Ludwig-Maximilians-Universität München

Department of Statistics

Master's Thesis

Chiara Wiedemann

On the multiplicity of analysis strategies and the resulting biased interpretations of a large-scale benchmark study

Supervisor: Prof. Dr. Anne-Laure Boulesteix
Institut für Medizinische Informationsverarbeitung, Biometrie und Epidemiologie (IBE)

Munich, July 15, 2020

Abstract

In this study, different degrees of freedom that every researcher has to confront are presented and their influence on the study results is analyzed. The aim is to demonstrate how simple it is to cause biased interpretations and how a procedure can lead to a situation where reproducibility is no longer guaranteed.
This work is based on the benchmark study by Herrmann et al. (2020), which compared 13 different statistical methods on several datasets for the prediction of survival time. A major question was whether the use of multi-omics data outperforms the simple Cox model, which considers only clinical covariates, and which statistical method is most appropriate for the given setting.
In this thesis, various analysis strategies to compare the performance of different methods are presented. The results showed that diverse procedures for imputing missing values, which occurred in the results of the benchmark study, strongly influence the later comparison of the statistical methods. The selection of the datasets used to evaluate the methods also played an important role. If the datasets are filtered according to certain properties, a method that previously showed high performance can perform poorly and vice versa. The probably greatest influence on the study results has the performance measure. It is shown that individual measures evaluate different properties of the methods and thus lead to unequal results. This becomes problematic if a researcher only presents the findings of the performance measure that provides the desired result.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, statisticians have been faced with the so-called "replication crisis". The trigger of this crisis was the numerous attempts to reproduce published statistics. A large number of these attempts failed to reproduce the results or received much weaker results than in the original study, see for example Aarts and Lin (2015). There was broad agreement among many statisticians that part of the problem, apart from poor documentation and possibly fraud, was due to the variety of possible analytical strategies (Goodman et al. (2016), Klau et al. (2020)). In order to make future results more easily reproducible for third parties, several papers have already been published that show the effects of different degrees of freedom and provide a guideline to counteract this problem like Hoffmann et al. (2020) and Simmons et al. (2011).

The aim of this thesis is to discuss possible degrees of freedom and different analysis strategies within a benchmark study and to show how these degrees of freedom and strategies can lead to different study results in order to make further researchers aware of the problem of the replication crisis and to counteract it in the future. We refer to the benchmark study by Herrmann et al. (2020). The goal of his study was to compare different methods from different modeling approaches for the prediction of survival time using multi-omics data. In total, the methods were applied to 18 datasets, i.e. 18 different cancer types. The reference models were Kaplan-Meier estimation and a Cox model, which only considers clinical features. Despite the large benchmark study, it was not possible to define one best method overall or to clearly determine whether the use of multi-omics data in general leads to better results than the simple Cox model. Therefore, the question arises how different degrees of freedom and possible analysis strategies influence the choice of the best method.

The structure of this work is as follows: in the *Background* section, the underlying data structure, the applied methods, and the evaluation measures are described. In the chapter *Failure Imputation* different methods are presented how missing values can be replaced and their effect on the performance of the individual methods is discussed. The chapter *Multiplicity of analysis strategies* covers different analysis strategies and compares the performance of the individual methods. This is followed by the chapter *Sampling*. This section is divided into random sampling, structured sampling, and sampling within certain groups of learners. Here different datasets are drawn, and the performance of the methods is compared. The goal of this procedure is to examine whether there are certain methods that perform best for given samples more often than other methods. In the section *Case Studies*, different approaches are presented to highlight certain methods and present them better than they actually are. The aim of this section is to show how small changes in analysis can lead to great changes in results. Finally, the drawn conclusions are outlined in the chapter *Discussion and Conclusion*.

2 Background

2.1 Datasets

The benchmark study by Herrmann et al. (2020) is based on 18 different datasets and this thesis is based on the results of the benchmark study. So, first the datasets used in Herrmann et al. (2020) are described and then the dataset on which this work is based on.

2.1.1 Cancer Types

The 18 datasets used for the benchmark study by Herrmann et al. (2020) represent 18 different cancer types collected by the TCGA Research Network: http://cancergenome.nih.gov/. Table 24 in the appendix gives an overview of the abbreviations used for the individual datasets. Each dataset was joined from five raw datasets. One of these datasets includes the clinical features, which comprises information such as gender or age. The other four raw datasets are molecular data comprising DNA or protein sequences. The datasets under consideration thus contain multi-omics data since several different omics types are analysed. The use of multi-omics data results in high dimensional datasets, which in this case means that each dataset has approximately 80,000 to 100,000 molecular features. The clinical features are usually low-dimensional and thus the smallest group of variables by far. For the considered cancer types the four molecular data *copy number variation (cnv), gene expression (rna), miRNA expression (mirna)* and *mutation* were analysed.

Table 1 gives an overview of the different datasets with further important key figures such as total number of features (p), number of observations (N), number of effective cases (i.e. events) (n_{eff}) and the ratio n_{eff}/N (r_{eff}).

dataset	cnv	mirna	mutation	rna	*clin.*	p	N	n_{eff}	r_{eff}
BLCA	57964	825	18650	23081	5	100525	382	103	0.27
BRCA	57964	835	18847	22694	8	100348	735	72	0.10
COAD	57964	802	19786	22210	7	100769	191	17	0.09
ESCA	57964	763	15162	25494	6	99389	106	37	0.35
HNSC	57964	793	17840	21520	11	98128	443	152	0.34
KIRC	57964	725	12017	22972	9	93687	249	62	0.25
KIRP	57964	593	11610	32525	6	102698	167	20	0.12
LAML	57964	882	6575	29132	7	94560	35	14	0.40
LGG	57964	645	13389	22297	10	94305	149	77	0.18
LIHC	57964	776	15924	20994	11	95669	159	35	0.22
LUAD	57964	799	18966	23681	9	101419	426	101	0.24
LUSC	57964	895	18832	23524	9	101224	418	132	0.32
OV	57447	975	16837	24508	6	99773	219	109	0.50
PAAD	57964	612	12882	22348	10	93816	124	52	0.42
SARC	57964	778	12478	22842	11	94073	126	38	0.30
SKCM	57964	1002	19488	22248	9	100711	249	87	0.35
STAD	57967	787	19141	26027	7	103929	295	62	0.21
UCEC	57447	866	21226	23978	11	103528	405	38	0.09

Table 1: Overview of datasets used for the benchmark study Herrmann et al. (2020) with corresponding key figures

The columns "cnv", "mirna", "mutation", "rna" and "clin." indicate the number of features in the respective group. As it can be seen the group of cnv is the one that contains the most features while mirna is the smallest group of omics data.

2.1.2 Results of the benchmark study

The analyses of this work are based on the results of the benchmark study by Herrmann et al. (2020). The benchmark experiment was implemented by using the R package *mlr* (Bischl et al. (2016)). To evaluate the performance of the learners, a repeated k-fold cross-validation (CV), see Vanwinckelen and Blockeel (2012) and Bischl et al. (2012) was applied. For this purpose, one dataset is split into k random subsets, where $k-1$ subsets are used for training and one of the k subsets is used for testing. Finally, the performance is averaged over all k testing folds. For the repeated CV, this procedure is repeated multiple times. For smaller datasets, a 10 x 5-fold CV was used and for larger datasets 5 x 5-fold CV to ensure computation time was kept viable. In total there were 7 large and 11 small datasets and 13 applied learners. Thus, the dataset on which this work is based on contains a total of $(7 \cdot 25 + 11 \cdot 50) \cdot 13 = 9425$ rows. Interesting for this work are especially the results of the ibrier and cindex, which were used to evaluate the performance of the learners. For some CV iterations, however, there were missing values for the performance measures, for example due to numerical problems. How these missing values were treated is described in the chapter 3.

2.2 Prediction methods

In this section 13 different statistical methods and their theoretical background are presented. For the analysis of the different cancer types, these methods should fulfil certain properties. For example, it must be possible to deal with the fact that the number of features far exceeds the number of observations. Furthermore, the molecular features are high dimensional while the clinical features have low dimensionality. Since it is known that the informational content of the clinical features is large, a prediction method should not neglect these features. In addition, the sparsity and interpretability of the method is often required. In the following, 13 different methods which belong to the modeling approaches penalised regression, boosting, random forest and reference methods are presented. The description of the methods is related to the description of Herrmann et al. (2020), since in this section not only the theoretical background of the methods but also their application at Herrmann et al. (2020) is to be described.

2.2.1 Penalised regression

The methods of this section include four Lasso methods and one ridge-based method. First, the Lasso methods are presented, which lead to sparse models by subset selection and regularization. Then the ridge-based method is introduced, which uses group specific co-data.

Standard Lasso

The basic idea of standard Lasso is to combine classical regression analysis with variable selection and regularization (Tibshirani (1997)). The estimation of the coefficients $\hat{\beta}$ is obtained by maximizing the penalized log partial likelihood. Since the Cox model assumes $\lambda(t^*|x_i) = \lambda_0(t^*)exp(x_i^T\beta)$ the partial likelihood is obtained by

$$L(\beta) = \prod_{r\in E} \frac{exp(x_{i_r}^T\beta)}{\{\sum_{l\in R_r} exp(x_l^T\beta)\}}, \tag{1}$$

with E as the set of event indices, R_r the set of indices of individuals at risk at time t_r and i_r is the index of the failure at time t_r. As described in Simon et al. (2011) the estimator is obtained by

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}}\{log(L(\beta)) - \lambda \sum_{j=1}^{p} |\beta|\}, \tag{2}$$

where λ controls the size of the penalization. If, for example, λ is set to 0, this leads to a standard regression estimation. Herrmann et al. (2020) chose the parameter λ by 10-fold CV. A disadvantage of this method is that if several features are correlated, only one of them will be selected and it is also not possible to include the group structure information of the multi-omics data. Herrmann et al. (2020) used the package *glmnet* (Friedman et al. (2010)) for implementation. In the further course of this thesis we will refer to this method as *Lasso*.

Integrative Lasso with Penalty Factors (IPF-Lasso)

Boulesteix et al. (2017) introduced an extension of the standard Lasso. The basic idea is to use different penalties for different groups according to their relevance. This method is specifically designed to take into account the group structure of the different omics data and to apply different penalties. So, if M modalities (in this case omics groups) are given with $x_{\cdot 1}^{(m)}, ..., x_{\cdot p_m}^{(m)}$ features of modality m ($m = 1, ..., M$) and let p_m be the number of features within modality m, then the coefficients are estimated by minimizing

$$\sum_{i=1}^{n}(y_i - \sum_{m=1}^{M}\sum_{j=1}^{p_m} x_{ij}^{(m)}\beta_j^{(m)})^2 + \sum_{m=1}^{M}\lambda_m||\beta^{(m)}||_1, \tag{3}$$

with β_j^m the coefficient of the feature $x_{\cdot j}^m$, $||.||_1$ the $L1$ norm, $\lambda_m > 0$ the penalty on modality m and $\beta^{(m)} = (\beta_1^{(m)}, ..., \beta_{p_m}^{(m)})^T$. A penalty vector $(1, \lambda_2/\lambda_1, ..., \lambda_M/\lambda_1)$ results, since the first group is set as reference. The hyperparameter λ must be set in advance. To keep the computational time feasible, a *two-step IPF-Lasso* was introduced by Schulze (2017). The first step is to define a single candidate vector for the penalty factor by applying a regression model to the data and calculating the mean of the coefficients in each group. In the second step these means are inverted and used as penalty factors for the IPF-Lasso method. In the benchmark study of Herrmann et al. (2020), the penalty factors were calculated in the first step by a ridge regression, for each feature group. For the inner resampling, a 5-fold CV was used in the first step and a 10-fold CV in the second step. For implementation, the package *ipflasso* (Boulesteix and Fuchs (2015)) was used. In the further course of this work we will refer to this method as *ipflasso*.

Priority-Lasso

In many cases, clinical researchers already have prior knowledge of the data used. For example, researchers may already know that one particular omics group contains more useful information than another. In order not to lose such knowledge and include it in the estimate, the priority-lasso (Klau et al. (2018)) method was developed. Here the priority order of the different groups is determined by the researcher. Afterwards, regression models are fitted using the features in the order of the group's priority. The linear predictor created in each step is used for the regression model with the next higher priority as offset to explain the variation that could not be explained by the features of the higher priority. Let in the following G be the number of groups, $\pi = (\pi_1, ..., \pi_G)$ the permutation of the groups $(1, ..., G)$, which represents the priority order, $\beta_j^{(\pi_g)}$ the coefficient of the feature j of the group π_g and p_{π_g} the number of features of the group π_g. The coefficients of the first step are obtained by applying standard Lasso to the features of the group with the highest priority in order to explain the greatest possible part of the variability of the outcome. The coefficients are therefore obtained by minimizing

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p_{\pi_1}} x_{ij}^{(\pi_1)}\beta_j^{(\pi_1)})^2 + \lambda^{(\pi_1)}\sum_{j=1}^{p_{\pi_1}}|\beta_j^{(\pi_1)}|. \tag{4}$$

This results in the linear predictor $\hat{\eta}_{1,i}(\pi) = \hat{\beta}_1^{(\pi_1)}x_{i1}^{(\pi_1)} + ... + \hat{\beta}_{p_{\pi_1}}^{(\pi_1)}x_{ip_{\pi_1}}^{(\pi_1)}$ which is used as offset for the Lasso model for the next group of features. This procedure is repeated until all groups are considered and a new offset term is used in each step. Klau et al. (2018) recommends estimating the offset terms by cross-validation, but this was avoided in the benchmark study of Herrmann et al. (2020) to keep the computational time viable. For missing prior knowledge Klau et al. (2018) describes a method to define the priority by cross-validation. Since such a procedure resulted in infeasible computational times for the extensive data of Herrmann et al. (2020), a two-step procedure was applied here. As with the two-step IPF-Lasso, the first step was to define a vector of coefficient means for each group by ridge regression. By inverting and ordering this means by size, the first element of the vector corresponds to the most important group and so on. Thus, the missing prior knowledge can be replaced, and the high computational time can be avoided. To set the parameter λ a 10-fold CV was performed in each step. For the implementation, the package *prioritylasso* (Klau and Hornung (2017)) was used.

Priority-Lasso favoring clinical features

For the priority-lasso method, clinical features can be easily favored by giving them the highest priority and are used as an offset for the next step of the fit. In the benchmark study of Herrmann et al. (2020), the priority is therefore only defined for the molecular groups in the first step. For the determination of the parameter λ a 10-fold CV was again performed in each step and the package *prioritylasso* (Klau and Hornung (2017)) was used. In the following we will refer to this method as *prioritylasso favoring.*

Adaptive group-regularized ridge regression
Van De Wiel et al. (2015), introduced an extension of ridge regression, which allows the use of additional information from features. The features are grouped according to this additional information and a L_2-based penalty term is used. However, this method was not originally intended for the use with multi-omics groups but was provided by a special routine by the package author in personal communication. The large differences in the group-specific penalties of the ridge allow a post-hoc variable selection, which was used in the benchmark study of Herrmann et al. (2020) and the maximum number of selected variables was set to 1000. The penalty parameter λ was determined by a 10-fold CV and the package *GRridge* was used for the implementation. In the further course of this work we will refer to this method as *grridge*.

2.2.2 Statistical boosting

Statistical boosting is an iterative approach in which weak models are fitted in each step and the estimates are updated accordingly. The basic idea is to find a function f which minimizes a certain loss function ρ. The function f can therefore be described as

$$f^*(.) = \underset{f(.)}{\operatorname{argmin}} \mathbf{E}[\rho(Y, f(\mathcal{X}))], \tag{5}$$

with Y as target variable and $\mathcal{X}$ as matrix of independent variables (Bühlmann et al. (2007)). For the application of statistical boosting methods, three important steps are always followed (De Bin (2016)): (1) computation of the negative gradient with respect to f; (2) fitting the base learner and update the estimate; (3) control learning through a learning rate v. For computing a boosting algorithm, the two parameters m_{stop}, i.e. the number of boosting steps and v, i.e. the learning rate are steered by the creator.
The use of boosting methods is suitable in the case of multi-omics data, since it is possible to use data for which the number of features exceeds the number of observations and still an interpretable variable selection is given. In the benchmark study of Herrmann et al. (2020) three different boosting methods were presented and compared with each other. These are *model-based boosting, likelihood-based boosting*, and *likelihood-based boosting with favoring clinical features.*

Model-based boosting
This method uses univariate linear models as base learners and a component-wise boosting algorithm, which considers each feature individually and only updates the one that reduces the loss the most (Hofner et al. (2014)). Therefore, there are p possible updates that must be computed and the feature, which minimizes the loss the most is chosen. For censored survival times, the loss function is set to the negative partial log-likelihood.
The following algorithm is computed (De Bin (2016)):

1. initialize $\hat{\beta} = (0, ..., 0)$
2. compute the gradient as $u_i^{[m]} = \delta_i - \sum\limits_{l \in R_i} \delta_l \frac{exp(x_l^T \hat{\beta}^{[m-1]})}{\sum\limits_{k \in R_l} exp(x_k^T \hat{\beta}^{[m-1]})}$, with R_i as risk set at time t_i and δ_i as indicator whether the i-th observed survival time t_i is censored or not
3. compute the possible updates $\hat{\beta}_j^{[m]} = (x_{.j}^T x_{.j})^{-1} x_{.j}^T \hat{u}^{[m]}$
4. choose the feature which minimizes the squared error loss the most

For implementation, the function *glmboost* of the package *mboost* was used (Hothorn et al. (2018)). Herrmann et al. (2020) determined the number of boosting steps m_{stop} by 10-fold CV and set the maximum m_{stop} to 100. The learning rate v was set to the default value 0.1. In the following we will refer to this method as *glmboost*.

Likelihood-based boosting
This boosting method uses in contrast to model-based boosting a penalised version of the partial likelihood as a loss function (De Bin (2016)), which is calculated by

$$pl_{pen}(\beta) = pl(\beta) - 0.5\lambda\beta^T \mathbf{P}\beta, \tag{6}$$

with $\mathbf{P}$ as $p \times p$ matrix, usually the identity matrix. Furthermore, the learning rate v is not fix, it is applied via a penalty parameter λ, that is directly applied in the coefficient estimation. To save the iterative updates of the parameter estimate, the offset term $\hat{\eta}_i^{[m-1]} = x_i^T \hat{\beta}^{[m-1]}$ is added to the penalised log-likelihood, resulting in the function

$$pl_{pen}(\beta|\hat{\beta}^{[m-1]}) = \sum_{i=1}^{n} \delta_i [\hat{\eta}_i^{[m-1]} + x_i^T \beta - log(\sum_{l \in R_i} exp(\hat{\eta}_l^{[m-1]} + x_l^T \beta))] - 0.5\lambda\beta^T \mathbf{P}\beta, \quad (7)$$

where R_i is again the risk set at time t_i and δ_i indicates whether the i-th observed survival time t_i is censored or not. To obtain the iterative updates, this function is maximized in every boosting iteration. For implementation, the package *CoxBoost* (Binder (2013)) is used. For the benchmark study of Herrmann et al. (2020) the actual m_{stop} was again calculated by 10-fold CV and the maximum number of boosting steps was set to 100. The penalty λ was computed according to the number of events (default setting).

Likelihood-based boosting favoring clinical features

The procedure is the same as for simple likelihood-based boosting, with the difference that clinical features are favored. Since the iterations of the previous steps are included as an offset and a penalty has been applied to the coefficients, one can define features that must be included in the mandatory by setting the corresponding diagonal elements of the matrix $\mathbf{P}$ to zero. It should be noted that only the clinical features are favored, any further group structure information is not used. Again, the package *CoxBoost* and the corresponding function (Binder (2013)) was used for implementation. The settings of boosting steps m_{stop} and penalty λ are the same as for the simple likelihood-based boosting. In the further course of this work we will refer to the simple likelihood-based boosting method as *CoxBoost* and to the likelihood-based boosting method which favors clinical features as *CoxBoost favoring.*

2.2.3 Random forest

Random forests are ensembles of decision trees and represent one of the most successful Machine Learning algorithms for both classification and regression (Breiman (2001)). Decision trees predict based on hierarchical decisions and are structured as a binary tree. Bootstrapped aggregation, where T decision trees are constructed that are trained on T bootstrapped training datasets, can be referred to as Bagging. Random Forests combine many decision trees with bagging to reduce the risk of overfitting and thus the induced high generalization error (Kho (2018)). Additionally, the high variance of the decision trees can be reduced by Bagging. Random Forest is a modified form of Bagging for trees where bootstrapped decorrelated trees are constructed through randomized splits (Hastie et al. (2009)). The following algorithm is defined by Hastie et al. (2009) in order to construct a Random Forest:

1. For $b = 1$ to B:
    (a) Draw a bootstrap sample of size n from training data
    (b) Fit a random forest tree T_b to the bootstrapped data by recursively repeating the following steps for each terminal node of the tree, until the predefined minimum node size
        i. Select *mrty* numbers of features randomly
        ii. Choose the best feature and split-point combination
        iii. Split the node into two daughter nodes
2. Output the ensemble of trees $\{T_b\}_1^B$

The predictions for new data are then calculated using $\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$.

In the benchmark study of Herrmann et al. (2020) three different implementations of random forests were used. These are *ranger*, *RandomForestSRC* and *Block Forest*. The *rfsrc* package (Ishwaran and Kogalur (2018)) stands for fast parallel computing for survival, competing risks, regression, and classification. The number of trees is 1000, the parameter *mrty* is set to $[\sqrt{(p)}]$ and the minimal node size is 3. The package *ranger* is known as well suited for high dimensional data (Wright and Ziegler (2017)), which is very useful in the context of multi-omics data. The number of trees is 500, the parameter *mrty* is again set to $[\sqrt{(p)}]$ with a minimal node size of 3. The package *blockForest* creates different subsets of data containing the individual data types referred to as "blocks" as described in Hornung and Wright (2019). For blockForest the number of trees is set to 2000 and the parameter *mtry* is set to $\sum_{m=1}^{M} \sqrt{(p_m)}$. The minimal node size is again 3. A disadvantage of the random forest methods is that the resulting models are difficult to interpret, although this is often required in the medical context.

2.2.4 Reference methods

In order to assess the performance of the different methods presented above, they are compared with two reference models. These are *Kaplan-Meier estimate* and *Cox proportional hazard model.*

Kaplan-Meier estimate
Kaplan-Meier estimates the survival probability without using any information from features as described in Goel et al. (2010). The estimator of the probability $P(t > t^*)$, that an event has not occurred until time t^*, is given by

$$\hat{S}(t) = \prod_{i:t_i \leq t} (1 - \frac{d_i}{n_i}), \tag{8}$$

with t_i a time when at least one event has occurred, d_i number of events (i.e. deaths) which have occurred at time t_i and n_i number of individuals for which no event has occurred at time t_i. Since the Kaplan-Meier estimate does not use information from the features, any other prediction method should exceed the accuracy of the Kaplan-Meier. The Kaplan-Meier estimate was created using the function *survt* from the package *survival* Therneau (2015).

Cox proportional hazard model

The second reference model is the Cox proportional hazard model, as described in Cox (1972). This model uses clinical variables to predict the survival probability $P(t > t^*)$, that an event has not occurred until time t^*. The model is given by

$$S(t^*) = S_0(t^*)^{exp(x_i^T \beta)}, \tag{9}$$

until time t^*, with $S_0(t^*) = exp(-\Lambda_0(t^*))$ as baseline survival and x_i as clinical covariates for observation i. The baseline survival can be calculated by applying the *Breslow estimate*

$$\hat{\Lambda}_0(t^*) = \sum_{t_i \leq t^*} \frac{1}{\sum\limits_{l \in R_i} exp(x_l^T \beta)}, \tag{10}$$

with R_i as risk at time t_i. Since the Cox model only considers clinical variables, the comparison with this model can be useful to assess whether the use of multi-omics data can lead to better prediction performance for survival time. To apply this method, the function *coxph* from the package *Survival* (Therneau (2015)) is used.

2.2.5 Overview

In the description of the different methods it could be seen that these methods often consider the group structure of clinical and molecular features differently. Table 2 gives an overview how these methods can be classified. In chapter 4.3 the different groupings of the methods are described in more detail and it is analyzed whether certain handlings of the group structure information or whether certain modeling approaches perform better than others.

	Penalised Regression	Boosting	Random Forest
Not using group structure	Lasso	Cox Boost glmboost	rfsrc ranger
Favoring clinical features	prioritylasso fav	Cox Boost fav	
Group structure without favoring clinical features	ipflasso prioritylasso grridge		blockForest

Table 2: Modeling approaches and handling of group structure information

It should also be mentioned that in the benchmark study by Herrmann et al. (2020) another method was applied, which will not be considered further in the following. This is *Sparse Group Lasso* (Simon et al. (2013)), another extension of the usual Lasso estimate. This method led to fatal error messages in R under windows and to extremely long computation times using Ubuntu. Therefore, this method could only be applied to the four smallest datasets and will not be considered in the study at hand.
Moreover, in the further course of the study at hand we will uniformly refer to the statistical method as *learner*.

2.3 Evaluation measures

The properties discrimination and calibration are often used to evaluate the performance of a method. For survival prediction, discrimination, as described in De Bin et al. (2014), indicates how well a method predicts the right order of survival times. The calibration describes how accurately a model's prediction of survival represent the survival in the observed data, see Royston and Altman (2013). The benchmark study of Herrmann et al. (2020) considered the measures c-index and integrated Brier-score for the evaluation of these properties for the learners. The integrated Brier-score measures both, discrimination, and calibration whereas the c-index only concentrates on the validation of the discrimination. These measures are described below, and an additional method is presented. The goal of this additional validation method is not only to determine one best method, but also to determine which methods perform similarly well using a measure such as integrated Brier-score or c-index. The following definitions of integrated Brier-score and cindex refer to Graf et al. (1999), Uno et al. (2011) and Gerds et al. (2013).

