2.5 Survival Prediction and Model Selection

Jörg Rahnenführer
Michel Lang
Jakob Richter

Abstract: Survival analysis comprises statistical methods for time-to-event data. The main prediction tasks include the estimation of the influence of prognostic factors for, say, medical treatments, and the modelling and prediction of survival times using regression models. In recent years, in molecular medicine, many omics technologies have been developed, generating complex high-dimensional genetic data that can be used as predictors.

For such complex tasks, the selection of the best prediction method out of a large set of candidates, along with potential feature selection and hyperparameter optimization, represents an optimization task under resource constraints. In this section, approaches for tackling the model selection problem in survival analysis are presented, specifically using Bayesian optimization and addressing feature selection for high-dimensional data.

2.5.1 Introduction

In medicine, times to events are compared between groups to estimate the effect of prognostic factors and medical treatments, and regression models are used to model and predict survival times of cells, animals, or patients. For two decades, high-dimensional genetic and genomic variables have been generated and analyzed as potential predictive and prognostic factors in biological and medical scenarios. The very large number of variables requires developing and using tailored methods to describe the complex relationships. Popular modeling approaches are based on penalized regression methods, gradient boosting methods, survival trees, and survival forests, often combined with suitable feature selection methods.

In recent years, machine learning approaches were used to find the best survival method from a large set of candidates. Efficient approaches are required, since it is crucial that runtimes especially in resampling scenarios with many repeated estimation tasks be kept short, especially for complex high-dimensional predictor settings. In CRC 876, we applied modern Bayesian optimization (BO) [303] techniques to efficiently identify the best survival prediction method, by modeling the relationship between the choice of the survival prediction method (as well as its hyperparameters) and its performance or quality, using so-called surrogate functions. On several lung cancer datasets the new approach was superior to established benchmark approaches [367,

369]. After a short introduction into the analysis of time-to-event data in Section 2.5.2, model selection for survival analysis is discussed in Section 2.5.3. To solve this task, various R packages were implemented both for the general candidate selection and for parallelization, as presented in Section 2.5.6.

The same principle idea was used in a scenario, where survival predictions for a specific cancer dataset are to be improved, by adding data from similar datasets. This is a frequent situation in, say, cancer survival analysis, where patient numbers in clinical trials are limited due to ethical, financial, and administrative reasons, but similar treatments are applied, e.g., in other clinical centers. However, simply adding similar datasets to the one of interest potentially deteriorates the predictions, due to structural differences between the datasets. Instead, one can estimate dataset-specific weights that determine how strong these datasets should be considered. In CRC 876, we applied BO to determine such optimal weights for inclusion of the respective observations in appropriate weighted likelihood-based modelling approaches [531], as shown in Section 2.5.4 below. In two other projects related to feature selection, we developed improved methods based on two-fold subsampling schemes [383] and benchmarked filter methods against each other for high-dimensional data [95]. These analyses are described in Section 2.5.5.

2.5.2 Analysis of Time-to-Event Data

Survival analysis, also called event-time analysis, deals with the analysis of times to certain events and is used in many application fields. In medicine, the overall survival (OS) of patients is often of direct interest. Alternatively, Progression-Free Survival (PFS) is frequently analyzed, which includes Event-Free Survival (EFS) and recurrence-free survival. An important property of survival data is that they are often not fully observable, such as when patients in a clinical trial have not yet experienced the event of interest at the time the trial ends and the data is analyzed. This situation is called right-censoring, since for patients without an observed event, the survival time must be greater than or equal to the time until the end of the study. Depending on the type of missing information, many other censoring mechanisms are defined and considered in the analysis techniques.

Specialized statistical methods for analyzing survival data have been developed and are widely used in literature and in practice. Most prominent are the Kaplan-Meier estimator for estimating survival curves under right-censoring, the log-rank test for comparing survival between patient groups, and the Cox proportional hazards model [149] for estimating survival dependent on a number of explanatory variables, such as tumor size or age in oncological studies.

