Statistical Consulting

Modeling Survival Times of Prostate Cancer Patients Using Gene Expression Data

Department of Statistics
Ludwig-Maximilians-Universität München

Laetitia Frost & Jonas Schernich

Munich, February 14th, 2025

Project Partner: Dr. Markus Kreuz (Fraunhofer IZI)
Supervisors: Prof. David Rügamer (LMU) & Dr. Andreas Bender (LMU).
Contact for Correspondence: Lae.Frost@campus.lmu.de

Abstract

This report focuses on the modeling of the prognostic endpoint "biochemical recurrence of prostate cancer" using time-to-event data, clinical features and patients' genetic data. The primary objective is to train a variety of models on nine training cohorts and to evaluate their performance on two independent test cohorts. The choice of models is based on the inherent challenges associated with modeling genetic data, such as high dimensionality, multicollinearity and complex interactions. In addition, some of the models, as well as the implemented preprocessing, specifically address the challenge of different subsets of genes in the available cohorts. The evaluation of these models indicates a better performance for model classes that exhibit some degree of sparsity, either in a data-driven way via explicit regularization (e.g. penalized Cox Proportional-Hazards) or artificially through the inclusion of prior biological knowledge (Cox-PASNet). As the secondary objective, the performances of these models are compared to the ProstaTrend-ffpe Score, an already existing prognostic score, which is also based on genetic data. While no consistent picture emerges, the median performance of all models exceeds the performance of the ProstaTrend-ffpe Score in most cohorts. Furthermore, the value of incorporating genetic data into the modeling process is examined, resulting in better performance when genetic data is used in combination with clinical data. Lastly, feature importance is analyzed, revealing a certain degree of correlation between the genes selected by the models and those included in the ProstaTrend-ffpe Score is identified. Even though there is no exact overlap, this might nevertheless indicate a certain degree of association with respect to the same functional relations. Lastly, this report investigates further research directions, including sophisticated methods to handle block-wise missing data and methods to better integrate existing biological knowledge.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Existing Research

Prostate cancer (PCa) is considered to be the most prevalent solid cancer in men in western countries. The particular challenge in treating prostate cancer lies in the high clinical variability of the disease. The severity of the disease can vary greatly between patients, resulting in very different optimal treatment methods. For example, highly aggressive types of PCa might require radical prostatectomy (RP) or radiotherapy, whereas less aggressive types can simply be surveilled actively. Due to the potentially severe consequences that can result from an erroneous over- or under-treatment, a precise risk stratification with respect to a patient's disease outcome is vital to arrive at the individual, optimal treatment decision. There are multiple ways of conducting this risk assessment, but one approach is to leverage biological information in the form of genetic biomarkers obtained e.g. via RNA-sequencing of prostate tissue. These markers may assist in the prediction of the PCa outcome per patient. Thus, there is ongoing research with respect to the inclusion of these markers in diagnosing and predicting PCa. For example, Kreuz et al. (2020) developed the ProstaTrend Score, an RNA-based expression score for predicting the PCa diagnosis per patient. They use gene data obtained from fresh-frozen prostate tissue specimens of 233 patients from four cohorts, all of whom had a RP prior to entering the respective study. As primary prognostic endpoint, the authors use death of disease (DOD), i.e. the time from RP to death of PCa. In addition, they also validate the ProstaTrend Score against the prognostic endpoint biochemical recurrence (BCR), so the time from RP to the recurrence of the cancer, defined by the Prostate-Specific Antigen levels (PSA, see Section 2.3.1) exceeding a value of 0.2 ng/ml. To construct the ProstaTrend Score, univariate Cox Proportional-Hazards regressions are conducted per gene and cohort. These are then summarized within a fixed-effects meta-analysis. Those genes that show a false discovery rate ≤ 0.05 are then selected as relevant genes, which results in the selection of roughly 1400 genes. To actually construct the ProstaTrend Score for an individual, these genes are then combined using a weighted median approach. Thus, the ProstaTrend Score can be understood as a continuous risk score where values > 0 indicate an increased risk and values ≤ 0 a reduced risk for DOD/BCR.
As already stated above, the original ProstaTrend Score is constructed using gene expressions obtained from fresh-frozen prostate specimens. However, the form of tissue preservation can affect the quality of gene expression data. For example, whereas fresh-frozen specimens are usually considered to be of very high quality, formalin-fixed paraffin embedded (FFPE) preservation techniques are prone to induce degradation of the preserved specimens and thus affect the obtained gene data. To robustify the ProstaTrend Score against different types of preservation, Rade et al. (2024) constructed a revised ProstaTrend-ffpe Score. They use gene data from one cohort (176 patients) stemming from FFPE tumor biopsies. During construction they use BCR as the primary prognostic endpoint. Using regression analysis, genes are filtered from the ProstaTrend Score that show an association (linear regression p-value ≤ 0.1) to degradation depending on specimen age/preservation method. In addition, they only keep those genes showing a consistent prognostic effect between the training cohort and a validation cohort. This results in a subset of 204 of the original ProstaTrend Score genes. Combining these genes

via a weighted median approach results in the ProstaTrend-ffpe Score, which is again a continuous risk score with values above zero indicating increased risk and vice versa.

1.2 Goals of the Consulting Project

Based on this existing research, the goal of this statistical consulting project is to model BCR using patients' time-to-event, gene expression, and clinical data via various machine learning models. The model performances are to be compared with each other. In addition, the performance of the models is to be compared with the performance of the existing ProstaTrend-ffpe Scores. Extending these original goals, the performance of both the models and the ProstaTrend-ffpe Scores is evaluated with respect to different cohorts. Furthermore, the value of including Gene Data in general is examined. Lastly, the resulting feature importance of the applied models is examined. Based on this, selected genes are compared to the genes used in the ProstaTrend-ffpe Score.

1.3 Outline of the Report

The following report gives a detailed overview regarding both the procedure and the results of this consulting project. The overall process of the project is outlined in Figure 1.

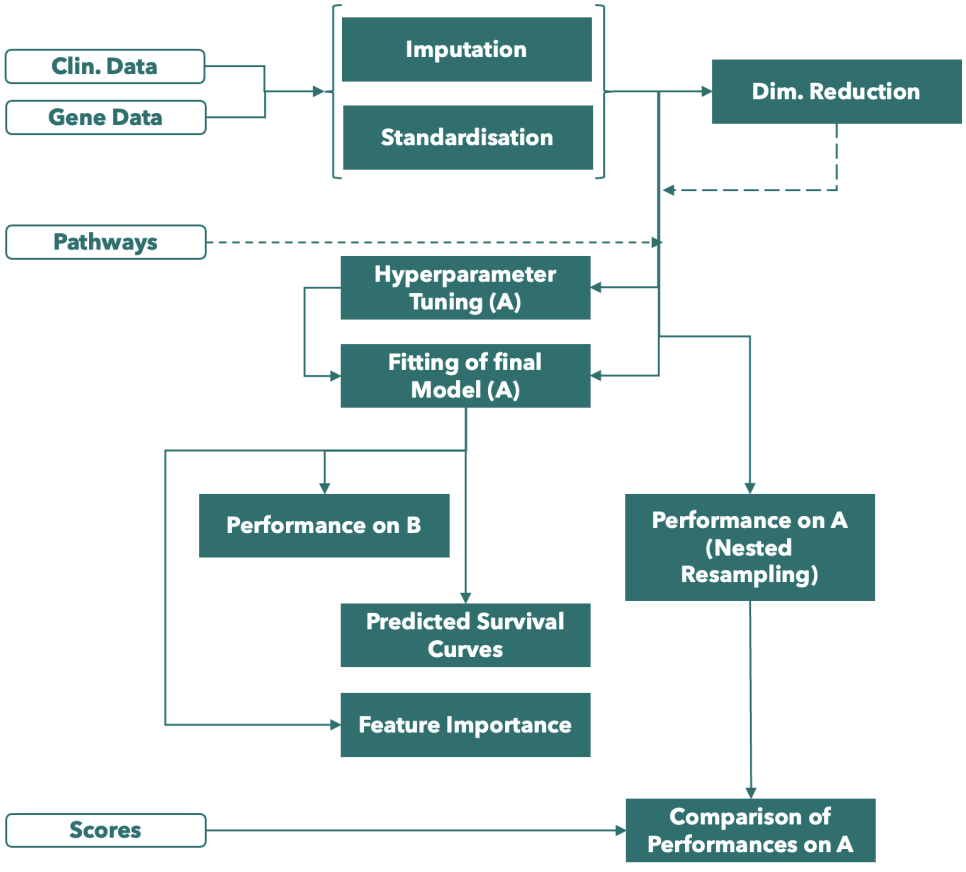

Figure 1: Process overview of the consulting project.

Following this outline, the second chapter initially provides an overview of the obtained data. Then, it focuses on the structure of the prognostic endpoint used, of the patients' clinical features and of the gene expression data. In addition, it elaborates on the challenges, especially those arising from the present gene expression data structure and addresses the measures taken to tackle these challenges. The third chapter explains the applied models, focusing on the most important structural characteristics of these models, as well as the reason for use. In addition, possible caveats of the methods are elaborated. Furthermore, the most important steps of the modeling process are explained. The fourth chapter presents the results in reference to the questions established in Section 1.2. The final chapter contains a summary of the results as well as a discussion of limitations of this project and extensions for future research.

2 Data

2.1 Data Overview

The project partner provided data from eleven distinct cohorts, with a total of 1587 patients. Since nine of them are publicly available, these cohorts were provided at the beginning of the consulting project. The remaining two cohorts were provided in the last quarter of the project due to the data security process. As a result, the data are divided into two groups. Group A contains the first nine cohorts with a total of 1091 patients and Group B contains Cohorts 10 and 11, totaling 496 patients. There is no overlap of patients between different cohorts. Due to the timeline, Group A is used for training the models while the cohorts in Group B are used as independent test cohorts for these models. Table 1 provides an overview of the details regarding each cohort.

Group	**Cohort**	**Name**	**Year**	**Location**	**Nmb. Patients**
A	Cohort 1	Atlanta	2014	USA	100
A	Cohort 2	Belfast	2018	UK	248
A	Cohort 3	CamCap	2016	UK	112
A	Cohort 4	CancerMap	2017	UK	133
A	Cohort 5	CPC	2017	Canada	73
A	Cohort 6	CPGEA	2020	China	120
A	Cohort 7	DKFZ	2018	Germany	82
A	Cohort 8	MSKCC	2010	USA	131
A	Cohort 9	Stockholm	2016	Sweden	92
B	Cohort 10	TCGA	2015	USA	332
B	Cohort 11	UKD2	2020	Germany	164

Table 1: Overview of cohorts.

The cohort data are provided as RDS files, which contain two separate datasets for each cohort, specifically an RNA gene expression dataset and a dataset containing both clinical and organizational information of each patient. In addition, metadata regarding the gene expression sets per cohort are provided. Since the metadata are not used in this project, they are not further discussed. Furthermore, another RDS file containing the ProstaTrend-ffpe Scores for the patients of all eleven cohorts is supplied, as well as two CSV files with Pathway Data, describing already known genetic pathways related to biological processes.