2.3.1 Integrated Brier-Score

The aim of the integrated Brier-Score in context of survival time, is to assess whether the probability $P(t_i > t^*|x_i)$, that an individual i survives until a certain time t^*, given the prognostic features x_i, is well estimated by the predictor $\hat{\pi}^{\mathcal{M}}(t^*|x_i)$, where $\mathcal{M}$ is a method obtained by using the training data $D\backslash\mathcal{D}$, where $\mathcal{D}$ is the set of observations used for testing. Let in the following apply $|\mathcal{D}| = n_T$ and $|D\backslash\mathcal{D}| = n$. Then the integrated Brier-score is calculated by

$$I\hat{B}S(t_0) = \int_0^{t_0} [n_T^{-1} \sum_{i=1}^{n_T} (I(t_i > t^*) - \hat{\pi}_i^{\mathcal{M}}(t^*|x_i))^2] dW(t^*). \tag{11}$$

The mean squared error $I(t_i > t^*) - \hat{\pi}_i^{\mathcal{M}}(t^*|x_i)$ indicates whether $\hat{\pi}_i^{\mathcal{M}}(t^*|x_i)$ is a good estimator for the probability $P(t_i > t^*|x_i)$. Here I is the indicator function, which gives 1 if $t_i > t^*$ and 0 otherwise. To consider all times up to t_0 the integrated Brier-score integrates the expected mean squared error up to the time t_0, where W is a weighting function. The smaller the integrated Brier-score, the better the prediction performance. A value of 0.25 means that the prediction has not taken any information into account and is therefore called the data independent value in the following. For the integrated Brier-score we will further use the abbreviation *ibrier*.

2.3.2 C-Index

The c-index evaluates the discrimination of a model $\mathcal{M}$. Again $\mathcal{D}$ is the set of observations used for testing with $|\mathcal{D}| = n_T$ and I is the indicator function. Furthermore $\hat{\pi}_i^{\mathcal{M}}(t^*|x_i)$ is the estimate for the probability $P(t_i > t^*|x_i)$ and $\mathcal{N}_i(t_i^*)$ is defined by $\mathcal{N}_i(t_i^*) = I[t_i \leq t^*, \delta_i = 1]$. Then the c-index can be calculated by

$$\hat{C}(t) = \frac{\frac{1}{n_T^2} \sum_{i=1}^{n_T} \sum_{k=1}^{n_T} I[\hat{\pi}^{\mathcal{M}}(t^*|x_i) > \hat{\pi}^{\mathcal{M}}(t^*|x_k)] I[t_i < t_k] \mathcal{N}_i(t^*)}{\frac{1}{n_T^2} \sum_{i=1}^{n_T} \sum_{k=1}^{n_T} I[t_i < t_k] \mathcal{N}_i(t^*)}. \tag{12}$$

The c-index thus describes the ratio of concordant pairs among all concordant or discordant pairs. To consider not only those pairs of which the status concordant and discordant is known, Herrmann et al. (2020) uses a special case of inverse probability-of-censoring weighting (IPCW) based concordance statistics as described in Uno et al. (2011). A value of 0.5 corresponds to a random prediction without using any information from covariates, which in the survival context corresponds to the Kaplan-Meier estimator. A learner which takes certain features into account should therefore at least perform better than the baseline method which does not use covariates. For the c-index, the higher the value, the better the prediction, so a learner should at least have a value above 0.5. For the c-index we will use the expression *cindex* in the following.

2.3.3 Best learner environment

Inspired by Westphal and Brannath (2020), which compares different selection rules within a benchmark study, an indicator was introduced for this work, which not only considers the best performing learner of a dataset and thus further information might be lost, but also indicates which learner performs similarly well as the learner with the best performance. Therefore, a learner i is defined as *in the 0.05 environment* of the best performing learner if

$$\frac{|\vartheta_i - \vartheta_{best}|}{\vartheta_{best}} < 0.05, \tag{13}$$

with ϑ_i as performance measure (cindex or ibrier) for any learner i and ϑ_{best} as performance measure for the learner with the best average performance. The procedure is exemplified in figure 1.

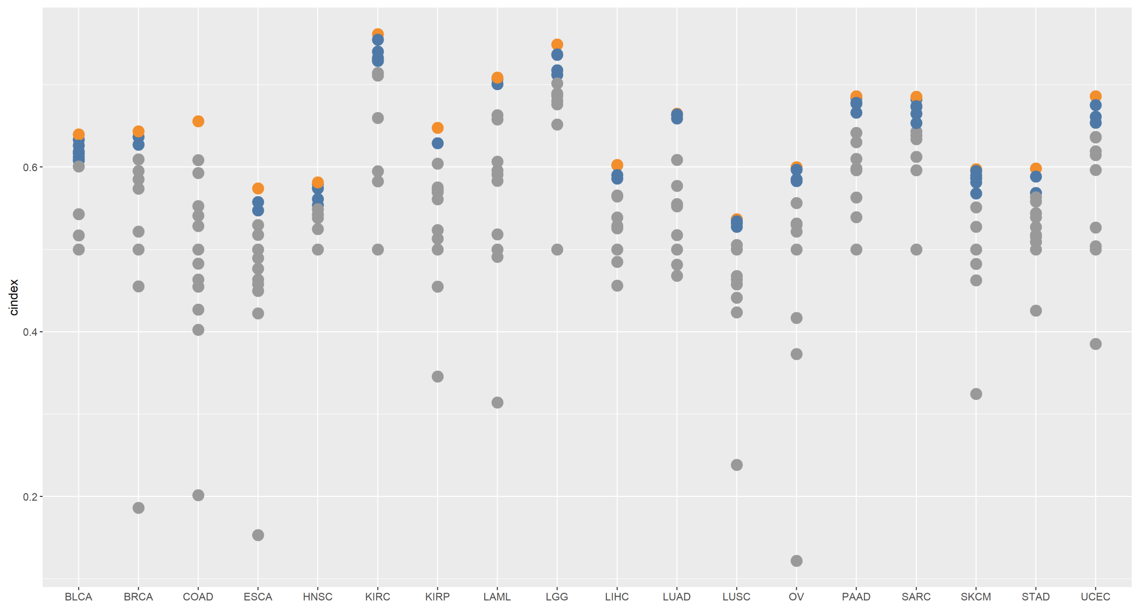

Figure 1: Illustration of the 0.05 environment calculation for cindex. Colors indicate the learner's performance - best performing learner measured on the mean cindex (orange), in the 0.05 environment of the best performing learner (blue), not within the 0.05 environment (gray).

The orange dot represents the best performing learner for each dataset and the blue dots are learners that are in the 0.05 environment of the best performing learner. All other learners are indicated as gray dots. For example, the dataset ESCA has one best performing learner and two learners that perform nearly as well as the best one. For the dataset COAD instead, the best learner has a cindex much higher than the other learners. Therefore, only the best learner itself is taken into account. However, it should be mentioned that the threshold of 0.05 for the best learner environment is a degree of freedom by the researcher. When comparing different learners in the following analysis, not only the mean performance of the learners is often considered but also the number of datasets for which the learners were in the 0.05 environment of the best learner. The goal of this approach is to compare different learners and to identify not only the learner that has the best mean performance across all datasets, but also those which often perform very well. For example, if a learner does not have the best average mean cindex but was in the environment of the best learner for most datasets, then one can conclude that this learner is not the best possible learner, but a very good learner for the given setting. It is possible that such a learner could be preferred if it has desired properties such as interpretability of the resulting model, but the best performing learner has not.

3 Failure imputation

In the following chapter, the problem of algorithm failures is explained in more detail. Different methods to replace algorithm failures are presented and their results are compared. The aim is to analyse the influence of the choice of a specific imputation method on the study result.

3.1 Imputation difficulty

When running an algorithm on large amounts of data, it is possible that it fails and an error message appears, for example due to numerical problems. Since in our case each learner has run on a dataset more often due to cross-validation iterations, there are datasets for which the learner has only failed for a part of the iterations but was successful for the other part. The question now arises how to replace these missing values of the individual CV iterations so that a learner with missing values can be compared fairly with a learner without missing values. Simply ignoring the missing values would unfairly favor the learner with the missing values, since it is likely to exclude the very values for which estimation is difficult and thus the other learners without missing values may not perform as well. Therefore, the results would be biased and a fair comparison between the learners would no longer be possible. Various imputation methods have been developed to counter this problem, but the choice of the imputation method is a degree of freedom by the author.

In the benchmark experiment of Herrmann et al. (2020) 13 different learners are compared with each other. Of these learners, six have at least one failure, while the other seven have not failed once. Furthermore, among the six learners that failed there are a few for which the proportion of failures is very high, and others for which the proportion is very low.
Table 3 shows the applied learners with at least one missing value and the number and percentage of CV iteration failures for each dataset. Lasso and glmboost are the learners with the most missing values. However, if we look at the individual datasets, we can see that there are some datasets for which the learners do not have any missing values at all, but also those for which they have an extremely high

percentage of missing values. In the case of BRCA, Lasso fails in 100% of the CV iterations. Prioritylasso fails in only 1% of total cases, but for UCEC in almost a third.

	Lasso		glmboost		ipflasso		prioritylasso fav		grridge		prioritylasso	
BLCA	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
BRCA	25	(100%)	13	(52%)	2	(8%)	0	(0%)	0	(0%)	0	(0%)
COAD	21	(42%)	17	(34%)	15	(30%)	0	(0%)	0	(0%)	1	(2%)
ESCA	34	(68%)	24	(48%)	5	(10%)	0	(0%)	4	(8%)	0	(0%)
HNSC	2	(8%)	0	(0%)	1	(4%)	0	(0%)	1	(4%)	0	(0%)
KIRC	0	(0%)	0	(0%)	0	(0%)	10	(20%)	0	(0%)	0	(0%)
KIRP	12	(24%)	10	(20%)	3	(6%)	0	(0%)	1	(2%)	0	(0%)
LAML	23	(46%)	12	(24%)	10	(20%)	0	(0%)	0	(0%)	0	(0%)
LGG	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
LIHC	12	(24%)	13	(26%)	28	(56%)	0	(0%)	0	(0%)	0	(0%)
LUAD	3	(12%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
LUSC	17	(68%)	9	(36%)	6	(24%)	0	(0%)	0	(0%)	1	(4%)
OV	31	(62%)	12	(24%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
PAAD	1	(2%)	1	(2%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
SARC	3	(6%)	2	(4%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
SKCM	19	(38%)	16	(32%)	1	(2%)	0	(0%)	0	(0%)	0	(0%)
STAD	12	(24%)	9	(18%)	0	(0%)	0	(0%)	4	(8%)	0	(0%)
UCEC	11	(44%)	5	(20%)	0	(0%)	4	(16%)	0	(0%)	7	(28%)
total	**226**	**(31%)**	**143**	**(20%)**	**71**	**(10%)**	**14**	**(2%)**	**10**	**(1%)**	**9**	**(1%)**

Table 3: Applied learners with the total number and proportion of CV iteration failures per dataset. Only the learners with at least one missing value are shown.

On the one hand, the difficulty of imputation lies in not presenting the learner better or worse than it actually is and on the other hand one would like to take into account the proportion of missing values. For example, if for a specific dataset Learner A has the same mean value (without taking NA values into account) as Learner B, but Learner A fails in only 2% of CV iterations and Learner B fails in 30% of CV iterations, it seems unfair to replace the NAs of both Learners with the same value. In the following chapter the different imputation methods are presented and it is shown how the missing values are replaced and how or if the proportion of missing values is included.

3.2 Imputation methods

Four different imputation methods are compared. Two naive methods which ignore the proportion of failures and two differentiating methods which take the proportion into account. Since the goal of this section is to compare the performance of the learners for the different imputation methods, only the missing values for ibrier and cindex are replaced.

3.2.1 Naive methods

The first method, which we will call the *mean value method* in the following, simply replaces the missing values with the mean value of the CV iterations for which the learner did not fail. Let M be the set of indices of not failed CV iterations, then the missing values for cindex and ibrier are replaced by

$$\mathcal{I}^{\mathcal{M}}_{cindex,ibrier} = \frac{\sum_{m \in M} x_m}{|M|}, \tag{14}$$

with x_m cindex resp. ibrier for the CV iteration m. However, this approach can lead to the learner being presented in a better way than it actually is, since the missing values are simply replaced by the values that are easier to estimate.

Another naive method is the *data independent method*. Here every CV iteration failure is replaced by the data independent value (0.5 for cindex and 0.25 for ibrier). So, the imputation values for cindex and ibrier are determined by

$$\mathcal{I}^{D}_{cindex} = 0.5 \qquad\qquad \mathcal{I}^{D}_{ibrier} = 0.25. \tag{15}$$

However, replacing the missing values with the data independent values may result in the learner being presented worse than it is. If, for example, a learner performs well but fails for some CV iterations, this imputation method can cause its performance to decrease considerably.

3.2.2 Differentiating methods

The first differentiating method considers the proportion of missing values by setting a threshold. This means that if the proportion of missing values is below the set threshold, the missing values are replaced by the mean value of the CV iterations that did not fail. If the proportion of missing values is higher than the defined threshold, the missing values are replaced by the data-independent values (0.5 for cindex and 0.25 for ibrier). This method is therefore a combination of the mean value and data independent method. In the following r is the proportion of the missing values, T the set threshold, M is the set of indices of the not failed CV iterations and x_m the cindex or ibrier value of the CV iteration m. Then the imputation values are calculated by:

$$\mathcal{I}^T_{cindex} = \begin{cases} \frac{\sum_{m \in M} x_m}{|M|}, & r < T \\ 0.5, & r > T \end{cases} \qquad \mathcal{I}^T_{ibrier} = \begin{cases} \frac{\sum_{m \in M} x_m}{|M|}, & r < T \\ 0.25, & r > T. \end{cases} \tag{16}$$

A common value for the threshold is 20% as defined in Probst et al. (2019) and Bischl et al. (2013). This threshold was also applied in the paper by Herrmann et al. (2020) on which this work is based on. However, it should be mentioned that the value of the threshold is an additional degree of freedom of the author. The influence of the choice of the threshold is discussed in more detail below.

An alternative differentiating imputation method was developed in this thesis and is called the *weighted method* in the following. The goal of this method is to avoid the degree of freedom of the threshold. This means that if a learner has many missing values, this learner is automatically weighted worse than a learner with a few number of missing values. If in the following r is again the proportion of missing values, M the set of indices of the not failed CV Iterations and x_m is the cindex or ibrier value of the CV iteration m, then the imputation value is calculated by

$$\mathcal{I}^W_{cindex} = 0.5 + (\frac{\sum_{m \in M} x_m}{|M|} - 0.5)_+ \cdot (1 - r) \tag{17}$$

$$\mathcal{I}^W_{ibrier} = 0.25 - (0.25 - \frac{\sum_{m \in M} x_m}{|M|})_+ \cdot (1 - r). \tag{18}$$

Three important features are included in this calculation. The first two features are included in the term $(\frac{\sum_{i=1}^{n-m} x_i}{n-m} - 0.5)_+$. This term causes that if the mean cindex for all CV iterations which did not fail is 0.5, then the imputation value automatically remains at 0.5. In addition, this term refers to the case that if a learner has a mean value less than 0.5 for the not failed CV Iterations, these values are uniformly set to 0.5, since all values below 0.5 are not relevant in this context. The same holds for ibrier with 0.25. The third feature $(1 - r)$ causes that if a learner has 100% failures for a dataset, then the missing values are all replaced by 0.5 for cindex and 0.25 for ibrier.

3.3 Results

In the following section the performance of the learners is presented using the ibrier and cindex. First, the performance of the learners is compared between the different imputation methods and then the imputation method by setting a threshold is discussed in more detail. Finally, it is analysed how the performance of certain learners changes if zero values, obtained for the cindex, are treated as missing values. For the comparison between the different methods, a default threshold of 20% is chosen.

3.3.1 Comparison of different methods

Figure 2 shows the performance based on the ibrier of the individual learners per imputation method aggregated over all datasets. The color of the lines indicates the proportion of failures per learner over all datasets. The largest differences between the imputation methods can be seen for the learners ipflasso, glmboost and Lasso. While the data Independent method presents these learners worse than the other methods, the mean value method presents them better. The method by setting the threshold at 20% for Lasso and glmboost is very similar to the data independent method, because these learners for the individual datasets predominantly show failures in more than 20% of the CV iterations (see table 3). Interesting are also the different ranks of the ipflasso. For the data independent method ipflasso would be ranked sixth, while for the mean value method it would be the best performing learner. The method with the threshold at 20% places ipflasso on rank five and the weighted method on rank two. Although prioritylasso fails in only 1% of total iterations, some differences can be observed. In this case the method with the threshold at 20% is almost equal to the data independent method, because for the dataset where most failures occurred prioritylasso failed in almost 30%. So, if the threshold would be set at 30% the method after threshold would be the same as the mean value method for ipflasso. The weighted method seems to be a compromise between the mean value method and the data independent method for all learners.

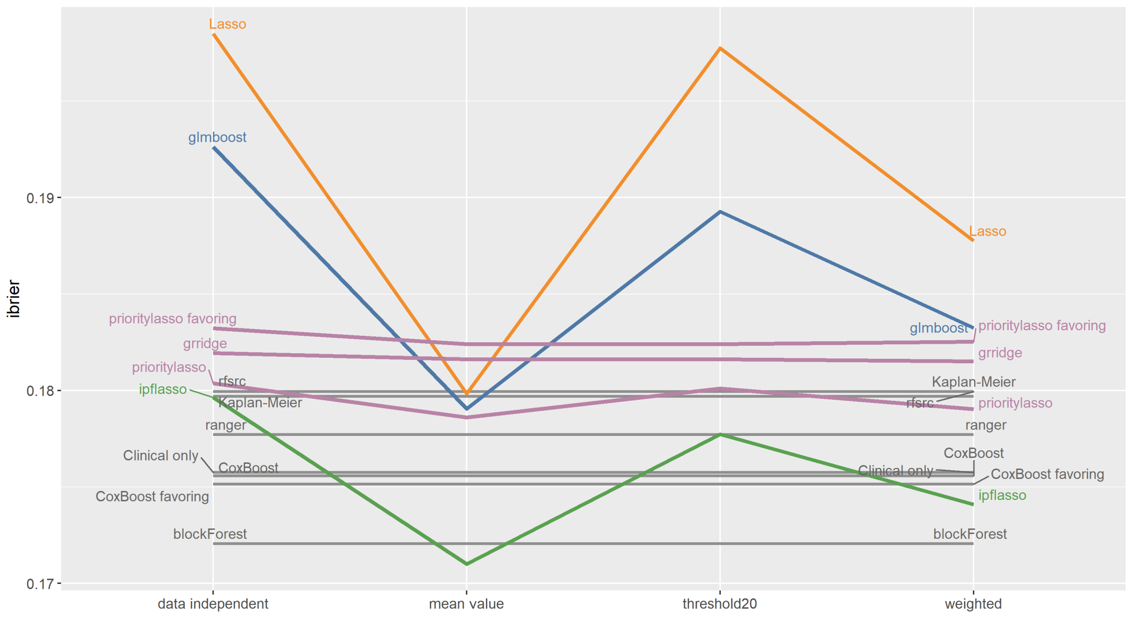

Figure 2: Performance comparison for different imputation methods - ibrier. Colors indicate the total percentage of failures: 0% (gray), below 5% (purple), below 10% (green), below 20% (blue), above 20% (orange).

Figure 3 shows the same setting for the cindex. The differences between the imputation methods are much smaller than for ibrier. The performance of the learners is estimated to be approximately the same by all imputation methods. The ranks also remain the same. However, Lasso and glmboost are rated worse for the mean value method than for all other methods. This is due to the fact that Lasso and glmboost usually have an average cindex below 0.5 for datasets with many missing values. For these cases all other methods assign the value 0.5 to Lasso and glmboost and only the mean value method assigns values below 0.5.

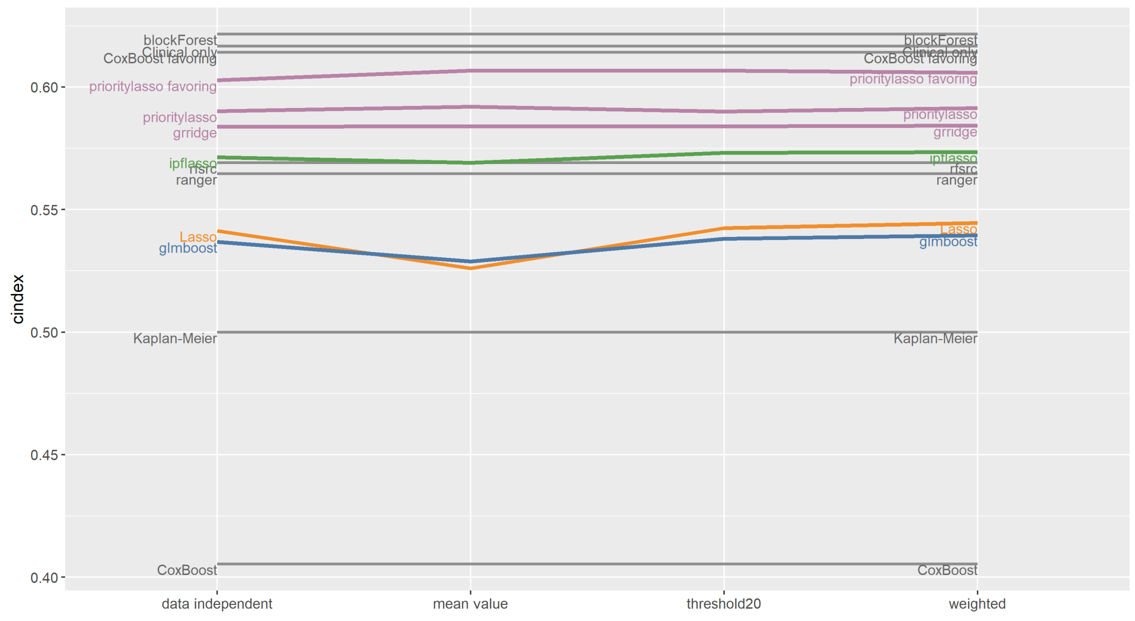

Figure 3: Performance comparison for different imputation methods - cindex. Colors indicate the total percentage of failures: 0% (gray), below 5% (purple), below 10% (green), below 20% (blue), above 20% (orange).

It should also be mentioned that in figure 2 and 3 the performance of the learners was shown aggregated across all datasets. However, an interesting example results if we look at the dataset UCEC separately. Here Lasso has 44%, prioritylasso 28%, glmboost just under 20%, prioritylasso favoring 16% and ipflasso and grridge 0% missing values. Figure 4 shows the performance of the learners for the different imputation methods for ibrier (A) and cindex (B). The differences for ibrier are substantial. For Lasso and prioritylasso, the 20% threshold method is the same as the data independent method, while for all other learners it equals the mean value method. If for example the threshold was set to 15%, then prioritylasso favoring and glmboost would also match the data independent method and would have a worse performance. The weighted method is again a compromise between the data independent and mean value method. Cindex seems to be more robust between the imputation methods. However, it is interesting to note that Lasso has 16% more failures than prioritylasso, but the values between the imputation methods differ less than the values for prioritylasso. The reason for this is that the mean value (without consideration of the failures) for Lasso is 0.55 and for prioritylasso 0.69. This means that the better the learner, i.e. the further the mean value of the not failed iterations deviates from 0.5, the more the learner is affected by the different imputation methods.

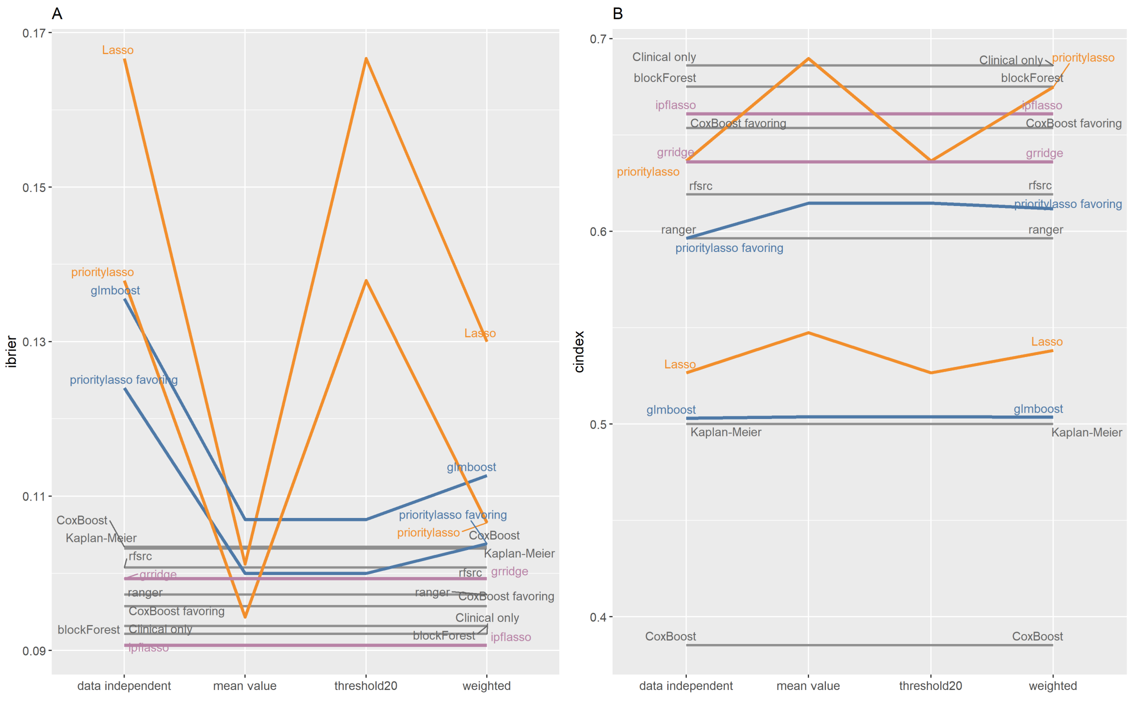

Figure 4: Performance comparison for different imputation methods for dataset UCEC - A: ibrier, B: cindex. Colors indicate the total percentage of failures: 0% (gray), below 5% (purple), below 10% (green), below 20% (blue), above 20% (orange).