In regard to the evaluation of performance, seemingly obvious approaches lead to wrong interpretations of the results. For example, simply predicting the event indicator that indicates if a patient has survived until the end of the study, neglects the different

time intervals that have passed since the patients entered the study. By contrast, methods based on hazard rates that model the instantaneous failure rates at different time points can cope with the censoring mechanisms. Alternatively, parametric likelihood methods that consider the missing information can also be used.

For evaluating the prediction accuracy of survival models, several suitable measures have been developed. Concordance statistics, in particular Harrell's C-index and the area under the (time-dependent) ROC curve, are the most popular measures. However, they consider only the discrimination ability of a survival model and not the calibration. This means that monotone transformations of predicted values of survival outcomes do not change the concordance score, which limits the interpretability of the score for clinicians. Alternatively, the Brier score is also widely used. It considers both calibration and discrimination, but interpretation is also difficult. An advantage is that it can be related to a time-specified horizon. A discussion of these important properties and an adaptation of the Brier score can be found in Kattan and Gerds [313].

In preclinical and clinical studies, genetic factors are of interest, and modern high-throughput technologies provide many thousand potential explanatory variables. Even the popular but controversial rule of thumb that the number of events per variable should be at least 10 cannot be used as a basis for sample size planning. Instead, tailored statistical and machine learning approaches are required. Aspects to consider for model selection in this scenario are discussed in the next subsection.

2.5.3 Model Selection for Survival Analysis

Model selection in survival analysis, compared with model selection in classical machine learning setups such as regression or classification, presents numerous additional challenges.

First, instead of having to solve a learning task with many observations (e.g. patients) and comparatively few variables, in survival analysis we often face a low sample size problem. Even worse, with the rise of omics technologies, thousands to hundreds of thousands of genetic features need to be included in the analysis to be able to identify the most important genes. However, most machine learning or statistical learning algorithms have been designed and heavily optimized for a large sample size n, and usually have worse than quadratic runtime in the number of features p. For this reason alone, we often face runtime issues in the $n \ll p$ scenario.

Second, a dual objective is often pursued, and the predictive performance of the models is not the only target criterion. Instead, it is desired to identify the important features (clinical covariates, genetic dispositions, or genes) in the given medical context. A good predictive performance often ensures that the model describes the data in a meaningful way, which is the prerequisite for extracting a set of important features. This restricts the analysis to models that either come with an embedded feature selection or models that still work reasonably well after a feature filter has been applied.

Third, all performance measures in survival analysis require a large enough test set to yield performance values that lead to reliable statements. For example, the popular C-index mentioned above assesses the ranking of the predictions for survival in the test set by comparing it with the true, observed ranking of survival times (while correcting for censoring). Obviously, having too few observations in the set results in a high variance of the performance estimator. Since usually only a few hundred observations are available in total per dataset, the number of repetitions must be increased during resampling in order to account for the larger variance. Of course, this exacerbates the runtime problems one is already facing.

These points taken together form a hard tuning problem with the following characteristics:

1. The models form a black box from the perspective of the tuner, as there are no known derivations. Therefore, the optimization problem itself is also called "black box".
2. To assess the predictive performance of a hyperparameter configuration θ and its resulting model, the data needs to be split into a training set and an independent test set. This introduces stochastic components into our tuning problem at the latest (some learners are non-deterministic either way).
3. The search space spanned by the hyperparameters to be tuned usually includes both numerical and categorical variables. This precludes the use of many tuners derived from discrete and steady optimization.
4. Each model fit is potentially resource demanding, in terms of computational time, memory requirements, or communication costs. The key word here is "potentially". Some models, e.g., a simple Cox model augmented with an aggressive feature filter, can easily be fitted in less than a minute even with $n = 200$ observations and $p = 10^6$ variables (features). Other learners, such as support vector machines, require a complete day for the same task on the same hardware while simultaneously consuming several orders of magnitude more memory. Obviously, the resource requirements are very heterogeneous, which should be taken into account during the mandatory parallelization.
5. Last but not least, during hyperparameter optimization, we generally have to deal with an additional type of censoring (besides the censoring of the survival times): It is not unusual that the learner implementations crash from time to time due, say, to numerical problems. And since the tuning is usually distributed on larger computation sites with shared and contested resources, computational jobs can hit a wall time and be killed by a scheduler. In such a case, the missing performance score must be imputed with a number to continue with the tuning, and it is unclear which value to choose.

