, and easy-to-use tool that can identify patients at high risk for treatment failure early in their treatment course, ideally within the first three cycles of induction chemotherapy. Clinicians can guide those patients labeled as high-risk by the tool to alternative treatments like salvage chemotherapy and CAR-T cells.

To this end, the International Prognostic Index for non-Hodgkin's lymphoma (IPI) was established in the 1990s. It incorporates five binary clinical features one can measure cheaply and reliably without batch effects: Is the patient older than 60 years? Is the cancer advanced (Ann Arbor Stage III or IV)? Does the patient have a higher-than-normal lactate dehydrogenase (LDH) level? Is the patient already bedridden (performance status > 1)? Is the patient's cancer in more than one extranodal site? The *IPI score* is then the number of positive answers to these questions, an integer between 0 and 5. Although primitive and arbitrary at first glance, the IPI is the result of a rigorous statistical analysis of a large dataset of 1872 patients: out of twelve candidate features, the IPI inventors first selected those features that were independently and significantly associated with survival (namely the five above-mentioned features), and fit a Cox proportional-hazards model to them; since the relative risks for all five features turned out to be similar, they simplified the model by just counting the number of present risk factors [26].

The IPI is a simple yet robust clinical tool used globally to predict risk and guide treatment decisions in DLBCL patients. It has been the cornerstone of risk assessment for the last three decades, no alternative has gained widespread acceptance outside of clinical trials [30]. Despite its effectiveness in large cohorts, the IPI and other individual biomarkers do not reliably predict the clinical course for each patient, particularly failing to identify those at high risk for early treatment failure who may benefit from alternative therapeutic approaches. E.g., in the prospective trial comprising 466 patients used as test set in [24], only 3.4% of patients have the maximum IPI score of 5 – this proportion is too low to gain special attention in clinical practice and to incentivize the pharmaceutical industry to develop new treatments. All other cohorts defined via IPI $\geq i, i = 0, 1, 2, 3, 4$, lack precision: the proportion of patients with progression-free survival (PFS) less than two years is below 50% – too low to persuade a patient to undergo an experimental treatment instead of a standard, R-CHOP-like therapy.

For this thesis, we define high-risk DLBCL patients as those who face cancer progression within two years after the start of the treatment. Two years is a time frame accepted by both patients and clinicians, which makes roughly a fourth of DLBCL patients high-risk [24]. It is also the threshold used in the MMML-Predict we will introduce next.

1.2 The MMML-Predict project

Renowned lymphoma experts from across academic Germany – clinical trialists, biostatisticians, bioinformaticians, lymphoma pathologists and translational lymphoma biologists – have formed the consortium MMML-Predict to develop and roll out a robust, simple-to-use and cost-effective prognostic tool for DLBCL which yields a clinically more relevant high-risk group. This tool, the MMML-Predictor, will allow patients and clinicians early in the treatment cycle to make an informed decision if they want to continue with the standard R-CHOP-like treatment or switch to a novel, more experimental treatment.

In a discovery phase, MMML-Predict will enroll 200 patients in a prospective trial at first diagnosis and collect a bunch of clinical and molecular risk features. They include clinical scores (like the IPI), gene-expression-based factors (like cell-of-origin and prognostic signatures) and genetic determinants (like MYC, BCL2, BCL6, TP53 and genetic signatures). In a novel approach, MMML-Predict will measure dynamic response determinants during treatment. It is unknown if these features capture similar or different risks. Combining them may finally bring to the patients' bedside the significant progress in the understanding of the DLBCL biology researchers have made over the last decades.

There are plenty of biological terms in the last paragraph and we want hold on for a second to introduce those that will become relevant over the course of this thesis. Many of the above-mentioned features rely on measuring *gene-expression levels* for a set of genes. For a single gene, its expression level reflects the number of RNA transcripts copied from this gene in a tissue, in our case a lymphoma biopsy. Gene-expression levels catch the tissue state on a molecular level. Measuring gene-expression levels for all or at least a high number of the genes that an organism expresses, yields a gene-expression profile. *Cell-of-origin signatures* indicate if the gene-expression profile of a tumor is more similar to that of a germinal-center B-cell (GCB) or an activated B-cell (ABC), which refer to two states of a B cell over its life cycle. Biologists call predictive models that predict from genetic features, especially gene-expression levels, *signatures*. The genes MYC, BCL2, BCL6 and TP53 are known to play a crucial role in DLBCL.

The group around Markus Loeffler in Leipzig plans to evaluate the readily trained MMML-Predictor on a test cohort of another 100 patients enrolled for this project for whom only those features used in the MMML-Predictor will be measured. The new classifier has to label at least 10% of patients in the test cohort as high-risk; stated differently, we require a rate of positive predictions or – synonymously – a precision of at least 10%. At the same time, the precision – i.e. the proportion of *truly* high-risk patients among the patients *labeled* as high-risk – must be *significantly* above that of the group defined by IPI ≥ 4.

In more detail, being high-risk or not is a Bernoullie random variable. The precision is the success probability of this random variable under the condition that the MMML-Predictor predicts a high risk. The test cohort comprises independent, identically distributed (i.i.d) samples. They stay independent under the condition of being labeled high-risk by the MMML-Predictor. The precision on the test cohort is therefore tailored for the Clopper-Pearson method, which calculates exact γ-confidence intervals (CIs) for the success probability of i.i.d. Bernoullie random variables for an arbitrary confidence level $\gamma \in (0, 1)$ [6]. By significantly outperforming the precision of the IPI, we mean that the 95%-CI of the new classifier's precision according to the Clopper-Pearson method must not include the precision of the group determined by IPI ≥ 4. According to data with 2721 patients pooled from prospective trials of the German High-Grade Non-Hodgkin Lymphoma Study Group (DSHNHL), this precision of the IPI is at around 35%.

1.3 The role of this thesis within MMML-Predict

Inside MMML-Predict, Rainer Spang's group in Regensburg will develop the MMML-Predictor. This is a supervised-learning task with a classification problem with binary response: PFS of less than two years – or high-risk DLBCL – is the positive group, PFS of more than two years – or low-risk DLBCL – is the negative group. Part of the task is figuring out which features are part of the model input, the predictor. While the number of samples is rather small, the number of available features is large thanks to high-throughput measurements like RNA-seq or NanoString. We have an enormous amount of freedom in how we design and train the classifier. Hence, we need to deal with the curse of high dimensions if we want to use the high-dimensional part of the data and, more importantly, we need to take care that in our internal validation, usually a cross-validation, we do not overfit the data. While we should be able to tackle the first problem with regularization, for the latter one we need a trustworthy internal validation strategy and, most importantly, we must not validate too many models in the first place.

Since, as of July 2024, the MMML-Predict training cohort has not yet arrived in Regensburg, this thesis will imitate the train-test scenario of MMML-Predict on already-available DLBCL data sets.

This thesis has two main goals.

- First, we want to show that, with data including traditional clinical and modern molecular features, we can indeed deliver the desired model, which yields a larger and more precise high-risk group of DLBCL patients than IPI and does so significantly. With an eye to rolling out the MMML-Predictor in clinical practice, we want to demonstrate that we can design this model in such a way that one can transfer it from one lab protocol to another without seriously comprised performance.
- Second, we want to develop heuristics and recommendations to answer the question which candidate models are worth training and validating and – more importantly – which are not. For this, we need to find out which models we can reliably validate and which perform well on independent test data. These findings will guide us on how to keep the number of candidate models we fit to the MMML-Predict training data low. This helps us avoid overfitting in the internal validation and submitting a model that convinces in the validation, but disappoints on the test cohort in Leipzig.

Chapter 2

Methods

Finding the best possible model for our given task will not be possible from just theoretical considerations; we will have to fit several models to our data and demonstrate the performance of the chosen one convincingly. Section 2.1 will lay out the state-of-the-art train-validate-test paradigm for this. In section 2.2, we will introduce the candidate models and the hyperparameters governing their fitting process. We will start with model-agnostic hyperparameters before we go on to present well-known model classes and finally introduce a method that lets us train compositions of multiple models.

2.1 Training, validation and testing

The design of MMML-Predict with its train and test cohort, where the people developing the predictor never get to see the test cohort, sets the scene for the standard two-step approach in supervised-learning tasks: We use the training cohort to fit multiple models (training) and choose the model among these models that we have confidence performs best on new data (validation). We then touch the test cohort for the first time as we evaluate the chosen model on it to persuade the outside world we have come up with model worth deploying.

Since we want to have the same conditions for data used in this thesis as later in MMML-Predict, we randomly split given data into a train and a test cohort. In more detail, we are initially given a cohort of samples $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$. Every *observation* or *sample* $(\mathbf{x}_i, y_i)$ corresponds to a DLBCL patient: $\mathbf{x}_i \in \mathbb{R}^p$ is a vector of p features, the *predictor*, that we want to use to predict the corresponding *response* $y_i \in \{0, 1\} \cup (\mathbb{R} \times \{0, 1\})$; we will explain in section 2.2 what this odd-looking format for the response encodes, but in this thesis it is always derived from the patient's survival. We summarize the responses to the response vector $y = (y_i)_{i=1,\ldots,n}$ and consider the predictors the row vectors of the predictor matrix $\mathbf{x} = (\mathbf{x}_i)_{i=1,\ldots,n} \in \mathbb{R}^{n \times p}$. We call n the *sample size* of the cohort. We randomly partition $I = \{1, \ldots, n\}$ into two disjoint subsets I_train and I_test and obtain a *training cohort* – often also called *training data* or *training set* – $(\mathbf{x}_\text{train}, y_\text{train}) = (\mathbf{x}_i, y_i)_{i \in I_\text{train}}$ and a *test cohort* – often also called *test data* or *test set* – $(\mathbf{x}_\text{test}, y_\text{test}) = (\mathbf{x}_i, y_i)_{i \in I_\text{test}}$. The following assumes a single, split data set.

2.1.1 Training and validation

To be able to discuss some probabilistic caveats of validation later, we introduce some formal notation. We start with a set of tuples of hyperparameters H, where every $h \in H$ defines a model up to its parameters. Determining the parameters of a model, by definition, is the job of the algorithm optimizing a given loss function; we refer to this as fitting.

Every $h \in H$ specifies a candidate model and there is a one-to-one mapping between H and the set of candidate models.

For every hyperparameter tuple $h \in H$, we

- fit the model to the training cohort subject to h and provide *validated predictions* for every sample in the training cohort. A validated prediction for a sample is a prediction made by a model fit subject to h to data that does not include this sample. We obtain a vector of validated predictions $\hat{y}_{\text{train}} = \text{val}(h)$ with $|I_{\text{train}}|$ entries. The most common method to obtain validated predictions is a k-fold cross validation [25]. This means we randomly assign the training samples into k equally sized subsets, called folds, and then fit k models: for every $i = 1, \ldots, k$, we fit a model to all samples that are *not* in the i-th fold and obtain its predictions for the samples *in* the i-th fold. These predictions serve as (cross-validated) predictions. Doing this for all folds, we obtain a cross-validated prediction for every training sample.
- Afterwards, we calculate the validation error $\text{err}(y_{\text{train}}, \hat{y}_{\text{train}})$. We will lay out our choice of the error function err in subsection 2.1.3. For now, its exact definition does not matter.

Finally, we select the model m^* with minimal validated error defined by the hyperparameter tuple

$$h^* = \operatorname*{arg\,min}_{h \in H} \text{err}(y_{\text{train}}, \text{val}(h)). \tag{2.1}$$

val is not a deterministic mapping. The procedure to calculate validated predictions often involves randomness, e.g. we randomly assign the training samples into folds in a cross validation and many fitting algorithms like that of random forests resort to randomness. The computer does this independently between the $\text{val}(h)$. Therefore, we treat the validation errors $\text{err}(y_{\text{train}}, \text{val}(h))$ as independent real random variables. It is a well known property of independent, identically distributed (i.i.d.) real random variables $X_i, i \in \mathbb{N}$, that their extreme values are notoriously unstable: for all $t \in \mathbb{R}$ with $P(X_1 \geq t) < 1$,

$$P\left(\min_{1 \leq i \leq n} X_i \geq t\right) = P(X_1 \geq t)^n \to 0 \quad \text{as } n \to \infty. \tag{2.2}$$

To avoid this disaster, which is nothing but overfitting the validated predictions to the training data, we need $\text{err}(y_{\text{train}}, \text{val}(h)), h \in H$, with clearly distinct distributions. How can we achieve this this? First, we can choose H in such a way that the fit models are notably distinct. There is, however, a trade-off: the rougher and smaller H is, the more distinct the models and their validated errors are, but the higher the chance is that we miss out on a very good model. Vice versa, the larger and more granular H is, the more likely one of the $h \in H$ specifies a very performant model, but the less likely validation is to reveal this model because, due to near-i.i.d. validation errors in subsets of H, another, actually worse model might sneak in a greatly underestimated validation error. Second, different methods to calculate validated predictions can leverage differences in the distribution of the $\text{err}(y_{\text{train}}, \text{val}(h))$ differently. We will not just use cross validation, but also so-called out-of-bag predictions in the case of random forests in this thesis. Third, the magic bullet of machine learning also helps cure this problem: more training samples will make it harder to overfit the validated predictions.

What do we conclude from this for practice? First, we aim to choose H in a smart and lean way. We do not want to have big subsets of H that specify more or less the same model. This is hard to foresee, but criteria may be theoretical considerations and experience by both us and others like default or automatically generated hyperparameters. Second, once we have fit and validated all models specified in H and tested the chosen one, we can unlock the test data for all other models to analyze the relationship between validated and tested errors. We can look out for subsets of H with validated errors strikingly deviating from tested errors or with high test errors and exclude them from H in the future. Third, we try to increase the number of samples in training and test data by combining data sets. This comes with some caveats and compromises we will deal with later, but we hope that more reliable validation errors and better models will compensate for this.

2.1.2 Testing

We calculate the predictions $m^*(\mathbf{x}_{\text{test},i}) = \hat{y}_{\text{test},i}$, $i = 1, \ldots, |I_{\text{test}}|$, of the best validated model m^* on the test cohort and estimate its performance on independent data via

$$\operatorname{err}(y_{\text{test}}, \hat{y}_{\text{test}}). \tag{2.3}$$

2.1.3 Choice of the error function

As stated in section 1.2, the MMML-Predictor should detect a high-risk group with an as-high-as-possible precision under the constraint that its prevalence must surpass 10%. Since an error should be the lower, the better the model is, we choose err to be the minimum of the negative precisions with a prevalence of at least 17%. We add seven percentage points to the 10% prevalence as it increases statistical power and makes the error function more robust as we will see next.

Usually, we need to take a minimum over *several* precisions because most models do not output the final classification. Instead, they return a continuous score where a higher score means a higher vote of being in the positive class, in our case being high-risk. This continuous score needs *thresholding* via $\hat{y}_i > t$ for some $t \in \mathbb{R}$. We notice that as we increase t and thereby decrease the prevalence, the obtained adjacent precision values fluctuate less and less; for high t and low prevalences, fluctuations of up to 50 percentage points would be possible, but requiring a prevalence of at least 17% caps such fluctuations at

$$\frac{1}{\lceil 0.17 \cdot |I_{\text{test}}| \rceil} \tag{2.4}$$

as an easy calculation shows, cf. also Fig. 3.4 in practice. This gives us an error function that is tailored for our problem and robust.

As indicated by the notation $\operatorname{err}(y_{\text{train}}, \hat{y}_{\text{train}})$ and $\operatorname{err}(y_{\text{test}}, \hat{y}_{\text{test}})$, we optimize t on the true outcomes of the training cohort when validating and we optimize it again on the true outcomes of the test cohort when testing. Strictly speaking, t is a hyperparameter, which we optimize on the test cohort during testing; this sounds delicate. The results however will show that the optimal choice for t (or at least an almost optimal choice) on both train and test cohort corresponds to a prevalence of slightly above 17% such that one can agree on the following when it will come to the MMML-Predict data: choose t as the 17%-quantile of the continuous model output on the test cohort.

Similarly, given some cohort, we threshold the IPI in such a way that it maximizes the precision under the constraint of a prevalence of at least 10% on this cohort and call this truly binary classifier thresholded IPI, in short tIPI. Usually this amounts to thresholding the IPI via IPI ≥ 4. The IPI only needs to deliver the minimum attention-triggering prevalence of 10% while its challengers, our models, need a somewhat higher prevalence to gain statistical power against the IPI as explained before. Also optimizing the IPI's threshold for a given cohort ensures a fair competition.

In general, given data $(\mathbf{x}, y)$ of sample size n, we denote the prevalence of a model m corresponding to $\operatorname{err}(y, m(\mathbf{x}_i)_{i=1,\ldots,n})$ by $\operatorname{prev}_{\mathbf{x},y}(m)$ and the corresponding precision by $\operatorname{prec}_{\mathbf{x},y}(m) = -\operatorname{err}(y, m(\mathbf{x}_i)_{i=1,\ldots,n})$. When it is clear from the context which data set or subset of a data set we are talking about, we just write $\operatorname{prev}(m)$ and $\operatorname{prec}(m)$.