2.2 Target Variable

As already stated in Section 1.2 the prognostic endpoint used across cohorts is BCR, where the variable "Months to BCR" is used as time variable and "BCR Status" is used to define the recurrence of cancer, with a status of 1 indicating PCa recurrence. However, there is an exception to this as in Cohort 11 all BCR endpoints are missing. In consultation with the project partner, DOD ("Months to DOD" and "DOD Status") is thus used as prognostic endpoint for this cohort. Generally, the number of months until DOD values are higher than the number until BCR, which makes sense when considering the logical order of the two events.
Since for some patients the observation time ended before an event was observed, the present data is right censored. In addition, due to the study designs, it is assumed that the censoring is independent, meaning the probability of an observation being censored is independent of the survival time, given the covariates.
Figure 2 shows the Kaplan-Meier curves regarding BCR for Cohorts 1-10, and Figure 3 shows the analog regarding DOD for Cohort 11. It appears that the probability for staying BCR free, decreases differently across cohorts. For example, Cohort 6 (yellow) and Cohort 2 (blue) have higher BCR-free probabilities throughout the observed follow-up period as compared to other cohorts. Their probabilities stay above about 70% across the observed time. In contrast, Cohorts 1 (red) and 10 (green) show much lower BCR-free probabilities. In addition, Cohort 1 exhibits a relatively steep decline at the beginning of the follow-up period, suggesting early cancer recurrence for many patients.

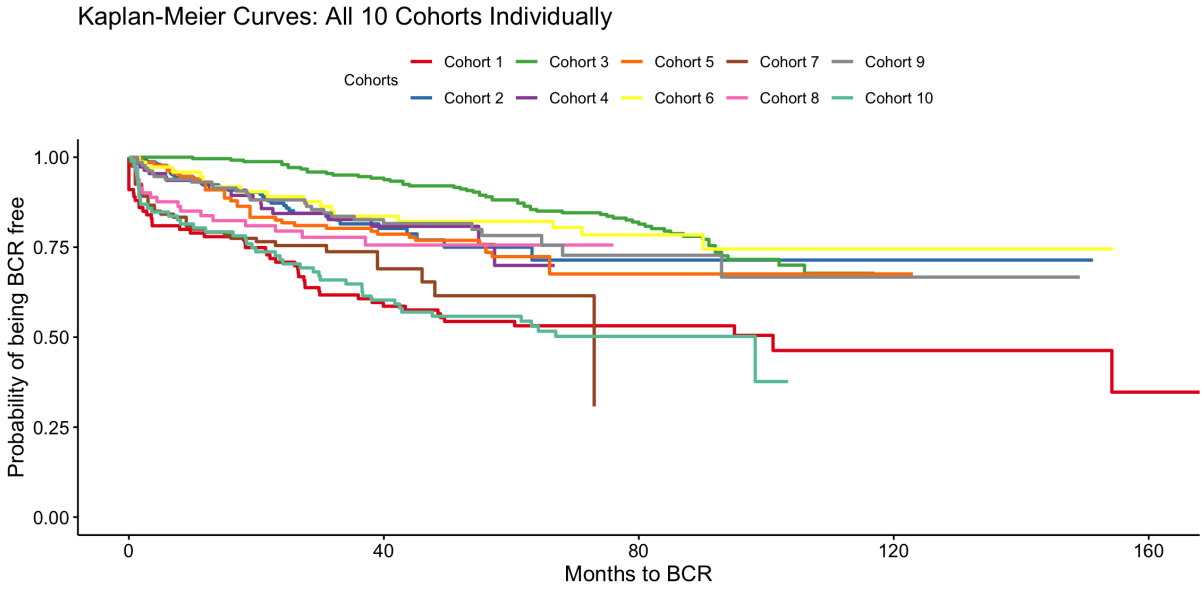

Figure 2: Observed Kaplan-Meier curves for Cohorts 1-10 (BCR).

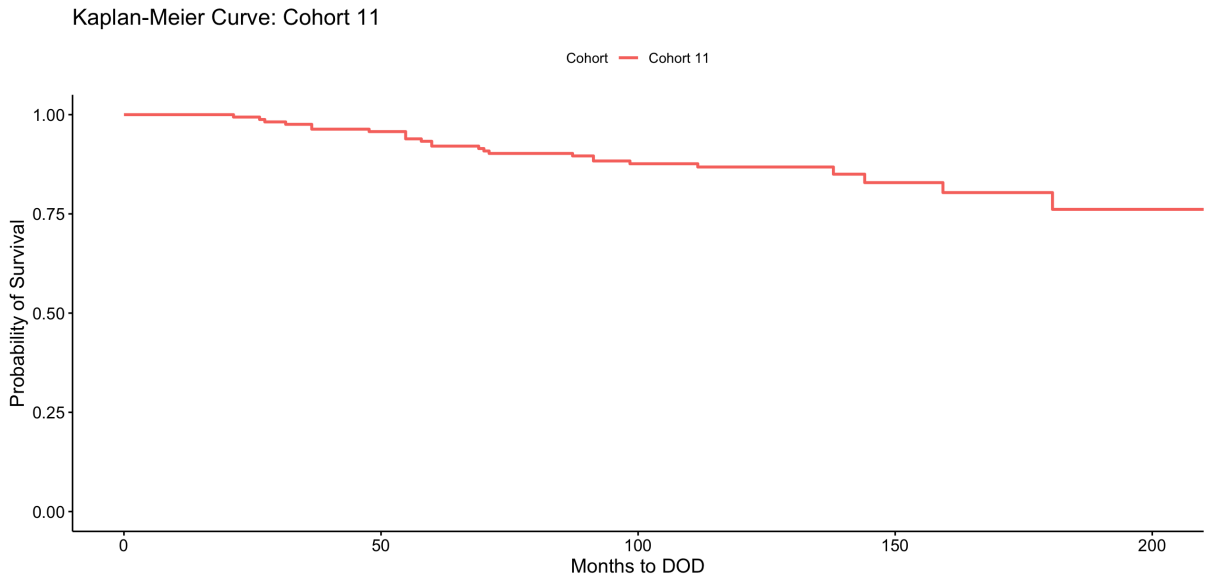

Figure 3: Observed Kaplan-Meier curves for Cohort 11 (DOD).

Regarding the differences between Group A and B, Figure 4 compares the Kaplan-Meier curve for the combined Cohorts 1-9 with Cohort 10. It becomes clear that cohorts from Group A show a more steady decline in probability for staying BCR free, as compared to Cohort 10. Here, the Kaplan-Meier curve plateaus at around 70 months, which indicates that no BCR is observed beyond that point for the remaining patients in this Cohort. Regarding the present censoring rates, the mean time until censoring is 53.62 months for the combined Cohorts 1-9, 24.96 for Cohort 10, and 122.8 for Cohort 11 (w.r.t. DOD). The mean time until observed BCR events is 26.12 for combined Cohorts 1-9, 17.71 for Cohort 10, and 59.9 for Cohort 11 (w.r.t. DOD) (Figure 3).

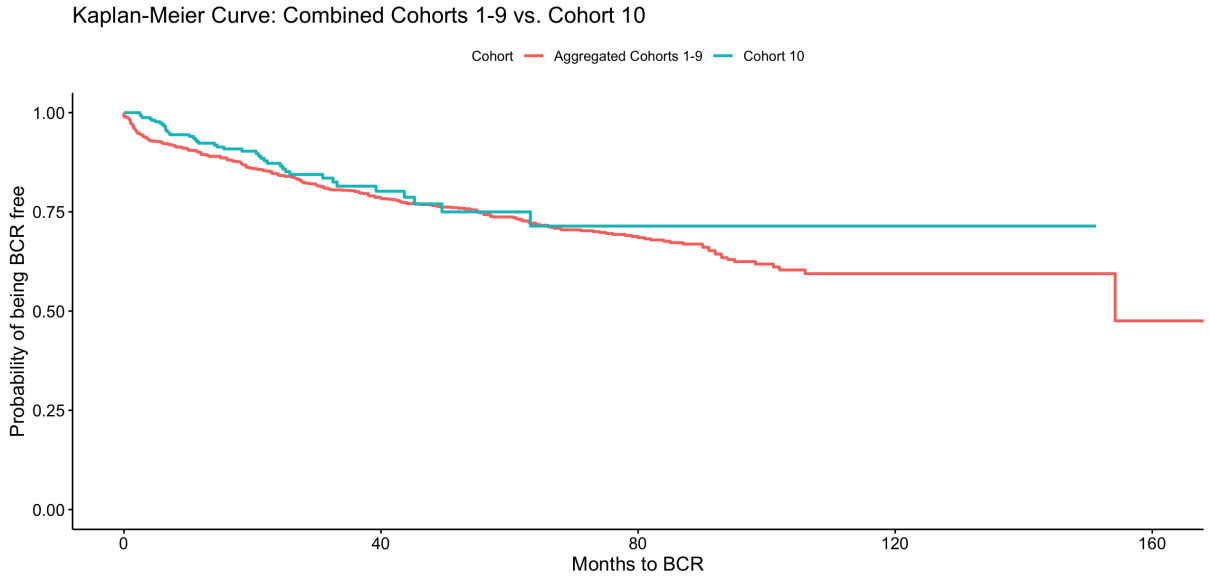

Figure 4: Observed Kaplan-Meier curves for combined Cohorts 1-9 vs. 10 (BCR).

Apart from the missing BCR data in Cohort 11, there are no missing values for the target variables in any other cohort, so no further imputation is necessary. However, to avoid any computational issues during modeling, the zero values of "Months to BCR" are replaced with a small offset of 0.0001.

2.3 Clinical Data

As already mentioned in Section 2.1, both clinical and organizational patient data were provided by the project partner. Table 2 gives an overview of all of these variables including the number of missing values per variable. However, based on input from the project partner and medical research, only four clinical variables are considered as features during modeling, and are thus discussed in the following subchapter. From this point forward, the term 'Clinical Data' refers to these four variables.

Variable	**Missing Count**	**Variable**	**Missing Count**
Sample ID	0	GSM Sample ID	202
SRR Sample ID	1487	Paper Sample ID	0
Sample Count	73	Age	225
Study	0	Platform	0
Tissue	0	Sample Class	0
Sample Type	0	Surgical Procedure	496
Clinical TNM Stage	496	Clinical T Stage	912
Clinical T Stage Group	914	Clinical N Stage	1357
Clinical M Stage	1357	Pathological TNM Stage	496
Pathological T Stage	256	Pathological T Stage Group	259
Pathological N Stage	691	Pathological M Stage	1281
Gleason Score	3	Gleason Score 1	274
Gleason Score 2	274	Preoperative PSA	37
Months to BCR	164	BCR Status	164
Months to Last Follow-up	1283	OS Status	1451
Months to Death (DOD)	1091	DOD Status	1091
Months to CEP	0	CEP Status	0

Table 2: Overview of clinical variables.

2.3.1 Clinical Features

Age of Diagnosis: Due to the supporting literature (Knipper et al., 2020) regarding the possible association between a patient's age and the corresponding PCa prognosis, the variable Age is considered in the modeling process. Figure 5 shows that Age seems evenly distributed across cohorts, with the exception of Cohort 7, whose patients are notably younger than those of the other cohorts. The figure also shows, that the variable Age is completely missing for Cohorts 4 and 9, which equals roughly 14% of patients (Table 2).

Gleason Score: The Gleason Score is a grading system to assess the aggressiveness of PCa, usually obtained via the examination of PCa tissue. Generally, medical literature relates high Gleason Scores to worse PCa outcomes (Egevad et al., 2024). It combines the evaluation of two patterns within the cancer tissue. For both patterns, the aggressiveness is rated in separate scores (Gleason Score 1 and 2), both ranging from one to five, where a score of one indicates low aggressiveness, while a score of five indicates high aggressiveness. The overall Gleason Score is the sum of both scores and thus ranges between two and ten. Owing to the relatively large amount of 274 missing values (Table 2) for Gleason Score 1 and 2 and to the informational overlap between these two variables and the overall Gleason Score, only the latter is considered as a feature during modeling. Figure 6 shows the distribution of Gleason Scores in Group A and B, indicating the most common Gleason Score to be seven, with very low or zero occurrence rates for the highest scores and values smaller than six.