Over the last decade, special strategies addressing the difficulties of hyperparameter optimization have emerged. An overview is given by Bischl et al. [68]. Roughly speaking, hyperparameter optimization is about finding the configuration θ of a model, which

leads to the best predictive performance (evaluated on an independent test set). If the evaluation of a single configuration is sufficiently expensive with regard to computational resources like runtime, every evaluation counts, which also means that rather wasteful optimization methods are not applicable. This applies, among others, to Evolutionary Algorithms (EAs). EAs usually require many hundreds of configurations before being able to make the first targeted decisions. Instead, a tuner that optimizes more aggressively from the start is needed.

One tuner that addresses all the problems of the outlined expensive black-box optimization problem is *iterated F-racing* [421]. The basic idea of F-racing is to race a population of configurations against each other, and to eliminate in each iteration candidates that are underperforming based on a Friedman test. Iterated F-racing extends this approach by assuming a probability distribution over the search space. This distribution gets updated iteratively so as to be centered around some elite configurations. We applied this tuning approach to a broad range of survival pipelines (consisting of the feature filter and the survival model) [369]. The benchmark considers 12 different datasets of four breast cancer cohorts where each dataset consists of clinical and/or genetic variables (features). The architecture of the pipeline, i.e. the choice of filter and the choice of model, is encoded as virtual hyperparameters passed to the tuner. This way, dominated combinations of filters and models are getting fewer evaluations, giving the tuner more opportunity to exploit hyperparameters of more promising combinations. As a baseline, four reasonable approaches that are popular in practice but are arguably less computationally intensive have been evaluated. To the best of our knowledge, this was the largest benchmark of survival models up to that point. In comparison with the baseline approach, the tuning yields significantly better results in terms of the C-index. The caveat is the effort to archive the results: with more than 10 000 hyperparameter evaluations, the tuning cannot be applied easily on new data or cohorts.

Another tuner which perfectly fits the requirements of hyperparameter optimization in survival analysis is Model-Based Optimization (MBO). Its performance has been verified by Lang [367] where the benchmark study from Lang, Kotthaus, Marwedel, Weihs, Rahnenführer, and Bischl [369] has been extended, with more datasets, more filters, more models, and more time budget.

Figure 2.24 visualizes the survival probability in the included cohorts. Although the studies are all on lung cancer, and share the same set of clinical and genetic features, they differ considerably with respect to survival times. This is a frequently observed characteristic, and makes the careless merging of the datasets into a larger dataset with more observations inappropriate. As a result, in this domain, it is usually not possible to configure a single model to perform sufficiently well on all cohorts. Instead, for each cohort, tuning starts from zero. One goal of the analysis was to thin out the portfolio of methods to consider for a new tuning run. If, e.g., only two pipelines consisting of a filter and a model have to be tried, the computational effort required for tuning is significantly reduced, rendering the tuning for new cohorts on a single computer