2.1.4 Inter-technical variability

One could instead suggest to already optimize the output threshold of our models on the cross-validated predictions. While this promises a stricter train-test regime at first glance, fixing the output threshold once and for all neglects *inter-technical variability*.

An always-present problem in bioinformatics, inter-technical variability refers to the fact that one and the same sample measured on different platforms, in different labs or at different times, in short: under different protocols, may lead to different values for the same feature. Values for the same feature and sample measured in different batches, even if the lab, preparation and technology are the same, may differ. We will therefore also use the term batch effects for inter-technical variability. Every data set adheres to its own protocol, so we need to deal with inter-technical variability whenever we combine data sets or have a model predict on a data set other than the one it was trained on.

If we measure the same p features of the n patients in $\mathbf{x} \in \mathbb{R}^{n \times p}$ again under another protocol, we end up with another predictor $\mathbf{z} \in \mathbb{R}^{n \times p}$. For logarithmized gene-expression levels, we can often well model the discrepancies with two independent biases, sample-wise effects θ_i and feature-wise effects ω_j, leading to

$$\Delta_{ij} = \mathbf{z}_{ij} - \mathbf{x}_{ij} = \theta_i + \omega_j + \epsilon_{ij} \tag{2.5}$$

for residues ϵ_{ij}. Assuming accurate modeling, i.e. small residues, we can well approximate

$$\mathbf{z}_{ij} \approx \tilde{\mathbf{z}}_{ij} = \mathbf{x}_{ij} + \theta_i + \omega_j. \tag{2.6}$$

We can now apply an ordinary linear model with parameters (β_0, β) to $\mathbf{z}$ and obtain

$$\begin{aligned} \beta_0 + \sum_{j=1}^{p} \beta_j \mathbf{z}_{ij} &\approx \beta_0 + \sum_{j=1}^{p} \beta_j \tilde{\mathbf{z}}_{ij} \\ &= \beta_0 + \sum_{j=1}^{p} \beta_j \mathbf{x}_{ij} + \sum_{j=1}^{p} \beta_j \theta_i + \sum_{j=1}^{p} \beta_j \omega_j. \end{aligned} \tag{2.7}$$

The third summand is zero under the zero-sum constraint $\sum_{j=1}^{p} \beta_j = 0$. The fourth summand is constant across all samples and can be absorbed by the intercept, cf. [2].

Under these assumptions, which are pretty realistic, going from one protocol to another just leads to a constant shift of the model output; even for generalized linear models, i.e. a Gauss model composed with some monotonic link function, inter-technical variability does not change the ordering of the samples. All one needs to do to obtain a final classification is calibrating the threshold for the output scores. However, this also demonstrates that while generalized linear models can cope with inter-technical variability pretty well, there is no point in using the same output threshold across all protocols.

2.2 Candidate models

2.2.1 Model-agnostic hyperparameters

Model-agnostic hyperparameters are those hyperparameters we can set and tune for every model. They concern the predictor matrix $\mathbf{x} \in \mathbb{R}^{n \times p}$ and the response vector $y \in (\{0, 1\} \cup (\mathbb{R} \times \{0, 1\}))^n$. The union in the latter set reflects the fact that can provide the response in two formats.

- In the first case, the response is a binary vector with 1 encoding the positive class, high-risk DLBCL, and 0 encoding the negative class, low-risk DLBCL. We call this the *binary format*. Here, we need to discard samples censored before two years, but this is typically only a small proportion of the samples in the data set.

- In the second case, the response has two entries per sample: the *time to the event* and *censoring status*. If the censoring status is 1, the event – in our case cancer progression – occurs *at* the time to the event. If the censoring status is 0, we only know that the event will occurs at some point *after* the time to the event. We call this the *Cox format*. More on this in subsection 2.2.2 when we present the Cox model.

Which of these two formats we provide for a sample depends on the model class. Our choice of err ultimately needs the true response y in the binary format, which err can easily derive from y in the Cox format if necessary.

A-priori feature selection

Not every feature in the data set should be part of the predictor matrix $\mathbf{x}$. Including features measured toward the end of a patient's therapy or even after it, is cheating and we should definitely exclude them. Excluding any of the remaining features is hard to justify: If we know for a feature that it alone is associated with the response, we include it in $\mathbf{x}$ without doubt. On the other hand, if a feature alone shows no link to the response, the right model might still be able to leverage it in combination with other features; but at this step, we do not know if such a model exists and if it is among our candidate models. The only remaining option for handpicking features a priori is to brute-force the problem and try out all 2^p combinations of features. Even if we only do this for part of the features, say 10 features, this would still leave us with $2^{10} = 1024$ combinations to try out – just for this hyperparameter. While this may be computationally feasible, validation would be a statistical fiasco as laid out in subsection 2.1.1. Therefore, we reduce this quickly exploding number of choices to only a handful and let *regularization* handle the large number of features in $\mathbf{x}$. More on this in subsection 2.2.3.

Gene-expression data The first and most important decision to make is: do we want to include the high-dimensional gene-expression part of the data at all? If we do not, we suddenly have $p \ll n$ instead of $p > n$ or even $p \gg n$. This strongly effects the models we fit, hence also H and validation.

Features in different formats Second, we need to make a couple of more decisions regarding the format of some features.

- Concerning the IPI, we can include the five features involved in the IPI in their original format, i.e. age and LDH level as a continuous feature, Ann Arbor stage and performance status as a categorical feature and the number of extranodal sites as a discrete feature. We can also binarize them with the same thresholds as the IPI inventors and include five binary features. We call this the *thresholded format of the IPI features*. Similarly, widely accepted thresholds may exist for other clinical and genetic features.

- For already existing models, we can either include their continuous output or threshold it. For the IPI, we can include its output as a single continuous feature. We can also include the so-called IPI group, a partitioning of the IPI output into three groups, which one obtains by thresholding it at 1.5 and 3.5. As another example, cell-of-origin signatures, whose output is continuous to begin with, can be thresholded into ABC, GCB and unclassified subtypes. Data sets often only provide this grouping and not the raw, continuous output.

E.g., imagine a data set provides the five IPI features in their original format, from which we can easily infer all other mentioned formats, and the continuous output of some gene-expression signature for which its inventors also provide a preferred threshold for a binary grouping. If we want to decide for exactly one format per feature, this leaves us with $4 \cdot 2$ possibilities for this simple case. If we want try out every combination of features in any format, the number of possibilities jumps to $2^4 \cdot 2^2 = 64$.

Our solution here again is to be generous and include all formats of a feature in **x** a certain model may benefit from. This sentence usually simplifies to: include all widely used formats of a feature in **x**. E.g., generalized linear models (more in cf. 2.2.2) cannot threshold continuous features, so a feature in its thresholded format may give a rough contribution to the prediction while the same feature in its continuous form may further refine it. Moreover, even decision trees can benefit from a feature additionally provided in its thresholded format if this threshold has been inferred on a much bigger data set since the decision tree may not be able to find the threshold itself.

Imputation

If values in **x** are missing, i.e. written as NA, actions we can take fall into two categories: we discard the part of data affected by missing values or we replace the missing values with some realistic estimate.

Discarding part of the data In a first step, we discard a feature if it is not available for a large proportion of the samples. This enables reasonable computing.

Imputing At this point, we dichotomize a categorical feature with c categories into $c - 1$ binary dummy features; if for a sample this categorical feature is not available, all $c - 1$ dummy features will be NA. This yields a modified **x** that bears exactly the same information as before. For *every* column in **x**, we calculate the mean of the available features in **x** and replace the NA values with it. For a missing categorical feature, every dummy feature then holds the marginal Bernoullie probability that the feature is in the k-th category. This is efficient, transparent and easily applicable to new data.

Adding combinations of categorical features to the predictor

Next, we add all combinations of at most n_{combi} categorical features that are positive in a share of at least s_{min} patients to **x**; e.g., we add a column "female & ABC-type tumor" if at least 5% of patients have this property. We always choose $s_{\text{min}} = 5\%$. We set $n_{\text{combi}} = 1$ for models that facilitate interactions between features themselves, otherwise we set $n_{\text{combi}} = 3$.

Technically, we need to combine the binary features we derived from distinct categorical features in the previous step. We realize this by multiplying the corresponding columns in **x** element-wise. We thereby keep treating the value of a binary feature as a Bernoullie probability under the assumption that the binary dummy features involved in the combination are independent, which is not always justified. The average of a combined (Bernoullie) feature over all samples, for which we apply the threshold s_{min}, therefore approximates the expected value of the combined feature.

Tuning the definion of "high risk"

Defining a patient as high-risk if and only if the PFS is less than two years is a clinically accepted, yet quite arbitrary decision. It may not be the time cutoff that separates PFS best from a biological point of view. Therefore, we can provide the fitting algorithm a refined response y, governed by the *training survival cutoff* $T \in \mathbb{R}_{>0} \cup \{+\infty\}$.

- For y in the binary format, we set $y_i = 1$ if the PFS of patient i is less than T and $y_i = 0$ otherwise. The higher T is, the more samples we need to discard due to censoring.

- For y in the Cox format, we censor all samples with time to the event exceeding T at T.

Patients may separate much more clearly for some $T \neq 2$ and as long as we come up with a positive cohort that comprises at least 15% of the samples of the test data, this allows us to fulfill our task more easily. We stress that T only influences how we train models. For validation and testing, a PFS of 2 years stays the crucial cutoff.

2.2.2 Core models

All models trained, validated and tested in this thesis at the core consist of ordinary linear, logistic and Cox proportional-hazards models with additional properties, as well as random forests. In this chapter, we want to introduce the design of these models and the hyperparameters governing their fitting process.

Formally, we deal with a probability space $(\Omega, \mathcal{A}, P)$ that represents the whole population of DLBCL patients. We do not and cannot specify this probability space any further because we only get in touch with two random variables

$$X = (X_1, \dots, X_p) : \Omega \to \mathbb{R}^p \text{ and } Y : \Omega \to \mathbb{R}, \tag{2.8}$$

the predictor and the response, respectively. More precisely, we observe independent training samples $(\mathbf{x}_i, y_i) \in \mathbb{R}^{p+1}, i = 1, \dots, n$, distributed according to (X, Y); $\mathbf{x}_i$ is the i-th row of the predictor matrix $\mathbf{x}$ and y_i is the i-th entry of the response vector y. The i.i.d. test samples follow the same distribution as the training samples, but in this subsection everything is about training. If we say samples in this subsection, we always refer to the training samples.

Generalized linear models

Here, we work with binary response, i.e. $\operatorname{im}(Y) \subset \{0, 1\}$. Both ordinary linear models and logistic regression models are generalized linear models (GLMs). In a GLM, Y follows an *overdispersed exponential family of distributions* and there is an invertible *link function* $g : \mathbb{R} \to \mathbb{R}$ and parameters $(\tilde{\beta}_0, \tilde{\beta}) \in \mathbb{R}^{p+1}$ such that

$$g(E(Y \mid X = x)) = \tilde{\beta}_0 + \sum_{j=1}^{p} \tilde{\beta}_j x_j \quad \text{for all } x \in \operatorname{im}(X) \subset \mathbb{R}^p. \tag{2.9}$$

where $E(Y \mid X = x)$ is the expected value of Y given $X = x$ [17]. We can express the *linear predictor* $\sum_{j=1}^{p} \tilde{\beta}_j x_j$ in terms of the Euclidean scalar product as $x^T \tilde{\beta}$, where x^T denotes the transpose of the column vector x. We will only introduce the two special cases of an overdispersed exponential family of distributions that matter to us in this thesis below. For a general definition, see [17, section 1.1].

In practice, we do not know the (true) parameters $(\tilde{\beta}_0, \tilde{\beta})$. But we do have our n i.i.d. observations $(\mathbf{x}_i, y_i)$ and we will use them together with a likelihood to estimate $(\tilde{\beta}_0, \tilde{\beta})$. For an arbitrary $(\beta_0, \beta) \in \mathbb{R}^{p+1}$, we can try to predict the conditional expectation as

$$\mu_i(\beta_0, \beta) = g^{-1}(\beta_0 + \mathbf{x}_i^T \beta), \quad i = 1, \dots, n. \tag{2.10}$$

We have $\mu_i(\tilde{\beta}_0, \tilde{\beta}) = E(Y \mid X = \mathbf{x}_i)$.

Gauss model Here, $g = \text{id}$. Y and $Y \mid X = x$ for all $x \in \mathbb{R}^p$ follow a normal distribution with fixed standard deviation $\sigma > 0$ (a property called homoscedasticity), but varying mean. For a normal distribution with standard deviation σ, the log likelihood of an expected value – or equivalently mean – μ given an observation $y \in \mathbb{R}$ is

$$\begin{aligned}\ell(\mu; y) &= \log\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right)\right)\\ &= -\frac{1}{2\sigma^2}(y-\mu)^2 - \log\left(\sqrt{2\pi}\sigma\right).\end{aligned} \tag{2.11}$$

Up to a constant shift and re-scaling, which does not affect maximizing the log likelihood, this is the well-known squared error.

Logistic model Here, the linear predictor is the log-odds for $Y = 1$ over $Y = 0$ given $X = x$, i.e.

$$g : \mu \mapsto \log\left(\frac{\mu}{1-\mu}\right), \text{ hence } g^{-1} : \eta \mapsto \frac{1}{1+\exp(-\eta)}. \tag{2.12}$$

Y and $Y \mid X = x$ for all $x \in \mathbb{R}^p$ follow a Bernoulli distribution with varying success probability. For the Bernoullie distribution, the log likelihood of an expected value – or equivalently success probability – μ given an observation $y \in \{0, 1\}$ is

$$\ell(\mu; y) = \log\left(\mu^y(1-\mu)^{1-y}\right) = y\log(\mu) + (1-y)\log(1-\mu). \tag{2.13}$$

Loss function Using Eq. (2.10), the log likelihood for (β_0, β) given our i.i.d. samples $(\mathbf{x}_i, y_i)$, $i = 1, \dots, n$, for both the Gauss and logistic model therefore is

$$L(\beta_0, \beta) = \sum_{i=1}^{n} \ell(\mu_i; y_i) = \sum_{i=1}^{n} \ell\left(g^{-1}\left(\beta_0 + \mathbf{x}_i^T\beta\right); y_i\right). \tag{2.14}$$

We can augment L with two hyperparameters, sample weights $w_i > 0$, $i = 1, \dots, n$, and the zero-sum constraint with respect to zero-sum weights $u_j \geq 0$, $j = 1, \dots, p$, yielding

$$L(\beta_0, \beta) = \sum_{i=1}^{n} w_i \ell\left(g^{-1}\left(\beta_0 + \mathbf{x}_i^T\beta\right); y_i\right) \quad \text{for } \beta \in \mathbb{R}^p \text{ with } u^T\beta = 0. \tag{2.15}$$

While we do not deviate from the default value $1/n$ for the sample weights in the results chapter 3, we will discuss in chapter 4 how we can use them in the future in a natural way without getting lost in the vast amount of freedom that one has in choosing them. With these augmentations, L is no longer a log likelihood, so we call it a *loss function*.

Fitting a Gauss or logistic model to the data means minimizing L with respect to (β_0, β) and coming up with a minimizer $(\hat{\beta}_0, \hat{\beta})$, which is our estimate for $(\tilde{\beta}_0, \tilde{\beta})$. Subsection 2.2.3 will deal with the question how we can make $(\hat{\beta}_0, \hat{\beta})$ a *unique* minimizer.

We want to demonstrate what the zero-sum constraint is good for in a more general setting. Imagine, part of the features suffer from sample-wise shifts. We denote these features by $J \subset \{1, \dots, p\}$ and the shifts by $s_i \in R$, $i = 1, \dots, n$. We now set the zero-sum weights

$$u_j = \begin{cases} c & \text{if } j \in J, \\ 0 & \text{else,} \end{cases} \tag{2.16}$$

for some $c > 0$ (usually $c = 1$); we can express this conveniently with the help of the indicator function χ_J as $u_j = \chi_J(j)c$. By the zero-sum constraint, we demand $\sum_{j \in J} c\beta_j =$

0 and hence $\sum_{j\in J}\beta_j = 0$; apart from that, (β_0, β) is arbitrary. In this situation, the linear predictor

$$\beta_0 + \sum_{j=1}^{p} \beta_j(\mathbf{x}_{ij} + s_i\chi_J(j)) = \beta_0 + \mathbf{x}_i^T\beta + \sum_{j\in J}\beta_j s_i = \beta_0 + \mathbf{x}_i^T\beta, \tag{2.17}$$

as $\sum_{j\in J}\beta_j s_i = s_i\sum_{j\in J}\beta_j = 0$, is invariant. If J consists of exactly the features holding gene-expression levels, we can transfer our model to other protocols and only need to re-calibrate the intercept, as shown in Eq. (2.7). Moreover, we can switch off the zero-sum constraint for all features by setting $u_j = 0$ for all $j = 1, \ldots, p$. The zero-sum constraint is cheap in terms of model complexity: it only removes one degree of freedom; but it is expensive in terms of computational complexity: enforcing it in every step of the coordinate descent leads to a considerably higher computation time.