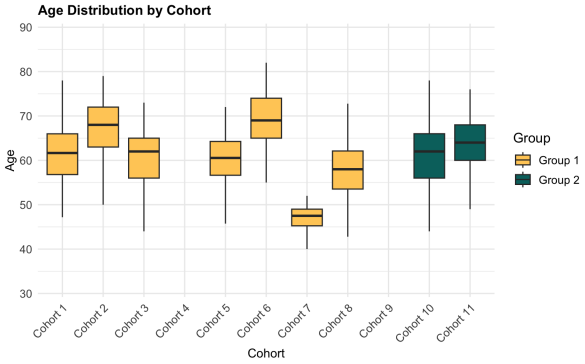

Figure 5: Distribution of Age at diagnosis.

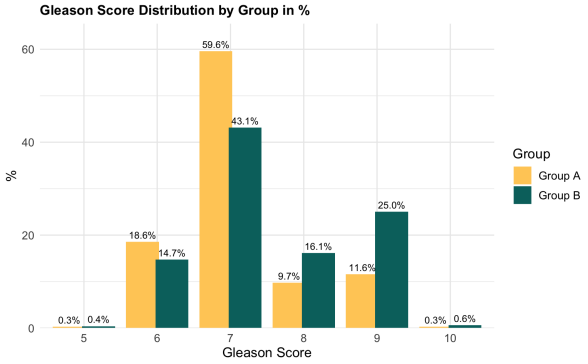

Figure 6: Distribution of Gleason Score.

Preoperative PSA: PSA is not only used to determine the BCR but is also a commonly used clinical blood marker for diagnosing PCa. Although PSA levels generally increase with age, unusually strong elevations of PSA can indicate the presence of PCa (Gretzer and Partin, 2002). Medical literature associates higher PSA levels with poorer PCa outcomes. This association is logical, given that PSA levels tend to increase with tumor growth. Consequently, Preoperative PSA, which refers to the patients' PSA levels before RP, is included as a feature in the modeling process. Figure 7 compares the Preoperative PSA distributions between the different cohorts. There does not appear to be major differences in the PSA levels for most of the patients across the cohorts. However, some cohorts show a few strong outliers, which is why Figure 7 is displayed using a log scale. The most extreme one belongs to Cohort 6, with a Preoperative PSA level of 1270.14 ng/ml. To provide the most comprehensive picture of the underlying data distribution, there is no specific treatment of these outliers.

Tissue Preservation Method: As the Tissue Preservation Method can affect the quality of the sample at the time of the investigation (Rade et al., 2024) the inclusion of this feature appears reasonable. Figure 8 shows that three preservation methods are used across the different cohorts. The most commonly used method is fresh-frozen, which is also the only method used for cohorts in Group B. For cohorts in Group A, Formalin-Fixed Paraffin-Embedded (FFPE) and Snap-Frozen methods are also used.

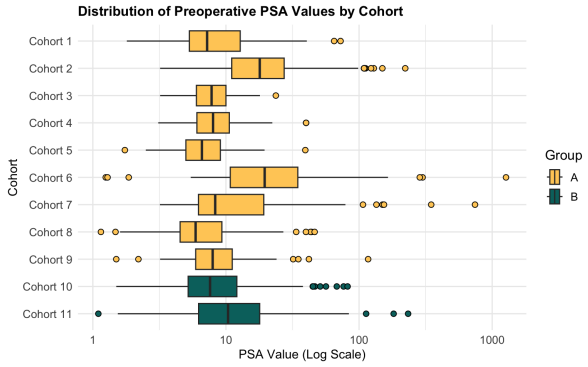

Figure 7: Distribution of Preoperative PSA values in ng/ml.

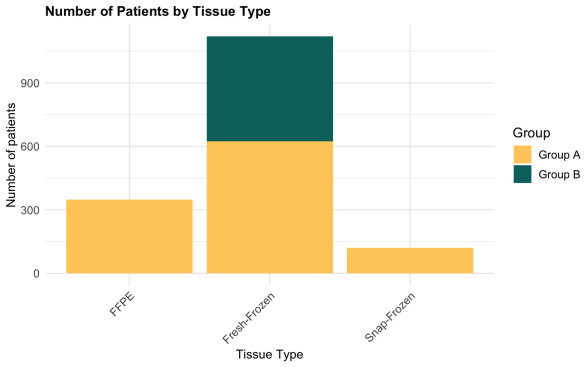

Figure 8: Distribution of Tissue Preservation Method.

2.3.2 Imputation of Clinical Features

As Table 2 shows, there are no missing values for the Tissue Preservation Method, and only a few missing values for both Gleason Score and Preoperative PSA across the cohorts. However, as already indicated, the Age variable is completely missing for Cohorts 4 and 9, resulting in a total of 225 missing values. To impute the missing data, advanced imputation methods such as Multiple Imputation by Chained Equations (MICE) were considered at the beginning of the project. MICE iteratively imputes missing values for each variable using predictive modeling based on the available data. Through this, MICE generates multiple plausible datasets to thus account for uncertainty in imputation (Austin et al., 2021). However, in the present data, there are no variables that can reasonably serve as predictors for missing data in variables like Age. Thus, in consultation with the project supervisors, the median value is used as the final imputation method. This approach is preferred over e.g. imputation via the mean to reduce the influence of significant outliers, which are for example present in the Preoperative PSA variable (see Section 2.3.1). Imputation is done by merging all cohorts from Group A and calculating the median values of Age, Gleason Score, and Preoperative PSA. The resulting medians are then used for all the missing values in the respective variable. In addition to this imputation, some further preprocessing steps are taken. Firstly, some data type corrections are performed, specifically on numeric variables that are originally stored as strings, as well as on missing data that are declared as strings instead of NA. Furthermore, all numeric variables are standardized via z-transformation.

2.4 Pathway Data

Biological pathways (Pathway Data) are known interactions and relations between different biological entities, which also include known connections and functionalities between different genes. Usually several genes load onto one specific pathway, whereas one gene can load onto multiple pathways. Properly modeled pathways can contribute to understanding, e.g. mechanisms of medical treatment or the progression of certain diseases (Hanspers et al., 2021, Otero-Carrasco et al., 2024). Thus, determining which pathways are connected to diseases and incorporating pathway data as prior knowledge can enhance

predictive modeling, which in turn can improve personalized strategies for diagnosis, treatment and prevention of disease (da Costa Avelar et al., 2023).
In the context of PCa, several pathways are known to be associated with either an increased or decreased risk (Shtivelman et al., 2014) of both DOD and BCR. This is why this prior biological knowledge is also considered during modeling in this consulting project, even though this inclusion was not initially planned. At the request of the project team, the project partner provided 143 pathways with roughly 8000 associated genes. 6000 of those are contained in the intersection of gene data available across cohorts and are thus used in the modeling approaches that use this Pathway Data (see Section 3.1.5).

2.5 Gene Data

2.5.1 General Structure and Resulting Challenges

Per patient, the gene data describe the RNA expressions of different genes sequenced across cohorts. Generally, a larger value indicates a stronger expression of the respective gene in the patient, which is usually associated with more pronounced effects related to that gene. In the provided data, 63008 distinct genes are present. However, these genes are not present across all cohorts with the number of available genes differing greatly across cohorts. As Figure 9 shows the cohort covering the largest number of genes is Cohort 10 with a total of 57178 genes, while the smallest one is Cohort 2, covering only 16810 genes. The intersection of genes that is available across all cohorts in Group A contains 13211 genes.
The different data availability across cohorts gives rise to the challenge of including as many of the available genes as possible during modeling. In addition, gene data itself comes with a unique set of challenges. For example, gene data are usually high-dimensional, which can not only lead to computational challenges, but can also increase the risk of overfitting (Alsalem et al., 2018). This is especially relevant in combination with low sample sizes, as in this project. In addition, gene data often exhibit complex (non-linear) interactions both in between genes and between genes and the respective target variable (Makrodimitris et al., 2023). Furthermore, gene data is prone to noise in the data, caused by technical variability or biological heterogeneity, which can then potentially mask true biological signals and relations (Alsalem et al., 2018). Moreover, gene data often exhibit a certain degree of multicollinearity (Christensen et al., 2022). To some degree, this is also relevant for the present gene data. For example, in the intersection of genes from Group A, 202 gene pairs had a correlation (Pearson) larger than 0.75. Lastly, in the context of gene data a certain degree of interpretability is desirable, and especially for decisions where the stakes are high, it has been argued that interpretable methods should be used wherever possible (Christensen et al., 2022).
The majority of these challenges (e.g., complex interactions, multicollinearity, noise, and interpretability) can be addressed via an adequate model choice. Additionally, the challenge of different gene availability across cohorts can also be addressed during preprocessing to either arrive at a subset or at a latent representation of the available gene data that reflects the given information as efficiently as possible. These measures are described in the following subchapter.

2.5.2 Data Imputation

One approach to tackle the different gene availability across cohorts is to simply use the intersection of genes that are available across cohorts. Again, due to the timeline and accessibility of the data, this intersection is obtained from all the genes contained in cohorts from Group A (56722 genes). As already mentioned, this intersection, from this point forward called "Intersection Genes", contains 13211 genes. In addition, to increase the amount of usable expression data during modeling, a set of common genes (Common Genes) is defined, which are genes for which expression data are available for at least 80% of patients from Group A (15495 genes). As already stated above, since Group A is used for training, both the Intersection Genes and the Common Genes are used during model training. Figure 10 provides an overview about the dimensions of the gene sets.

The missing expression values in Common Genes are imputed via the k-Nearest Neighbors (k-NN) algorithm using all the data in Group A. Figure 11 shows the number of missing genes imputed per Cohorts 1-9. In general, k-NN is a supervised machine learning algorithm that predicts the class of a data point based on the majority class among its k closest neighbors, utilizing a distance metric to determine proximity (Taunk et al., 2019). Since in this project k-NN is not applied to a categorical target, the algorithm uses the average of gene expression values instead of using the majority vote. The nearest neighbors are calculated based on the Euclidean distance, only using non-missing coordinates. Specifically, this approach is implemented via the R library impute (Hastie et al., 2024), in which the k-NN option is specifically designed for the imputation of gene expression data. Generally, concerns regarding the use of k-NN in high dimensional data are reasonable, as the sparsity in high-dimensional spaces can lead to issues in accurately identifying true nearest neighbors. In addition, the requirement of $\mathcal{O}(p \log(p))$ (Hastie et al., 2024) operations per gene for imputation can cause a high computational load. However, those concerns are addressed by Troyanskaya et al. (2001) through clustering of genes. Specifically, the impute package, which is based on Troyanskaya et al. (2001), uses two-mean clustering for creating blocks and then performs a block-wise k-NN. A cutoff of 20% missing expressions for a gene is chosen for the Common Genes dataset. This is also based on Troyanskaya et al. (2001), as they argue for a reliable imputation if the percentage of missing data does not exceed this level. While there are several hyperparameters

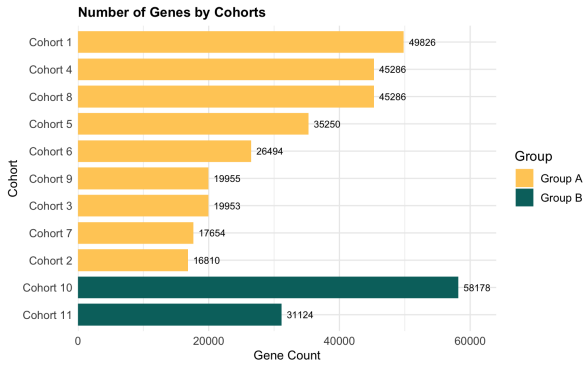

Figure 9: Number of genes available across all cohorts.