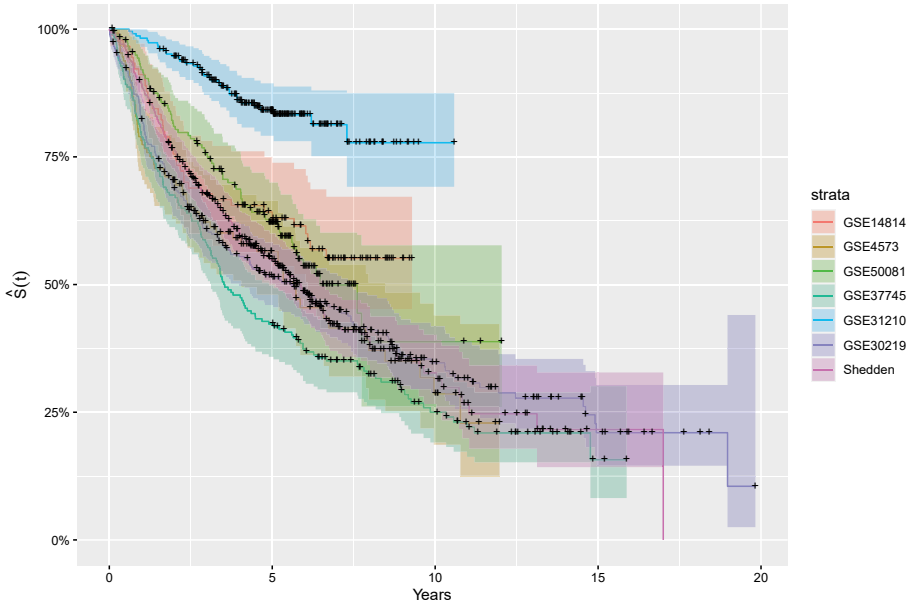

Fig. 2.24: Plot of Kaplan-Meier estimators including confidence bounds for the survival time $S(t)$, stratified by the cohorts that are included in [367]. In the Kaplan-Meier plots, the time t is plotted against the estimated proportion of patients still alive at t. Lines represent survival curves of the seven cohorts. A vertical drop in a curve indicates an event, and a plus on a curve means that an observation was censored at this time.

possible and therefore applicable for practice. This has been systematically analyzed by Lang [367]. Parts of the results are summarized in Figure 2.25. Additionally, the mean ranks of filters and learners have been analyzed and revealed the following important take-home messages in the context of the datasets analyzed:

- If one base learner has to be chosen, random survival forests perform best on average.
- One of the most popular approaches due to its embedded feature selection–fitting a Cox proportional hazards model with a LASSO penalty (L_1)–performs the worst on average.
- Tuning over multiple base algorithms jointly with MBO results in the best performance on average.
- Tuning each pipeline individually and picking the best performing pipeline (approach BenchOpt in Figure 2.25) in a second step is not only a waste of computational resources; it also leads to overoptimistic performance estimates. As each tuning run is stochastic, and the pipelines often perform comparably well, picking the best configuration is determined by the stochastic noise to some degree. This

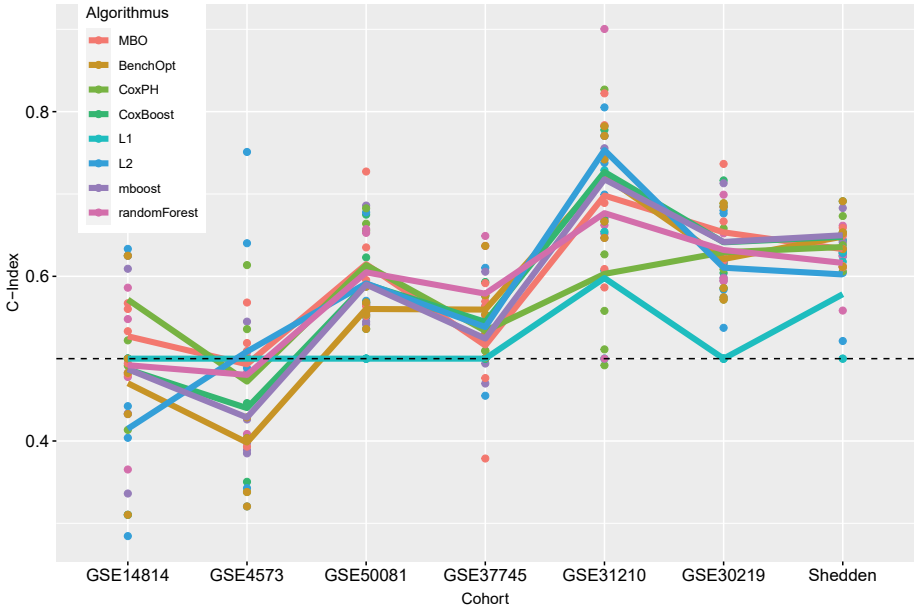

Fig. 2.25: Resulting average C-index on independent test sets of multiple algorithms, stratified by cohort. All base algorithms are individually tuned together, jointly with the choice of filter and the filter's hyperparameters. MBO tunes over all filters and algorithms simultaneously. BenchOpt expresses the resulting performance on an independent test set after picking the best performing base algorithm on the training data [367].

is in particular a very alarming result, as the described manual benchmarking is common practice.