As we have stressed several times already, the threshold that makes a GLM a binary classifier is rather a property of the underlying data set than of the model itself. The essential information the modelling function contributes to the classification of a cohort of samples is therefore the ordering of the output scores. Because g and therefore also g^{-1} are monotonically increasing, the essential part of the model is the linear predictor $\mathbf{x}_i^T\beta$. For this reason, when we talk about the output of a GLM, we always mean the output of the linear predictor. For us, this makes g more a tweak in the loss function, important for training, than a part of the modelling function.

We are not done, yet, with L: at the end of the day, a coordinate descent minimizes Eq. (2.15) with a regularization term added. Before we discuss this, we want to introduce a model that is closely related to GLMs.

Cox proportional-hazards model

For the Cox proportional-hazards model – in short: Cox model – the response variable Y measures the time *until* the event of interest occurs ("event") or the time *after* which the event occurs ("censoring"). Another binary random variable $\delta : \Omega \to \mathbb{R}$ encodes which of these two options is the case: $\delta(\omega) = 1$ for the event, $\delta(\omega) = 0$ for censoring. This pays tribute to an important characteristic of survival trials: some participants drop out of the trial before the event could happen – e.g. the trial terminated, the patient decided to withdraw or died from another cause – or the event luckily never happens.

Hazard function To understand what the Cox model predicts, we need another random variable $T : \Omega \to \mathbb{R}$, the time to the event. Unlike Y, T is not affected by censoring. We require T to have a density $f_T : \mathbb{R} \to \mathbb{R}$ and define its hazard function $h : \mathbb{R} \to \mathbb{R}$ via

$$h(t) = \lim_{\Delta t\to 0} \frac{P(t \le T < t + \Delta t \,|\, T \ge t)}{\Delta t}. \tag{2.18}$$

We can interpret $h(t)$ as the instantaneous rate at which the event is occurring at time t, given that the event has not occurred before time t. Let F_T be the distribution function of T. Then we can write

$$\frac{P(t \le T < t + \Delta t \,|\, T \ge t)}{\Delta t} = \frac{F_T(t + \Delta t) - F_T(t)}{\Delta t \cdot (1 - F_T(t))}. \tag{2.19}$$

Hence,

$$h(t) = \lim_{\Delta t\to 0} \frac{F_T(t + \Delta t) - F_T(t)}{\Delta t \cdot (1 - F_T(t))} = \frac{f_T(t)}{1 - F_T(t)} = \frac{f_T(t)}{S_T(t)}, \tag{2.20}$$

where $S(t) = 1 - F_T(t)$ is the survival function, and $h(t)$ is well-defined as long as $S_T(t) > 0$.

Conditional hazard The Cox model, proposed in [7], wants to get a hand on *conditional* hazards. Intuitively, it is clear that the population with one value of X can have a vastly different hazard function than the population with another value of X if X has predictive power for T. Formally, however, it is hard to define a conditional hazard function: X may very well follow a continuous distribution, hence we condition on an event of probability 0 when we write $P(t \leq T < t + \Delta t \,|\, T \geq t, X = x)$ – it is hard to define this in a natural and straightforward way. In practice, the measurements in X often fluctuate around their true values anyway, so it is sensible to condition on a small neighborhood of x, say an ϵ-ball around x with respect to the Euclidean norm $|\cdot|_2$ for some small $\epsilon > 0$. This amounts to

$$h(t \,|\, X = x) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \,|\, T \geq t, |X - x|_2 < \epsilon)}{\Delta t} \tag{2.21}$$

or – equivalently – $h(t \,|\, X = x)$ is the hazard of T restricted to $(\Omega, \mathcal{A}, P)$ conditioned on $\Omega_x = \{\omega \in \Omega : |X(\omega) - x| < \epsilon\}$; unlike before, we usually now have $P(\Omega_x) > 0$.

Proportional hazards The Cox model assumes *proportional* conditional hazards, i.e.

$$h(t \,|\, X = x) = \lambda_x h_0(t) \tag{2.22}$$

for all $x \in \text{im}(X)$ and some (unspecified) baseline hazard h_0 and $\lambda_x > 0$. Moreover, it assumes λ_x depends on x via

$$\lambda_x = \exp(x^T \tilde{\beta}), \tag{2.23}$$

for parameters $\tilde{\beta} \in \mathbb{R}^p$.

Partial log likelihood Given our independent samples $(\mathbf{x}_i, y_i, \delta_i) \in \mathbb{R}^{p+2}$, $i = 1, \ldots, n$, distributed according to (X, Y, δ), how can we estimate $\tilde{\beta}$? The Cox model is a semi-parametric model in that it leaves the baseline hazard h_0 unspecified. Consequently, estimation with maximum likelihood is not possible. This led to another break-through invention in [7], the *partial* likelihood, which we will present now. Let $\beta \in \mathbb{R}^n$ be arbitrary and let $E \subset \{1, \ldots, n\}$ refer to the non-censored samples ($\delta_i = 1$). For $i \in E$, let R_i denote the set of samples where no event has occurred until y_i, i.e. $y_j \geq y_i$ for all $j \in R_i$. For fixed $i \in E$ and time to event y_i, conditionally on the risk set R_i, the probability that the event occurs on sample i as observed is

$$\frac{h(y_i \,|\, X = \mathbf{x}_i)}{\sum_{j \in R_i} h(y_i \,|\, X = \mathbf{x}_j)} = \frac{\exp(\mathbf{x}_i^T \beta) h_0(y_i)}{\sum_{j \in R_i} \exp(\mathbf{x}_j^T \beta) h_0(y_i)} = \frac{\exp(\mathbf{x}_i^T \beta)}{\sum_{j \in R_i} \exp(\mathbf{x}_j^T \beta)} \tag{2.24}$$

according to [7, Eq. (12)].

Taking all events into account yields the partial likelihood

$$\tilde{\ell}(\beta) = \prod_{i \in E} \frac{\exp(\mathbf{x}_i^T \beta)}{\sum_{j \in R_i} \exp(\mathbf{x}_j^T \beta)}. \tag{2.25}$$

Because T is continuous, there is a zero probability for two distinct samples having the same event time, so we need not resort to one of the more sophisticated likelihoods capable of handling ties. Analogously to Eq. (2.15), we can equip the partial log likelihood with sample weights $w_i > 0$ and zero-sum weights $u_j \geq 0$ such that the refined log partial likelihood, which we now call a loss function, reads

$$L(\beta) = \sum_{i \in E} w_i \mathbf{x}_i^T \beta - \log \left(\sum_{j \in R_i} w_j \exp(\mathbf{x}_j^T \beta) \right) \quad \text{for } \beta \in \mathbb{R}^p \text{ with } u^T \beta = 0. \tag{2.26}$$

As with GLMs, we can leverage the zero-sum constraint to reach invariance under sample-wise shifts. Our estimate for $\tilde{\beta}$ is a minimizer of L with respect to β, denoted by $\hat{\beta}$. More on its unique existence in subsection 2.2.3.

There are more analogies to GLMs. In Eq. (2.23), we also compose the linear predictor with an increasing function. Given a cohort of samples, we are still most interested in the ordering of the model output and the linear predictor $\mathbf{x}_i^T \beta$ already uniquely determines this ordering. Again, when we talk about the output of a Cox model, we mean the output of the linear predictor. In light of all these similarities, in this thesis we refer to both GLMs in the typical sense and Cox models as GLMs. Looking closely, the Cox model gets along without an intercept. We will therefore always ignore β_0 if mentioned in the context of a Cox model in the following.

This paves the way to define some special cases of the zero-sum constraint for the three model classes. We say that a subset of the coefficients of a GLM indexed by $J \subset \{1, \ldots, p\}$ fulfills the *zero-sum property* if $\sum_{j \in J} \beta_j = 0$. If $J = \{1, \ldots, p\}$, we say that the GLM fulfills the zero-sum property. If, additionally, the GLM is a signature, we call it a *zero-sum signature*. By Eq. 2.17, being a *zero-sum* gene-expression signature can be a crucial advantage over just being gene-expression signature.

Hazard ratio We can use the Cox model to calculate a *hazard ratio* (HR) between two groups of samples. Suppose $g_i \in \{0, 1\}$ encodes this grouping. In our case, $g_i = f(\mathbf{x}_i)$ is always the output of a binary classifier f. We now fit a univariate Cox model to (g_i, y_i, δ_i), $i = 1, \ldots, n$, by maximizing the partial likelihood in Eq. (2.25) with respect to β and obtain an estimate for the (scalar) model coefficient $\hat{\beta}$. By Eq. (2.23), $\exp(\hat{\beta})$ is the relative risk increase of being in group $g_i = 1$ compared to group $g_i = 0$ and we call it hazard ratio.

We can view the maximum partial-likelihood estimate $\hat{\beta}$ as a random variable that depends on i.i.d. sampling n training samples. In analogy with maximum-likelihood estimation, there are good arguments to believe that the maximum *partial* log-likelihood estimator $\hat{\beta}$ is

- *consistent*, i.e., with growing sample size n, $\hat{\beta}$ converges in probability to the true coefficient of the Cox model $\tilde{\beta}$, and
- *asymptotically normal*, i.e., with growing sample size, $\hat{\beta}$ converges in distribution to a normal distribution with mean $\tilde{\beta}$ and a standard deviation we can estimate with the help of the second derivative of the partial log likelihood with respect to β evaluated at $\hat{\beta}$ [15, 8.1–8.4].

This justifies to treat $\hat{\beta}$ as normal with known variance and allows us to calculate confidence intervals for the HR $\exp(\tilde{\beta})$ – the mean of this normal distribution – as well as a two-sided p-value for the null-hypothesis $\exp(\tilde{\beta}) = 1$, i.e. identical risk between the two groups.

2.2.3 Elastic-net regularization

Modern molecular measurements are a big factor in our hope that MMML-Predict can succeed and improve the IPI. These measurements, like gene-expression levels, show up as hundreds if not thousands of features in our data, meaning we usually have $p > n$ if not $p \gg n$. In this situation, we cannot uniquely determine the parameters $(\hat{\beta}_0, \hat{\beta})$ by minimizing Eq. (2.15) or Eq. (2.26): the models have way more degrees of freedom than we have samples, they will find biologically meaningless and non-reproducible structures in the training data and generalize poorly to new data.

A solution to this is to restrict the freedom of the model to the extent that it is forced to learn the gist from the data. For that reason, we want to penalize model complexity. Elastic-net regularization, a widely adopted method proposed by Zou and Hastie [31], does this job for us. The elastic net generalizes two well-established regularization methods,

- ridge regularization [14], which forces the parameter vector β to stay inside a ball around the origin by limiting the squared ℓ_2 norm of the parameters, and
- LASSO (least absolute shrinkage and selection operator) regularization [28], which forces β to stay inside a diamond centered at the origin by limiting the ℓ_1 norm of the parameters.

It does so by combining them with a weight factor $\alpha \in [0, 1]$ and feature-wise penalty weights $v \in \mathbb{R}^p_{\geq 0}$ into

$$p_{\text{enet},\alpha,v}(\beta) = \sum_{j=1}^{p} v_j \left(\frac{1-\alpha}{2} \beta_j^2 + \alpha |\beta_j| \right). \tag{2.27}$$

Multiplied with the regularization strength $\lambda \in \mathbb{R}_{\geq 0}$, we can add it to the loss function L from Eq. (2.15) and Eq. (2.26) to obtain the regularized loss function in its Lagrangian form as

$$\mathcal{L}(\beta_0, \beta) = -L(\beta_0, \beta) + \lambda p_{\text{enet},\alpha,v}(\beta) \quad \text{for } \beta \in \mathbb{R}^p \text{ with } u^T \beta = 0. \tag{2.28}$$

In all three cases – Gauss, logistic and Cox model – it is possible to approximately minimize $\mathcal{L}$ with respect to (β_0, β) with a coordinate-descent algorithm [1, section 2.3]. We have finally arrived at the loss function we will optimize to fit models in chapter 3.

The default value for the penalty weights is $v_j = 1$ for all $i = 1, \dots, p$. Deviating from the default can be useful in at least two cases. First, sometimes we are so convinced of the predictive power of a feature in our model that we set $v_j = 0$ for it to all but ensure a non-zero coefficient for it. The second use case is *standardizing the predictor matrix*, which we will deal with further below.

Effect on correlated features While both ridge and LASSO regularization shrink the coefficients, they act differently in the case of correlated features. Ridge regression tends to force the coefficients of correlated features to similar values while the LASSO tends to pick one of them and set the others to zero. E.g, in the extreme case of k identical features, the LASSO arbitrarily picks one of them with, say, coefficient a; the ridge regression, meanwhile, assigns every of the k features the coefficient a/k. Relying on very few features comes with the advantage of sparse models, for which we can cheaply generate new data, but it is very sensitive to measurement errors in one of the picked features; in the latter case, ridge regularization would average out errors with the help of other correlated features. Elastic-net regularization balances the sparsity of the LASSO with the robustness of the ridge regularization, especially for values of α close to 1 [31].

Standardizing the predictor matrix The second application of the penalty weights is an aspect of a bigger topic: standardizing the predictor matrix. We can use the weights to standardize $\mathbf{x}$ by setting v_j to an estimate of the standard deviation of feature X_j. Why should we do this? Let us consider two features X_1 and X_2 with standard deviation σ_1 and σ_2, respectively, and $\sigma_1 < \sigma_2$. To change the output of the model by 1 when the feature changes by one standard deviation and all other features keep their values, X_1 needs the coefficient $\beta_1 = 1/\sigma_1$ and X_2 needs $\beta_2 = 1/\sigma_2$. We have $\beta_1 > \beta_2$. For the default penalty weights $v_1 = v_2 = 1$, the elastic net penalizes X_1, the feature with a smaller standard deviation, which in turn needs a higher coefficient, more. For α close to 1, setting $v_1 = \sigma_1, v_2 = \sigma_2$, we approximately offset this difference. Standardization strives to guarantee equal justice for features on different scales.

Standardization works, however, contrary to the zero-sum idea. Thanks to the zero-sum constraint, the fitting process and the final model are invariant under sample-wise shifts on a subset of features if we set the zero-sum weights accordingly (cf. Eq. (2.17)).

Sample-wise shifts change the standard deviation of the affected features, which, with standardization, affects the cost function and likely leads to another model. Consequently, when deploying the zero-sum constraint in some way, one should stick to the default value $v_j = 1$ for the features for which we wish to have shift-invariance; these are usually the gene-expression levels, which are on comparable scales anyway.

Choosing a λ sequence As we increase the regularization strength λ, the set of minimizers for Eq. (2.28) becomes smaller and smaller. For λ large enough, there exists a unique, trivial solution $(\hat{\beta}_0, \hat{\beta})$ with $\hat{\beta} = 0$. For the above-mentioned coordinate descent, given training data, we can approximate the smallest λ that still yields a trivial solution, denoted by λ_{max} [20, section 3.3]. Following [9, section 2.5], we choose a minimum value $\lambda_{\text{min}} = \epsilon\lambda_{\text{max}}$ for a small $\epsilon > 0$ and construct a sequence of n_λ values decreasing from λ_{max} to λ_{min} equidistantly on the log scale. Typical values are $\epsilon = 0.001$ and $n_\lambda = 100$, and we will stick to them throughout chapter 3. This allows us to tune λ in a rather natural way.

We minimize Eq. (2.28) for λ from this sequence in decreasing order. This gives rise to two tweaks to tune λ efficiently. First, we expect the minimizers of two adjacent λ values to be quite close and therefore use the minimizer from the previous run of the iterative coordinate descent as the starting value for its next run in a so-called *warm start*. Second, we *stop early*, i.e., we fit no more models for the remaining λ values at the lower end of the sequence, if the cross-validated error does not improve for a certain number of consecutive λ values. In chapter 3, we will use a typical value of 10 for this.

Random forests

More precisely, we talk about classification random forests (RFs), hence Y is binary in this subsection. To simplify notation, -1 instead of 0 encodes the negative class of the response, hence Y maps to $\text{im}(Y) \subset \{\pm 1\}$. When we say random forests in this thesis, we mean classification random forests. Random forests, introduced in [4], are an ensemble of classification trees – in short: trees – and aggregate the classification of every constituent tree.

Trees A tree is a simple function

$$T = \sum_{m=1}^{M} c_m \chi_{R_m} : \mathbb{R}^p \to \{\pm 1\} \tag{2.29}$$

with $c_m \in \{\pm 1\}$ and the R_m being disjoint rectangle sets, i.e. $R_m = \prod_{j=1}^{p} (a_j, b_j]$ for $a_j, b_j \in \mathbb{R} \cup \{\pm\infty\}$. By means of such a tree, we want to infer the conditional majority class of the response Y from the predictor X. In the perfect case, we have

$$\text{sgn}(E(Y \mid X = x)) = \tilde{T}(x) \quad \text{for all } x \in \text{im}(X) \tag{2.30}$$

for a tree $\tilde{T}$. sgn is the signum function.