If the focus is only on the best performing learner per dataset (measured on the mean ibrier or cindex), the differences are hardly noticeable. Table 4 shows the best learners per dataset for ibrier.

	data independent		mean value		threshold 20%		weighted	
BLCA	CoxBoost fav	0.190	CoxBoost fav	0.190	CoxBoost fav	0.190	CoxBoost fav	0.190
BRCA	blockForest	0.141	blockForest	0.141	blockForest	0.141	blockForest	0.141
COAD	blockForest	0.087	**Lasso**	**0.086**	blockForest	0.087	blockForest	0.087
ESCA	ipflasso	0.214	ipflasso	0.209	ipflasso	0.209	ipflasso	0.210
HNSC	glmboost	0.202	glmboost	0.202	glmboost	0.202	glmboost	0.202
KIRC	ipflasso	0.144	ipflasso	0.144	ipflasso	0.144	ipflasso	0.144
KIRP	ranger	0.118	ranger	0.118	ranger	0.118	ranger	0.118
LAML	ranger	0.182	ranger	0.182	ranger	0.182	ranger	0.182
LGG	Lasso	0.145	Lasso	0.145	Lasso	0.145	Lasso	0.145
LIHC	ranger	0.146	ranger	0.146	ranger	0.146	ranger	0.146
LUAD	CoxBoost fav	0.172	CoxBoost fav	0.172	CoxBoost fav	0.172	CoxBoost fav	0.172
LUSC	grridge	0.210	grridge	0.210	grridge	0.210	grridge	0.210
OV	ipflasso	0.169	ipflasso	0.169	ipflasso	0.169	ipflasso	0.169
PAAD	Clinical only	0.190	Clinical only	0.190	Clinical only	0.190	Clinical only	0.190
SARC	**rfsrc**	**0.179**	glmboost	0.179	glmboost	0.179	glmboost	0.179
SKCM	Clinical only	0.191	Clinical only	0.191	Clinical only	0.191	Clinical only	0.191
STAD	Clinical only	0.192	Clinical only	0.192	Clinical only	0.192	Clinical only	0.192
UCEC	ipflasso	0.091	ipflasso	0.091	ipflasso	0.091	ipflasso	0.091

Table 4: Best performing learners and corresponding ibrier per dataset and imputation method. The mean ibrier is obtained by averaging over all CV-Iterations.

There are only two datasets for which the best learner for one imputation method differs from the others (written in bold). For COAD Lasso has missing values in 42% of CV iterations. For the remaining iterations, however, there is a good mean ibrier value of 0.086, which is used for the mean value method to replace the missing values. The data independent and threshold methods replace the missing values with 0.25 and the weighted method with $0.25 - (0.25 - 0.086) \cdot (1 - 0.42) \cdot I_{0.086<0.25} = 0.155$. Therefore, for all methods except the mean value method blockForest is the better learner with a mean ibrier of 0.087. For SARC the situation is the other way around for the data independent method. Since glmboost fails for only 2 iterations (4%) and otherwise has the best ibrier score (0.179), the missing values of glmboost are replaced with good ibrier values for all methods except the data independent method. The data independent method replaces all missing values with 0.25, which increases the mean ibrier for glmboost and therefore rfsrc is considered as the best learner. It is also interesting that ipflasso is the best learner for all imputation methods for the dataset ESCA, although ipflasso fails in 10% of CV iterations. The reason for this is that the mean value of the not failed CV iterations is much better than the mean ibrier values of all other learners, so ipflasso is still better than the other

learners, even if the missing values are replaced with 0.25. For all other datasets, the respective best learner has no missing value for this particular dataset. For LGG, for example, Lasso is the best performing learner. Although Lasso has the most failures of all learners, it has not a single failure for this dataset and performs very well and is therefore defined as the best learner by all imputation methods. For the 20% threshold and the weighted method there is no difference if only the best learner is considered.
Similar results are obtained for cindex. Here the best learners differ only for the dataset UCEC, because the mean value method determines the learner prioritylasso with 28% failures as the best learner, whereas all other methods determine the Cox model as the best model. See Figure 4, where the UCEC dataset is visualized in detail. For all other datasets, the same best learner is obtained across all applied imputation methods.

3.3.2 Comparison of different thresholds

In this section the influence of the choice of the threshold is examined. The thresholds 5%, 10%, 20%, 30% and 50% are considered. This means, for example, for a threshold of 10%, if a learner fails for a specific dataset in less than 10% of the CV iterations, the missing values are replaced by the mean value of the not failed CV iterations. However, if the learner fails in more than 10% of the CV iterations, the missing values are replaced by the data independent values (0.5 for cindex and 0.25 for ibrier).

For the benchmark experiment six learners have missing values. Table 5 shows the learners with the percentage of failed CV iterations across all datasets. The third column shows the mean cindex (A) and ibrier (B) for a threshold of 20% and columns four to seven show the differences of the mean cindex resp. ibrier for the different thresholds. As expected, for cindex, the mean values of the threshold at 5% and 10% have a negative difference to the threshold at 20% for the Learner Lasso, glmboost, ipflasso and prioritylasso favoring. This means that these methods would classify the learner worse, because the threshold at which the data independent value is assigned is lower. Nevertheless, the maximum difference of -0.004 for prioritylasso favoring is quite small. For grridge and prioritylasso there is no difference to threshold 20%.

One would intuitively expect that the imputations after threshold 30% and 50% would only show positive differences, i.e. that the learners would have a better overall assessment. However, since for most datasets for which the learners have a large proportion of missing values, they have a mean value below 0.5 for the CV iterations that did not fail. Therefore, missing values are replaced with a value below 0.5 and are thus rated even worse than with a threshold of 20%. Only prioritylasso is rated better by both the 30% and 50% thresholds, since prioritylasso has 28% failures for the UCEC dataset, but apart from that has a good mean cindex. If only these six learners were compared, despite the differences, all five thresholds would classify the learner prioritylasso favoring as the best learner for cindex.

A

			differences to threshold 20%			
	failures (%)	threshold 20%	5%	10%	30%	50%
Lasso	31.2%	0.542	-0.001	0	+0.002	-0.001
glmboost	19.7%	0.538	-0.001	-0.001	-0.001	-0.008
ipflasso	9.8%	0.573	-0.002	-0.002	-0.003	-0.003
prioritylasso fav	1.9%	0.607	-0.004	-0.004	0	0
grridge	1.4%	0.584	0	0	0	0
prioritylasso	1.2%	0.590	0	0	+0.002	+0.002

B

			differences to threshold 20%			
	failures (%)	threshold 20%	5%	10%	30%	50%
Lasso	31.2%	0.198	+0.001	0	-0.004	-0.013
glmboost	19.7%	0.189	+0.003	+0.003	-0.003	-0.009
ipflasso	9.8%	0.178	+0.002	+0.001	-0.003	-0.003
prioritylasso fav	1.9%	0.182	+0.001	+0.001	0	0
grridge	1.4%	0.182	0	0	0	0
prioritylasso	1.2%	0.180	0	0	-0.002	-0.002

Table 5: Differences for various thresholds - A: cindex, B: ibrier. The second column shows the percentage of missing values across all datasets for the corresponding learners. The third column shows the mean cindex resp. ibrier aggregated over all CV iterations and datasets for the threshold at 20%. Columns three to seven show the difference of the cindex reps. ibrier between the respective thresholds and the threshold at 20%.

Since hardly any learner has a mean ibrier value greater than 0.25 for the CV iterations that did not fail, all thresholds greater than 20% rate the learners better and all thresholds below that rate the learners worse. The largest difference of -0.013 is for Lasso at a threshold of 50%, which is a change of about 7%. Nevertheless, ipflasso would always be rated as best learner despite the different thresholds.

If only the learners with at least one missing value are compared with each other per dataset and the focus is on the best performing learner, then there are only slight differences. For the dataset COAD, the threshold at 50% chooses Lasso as best learner, since it has 42% failures and the mean value for the CV failures that did not fail is very good. All other thresholds rate grridge as best learner. This difference, however, only occurs for ibrier. For cindex all 5 thresholds choose grridge as best learner. For the dataset LAML there are differences for ibrier and cindex. Here, all thresholds from including 20% for ibrier and cindex rate ipflasso as the best learner. The thresholds below 20% choose grridge for ibrier and prioritylasso for cindex. The last difference occurs for UCEC, but only for cindex. Here the thresholds at 30% and 50% choose prioritylasso as best learner and all thresholds below choose ipflasso. For the ibrier ipflasso is the best learner for all thresholds.

3.3.3 Zero values as missing values

For a few learners, especially CoxBoost, it has occurred that cindex has a value of zero. For the boosting learners, this is the case if *mstop* is set to 0 for the affected iterations. This means that the resulting model contains no variables, therefore no cindex can be calculated, ibrier, however does. The question now is how such zero values should be handled. On the one hand, there is the possibility to leave the zero values just like that and consider them as zeros, and on the other hand, these values could be considered as missing values and thus have to be replaced. Table 6 gives an overview of the learners with the corresponding total number and proportion of zero values per datasets. Especially learners, which also have missing values, are affected. CoxBoost however does not have a single missing value, but a total of 207 (29%) CV iterations for which the cindex is zero. As shown in table 6, there are datasets for which CoxBoost does not have a single zero value (BLCA, KIRC and LGG) and then again there are datasets for which CoxBoost has zero values for more than half of the CV iterations (BRCA, COAD, ESCA and OV). All other learners always have a proportion of zero values below the 20% threshold for each dataset.

	CoxBoost		glmboost		Lasso		ipflasso		prioritylasso		CoxBoost fav	
BLCA	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
BRCA	16	(64%)	2	(8%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
COAD	27	(54%)	6	(12%)	3	(6%)	0	(0%)	0	(0%)	0	(0%)
ESCA	33	(66%)	1	(2%)	1	(2%)	0	(0%)	0	(0%)	0	(0%)
HNSC	1	(4%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
KIRC	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	1	(2%)
KIRP	14	(28%)	0	(0%)	1	(2%)	2	(4%)	0	(0%)	0	(0%)
LAML	20	(40%)	0	(0%)	2	(4%)	1	(2%)	1	(2%)	0	(0%)
LGG	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
LIHC	11	(22%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
LUAD	2	(8%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
LUSC	12	(48%)	2	(8%)	1	(4%)	1	(4%)	0	(0%)	0	(0%)
OV	35	(70%)	7	(14%)	3	(6%)	0	(0%)	0	(0%)	0	(0%)
PAAD	1	(2%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
SARC	1	(2%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
SKCM	17	(34%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
STAD	10	(20%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
UCEC	7	(28%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)	0	(0%)
total	**207**	**(29%)**	**18**	**(2%)**	**11**	**(2%)**	**4**	**(1%)**	**1**	**(0%)**	**1**	**(0%)**

Table 6: Applied learners with the total number and proportion of zero values per dataset. Only the learners with at least one zero value are shown.

If the zero values are considered as missing values, then the number and proportion of missing values per dataset and learner increases. This means for the differentiating imputation methods, that there may now be learners which exceed the threshold more frequently or are now assessed worse by the weighted method. For this specific benchmark experiment, however, this is hardly noticeable, since CoxBoost previously had no missing values and the other learners have only a small number of zero values.

Since mainly CoxBoost is affected by the difficulty of zero values, the following analysis will focus on CoxBoost and its performance changes due to different handlings of zero values. For the performance change for glmboost, Lasso and ipflasso see figure 21 in the appendix. Figure 5 shows the performance of CoxBoost, for the case when zero values are kept as zero values (gray) and for the different imputation methods when zero values are treated as missing values. Since CoxBoost has no further missing values, only the imputation method by setting a threshold is shown for the case that the zero values remain zero.

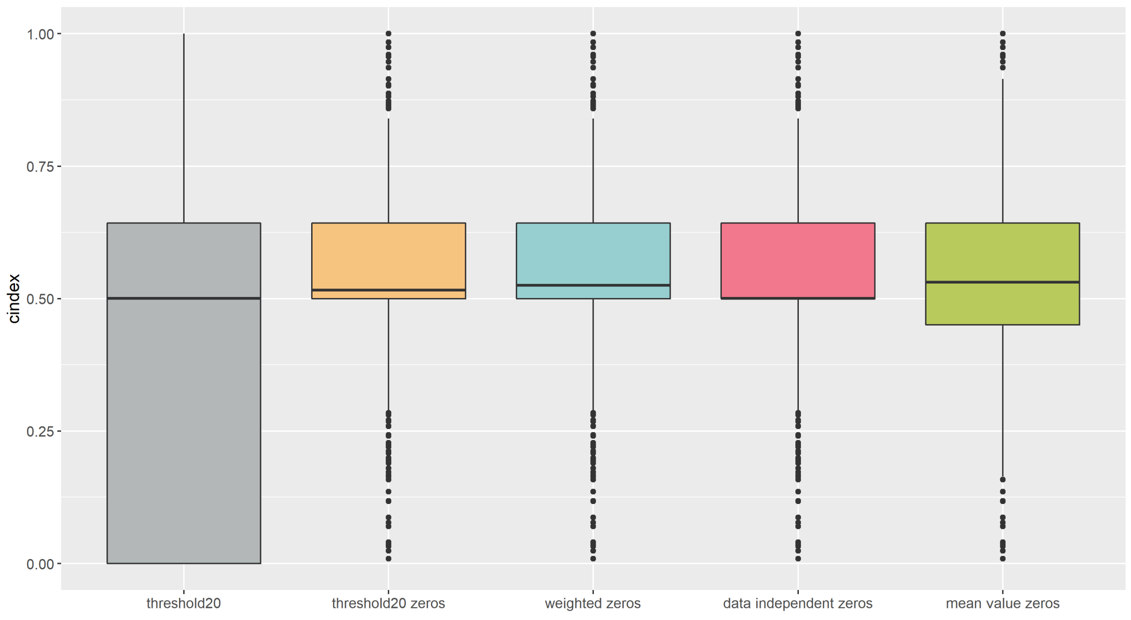

Figure 5: Zero values as missing values - Performance changes for CoxBoost. Each boxplot includes the cindex of all CV iterations over all 18 datasets for CoxBoost.

As expected, CoxBoost is rated much worse if the zero values are kept as zeros. For the threshold and weighted methods CoxBoost performs quite similar and has a median slightly higher than 0.5. For the data independent method, the median remains at 0.5 while it is again slightly higher for the mean value method, but the 0.25 quantile is lower than for the other methods. This is because CoxBoost has a mean cindex below 0.5 for certain datasets, so the mean value method replaces the missing values with a value below 0.5. Considering the mean cindex across all datasets, CoxBoost has a mean value of 0.405 if the zero values are maintained as zeros and is therefore the worst performing learner. If the zeros are treated as missing values, the mean value is 0.548 for the data independent method, 0.541 for the mean value method, 0.549 for the method at a threshold of 20% and 0.551 for the

weighted method. The weighted method therefore represents CoxBoost best, but the mean value method is the only method that gives CoxBoost a better rank. Here CoxBoost not only performs better than Kaplan-Meier, but also as Lasso. This means that if cindex is aggregated across all datasets, CoxBoost performs worse than Kaplan-Meier if the zero values are maintained as zeros. If the zero values are treated as missing values, then CoxBoost performs better than Kaplan-Meier but still worse than all other learners, except for the mean value method where CoxBoost also performs better than Lasso.
However, if the individual datasets are considered separately, large changes can occur. For example, for the dataset LIHC, CoxBoost is the worst performing learner when zero values are maintained. If the zero values are considered as missing values and are replaced by the imputation method with a threshold at 20%, CoxBoost performs better than Lasso, glmboost, prioritylasso, rfsrc, ranger and ipflasso. So, if the goal is to compare the performance of different learners for certain datasets, it is very important how zero values are treated.
Overall, CoxBoost is much better represented if the zero values are treated as missing values. However, it is not possible to determine which method is the correct one. Therefore, it is very important not only to give the mean cindex, but also the number of zero values and missing values and to document exactly how zero values were handled and which imputation method was used.

In this section it was shown that the method of replacing the missing values has an influence on the performance of the learner. The mean value method can lead a learner with a lot of CV iteration failures to becoming one of the best learners, if the mean value of the not failed iterations is very good. However, this method can also lead to a learner being rated worse than all other imputation methods. This is the case if a learner has a mean value below 0.5 for cindex or above 0.25 for ibrier for the iterations that did not fail. Replacing the failures with the data independent method can cause, that a learner that has just a few number of failures and apart from that performs well, to be rated quite bad. An example for this is glmboost for the dataset SARC. When using the imputation method by setting a threshold, attention should be paid to what exactly the threshold is set. It was shown that a

high threshold does not automatically mean that a learner is rated better than if the threshold is set low. Nevertheless, the choice of the threshold remains an additional degree of freedom. This degree of freedom is not present in the weighted method. This method often turned out to be a compromise between the mean value and the data independent method taking into account the proportion of failed CV iterations. Furthermore, the influence of the imputation method also depends on the robustness of the performance measure. Ibrier is obviously more sensitive than the cindex. In addition, the size of the difference between the methods also depends on the performance of the learner. The further the performance of the learner deviates from the data independent values, the greater the influence of the imputation method.
If the individual datasets are considered separately and the performance of individual learners is of interest, it is important which imputation method is used. However, if the focus is only on the best performing learner, only slight differences arise between the different imputation methods and thresholds. If one compares different learners for certain datasets with each other and zero values for the cindex occur, one should carefully consider how these zero values are treated.
To achieve a transparent comparison of different learners, a detailed documentation is always necessary for the handling of zero and missing values.

4 Multiplicity of analysis strategies

The aim of this chapter is to draw attention to how different analysis strategies may lead to different study results. For this purpose, the performance of the learners is analysed on different levels. First, the performance is evaluated across all datasets, then the learners are evaluated according to certain characteristics of the dataset (e.g. for large resp. small datasets) and after that the performance of the learners is evaluated according to certain characteristics of the learner (e.g. membership to the different modeling approaches). A detailed analysis for each dataset and learner follows and finally a test statistic was applied which compares the mean performance of the individual learners and tests for statistical significance. For the following analysis, the imputation method by setting the threshold at 20% is used as default. In addition, the zero values for CoxBoost are maintained as zeros and are therefore not replaced.

4.1 Analysis across all datasets

Within each evaluation measure there are different possibilities to identify the best performing learner. In the following, in addition to the mean cindex resp. ibrier, the median, the number of datasets for which the learner was best performing learner (measured by the mean value), the number of datasets for which the learner was in the 0.05 environment of the best performing learner and the average rank are considered. Table 7 shows the results for the cindex. Here blockForest has the highest cindex mean and the best average rank. In total, blockForest was the best performing learner for 4 datasets just like the simple Cox model. However, if the median of the cindex is considered, the simple Cox model comes out better and is followed by CoxBoost favoring. If one is interested in the number of datasets for which the learner is the best learner or in the 0.05 environment of the best learner, CoxBoost favoring is in first place with a number of 15 datasets, followed by the Cox model and then blockForest.

	mean	median	best/ 0.05 env	rank
blockForest	0.6215	0.6256	4/ 12	3.6
Clinical only	0.6167	0.6309	4/ 14	3.8
CoxBoost fav	0.6142	0.6259	2/ 15	3.8
prioritylasso fav	0.6066	0.6210	2/ 11	4.9
prioritylasso	0.5901	0.5990	2/ 4	6.3
grridge	0.5840	0.5860	1/ 8	6.2
ipflasso	0.5732	0.5859	0/ 6	7.0
rfsrc	0.5690	0.5615	1/ 2	8.3
ranger	0.5646	0.5598	1/ 2	8.4
Lasso	0.5424	0.5000	0/ 4	8.5
glmboost	0.5381	0.5379	1/ 4	8.3
Kaplan-Meier	0.5000	0.5000	0/ 0	10.9
CoxBoost	0.4054	0.5003	0/ 2	11.0

Table 7: Learner performances across all datasets - cindex. Column two shows the mean cindex which is obtained by averaging over all CV iterations and datasets. The "median" column shows the median of the cindex. The fourth column shows the number of datasets for which the learner is the best performing learner according to the mean cindex and the number of datasets for which the learner is in the 0.05 environment of the best performing learner. The last column shows the average rank.

Table 8 shows the results for ibrier. Although blockForest has the best mean value, for almost each additional measure there is at least one other learner that scores better. For the median CoxBoost favoring and Cox model have a lower and therefore better value than blockForest. Looking at the number of datasets for which the learner was the best performing learner, ipflasso is in first place with four datasets, followed by Cox model and ranger with three datasets. Considering the number of datasets for which a learner performs almost as well as the best learner, blockForest and the simple Cox model come first. Overall ipflasso has the lowest average rank. So, for the ibrier almost every possible rating category result in a different learner as best performing learner.

	mean	median	best/ 0.05 env	rank
blockForest	0.1721	0.1779	2/ 12	5.3
CoxBoost fav	0.1752	0.1725	2/ 9	5.2
CoxBoost	0.1756	0.1799	0/ 6	6.2
Clinical only	0.1757	0.1736	3/ 12	5.2
ranger	0.1777	0.1824	3/ 6	7.3
ipflasso	0.1777	0.1806	4/ 10	4.8
Kaplan-Meier	0.1797	0.1884	0/ 4	7.6
rfsrc	0.1799	0.1835	0/ 5	8.6
prioritylasso	0.1801	0.1786	0/ 5	7.5
grridge	0.1816	0.1848	1/ 5	7.1
prioritylasso fav	0.1824	0.1774	0/ 6	8.2
glmboost	0.1893	0.1925	2/ 3	8.6
Lasso	0.1977	0.2024	1/ 3	9.6

Table 8: Learner performances across all datasets - ibrier. Column two shows the mean ibrier which is obtained by averaging over all CV-iterations and datasets. The "median" column shows the median of the ibrier. The fourth column shows the number of datasets for which the learner is the best performing learner according to the mean ibrier and the number of datasets for which the learner is in the 0.05 environment of the best performing learner. The last column shows the average rank.

The differences between ibrier and cindex are also interesting. For example, prioritylasso and prioritylasso favoring perform quite good for cindex in terms of mean and median and become best learner each for two datasets. For the ibrier, however, they do not become the best learner for any dataset. The average rank also differs greatly. Another extreme example is CoxBoost. The performance of CoxBoost is very low for cindex, regardless of how zero values are treated. For ibrier however it scores quite good and is even on place 3 if only the mean ibrier is considered.

When comparing different learners, the central question is usually which learner performs better than all the others. If we look only at the mean value, then for ibrier and cindex blockForest is the best performing learner. If considering the median, then for cindex Cox model and for ibrier CoxBoost favoring is the best learner. If we look at the number of datasets for which the learner has the highest mean cindex, then blockForest and Cox model are the best, for ibrier ipflasso. If the number of datasets for which the learner is not only the best learner but also in the 0.05 environment of the best learner is of interest, then for cindex CoxBoost favoring is first and for ibrier blockForest and the simple Cox model. For the average rank, blockForest is at the top for cindex and ipflasso for ibrier. After such an analysis it can be said that blockForest, the simple Cox model, ipflasso and CoxBoost favoring are good learners for the underlying data, but it does not seem to be possible to determine one learner that generally performs better than the others.

4.2 Analysis based on dataset characteristics

For the following analysis, the datasets were divided into different groups. The characteristics *number of observations* (N), *number of clinical features* ($clin$), *total number of features* (p) and *number of effective cases* i.e. events (n_{eff}) are considered. A dataset is assigned to group 1 if the value of the characteristic is less than the mean value of all datasets and to group 2 if the value is greater than the mean value. For example, the mean of the number of observations across all datasets is 271, so all datasets with fewer observations are assigned to group 1 and all datasets with more observations are assigned to group 2. Figure 6 shows four histograms illustrating the distribution of the cindex across all learners for the different dataset groups and each characteristic. The largest accumulation of values is at 0.5, since many missing values have been replaced by 0.5 and Kaplan-Meier only has values of 0.5. For the number of observations N (figure A) and the number of effective cases n_{eff} (figure D), only slight differences between the groups can be observed. For the number of clinical features (figure B) and the number of features in total (figure C), the differences in the mean values (dashed lines) are quite large. Considering the number of clinical features, the learners have better values for the cindex when the number of features is larger. For the number of features in total the opposite is true. Here the cindex values are better for a smaller number of features. Furthermore, it is visible that most of the zero values of the cindex are in the group of the smaller number of clinical features while for the total number of features most of the zero values are in the larger group. Now the question arises whether there are learners which perform particularly well for certain characteristics of the dataset.