The heterogeneous runtimes (or more general, the heterogeneous resource demands) have been addressed by Richter, Kotthaus, Bischl, Marwedel, Rahnenführer, and Lang [530] and Kotthaus et al. [346]. Instead of fitting only a single surrogate model, guiding the optimization to areas with the best predictive performance, multiple surrogate models are fitted in each iteration. On the one hand, the usual surrogate based on the observed predicted performance is calculated. On the other, one or more surrogate models for computational resources are fitted, e.g., one surrogate for the runtime and one surrogate describing the memory consumption. As a result, we can query the models for the estimated predictive performance and the estimated resource demands for all hyperparameter configurations. All the information is fed into a scheduler that selects a subset of the configurations and maps them to multiple CPUs or workers based on their priority (as derived from the estimated predicted performance) while minimizing the idle times (based on the estimated runtime).

2.5.4 Weighted Subgroup Selection for Survival Analysis

Obtaining a reliable prediction model for a specific cancer subgroup or cohort $s^\star$ is often difficult due to a limited sample size and, in survival analysis, due to potentially high censoring rates. Sometimes similar data from other patient subgroups is available, e.g., from other clinical centers. Simple pooling of all subgroups can decrease the variance of the predicted parameters of the prediction models, but also increase the bias due to heterogeneity between the cohorts.

Different approaches exist to improve the predictive quality by including data from other patient subgroups in a weighted fashion. One possible way is to include one further weighted subgroup, as proposed by Weyer and Binder [726]. Alternatively, individual weights for each patient can be estimated from the training data, as described by Bickel et al. [63]. The idea is that weights match the joint distribution of the combined data to the distribution in each subgroup, such that a patient who is likely to belong to the target subgroup receives a higher weight in the subgroup-specific model. Weights correspond to the conditional probability of belonging to the target subgroup $s^\star$ given the observed covariates and outcome divided by the prior probability for $s^\star$. The former is estimated from the training data by multi-class classification, and the latter by the relative frequency of $s^\star$.

The goal is to optimize the predictive performance of our model for the target subgroup $s^\star$. Including data from additional subgroups in the training data should increase the predictive performance of the target subgroup. Accordingly, this forms a combinatorial optimization problem where additional subgroups must be chosen to maximize the predictive performance.

However, completely abstaining from using certain subgroup data seems overly drastic since there might be relevant information contained in each additional subgroup data. Luckily, most machine learning methods and also those that can be used for time-to-event data allow observational weights. This allows us to give a low weight to observations that do not represent our problem. However, finding an optimal weight for each observation is exceedingly complex. Instead, we introduce subgroup weights as presented by Richter et al. [531]. The observation weight is then determined by the subgroup membership of each observation. This enables the inclusion of certain subgroups with a specific weight. Hence, including additional data in a weighted way might lead to a better solution than the binary choice of including a subgroup with full weight or not at all.

By introducing subgroup weights, we relaxed the combinatorial problem into a numerical optimization problem. However, setting those subgroup weights in an optimal way remains a difficult optimization problem. First, each additional subgroup leads to a further weight parameter that has to be chosen optimally. Second, the evaluation of a weight parameter combination can take fairly long, since the datasets themselves tend to be rather large, especially when they include high-dimensional genetic mea-

surements in the scenario of survival analysis. In this case, it becomes infeasible to try out many weight parameter combinations in order to find an optimal one.

Therefore, we can apply state-of-the art optimization methods for expensive black-box problems such as MBO (model-based optimization) in order to find the optimal subgroup weights without the cost of having to evaluate hundreds of different weight parameter combinations. For our evaluation, we optimize the subgroup-specific weights $w^{(g)}$ in the weighted Cox model. Note that in Section 2.5.3 a study was reported where random survival forests performed on average better than fitting a Cox proportional hazards model with a LASSO penalty. However, here we use the much more frequently used penalized regression approach to evaluate the potential improvement due to the weighted analysis.