How can we use our samples $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, to learn a tree T that does good job on Eq. (2.30)? While we can trivially fulfill Eq. (2.30) exactly for our samples, this would only lead to heavily overfitted trees. Instead, we confine ourselves with simpler, rougher trees that will make errors on training samples, but generalize better. In the algorithm further below, we will only need to calculate an error for the samples inside a rectangle set $R \subset \mathbb{R}^p$. Let $n(R) = |\{1 \leq i \leq n : \mathbf{x}_i \in R\}|$ be the number of samples in R and

$$\hat{p}_R = \frac{|\{1 \leq i \leq n : \mathbf{x}_i \in R, y_i = 1\}|}{n(R)} \tag{2.31}$$

be the proportion of samples in R with positive response. The error measure – in the context of trees usually termed impurity measure – we use in this thesis is the *Gini impurity* $Q(R) = 2\hat{p}_R(1 - \hat{p}_R)$. It is low for pure R dominated by samples with either positive or negative response and grows quadratically as the response of the samples in R gets more and more imbalanced. The impurity measure is a hyperparameter: other options include the misclassification error and the cross entropy. To govern the complexity of T, we demand that every rectangle set R_m contain at least n_{min} samples. Even this constraint still leaves us with a computationally infeasible number of possible trees, and this is why we fit T with a greedy algorithm. We recursively split every rectangle set along a feature into two sub-rectangles in such a way that the impurity measure gets minimal and stop if a rectangle set comprises less than n_{min} samples. Formally, for a rectangle set R, a feature $j \in \{1, \ldots, p\}$ and split point $s \in \mathbb{R}$, we define the pair of sub-rectangles

$$R^{(1)}_{j,s} = \{r \in R : r_j \leq s\} \text{ and } R^{(2)}_{j,s} = \{r \in R : r_j > s\}. \tag{2.32}$$

With respect to the impurity measure, there are only finitely many split points of interest since we only have finitely many samples in R, and hence we need to consider no more than $n(R) \cdot p$ split points when considering R for partitioning. All of this allows us to compactly describe a procedure to grow a tree in Alg. 1, cf. [5].

Algorithm 1 Growing a tree by recursive binary partitioning.

```
1:  function TREE($(\mathbf{x}_i, y_i)_{i=1,\ldots,n}$; $Q$, $n_{\text{min}}$)
2:      $\mathcal{R} \leftarrow \emptyset$                                   ▷ Final partioning
3:      $\mathcal{I} \leftarrow \{\mathbb{R}\}$                              ▷ Intermediate partioning
4:      while $\mathcal{I} \neq \emptyset$ do
5:          Choose some $R \in \mathcal{I}$
6:          $\mathcal{I} \leftarrow \mathcal{I} \setminus \{R\}$
7:          $(j_*, s_*) \leftarrow \arg\min_{j,s} n\left(R^{(1)}_{j,s}\right) Q\left(R^{(1)}_{j,s}\right) + n\left(R^{(2)}_{j,s}\right) Q\left(R^{(2)}_{j,s}\right)$
8:          for $\ell = 1, 2$ do
9:              $R_\ell \leftarrow R^{(\ell)}_{j_*,s_*}$
10:             if $n(R_\ell) < n_{\text{min}}$ then
11:                 $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_\ell\}$
12:                 $c_{R_\ell} \leftarrow 1$ if $\hat{p}_{R_\ell} > 0.5$ else $-1$
13:             else
14:                 $\mathcal{I} \leftarrow \mathcal{I} \cup \{R_\ell\}$
15:             end if
16:         end for
17:     end while
18:     return $\sum_{R \in \mathcal{R}} c_R \chi_R$
19: end function
```

Forests Trees are known to be notoriously *noisy*, meaning if we fix $x \in \mathbb{R}^p$, the variance $V_{\mathbf{z}}(T(x))$ is quite high; the index $\mathbf{z}$ makes explicit that we consider $T(x)$ as a random variable that depends on drawing n i.i.d. training samples $\mathbf{z}$ (and then fitting a tree to them according to Alg. 1). On the other hand, if grown deep, trees have pretty low *bias*, meaning for fixed $x \in \mathbb{R}^p$ the expectation $E_{\mathbf{z}}(T(x))$ is close to $E(Y \mid X = x)$. High variance and low bias makes trees ideal candidates for a method called *bagging*, which averages the predictions of many noisy, approximately unbiased models with the goal to reduce their variance, cf. [3]. Given trees T_b, $b = 1, \ldots, B$, fitted to identically distributed training data

$\mathbf{z}_b$, we denote the new model by

$$\overline{T} = \frac{1}{B} \sum_{b=1}^{B} T_b. \tag{2.33}$$

We again fix some $x \in \mathbb{R}^p$. Since the expected value is linear and the $T_b(x)$ are identically distributed, we have $E_{\mathbf{z}}(\overline{T}(x)) = E_{\mathbf{z}}(T_1(x))$ and we see that bagging leaves the bias unchanged. Consequently, our hope rests on reducing the variance. Say, every $T_b(x)$ – again understood as a random variable dependent on sampling the training data – has variance σ^2. If the $T_b(x)$ are independent, the variance is additive and we have $V_{\mathbf{z}}(\overline{T}(x)) = \sigma^2/B$, which tends to zero as $B \to \infty$. In practice, the $T_b(x)$ are not totally independent as they have training samples in common: we do not sample $\mathbf{z}_b$ from the whole population (X, Y), but draw $\mathbf{z}_b$ as a *bootstrap* sample of size n from our training data $(\mathbf{x}_i, y_i)$, $i = 1, \dots, n$. Bootstrapping – uniformly sampling without replacement – is the key concept behind bagging, explaining why *bootstrap aggregation* is a synonym for it. If, in this situation, the $T_b(x)$ have a pairwise correlation of $\rho > 0$, we have

$$V_{\mathbf{z}}\left(\overline{T}(x)\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2, \tag{2.34}$$

which tends to $\rho\sigma^2$ as $B \to \infty$, cf. [13, Eq. (15.1)].

Thus, in addition to bagging, random forests strive to reduce to correlation between the trees without increasing the variance and bias too much. We achieve this by randomly selecting a subset of $m \leq p$ features every time we split a rectangle into two sub-rectangles when growing the tree in Alg. 1. In more detail, the indices j over which we take the minimum in line 7 only range over these m features. We denote the modified function with the appended parameter m by RfTree. Alg. 2 applies these ideas all together.

Algorithm 2 Growing a forest with bagging and random feature selection.

```
1: function RandomForest($(\mathbf{x}_i, y_i)_{i=1,\dots,n}$; $B$, $m$, $Q$, $n_{\min}$)
2:     $\mathcal{E} \leftarrow \emptyset$                                ▷ Ensemble to be filled
3:     for $b = 1, \dots, B$ do
4:         Draw a bootstrap sample $(\tilde{\mathbf{x}}_i, \tilde{y}_i)_{i=1,\dots,n}$ from $(\mathbf{x}_i, y_i)_{i=1,\dots,n}$
5:         $T \leftarrow$ RfTree($(\tilde{\mathbf{x}}_i, \tilde{y}_i)_{i=1,\dots,n}$; $Q$, $n_{\min}$, $m$)
6:         $\mathcal{E} \leftarrow \mathcal{E} \cup \{T\}$
7:     end for
8:     return sgn $\circ \left(\frac{1}{B} \sum_{T \in \mathcal{E}} T\right)$
9: end function
```

In contrast to GLMs with their continuous output, random forests output a binary classification, so we cannot control their prevalence. Widely used default values are $m = \lfloor \sqrt{p} \rfloor$ and $n_{\min} = 1$. As for B, we have shown above that increasing it reduces the variance at constant bias; in practice, therefore, only computation time limits the number of trees in the forest. We can go for a large B as fitting the constituent trees is an embarrassingly parallel problem. We can also validate random forests very efficiently by facilitating *out-of-bag* (*OOB*) samples: for each sample $(\mathbf{x}_i, y_i)$, we construct its *OOB prediction* by averaging only those $T_b(\mathbf{x}_i)$ for which $(\mathbf{x}_i, y_i)$ was not part of the bootstrapped training sample – no need for a time consuming cross validation. For random forests, we use OOB predictions as validated predictions.

2.2.4 Nested models

Given some *early* models – e.g. core models – $f_i : \mathbb{R}^p \to \mathbb{R}$, $i = 1, \dots, m$, we can nest them into another, *late* model $f : \mathbb{R}^m \to \mathbb{R}$ and obtain a new model $g = f \circ (f_1, \dots, f_m)$.

Early models trained on another data set

Often, the early models have been trained on another, independent data set, so we observe their output as features in our data set. Examples are the cell-of-origin signature, the IPI score or the LAMIS. Such f_i are merely projections onto a feature.

Early models trained on the same data set

If we need to fit some of the early models to the training data ourselves, getting reliable cross-validated predictions for f becomes trickier. Without loss of generality, we assume the only early model left to be fitted is f_1. We want to fit f_1 according to a hyperparameter tuple h_1 and f according to h_2, giving us the hyperparameter tuple $h = (h_1, h_2) \in H$ for the nested model. How do we get realistic cross-validated predictions $\hat{y}_i$, $i \in F_\ell$, for the ℓ-th fold? We start by fitting f_1 to $(\mathbf{x}_i, y_i)$, $i \in \{1, \dots, n\} \setminus F_\ell = F_\ell^c$ subject to h_1. The question reduces to: which values for the output of f_1, denoted o, do we provide the algorithm that fits f to $(o_i, f_2(\mathbf{x}_i), \dots, f_m(\mathbf{x}_i); y_i)$, $i \in F_\ell^c$?

The choice $o_i = f_1(\mathbf{x}_i)$ is most likely too optimistic since f_1 overfits its training data to some extent. Consequently, f will put too much trust in o and will generalize poorly, as the cross-validated predictions of f will already show. We want to tackle this problem by using cross-validated predictions for f_1, instead. The theoretically cleanest approach is to do an *inner* cross validation for f_1, i.e. another cross-validation for fitting f_1 to $(\mathbf{x}_i, y_i)$, $i \in F_\ell^c$, and use the cross-validated predictions from *this* cross validation for o. However, the additional loop increases time complexity by a factor of k for a k-fold cross-validation. Because we fit f_1 to $(\mathbf{x}_i, y_i)$, $i = 1, \dots, n$, in a cross validation anyway, we prefer to use the cross-validated predictions we get in this process for o. This even allows to formulate the approach for arbitrary validated predictions for f_1 and f. Alg. 3 writes this down clearly.

Algorithm 3 Nested pseudo cross validation.

```
1: function NestedPseudoValidation($(x_i, y_i)_{i=1,\dots,n}$; $h = (h_1, h_2)$, $k$)
2:     Fit $f_1$ to $(x_i; y_i)$, $i = 1, \dots, n$, subject to $h_1$ , yielding validated predictions $o$.
3:     Fit $f$ to $(o_i, f_2(x_i), \dots, f_m(x_i); y_i)$, $i = 1, \dots, n$, subject to $h_2$, yielding validated
       predictions $\hat{y}$.
4:     $g \leftarrow f \circ (f_1, \dots, f_n)$
5:     return $(\hat{y}, g)$
6: end function
```

The gain in time complexity compared to the approach involving an inner cross validation comes at a little price: a not thoroughly clean validation of g. To see this, we assume that the validated predictions for f_1 and f are both cross-validated predictions with the same assignment of samples into folds – a very realistic assumption as we usually choose $k = n$. We fix a sample i and another sample j that is in another fold. Sample i's cross-validated prediction $\hat{y}_i$ for g – or equivalently its cross-validated prediction for f – depends on the cross-validated prediction o_j of sample j for f_1. o_j in turn depends on sample i. All in all, $\hat{y}_i$ very slightly depends on $(\mathbf{x}_i, y_i)$.

Next, we want to talk about how we tune the hyperparameters of a nested model. In practice, one thinks of the hyperparameter tuples H as partitioned into subsets. We summarize the hyperparameters of a model that make it most distinct to other models in its *architecture*. For a core model, the architecture concerns its class – Gauss, logistic, Cox model or random forest – and the a-priori selected features. For a nested model, its architecture is defined by the architectures of the involved core models and the hierarchy that nests them into one another.

For a given architecture of a nested model, we need to tune the hyperparameter tuple $h = (h_1, h_2)$ of Alg. 3, i.e., we repeatedly call NestedPseudoCV as we vary $h \in H_0$ with

H_0 being the set of tried out hyperparameter tuples for this model architecture. What should H_0 look like? We have candidate hyperparameter tuples $H_{0,1}$ for f_1 and $H_{0,2}$ for f. An unprejudiced strategy is to set $H = H_{0,1} \times H_{0,2}$. Compared to non-nested models, we try out a considerably higher number of hyperparamter tuples – roughly a factor of $|H_{0,1}| \approx |H_{0,2}|$ – and the validated errors of nested models have a higher chance of being overestimated, cf. subsection 2.1.1. We therefore settle on a greedier strategy: we first tune $h_1 \in H_{0,1}$, yielding a best validated hyperparameter tuple $\hat{h}_1$, and then go on to tune $h_2 \in H_{0,2}$. This means $H_0 = \{\hat{h}_1\} \times H_{0,1}$.

f_1 often deals with high-dimensional input. E.g., f_1 is a Gauss model predicting from several thousand gene-expression levels. Hence, we undoubtedly need elastic-net regularization in its training and need to tune the regularization strength λ. With this alone, we quickly end up with several hundred hyperparameters to be tuned for f_1. With $H_{0,1}$ being so big, good validated performances have one thing in common: they and the corresponding cross-validated predictions are too optimistic. This means, we still have not gotten entirely rid of the problem that f puts too much trust into f_1's output misled by too optimistic values in the training. This problem will propagate into another problem: too optimistic cross-validated predictions for g and an overestimated cross-validated performance of g. It is hard to quantify the effect, so we will have a very close eye on models nested according to Alg. 3 in chapter 3.

Nested models and inter-technical variability

Considering a nested model $g = f \circ (f_1, \dots, f_m)$, what happens if the output of some of the early models f_i is sensitive to inter-technical variability? Without loss of generality, let only f_1 be such an early model. Moreover, let both f_1 and f be a GLM, which we describe with the linear predictor as $f_1 : \mathbb{R}^p \to \mathbb{R}, x \mapsto x^T\beta$ and $f : \mathbb{R}^m \to \mathbb{R}, x \mapsto x^T\gamma$. $\mathbf{z} \in \mathbb{R}^{n\times p}$ holds the same features for the same samples as the predictor matrix $\mathbf{x}$, but measured under a different protocol. Let $J = \{1, \dots, p\}$ denote the features that are affected by the change of protocol, e.g. the gene-expression levels. Let β fulfill the zero-sum constraint $\sum_{j\in J} \beta_j = 0$. We assume that Eq. (2.5) holds with small residues ϵ_{ij}. For $j \notin J$, we this is trivial since we even have $\epsilon_{ij} = 0$. For $j \in J$, this is also a realistic assumption since, in this thesis, f_1 is always a gene-expression signature.

With Eq. (2.7), we obtain

$$\begin{aligned} g(\mathbf{z}_i) &= (f \circ (f_1, \dots, f_m))(\mathbf{z}_i) \\ &= (\mathbf{z}_i^T\beta, f_2(\mathbf{z}_i), \dots, f_m(\mathbf{z}_i))^T\gamma \\ &\approx (\mathbf{x}_i^T\beta + c, f_2(\mathbf{x}_i), \dots, f_m(\mathbf{x}_i))^T\gamma \\ &= c\gamma_1 + (f \circ (f_1, \dots, f_m))(\mathbf{x}_i) = c\gamma_1 + g(\mathbf{x}_i) \end{aligned} \tag{2.35}$$

for some $c \in \mathbb{R}$ and all samples $i = 1, \dots, n$. The sample-wise shift in the output of f_1 propagates into a sample-wise shift in the output of g – qualitatively this is the same as in Eq. 2.7. The residuals ϵ_{ij}, β and γ impact the above approximation. From our perspective, where output ordering matters most, we can say: the better this approximation, the more monotonic the $g(\mathbf{z}_i)$ are in $g(\mathbf{x}_i)$. We do not elaborate on this theoretically any further; in our practice in section 3.3, we will see that the approximation is indeed very good.

While f_1 is always a GLM in this thesis, one might consider a random forest for f. Random forests, which predict solely by thresholding single features, are highly sensitive to shifts in the output of f_1. In this situation, we should rather make f_1 agnostic to inter-technical variability once and for all by thresholding its output on the respective batch of samples, but this risks losing the nuanced information a continuous output provides. These considerations turn the simplicity of GLMs into an asset against more complicated models: for GLMs we can better foresee how they will react to inter-technical variability.