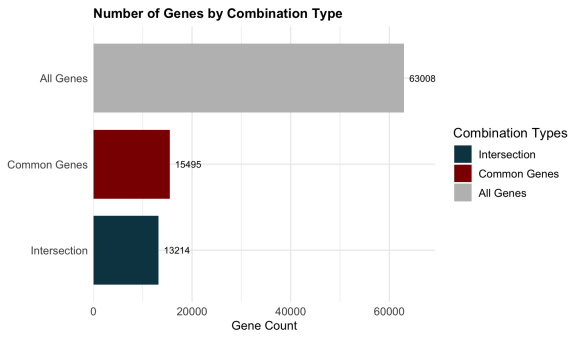

Figure 10: Number of genes available in Intersection, Common and All Genes.

that can be adjusted, in this project only the k-value is adapted. In agreement with the supervisors, and due to the otherwise massively higher computational effort, k is tuned independently from the general training/modeling pipeline. In addition, one can argue that a realistic imputation should not try to optimize the final model performance per se but should try to minimize the imputation error. Specifically, the optimal k is chosen by masking 10% of the existing gene expressions and then imputing those for a variety of k-values. The final selection of k = 35 minimizes the MSE of the imputed expression data.

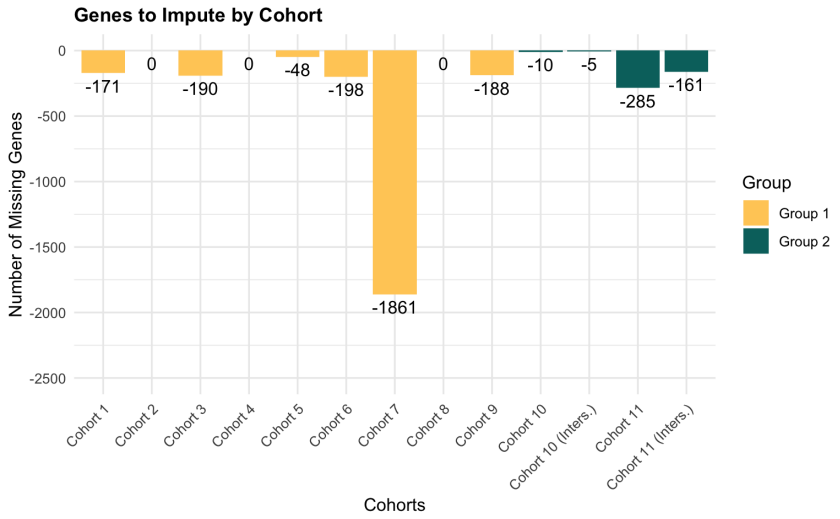

Figure 11: Number of imputed genes per cohort. For Cohorts 1-9 only w.r.t. Common Genes, for Cohorts 10 and 11 also w.r.t. Intersection Genes.

The cohorts from Group B also contain partially different genes compared to one another and to Group A. To use the data from Group B during the evaluation of the models fitted on either the Intersection Genes or Common Genes, these missing values had to be imputed as well. Figure 11 shows that although some expressions in Cohorts 10 and 11 were missing, with respect to Intersection Genes, the number of genes for which expressions had to be imputed was rather small. Regarding the Common Genes, the number for Cohorts 10 and 11 was higher. Whereas only cohorts from Group A are used to impute data in cohorts of Group A, for the imputation of cohorts from Group B data of all eleven cohorts is used. This decision is mostly based on the different time points at which the groups became available.
In general, the imputation of expression data leaves room for improvement, both within the implementation of k-NN as well as with respect to considering other imputation methods. However, since the focus of this project is the evaluation of the predictive performance of different machine learning models, the chosen imputation method offers a solid strategy to make use of more Gene Data than offered by Intersection Genes, without disproportionately shifting focus away from the key topic of the project. Besides this imputation, the gene expressions are standardized using z-transformation per cohort. The code performing this standardization in a cohort-wise manner was provided by the project partner.

2.5.3 Dimensionality Reduction

One way to tackle the issues induced by high dimensionality and multicollinearity is via dimensionality reduction. Ideally, this results in a latent representation of the Gene Data that is both lower-dimensional and contains uncorrelated features. A very common approach is to use principal component analysis (PCA) to arrive at such a latent space. However, one drawback of PCA is that it only models linear relationships in the data (Scholz et al., 2008), which limits the effective representational capacity of the method. Thus, this method is not ideal, especially in a context like Gene Data, where non-linear relationships and complex interactions are common. A more flexible method are Autoencoders (AEs), which consist of two main components: an encoder and a decoder. The encoder constructs a lower-dimensional latent space $\mathbf{z}$ of the original input $\mathbf{x}$ with the goal to obtain as much relevant information in $\mathbf{z}$ as possible. To do so, during training a decoder reconstructs the original feature space from the latent space $\mathbf{z}$. Therefore, the reconstruction error is minimized, which evaluates how well the original features space is reconstructed (Michelucci, 2022).
Using AEs in the context of Gene Data is a relatively common approach. For example, da Costa Avelar et al. (2023) apply an AE to Gene Data in the context of ovarian cancer. Here, the encoder consists of 128 nodes in the first hidden layer, which are then further reduced to 64 nodes in the encoding layer, thus resulting in a latent space with 64 features. To enable the representation of non-linearities, a ReLU activation function is applied. To measure the quality of reconstruction, the MSE is used during optimization. Due to computational constraints, this architecture is adapted in the context of this consulting project, instead of jointly tuning an AE together with the models. Nevertheless, a separate AE is trained for each potential training dataset, so for all possible training sets in nested resampling, tuning, and final model training (see Section 3.2) to provide a realistic latent representation of the respective test sets and to thus arrive at performance estimates as unbiased as possible.

3 Statistical Methodology

3.1 Models

3.1.1 Penalized Cox Proportional-Hazards Model

Architecture and Functionality: The Cox Proportional-Hazards (Cox PH) model models the so-called hazard function $h(t|\mathbf{x})$, which represents the risk of experiencing the event (here: BCR) at the given time point t, given that the event has not yet occurred until time point t. This is expressed via

$$h(t|\mathbf{x}) = h_0(t)\exp(\eta)$$

where $h_0(t)$ is the baseline hazard, $\eta = \mathbf{x}^T\boldsymbol{\beta}$ represents the linear predictor (wo. intercept) and $exp(\eta)$ corresponds to the relative risk or hazard ratio . The Cox PH model is semi-parametric, meaning that it does not make any distributional assumptions with respect to specific distribution of the survival times (Bradburn et al., 2003). However, it nevertheless requires a number of other assumptions to hold. First of all, it assumes proportional hazards, meaning that the hazard ratio stays constant over time. One approach to test this assumption is using the scaled Schoenfeld residuals. For a given covariate they represent the difference between the observed value and the expected value given the risk set R_k at time $t_{(k)}$. Weighting them inversely with respect to their (co)variances results in the scaled Schoenfeld residuals, which can then be used in a Schoenfeld residual test. Essentially a goodness-of-fit test, it tests the correlation between the Schoenfeld residuals (per covariate) and the survival time, where a correlation of zero indicates that the proportional hazards assumption (the null hypothesis) holds (In Junyong, 2019). While the proportional hazard assumption is arguably the most central one of the Cox PH model, it also assumes independent observations and non-informative or independent censoring. By design, it also assumes linearity of the coefficients $\boldsymbol{\beta}$ with respect to the log hazard. This structure of the Cox PH model also gives an intuitive way of interpreting the effect per covariate - either additively with respect to the log hazard or multiplicatively with respect to the hazard. Parameters are estimated using the negative partial log-likelihood as objective function

$$\log PL(\boldsymbol{\beta}) = P\ell(\boldsymbol{\beta}) = -\sum_{k=1}^{m}\left\{\mathbf{x}_{(k)}^{\top}\boldsymbol{\beta} - \log\sum_{j\in R_k}\exp(\mathbf{x}_j^{\top}\boldsymbol{\beta})\right\}$$

where R_k again denotes the risk set at time point $t_{(k)}$. The traditional Cox PH model is designed for use cases where $n >> p$. However, in scenarios where $n = p$ or $p >> n$, this traditional approach encounters problems in optimization. Penalized regression approaches offer a solution to that problem, by introducing a constraint on the estimated coefficients, resulting in the penalized Cox partial log-likelihood:

$$P\ell_{pen}(\boldsymbol{\beta}) = P\ell(\boldsymbol{\beta}) + J_{\alpha,\lambda}(\boldsymbol{\beta}),$$

where

$$J_{\alpha,\lambda}(\beta) = \lambda\left(\alpha\sum_{i=1}^{p}|\beta_i| + 0.5(1-\alpha)\sum_{i=1}^{p}\beta_i^2\right)$$

with $\lambda > 0$ and $0 \leq \alpha \leq 1$. For $\alpha = 0$ this results in a Ridge penalty, leading to smaller estimated coefficients but no induced sparsity. Setting $\alpha = 1$ results in Lasso, which not only reduces the magnitude of coefficients but also shrinks some of them to zero, providing an inherent method for feature selection (Cygu et al., 2021, Zou and Hastie, 2005). Combining Lasso and Ridge results in an Elastic Net penalty. The degree of regularization/shrinkage also depends on λ, with higher λ values yielding a more aggressive regularization. For a group of correlated predictors the Ridge penalty tends to shrink them towards each other, while Lasso tends to select one out of the group and to completely shrink the others. Elastic Net, mixes these behaviors, and by combining the strengths of both Lasso and Ridge can thus lead to an improved predictive performance (Zou and Hastie, 2005, Friedman et al., 2010, Simon et al., 2011).

Reason for Use and Potential Caveats: As a consequence of the high dimensionality of the present Gene Data, sparsity inducing penalties are an attractive approach, both due to their effective handling of high dimensionality and the inherent feature selection. Thus, both Lasso and Elastic Net are viable options, especially due to their different approaches of handling correlated features, which are also present in the available data. Another advantage of this model class is the interpretability of the estimated coefficients, which gives both insight into the estimated associations and a means to assess the feature importance. A potential limitation of the Cox PH model is that non-linear effects in the covariates can only be considered by explicitly transforming the respective covariates. Furthermore, interactions between covariates must be explicitly pre-specified by including interaction terms. Thus, the model's ability to describe the underlying hazard structure heavily depends on these predefined structures, which can be extremely labor-intensive for high-dimensional data.
As already mentioned, the general applicability of Cox PH stands and falls with the proportional hazards assumption. Thus, the above mentioned (dis-)advantages are only really relevant if this assumption holds in the first place. To test this assumption, a Schoenfeld residual test is applied per model (obtained from the different datasets, see Section 3.1.7). Then, using the selected covariates, a Cox PH model is refitted and then tested. The only variable for which the null hypothesis is rejected at a level of 0.05 is the Gleason Score. The global p-values per model and Gleason Score can be found in the Appendix (see A.1). To properly account for this phenomenon, a linear interaction between Gleason Score and time is added to the linear predictor of those models that use Clinical Data. This is done via a start-stop structure, splitting the patient's follow-up period into a maximum value of three intervals $[0, 6], (6, 64], (64, \infty)$. These interval boundaries are chosen via a visual inspection of the plotted Schoenfeld residuals of the Gleason Score across months for the different models (see A.1).