Weighted Cox Model Assume the observed data of the patient i consists of the tuples (t_i, δ_i), the covariate vectors $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^{'} \in \mathbb{R}^p$, and the subgroup membership $s_i \in \{1, \ldots, S\}$ with S being the total number of available subgroups, and $i = 1, \ldots, n$. t_i denoting the observed time of patient i, the minimum of the event time, and the censoring time. δ_i indicates whether a patient experienced an event ($\delta_i = 1$) or was (right-)censored ($\delta_i = 0$). As mentioned above, one of the most popular regression models in survival analysis is the Cox proportional hazards model [149]. It models the hazard rate $h(t|\boldsymbol{x}_i)$ of a patient at time t as

$$h(t|\boldsymbol{x}_i) = h_0(t) \cdot \exp(\boldsymbol{\beta}^{'} \boldsymbol{x}_i) = h_0(t) \cdot \exp\left(\sum_{j=1}^{p} \beta_j x_{ij}\right),$$

where $h_0(t)$ is the baseline hazard rate, and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^{'}$ is the unknown parameter vector. The parameters are estimated by maximizing the partial log-likelihood [326, Chapter 8.3]. A version of the partial log-likelihood that uses observation weights is introduced in [726]:

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \delta_i w_i \left(\boldsymbol{\beta}^{'} \boldsymbol{x}_i - \ln \left[\sum_{k=1}^{n} \mathbf{1}_{(t_i \leq t_k)} w_k \exp\left(\boldsymbol{\beta}^{'} \boldsymbol{x}_k\right) \right] \right). \tag{2.18}$$

Instead of an individual weight for each patient, we introduce an individual weight for each subgroup. Therefore, we assign the same weight to each patient of the same subgroup:

$$w_i = \begin{cases} 1, & \text{if } s_i = s^{\star} \\ w^{(g)}, & \text{if } s_i = g,\ g \in \{1, \ldots, S\} \setminus s^{\star} \end{cases} \tag{2.19}$$

where $w^{(g)} \in [0, 1]$ is the specific weight for the subgroup g, and $s^{\star}$ is the subgroup for which we want to obtain predictions. Patients for subgroup $s^{\star}$ enter with full weight 1 in the prediction model.

Standard subgroup analysis is based only on the patients in the subgroup of interest (target subgroup $s^\star$), which corresponds to $w_i = 0$ for all patients not belonging to $s^\star$. A combined model that pools patients from all subgroups corresponds to $w_i = 1$ for all patients.

In high-dimensional settings where the number of covariates p is typically much larger than the sample size n, standard maximum likelihood cannot be used for parameter estimation, since it does not result in a unique solution. Therefore, we add a LASSO penalty [677] to the partial log-likelihood. Lasso regression performs feature selection and yields a sparse model solution. The resulting maximization problem of the penalized partial log-likelihood is given by

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}} \left\{ l(\boldsymbol{\beta}) - \lambda \cdot \sum_{j=1}^{p} |\beta_j| \right\}.$$

The LASSO penalization parameter λ is optimized through an internal 10-fold cross-validation.

Evaluation We are interested in maximizing the predictive performance for a target subgroup $s^\star$. The predictive performance of the weighted Cox model is evaluated using the C-index. For the evaluation of the model for a given target subgroup $s^\star$, a dataset that contains S subgroups, and a subgroup weight vector $w = (w^{(1)}, \ldots, w^{(S-1)})$, we conduct a modified 10-fold cross-validation. The validation data should only contain the target subgroup, because we are only interested in the predictive performance on the target subgroup. In order to obtain the 10 necessary splits for the cross-validation, we only divide the data of the target subgroup into 10 chunks. To obtain the prediction for one chunk, all remaining 9 chunks plus all observations from the additional subgroups are combined to the training dataset. By performing the modified cross-validation, we obtain an estimation on the C-index for the given combination of dataset, target subgroup and subgroup weight vector.

Now, the goal is to find the subgroup weight vector that maximizes the C-index. This optimization problem can be solved with MBO, with a search space $[0, 1]^{S-1}$ that directly maps to the weight vector. The acquisition function that selects the next weight vector to be evaluated should take into consideration that results are not deterministic. Therefore, we proposed the augmented expected improvement [288], which is well suited for such scenarios. For the Gaussian process regression within the Bayesian optimization, we proposed the Matern 3/2 kernel with an estimated *nugget effect* to account for the noisy response of our objective.