2.3 Software

We have designed the software that generates the results of this thesis in such a way that it can be the foundation of the machine-learning part of MMML-Predict as the project develops and other researchers take over. Beyond this, we wrote code, wherever it was possible, in such a general way that one can apply it not just for the MMML-Predict problem, but for all equivalent and closely related problems. We ran all code in the statistical-computing environment `R` [18].

2.3.1 The R package `patroklos`

The R package `patroklos` holds all of the highly reusable code of this thesis. It brings all the methods described in this chapter to life and is available on GitHub [10] together with a website[1]. In fact, one can deploy `patroklos` to solve any binary classification problem in a supervised-learning manner. `patroklos` provides some extra functionality for those problems among them where the response is a thresholded survival time, high-dimensional data is involved and one wants to maximize the precision of the classifier for a sufficiently large positive cohort.

Training `patroklos` allows the user to quickly specify model architectures and attach hyperparameters tuples to them, including model-agnostic hyperparameters to all of them. It provides a decorator to transform a function that fits merely one model for a single hyperparameter tuple into a function that fits and validates models for a variety of hyperparameter tuples. It supports fitting core models with the support of the R packages `zeroSum` [21] and `ranger` [29] out of the box. `zeroSum` augments the popular `glmnet` package by the zero-sum constraint: it fits and tunes Gauss, logistic and Cox models in a cross validation and endows their loss function with elastic-net regularization and – unlike `glmnet` – the zero-sum constraint. The time-critical work of `zeroSum` happens in a fast, well-structured C++ backbone. `ranger` fits random forests fast, also thanks to a performant C++ backbone.

Validation and testing If fitting functions from existing R packages calculate a validated error of their fitted models at all, they do so in a plethora of ways, usually using an error measure derived from the loss function they minimize. E.g., `zeroSum` uses the binomial deviance for logistic models and `ranger` uses the classification error for classification random forests. `patroklos` takes full control of validation and unifies it by calculating the same error measure across all models and even more statistics of the model with the help of the `AssScalar` R6 class. With the `Ass2d` R6 class, the user can plot one property of a truly binary classifier against another, which guides thresholding models with continuous output as in Fig. 3.4. `patroklos` calculates logrank p-values and hazard ratios together with their confidence intervals by means of the R package `survival` [27].

Modular, extendible design `patroklos` defines function interfaces so the user can easily wrap fitting functions implemented in other R packages or by the user to make them a part of the `patroklos` pipeline. The two R6 classes `Model` and `Data` strive to abstract models from the data in such a way that the user can easily apply H to new data.

Meta analysis `patroklos` provides tools to analyze how the validated error depends on the test error and allows to highlight subgroups of model architectures to unveil systematic flaws in validation and test performance.

[1] `https://lgessl.github.io/patroklos`

2.3.2 `patroklos` in action

The Git repository that brings `patroklos` into action for this thesis is available in a public version on GitHub [11]. We try our best at presenting the results in a comprehensive and understandable way in the following chapter 3. Yet, since code is more precise than freely written text, the Git repository rigorously defines everything we have done to obtain the results.

The public Git repository does not contain the data because part of it must not be publicly available. To reproduce the results of this thesis, the reader may request access to the scientific computing servers of the Chair of Statistical Bioinformatics, University of Regensburg, from Christian Kohler[2]; on these servers, a mounted volume holds all data sets.

[2]Email: `christian.kohler@ur.de`

Chapter 3

Results

In this triune chapter, we start with introducing three DLBCL data sets that include survival, clinical and molecular features. Next, in *intra-trial experiments*, we split every of these three data sets into a train and test cohort, fit a variety of candidate models to the training cohort, validate them on the same and test the best on the test cohort. This is less about presenting a high-performing model, but more about analyzing validated and tested errors to make the set of tried out hyperparameter tuples H slimmer and better for the future. This future plays out in the last part as we train and validate on one of the three data sets and test on another and deal with cross-platform variability in *inter-trial experiments*.

3.1 The data

Our DLBCL data sets are taken from papers by Schmitz et al. [22], Reddy et al. [19] and Staiger et al. [24]. See Table 3.1 for key properties and comparison. We will refer to them as Schmitz, Reddy and Staiger data, respectively.

Schmitz data The data by Schmitz et al. includes the five IPI features in their continuous form. By heuristically optimizing a novel genetic distinctiveness metric, Schmitz et al. clustered 574 DLBCL biopsy samples into four genetic subtypes. They unblinded the survival data only after the clustering was complete and the model was frozen [22, Appendix 1, pp. 16–18]. The following survival analysis on a subset of 229 patients, for who survival

	Schmitz	**Reddy**	**Staiger**
prospective trial	no	no	yes
response	PFS	OS	PFS
# samples	229	604	466
# genes with expression level	25 066	13 302	145
technology	RNA-seq	RNA-seq	NanoString
high risk [%]	36.6	31.5	24.3
prev(tIPI) [%]	12.9	21.6	17.0
prec(tIPI) [%]	65.2	54.1	38.2

Table 3.1: Overview on used data sets. All datasets include the five IPI features in their thresholded format, gender, cell of origin, and, added by us, the LAMIS signature. The Schmitz and Staiger data use the high-risk definition of MMML-Predict: PFS below two years; because the Reddy data reports only overall, no progression-free survival, we define high risk as overall survival below 2.5 years there, leading to a comparable high-risk proportion. Technology refers to the technology used to measure gene-expression levels.

data was available and who had undergone R-CHOP or CHOP-like treatment, unveiled significantly differing PFS between the four genetic subtypes. The IPI score did not vary significantly between the subtypes, indicating that their new classifier gives us additional information to predict survival.

There are two caveats: First, the genetic classifier saw the entire data set during training and this runs afoul of a strict train-test regime even if training was survival-agnostic. This only affects our intra-trial experiments as in the inter-trial experiments we do not include the genetic subtype as a feature. More importantly, we always need to be careful with using features in a data set that are the output of some model trained on this data set. Second, this data set is not the result of a prospective, representative trial, but consists of opportunistically collected patients from highly renowned U.S. hospitals, which preferably treat difficult cases. As a result, the high-risk proportion in the Schmitz data is at 36.6% – compared to 24.3% in the prospective Staiger data – and the tIPI reaches a precision of 65.2% for classifying high-risk patients at a prevalence of 12.9% – compared to 38.2% precision at 17.0% prevalence in the Staiger data. The tIPI already meets the MMML-Predict goals, but we want to see if we can do even better in such a high-risk regime.

Reddy data Compared to a prospective study, also the Reddy data, comprised of 604 patients treated with rituximab-containing regimens, is enriched for high-risk patients. The tIPI has a performance that already satisfies the MMML-Predict goals, even if a bit less convincingly than on the Schmitz data. After identifying 150 DLBCL driver genes, Reddy et al. trained a Cox model, named genomic risk model, that predicts overall survival (OS) from combinations of genetic events and gene-expression markers (cell of origin, MYC and BCL2 overexpression) and thresholded its output into low, intermediate and high risk. This is a model whose predictions we cannot use because the authors do not tell us which samples they assigned into the train and test set, but that inspired us to use combinations of discrete features, cf. subsection 2.2.1. Nevertheless, with high expression and translocation of MYC, BCL2 and BCL6, the data provides some of the input features of the genomic risk model as well as three binary clinical features: B symptoms at diagnosis, testicular and central-nervous-system involvement. B symptoms refer to the triad of fever, night sweats and unintentional weight loss being simultaneously present.

Staiger data The Staiger data is composed of 466 patients treated with CHOP-like and R-CHOP-like regimens and enrolled in prospective clinical trials. Staiger et al. first determined 731 gene pairs with highly correlated gene-expression levels between their training cohort – 233 DLBCLs with gene-expression levels built from the Affymetrix GeneChip technology – and the Staiger data – 466 DLBCLs with gene-expression levels built from the NanoString nCounter technology – with the help of six paired nCounter-GeneChip samples, cf. [24, Supplementary Methods]. Next, they learned a LASSO-regularized Cox model on the differences of the (logarithmized) gene-expression levels from these gene pairs and the five thresholded IPI features. Afterwards, they removed the five IPI features from the model aiming to make it independent of the IPI. One can expand the differences of gene-expression levels in the signature to obtain an ordinary gene-expression signature with coefficients corresponding to single genes: the LAMIS (lymphoma-associated macrophage interaction signature). It is based on 17 genes, but dominated by just two genes, CSF1 and CPT1A.

By dichotomizing the continuous output of the LAMIS, which we call the *LAMIS score*, at the 75%-quantile into their *LAMIS group* (low or high), Staiger et al. present two groups on the Staiger data with significantly differing PFS and OS; meanwhile, the IPI features, breaks in MYC, BCL2, BCL6, and cell of origin remain prognostic indicators independently of the LAMIS group.

Since the LAMIS coefficients fulfill the zero-sum property, we apply the LAMIS unchanged on the two other data sets. Under allegedly realistic assumptions and according

	Schmitz	**Reddy**	**Staiger**
# samples	58	151	117
high risk [%]	37.0	31.6	24.3
prev(tIPI), prec(tIPI) [%]	$17.0, 50.0$	$19.2, 42.1$	$39.2, 38.7$
prev(m^*), prec(m^*) [%]	$35.1, 68.4$	$23.0, 55.6$	$34.6, 45.9$
precision 95%-CI m^* [%]	$[43.5, 87.4]$	$[35.3, 74.5]$	$[29.5, 63.1]$
HR m^*	1.00	1.04	13.6
HR 95%-CI m^*	$[1.00, 1.00]$	$[1.02, 1.06]$	$[1.99, 93.2]$
logrank p m^*	3.69×10^{-4}	1.82×10^{-3}	9.38×10^{-4}

Table 3.2: Statistics of intra-trial experiments on respective test cohort. m^* denotes the best model trained and validated model on the respective training cohort.

to Eq. (2.7), the LAMIS scores have a data-set-dependent shift. This chapter, especially section 3.3, will show how realistic these assumptions are. We also add the LAMIS group by dichotomizing the LAMIS output at its 75%-quantile of the respective data set.

Combined data We will conduct inter-trial experiments on a large data set consisting of the samples from the Schmitz, Reddy and Staiger data. As for the gene expression-levels, we map the Ensemble gene IDs used in the Reddy data to HGNC gene symbols with the help of the `BiomaRt` R package [8] to end up with the same gene nomenclature in all three data sets. Intersecting the gene-expression features leaves us with gene-expression levels for 119 genes and the features gender, age, the five thresholded IPI features, the IPI score, the IPI group, the LAMIS score and the LAMIS group. The resulting data set is 1299 samples strong.

3.2 Intra-trial experiments

To gain first insights into our methods, we conduct intra-trial experiments separately on the Schmitz, Reddy and Staiger data. To this end, we split every data set into a train and test cohort. We do so uniformly at random, with two constraints: first, a ratio of 3 to 1 between train and test cohort and, second, the ratio between high-risk and low-risk patients in training cohort, test cohort and overall data set is the same. As shown in Table 3.2, the performance of the tIPI can notably differ between the whole data set and the subsampled test cohort.

3.2.1 Model architectures

We want to give a brief summary of the models we send into the race and take a closer look at the best validated one, m^*, on every data set.

Candidates

Gene-expression levels only Models trained in a leave-one-out cross-validation only on the gene-expression levels include the Gauss, logistic and Cox model. Regarding noteworthy hyperparameter decisions, we both apply and do not apply standardization of the predictor; we do not demand the zero-sum constraint, i.e., all zero-sum weights are 0, because, at this point, we do not want to transfer our models to other data sets; we regularize with elastic-net penalty factor $\alpha \in \{0.1, 1\}$. We choose the training survival cutoff $T \in \{1, 1.25, 1.5, \dots, 2.5\}$. For Cox models, we additionally set $T = \infty$. For the Reddy data, we add 0.5 to these values of T to account for classifying OS > 2.5 years as opposed to PFS > 2 years.

We train and validate a model for every combination of these hyperparameters. Not taking into account the values of the regularization strength λ (which are hard to foresee due to early stopping), this adds up to 88 models.

Looking at the models with top validated performance (cf. projection upon validation error in Fig. 3.1), for the following, we narrow down T to one or two values (usually at or slightly below two years) and set α to 1 as the more complex models trained for $\alpha = 0.1$ cannot clearly outperform the sparse LASSO-regularized models. For models only predicting from gene-expression levels, we do not standardize the predictor anymore.

Core models with other features We now add all available remaining features. For the IPI, we either add the five IPI features in their thresholded format, the IPI score, the IPI group or all of them at once. For $s_{\text{min}} = 0.05$ and $n_{\text{combi}} \in \{1, 2, 3, 4\}$, we add combinations of categorical features to the predictor. We both include and exclude gene-expression levels in the predictor. As all of these models need to deal with features on different scales, we always standardize the predictor. We regularize all GLMs with the LASSO because we may end up up adding several hundred combined categorical features to the predictor. For random forests, we let n_{min}, m and B fluctuate around their default values.

Nested models We nest models according to Alg. 3, where the early model f_1 is the best validated model among the gene-expression-only models. If the late model f is not a random forest, but a GLM, we regularize it with the LASSO. We select features a priori as in the previous paragraph.

Best validated models

Schmitz and Reddy For both the Schmitz and Reddy data, the best model according to the validation error is a nested model as in Alg. 3 with the early model f_1, a Gauss model, predicting from the gene-expression levels and the late model f being a Cox model. They only differ in their features: For the Schmitz data, the predictor holds IPI-related features in all of the above-mentioned formats as opposed to only the five IPI features in their thresholded format for the Reddy data. We have $n_{\text{combi}} = 2$ for Schmitz as opposed to $n_{\text{combi}} = 3$ for Reddy.

On the Schmitz data, m^*'s validated precision of 89.3% drops by more than 20 points to 68.4% on the test set. In an even bigger drop, m^*'s validated precision of 78.6% declines to 55.6% on the Reddy test set. These way too optimistic cross-validated errors demand further investigation.

Staiger On the Staiger data, m^* is a very simple model: a logistic model that neither uses gene-expression levels nor combinations of discrete features ($n_{\text{combi}} = 1$). Of note, the model incorporates the LAMIS score and group with decisively non-zero coefficient. Here, m^*'s validated precision of 58.2% is more in line with the 45.9% precision on the test set.

Logrank test The last row of Table 3.2 holds two-sided p-values for the logrank test statistic as proposed by Mandel [16], which is calculated under the null hypothesis that the positive and negative group according to m^* have the same hazard. While m^* for all three data sets fails to beat the precision of the tIPI significantly, all logrank test statistics are highly significant. In fact, all models picked as the winner in a validation in this chapter achieve a highly significant logrank test. This shows that significantly beating the IPI's precision is much harder than classifying patients into two groups with clearly different hazard. For this reason, the logrank test does not play a role in this thesis and we did not introduce it in more detail in chapter 2.

3.2.2 Meta analysis

Now that we have frozen all models and do not add further ones, we can unlock the test set and evaluate the test performance of more models. Our main focus rests on the discrepancy between validated and test error. As we laid out in section 2.1, the models with the best validated errors most likely have an underestimated test error and it is hard to fight this effect qualitatively. However, we use several methods to calculate a validated error – cross validation, the pseudo cross validation of Alg. 3 and the OOB error of random forests – and, for every model architecture, we tune different hyperparameters and a different number of hyperparameter tuples. Both validation error and test error of a model are random variables and it is hard to infer statistical properties of them based on a single realization of these random variables. By grouping the models according to their hyperparameters, we gain statistical power and can look for patterns in the validation and test error as well as discrepancies between them.

Models predicting from gene-expression levels only

Staiger Looking at Fig. 3.1, we see that, for Staiger, the models only using gene-expression levels cannot compete with the tIPI. While the validated errors of these models are below the -36.4% negative precision of the tIPI, the test errors all are above -30%. In all cases, the test error is at least 10 points higher than the validated error – a discrepancy that is busted by the fact that the identity line is outside the plot area.

Schmitz and Reddy Also for the Schmitz and Reddy data, the dependence between validated and test error is far from being monotonic; in both cases, the plot looks noisy and the correlation between validated and test error is negative such that the model with the lowest validated error happens to have the highest (Schmitz) or second highest (Reddy) test error. The plots do not show errors for all tried out hyperparameter tuples in H, but only for a high-performing subset with λ, T and n_{combi} already optimized according to validation; this optimization was successful in the sense that even the worst models post a test error below that of tIPI.

There is no model class reliably outperforming the others. Both data sets suggest that ridge regularization yields models with a test error lower than that of LASSO regularization; validation, however, does not reveal this difference. The picture on standardizing the predictor matrix or not is mixed: on the Schmitz data, models with non-standardized predictor fluctuate less around the identity line than those with standardized predictor, meaning validation is more reliable for the former model group; on the Reddy data, meanwhile, we observe the same phenomenon with roles swapped.