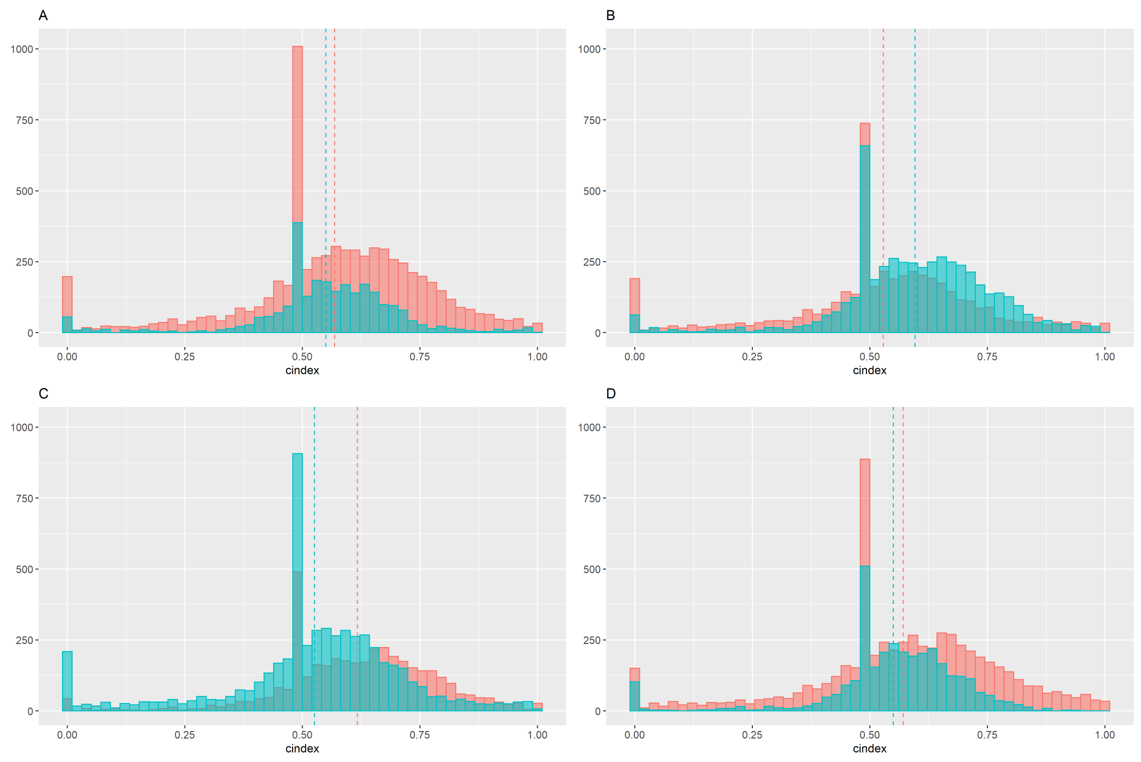

Figure 6: Distribution for different dataset groups - cindex. A: number of observations (N), B: number of clinical features ($clin$), C: total number of features (p), D: number of effective cases i.e. events (n_{eff}). The histogram shows the cindex of all CV iterations for the datasets in the respective group. Colors indicate the group, smaller than mean (red), larger than mean (blue). The dashed lines indicate the mean values of the groups. See Figure 20 in the appendix for the ibrier.

The following groups of allocations result for the different characteristics:

Number of observations:
Group 1: COAD, ESCA, KIRC, KIRP, LAML, LGG, LIHC, OV, PAAD, SARC, SKCM
Group 2: BLCA, BRCA, HNSC, LUAD, LUSC, STAD, UCEC

Number of clinical features:
Group 1: BLCA, BRCA, COAD, ESCA, KIRP, LAML, OV, STAD
Group 2: HNSC, KIRC, LGG, LIHC, LUAD, LUSC, PAAD, SARC, SKCM, UCEC

Total number of features:
Group 1: HNSC, KIRC, LAML, LGG, LIHC, PAAD, SARC
Group 2: BLCA, BRCA, COAD, ESCA, KIRP, LUAD, LUSC, OV, SKCM, STAD, UCEC

Number of effective cases:
Group 1: COAD, ESCA, KIRC, KIRP, LAML, LIHC, PAAD, SARC, STAD, UCEC
Group 2: BLCA, BRCA, HNSC, LGG, LUAD, LUSC, OV, SKCM

Tables 9 and 10 show the mean cindex and ibrier for the different groups and the corresponding learners. The last line shows the mean cindex and ibrier across all learners per group. As seen in figure 6, the mean cindex is considerably better for a larger number of clinical features than for smaller ones and better for a small number of total features than for large values. For the ibrier this is exactly the opposite. For the number of observations and effective cases, the same tendency is evident for cindex and ibrier.

	N		$clin$		p		n_{eff}	
	$<$	$>$	$<$	$>$	$<$	$>$	$<$	$>$
Kaplan-Meier	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
Clinical only	0.618	**0.613**	0.588	0.644	0.650	**0.593**	0.624	0.603
Lasso	0.546	0.533	0.491	0.591	0.612	0.493	0.553	0.522
ipflasso	0.571	0.579	0.530	0.613	0.638	0.528	0.570	0.579
prioritylasso	0.597	0.572	0.562	0.616	0.631	0.561	0.590	0.591
prioritylasso fav	0.612	0.593	0.576	0.636	0.644	0.580	0.610	0.601
grridge	0.590	0.569	0.543	0.623	0.638	0.546	0.588	0.577
glmboost	0.542	0.529	0.483	0.590	0.621	0.479	0.555	0.507
CoxBoost	0.401	0.417	0.281	0.522	0.555	0.299	0.422	0.373
CoxBoost fav	0.617	0.606	0.582	**0.644**	0.654	**0.586**	0.615	**0.612**
blockForest	**0.635**	0.586	**0.600**	0.642	**0.674**	0.585	**0.637**	0.592
rfsrc	0.584	0.530	0.574	0.565	0.607	0.542	0.585	0.539
ranger	0.580	0.525	0.567	0.563	0.595	0.543	0.571	0.552
mean	0.569	0.550	0.529	0.596	0.617	0.526	0.571	0.550

Table 9: Performance per dataset group - cindex. Bold values indicate the highest cindex per group. The columns "<" shows the mean cindex for the groups with the smaller values of the specific dataset characteristic and ">" shows the mean cindex for the groups with the larger values. The last row shows the mean cindex over all CV iterations and learners for the specific group.

For the smaller group of each characteristic, blockForest is the best learner for cindex and ibrier, for the larger groups not even once. For large values of N simple Cox model is clearly better than the other learners for cindex, but only slightly better than CoxBoost favoring and ipflasso for ibrier. Considering the datasets with a large number of clinical features and effective cases, CoxBoost favoring is the best learner for ibrier and cindex. For a large number of total features, Cox model performs best for ibrier and cindex. The Cox model, however, only includes clinical features in the prediction and no multi-omics data. Among the learners which do consider multi-omics data, CoxBoost favoring is the best learner, but only slightly better than blockForest.

Looking at how the performance of the individual learners changes for the different groups of datasets, then we can see that for cindex, the largest differences for the two groups according to N indicate the random forest learners while for the different sizes of the variable *clin* especially the boosting and penalised learners are affected. For the total number of features all learners seem to perform much better for smaller values. Overall, CoxBoost shows the largest differences. For example, for group 1 of total number of features, it has a mean cindex of 0.555, while for group 2 it only reaches a value of 0.299. For ibrier, the random forest learners have the largest differences between the groups for the variables N, *clin* and n_{eff}. Among the groups according to p, the penalised learners have the largest differences between group 1 and 2. For ibrier it is not the performance of CoxBoost that is most influenced by the characteristics of the datasets, but the performance of the random forest learners for the variable n_{eff}. The difference between each random forest learner is 0.034. Therefore, they move from the three best performing learners in group 1 to the worst performing learners with Lasso and glmboost in group 2.

	N		*clin*		p		n_{eff}	
	<	>	<	>	<	>	<	>
Kaplan-Meier	0.177	0.186	0.170	0.188	0.190	0.172	0.172	0.194
Clinical only	0.175	**0.177**	0.174	0.177	0.188	**0.167**	0.172	0.183
Lasso	0.193	0.210	0.208	0.189	0.187	0.205	0.190	0.213
ipflasso	0.178	0.177	0.173	0.182	0.187	0.171	0.175	0.183
prioritylasso	0.177	0.189	0.177	0.183	0.187	0.175	0.177	0.187
prioritylasso fav	0.182	0.183	0.183	0.182	0.194	0.174	0.180	0.187
grridge	0.180	0.185	0.177	0.185	0.187	0.178	0.176	0.192
glmboost	0.188	0.192	0.195	0.183	0.188	0.190	0.183	0.201
CoxBoost	0.173	0.183	0.173	0.178	0.180	0.172	0.170	0.187
CoxBoost fav	0.174	0.177	0.174	**0.177**	0.184	**0.169**	0.172	**0.181**
blockForest	**0.167**	0.186	**0.165**	0.178	**0.175**	0.170	**0.160**	0.194
rfsrc	0.173	0.198	0.170	0.189	0.184	0.177	0.168	0.202
ranger	0.171	0.195	0.169	0.186	0.181	0.175	0.166	0.200
mean	0.178	0.187	0.178	0.183	0.186	0.177	0.174	0.193

Table 10: Performance per dataset group for ibrier. Bold values indicate the lowest ibrier per group. The columns "<" shows the mean ibrier for the groups with the smaller values of the specific dataset characteristic and ">" shows the mean ibrier for the groups with the larger values. The last row shows the mean ibrier over all CV iterations and learners for the specific group.

The tendencies of the learners between the individual groups usually remain the same. There are a few exceptions, such as rfsrc which is the only learner for cindex that performs better for a small number of clinical variables than for a larger number, but no learner can be identified that has a completely different tendency than all other learner.

4.3 Analysis based on learner characteristics

In this section the learners are categorised and analysed according to their modeling approach on the one hand and then, according to the respective handling of the group structure information. Table 2 gives an overview to which modeling approach each learner is assigned and how the group structure information is considered. The aim is to find out if there is a modeling approach or handling of the group structure which performs generally better than others or the reference models and to examine the individual learners of a modeling approach or handling of the group structure to analyse if there is a certain best performing learner within this group of learners.

4.3.1 Analysis of different modeling approaches

Figure 7 shows the performance distribution over all datasets for the respective modeling approaches. The first thing to notice is that there is no modeling approach that generally outperforms the simple Cox model (indicated by the red horizontal line). For cindex the modeling approaches all perform better than Kaplan-Meier (dashed black line), only the boosting approach has more values below Kaplan-Meier than the other modeling approaches, but this is due to the fact, that the zero values of CoxBoost were maintained as zeros. For ibrier the modeling approaches perform only slightly better than Kaplan-Meier. The variability of the approaches is very similar and there is no modeling approach that performs on general better than the others.

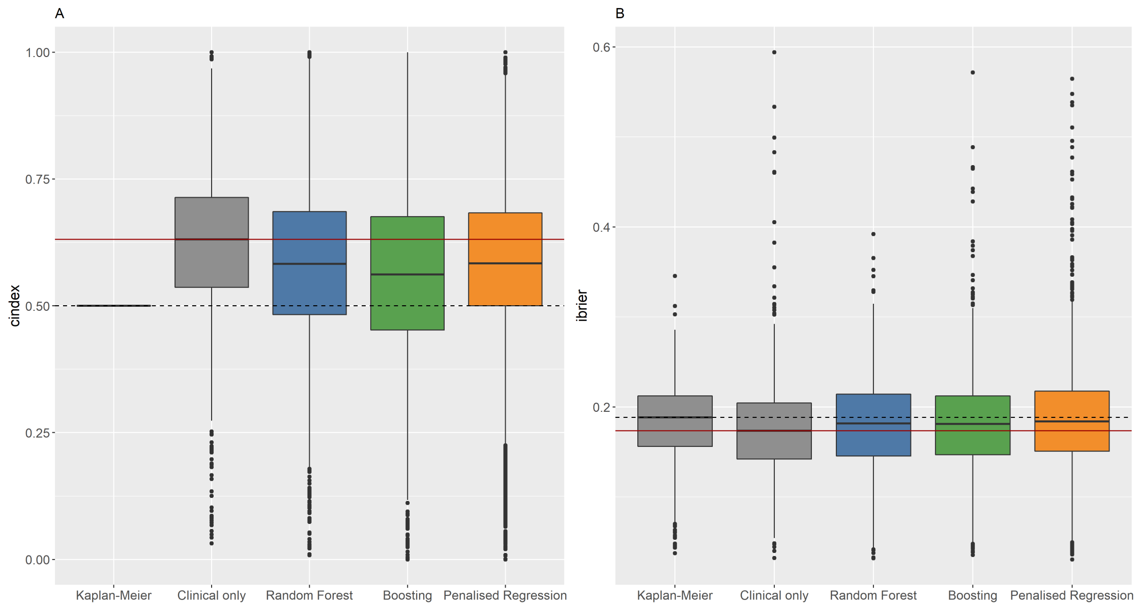

Figure 7: Comparison of modeling approaches - A: cindex, B: ibrier. The solid red and dashed black horizontal lines correspond to the median performance of the simple Cox model and Kaplan-Meier, respectively. Each boxplot includes the cindex resp. ibrier of all CV iterations over all 18 datasets of the learners which belong to the respective modeling approach.

Table 11 shows the results of the different modeling approaches. The rows "total" show the result aggregated or summed up for the entire modeling approach. Here it can be seen that the random forest methods for cindex and ibrier have the best overall mean. For cindex these learners are also the best learners for the individual datasets. For ibrier, the learners of penalised regression perform best for six datasets and thus more often than all other modelling approaches, although this modelling approach has the worst mean ibrier. The high number of datasets as best performing learners is due to the fact that ipflasso belongs to this modeling approach and performs best for ibrier for most datasets. Considering the other learners belonging to the penalised regression, prioritylasso and prioritylasso favoring do not perform best for any dataset and grridge and standard Lasso perform best for one dataset only. The boosting learners have a clearly worse mean cindex than the other modeling approaches, because of the low cindex values of CoxBoost.

By looking at the individual learners, it becomes clear that blockForest has the best mean cindex and ibrier among the random forest learners. Out of 18 datasets blockForest is 15 times better for cindex and 12 times better for ibrier than the other random forest learners. For cindex, blockForest is also most often the best learner overall (compared to all other learners), but for ibrier ranger is more often the best learner overall. For the boosting approaches, CoxBoost favoring is best learner in all categories. For the penalised approaches, it is not clear which learner performs better. For cindex, prioritylasso favoring is the best learner compared to the other penalised learners, whereas for ibrier ipflasso clearly performs best in all categories.

	cindex			ibrier		
	mean	b_total	b_approach	mean	b_total	b_approach
Kaplan-Meier	0.500	0	-	0.180	0	-
Clinical only	0.617	4	-	0.176	3	-
Random Forest						
blockForest	0.622	4	15	0.172	2	12
rfsrc	0.569	1	1	0.180	0	2
ranger	0.565	1	2	0.178	3	4
total	**0.585**	**6**	-	**0.177**	**5**	-
Boosting						
CoxBoost fav	0.614	2	15	0.175	2	11
glmboost	0.538	1	3	0.189	2	2
CoxBoost	0.405	0	0	0.176	0	5
total	**0.519**	**3**	-	**0.180**	**4**	-
Penalised Regression						
prioritylasso fav	0.607	2	6	0.182	0	2
prioritylasso	0.590	2	2	0.180	0	0
grridge	0.584	1	5	0.182	1	5
ipflasso	0.573	0	4	0.178	4	9
Lasso	0.542	0	1	0.198	1	2
total	**0.579**	**5**	-	**0.184**	**6**	-

Table 11: Comparison of modeling approaches. The column "b_total" shows the number of datasets for which the learner was the best performing learner according to the mean cindex resp. ibrier compared to all other learners. The column "b_approach" shows the number of datasets for which the learner was the best, regarding the mean cindex resp. ibrier within the respective modeling approach. The rows "total" show the respective values aggregated for the entire modeling approaches.

4.3.2 Analysis of different group structure handlings

Figure 8 shows the performance distribution across all datasets for the respective possibilities to handle group structure information. There is no handling that generally outperforms the simple Cox model (indicated by the red horizontal line), but the learners which favor clinical features perform almost as good as the cox model. For cindex all handlings perform better than Kaplan-Meier (dashed black line), but for ibrier they are only slightly better and the learners which do not take the group structure into account are even worse. The variability of the handlings is very similar, but the learners which do not consider group structure perform clearly worse than the others for both cindex and ibrier. It seems like in general, the best performers are the learners which favor the clinical features.

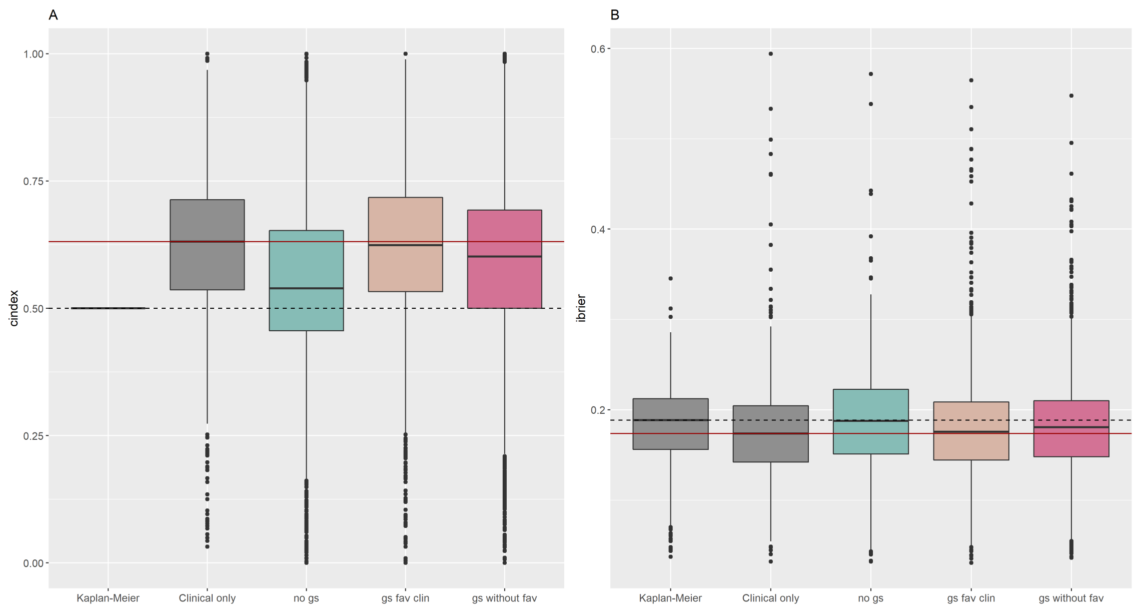

Figure 8: Comparison of group structure handlings - A: cindex, B: ibrier. The solid red and dashed black horizontal lines correspond to the median performance of the Cox model and Kaplan-Meier, respectively. Abbreviations: Kaplan-Meier = Kaplan-Meier, Clinical only = Clinical only, no gs = no group structure, gs fav clin = group structure with favoring clinical features, gs without fav = group structure without favoring. Each boxplot includes the cindex resp. ibrier of all CV iterations over all 18 datasets of the learners which handle the group structure accordingly.

Table 12 shows the results of the different handlings for using group structure. The rows "total" show the results aggregated or summed up for the entire handling. Comparing the different handlings of the group structure, the learners that favor clinical features have the highest mean cindex. The best mean ibrier, however, have the learners that take group structure into account without favoring clinical features. Within the individual handlings for cindex rfsrc has the highest mean value, but glmboost is seven times better out of 18 datasets than the other learners which do not consider group structure and rfsrc only four times. For ibrier, CoxBoost has the best mean value and performs better than the other learners for eight datasets, but of these 8 times, CoxBoost does not once become the best learner overall. For the two learners which favor clinical features, CoxBoost favoring is clearly the best learner for cindex and ibrier. Within the learners which take group structure into account without favoring clinical features, blockForest is the best learner for cindex. For the ibrier blockForest also has the lowest mean value, but ipflasso is more often best learner within this group of learners and compared to all other learners.

	cindex			ibrier		
	mean	b_total	b_group	mean	b_total	b_group
Kaplan-Meier	0.500	0	-	0.180	0	-
Clinical only	0.617	4	-	0.176	3	-
No group structure						
rfsrc	0.569	1	4	0.180	0	0
ranger	0.565	1	4	0.178	3	6
Lasso	0.542	0	2	0.198	1	1
glmboost	0.538	1	7	0.189	2	3
CoxBoost	0.405	0	1	0.176	0	8
total	**0.524**	**3**	-	**0.184**	**6**	-
Favoring clinical features						
CoxBoost fav	0.614	2	12	0.175	2	16
prioritylasso fav	0.607	2	6	0.182	0	2
total	**0.610**	**4**	-	**0.179**	**2**	-
Group structure without favoring						
blockForest	0.622	4	9	0.172	2	7
prioritylasso	0.590	2	2	0.180	0	0
grridge	0.584	1	4	0.182	1	1
ipflasso	0.573	0	3	0.178	4	10
total	**0.592**	**7**	-	**0.178**	**7**	-

Table 12: Comparison of group structure handlings. The column "b_total" shows the number of datasets for which the learner was the best performing learner according to the mean cindex resp. ibrier. The column "b_group" shows the number of datasets for which the learner was the best, regarding the mean cindex and ibrier within the respective group structure handling. The rows "total" show the respective values aggregated for the entire group structure handlings.

For cindex, of all 18 datasets the structured learners perform 11 times better than the unstructured or the Cox model. For ibrier this is the case for 9 datasets. In Table 6 in Herrmann et al. (2020) it was shown that when cindex and ibrier are aggregated across all corresponding learner types (structured/ unstructured), the structured learners perform better than the unstructured learners for cindex for 17 of 18 datasets and for 14 of 18 datasets for ibrier. It can also be seen in table 12, that the total mean of the learners which do not take the group structure into account, is only slightly better than Kaplan-Meier for cindex and even worse for ibrier. This does not hold for the structured learners.

4.4 Detailed Analysis

In this subsection the datasets and learners are analysed individually without categorizing them into specific groups. Figure 9 and 10 show the overall performance of the datasets with the performance of the best performing learner. Figure 9 shows the performance for cindex. For the last three datasets (COAD, ESCA and LUSC) the median of all learners and CV iterations is 0.5. This means, that half of the cindex values for the CV iterations and learners are worse than those of Kaplan-Meier. The variance between datasets also differs widely. While LUSC, OV and HNSC are examples of quite small variances, the variances of COAD, KIRP and LAML and their best learners are large, probably because these are the three datasets with the lowest values of effective cases n_{eff}. Interesting is also the difference between the best performing learner and all learners together. For OV and LUSC, the best learners clearly perform better than the overall performance for the dataset, while for BLCA or LIHC the difference between the performance of the best learner and the overall performance is small.

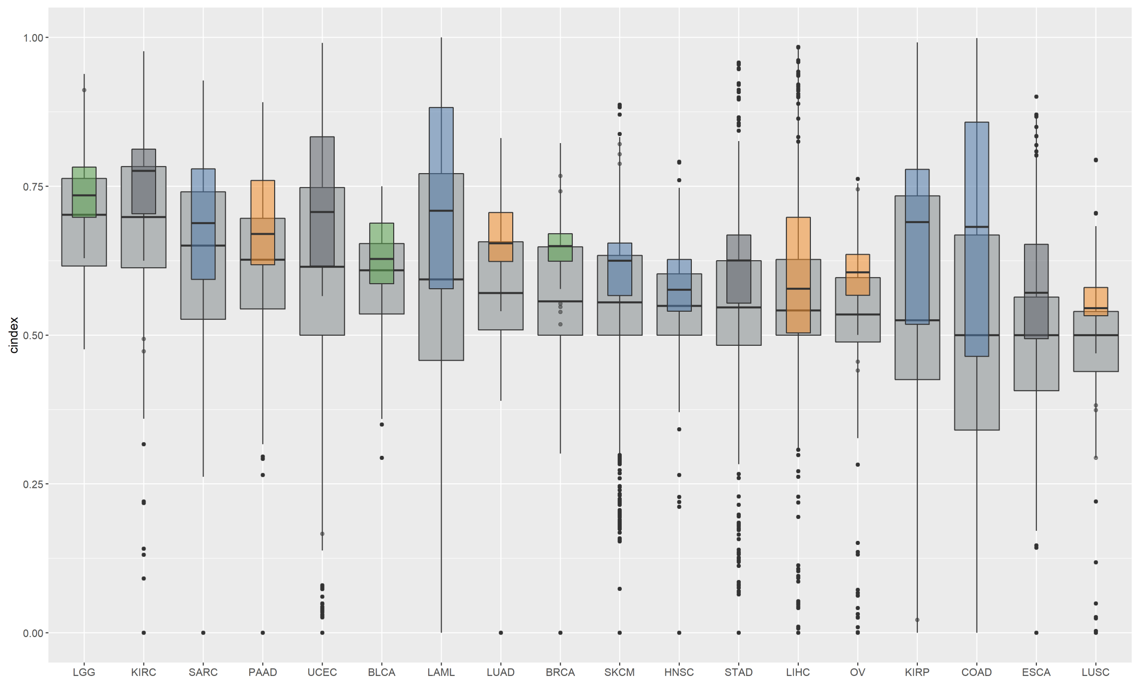

Figure 9: Performance for individual datasets - cindex. The large light gray boxplot is the performance over all CV iterations and learners, the small boxplot is the performance of the best performing learner over all CV iterations. Colors indicate the modeling approaches of the best performing learner: reference methods (dark gray), random forest (blue), boosting (green), penalised regression (orange).

Figure 10 shows the same setting for ibrier. The dataset COAD for ibrier has the lowest and therefore best median with a value of 0.082, although it is one of the datasets with a median of the data independent value 0.5 for cindex. For ibrier, no dataset has a median worse or equal the data-independent value 0.25. The highest median is 0.228 for the dataset ESCA. As for cindex, the datasets LAML and COAD are the ones with the greatest variance. It seems that the random forest learners perform better especially for datasets that generally show a good performance of all learners, while the boosting learners and the Cox model perform better especially for datasets with a higher ibrier distribution.