Implementation Details: For implementation, the glmnet (Friedman et al., 2010, Simon et al., 2011) package is used, mostly due to its very efficient optimization via the coordinate descent algorithm, the efficient computation of entire λ sequences, and a built-in cross-validation functionality. Besides λ, the α value is also optimized during hyperparameter tuning to find optimal trade-offs for both the regularization/sparsity and the handling of correlated features. While the λ-sequence is dynamically created by glmnet,

the potential α values are $\alpha \in \{0.6, 0.8, 1\}$.

3.1.2 Random Survival Forest

Architecture and Functionality: Random Survival Forests (RSF) extend traditional random forests for the setting of right-censored survival data. As a non-parametric method, RSF makes no assumptions with respect to the underlying hazard function. Following the general principles of bagging, RSF grows decorrelated trees by using bootstrapped subsets of the training data. At each split, a random subset of features is considered to determine the optimal split point. Deep trees are constructed, and the ensemble output is obtained by averaging the terminal node statistics (TNS). By averaging these TNS, RSF reduces variance compared to a single tree. At the same time, RSF still benefits from the high flexibility and effective modeling capacity of deep trees, leading to a model class that ideally is relatively robust against overfitting and at the same time can model highly complex interactions and relations within the data. Additionally, RSF specifically accounts for censoring in the splitting rules for growing the trees with the goal of constructing splits that maximize between-node survival differences (Ishwaran et al., 2021, Ishwaran et al., 2008). The log-rank test statistic is a common splitting rule in this context. Given a feature x, it maximizes the degree of separation using a cutoff c between the child nodes $L = \{x \leq c\}$ or $R = \{x > c\}$. For a time point t_k let $d_{k,L}, d_{k,R}$ be the number of events and $R_{k,L}, R_{k,R}$ the risk sets in the child nodes L and R. Consequently, the risk set in the "mother" node is defined as $Y_j = R_{k,L} + R_{k,R}$ with the number of events $d_k = d_{k,L} + d_{k,R}$. The log-rank split-statistic value for the split is then given by

$$L(x, c) = \frac{\sum_{k=1}^{m} \left(d_{k,L} - R_{k,L} \frac{d_k}{R_k} \right)}{\sqrt{\sum_{k=1}^{m} \frac{R_{k,L}}{R_k} \left(1 - \frac{R_{k,L}}{R_k} \right) \left(\frac{R_k - d_k}{R_k - 1} \right) d_k}}$$

where $|L(x, c)|$ can be interpreted as a measure of node separation, with higher values indicating a higher degree of separation (Ishwaran et al., 2021). The feature-split-point-combination (x^*, c^*) which maximizes $|L(x, c)|$ is thus selected as the optimal split. Per tree and per terminal node h in each tree, the RSF then estimates its TNS, namely both the survival function and the cumulative hazard function

$$H_h(t) = \sum_{t_{k,h} \leq t} \frac{d_{k,h}}{R_{k,h}}, \quad S_h(t) = \prod_{t_{k,h} \leq t} \left(1 - \frac{d_{k,h}}{R_{k,h}} \right).$$

Due to the inherent structure of an RSF, these are also the estimates $H(t|\mathbf{x})$, $S(t|\mathbf{x})$ for a given $\mathbf{x}$ given that the values of $\mathbf{x}$ resulted in an assignment to h. The ensemble cumulative hazard function and survival function are then obtained by simply averaging the individual tree outputs

$$\bar{H}(t \mid \mathbf{x}) = \frac{1}{\text{ntree}} \sum_{b=1}^{\text{ntree}} H_b(t \mid \mathbf{x}), \quad \bar{S}(t \mid \mathbf{x}) = \frac{1}{\text{ntree}} \sum_{b=1}^{\text{ntree}} S_b(t \mid \mathbf{x}).$$

Reason for Use and Potential Caveats: Due to its non-parametric nature, RSF evades any distributional assumptions regarding the underlying data structure. This inherent flexibility makes RSF particularly advantageous and applicable in scenarios where the assumptions of the Cox PH model may not be fully satisfied. Since this is the case for the models that include Clinical Data (Section 3.1.1), RSF is a viable option in this consulting project. In addition, since gene expression data often exhibit complex interactions, RSF poses an adequate method to flexibly and automatically model these dependencies without having to pre-specify them (as opposed to, e.g. in the Cox PH model). In addition, the stepwise form of the discriminative function allows to automatically capture nonlinear relationships between the covariates and the hazard structure. Again, this eliminates the need for any pre-specifications, making RSF a very convenient method in this context of high-dimensional data. Additionally, RSF provides a direct method for evaluating the feature importance, via the so-called variable importance (VIMP). For a given covariate, this measures the change in prediction error on a fresh test case if the respective variable is not available, given that the original forest was grown using that variable (for further information see Ishwaran et al. (2008)). While the VIMP gives a general measure for feature importance, interpreting the exact effect on survival per feature is not as straightforward as, in e.g., the Cox PH model. Another potential caveat is that RSF can become computationally very expensive, especially for high-dimensional datasets. Of course, this can partly be mitigated via a cost-effective choice of hyperparameters. For example, one can simply restrict the number of potential split points or the number of features to be considered during node construction. However, to ensure that predictive performance does not suffer from a overly cost-effective choice of hyperparameters, thorough hyperparameter tuning is necessary, which itself can then become computationally expensive.

Implementation Details: For this project the R-package randomForestSRC (Ishwaran and Kogalur, 2007) is used. Besides its fast and memory efficient implementation, it also uses categorical data without One-Hot-Encoding and thus handles it accordingly to the original R(S)F approach. Since the tissue variable is categorical, this is a clear advantage of this package over other options like RandomSurvivalForest from scikit-survival (Pölsterl, 2020), which does not have this feature. Table 3 shows the hyperparameters considered during tuning.

Parameter	**Values**
Nmb. of Trees	100
Nmb. of Rand. Split Points	10
Nmb. of Features for Splitting (Clin. Data and/or Gene Data)	sqrt, log2
Nmb. of Features for Splitting (Only Clin. Data)	sqrt, log2, None
Size of Terminal Nodes	10, 15, 20

Table 3: RSF hyperparameter grid.

3.1.3 Gradient Boosting

Architecture and Functionality: While boosting also falls into the category of ensemble techniques, it differs from bagging approaches. As opposed to averaging the estimates of multiple strong learners, the idea behind boosting is to combine multiple weak base learners into a strong model. These base learners are applied iteratively to stepwise reduce the residual error with respect to the respective loss function, which aims at jointly reducing the variance (by using weak base learners) and the bias (by combining these learners) of the resulting model. Starting from an initial model $f_\theta^{[0]}(\mathbf{x})$, in each iteration v, the loss of the current model $f_\theta^{[v-1]}(\mathbf{x})$ is examined. The choice of this loss function heavily depends on the given context, since it a) determines how the output of the Gradient Boosting (GBoost) model is interpreted and b) how to account for further particularities of the given data structure. Thus, to properly account for censoring, in the context of survival, a similar objective function to the objective function of the Cox PH model is used

$$Obj(\boldsymbol{\theta}) = -\sum_{k=1}^{m} \left\{ f_\theta^{[v-1]}(\mathbf{x}_{(k)}) - \log \sum_{j \in R_k} \exp(f_\theta^{[v-1]}(\mathbf{x}_j)) \right\}.$$

The outputs of the model $f_\theta^{[v-1]}(\mathbf{x})$, can be seen as a replacement for the linear predictor of the Cox PH model. Based on this, the pseudo residuals are then obtained, which are the derivative of the loss function with respect to the model output $f_\theta^{[v-1]}(\mathbf{x})$. Then, a base learner $b(\mathbf{x}, \hat{\boldsymbol{\theta}}^{[v]})$ is fitted to these pseudo residuals. Often, regression trees are used as base learners. In contrast to RSF, these trees are usually relatively shallow, following the rationale of weak base learners. The fitted base learner is then added to the current model via

$$f_\theta^{[v]}(\mathbf{x}) = f_\theta^{[v-1]}(\mathbf{x}) + \alpha^{[v]} b(\mathbf{x}, \hat{\boldsymbol{\theta}}^{[v]})$$

thus updating the model in a greedy manner. The weight of the respective base learner in the model is determined by the step size α, with higher values indicating a higher weight (Pölsterl, n.d., CatBoost, n.d.).

Reason for Use and Potential Caveats: Similar to RSF, GBoost generally benefits from all the advantages of tree-based models, e.g. automatic modeling of complex interactions and non-linearities, which again makes it a promising approach for Gene Data. However, the degree to which both challenges can be handled i.a. depends on the depth of the base learners. Deep base learners by themselves have a higher modeling capability but also increase the risk of overfitting for the complete ensemble. On the other hand, the reduced modeling capability of shallow trees is (by design) of course mitigated via the sequential combination of these base learners. Nevertheless, choosing trees that are too shallow can still be a caveat in modeling highly intricate interactions and relationships between multiple features. One way to combat this is to choose a middle ground for the tree depth. To prevent overfitting, one could then either reduce the number of iterations or the learning rate applied during the model updates. Generally speaking, while GBoost can potentially be a very strong model class for modeling highly complex data, its performance heavily depends on proper hyperparameter tuning, due to

its tendency to overfit. Another reason for using GBoost in this project is that it also provides a measure of feature importance. In the context of this consulting project, the so-called 'Prediction Values Change' is used, which measures how much, on average, the prediction changes if the feature value changes. The higher the importance of a feature, the bigger on average is the change to the prediction value, if this feature is changed (CatBoost, n.d.). However, similarly to the VIMP metric for RSF, the direction of a feature's effect can not be assessed using this method. As with all the other methods, the computational performance of GBoost also depends on the chosen hyperparameters. Similarly to RSF, a large number of potential split points or features to be considered during node construction can cause high computational expenses.

Implementation Details: To handle both categorical features and the computational challenges of GBoost efficiently, the Python package CatBoost (Dorogush et al., 2018) is used. In addition, a wrapper class is implemented that adheres to the scikit-learn API for estimator objects (Buitinck et al., 2013). This, in turn, allows the use of CatBoost in scikit-learns modeling pipelines. Thereby aspects like model selection or evaluation are handled efficiently and consistently during the complete modeling process (Section 3.2) (Pedregosa et al., 2011, Buitinck et al., 2013). Regarding the hyperparameter specification, early stopping is used to determine the maximum number of iterations necessary during training and to simultaneously decrease the risk of overfitting. The resulting hyperparameter grid can be found in Table 4. Here 'Iterations' refers to the maximum number of iterations possible and 'Rsm' to the percentage of features used at each split selection.

Parameter	**Values**
Iterations	500
Learning Rate	0.1
Tree Depth	3, 5
Min. Data in leaf	1, 3, 5, 10
Rsm (Clin. Data and/or Gene Data)	0.2
Rsm (Only Clin. Data)	0.2, 1

Table 4: GBoost hyperparameter grid.