Models with the full range of features

All analysis here is based on Fig. 3.2.

Schmitz and Reddy Again, we analyze the non-prospective, high-risk-heavy Schmitz and Reddy data together. When we apply the full range of a-priori selected features, validation and test error for these two data sets stay out of touch. Partitioning according to the model architecture reveals some patterns.

Models nesting a Gauss model into a GLM as in Alg. 3 all have low validated errors; on the Reddy data, their validated errors are lower than those of all other models. The test error is always higher than the validated error and usually it is *much* higher. Alg. 3 yields way too optimistic validated errors if the late model is a GLM, prompting us to no longer fit such models in the following experiments. Discarding Alg. 3 once and for all might go too far: when nesting a Gauss model into a random forest, we get validated errors more in

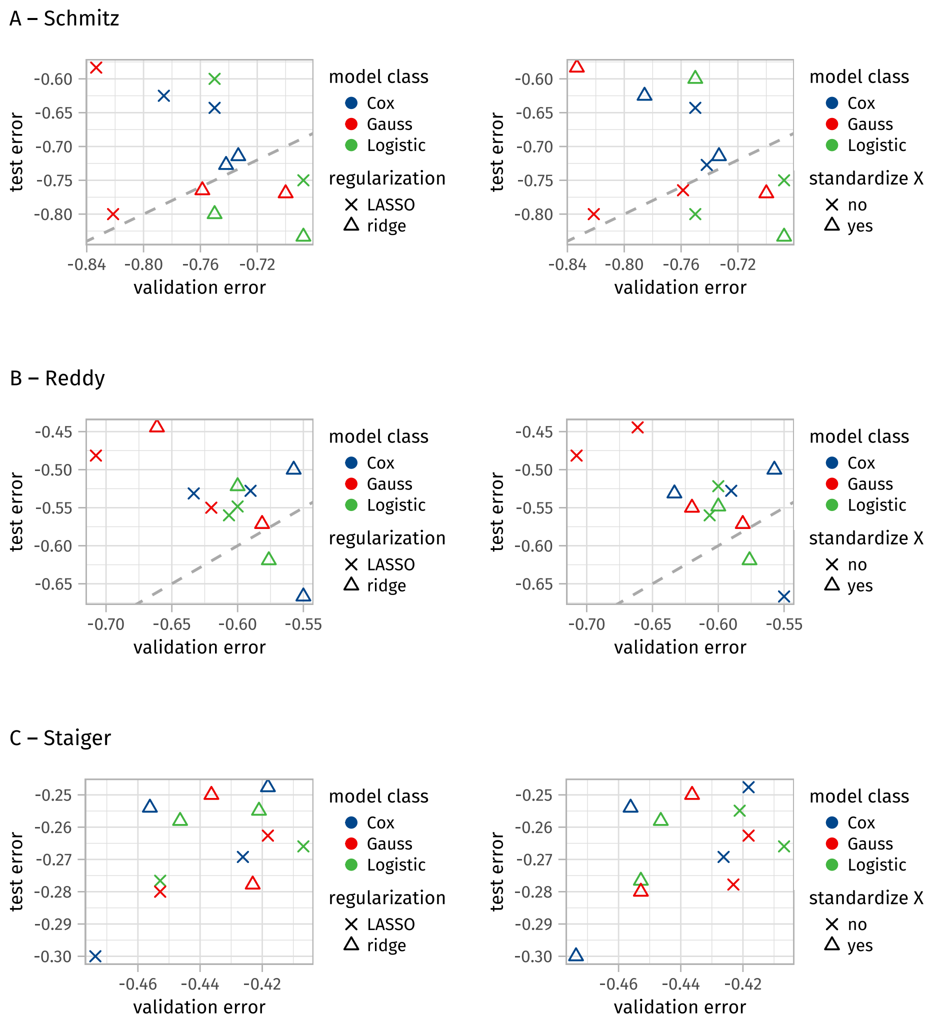

Figure 3.1: Validation versus test error in intra-trial experiments for models only predicting from gene expression levels for all three data sets in the rows. The left and right column show the same points, but highlight different groupings of the models. The elastic-net regularization strength λ, the training survival cutoff T and n_{combi} have already been tuned.

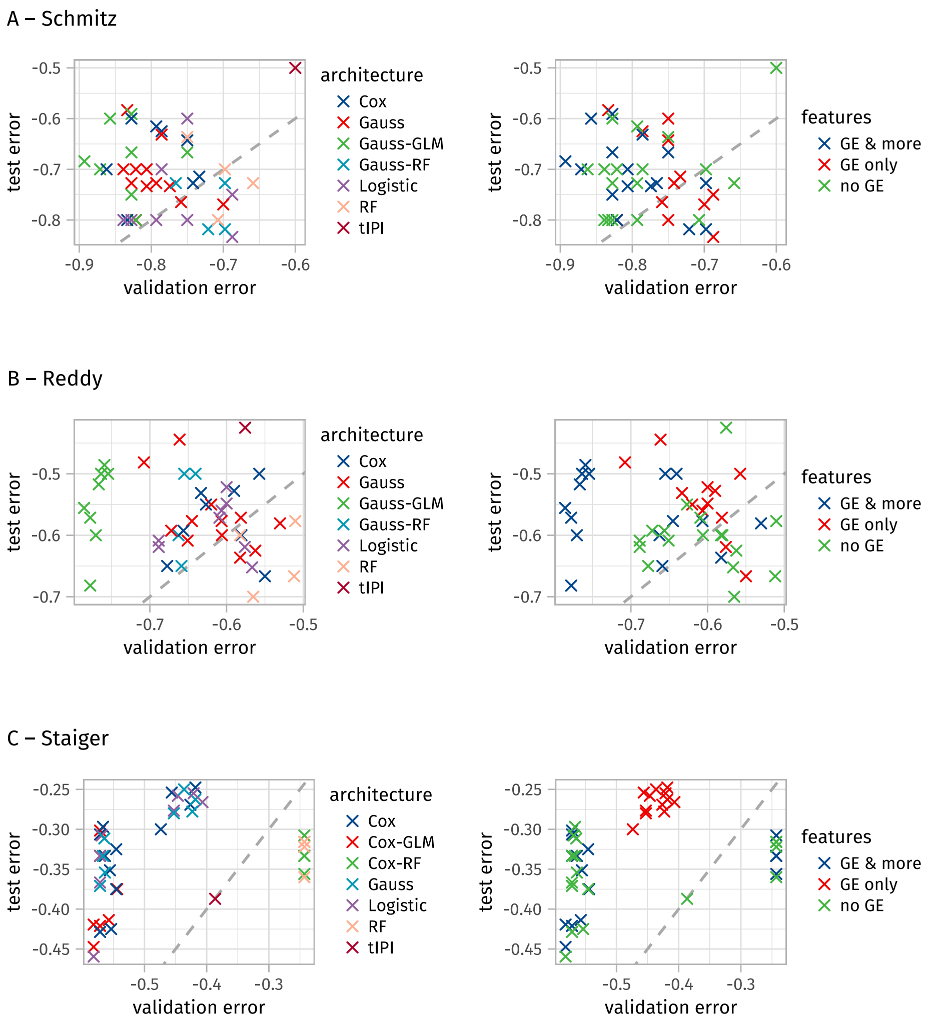

Figure 3.2: Validation versus test error in intra-trial experiments for models predicting from the full range of a priori selected features for all three data sets in the rows. The dashed gray line is the identity line. The notation c_1-c_2 encodes a model of class c_1 as the early model f_1 nested into a model of class c_2 as the late model f according to Alg. 3. “GE” is an acronym for gene expression. Both columns show the same points, but highlight different groupings. The elastic-net regularization strength λ, training survival cutoff T, n_{combi} and the random-forest hyperparameters B, m and n_{min} have already been tuned. The validation error for the tIPI is in fact also a test error on independent data, namely our training cohort.

line with test errors, especially on the Schmitz data. On both data sets, OOB predictions for models consisting of a random forest alone very well estimate the test error. In all but one case they even underestimate the test error.

As for feature selection, we see that including gene-expression levels does not lead to greatly improved models. Even the high-performing gene-expression-only models do not benefit very much if we train them with additional features. On the Reddy data, the points belonging to models not using gene-expression data are all close to or underneath the identity line; this means the validation errors catch the test errors well, their test errors are less widespread and on average lower than those of the remaining models. This does not mean that gene-expression levels do not provide important information: all of the models that do not use the gene-expression levels directly from our data do so indirectly by means of the LAMIS. We have simply outsourced developing a gene-expression based model to the LAMIS authors, thereby getting rid of $p \gg n$ and ending up with a more accurate validation.

Staiger On the Staiger data, the validated errors align better with the test errors. As with the two other datasets, OOB-based test-error estimates are conservative leading to underestimated test errors for models incorporating a random forest. As a result of highly correlated validation and test errors, the best validated model is also the best tested model. Compared to above, the validated errors of models nesting a Cox model into a GLM overestimate the test errors less; since the early Cox model yields pretty poor cross-validated predictions, we hypothesize that the training algorithm of the late GLM is less tempted to put as much weight on these predictions as with the two other data sets. Models profit a lot from not just predicting from the gene-expression levels: all the gene-expression-only models form a bulk with test errors higher than those of all other models.

3.3 Inter-trial experiments

The results of the intra-trial experiments are encouraging: we beat the precision of the tIPI by at least 7 points on three data sets, including two where the IPI already did a good job. In the following analysis, we saw validated errors often being detached from test errors. To tackle this issue, we decided to no longer train certain models. We also got the impression that the LAMIS is at least as capable of catching the molecular information in the gene-expression levels as the models we trained for that purpose ourselves, but allows training more precise models and validating them more accurately. Another way to close the validation-test gap is to increase the number of samples in both the train and test cohort: it both makes overfitting the validated predictions to the training cohort harder and makes it less likely that the test cohort includes cases biologically not covered at all in the training cohort. We will now go along this path as we use every of the three data sets in their entirety for training and validation and then test on the other two.

3.3.1 Model architectures

Here, we want to sketch the architectures of the candidate models and take a closer look at the best validated model m_i^* trained on every data set.

Candidates

Even with combined discrete features, the number of features in the predictor p never exceeds 300 in this section, meaning computation is comparatively cheap. Consequently, we can afford a leave-one-out cross-validation for every model that is not a random forest and $B = 1000$ trees for every random forest.

Gene-expression levels only This time, we require every Gauss, logistic and Cox model to fulfill the zero-sum property for the gene-expression levels it deals with as features since every data set has its own protocol for measuring gene-expression levels. We have gene-expression levels for 119 genes in the combined data and these genes are a subset of the 145 genes for which the Staiger data includes gene-expression levels. Even though the gene-expression-only models trained on the LAMIS data had high errors in both validation and testing, we once more train LASSO-regularized Gauss, logistic and Cox models predicting only from gene-expression levels hoping they can exploit the improved data situation with more samples. In line with the zero-sum idea, we do not standardize the predictor and have T range from 1 to 2.6 years in steps of 0.2 years for models trained on the Schmitz and Staiger samples; for the Reddy samples, here and for the rest of this section, we shift T by 0.5 years to the future accounting for the fact that on the Reddy data, absent PFS, we need to classify OS < 2.5 years as opposed to PFS < 2 years.

Core models with other features As in the intra-trial experiments, we generously add all remaining features to the predictor. Only for the LAMIS, we vary its format and add just the score, just the group or both. We augment the predictor with combinations of discrete features according to $s_{\text{min}} = 0.05$ and $n_{\text{combi}} \in \{1, 2, 3\}$. We restrict the training survival cutoff T to range between 1.4 and 2 years in steps of 0.2 years for the Schmitz and Staiger data. We both include and exclude the expression levels in the predictor of the Cox and logistic models, which we fit for all combinations of these hyperparameter values.

Random forests have trouble dealing with features systematically shifted between data sets, so we always exclude the gene-expression levels from the predictor and, of the LAMIS formats, we only include the LAMIS group in the predictor. Moreover, random forests can realize combinations of categorical features themselves, so we set $n_{\text{combi}} = 1$ for them.

Best validated models

Table 3.3 presents the performance of the best models validated and trained on every cohort, m_i^*, $i \in \{\text{Schmitz}, \text{Reddy}, \text{Staiger}\}$. We now loop through i.

Schmitz m^*_{Schmitz} is a Cox model predicting from only categorical features and combinations of up to three of them. In particular, no gene-expression levels are directly involved and the model confines itself with the LAMIS *group*. All in all, the sparse model uses 7 features. During training, we provided the model with a fairly low survival cutoff of $T = 1.4$ years.

When tested on the Reddy data, m^*_{Schmitz} beats the precision of the tIPI by 5 points. This is not enough to show significant superiority in the sense that the 95%-confidence interval of our model's precision does not include the precision of the tIPI.

On the Staiger set, meanwhile, m^*_{Schmitz} accomplishes all objectives: with 50.7% precision, it overtops the precision of the tIPI by more than 12 points, narrowly surpasses the psychologically important 50% threshold, and the 95%-confidence interval of our model's precision excludes the precision of the tIPI. m^*_{Schmitz}, originating from the high-risk, non-prospective Schmitz data with the LAMIS measured with the RNA-seq technology, acclimatizes well to the prospective, NanoString regime of the Staiger data and performs better than the models we trained in the previous section on part of the Staiger data.

Reddy m^*_{Reddy} looks quite similar to m^*_{Schmitz}. Instead of the LAMIS group, the logistic model uses the LAMIS score making it its only continuous feature. Its predictor contains combinations of up to two categorical features. Hyperparameter tuning yielded a training survival cutoff of 2.3 years, somewhat below the 2.5 cutoff separating patients into high-risk and low-risk on the Reddy data.

	Schmitz	**Reddy**	**Staiger**
Schmitz	12.9, 65.2	17.7, 59.6	17.1, 50.7
Reddy	17.8, 71.1	21.6, 54.1	18.0, 53.2
Staiger	17.4, 75.7	22.5, 50.4	17.0, 38.2

(a) Prevalence and precision. Diagonal entries (i, i) hold prevalence, precision of the tIPI on cohort i. Off-diagonal entries (i, j) hold prevalence, precision on cohort j of the best model trained and validated on cohort i, m_i^*.

	Schmitz	**Reddy**	**Staiger**
Schmitz	65.2	48.6	38.7
Reddy	54.1	54.1	41.5
Staiger	58.5	40.9	38.2

(b) Lower limit of the 95%-confidence interval of the precision. Diagonal entries (i, i) hold the precision of the tIPI on cohort i. Off-diagonal entries hold the lower limit of the 95%-confidence interval of the precision on cohort j of the best model trained and validated on cohort i, m_i^*.

	Schmitz	**Reddy**	**Staiger**
Schmitz	1.61, 1.37–1.91, 0	1.62,1.45–1.80, 0	1.59, 1.39–1.83, 0
Reddy	53.36, 18.73–152.08, 0	1.58, 1.40–1.78, 0	23.87, 9.89–57.61, 0
Staiger	1.46, 1.32–1.60, 0	1.32, 1.24–1.41, 0	1.46, 1.25–1.70, 0

(c) Hazard ratio, its 95%-confidence interval and p-value for the null hypothesis of the hazard ratio being equal to one. Diagonal entries show these properties for the tIPI, off-diagonal entries for the best model trained and validated on cohort i, m_i^*. All p-values are below 5×10^{-6} and hence rounded to 0.

Table 3.3: Statistics of inter-trial experiments. Rows i always refer to the training cohort, columns j to the test cohort. Diagonal entries (i, i) hold some statistic about the IPI on the respective cohort.

On the Schmitz data, m^*_{Reddy} is more precise than the tIPI, but fails to outperform the tIPI significantly. It yields a fairly high hazard ratio of 53.4 $(95\%$-CI 18.7–$152.1)$.

On the Staiger set, m^*_{Reddy} posts a precision of 53.2 that is by 2.5 points higher than that of m^*_{Schmitz} and by 15 points higher than that of the tIPI. Consequently, the lower limit of the 95%-confidence interval of its precision, at 41.5%, is clearly above the precision of the tIPI and the hazard ratio rises to 23.9 $(95\%$-CI 9.9–$57.6)$. Even more than m^*_{Schmitz}, m^*_{Reddy} defies systematic differences between data sets: comparing the Reddy to the Staiger data, we notice the contrasts non-prospective versus prospective trial, RNA-seq versus NanoString nCounter technology to measure gene-expression levels and, most strikingly, classifying OS below 2.5 years versus PFS below 2 years.

Staiger The simplest of the three picked models is m^*_{Staiger}. A Cox model, it predicts from the LAMIS score and five more categorical features – no combinations of categorical features involved, $n_{\text{combi}} = 1$. With $T = 1.4$, it uses the same low training survival cutoff as m^*_{Schmitz}.

On the Reddy data, our model's precision, at 50.7%, stays below that of the tIPI at 54.1%. Speaking for the whole thesis, this renders the Reddy data a challenging test cohort with a hard-to-beat IPI, but – as we have just seen – a very helpful training cohort.