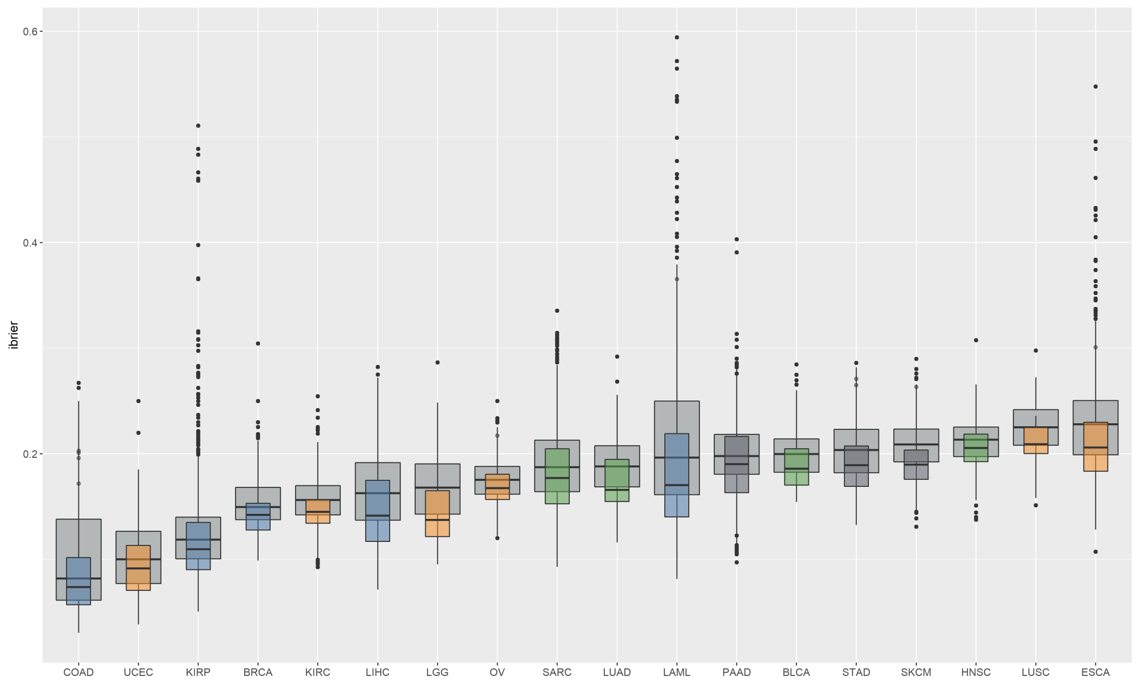

Figure 10: Performance for individual datasets - ibrier. The large light gray boxplot is der performance over all CV iterations and learners, the small boxplot is the performance of the best performing learner over all CV iterations. Colors indicate the modeling approaches of the best performing learner: reference methods (dark gray), random forest (blue), boosting (green), penalised regression (orange).

Table 13 shows a detailed view of the individual datasets. It shows the total mean and standard deviation for each dataset for cindex (A) and ibrier (B). In addition, the best performing learner and the respective mean value are shown. For comparison, the mean value of the simple Cox model is displayed. The last three columns show the mean values of the different handlings of the group structure.

A

	total		best learner		clin only	mean		
	mean	sd	learner	mean	mean	no gs	gs without fav	gs fav clin
BLCA	0.596	0.082	CoxBoost fav	0.640	0.633	0.580	0.614	0.628
BRCA	0.543	0.161	CoxBoost fav	0.643	0.637	0.467	0.587	0.619
COAD	0.493	0.249	blockForest	0.656	0.541	0.452	0.507	0.540
ESCA	0.472	0.172	Clinical only	0.574	0.574	0.405	0.483	0.552
HNSC	0.551	0.089	blockForest	0.582	0.554	0.549	0.562	0.561
KIRC	0.688	0.138	Clinical only	0.761	0.761	0.671	0.716	0.730
KIRP	0.544	0.247	rfsrc	0.648	0.572	0.527	0.557	0.567
LAML	0.587	0.241	ranger	0.709	0.596	0.548	0.651	0.599
LGG	0.686	0.104	glmboost	0.749	0.652	0.721	0.692	0.700
LIHC	0.549	0.168	grridge	0.602	0.586	0.530	0.552	0.594
LUAD	0.583	0.106	prioritylasso	0.665	0.663	0.519	0.623	0.661
LUSC	0.467	0.146	prioritylasso fav	0.537	0.531	0.406	0.484	0.536
OV	0.501	0.171	prioritylasso	0.600	0.598	0.393	0.568	0.591
PAAD	0.621	0.113	prioritylasso fav	0.686	0.683	0.581	0.654	0.682
SARC	0.638	0.133	blockForest	0.685	0.673	0.624	0.671	0.659
SKCM	0.535	0.161	blockForest	0.597	0.581	0.478	0.576	0.589
STAD	0.535	0.158	Clinical only	0.598	0.598	0.504	0.545	0.579
UCEC	0.592	0.252	Clinical only	0.686	0.686	0.526	0.652	0.634

B

	total		best learner		clin only	mean		
	mean	sd	learner	mean	mean	no gs	gs without fav	gs fav clin
BLCA	0.200	0.023	CoxBoost fav	0.190	0.192	0.203	0.199	0.195
BRCA	0.162	0.040	blockForest	0.141	0.147	0.180	0.150	0.155
COAD	0.107	0.061	blockForest	0.087	0.101	0.114	0.104	0.104
ESCA	0.231	0.051	ipflasso	0.209	0.214	0.233	0.237	0.230
HNSC	0.212	0.023	glmboost	0.202	0.210	0.214	0.211	0.210
KIRC	0.157	0.023	ipflasso	0.144	0.146	0.159	0.153	0.154
KIRP	0.132	0.058	ranger	0.118	0.140	0.133	0.127	0.143
LAML	0.210	0.073	ranger	0.182	0.231	0.211	0.198	0.225
LGG	0.167	0.033	Lasso	0.145	0.168	0.158	0.173	0.161
LIHC	0.167	0.044	ranger	0.146	0.169	0.164	0.171	0.170
LUAD	0.189	0.028	CoxBoost fav	0.172	0.172	0.199	0.185	0.173
LUSC	0.223	0.023	grridge	0.210	0.216	0.231	0.221	0.218
OV	0.179	0.026	ipflasso	0.169	0.173	0.189	0.172	0.173
PAAD	0.200	0.034	Clinical only	0.190	0.190	0.205	0.199	0.192
SARC	0.191	0.042	glmboost	0.179	0.202	0.180	0.191	0.208
SKCM	0.208	0.024	Clinical only	0.191	0.191	0.221	0.204	0.192
STAD	0.204	0.031	Clinical only	0.192	0.192	0.211	0.201	0.196
UCEC	0.107	0.047	ipflasso	0.091	0.092	0.115	0.105	0.098

Table 13: Detailed analysis of individual datasets - A: cindex, B: ibrier. The second and third column show the total mean and standard deviation over all learners and CV iterations. The columns "best learner" show the best learner with the corresponding mean. The column "clin only" shows the mean for the Cox model. The last three columns show the mean per handling of group structure. All means are obtained by averaging over all CV iterations. Abbreviations: clin only: Cox model, no gs: no group structure, gs without fav: group structure without favoring clinical features, gs fav clin: favoring clinical features.

Looking first at the performance of the Cox model over the different datasets, one can see a similarity with the performance of the learners which favor the clinical features. The Cox model almost always performs best when the learners, which favor the clinical features perform best compared to the other group structure handlings. On the contrary, the Cox model is outperformed by at least two groups of learners that take into account the group structure of the multi-omics data for the datasets HNSC, LAML and LGG for cindex. For ibrier, this is the case for KIRP, LAML, LGG and SARC. Looking at the performance of the learners which favor the clinical features for these datasets, it can be seen that at least one different handling of the group structure outperforms these learners. One could therefore assume that the Cox model often performs similarly to the learners, which favor the clinical features. Using a correlation test, the correlation between the cindex of the Cox model and the learners which favor the clinical features is 0.892 and for ibrier 0.965, with a p-value far below 0.05. It is questionable whether there are certain features of a dataset for which the Cox model or the learners which favor the clinical features outperform the other learners. Looking at the individual datasets for which either the Cox model or a learner that favors clinical features performs best for at least one of the performance measures, the mean value of total features is 99858 whereas the mean is 97497 for datasets where another learner performs best. Therefore, one could assume that, especially for a high number of features in total, it would be better to favor clinical features or only to consider the clinical features (see table 1 for the number of total features per dataset). This assumption would also be in accordance with table 9 and 10, where it was shown, that CoxBoost favoring and the simple Cox model have the best mean cindex resp. ibrier for datasets with a large value of total features. A similar assumption can be drawn for the number of observations, since the average number of observations for datasets where the Cox model or a learner that favors clinical features performs best is 339, whereas the mean for the datasets for which none of these learners performs best is 186. However, there are also exceptions to these assumptions, such as the datasets COAD and KIRP, where the total number of features exceeds 100000, but neither the Cox model nor a learner that favors the clinical features perform best for ibrier or cindex.

If we now look at the datasets which have a very small number of effective cases, e.g. COAD, KIRP and LAML, each have a maximum of 20 effective cases, we see that for these learners the standard deviation is highest. A clear trend can be seen, visualizing the standard deviation with the corresponding number of effective cases (see figure 11 1A and 1B). Especially for ibrier, it can be seen, the higher the number of effective cases, the lower the standard deviation of the performance of the dataset.

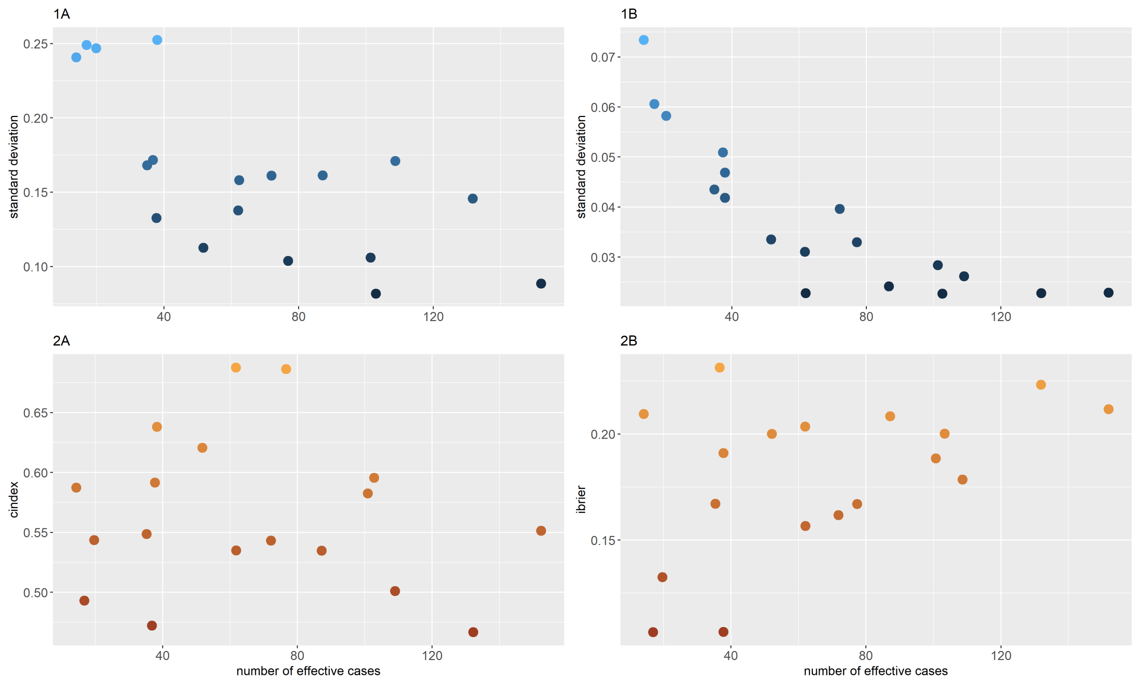

Figure 11: Standard deviation and mean value according to the number of effective cases - standard deviation (1), mean cindex resp. ibrier (2), cindex (A), ibrier (B).

The second row of figure 11 shows the mean cindex (A) and ibrier (B) for the corresponding number of effective cases. The mean values are averaged over all learners and CV iterations for each dataset. In contrast to the standard deviation, no clear trend can be seen here. However, contrary to expectations, one could assume that the smaller the number of effective cases, the smaller and therefore better the ibrier. This would also be in accordance with table 10. Here it can be seen that the mean ibrier for datasets with smaller n_{eff} values is better than the one for datasets with large n_{eff} values. For the mean cindex there is hardly any difference between

datasets with small or high values of effective cases.
Considering the total performance for the individual datasets in table 13, for ibrier and cindex the total performance is worst for the datasets ESCA and LUSC. For these datasets, the average cindex is below the data independent value 0.5, since none of the learners that do not take the group structure into account have a cindex above 0.5. Furthermore, for 33 out of the 50 CV iterations for the dataset ESCA CoxBoost has zero values (see table 6). For ibrier grridge is the only learner which has a value above the data independent value 0.25 for the dataset ESCA. Nevertheless, many other learners have an ibrier value close to 0.25 and every learner has a value above 0.2 for the datasets ESCA and LUSC. In addition, table 3 in section 3, shows that Lasso, glmboost and ipflasso have a high percentage of missing values for these datasets. Looking at the key figures of these datasets in table 1, one could deduce the poor performance of the learners for ESCA from the fact that it has a small number of clinical features, a small number of observations and a small number of effective cases but a large number of total features. However, this is not the case for LUSC. This dataset has many effective cases, a large number of observations and many clinical features. Interestingly, the two datasets COAD and UCEC, which have the lowest ratio of effective cases with a value of 0.09, have the best mean ibrier. For cindex, the total performance for the dataset KIRP is best. It does not seem to be possible to explain the average performance of the learners based on different key figures of these datasets.

Based on previous analyses, especially blockForest, the simple Cox model, CoxBoost favoring and ipflasso perform very well. Therefore, these learners will be examined more closely in the following. Table 14 shows these learners with the corresponding mean cindex resp. ibrier and rank per dataset.
Overall, the ranks between the individual datasets vary greatly. Although blockForest has the best mean cindex and ibrier overall, blockForest has rank 8 for cindex once and rank 11 for ibrier three times. For most datasets, ipflasso has the best mean for ibrier, but for cindex it has a mean value below the data independent value 0.5 for the five datasets COAD, ESCA, KIRP, LIHC and LUSC. The biggest difference is for the dataset ESCA. Here ipflasso is on rank 12 for cindex, but for

ibrier on rank 1 and performs therefore better than all other learners.
For cindex blockForest and the Cox model are four times on rank 1 each, CoxBoost favoring two times and ipflasso not once. For ibrier ipflasso is four times on rank 1, the Cox model three times and blockForest and CoxBoost favoring two times each. Comparing the performance of these four learners with each other, for the datasets for which none of them is best performing learner, then the result is quite evenly distributed for cindex. Here blockForest performs three times better, the Cox model and CoxBoost favoring twice and ipflasso once. Comparing the four learners for the ibrier, blockForest performs four times better, the Cox model and ipflasso once each and CoxBoost favoring for no dataset.

This overview shows how large the difference in performance can be for one learner for different datasets. For example, if for the benchmark experiment only the datasets ESCA, LUSC, OV and STAD were available, the performance of blockForest would be very poor compared to the other learners, although previous analyses showed that blockForest often performs best.

A

	best learner	blockForest		Clinical only		CoxBoost fav		ipflasso	
	mean	mean	rank	mean	rank	mean	rank	mean	rank
BLCA	0.640	0.619	4	0.633	2	0.640	1	0.626	3
BRCA	0.643	0.627	4	0.637	2	0.643	1	0.574	8
COAD	0.656	0.656	1	0.541	5	0.553	4	0.427	11
ESCA	0.574	0.530	4	0.574	1	0.558	2	0.422	12
HNSC	0.582	0.582	1	0.554	6	0.574	4	0.542	10
KIRC	0.761	0.740	3	0.761	1	0.729	7	0.755	2
KIRP	0.648	0.575	4	0.572	6	0.561	8	0.455	12
LAML	0.709	0.701	3	0.596	7	0.607	6	0.663	4
LGG	0.749	0.718	4	0.652	12	0.712	5	0.676	11
LIHC	0.602	0.590	3	0.586	5	0.602	2	0.485	12
LUAD	0.665	0.609	6	0.663	3	0.663	4	0.664	2
LUSC	0.537	0.463	8	0.531	3	0.534	2	0.441	10
OV	0.600	0.556	6	0.598	2	0.585	4	0.583	5
PAAD	0.686	0.677	4	0.683	2	0.678	3	0.642	6
SARC	0.685	0.685	1	0.673	4	0.665	5	0.674	3
SKCM	0.597	0.597	1	0.581	6	0.590	3	0.551	8
STAD	0.598	0.558	5	0.598	1	0.569	3	0.563	4
UCEC	0.686	0.675	2	0.686	1	0.654	4	0.661	3

B

	best learner	blockforest		Clinical only		CoxBoost fav		ipflasso	
	mean	mean	rank	mean	rank	mean	rank	mean	rank
BLCA	0.190	0.195	4	0.192	2	0.190	1	0.193	3
BRCA	0.141	0.141	1	0.147	4	0.149	6	0.152	9
COAD	0.087	0.087	1	0.101	8	0.107	10	0.144	11
ESCA	0.209	0.219	4	0.214	2	0.216	3	0.209	1
HNSC	0.202	0.218	11	0.210	8	0.203	4	0.203	2
KIRC	0.144	0.146	2	0.146	3	0.152	5	0.144	1
KIRP	0.118	0.120	3	0.140	11	0.140	10	0.129	6
LAML	0.182	0.185	3	0.231	10	0.215	8	0.198	5
LGG	0.145	0.180	11	0.168	8	0.155	4	0.159	5
LIHC	0.146	0.149	2	0.169	9	0.166	7	0.212	13
LUAD	0.172	0.202	11	0.172	2	0.172	1	0.172	3
LUSC	0.210	0.232	10	0.216	4	0.217	5	0.224	8
OV	0.169	0.177	10	0.173	7	0.172	5	0.169	1
PAAD	0.190	0.190	3	0.190	1	0.190	2	0.195	5
SARC	0.179	0.180	4	0.202	11	0.203	12	0.192	7
SKCM	0.191	0.209	6	0.191	1	0.192	2	0.197	4
STAD	0.192	0.202	6	0.192	1	0.196	4	0.193	2
UCEC	0.091	0.093	3	0.092	2	0.096	4	0.091	1

Table 14: Detailed analysis of specific learners - A: cindex, B: ibrier. The second column shows the mean cindex resp. ibrier for the best performing learner per dataset. The other columns show the mean and rank for the specific learner and dataset. All mean values are obtained by averaging over all CV iterations.

The detailed analysis showed that the performances of the learners vary greatly for different datasets. For example, for some datasets, the performance of all learners is quite poor although this dataset has a large number of effective cases, while the performance of the learners is very good for datasets with a small number of effective cases. For some cases it seems random how the learners perform for a particular dataset. The same holds when looking at individual learners for each dataset. A learner that performs very well in general, may perform much worse than all other learners for specific datasets. It has also been seen that the simple Cox model often performs similarly to the learners, which favor clinical features. For the effective cases it was shown that the smaller the number of effective cases, the higher the variance.

4.5 Testing for significance

In this section the differences of performance of the learners are tested for significance. First, a one-way analysis of variance (ANOVA) with repeated measures is carried out. Afterwards, the differences of the mean values are examined by a post-hoc analysis. A paired t-test is applied, which adjusts the p-values using the procedure by Holm (Fu et al. (2014)). For the statistical tests, the zero values of the learners were treated as missing values, since CoxBoost is strongly affected and otherwise the approximate normal distribution cannot be assumed.

Probably the best-known statistical tests are the analysis of variance (ANOVA) and the standard t-test. One assumption for these tests is that the observations are independently obtained, but since in this case the different learners were performed on the same CV iterations, this assumption is not confirmed. Therefore, ANOVA for repeated measures is used in the following. Here the hypothesis $H_0 = \mu_1 = \mu_2 = ... = \mu_k$ is tested, i.e. the alternative hypothesis is $H_1 : \mu_i \neq \mu_j$ for at least one pair (i, j). In this benchmark study this means that it is tested whether the mean values of the performance measures are the same across all learners or if there is at least one pair of learners for which this does not hold. Additionally, similar variances between the learners are assumed. This is given here since the ratio between the smallest and largest variance is 1.297 for cindex and 1.853 for ibrier, see Blanca et al. (2018). However, Kaplan-Meier must be excluded for testing using cindex, since it has a variance of 0 due to the fact that Kaplan-Meier always has the data independent value of 0.5.

For this study, the repeated measures ANOVA contains one between-subjects factor (datasets) and one within-subjects factor (learners). The CV iterations are treated as individuals within one dataset. From this, we get the group design as described in Huck and McLean (1975) and illustrated in table 15. The values $X_{i,j,k}$ stand for the cindex respectively ibrier in dataset i, for the learner j and the CV iteration k.

			factor B (learners)			
		iteration	learner 1	learner 2	...	learner p
factor A (datasets)	BLCA	BLCA-1	$X_{1,1,1}$	$X_{1,2,1}$	...	$X_{1,p,1}$
		BLCA-2	$X_{1,1,2}$	$X_{1,2,2}$	...	$X_{1,p,2}$
		⋮	⋮	⋮	⋮	⋮
		BLCA-25	$X_{1,1,25}$	$X_{1,2,25}$	...	$X_{1,p,25}$
	BRCA	BRCA-1	$X_{2,1,1}$	$X_{2,2,1}$	...	$X_{2,p,1}$
		BRCA-2	$X_{2,1,2}$	$X_{2,2,2}$	...	$X_{2,p,2}$
		⋮	⋮	⋮	⋮	⋮
		BRCA-25	$X_{2,1,25}$	$X_{2,2,25}$	...	$X_{2,p,25}$
	⋮	⋮	⋮	⋮	⋮	⋮
	UCEC	UCEC-1	$X_{18,1,1}$	$X_{18,2,1}$	...	$X_{18,p,1}$
		UCEC-2	$X_{18,1,2}$	$X_{18,2,2}$	...	$X_{18,p,2}$
		⋮	⋮	⋮	⋮	⋮
		UCEC-25	$X_{18,1,25}$	$X_{18,2,25}$	...	$X_{18,p,25}$

Table 15: Analysis of variance - group design. The number of learners is 12 for cindex ($p = 12$), since Kaplan-Meier is excluded while the number for ibrier is 13 ($p = 13$), since Kaplan-Meier can be included for ibrier.

After applying the ANOVA, we get three F tests: one for the main effect of datasets, one for the main effect of learners, and one for the datasets $\times$ learners interaction. For ibrier and cindex all three p-values are far below 0.05, which means that the null hypothesis can be rejected for cindex as well as for ibrier and there is at least one pair of learners which do have a significantly different mean performance. As described in Wollschläger (2016), after applying ANOVA, paired t-tests can be performed to test pairwise group differences for significance. The α value should be adjusted to ensure that the actual α level is below the nominal level. Table 16 shows the significance of pair comparisons with the α adjustment according to Holm (Fu et al. (2014)). The paired t-test does not test a two-sided hypothesis as in ANOVA, but a one-sided, since the main interest of this benchmark study is whether one learner performs significantly better than another. This means the hypothesis $H_0 : \mu_i \geq \mu_j$ is tested.

A

	BF	CoxPH	CoxB$_f$	prior$_f$	prior	GRr	IPF	rfsrc	ranger	glmB	Lasso
CoxPH	ns										
CoxB$_f$	ns	ns									
prior$_f$	ns	ns	ns								
prior	***	**	**	ns							
GRr	****	***	***	*	ns						
IPF	****	****	****	****	*	ns					
rfsrc	****	****	****	****	ns	ns	ns				
ranger	****	****	****	****	*	ns	ns	ns			
glmB	****	****	****	****	****	****	*	ns	ns		
Lasso	****	****	****	****	****	****	*	ns	ns	ns	
CoxB	****	****	****	****	****	****	*	ns	ns	ns	ns

B

	BF	CoxB$_f$	CoxB	CoxPH	ranger	IPF	KM	rfsrc	prior	GRr	prior$_f$	glmB
CoxB$_f$	ns											
CoxB	**	ns										
CoxPH	ns	ns	ns									
ranger	****	ns	ns	ns								
IPF	***	ns	ns	ns	ns							
KM	****	ns	****	ns	ns	ns						
prior	****	*	ns	ns	ns	ns	ns					
rfsrc	****	ns	***	ns	****	ns	ns	ns				
GRr	****	*	****	ns	ns	ns	ns	ns	ns			
prior$_f$	****	****	**	****	ns	ns	ns	ns	ns	ns		
glmB	****	****	****	****	****	****	****	****	****	***	*	
Lasso	****	****	****	****	****	****	****	****	****	****	****	****

Table 16: Significance of pairwise t-tests - A: cindex, B: ibrier. Level of significance is indicated by *, $p \leq 0.05$ (*), $p \leq 0.01$ (**), $p \leq 0.001$ (***), $p \geq 0.05$ (ns). Abbreviations: KM = Kaplan-Meier, Lasso = Lasso, glmB = glmboost, CoxB = CoxBoost, CoxPH = Clinical only, prior = prioritylasso, prior$_f$ = prioritylasso favoring, IPF = ipflasso, CoxB$_f$ = CoxB favoring, GRr = grridge, BF = blockForest, rfsrc = rfsrc, ranger = ranger.

In table 16, the learners are sorted in descending order of mean value, therefore a descending trend of significance can be observed. For cindex (A), the null hypothesis for the comparison of blockForest and the Cox model, CoxBoost favoring and prioritylasso favoring cannot be rejected. For all other learners one can reject the null hypothesis that blockForest performs worse or equal. There is no learner that performs significantly better than the simple Cox model, on the other hand the Cox model performs significantly better than all learners except blockForest, CoxBoost favoring and prioritylasso favoring for cindex. Considering the ibrier (B) substantial differences can be observed. For example, while no learner performs significantly better for prioritylasso favoring for cindex, BlockForest, CoxBoost favoring, CoxBoost and the simple Cox model perform significantly better for ibrier. Furthermore, for ibrier only BlockForest and CoxBoost perform significantly better than Kaplan-Meier and glmboost and Lasso significantly worse. Overall, every learner seems to perform better than Lasso for ibrier.

5 Sampling

In this section datasets are drawn randomly, and it is analyzed how often a learner performs best for certain samples. First, the datasets are drawn with equal probability, then the probability varies between datasets, and finally, specific groups of learners are compared. The goal of this procedure is to find out if there are certain learners that perform more often better than other learners for given samples.