The benefit of optimizing the subgroup weights is twofold: First, the resulting optimal subgroup weight vector does not only maximize the C-index for the target subgroup; it also allows drawing conclusions about the similarity of certain subgroups. If a certain subgroup weight is small, it can be assumed that this subgroup does not have a similar relationship between the explanatory variables and the event times as

the target subgroup. Second, as shown by Richter et al. [531], the predictive performance of the method does not deteriorate if additional subgroups are included that contain inconsistent data.

Benefits could arise from using the penalization of the weights, which would allow researchers to completely exclude data with weights close to zero. Then the model becomes computationally cheaper and possibly more stable. Finally, the presented approach can be used for any situation where data is pooled from different cohorts and a machine learning method is used that supports observational weights.

2.5.5 Feature Selection for High Dimensional Data

The problem of feature selection is particularly important in the domain of high dimensional data, as already described in detail in Section 2.5.3. One challenging problem in this context is the stability of feature selection. Some learners can be restricted to using only a small subset of the thousands of available features, and learners can be combined with a feature filter to achieve the same in a generic fashion. However, the set of selected features is often highly variable. For example, if a Cox proportional hazard model is extended with an L_1 penalty λ tuned to include only up to 20 features in a 3-fold cross-validation, the resulting three sets of selected features can be pairwise disjoint. This has a simple reason: if two features x_1 and x_2 are highly correlated, they are also comparably good predictors. If the model has to choose between x_1 and x_2, a few observations can tip the scales in one direction or the other. If the dataset is now resampled and these observations are removed from the training set, the scale can easily swing in the other direction. This is particularly annoying because in this way no features can be reliably selected for a later analysis, such as a biological analysis.

Lee et al. [383] tackle this problem in two ways: First, a special extension to the LASSO regression is used. The preconditioned LASSO [495] is a two-step procedure designed to address the problems of high bias in LASSO estimates. Second, the preconditioned LASSO is embedded in a two-fold subsampling procedure to improve the stability of the feature selection via model averaging and extra shrinking of covariates based on the selection probability in the inner subsampling.

The approach has been applied to datasets on neuroblastoma, lung adenocarcinoma, and breast cancer. Both predictive performance (measured by the C-index) and stability (measured by the Jaccard index and the Kuncheva index) are improved. However, the comparison with popular univariate selection methods does not provide a clear picture.

Another take on this topic was presented by Bommert et al. [95], where more than 20 filter methods are benchmarked against each other for high-dimensional data. Although this work is based on classification data, there is no reason why the core results should not be transferable to survival problems, and confirming this is currently work in progress. One key result is shown in Figure 2.26. There are clear groups of

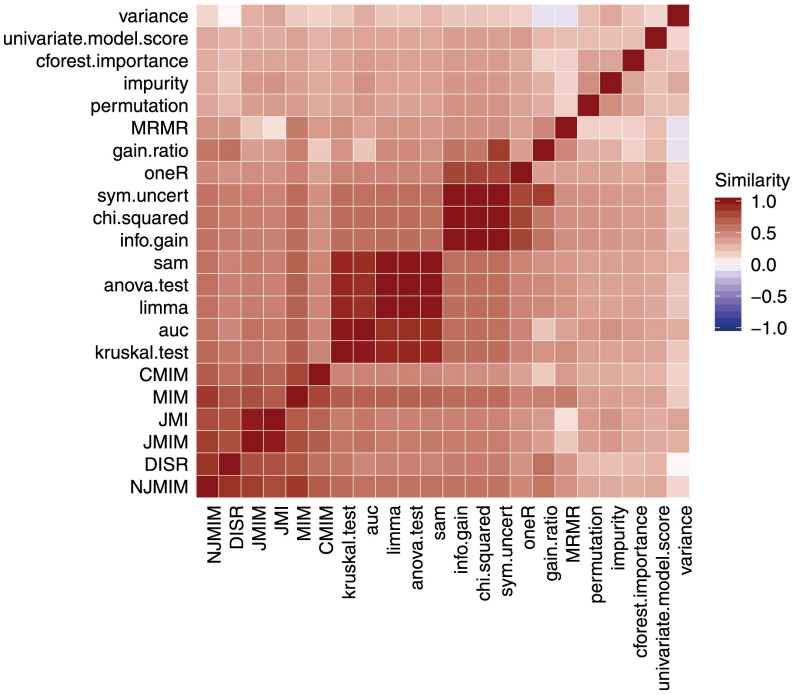

Fig. 2.26: Rank correlations between the feature selection order, for all pairs of a large set of filter methods, averaged across several datasets by the arithmetic mean. The filter methods are ordered by average linkage hierarchical clustering using the mean rank correlation as a similarity measure [95].