3.1.4 DeepSurv

Architecture and Functionality: DeepSurv is a neural network-based approach for survival prediction, introduced by Katzman et al. (2018). The authors introduce the concept as a "Cox proportional hazards deep neural network", since it combines the idea of the Cox PH model and a deep neural network. Using fully connected layers, the model outputs a so-called risk score $\hat{h}_\theta(x)$, which has a similar role as the linear predictor of the

Cox PH model. Following the Cox PH model, DeepSurv uses the objective function

$$Obj(\boldsymbol{\theta}) = -\sum_{k=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}_k) - \log \sum_{j \in R_k} exp(h_{\boldsymbol{\theta}}(\mathbf{x}_j)) \right)$$

which is similar to the negative partial log-likelihood used in Cox PH. However, in the context of DeepSurv, the trainable parameters of the network are represented by $\boldsymbol{\theta}$. The authors do not make any predefined restrictions regarding the model architecture and the training/optimization process, which leaves room for an extensive hyperparameter tuning. Regarding the architecture of the model, this includes the number of hidden layers, nodes per layer, and the activation functions used across layers. For the optimization of the network this includes the respective optimization algorithm, and based on this refinements such as the learning rate or the use of momentum (if applicable). A notable feature of DeepSurv, though not relevant in the context of this project, is the model's ability to make treatment recommendations.

Reason for Use and Potential Caveats: Given that non-linear activation functions are used, the neural network structure of DeepSurv enables learning of complex, non-linear feature interactions, which makes it a viable approach in the context of this project. Furthermore, DeepSurv specifically serves as a baseline neural network based survival model. It provides insight into the predictive capability of such a model class, when only Gene and Clinical Data and relatively little regularization is used. This is a relevant baseline as the Cox-PASNet (Section 3.1.5) extends this concept by also implementing Pathway Data and thus artificially induced sparsity. However, one potential drawback of DeepSurv is its high risk of overfitting, depending on the chosen architecture. In addition, the dense architecture of DeepSurv might hinder the disentangling of separate effects. In addition, especially for deep and wide nets it comes with a high computational effort. Another disadvantage is the black-box nature of DeepSurv, due to its neural network architecture. While some other models provide measures for feature importance and/or effect directions, the interpretability of DeepSurv is much more limited.

Implementation Details: The implementation of DeepSurv is done using PyTorch. Similarly to CatBoost, a wrapper class that follows the scikit-learn API (Buitinck et al., 2013, Pedregosa et al., 2011) for estimators is also implemented. Again, this ensures that scikit-learn functionalities especially with respect to model selection and evaluation can be used for DeepSurv. To address the computational challenges, the DeepSurv modeling process (including all relevant classes) was not only implemented locally but also in a Google Colab Notebook. This allows the use of GPU during training, without the requirement for a GPU on the local computer. In the context of this project the number of hidden layers and layer width, the dropout rate, the learning rate, batch size and number of epochs are tuned. Additionally, early stopping is implemented with a patience of ten, and Adam optimization and ReLU activation are used, analogously to the implementations by Katzman et al. (2018). The grid used for hyperparameter optimization is specified in Table 5.

Parameter	Values
Nmb. Layers (Clin. Data and/or Gene Data)	[512, 256, 128, 64], [512, 256, 128], [512, 128], [512, 256], [256, 128], [1024], [512], [256], [128]
Nmb. Layers (Only Clin. Data)	[512, 128], [512, 256], [256, 128], [1024], [512], [256], [128], [64], [32], [16]
Learning Rates	0.00001, 0.0001
Batch Size	64
Epochs	500
Dropout	0.2, 0.4

Table 5: DeepSurv hyperparameter grid.

3.1.5 Cox-PASNet

Architecture and Functionality: Cox-PASNet (Cox PN) extends the general idea of DeepSurv by including Pathway Data in the network architecture (Hao et al., 2019). To achieve this, Cox PN consists of two input layers, namely the clinical and the gene layer. While the former simply receives the Clinical Data as input, the latter has the Gene Data as input features. This layer is then connected to the so-called pathway layer, in which each node corresponds to a pathway and thus a known functional relation between genes. To properly represent the fact that only a subset of genes is connected to a pathway and that a gene can load onto multiple pathways at once, a bi-adjacency matrix $\mathbf{M}^{(0)}$ is used to construct sparse connections. This is done via setting the weights $\mathbf{W}^{(0)}$ between gene and pathway layer to $\mathbf{W}^{(0)} = \mathbf{W}^{(0)} * \mathbf{M}^{(0)}$. The following layers then process this information further, again using sparse connections. For further information on these sparse connections see Hao et al. (2019). After the ultimate hidden layer h_2, the Clinical Data and the learned representations of the gene layer are concatenated into h^I and fed into the output layer, which then outputs, analogously to DeepSurv, a risk score. Similar to DeepSurv, this is then used in an objective function

$$Obj(\boldsymbol{\theta}) = -\sum_{k=1}^{m} \left(\mathbf{h}_k^I \boldsymbol{\beta} - \log \sum_{j \in R_k} \exp(\mathbf{h}_j^I \boldsymbol{\beta}) \right) + \lambda \left(\|\boldsymbol{\theta}\|_2 \right),$$

where $\mathbf{h}_j^I \boldsymbol{\beta}$ replaces the linear predictor of a traditional Cox PH model. Here, $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \mathbf{W}\}$, where $\boldsymbol{\beta}$ is similar to Cox PH coefficients and $\mathbf{W}$ is the set of weights used in the pathway and the two hidden layers.

Reason for Use and Potential Caveats: Examining the potential value of including biological data is the main argument for considering Cox PN in the context of this consulting project. The artificially enforced sparse connection between the gene layer and the pathway layer can be both an advantage and limitation. On one hand, using these predefined relations between genes might help the network to disentangle relevant associations both within the features and between the features and the output of the network properly. This can also prevent the network from overfitting on noise or associations irrelevant for predicting the outcome. On the other hand, restricting the interactions along

these predefined paths might hinder the effective capacity of the network. Furthermore, in order to use the Cox PN properly, only genes that are connected to a pathway are considered as input data. In the present context this limits the number of genes used as input to roughly 6000, leading to a potential information loss, compared to methods without that restriction on the input data. However, this restriction on genes with an already known connection to PCa could also have a refining effect on the input data, excluding genes that potentially would not have had any predictive value.

Implementation Details: The implementation of Cox PN is based on the official implementation provided by the authors, which is based on PyTorch (Paszke et al., 2019). Analogously to both DeepSurv and GBoost, a wrapper class is implemented, which enables the use of Cox PN in scikit-learn's model selection and evaluation functionalities (Buitinck et al., 2013, Pedregosa et al., 2011), ensuring an efficient and consistent implementation of the modeling process. Similarly to DeepSurv, Cox PN is also implemented in a Google Colab notebook, to thus use the computational advantages of GPUs during model training. Following the architecture proposed by the authors, the network consists of two hidden layers for the Gene Data. The number of nodes per hidden layer, as well as the dropout rate, the learning rate, and the weight decay applied during optimization, are tuned. Table 6 shows the hyperparameter space.

Parameter	**Values**
Learning Rate	0.001, 0.01
L2	0, 0.1, 0.2
Nmb. Epochs	500
Dropout Rate	0.1, 0.4, 0.6
Hidden Nodes h_1	64, 128, 256
Hidden Nodes h_2	32, 64, 128

Table 6: Cox PN hyperparameter grid.

3.1.6 Priority-Lasso

Architecture and Functionality: Klau et al. (2018) introduce Priority-Lasso (PrioLasso), which fits a number of penalized Cox PH regressions on ordered blocks of training data. Starting with the covariate block with the highest associated priority, the information already obtained in previous blocks is passed down to lower-priority blocks. More precisely, the linear predictor η_1 of the first block is added to the linear predictor of the second block as an offset. After fitting a penalized Cox PH regression on the data of the second block, the obtained linear predictor η_2 is then again passed as an offset to the third block, and so on. For a given block b, this results in the general formulation

$$\eta_b = \exp(\eta_{b-1} + \mathbf{x}_b^T \boldsymbol{\beta_b})$$

where x_b refers to the covariates used in block b and $\boldsymbol{\beta_b}$ to the corresponding coefficients. This hierarchical consideration of previously obtained information ensures that

only variance yet unexplained is considered in the respective block. According to Klau et al. (2018), the standard offset resulting from a block b is overly optimistic regarding the predictive value of the information contained in the respective block. This can lead to the underestimation of the influence of block $b+1$ conditioned on block b. This might be especially relevant in cases where low-priority blocks hold high predictive value. To mitigate this effect, the authors recommend using a cross-validated estimate of the offset. However, this option is currently unavailable for Cox PH models in the existing package. In addition, Priority Lasso offers an option to handle (block-wise) missing data. The first approach is to simply ignore the observations with missing data in the respective block for the fitting of the block. During prediction, the values of missing covariates are then simply set to zero, so that the offset of the previous block is propagated to the following block for these observations. The second approach is to impute the offset for missing values in the respective block, based on available data from other blocks (Klau et al., 2019).

Reason for Use and Potential Caveats: Due to the cohort-wise missing Gene Data, the ability to handle missing values makes PrioLasso an interesting modeling approach in the context of this consulting project. Thus, the blocks are defined according to the present missingness pattern. More specifically, the first and highest priority block contains the covariates that are present across all cohorts. The second block contains covariates that are present in all cohorts except for the first one. The third block consists of all covariates available in all cohorts but the second one and so on. This definition is further refined so that blocks contain at least data from three cohorts and at least 100 genes are contained per block, resulting in 16 blocks and 42995 genes. This refinement is mainly due to computational reasons. However, it should be noted that this refinement can, of course, impact performance negatively, especially if it removes blocks that would have had a high predictive value. To handle the missing data in the respective blocks, the first approach of ignoring the missing observations is used. Similarly to the restriction on defined block-data this is mostly due to computational reasons.

Implementation Details: The official R package prioritylasso (Klau et al., 2019) is used for implementation. Here, under the hood, the individual Cox PH models are fitted using the glmnet package. The package only allows for a Lasso penalty (i.e., $\alpha = 1$). Thus, the only hyperparameter optimized during tuning is λ, where specific λ-sequence is obtained in the under the hood call of cv.glmnet. One big caveat of this package is that no split IDs can be defined during resampling. This means that the complete resampling does not adhere to the leave-one-cohort-out strategy, which is vital for obtaining reliable performance estimates during resampling (see Section 3.2).

3.1.7 Overview of the Used Datasets

As already indicated by the previous chapters, some model classes are trained on the data sets introduced in Section 2. Namely, as Table 7 shows, Cox PH, RSF, GBoost and DeepSurv are trained on either Intersection Genes, Common Genes, Autoencoder Representations and/or Clinical Data. Cox PN is trained on a subset of Intersection Genes (those genes with associated pathways) and Clinical Data. For PrioLasso two

block structures are defined. In the first one, the first block only contains the Intersection Genes, whereas in the second one, this block also includes Clinical Data. Fitting all these models on all these datasets, thus results in 31 final models.

Model	Clin.	Inter.	Comm.	AE	Inter. + Clin.	Comm. + Clin.	AE + Clin.	Block	Block + Clin.	Inter.+Clin. + Pathway
Cox PH	X	X	X	X	X	X	X			
RSF	X	X	X	X	X	X	X			
GBoost	X	X	X	X	X	X	X			
DeepSurv	X	X	X	X	X	X	X			
Cox PN										X
PrioLasso								X	X	

Table 7: Resulting training datasets per model class.