5.1 Random sampling

In this section datasets are randomly selected and a best learner for the random sample of datasets is determined. Since there are 18 datasets in total, the probability of a dataset D_i being drawn into the sample is given by $P(D_i) = \frac{1}{18}$. Each sample contains a fixed number k of datasets. For example, five datasets are drawn from all 18 possible datasets and then the best learner is determined for these five datasets by averaging the cindex or ibrier over the whole sample and all CV iterations. Two to 16 datasets per sample are drawn without repetition and each sample is repeated 50 times. Therefore, there are 750 samples in total. Figure 12 shows the number of samples for which a learner has performed best across all sample sizes. Figure 13 shows the more detailed view of how many samples a learner performed best for each sample size. As illustrated in figure 12, blockForest is obviously most often best learner for both ibrier (A) and cindex (B) with over 60% of samples each. Apart from this, many differences are observable between ibrier and cindex. The greatest difference occurs for ipflasso, while it is second most often best learner for ibrier with a frequency of 112 samples, it is only for two samples best learner for cindex. The differences for the Cox model are also considerable. Cox model performs best for 211 (28%) samples for cindex, but only 68 times (9%) for ibrier. CoxBoost is not once best learner for cindex, even if the zero values are treated as missing values, but 32 (6%) times for ibrier.

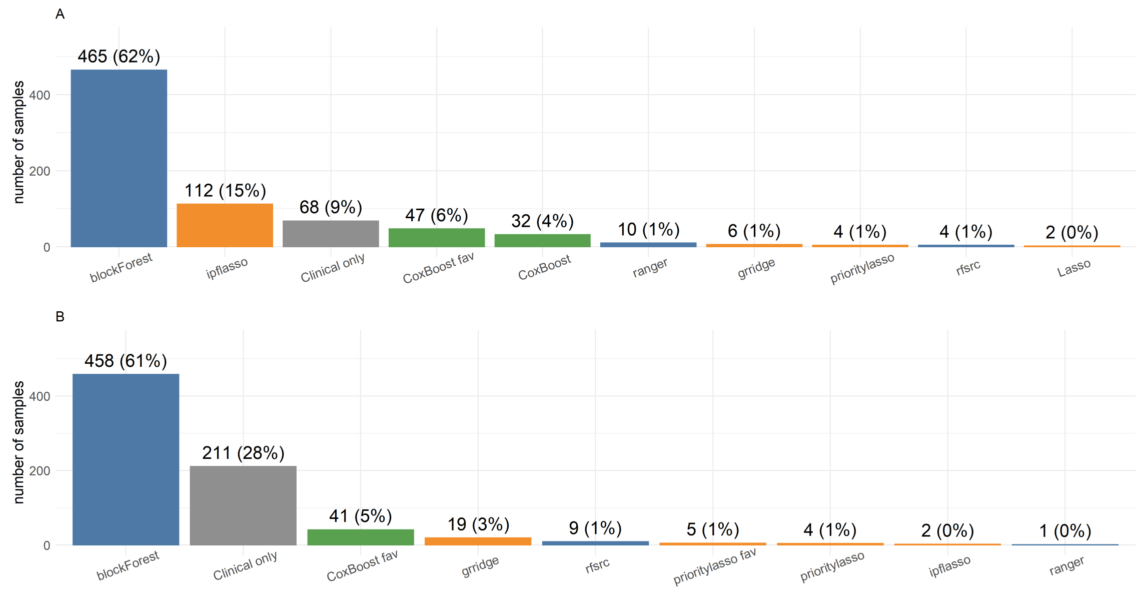

Figure 12: Best performing learners for random samples - A: ibrier, B: cindex. Colors indicate the modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange).

Figure 13 shows the results per sample size. The y-axis shows the number of datasets that were drawn into the sample, i.e. the sample size. For ibrier and cindex blockForest becomes best learner more often for large sample sizes. For ibrier blockForest is best learner for 49 out of 50 samples with the maximum sample size 16. Ipflasso seems to perform better than Clinical only for small sample sizes but worse for large sample size. The other learners however seem to perform better for smaller sample sizes and therefore, the variability between the best learners is higher for small sample sizes. It is also interesting to mention that blockForest performs best most often for the sample size of 2 datasets neither for ibrier nor cindex. For this sample size ipflasso is most often the best learner for ibrier and the simple Cox model for cindex. The same applies for ibrier for a sample size of 5 datasets.

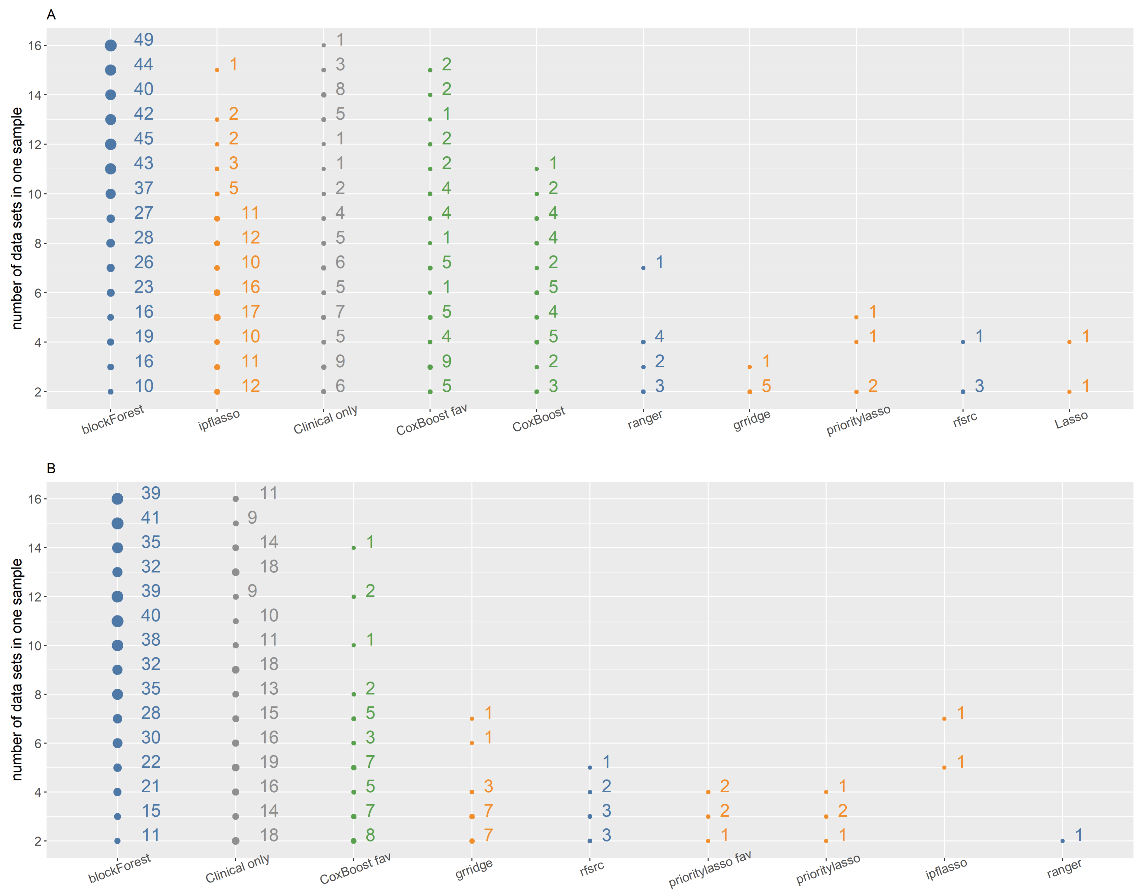

Figure 13: Detailed view of the best performing learners for random samples - A: ibrier, B: cindex. Colors indicate the modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange). The size of the circles is proportional to the number of samples as best performing learner.

Table 17 shows the frequencies of the best learners for the different imputation methods. For ibrier (A) there are large differences especially for the naive imputation methods. For the method with a threshold at 20%, the data independent and the weighted method blockForest is the learner with most samples, as best performing learner. For the mean value method, ipflasso performs for 412 samples best, while blockForest performs best for only 240 samples. For the method with a threshold at 20% and the weighted method, ipflasso is the second-best learner, while for the data independent method the simple Cox model and CoxBoost favoring perform better more often. The large differences for ipflasso are due to the fact that ipflasso performs well for ibrier, but for some datasets it has a rather high percentage of missing values. The largest difference between the method at a threshold of 20% and the weighted method also occurs for ipflasso. Apart from that the number of samples as best performing learners is quite similar when comparing these two imputation methods. For the cindex (B) there are hardly any differences between the imputation methods. Here the frequencies match almost perfectly, and the ranks of the learners remain the same, except for prioritylasso. This again shows how the influence of the imputation method also depends on the evaluation measure.

A

	threshold 20%	weighted	data independent	mean value
blockForest	465	434	497	240
ipflasso	112	164	56	412
Clinical only	68	57	81	22
CoxBoost fav	47	38	58	25
CoxBoost	32	30	32	15
ranger	10	10	10	10
grridge	6	6	6	6
prioritylasso	4	4	4	3
rfsrc	4	4	4	4
Lasso	2	1	0	12
glmboost	0	2	1	1
Kaplan-Meier	0	0	1	0

B

	threshold 20%	weighted	data independent	mean value
blockForest	458	459	460	457
Clinical only	211	211	212	209
CoxBoost fav	41	41	42	40
grridge	19	19	16	19
rfsrc	9	9	9	9
prioritylasso fav	5	5	5	5
prioritylasso	4	5	4	8
ipflasso	2	0	0	2
ranger	1	1	2	1

Table 17: Number of random samples as best performing learner for different imputation methods- A: ibrier, B: cindex.

5.2 Sampling by dataset characteristics

The samples in this section are designed in the same way as in section 5.1, only the probabilities for drawing a dataset are calculated differently. The probabilities are now proportional to the size of certain dataset characteristics. For example, if the number of observations N is considered, the probability of drawing a dataset D_i into the sample is given by

$$P(D_i) = \frac{N_i}{\sum_{j=1}^{18} N_j}. \tag{19}$$

This means that the more observations a dataset has, the higher the probability that this dataset will be drawn into the sample. Figure 14 shows the number of samples for which a learner performs best by calculating the probabilities proportional to the total number N of observations. For ibrier (A) and cindex (B) Cox model performs best more often than blockForest, for cindex even in more than the half of all samples. This is in accordance with tables 9 and 10, which show that the simple Cox model has the best mean cindex and ibrier when comparing the learners for the group of datasets for which the number of observations is large. Ipflasso is now in third place for ibrier, but with a higher number of samples than for the random samples. BlockForest is less likely to be best performing learner than if the datasets are all drawn with the same probability. CoxBoost favoring remains in fourth place for ibrier and in third place for cindex, but performs best for ibrier and cindex more often, as in section 5.1.

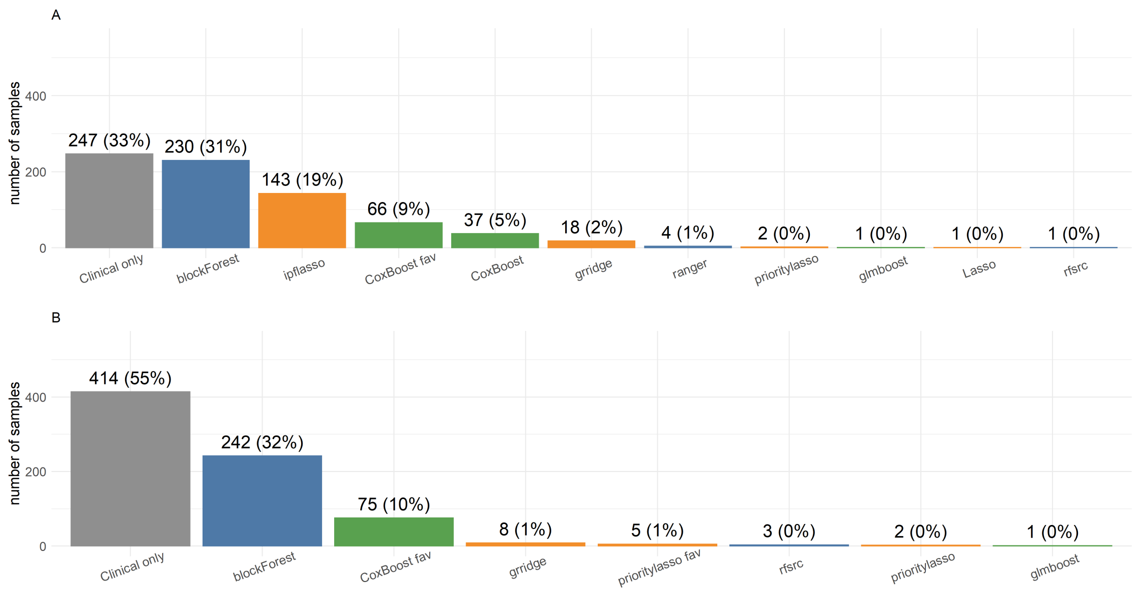

Figure 14: Best performing learners for sampling by dataset size - A: ibrier, B: cindex. Colors indicate the modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange).

As can be seen in Figure 15, when drawing the probabilities proportional to the number of effective cases (n_{eff}) ipflasso performs most frequently best for ibrier (A) and Cox model for cindex (B). For ibrier the frequencies of the first four learners are quite similar distributed, while for cindex the simple Cox model performs obviously most frequently best (59%). Comparing the samples which are drawn according to the number of effective cases with the previous samples, then blockForest performs best for the lowest percentage of samples. BlockForest is the best performing learner for random samples in slightly over 60%, for both cindex and ibrier and for the samples according to effective cases in only 22%.

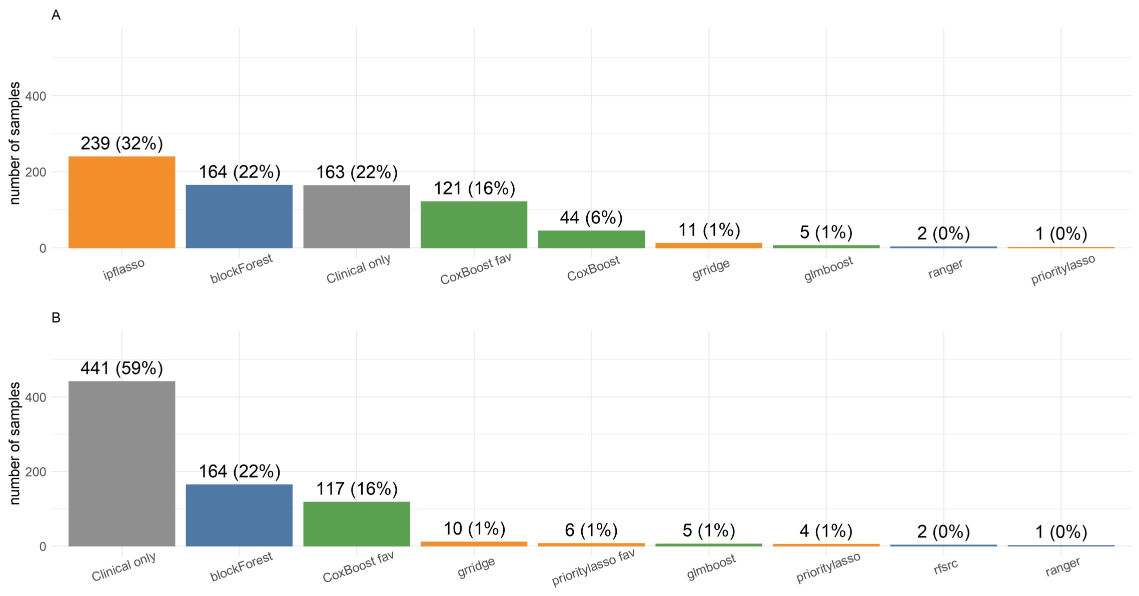

Figure 15: Best performing learners for sampling by effective cases - A: ibrier, B: cindex. Colors indicate the modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange).

When calculating the probabilities proportional to the total number of features and proportional to the number of clinical features, the distribution of samples of the best performing learners is very similar to that of the random sampling, see figure 22 and 23 in the appendix.

Considering the results of the samples per sample size, as shown in figure 16 and 17, the simple Cox model for cindex performs best for almost every sample size, for probabilities proportional to N and n_{eff}. However, this is not true for ibrier. If the samples are drawn proportional to the number of observations, the Cox model, as shown in figure 14, performs best for most datasets. However, for the detailed view (figure 16), it can be seen that for large samples (samples of 11, 14, 15 and 16 datasets) blockForest performs best most often and for samples of size two, three and five, ipflasso performs best most often. For the samples drawn proportionally to the number of effective cases, according to figure 15 ipflasso is the learner that performs best most often. After a closer examination (figure 17), it is shown that blockForest and the Cox model for sample sizes of 14, 15 and 16 perform better than ipflasso. In addition, it can be observed especially in figure 17 that CoxBoost favoring often performs best for certain sample sizes.

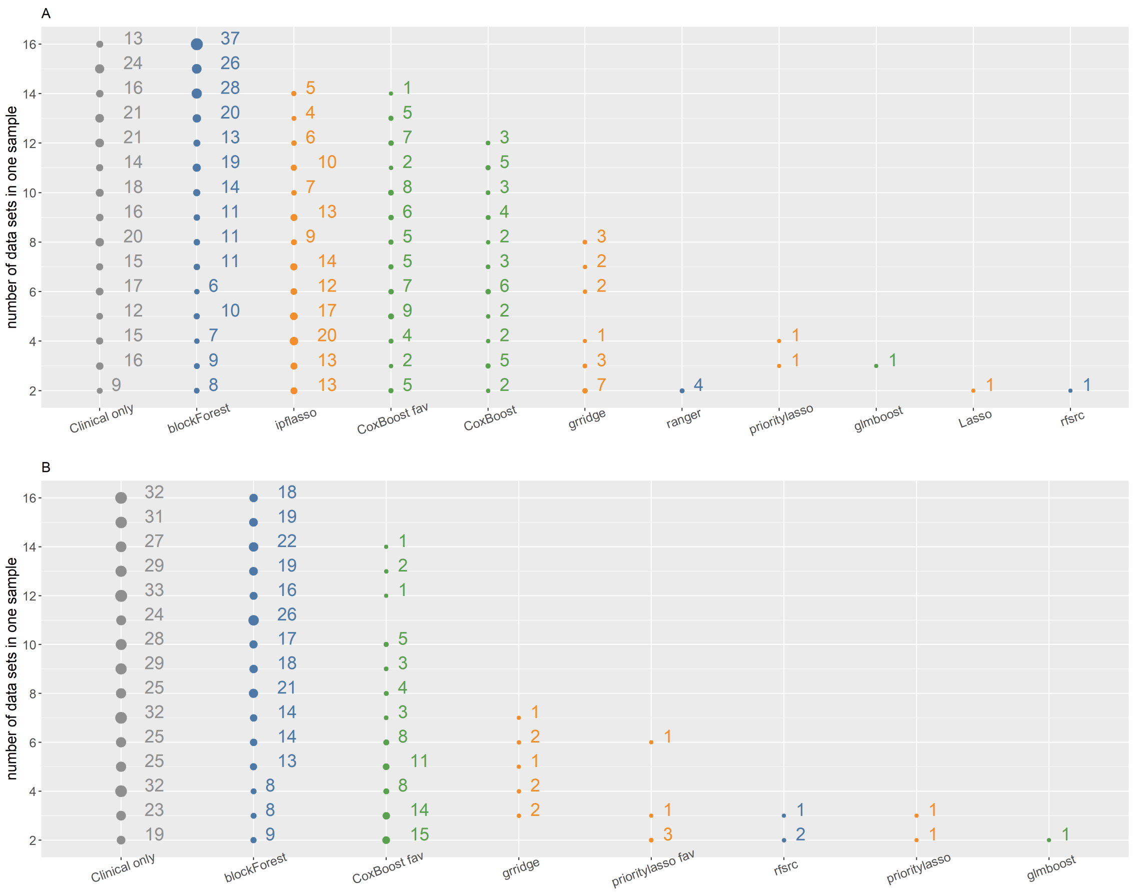

Figure 16: Detailed view of the best performing learners for sampling by dataset size - A: ibrier, B: cindex. Colors indicate the modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange). The size of the circles is proportional to the number of samples as best performing learner.

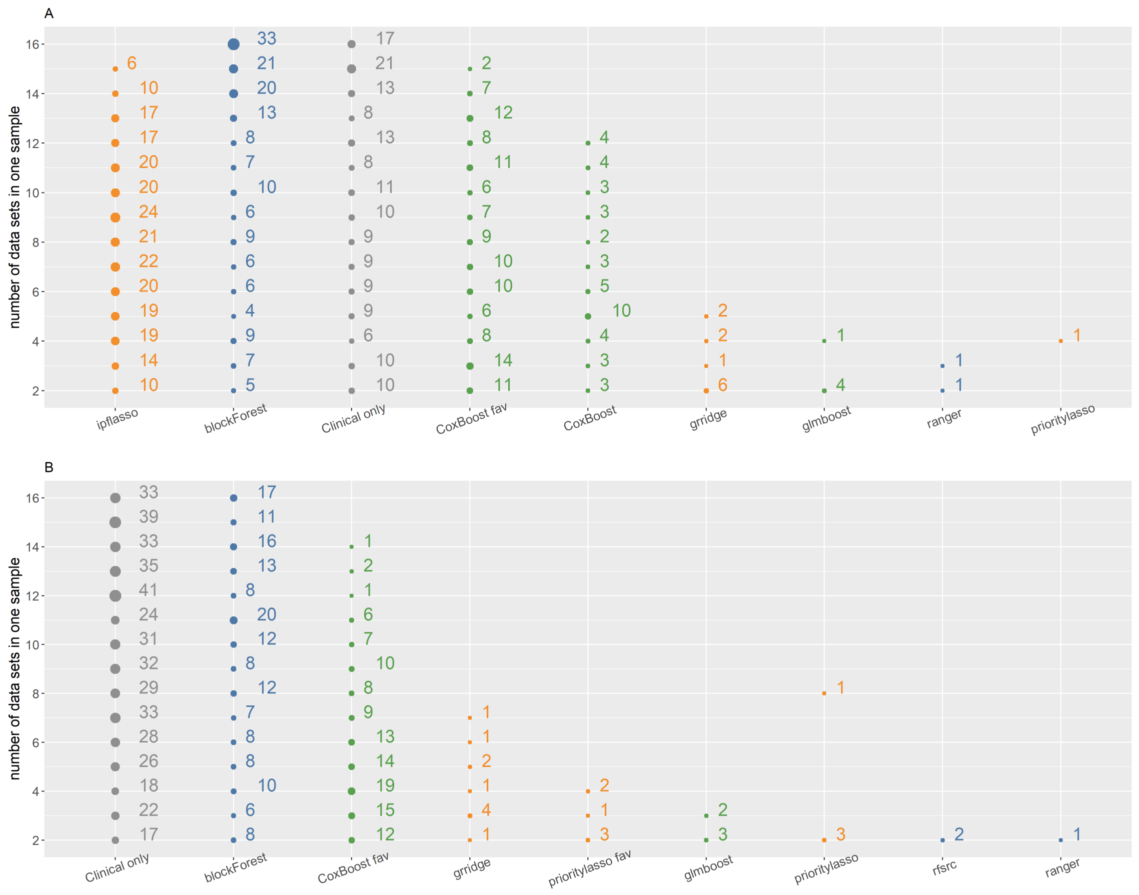

Figure 17: Detailed view of the best performing learners for sampling by effective cases - A: ibrier, B: cindex. Colors indicate the modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange). The size of the circles is proportional to the number of samples as best performing learner.

In contrast to Section 4, the number of samples is now large enough to say that certain characteristics of a dataset influence the performance of a learner. It seems that the simple Cox model is still the best prediction method when the number of observations is large enough. This is the case for ibrier and cindex. For a large number of effective cases, the difference between the Cox model and blockForest has become even greater for cindex, while ipflasso scores best most often for ibrier. However, caution should be taken with such statements. Looking at individual sample sizes, different results are obtained, especially for the ibrier. Here, blockForest

still performs best for large samples.
If the datasets are drawn randomly, blockForest is the best performing learner, almost for every single sample size and imputation method. Only for the mean value imputation method ipflasso was the best performing learner aggregated over all sample sizes.

5.3 Sampling within groups of learners

In this section the learners within each modeling approach and handling of group structure are compared for random samples, i.e. the sample design is the same as in section 5.1, with 50 repetitions of each sample size. That means, for each group of learners, 750 samples are considered. Within each learner group, the simple Cox model is always included in order to be able to compare the performance of the learners with the reference model.

In figure 18 the results for the different modeling approaches are presented. For ibrier (A) and cindex (B), blockForest is the learner that performs best most frequently (74% for ibrier and 65% for cindex), in the random forest group of learners (1). After that comes Cox model with 24% and 33%. Ranger and rfsrc are best learners for not more than 3% of the samples for ibrier and cindex. For the boosting approaches (2) there are major differences between ibrier and cindex. For ibrier, CoxBoost performs best most often (for 42% of the samples), while it does not perform best at all for cindex. The simple Cox model clearly performs best for most datasets for cindex. CoxBoost favoring is second for both evaluation measures, while glmboost performs best for ibrier in only 1% of samples and for no sample for cindex. For the penalised methods (3), Cox model performs best for both measures. For ibrier, however, ipflasso also performs best for a large number of datasets, while for cindex all penalised learners together perform better than Cox model, in only 11% of datasets. For cindex Cox model thus performs better than the learner in the respective group in more than half of the datasets for the boosting and penalised methods. Only for the random forest learners blockForest performs clearly better more often than the simple Cox model. Looking at the ibrier, there is at least one learner that performs better than Cox model for both the random forest methods and the boosting methods. For the penalised methods Cox model is in first place.