Meanwhile on the Schmitz data, the precision of m^*_{Staiger} of 75.7% exceeds that of the tIPI by more than 10%, but falls short of outrivaling the tIPI significantly. Still, this demonstrates that transferring models between the Schmitz and LAMIS data works in both directions.

3.3.2 Meta analysis

We again freeze all of our models and unlock the test sets for all of them to analyze the discrepancy between validated and test error and to gain insights into how stable thresholding the models on the test cohort is.

Validated and test errors

Fig. 3.3 compares the validation to the test error for all candidate models participating in the inter-trial experiments; note that again several hyperparameters have already been optimized in the validation.

We have emphasized that the risk profiles of the three data sets differ, especially when comparing the Schmitz and Reddy to the Staiger data, so we cannot expect the validation-test-error tuples to perfectly align at the identity line. The motivation to increase the sample size was rather to observe test errors that are more monotonic in their validation errors. Indeed, Fig. 3.3 shows test errors that are way more correlated with their validation errors than in the intra-trial experiments. As a result, the model with minimal validation error in all cases is close to being the one with minimal test error. In more detail, with the exception of the models trained on the Staiger and tested on the Reddy data, the tested precision of the best validated model and that of the best tested model deviate by no more than 2 points. In three of the six cases, the best validated and tested model coincide. This is a solid foundation to train and validate more models.

As for the model class, we see that no model class prevails clearly. Ignoring the tIPI, the random forest always finishes last in validation and never reaches a top position in testing. In fact, it claims the last testing place in four out of the six plots. At the top positions, Cox and logistic models compete closely: Among the models trained on the Reddy data, a logistic model narrowly secures the pole position in validation. Testing on both the Schmitz and Staiger data reveals how crucial this victory was as the closest validation competitors of m^*_{Reddy} clearly lag behind in terms of the test error. Among the models trained on the

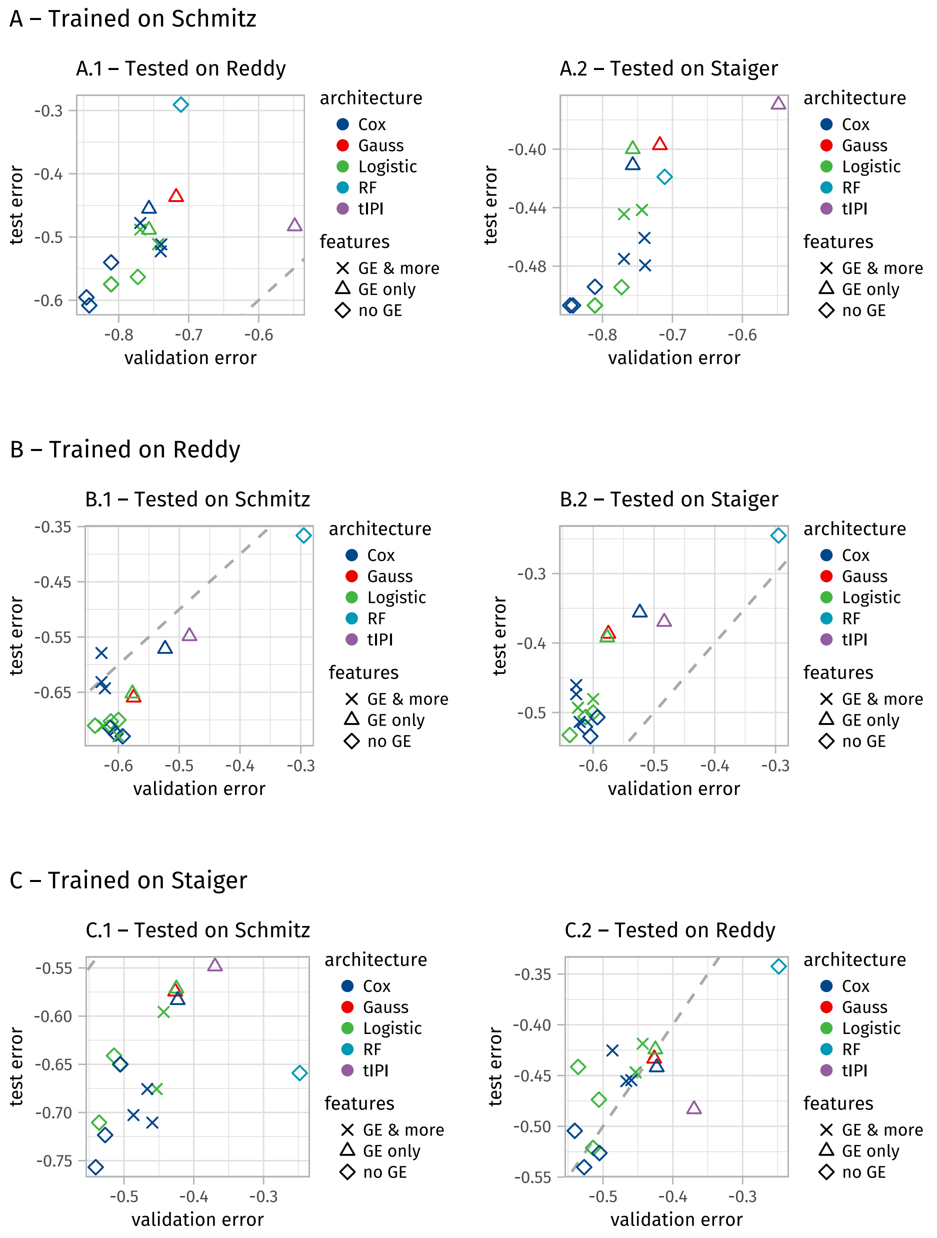

Figure 3.3: Validation versus test error in inter-trial experiments. The dashed gray line is the identity line. The elastic-net regularization strength λ, training survival cutoff T, n_{combi} and the random-forest hyperparameters B, m and n_{combi} have already been tuned in the validation. The validation error for the tIPI is in fact also a test error on independent data, namely our training cohort.

Staiger data, the same holds true with roles of Cox and logistic swapped. This shows that it was important to try out both Cox and logistic models although they are closely related. Thanks to more correlated validation and errors, we need to worry less about overfitted validated predictions and might even consider sending analogous Gauss models into the race.

Regarding the question if the predictor should directly contain the gene-expression levels or not, Fig. 3.3 speaks a plain language. In testing, the top ranking models are always such models that do not use gene-expression levels. Often the test errors of the models whose predictor excludes gene-expression features separate from the rest (cf. plots A.1, A.2, B.1). In validation, however, this distinction can be less obvious, especially for those models trained on the Reddy data (cf. plots B.1, B.2). In summary, we gain nothing in terms of test performance, but risk choosing a low-performing model in the validation. Indeed, the models that make the choice of the best validated model on the Reddy data close and lucky with regard to testing all use gene-expression features.

Output thresholding

For all i, the best validated model m_i^* outputs a continuous score and we need to threshold these scores in a protocol-dependent manner to obtain a binary classifier. As a reminder, the errors we presented so far are all optimized on the respective cohort: the maximum precision under the constraint to have a prevalence of at least 17% for our models and of at least 10% for the tIPI. When given a new data set with blinded outcome, we suggested to threshold a model at the 17%-quantile of its continuous output on this data set. This, by definition, leads to higher test errors than with the optimization procedure. The question is by how much the test error of the optimally thresholded model and the 17%-quantile-thresholded model will differ.

If we rank patients by the scores a good model assigns them, high-risk patients accumulate at the top of the ranking and low-risk patients at the bottom. As a result, as we decrease the threshold, the group of patients with a score above a threshold tends to comprise a lower and lower proportion of high-risk patients. Stated differently, for good models the precision tends to drop as the threshold drops and the relationship between prevalence and precision is roughly monotonic. In the case of a perfectly monotonic dependence between prevalence and precision, the α-quantile would be an optimal threshold with a prevalence of at least α. In the case of a somewhat violated monotonicity, we still have reason to hope that the α-quantile is a near-optimal threshold.

To check this out in practice, we plotted prevalence versus precision for every possibility to threshold m_i^* for all i and test cohorts as long as the prevalence is in $[0.15, 0.50]$, in Fig. 3.4. For a given cohort of n samples, a GLM with a non-zero coefficient β_j for a continuous feature almost surely outputs n different scores. This leads to n possible thresholds if we demand that at least one sample be in the positive group. Indeed, all plots reveal a clearly decreasing overall trend for prevalences $\geq 17\%$. With the exception of plot C.2, this overall trend is never seriously compromised over smaller prevalence intervals. This means, the 17%-quantile of the model output is a near optimal choice in all six cases – as expected.

In plots B.1–2 and C.1–2, the curve falls sharply for prevalences between 10% and 15%. We might wonder if the sharp increase in precision that we observe as we reduce the prevalence from 15% to 10% overcompensates for the lost statistical power that we need to significantly outperform the IPI. Fig. 3.5 answers this question with a clear yes. With a reduced prevalence of 10%, we score two more significant victories over the tIPI, this time on the Schmitz data: an easy one in the case of m^*_{Reddy} and a close one for m^*_{Staiger}.

This makes m^*_{Reddy} a transferable, versatile model that is able to perform in both a representative and high-risk setting for predicting PFS despite being trained to predict OS. Vice versa, for m^*_{Schmitz} and m^*_{Staiger}, the Reddy data stays challenging and the only data set where a best validated model m_i^* fails to significantly surpass the precision of the tIPI.

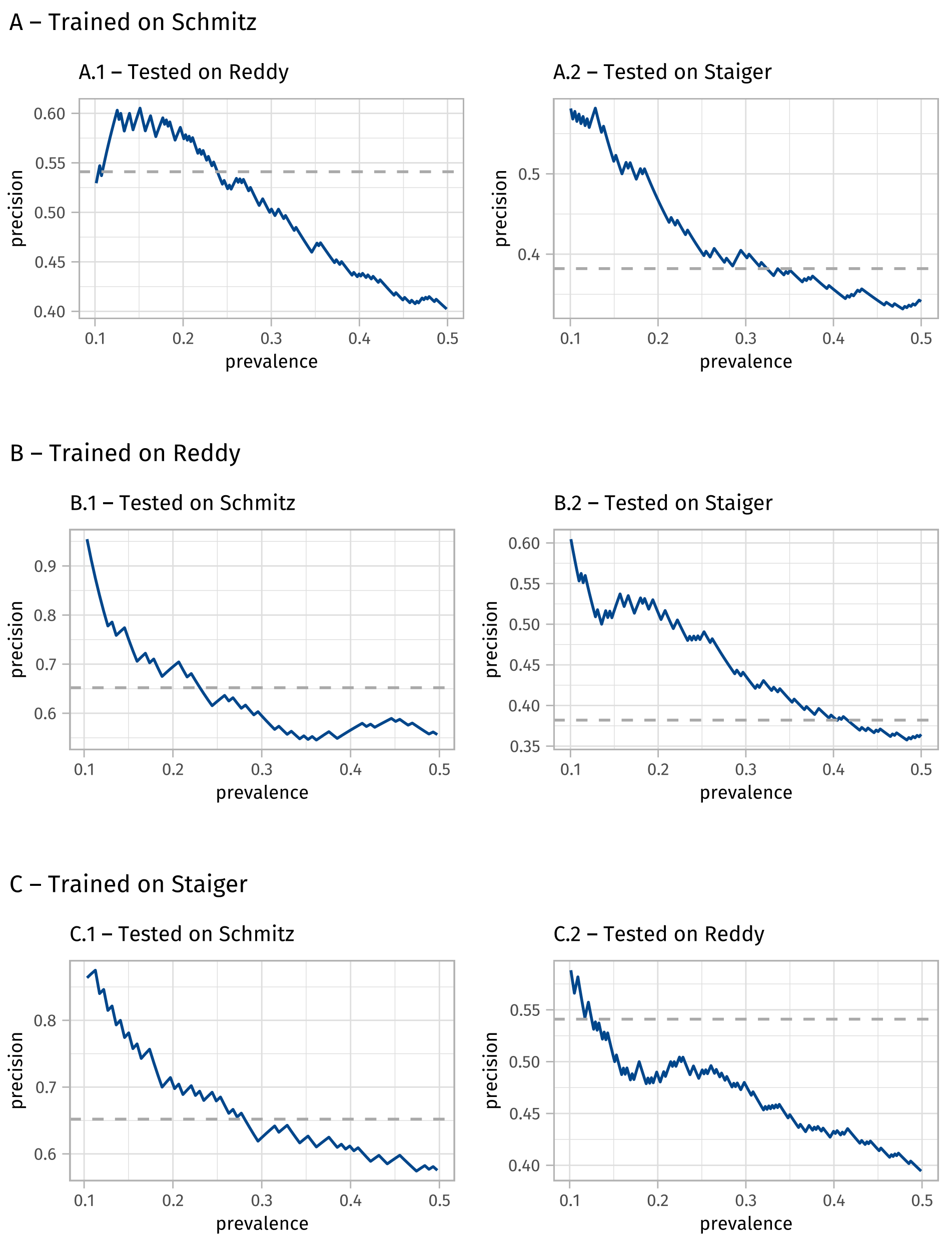

Figure 3.4: Prevalence versus precision for all possible ways to threshold m_i^*, $i \in \{\text{Schmitz}, \text{Reddy}, \text{Staiger}\}$, for a prevalence in $[0.10, 0.50]$ in the inter-trial experiments. i – or the training cohort – corresponds to the rows, the test cohort varies to the columns. The dashed gray line marks the precision of the tIPI on the respective cohort.

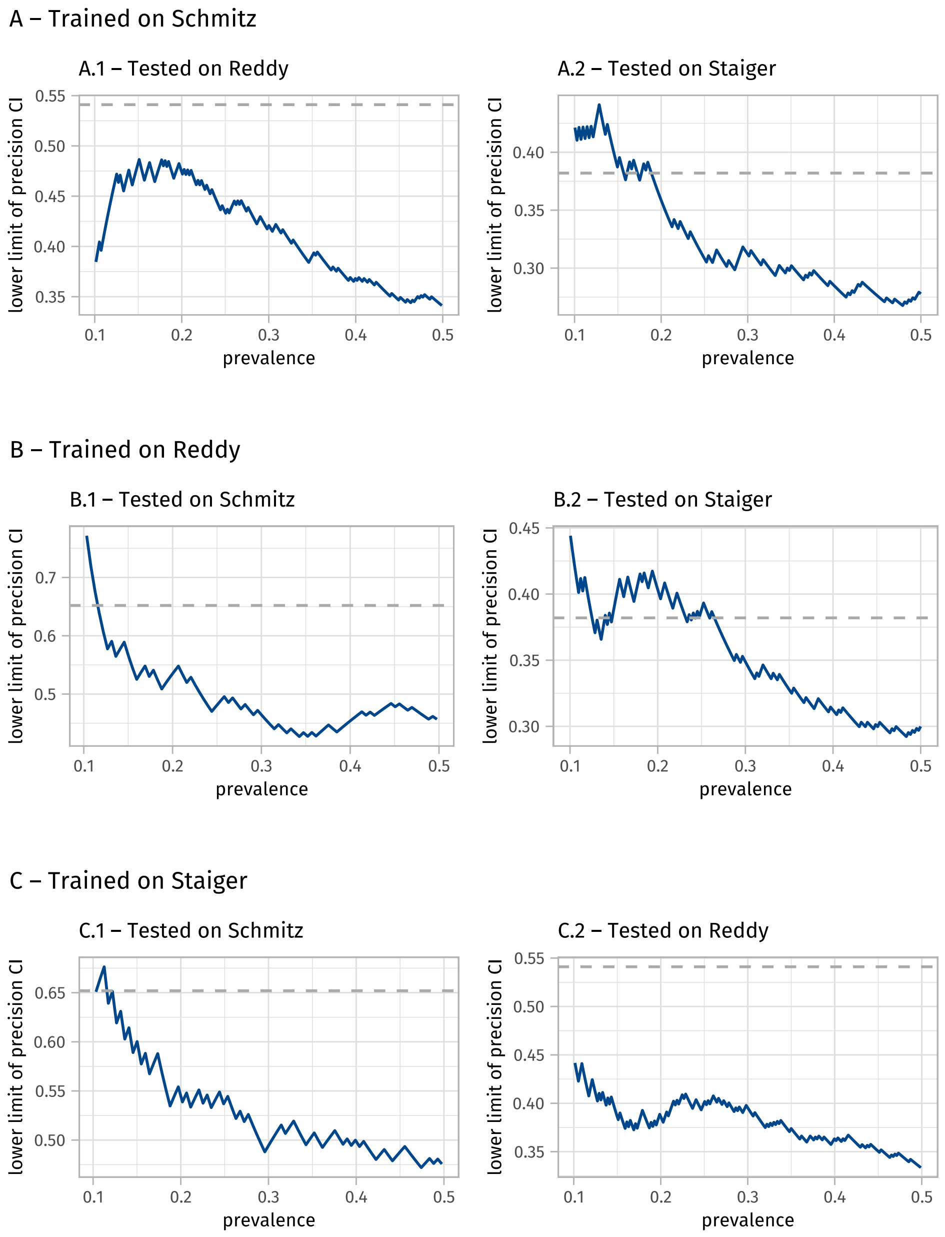

Figure 3.5: Prevalence versus lower limit of the 95%-CI of the precision for all possible ways to threshold m_i^*, $i \in \{\text{Schmitz}, \text{Reddy}, \text{Staiger}\}$, for a prevalence in $[0.10, 0.50]$ in the inter-trial experiments. i – or the training cohort – corresponds to the rows, the test cohort to the columns. The dashed gray line marks the precision of the tIPI on the respective cohort. This figure only differs from Fig. 3.4 in the y-axis.