feature filters available. Filters from the same group are expected to give very similar results across different datasets. Therefore, instead of including more than 20 filters into the machine learning pipeline, it is sufficient to thin out this portfolio to a smaller set. Additionally, the filters have been analyzed regarding performance and stability to provide general recommendations for feature filtering in high-dimensional settings.

2.5.6 Software

Many machine learning frameworks exist that can be conveniently employed for model selection or feature selection. However, most of these frameworks have a strong focus on classification and regression. Support for survival analysis is often not existent or insufficient. For proper evaluation, two frameworks have been extended to support time-to-event data.

First, the R package `mlr` [69] has been extended with an object for survival tasks, including the most common survival learners and survival measures. By building upon

the existing infrastructure for resampling and tuning, survival learners can be tuned with state-of-the-art tuners such as model-based optimization. For larger tasks, i.e. tasks with thousands of features of genetic data, parallelization of the benchmark experiments is mandatory. The package `BatchJobs` [67] and its successor `batchtools` [368] provide the bridge between `mlr` and managed high-performance computing clusters, allowing to compute comprehensive benchmarks on hundreds of CPUs simultaneously. In this way they can define and execute exhaustive benchmark studies, such as those from Lang, Kotthaus, Marwedel, Weihs, Rahnenführer, and Bischl [369], Lang [367] and Richter et al. [531].

The second framework extended for survival analysis is `mlr3` [370], the successor of `mlr`. The extension package `mlr3proba` [640] provides a general framework for probabilistic regression. Compared with the survival capabilities of `mlr`, `mlr3proba` connects much more learners and, even more importantly, connects and implements much more survival measures. Additionally, `mlr3proba` can be embedded in the infrastructure of the `mlr3pipelines` [65] package, which provides a language to build complex machine learning workflows as directed acyclic graphs. `mlr3pipelines` is also used to convert and unify the many predict types of survival models: while some models return a linear prediction vector, others return a continuous ranking, relative risks, or a complete time-dependent distribution such as individual survival function estimates. `mlr3proba` provides several pipeline operators for converting between predict types or even for composing multiple types.

Thanks to the unified interface of `mlr3` and `mlr3proba`, it is directly possible to use state-of-the-art methods to optimize the hyperparameters of the survival methods via `mlr3tuning`. Especially in the survival context, data preprocessing is often a crucial step. Decisions on how to configure the preprocessing should be included in this optimization process to obtain an unbiased estimate of the predictive performance. Modeling preprocessing through `mlr3pipelines` allows the building of a whole pipeline that can be resampled and optimized. To obtain an unbiased estimate of the performance of a pipeline identified through optimization, the whole optimization can be resampled, resulting in a nested resampling setting. Multi-criteria optimization is also supported, e.g. to tune for predictive performance, sparsity, and feature selection stability simultaneously by connecting the `stabm` [94] package.

2.5.7 Conclusion

The analysis of survival data requires the use of adequate statistical methodology, especially when it comes to accounting for missing information due to censoring. Corresponding methods are available and established. However, for modern high-dimensional data increasingly being generated today, omics data in particular, additional challenges emerge. Estimating prediction models often requires elaborate feature selection and hyperparameter optimization. For this task, Bayesian optimization meth-

ods provide a beneficial solution. They can efficiently identify models with competitive prediction accuracy out of a large set of candidate models. Of great importance is the availability and use of software frameworks for reproducible analysis pipelines. One valuable example is the widely used R package `mlr3`, which provides efficient, object-oriented programming on the building blocks of machine learning, together with its extension package `mlr3proba`, which provides a general framework for probabilistic regression, including many popular survival models and survival measures.