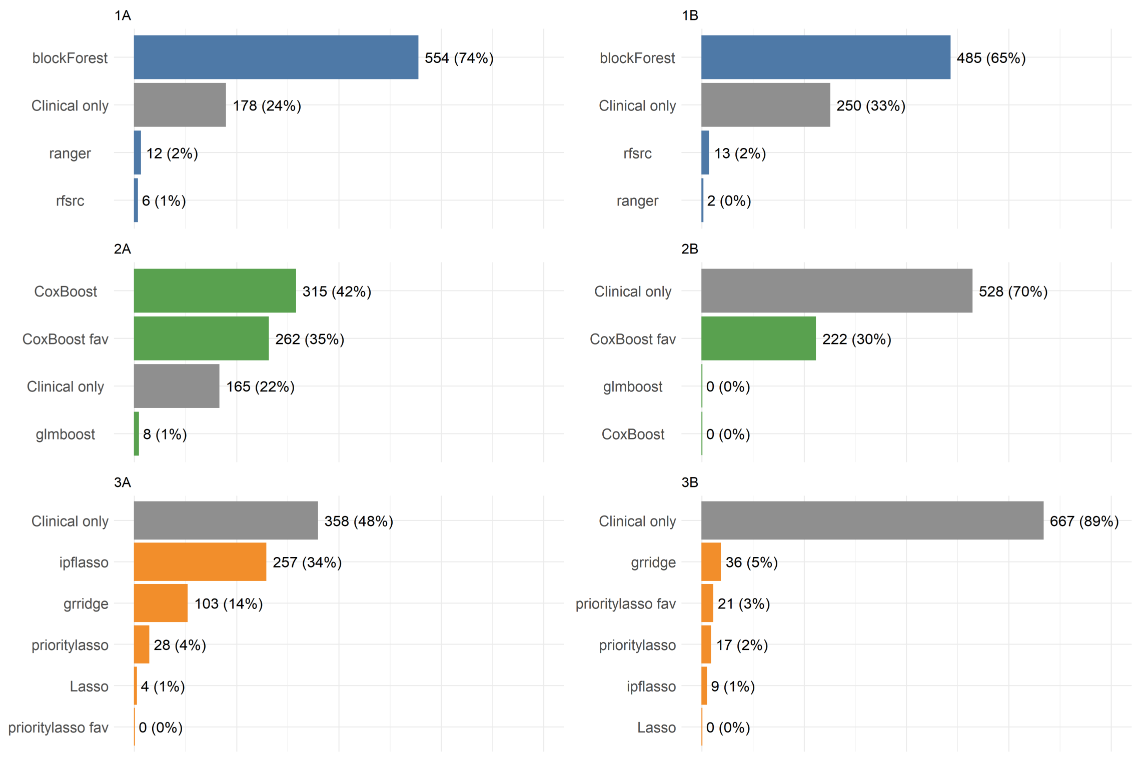

Figure 18: Best performing learners within the general modeling approaches for random samples - A: ibrier, B: cindex, 1: random forest, 2: boosting, 3: penalised regression. Colors indicate the general modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange).

Figure 19 is structured in the same way as figure 18, except that here the learners are grouped according to the different handlings of group structure. Again, large differences between ibrier and cindex can be observed. For the learners which do not take group structure into account, Cox model performs best most often for ibrier and cindex, but for ibrier there is at least one learner in 57% of the samples that performs better than the Cox model. For cindex this proportion is only 7% of the samples. Regarding the learners which favor clinical features (2) for ibrier CoxBoost favoring performs better than the Cox model in 65% of samples, but prioritylasso favoring does not perform better even once. For cindex, Cox model is again in first place for this group of learners with 70% of the samples. Looking at learners which take the group structure into account without favoring the clinical features (3), Cox model performs best for both measures, ibrier and cindex, in less than half of the samples. Here, blockForest is clearly in first place for both measures. It is also interesting that while ipflasso performs best for ibrier second most often, it is last for cindex within this group of learners.

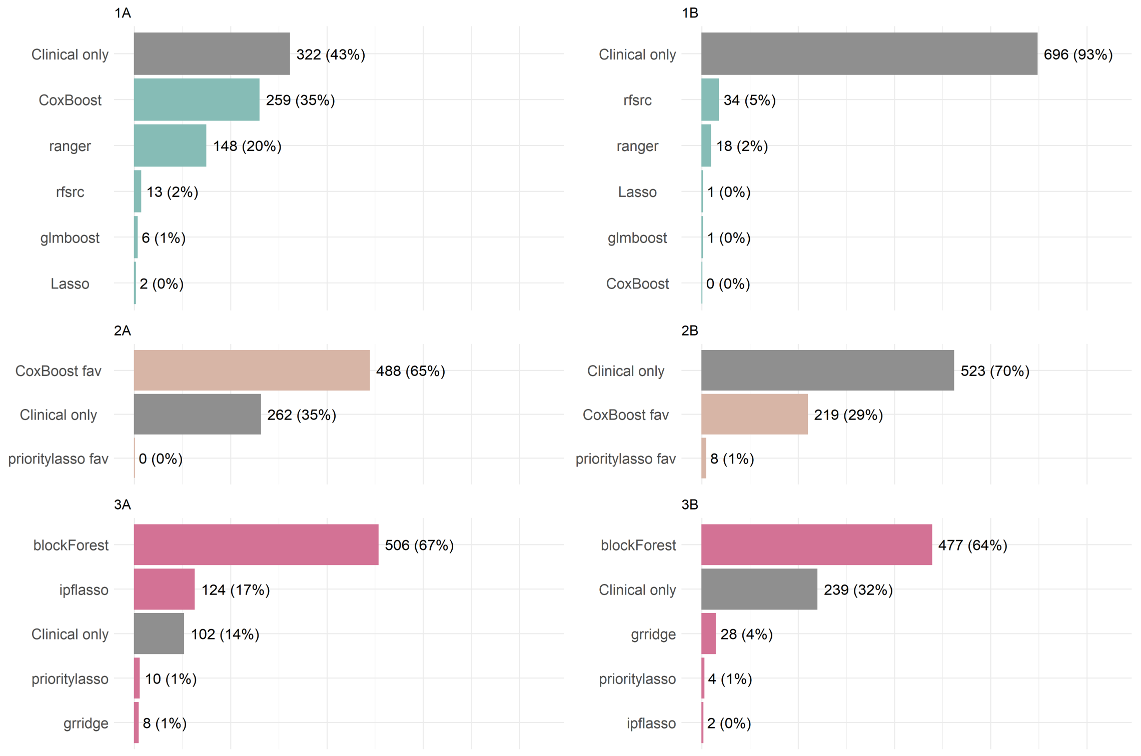

Figure 19: Best performing learners within different group structure handling for random samples - A: ibrier, B: cindex, 1: no group structure, 2: favoring clinical features, 3: group structure without favoring. Colors indicate the group structure handling: Clinical features only (gray), no group structure (blue), favoring clinical features (brown), group structure without favoring (pink).

The sampling was also applied in case zero values are treated as missing values. However, there are hardly any differences. Reasons for this are that all learners except CoxBoost have a negligible number of zero values and CoxBoost has a poor performance compared to the other learners even when replacing the zero values. It performs best only for one sample, when comparing only those learners which do not take group structure into account. Comparing all learners with each other, as in figure 12, or only the learners in the respective modeling approaches, as in figure 18, CoxBoost is still not the best learner for any sample.

Overall, sampling showed that there are factors that can influence the performance of certain learners and that the simple Cox model still plays an important role and often provides better predictions than learners using the multi-omics data. In total, blockForest often performs better than other learners. When comparing certain groups of learners with the simple cox model, the cox model performs better for cindex and ibrier, than learners which do not take the group structure into account for most samples. The same is true when comparing the simple cox model with the learners of penalised regression. For cindex only blockForest seems to perform better than the Cox model in general, while for ibrier in addition to blockForest also ipflasso and CoxBoost favoring often perform better.

6 Case Studies

In this section it will be shown how several degrees of freedom and different analysis strategies can be used to achieve desirable study results. Even small adjustments can cause misleading interpretations and results. First, the possible degrees of freedom are presented in general terms in a benchmark study and then the procedure and results of various exemplary case studies are described.

6.1 Possible degrees of freedom

As already described in Jelizarow et al. (2010), there are several optimizations a researcher can apply to better represent new methods. In most cases, these optimizations are not documented in detail and can lead to bias in the study results. In this work various possible degrees of freedom were presented. As described in section 3, there are several possibilities to replace missing values. If a learner with few missing values is to be presented well, the threshold method could be used and the threshold could be set very low, so that the learner with few missing values is presented well and the learners with more missing values are presented worse. If a learner with a high percentage of missing values is to be presented, but otherwise performs well, the mean value method could be used to replace the missing values. The data independent method could be used for example if the learner to be presented has no missing values but the learners to be compared have some missing values. Another optimization possibility is the selection of datasets. In Jelizarow et al. (2010) this is called "optimization of the dataset". As described above, the performance of a learner often varies for different datasets. If one selects the datasets for which the learner to be presented performs better without documenting this, the learner is presented better than it actually is. A further optimization possibility is as defined in Jelizarow et al. (2010) "optimization of the competing methods". This refers to the learners which are compared with the learner to be presented. If learners that perform better are not presented for any reason, the reader gets the impression that the learner to be presented is the best possible learner for the described setting. Furthermore, it was shown in this thesis that many assessments depend on the respective evaluation measure. Often different results have been obtained for ibrier and cindex. Examples are CoxBoost and ipflasso. If one is interested in presenting these learners well, it would be useful to evaluate them using the ibrier. It also depends on how the results are evaluated. As shown in Tables 7 and 8, values such as mean, median, number of datasets as best performing learner or in the 0.05 environment of the best performing learner and average rank often lead to different results.

Table 25 in the Appendix shows for how many datasets a learner was either the best performing learner or in the 0.05 environment of the best performing learner for each imputation method and evaluation measure. This table is intended to serve as a guideline for which study which imputation method and which evaluation measure is most appropriate.

6.2 Study 1: Lasso vs. random forest

In the previous analysis it was shown that Lasso performs best for cindex for not a single dataset. For ibrier, Lasso performs better than any other learner for one dataset, but has the worst mean, median, and average rank. And although Lasso has a worse mean ibrier than all random forest learners, the following will show how, through certain adjustments to the analysis and withholding of further information, Lasso performs better than all random forest learners. An appropriate hypothesis would be:

"Predictions on multi-omics data estimated with standard Lasso lead to better results than random forest methods."

First, it can be seen from table 25 that Lasso is most often best performing learner or in the 0.05 environment of the best performing learner for the ibrier and mean value method. This seems to make sense since Lasso has many missing values and is therefore often better rated by the mean value method. That means in the following all missing values are replaced by the mean value method. Since the random forest learners have no missing values, this imputation method is only useful for Lasso. If Lasso is only compared with the random forest learners and the mean value imputation method is applied, Lasso has the best performance for ibrier no longer for only one dataset, but for six. These are: COAD, HNSC, LGG, LUAD, LUSC and SARC. However, since there is no recognizable pattern for these datasets, for example, that these datasets all have a large number of observations, it cannot be argued that Lasso performs particularly well once the datasets have a certain property. However, if one looks at the cancer types more closely, LUAD and LUSC, i.e. Lung Adenocarcinoma and Lung Squamous Cell Carcinoma, are the only cancer types in the benchmark study that affect the respiratory system. LGG, the low-grade glioma is the only cancer in the study that affects the central nervous system. This means that Lasso could be said to be more predictive of cancer types affecting the respiratory and central nervous systems than any random forest learner.

If one argues like this and filters the datasets according to these cancer types, one gets that Lasso has by far the best mean and median for ibrier, has rank one for each dataset and no random forest learner was in the 0.05 environment of Lasso, so that Lasso performs better for each dataset by far. This result is shown in table 18.

	mean	median	rank	best/ 0.05 env
Lasso	0.1844	0.1893	1.00	3/3
blockForest	0.2048	0.2106	2.67	0/0
ranger	0.2098	0.2197	2.67	0/0
rfsrc	0.2124	0.2189	3.67	0/0

Table 18: Results of study 1. Column 2 shows the ibrier averaged over all CV iterations and datasets, column 3 shows the median over all CV iterations and datasets, column 4 shows the average rank over all datasets and the last column shows the number of datasets as best performing learner or in the 0.05 environment of the best performing learner.

This shows that if a hypothesis is made a priori and then certain datasets that confirm the hypothesis are fished posteriori, the results are biased. If all 18 datasets are considered and the imputation method with a threshold of 20% is applied, a contradictory hypothesis could be made as shown in table 7. From this table it can be concluded that blockForest in general performs better than all learners of penalised regression. BlockForest has a better mean and median for cindex than all penalised learner, performs better for most of the datasets and has a better average rank.

6.3 Study 2: Considering group structure vs. simple Cox model

As shown in table 2, Clinical only has a higher mean ibrier than blockForest, but a lower mean ibrier than all other learners which take the group structure into account. Nevertheless, the hypothesis:

"Taking group structure into account without favoring clinical features always outperforms the simple Cox model"

can be formulated and proven by the application of data dredging and the use of an appropriate imputation method. As except for blockForest, all learners considering the group structure have missing values, in the following the mean value method is used to replace the missing values. In addition, ibrier is used for the performance evaluation, since the Cox model performs slightly worse for ibrier and more learners which consider group structure perform better for ibrier. When looking more closely at the datasets for which the learners consider group structure perform better than the Cox model, they often have small values for effective cases n_{eff}. So, if all missing values are replaced by the mean value method, the results are evaluated by the ibrier and it is argued that the focus is mainly on cancer types that have a small number of effective cases ($n_{eff} < 40$), then the results are as in table 19. The datasets which remain in the analysis for this case study are COAD, ESCA, KIRP, LAML, LIHC, SARC and UCEC.

A

	mean	rank
blockForest	0.152	1.57
ipflasso	0.159	2.29
grridge	0.168	3.14
prioritylasso	0.169	3.86
Clinical only	0.170	4.14

B

	mean	median	rank	best learner
gs without fav	0.1570	0.1604	2.71	7
Clinical only	0.1643	0.1691	4.14	0

Table 19: Results of study 2 for ibrier - A: Comparison on learner level B: Aggregated comparison

Table A shows the results at learner level. Each learner that takes into account the group structure has a better mean ibrier and average rank than the simple Cox model. One could argue that no matter which modeling approach a learner belongs to, as soon as a learner takes the group structure into account it outperforms the simple Cox model especially for a small number of effective cases. For the aggregated table (B), it can also be seen that even if it is aggregated over all datasets, CV iterations and the learners which take into account the group structure, the structured learners have a better mean ibrier than the Cox model. Furthermore, the median has a smaller value and for all the seven considered datasets, the best performing learner is one that takes the group structure into account.

6.4 Study 3: CoxBoost within Boosting approaches

The goal of this case study was to show that if one treats the zero values of CoxBoost as missing values and replaces them with the weighted method, and filters the datasets accordingly, then it can be shown that CoxBoost performs better than the other learners within the boosting modeling approaches. The weighted method would be the most appropriate, because this imputation method guarantees that CoxBoost is never assigned a value below 0.5 (as it would be possible with the mean value method) and at the same time CoxBoost has the highest mean cindex with the weighted method as described in section 3.3.3. But even if one proceeds in this way, one gets the results as shown in table 20. CoxBoost is still not better than the other two Boosting learners for any dataset. This is obviously too weak to support the hypothesis that CoxBoost performs better.

	CoxBoost	glmboost	CoxBoost fav
BLCA	0.618	0.612	**0.640**
BRCA	0.510	0.495	**0.643**
COAD	0.471	0.462	**0.553**
ESCA	0.483	0.500	**0.558**
HNSC	0.565	**0.579**	0.574
KIRC	0.714	0.732	**0.744**
KIRP	0.486	0.523	**0.561**
LAML	0.520	0.523	**0.607**
LGG	0.737	**0.749**	0.712
LIHC	0.581	0.581	**0.602**
LUAD	0.562	0.577	**0.663**
LUSC	0.478	0.463	**0.534**
OV	0.472	0.443	**0.585**
PAAD	0.608	0.610	**0.678**
SARC	0.625	0.638	**0.665**
SKCM	0.495	0.463	**0.590**
STAD	0.531	0.538	**0.569**
UCEC	0.532	0.504	**0.654**

Table 20: Results of study 3 for cindex. The mean cindex is obtained by averaging over all datasets and CV iterations. Bold values indicate greater values for the given dataset.

Considering the ibrier, CoxBoost performs better than the other boosting learners more often for datasets with a number of observations below 200. If therefore the datasets are filtered accordingly, that is, the datasets COAD, ESCA, KIRP, LAML, LGG, LIHC, PAAD and SARC remain in the analysis, with the argument that one wants to find a learner that performs well for small values of N, one gets the results as shown in table 21. CoxBoost has the best mean ibrier, best average rank, is the best performing learner for four of the eight datasets considered and for all eight datasets at least in the 0.05 environment of best performing learner.

	mean	rank	best/ 0.05 env
CoxBoost	0.1698	1.5	4/ 8
CoxBoost fav	0.1752	2.0	3/ 3
glmboost	0.1802	2.5	1/ 3

Table 21: Results of study 3 for ibrier. The mean ibrier is obtained by averaging over all datasets and CV iterations. The column "best / 0.05 env" shows the number of datasets for which the learner was the best performing learner out of the number of datasets for which the learner was in the 0.05 environment of the best performing learner.

For this comparison, the missing values of glmboost were also replaced by the weighted imputation method since it does not affect the performance of CoxBoost. CoxBoost itself has no missing values for ibrier and glmboost for the dataset SARC for which it performs best would also perform best for each possible imputation method, since the proportion of missing values is at only 4%. The hypothesis:

"CoxBoost without favoring clinical features is the best possible learner within boosting modeling approaches"

can be confirmed by using the ibrier. This case study has again shown how different the results can be for different evaluation measures. Thus, if the researcher does not mention that the same analyses were also carried out for the cindex but only shows the results of the ibrier, since these support the hypothesis, and if the researcher does not document that the datasets were filtered posteriori, the reader gets a wrong impression.

6.5 Study 4: Cox model as best possible predictor

As can be seen in table 9 and 10, the simple Cox model has the best mean cindex and ibrier for datasets with many observations. Also, table 25 shows that the Cox model performs well for ibrier and cindex. Therefore, for the following case study, both evaluation measures are used to strengthen the hypothesis:

"The simple Cox model is still the best possible predictor for survival time."

Since the Cox model does not have any missing values, all missing values of the other learners are replaced by the data independent method. Looking at the datasets for which the Cox model performs worse than the 75% quantile, many of these datasets can be excluded by filtering the datasets for $N > 200$. This means that the datasets BLCA, BRCA, HNSC, KIRC, LUAD, LUSC, OV, SKCM, STAD and UCEC remain in the analysis. So, if all missing values are replaced by the data independent method and the datasets are filtered by the condition $N > 200$, then one gets the results as shown in table 22. The left side shows the results for cindex, the right side for ibrier. The simple Cox model has the best mean and median for cindex and ibrier, the best average rank and is for most datasets in the 0.05 environment of the best performing learner or is itself the best performing learner for both measures.

	cindex				ibrier			
	mean	median	rank	0.05 env	mean	median	rank	0.05 env
Clinical only	0.627	0.635	2.70	10	0.174	0.175	3.4	10
CoxBoost fav	0.618	0.624	3.10	10	0.175	0.177	3.6	7
blockForest	0.606	0.611	4.00	6	0.182	0.190	6.4	5
prioritylasso fav	0.605	0.612	5.30	6	0.181	0.180	7.1	5
ipflasso	0.600	0.614	5.50	5	0.175	0.177	3.8	8
prioritylasso	0.598	0.605	5.60	4	0.182	0.182	6.7	5
grridge	0.576	0.576	6.90	4	0.186	0.192	6.4	4
ranger	0.544	0.540	9.10	1	0.191	0.196	9.4	1
rfsrc	0.538	0.533	9.60	0	0.194	0.196	10.9	1
Lasso	0.536	0.500	8.65	3	0.207	0.213	10.5	1
glmboost	0.525	0.527	8.60	3	0.195	0.197	9.0	1
Kaplan-Meier	0.500	0.500	11.05	0	0.186	0.193	7.7	3
CoxBoost	0.404	0.503	10.90	1	0.182	0.185	6.1	4

Table 22: Results of study 4 - cindex (left), ibrier (right). The column "mean" shows the mean cindex resp. ibrier obtained by averaging over all datasets and CV Iterations. The column "median" shows the median for cindex and ibrier obtained over all datasets and CV iterations. The column "rank" shows the average rank and "0.05 env" shows the number of datasets, a learner was the best performing learner or in the 0.05 environment of the best performing learner.

If the results of the benchmark study are presented as in table 22, clearly Cox model would still be the best possible prediction method. One could conclude that the use of multi-omics data brings no added predictive value, no matter how the group structure information is handled, and that the simple Cox model outperforms any other learner, regardless of whether it is random forest, statistical boosting or penalised regression.

6.6 Study 5: Ipflasso as best possible predictor

For this case study we use the same hypothesis as in section 6.5, with the difference that ipflasso is now presented as best possible predictor. Therefore, the hypothesis is:

"Integrative Lasso with Penalty Factors is the best possible predictor for survival time."

The datasets are also filtered according to the condition $N > 200$, which means that the number of observations of a dataset should be greater than 200. The only difference to Study 4 is that the missing values are replaced by the weighted method, and we evaluate only by the measure ibrier. As it can be seen in table 22 the simple Cox model performs best for ibrier when replacing the missing values by the data independent method. The same result would be obtained by replacing the missing values by the threshold method at 20%, but then the difference in mean values between ipflasso and the Cox model would be slightly smaller. If the missing values are replaced by the weighted imputation method, then ipflasso performs best for ibrier, see table 23.

	mean	rank	best learner
ipflasso	0.1737	3.3	3
Clinical only	0.1739	3.4	2
CoxBoost favoring	0.1751	3.7	2
prioritylasso favoring	0.1793	7.4	0
prioritylasso	0.1797	6.9	0
CoxBoost	0.1818	6.2	0
blockForest	0.1820	6.4	1
grridge	0.1853	6.7	1
Kaplan-Meier	0.1861	7.9	0
glmboost	0.1875	8.6	1
ranger	0.1911	9.7	0
rfsrc	0.1937	11.2	0
Lasso	0.1980	9.6	0

Table 23: Results of study 5 - ibrier. The column "mean" shows the mean ibrier obtained by averaging over all datasets and CV Iterations. The column "rank" shows the average rank and "best learner" shows the number of datasets, for which a learner was the best performing learner.

For this table, the median was deliberately omitted, since for this measure, the Cox model would have the better value. If the missing values would be replaced by the mean value method, ipflasso would have a mean ibrier of 0.17354 and therefore the difference between Cox model and ipflasso would be a bit larger. But all in all, if one wants to test two exactly identical hypotheses, i.e. if a certain learner performs better than others and also filters the datasets according to the exact same criterion, one can get two different results by just applying two different imputation methods. In section 3 we have seen that the threshold method, with a value of 20%, very often rates the learners similar to the weighted method, but still these methods of replacing the missing values can give two different study results. However, it should be mentioned that for the cindex, the Cox model still performs best, regardless of the imputation method used. Here ipflasso still remains in fifth place.

This section has shown that there are many factors that have a great influence on the study results. Probably the biggest influence is the choice of the evaluation measure and the choice of datasets to be considered for the analysis. Case study 3 has shown how differently the results can be presented if further analyses performed in the study are concealed and only those results are shown that support the hypothesis. This makes it clear how important it is to present all the analyses carried out in a study and not only those that provide the desired result. It was also shown how great the influence is when a hypothesis is set up a priori and then the datasets are filtered posteriori, thus fishing for datasets that support the hypothesis. By this approach, each hypothesis could be easily verified. Another influence that should not be forgotten, especially in benchmark studies, is the "optimization of the competing methods" (Jelizarow et al. (2010)). If the number of competing learners had not been reduced that much in case study 1, Lasso could not have been presented as the best learner. Furthermore, case studies 4 and 5 showed that the choice of the imputation method also has a strong influence on the result. Even the differentiating methods with a threshold at 20% and the weighted method showed large differences. A further role plays the choice of the key figures to be presented. For example, certain key figures such as median or number of datasets as best performing learner were often deliberately omitted because they did not support the hypothesis. Basically, it has been shown that almost any hypothesis can be supported by using an appropriate imputation method, selecting certain competing methods, posterior filtering of datasets and the suitable choice of a performance measure.

7 Discussion and Conclusion

The basis of this work was the benchmark experiment of Herrmann et al. (2020). The goal was to compare the performance of different learners and to analyse whether the use of multi-omics data leads to better results for the prediction of survival time. Although it could be concluded that blockForest performs better on average than the simple Cox model and that taking into account the structure of the multi-omics data can lead to better performance, there were some datasets out of the 18 datasets considered for which this conclusion was not appropriate. The aim of this work was to present the multiplicity of analysis strategies and the problem of data dredging for benchmark studies, using the work of Herrmann et al. (2020). For this purpose the effect of different imputation methods of missing values was discussed, different possible analysis methods were presented to compare the learners and it was shown how small changes in the analysis strategy can lead to large differences in the study results. In total 13 different learners were compared on 18 different datasets.

When comparing the different imputation methods, it was shown that it would be advantageous to avoid naive methods. These naive methods often lead to the fact that a fair comparison of learners is not possible. However, even with differentiating methods, attention must be paid to how they are conducted (see case study 4 and 5). It remains to be discussed how zero values for the cindex should be handled. In total it was shown that the use of missing or zero values is a great degree of freedom for the researcher and that the different strategies strongly influence the results of the study. In any case, it is very important to document exactly how many missing or zero values a learner has and how exactly the imputation was proceeded, so that the reader has the possibility to get a clear picture.

For the different analysis strategies, it was difficult to make a consistent statement for the performance of a learner. For only 18 datasets (which is already a lot for a benchmark study) it is difficult to identify patterns in a learner's behavior. The sampling of datasets can help in any case. However, even if a learner performs well across all datasets, there are still some datasets for which the learner performs worse than others. See Table 14, which compared the four learners which often performed best, and showed that there are certain datasets where these learners perform poorly. This shows that the comparison of different learners strongly depends on the datasets used. Therefore, it is very important that for a fair comparison in a benchmark study as many datasets as possible are used and are not filtered posterior according to certain criteria. Case study 4 is an example of how the study results of Herrmann et al. (2020) would change if the datasets were filtered according to the number of observations and only datasets with more than 200 observations were considered.