Chapter 4

Discussion

The previous chapter showed that our best models can significantly outperform the IPI in both a high-risk and representative-risk setting. Moreover, we demonstrated that we can reliably validate our trained models and pick a near-optimal if not optimal one if we restrict the methods from chapter 2 appropriately and increase the sample size.

4.1 Learnings from this thesis

4.1.1 Heuristics for candidate models

Key to ensuring realistic validated errors is excluding from H in the first place those candidate models that sneak in a validated error considerably below their test error and those that perform poorly on test data. In this thesis, we developed a series of heuristics that aim to do precisely that.

Nesting models

We saw that we should nest models predicting from hundreds or even thousands of gene-expression levels together with more features into another model with care. NestedPseudoValidation (cf. Alg. 3), which fits an early and the late model to the same data and uses the cross-validated predictions of the early model to train the late model, delivers way too optimistic validated errors for the nested model. Two things may make this validation approach unreliable. First, the early model, which we usually train on much more features than training samples, has plenty of freedom to overfit and it will exploit this freedom when we tune its hyperparameters in a cross validation. The algorithm of the late model then recognizes the output of the early model as a very predictive feature and is misled to make it a prominent feature in the final model. On independent test data, the input for the late model systematically differs from the training, causing it to generalize poorly. Second, we call Alg. 3 nested *pseudo* validation for a good reason, namely because the model that outputs the validated prediction of a sample saw this sample during its training.

Nevertheless, we also witnessed that OOB predictions for the late model in Alg. 3 are more trustworthy than cross-validated predictions. A reason might be the following: A sample's cross-validated prediction comes from one model and we have one model per sample in the best case, a leave-one-out cross-validation (which we did for all models involved in nested models). In contrast, a sample's OOB prediction comes from a whole forest of models with expected size roughly $B/3$ because the probability of a sample not being in a bootstrap sample is

$$\left(1 - \frac{1}{n}\right)^n \to \frac{1}{e} \approx \frac{1}{3} \quad \text{as } n \to \infty, \tag{4.1}$$

and we can scale it by scaling B, which is cheap as we laid out in subsection 2.2.3. For nested models with random forests as the late model, test errors are well in line with validation errors, but they are also pretty high.

Gene-expression levels in the predictor

In subsection 3.3.2, core models that predict from the full range of gene-expression levels often misled in validation and, in testing, always failed to outperform models that do not predict from any gene-expression levels directly. Considering this and the above-mentioned flaws of NestedPseudoValidation, we should refrain from incorporating gene-expression levels directly into the models trained on our data, but only use their valuable information condensed into the output of already-existent models. We did the latter with the LAMIS and cell of origin. With the curse of high dimensions gone, we observed much more trustworthy validated errors. Furthermore, this reduces the training time and enables us to use more complex models. With gene-expression levels always come batch effects, and we will elaborate on them further below.

More heuristics

The inter-trial experiments suggest that model performance benefits from using a training survival cutoff T that is lower than the time splitting the patients into the high-risk and low-risk group for the ground truth. Random forests, whose OOB predictions proved to yield a very accurate estimate of the test error, fare worse than GLMs in testing. Compared to GLMs, they cannot deal with systemically shifted features, which makes them unsuited to leverage across protocols the nuanced information that the continuous output of gene-expression signatures may bear (cf. subsection 2.2.4).

4.1.2 Data situation

In addition to restricting the hyperparameter-tuple space H carefully, we raised the number of samples by combining three data sets in section 3.3. We hoped to get more representative train and test cohorts, harder-to-overfit train cohorts and, as a result, validation errors that better reflect test errors. Whatever the reason, in the latter we succeeded. To combine the data, we had to make sacrifices. We discarded the features that are not present in all three data sets – and this included valuable features like MYC translocations – and we often trained models to predict PFS and then tested them for predicting OS or vice versa. Still, our best validated models defied this together with technological differences between the data sets and, at roughly 17% prevalence, outperformed the IPI significantly on the prospective Staiger data.

In the inter-trial experiments, with more samples in the test cohort, we could lower the prevalence of our selected models to 10%, drastically raise the precision and still retain enough statistical power to significantly defeat the IPI in even more cases. For the best validated models, we saw an almost perfectly monotonic medium-term relationship between the prevalence and the precision, which made the 17%-quantile or – even better – 10%-quantile of the model output a near-optimal threshold.

This thesis strongly suggests that a sufficiently large sample size is a crucial advantage to solve our problem. Furthermore, we should be ready to make sacrifices – less features, even a different kind of the response – to increase the number of samples in both train and test cohort.

4.2 Applying these learnings to MMML-Predict

With m^*_{Schmitz} and even more m^*_{Reddy}, this thesis presents two models that meet all requirements of MMML-Predict: a prevalence of at least 10% and a precision above 50% and significantly above that of the IPI on an independent, prospective data set.

And still, every percentage point we gain in prevalence will convince more clinicians, researchers and pharmaceutical companies to pay attention to the high-risk group identified by the MMML-Predictor. Every percentage point we gain in precision will not just spur more interest of the above people and companies, but will also convince more patients to let the MMML-Predictor guide their treatment and will avoid both unnecessary and failed therapies. This thesis did not suggest that model performance has already saturated, so we can hope for more in the future.

4.2.1 Combining data sets

In light of subsection 4.1.2, we should strive to increase the sample size. For the final MMML-Predictor, one might thus consider using more than 100 of the 300 samples registered for MMML-Predict, maybe even all 300, for a sufficiently large, statistically more powerful test cohort. One can combine a series of available data sets into a big training cohort. With combining data sets come two caveats.

First, we need to intersect over the sets of features, which amounts to discarding plenty of features. Nevertheless, many modern DLBCL data sets include gender, age, the IPI features in their thresholded format and gene-expression levels as molecular features. More modern data sets also hold the double-hit and triple-hit status, which refers to the simultaneous translocation of MYC and either BCL2 or BCL6, or MYC, BCL2 and BCL6, respectively. From the gene-expression features, we can calculate already-existent molecular signatures as well as the double-expressor and triple-expressor status, which describe the phenomena analogous to double- and triple-hit status with overexpression instead of translocation of the three involved genes.

Second, we need to take care of inter-technical variability. Among the features mentioned above, this concerns the gene-expression levels. The first option is to get rid of protocol effects by once and for all thresholding the output of molecular signatures. Most signatures come with canonical thresholds, such as the 75%-quantile to obtain the LAMIS group, and we can calculate them on the respective data set even before combining the data sets. In this case, we can deploy quite any model as we no longer need to fight the curse of high dimensions or batch effects. The second option is to only apply zero-sum signatures (or scale the gene-expression levels to an ℓ_1 norm of 1 across all samples). Using the output of these signatures as continuous features of a GLM again results in a protocol-dependent shift in the output of the GLM according to our reasoning in subsection 2.2.4 – a reasoning that practice in chapter 3 did not refute. For training, we can add the data set the sample was taken from as a categorical feature to the predictor so we can correct for the data-set-dependent shifts in the loss function. After training, we remove the data-set feature from the model. In testing, we threshold the continuous output of the GLM at the α-quantile for some α, as we did in this thesis. A shortcoming of this testing procedure is that we always need a sufficiently large test cohort to be able estimate the α-quantile reliably. An alternative approach might involve internal standards, i.e. a small number of samples one has measured for a test cohort for which we already know a good threshold and that we can measure again for any new protocol to shift the threshold accordingly. Even for MMML-Predict, this is a problem for the distant future.

4.2.2 Custom gene-expression signatures

In this thesis, we failed to integrate the output of gene-expression signatures that we trained on our data as a feature into another model and obtain a model with both a low validation and test error. In addition to resorting to signatures that *other* people trained on other data, we can train our *own* signatures on other data. The project would then consist of two training data sets. On the first one, we train and validate gene-expression signatures tailored for our problem. We then take the best validated gene-expression signature and write its output as a new feature into the second training data set. We might not just do this for the very best gene-expression signature from the first training set, but for the top k validated gene-expression signatures. On the second training data set, we fit and validate the final models.

4.2.3 Choice of the error function

Validation always depends on decisions we cannot uniquely derive from the problem at hand. While we strive to make them as natural as possible, they remain arbitrary to a certain degree. The most important example for this in this thesis is the error function err we use for validation and testing.

While, for testing, our choice of err is indeed quite natural for our problem and err is easy to interpret, there might be better error functions for validation. E.g., our choice of err is totally unaware of a steep decline in the prevalence-versus-precision curve for prevalences below 17% as we see it in plot B.1 of Fig. 3.4. Defining err as the average negative precision for prevalences in $[a, b]$, with e.g. $a = 0.10$ and $b = 0.17$, takes more information from the prevalence-versus-precision curve into account and, in particular, better catches this phenomenon.

Still, with our choice of err, validated and test errors were pretty much in line in section 3.3. This underscores that, under the conditions we described in section 4.1, our choice of err already is a pretty good one.

4.2.4 A natural way to choose sample weights

Our last idea concerns the sample weights in loss functions. Many loss functions are derived from the log likelihood of i.i.d. samples and thus sum over the training samples. Therefore, they offer to weight every summand with a sample weight, as we have seen in the loss functions of the presented GLMs in Eq. (2.15) and (2.26). One can even provide sample weights for random forests. We have a huge amount of freedom in how we choose the sample weights and trying out too many choices risks torpedoing validation. We will now describe a single, natural choice. Let q denote the proportion of high-risk samples in the training data. In the prospective Staiger data, we had $q = 24.3$, which renders our classification problem imbalanced. Setting $w_i = 1/q$ for high-risk and $w_i = 1/(1 - q)$ for low-risk samples perfectly balances the classification task in the sense that the sum of sample weights belonging to high-risk samples equals the sum of sample weights belonging to low-risk samples.

Looking back at this thesis, we can say with confidence: the state of MMML-Predict is strong and it is getting stronger.

Bibliography

[1] M. Altenbuchinger, T. Rehberg, H. U. Zacharias, F. Stämmler, et al. Reference point insensitive molecular data analysis. *Bioinformatics*, 33(2):219–226, 09 2016. ISSN 1367-4803. doi: 10.1093/bioinformatics/btw598.

[2] M. Altenbuchinger, P. Schwarzfischer, T. Rehberg, J. Reinders, et al. Molecular signatures that can be transferred across different omics platforms. *Bioinformatics*, 33(14): i333–i340, 07 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx241.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. doi: 10.1201/9781315139470.

[4] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. doi: 10.1023/A:1010933404324.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. *Biometrics*, 40:874, 1984. doi: 10.1201/9781315139470.

[6] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 12 1934. ISSN 0006-3444. doi: 10.1093/biomet/26.4.404. URL `https://doi.org/10.1093/biomet/26.4.404`.

[7] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972. doi: https://doi.org/10.1111/j.2517-6161.1972.tb00899.x. URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1972.tb00899.x`.

[8] S. Durinck, P. T. Spellman, E. Birney, and W. Huber. Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. *Nature Protocols*, 4:1184–1191, 2009. R package version 2.60.0.

[9] J. H. Friedman, T. J. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33 1:1–22, 2010. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2929880/`.

[10] L. Gessl. patroklos: An R package pipelining binary classification of survival, 2024. URL `https://github.com/lgessl/patroklos/`. R package version 0.5.1.

[11] L. Gessl. Computational detection of a high-risk DLBCL group, 2024. URL `https://github.com/lgessl/master-thesis`.

[12] B. Glass, A. Dohm, L. Truemper, M. Pfreundschuh, A. Bleckmann, G. Wulf, A. Rosenwald, M. Ziepert, and N. Schmitz. Refractory or relapsed aggressive B-cell lymphoma failing (R)-CHOP: an analysis of patients treated on the RICOVER-60 trial. *Annals of Oncology*, 28(12):3058–3064, 2017. ISSN 0923-7534. doi: https://doi.org/10.1093/annonc/mdx556. The antibody-drug conjugate target landscape.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.

[14] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. ISSN 00401706. URL `http://www.jstor.org/stable/1267351`.

[15] J. P. Klein and M. L. Moeschberger. *Survival Analysis: Techniques for Censored and Truncated Data*. Springer New York, 2 edition, 2003. ISBN 978-1-4419-2985-3. doi: 0.1007/b97377. URL `https://api.semanticscholar.org/CorpusID:122040187`.

[16] N. Mantel. Evaluation of survival data and two new rank order statistics arising in its consideration. *Cancer chemotherapy reports*, 50(3):163—170, March 1966. ISSN 0069-0112. URL `http://europepmc.org/abstract/MED/5910392`.

[17] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972. doi: https://doi.org/10.2307/2344614. URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2344614`.

[18] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2024. URL `https://www.R-project.org/`. Version 4.4.1.

[19] A. Reddy, J. Zhang, N. S. Davis, A. B. Moffitt, et al. Genetic and functional drivers of diffuse large B cell lymphoma. *Cell*, 171(2):481–494.e15, 2017. ISSN 0092-8674. doi: 10.1016/j.cell.2017.09.027.

[20] T. Rehberg. *Zero-sum regression: scale-invariant molecular data analysis*. PhD thesis, University of Regensburg, 2018.

[21] T. Rehberg. zeroSum: R package for elastic net regularized regression with zero sum constraint, 2024. URL `https://github.com/rehbergT/zeroSum`. Version 2.0.7.

[22] R. Schmitz, G. W. Wright, D. W. Huang, C. A. Johnson, et al. Genetics and pathogenesis of diffuse large B-cell lymphoma. *New England Journal of Medicine*, 378(15): 1396–1407, 2018. doi: 10.1056/NEJMoa1801445.

[23] L. H. Sehn and G. Salles. Diffuse large B-cell lymphoma. *New England Journal of Medicine*, 384(9):842–858, 2021. doi: 10.1056/NEJMra2027612. URL `https://www.nejm.org/doi/full/10.1056/NEJMra2027612`.

[24] A. M. Staiger, M. Altenbuchinger, M. Ziepert, C. Kohler, et al. A novel lymphoma-associated macrophage interaction signature (LAMIS) provides robust risk prognostication in diffuse large B-cell lymphoma clinical trial cohorts of the DSHNHL. *Leukemia*, 34(2):543–552, 2020. doi: 10.1038/s41375-019-0573-y.

[25] M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 12 1974. doi: 10.1111/j.2517-6161.1974.tb00994.x. URL `https://doi.org/10.1111/j.2517-6161.1974.tb00994.x`.

[26] The International Non-Hodgkin's Lymphoma Prognostic Factors Project. A predictive model for aggressive non-hodgkin's lymphoma. *New England Journal of Medicine*, 329(14):987–994, 1993. doi: 10.1056/NEJM199309303291402. PMID: 8141877.

[27] T. M. Therneau. *A Package for Survival Analysis in R*, 2024. URL `https://CRAN.R-project.org/package=survival`. R package version 3.7.0.

[28] R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 12 2018. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1996.tb02080.x. URL `https://doi.org/10.1111/j.2517-6161.1996.tb02080.x`.

[29] M. N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01. R package version 0.16.2.

[30] M. Ziepert, D. Hasenclever, E. Kuhnt, B. Glass, N. Schmitz, M. Pfreundschuh, and M. Loeffler. Standard international prognostic index remains a valid predictor of outcome for patients with aggressive CD20+ B-cell lymphoma in the rituximab era. *Journal of Clinical Oncology*, 28(14):2373–2380, 2010. doi: 10.1200/JCO.2009.26.2493. URL `https://doi.org/10.1200/JCO.2009.26.2493`. PMID: 20385988.

[31] H. Zou and T. Hastie. Regularization and Variable Selection Via the Elastic Net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 03 2005. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2005.00503.x. URL `https://doi.org/10.1111/j.1467-9868.2005.00503.x`.

Declaration of authorship

Ich habe die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und bisher keiner anderen Prüfungsbehörde vorgelegt. Außerdem bestätige ich hiermit, dass die vorgelegten Druckexemplare und die vorgelegte elektronische Version der Arbeit identisch sind, dass ich über wissenschaftlich korrektes Arbeiten und Zitieren aufgeklärt wurde und dass ich von den in § 26 Abs. 5 vorgesehenen Rechtsfolgen Kenntnis habe.

Regensburg, August 17, 2024