3.2 Modeling Process

3.2.1 Resampling Strategy

Both during hyperparameter tuning for the final models and during nested resampling, a leave-one-cohort-out cross-validation (LOCO CV) is used as a resampling strategy. As the name implies, in this approach the cross-validation process iterates through the cohorts, excluding one cohort per split from the training data to use it as a test/validation set. Thus, each cohort serves as the test/validation set exactly once, e.g., resulting in nine splits for nine available cohorts (see Figure 12). The primary reason for using this strategy is the multi-cohort structure of the given data. Disregarding this structure would lead to overly optimistic performance estimates, whereas the chosen approach mimics the use case of a new patient (stemming from a new, unseen cohort) being modeled, which leads to more reliable estimates (Hornung et al., 2023).

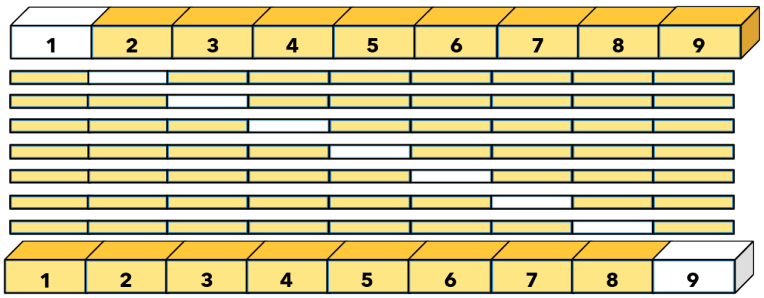

Figure 12: Exemplary LOCO CV process for nine cohorts.

As a search strategy, grid search is used, iterating through all possible combinations of the respective hyperparameter space. The main argument for using this strategy is the evaluation of relatively small hyperparameter grids per model.
As a performance metric during resampling, the Concordance Index (C-Index) is used.

The general intuition behind the C-Index is to evaluate how well a model adheres to the idea that observations with earlier events should also receive a higher notion of risk for the event in the respective model. In the context of this consulting project, Harrel's estimator of the C-Index is applied. It computes the proportion of comparable pairs whose relative risk is correctly modeled, i.e. the proportion of concordant pairs. A pair is considered to be comparable, if the true order of experience of the event (here: BCR) is known for a pair. A comparable pair is deemed concordant, if the higher risk of a subject also coincides with an earlier occurrence of the event than the other observation. Similarly to the AUC/ROC metric, a C-Index of 0.5 indicates a completely random assignment and a value of 1 indicates a perfectly predicted order (Longato et al., 2020). The main reason for using the C-Index, is its versatility in application, which makes it suitable for comparing model classes with different types of outputs reflecting the risk with respect to BCR occurring. However, this versatility also comes with certain drawbacks. Due to its ranking-based nature, the C-Index doesn't really assess the absolute accuracy of e.g. the predicted survival probabilities. Thus, a model might consistently over- or underestimate survival probabilities, but if it ranks the patients correctly, it still obtains a high C-Index. In addition, the C-Index can be less reliable for cases of high censoring, since the number of comparable pairs is relatively low in that setting.

3.2.2 Performance Estimation

As already stated in Section 1.2, there are two main performance evaluation tasks at hand. The first involves comparing the performance of the applied models against each other. As mentioned in Section 2.1, the final models are trained on the cohorts from Group A, with the optimal hyperparameters per final model obtained during the respective hyperparameter tuning via resampling on A. Cohorts 10 and 11 from B are thus used as independent test cohorts to compare the performance across the final models.
In the second task, the performances of the applied models are compared to the performance of the ProstaTrend-ffpe Score. Since the former are fitted on Group A and the latter is partly constructed using Group B, evaluating both the final model and the ProstaTrend-ffpe Score on either A or B, would lead to an unfair advantage for one method over the other.
To prevent this kind of bias, the performances for the ProstaTrend-ffpe Score are obtained for the cohorts from Group A. For a fair comparison, reliable performance estimates of the applied models on the A cohorts are obtained via nested resampling. As the name already suggests, nested resampling consists of two nested resampling procedures, where an inner resampling procedure is done on the training data of each outer split. This inner resampling returns the optimal hyperparameters for the training data of each outer split. These are then used to obtain an intermediate model, which then provides a performance estimate on the respective test data. The main reason for using nested resampling in this context is that it provides reliable performance estimates which are vital for a fair comparison between the applied models and the ProstaTrend-ffpe Score (Bischl et al., 2012).

4 Results

4.1 Performance Between Models

4.1.1 Performance Overview

As already stated in Sections 1.2, 2.1 and 3.2, the performance of the final models (trained on the different datasets based on Group A) is evaluated using the cohorts from Group B. Per model class Figure 13 shows the performance on these cohorts for the model-dataset combination, which yields the best average performance during nested resampling. The blue reference line represents the average performance (0.6928) obtained during nested resampling for all fitted models (other benchmarks are displayed in the Appendix A.2).

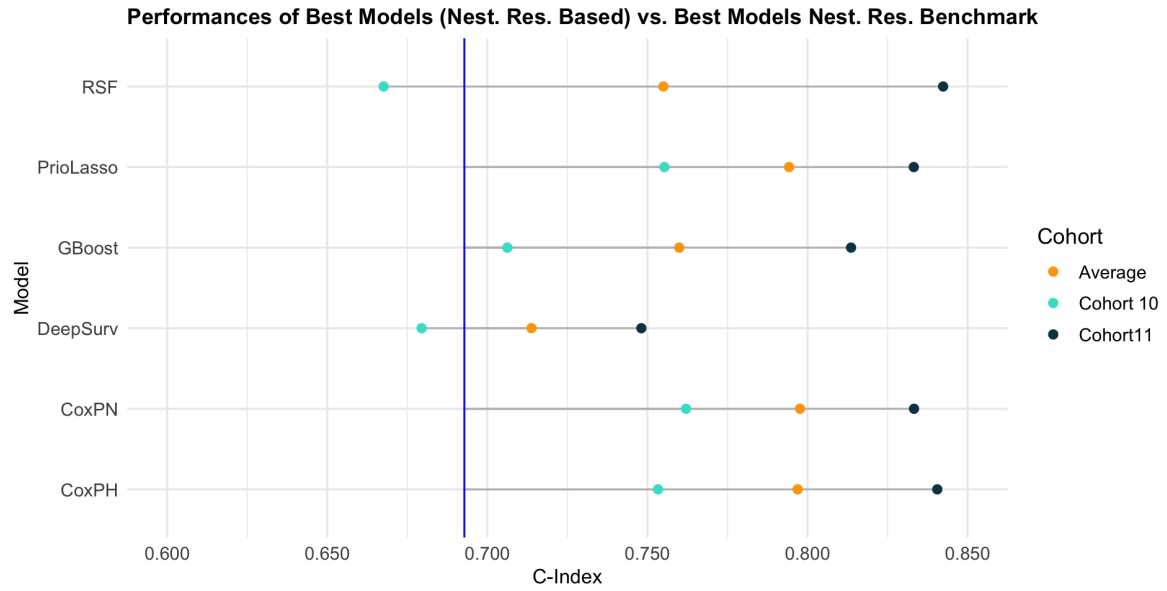

Figure 13: Model performance on Group B. For each model class, the best model-dataset combination (based on nested resampling result) is evaluated on Group B cohorts. The average performance of all models obtained during nested resampling serves as reference benchmark (blue line).

Regarding Cohort 10, this visualization shows the best performance for Cox PN, with Cox PH, and PrioLasso being in a relatively similar ballpark. RSF performs worst on this cohort with a C-Index even worse than the average performance in nested resampling. Interestingly, RSF performs rather well on Cohort 11, where it achieves the highest performance. Again, Cox PN, Cox PH and PrioLasso obtain relatively similar performances. DeepSurv has the worst performance on Cohort 11, as well as the worst average C-Index, whereas Cox PN has the best.
Looking at the performances on Cohorts 10 and 11, a high variability across model classes becomes apparent. While all models perform better on Cohort 11 than on Cohort 10, for Cox PN, Cox PH, PrioLasso, and DeepSurv this difference is relatively similar ($\sim$ 0.1 difference between performances). In contrast, RSF has a much higher performance difference of roughly 0.175.

Generally, when comparing the performances on these two cohorts it is important to keep in mind that for Cohort 11 a different prognostic endpoint with longer times until event/censoring is used. Since the C-Index only provides a notion of "ranking capability" per model, the models might indeed be able to rank the observations from Cohort 11 well, and even more accurately than for Cohort 10. However, this does not necessarily mean that event probabilities are actually predicted well. Thus, if one wants to assess the performances of the models with respect to BCR, the performances on Cohort 10 likely provide the more reliable and representative picture. Based on this, one can consider Cox PN, PrioLasso and Cox PH as the most promising models classes in the context of modeling BCR, whereas DeepSurv and RSF seem to struggle on Cohort 10.

4.1.2 Survival Curves and Discriminative Capabilities

Based on this assessment, Figure 14 illustrates the discriminative capabilities of a very strong model-dataset combination (Common Genes + Clinical Data - Cox PH) and a relatively weak model-dataset combination (AE - DeepSurv) for the patients of Cohort 10. The plot shows the predicted survival curves obtained for each model class separately for high and low risk patients, following the risk definition based on the respective ProstaTrend-ffpe Score values (see Section 1). Both models predict a lower BCR probability across months for low-risk patients. However, it is apparent that the discriminative capability of the Cox PH model is better than the discriminative capability of the DeepSurv model, which is consistent with their respective performances on Cohort 10.

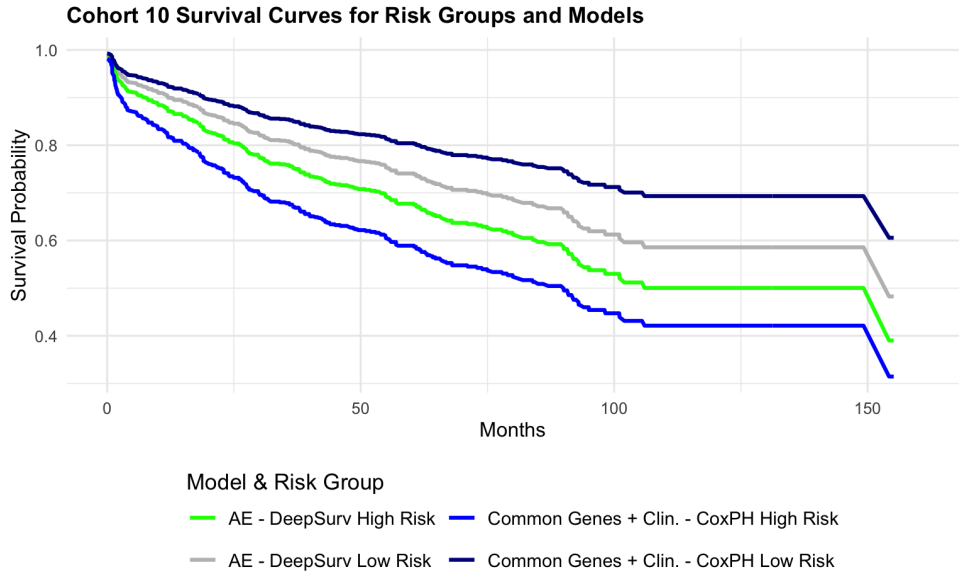

Figure 14: Estimated survival curves for Cohort 10 based on Cox PH and DeepSurv. Cox PH is trained on Clinical Data+Common Genes and DeepSurv on the AE Representation. Displayed are the estimated, aggregated survival probabilities for low and high risk patients for both models.