Most noticeable were the large differences between ibrier and cindex. Especially the case studies have shown how big the effects are when only one performance measure is used. If a learner that is to be presented performs particularly well for a certain performance measure but poorly for another, the researcher is tempted to publish only the results of the better performance measure. However, this leads to biased study results. Furthermore, especially in a benchmark study, several performance measures should be used since they often measure different properties as described in section 2.3. For example, a learner may have a good calibration but a poor discrimination. Therefore, it is strongly recommended to evaluate the performance using more than one measure. In addition, the learners should be analysed using different key figures to identify other well performing learners. Table 7 and 8 have shown how different key figures favor different learners. For example, a learner may have a worse mean performance than other learners, but still have a better average rank.

As already described in Herrmann et al. (2020), it was shown that the simple Cox model is still very important for the prediction of survival data and that the clinical features have a high information content. It cannot be said that in general the consideration of multi-omics data leads to a better prediction. The gain from the use of multi-omics data depends on the learner using it. It has been shown that the Cox model outperforms many learners which use multi-omics data. The test for significance has also shown that none of the learners perform significantly better than the simple Cox model, but the Cox model performs significantly better than other learners which take into account the multi omics-data. As in Herrmann et al. (2020) it was still not possible to define one uniformly best learner. But in any case, it can be said that blockForest, the simple Cox model, CoxBoost favoring and ipflasso are suitable methods to provide a good prediction of survival time.

Overall, this work has shown how difficult it is to ensure a fair comparison and how quickly different study results can be obtained through small changes in the analyses. For each analysis, the author has a certain amount of degrees of freedom. If the procedures of these degrees of freedom are not properly documented, this can quickly lead to non-reproducible study results. For future studies it is therefore strongly recommended to always provide good and transparent documentation to ensure reproducibility.

Bibliography

Aarts, A. and Lin, S. C. (2015). Estimating the Reproducibility of Psychological Science, *Science* **349**(6251): 943 – 950.

Binder, H. (2013). Coxboost: Cox models by likelihood based boosting for a single survival endpoint or competing risks. R package version 1.4.
URL: *https://CRAN.R-project.org/package=CoxBoost*

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G. and Jones, Z. M. (2016). mlr: Machine Learning in R, *Journal of Machine Learning Research* **17**(170): 1–5.

Bischl, B., Mersmann, O., Trautmann, H. and Weihs, C. (2012). Resampling methods for Meta-Model Validation with Recommendations for Evolutionary Computation, *Evolutionary Computation* **20**(2): 249–275.

Bischl, B., Schiffner, J. and Weihs, C. (2013). Benchmarking local classification methods, *Computational Statistics* **28**(6): 2599–2619.

Blanca, M. J., Alarcón, R., Arnau, J., Bono, R. and Bendayan, R. (2018). Effect of variance ratio on ANOVA robustness: Might 1.5 be the limit?, *Behavior Research Methods* **50**(3): 937–962.

Boulesteix, A.-L., De Bin, R., Jiang, X. and Fuchs, M. (2017). IPFLASSO: Integrative L1-Penalized Regression with Penalty Factors for Prediction Based on Multi-Omics Data, *Computational and Mathematical Methods in Medicine* pp. 1–14.

Boulesteix, A.-L. and Fuchs, M. (2015). ipflasso: Integrative Lasso with Penalty Factors. R package version 1.1.
URL: *https://CRAN.R-project.org/package=ipflasso*

Breiman, L. (2001). Random Forests, *Machine Learning* **45**(1): 5–32.

Bühlmann, P., Hothorn, T. et al. (2007). Boosting Algorithms: Regularization, Prediction and Model Fitting, *Statistical Science* **22**(4): 477–505.

Cox, D. R. (1972). Regression Models and Life-Tables, *Journal of the Royal Statistical Society: Series B (Methodological)* **34**(2): 187–220.

De Bin, R. (2016). Boosting in Cox regression: a comparison between the likelihood-based and the model-based approaches with focus on the R-packages CoxBoost and mboost, *Computational Statistics* **31**(2): 513–531.

De Bin, R., Sauerbrei, W. and Boulesteix, A.-L. (2014). Investigating the prediction ability of survival models based on both clinical and omics data: two case studies, *Statistics in Medicine* **33**(30): 5310–5329.

Friedman, J., Hastie, T. and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent, *Journal of Statistical Software* **33**(1): 1–22.

Fu, G., Saunders, G. and Stevens, J. (2014). Holm multiple correction for large-scale gene-shape association mapping, *BMC Genetics*, Vol. 15, p. S5.

Gerds, T. A., Kattan, M. W., Schumacher, M. and Yu, C. (2013). Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring, *Statistics in Medicine* **32**(13): 2173–2184.

Goel, M. K., Khanna, P. and Kishore, J. (2010). Understanding survival analysis: Kaplan-Meier estimate, *International Journal of Ayurveda Research* **1**(4): 274–278.

Goodman, S. N., Fanelli, D. and Ioannidis, J. P. (2016). What does research reproducibility mean?, *Science Translational Medicine* **8**(341): 341ps12.

Graf, E., Schmoor, C., Sauerbrei, W. and Schumacher, M. (1999). Assessment and comparison of prognostic classification schemes for survival data, *Statistics in Medicine* **18**(17-18): 2529–2545.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*, Springer Series in Statistics, New York: Springer.

Herrmann, M., Probst, P., Hornung, R., Jurinovic, V. and Boulesteix, A.-L. (2020). Large-scale benchmark study of survival prediction methods using multi-omics data. Briefings in Bioinformatics (accepted), previous version available at: `https://arxiv.org/abs/2003.03621`.

Hoffmann, S., Schönbrodt, F., Elsas, R., Wilson, R., Strasser, U. and Boulesteix, A.-L. (2020). The multiplicity of analysis strategies jeopardizes replicability: lessons learned across disciplines. Preprint available at: `https://osf.io/preprints/metaarxiv/afb9p/`.

Hofner, B., Mayr, A., Robinzonov, N. and Schmid, M. (2014). Model-based boosting in r: a hands-on tutorial using the r package mboost, *Computational Statistics* **29**(1-2): 3–35.

Hornung, R. and Wright, M. N. (2019). Block Forests: random forests for blocks of clinical and omics covariate data, *BMC Bioinformatics* **20**(1): 358.

Hothorn, T., Bühlmann, P., Kneib, T., Schmid, M. and Hofner, B. (2018). mboost: Model-Based Boosting. R package version 2.9.0.
URL: *https://CRAN.R-project.org/package=mboost*

Huck, S. W. and McLean, R. A. (1975). Using a Repeated Measures ANOVA to Analyze the Data from a Pretest-Posttest Design: A Potentially Confusing Task., *Psychological Bulletin* **82**(4): 511–518.

Ishwaran, H. and Kogalur, U. (2018). Random forests for Survival, Regression and Classification (RF-SRC). R package version 2.6.1.
URL: *https://cran.r-project.org/package=randomForestSRC*

Jelizarow, M., Guillemot, V., Tenenhaus, A., Strimmer, K. and Boulesteix, A.-L. (2010). Over-optimism in bioinformatics: an illustration, *Bioinformatics* **26**(16): 1990–1998.

Kho, J. (2018). Why random forest is my favorite machine learning model. Accessed: 2019-08-03.
URL: *https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706*

Klau, S. and Hornung, R. (2017). Analyzing Multiple Omics Data with an Offset Approach. R package version 0.2.1.
URL: *https ://CRAN.R-project.org/package=prioritylasso*

Klau, S., Jurinovic, V., Hornung, R., Herold, T. and Boulesteix, A.-L. (2018). Priority-Lasso: a simple hierarchical approach to the prediction of clinical outcome using multi-omics data, *BMC Bioinformatics* **19**(1).

Klau, S., Schönbrodt, F., Patel, C., Ioannidis, J., Boulesteix, A.-L. and Hoffmann, S. (2020). Comparing the vibration of effects due to model, data pre-processing and sampling uncertainty on a large data set in personality psychology, *Technical Report 232*, Department of Statistics, University of Munich.

Probst, P., Wright, M. and Boulesteix, A.-L. (2019). Hyperparameters and Tuning Strategies for Random Forest, *WIRES Data Mining and Knowledge Discovery* **9**(3): e1301.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *https://www.R-project.org/*

Royston, P. and Altman, D. G. (2013). External validation of a Cox prognostic model: principles and methods, *BMC Medical Research Methodology* **13**(1): 33.

Schulze, G. (2017). *Clinical Outcome Prediction Based on Multi-Omics Data: Extension of IPF-LASSO*, Master's thesis, Ludwig-Maximilians-University, Department of Statistics, Munich.

Simmons, J. P., Nelson, L. D. and Simonsohn, U. (2011). False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant, *Psychological Science* **22**(11): 1359–1366.

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011). Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent, *Journal of Statistical Software* **39**(5): 1–13.

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2013). A sparse-group lasso, *Journal of Computational and Graphical Statistics* **22**(2): 231–245.

Therneau, T. M. (2015). A package for survival analysis in s. R package version 2.38.
URL: *https://CRAN.R-project.org/package=survival*

Tibshirani, R. (1997). The lasso method for variable selection in the Cox model, *Statistics in Medicine* **16**(4): 385–395.

Uno, H., Cai, T., Pencina, M. J., D'Agostino, R. B. and Wei, L. (2011). On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data, *Statistics in Medicine* **30**(10): 1105–1117.

Van De Wiel, M. A., Lien, T. G., Verlaat, W., van Wieringen, W. N. and Wilting, S. M. (2015). Better prediction by use of co-data: Adaptive group-regularized ridge regression, *Statistics in Medicine* **35**(3): 368–381.

Vanwinckelen, G. and Blockeel, H. (2012). On Estimating Model Accuracy with Repeated Cross-Validation, *BeneLearn 2012: Proceedings of the 21st Belgian-Dutch Conference on Machine Learning*, pp. 39–44.

Westphal, M. and Brannath, W. (2020). Evaluation of multiple prediction models: A novel view on model selection and performance assessment, *Statistical Methods in Medical Research* **29**(6): 1728–1745.

Wollschläger, D. (2016). *R kompakt - Der schnelle Einstieg in die Datenanalyse*, Springer Spektrum, Berlin, Heidelberg, Heidelberg.

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

A Figures

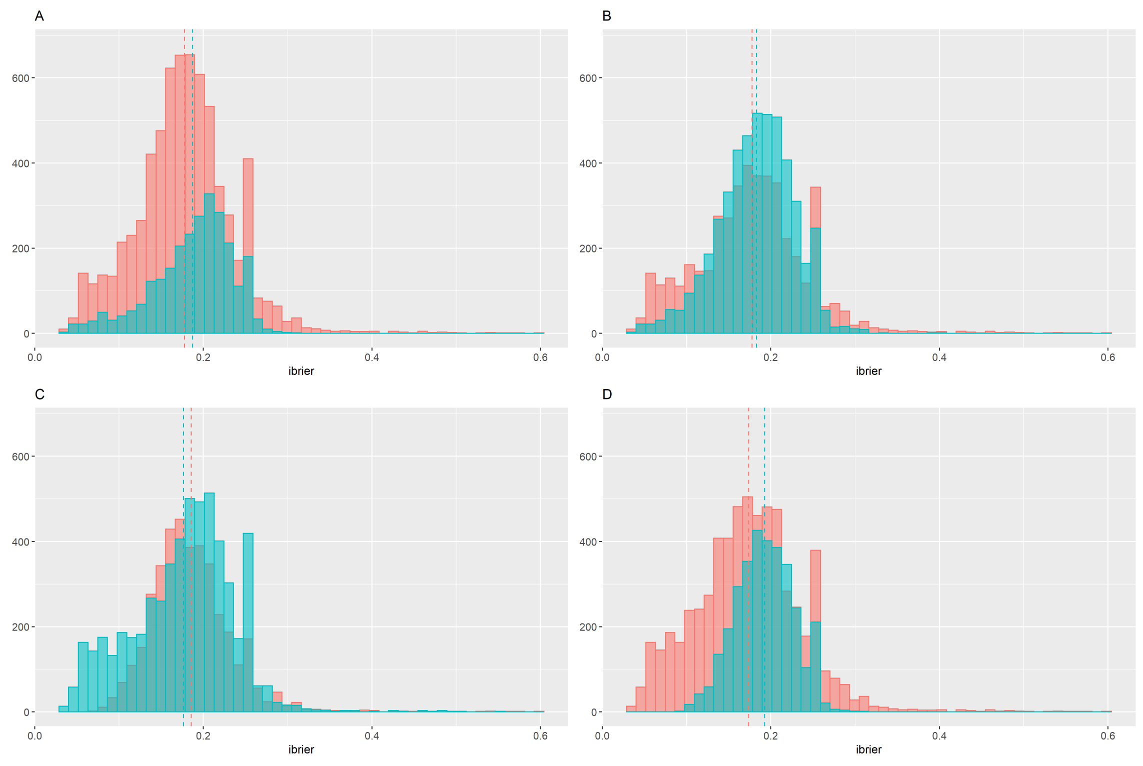

Figure 20: Density for different dataset groups - ibrier. A: number of observations (N), B: number of clinical features (clin), C: total number of features (p), D: number of effective cases i.e. events (n_{eff}). Colors indicate the group, smaller than mean (red), larger than mean (blue).

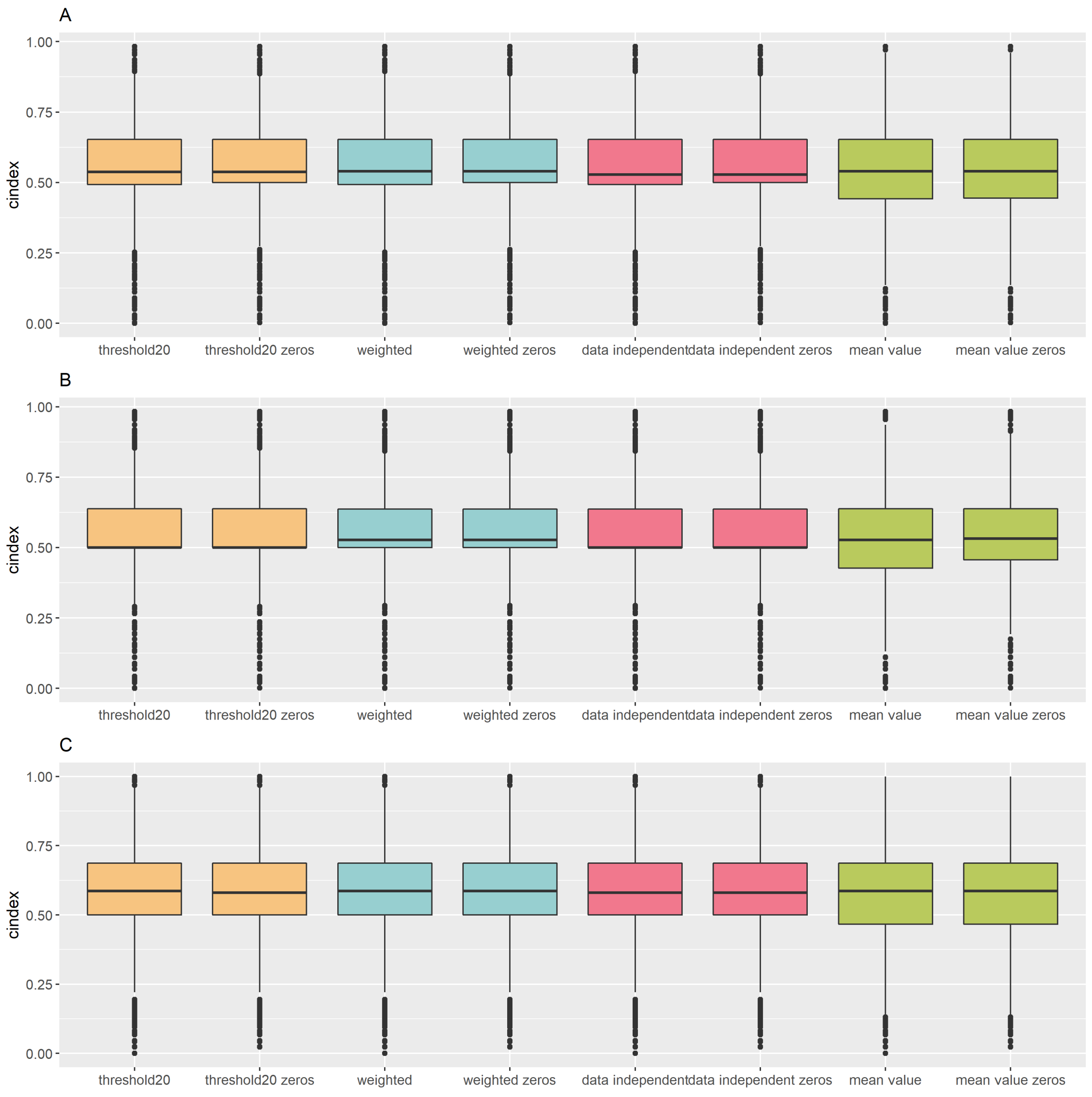

Figure 21: Comparison of imputation methods for treating zero values as missing values - A: glmboost, B: Lasso, C: ipflasso

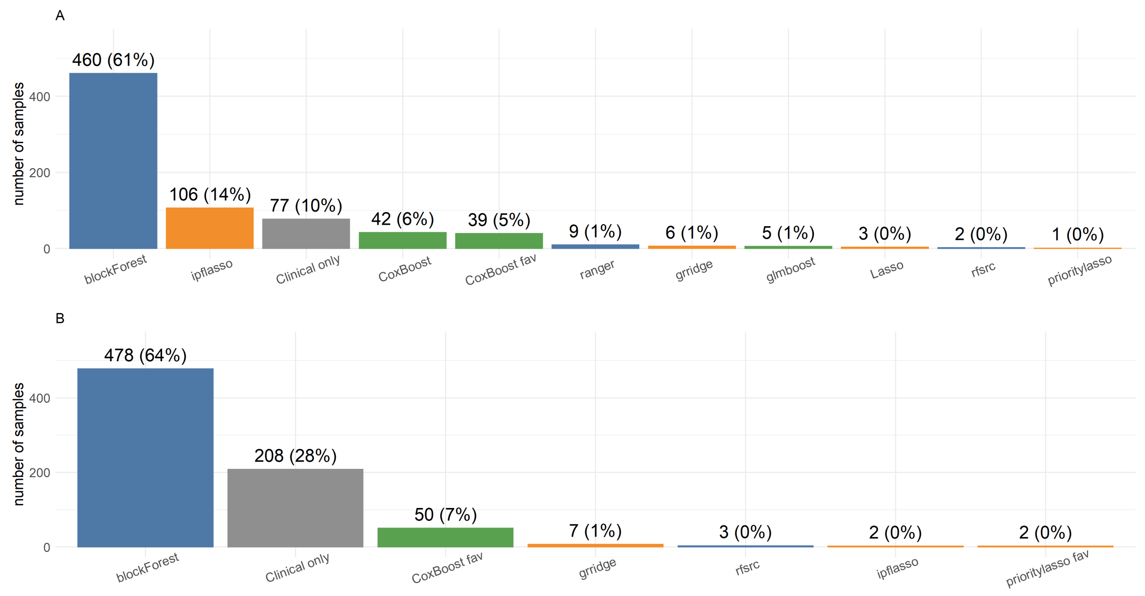

Figure 22: Best performing learners for sampling by higher number of clinical features - A: ibrier, B: cindex. Colors indicate the modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange).

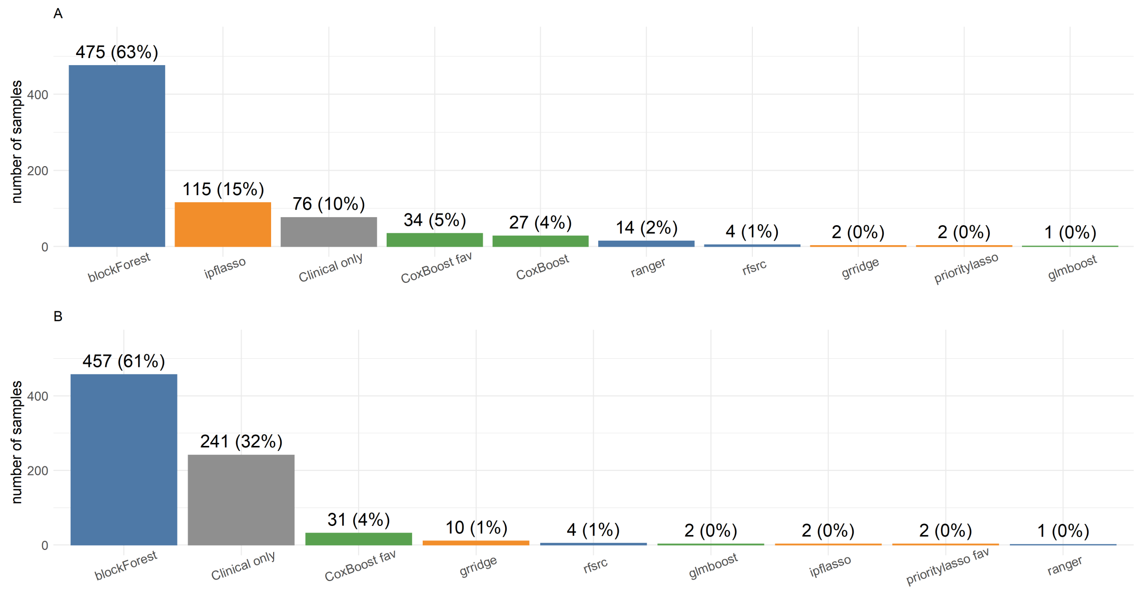

Figure 23: Best performing learners for sampling by higher number of total features - A: ibrier, B: cindex. Colors indicate the modeling approaches: reference methods (gray), random forest (blue), boosting (green), penalised regression (orange).

B Tables

Abbrevation	Cancer typ
BLCA	Bladder Urothelial
BRCA	Breast Invasive Carcinoma
COAD	Colon Adenocarcinoma
ESCA	Esophageal Carcinoma
HNSC	Head-Neck Squamous Cell Carcinoma
KIRC	Kidney Renal Clear Cell Carcinoma
KIRP	Cervical Kidney Renal Papillary Cell Carcinoma
LAML	Acute Myeloid Leukemia
LGG	Low-Grade Glioma
LIHC	Liver Hepatocellular Carcinoma
LUAD	Lung Adenocarcinoma
LUSC	Lung Squamous Cell Carcinoma
OV	Ovarian Cancer
PAAD	Pancreatic Adenocarcinoma
SARC	Sarcoma
SKCM	Skin Cutaneous Melanoma
STAD	Stomach Adenocarcinoma
UCEC	Uterine Corpus Endometrial Carcinoma

Table 24: Cancer types - abbreviations

	data independent		mean value		threshold 20%		weighted	
	cindex	ibrier	cindex	ibrier	cindex	ibrier	cindex	ibrier
Clinical only	4/ 14	3/ 12	3/ 14	3/ 12	4/ 14	3/ 12	4/ 14	3/ 12
blockForest	4/ 12	2/ 12	4/ 12	1/ 12	4/ 12	2/ 12	4/ 12	2/ 12
prioritylasso	2/ 4	0/ 5	3/ 5	0/ 6	2/ 4	0/ 5	2/ 5	0/ 5
prioritylasso fav	2/ 10	0/ 6	2/ 11	0/ 6	2/ 11	0/ 6	2/ 10	0/ 6
CoxBoost fav	2/ 15	2/ 9	2/ 14	2/ 9	2/ 15	2/ 9	2/ 15	2/ 9
glmboost	1/ 4	1/ 3	1/ 5	2/ 4	1/ 4	2/ 3	1/ 5	2/ 4
grridge	1/ 8	1/ 5	1/ 8	1/ 4	1/ 8	1/ 5	1/ 8	1/ 5
rfsrc	1/ 2	1/ 5	1/ 2	0/ 5	1/ 2	0/ 5	1/ 2	0/ 5
ranger	1/ 2	3/ 6	1/ 2	3/ 6	1/ 2	3/ 6	1/ 2	3/ 6
Kaplan-Meier	0/ 0	0/ 5	0/ 0	0/ 3	0/ 0	0/ 4	0/ 0	0/ 4
Lasso	0/ 4	1/ 3	0/ 5	2/ 6	0/ 4	1/ 3	0/ 5	1/ 3
CoxBoost	0/ 2	0/ 6	0/ 2	0/ 6	0/ 2	0/ 6	0/ 2	0/ 6
ipflasso	0/ 6	4/ 10	0/ 6	4/ 11	0/ 6	4/ 10	0/ 6	4/ 11

Table 25: Number of datasets as best performing learner or in the 0.05 environment per imputation method and evaluation measure

C Electronic appendix

The electronic appendix comprises an electronic version of this master thesis (*report.pdf*) and the three folders *Data*, *Plots* and *RCode*.
The folder *Data* contains all data needed to reproduce the results.
Run the file *tables_figures.R* in the folder *RCode*[1] to reproduce all tables and figures. All data required for reproducing is already contained in the *Data* folder. Make sure that all packages listed in *packages.R* are installed and loaded. The file *functions.R* contains all created functions to reproduce the code. Make sure that these functions are loaded. It is necessary to set the *wd* variable to the directory where the electronic appendix folder is located.
To reproduce the data itself, first run the file *NA_imputation.R*. This is followed by the file *analysis_strategies.R*, which contains the code for the sections *Multiplicity of analysis strategies* and *Sampling*. Finally let *case_studies.R* run. For each of these files, the variable *wd* must be set to the directory where the electronic appendix folder is located. In addition, all packages from *packages.R* must be installed and loaded and the functions of the file *functions.R* must be loaded.
The folder *Plots* already contains all figures presented in the master thesis. These will be overwritten when the file *tables_figures.R* is run.

[1]All analyses were performed with *R* version 4.0.1 (R Core Team (2020)).

D Declaration of Authorship

I declare that I completed this thesis on my own and that information which has been directly or indirectly taken from other sources has been cited as such. The material, either in full or in part, has not been previously presented to an examination committee.

Munich, July 15, 2020

. . . .
Chiara Wiedemann