4.1.3 Performance For Different Training Datasets

To provide a more detailed overview of the model performances Figure 15 shows the results for each model-dataset combination trained on Group A and evaluated on Cohort 10. The main reason for only displaying performances on Cohort 10, is that this cohort provides a more reliable evaluation with respect to BCR compared to Cohort 11 (see Section 4.1). Overall, Cox PN achieves the highest C-Index (0.7620). Across the models that are trained on the different combinations of Intersection Genes, Common Genes, Autoencoder Representations and/or Clinical Data, Cox PH outperforms all the others across all datasets. In addition, the best performance of Cox PH (0.7533 on the combination of Clinical Data and Common Genes), is relatively close to the performance of Cox PN. Generally, the ranking of the model classes based on their performances is rather consistent across datasets. The only major exception to this is DeepSurv trained on the Autoencoder Representations (with and without Clinical Data). In this context, the performance of DeepSurv falls to the last place, whereas it has the third best performance for the other datasets. A detailed overview of these results can be found in the Appendix A.4.
Regarding the value of using Gene Data in general compared to only Clinical Data, the plot does not provide an entirely coherent picture. While some models such as Cox PH and GBoost generally perform better for only Gene Data compared to only Clinical Data, other model classes like RSF do not show this relation. Examining the value of combining Clinical Data and Gene Data, the figure indicates that this combination outperforms each one of them as standalone input datasets.

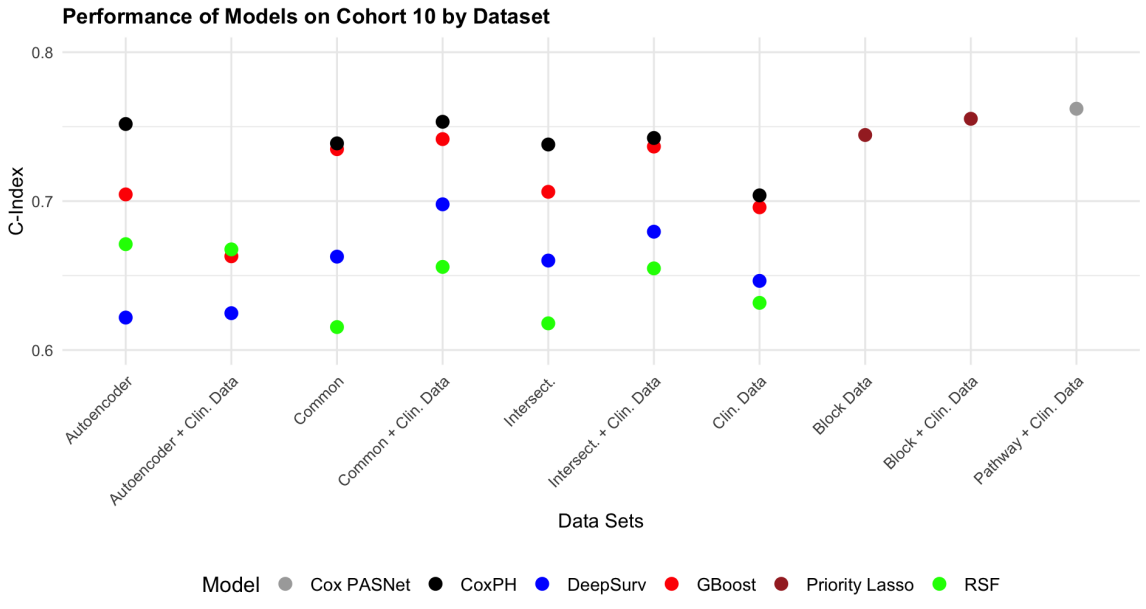

Figure 15: Model performance on Cohort 10 per training dataset.

4.2 Performance Between Models and ProstaTrend-ffpe Scores

Since the ProstaTrend-ffpe Score was constructed solely using Gene Data, only models that are purely trained on Gene Data are included in the comparison between the ProstTrend-ffpe Score C-Indices and the models' C-Indices (obtained during nested resampling, see Section 3.2). Figure 16 shows the performances of the best purely Gene Data based model per model class compared to the C-Index of the ProstaTrend-ffpe Score per cohort from Group A (A.3 displays the analog based on all datasets). Generally, the models seem to (slightly) outperform the ProstaTrend-ffpe Score. For Cohorts 3, 4, 7, 8, and 9 the median model performance is above the ProstaTrend-ffpe Score performance. On Cohorts 2, 5, and 6, the ProstaTrend-ffpe Score outperforms the median of models, whereas for Cohort 1, the performance between the two is nearly on par.

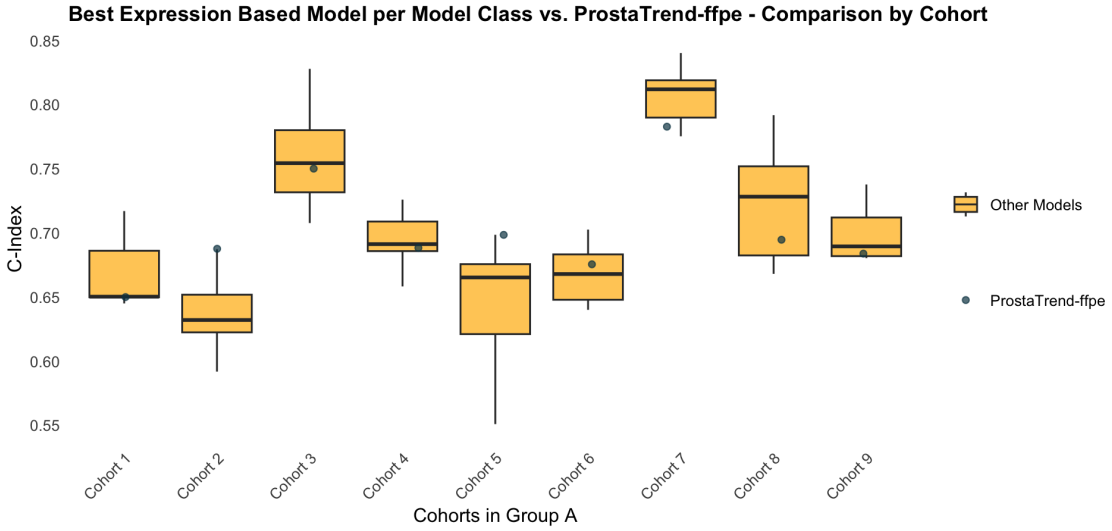

Figure 16: Performance comparison on Group A of ProstaTrend-ffpe Score and gene based models. For models, the nested resampling performance is used.

Table 8 provides further insight into the performances of the models compared to the ProstaTrend-ffpe Score. It provides the nested resampling based performances per model class and Gene Data combination aggregated across the cohorts from Group A. Comparing this with the aggregated performance of the ProstaTrend-ffpe Score across Cohorts 1-9, this indicates that, on average, only Cox PH outperforms the ProstaTrend-ffpe Score across cohorts.

Dataset	CoxPH	GBoost	RSF	DeepSurv	Cox PN	PrioLasso	ProstaTrend-ffpe
AE	0.7076	0.6497	0.6604	0.6188	-	-	-
Common	0.7161	0.6556	0.6691	0.6865	-	-	-
Intersect.	0.7166	0.6928	0.6796	0.6742	-	-	-
Group B Genes	-	-	-	-	-	-	0.7013

Table 8: Performance comparison of ProstaTrend-ffpe Score and models across Gene Data. For the models, the mean nested resampling performance across cohorts is used.

4.3 Model Performance Across Different Cohorts

Regarding the general performance across cohorts, Figure 16 additionally shows that there is a high variability in performance for both the ProstaTrend-ffpe Score and the models. A potentially interesting observation in this context is the relation between the proportion of censored patients per cohort and the performances of the methods on the respective cohort. Figure 17 shows that some cohorts have a much higher percentage of censoring than others, with the highest rate of 87% being obtained by Cohort 10, while Cohorts 1 and 9 have the lowest rates of 51%. Specifically, based on the performances on Group A, Figure 18 shows a correlation between method performance on a given cohort and the percentage of censored patients in that cohort. Here, the ProstaTrend-ffpe Score shows the strongest correlation with a value of 0.607. Regarding the models, Cox PN has the highest correlation (0.491), whereas Cox PH has the lowest correlation (0.066) between cohort performance and percentage of censored patients.

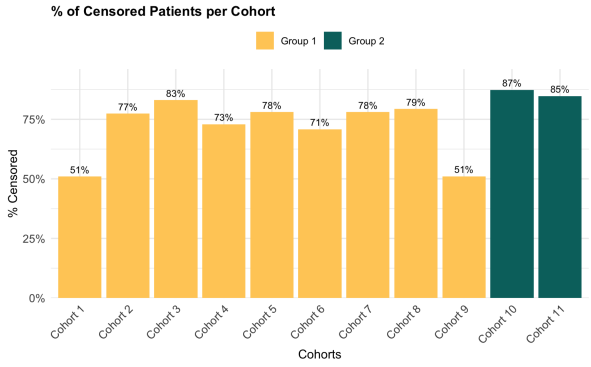

Figure 17: Overall share of censored patients across all cohorts.

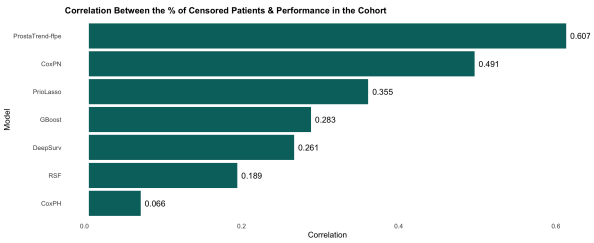

Figure 18: Correlation between censoring rates and performance.

However, it remains unclear whether the underlying reason of this performance discrepancy across cohorts is inherently caused by the different censoring rates. It might be that some model classes may be more prone to bias towards predicting lower BCR probabilities during training, particularly when exposed to data with a high proportion of censored cases. This could then result in struggles on cohorts that exhibit lower censoring rates, where the model may struggle to generalize accurately. Another reason could also be that some cohort specific covariates are missing during modeling. For example, if a high risk variable is missing, it may not affect high-censoring cohorts, as most patients have no observed event in respective follow-up period. However, it could be crucial for low-censoring cohorts since the variable would help to predict BCR. Another potential reason of this discrepancy across cohorts and the correlation between censoring proportion and model performance, could be the C-Index as a performance metric. As already stated in Section 3.2.1, the C-Index can be less reliable in cases where high censoring is present. The reduced number of comparable pairs in such context might inflate the C-Index thus giving a false impression of the model performance. Regarding the ProstaTrend-ffpe Score, another potential reason might be that this method is trained on cohorts with relatively high overall censoring percentages (Group B), thus potentially struggling with lower censoring patterns.

However, it has to be noted that the censoring percentage alone does not provide a complete picture of the underlying target distribution, since it does not reflect the rate of BCR happening during the follow-up and thus does not fully describe the progression of the survival process. As a consequence, the above analysis sheds some light on potential weaknesses of the models classes, the evaluation metric and the underlying training data, but it does not provide a complete explanation of the observed phenomenon.

4.4 Feature Importance Across Models

As already mentioned in Section 3.1, only Cox PH, RSF, GBoost and PrioLasso provide measures of feature importance. To provide a proper and meaningful comparison of feature importance across models, only models that are trained on the same dataset are compared. Thus, PrioLasso is excluded during the following analysis, since it is trained on the block-wise dataset of Gene Data and/or Clinical Data.

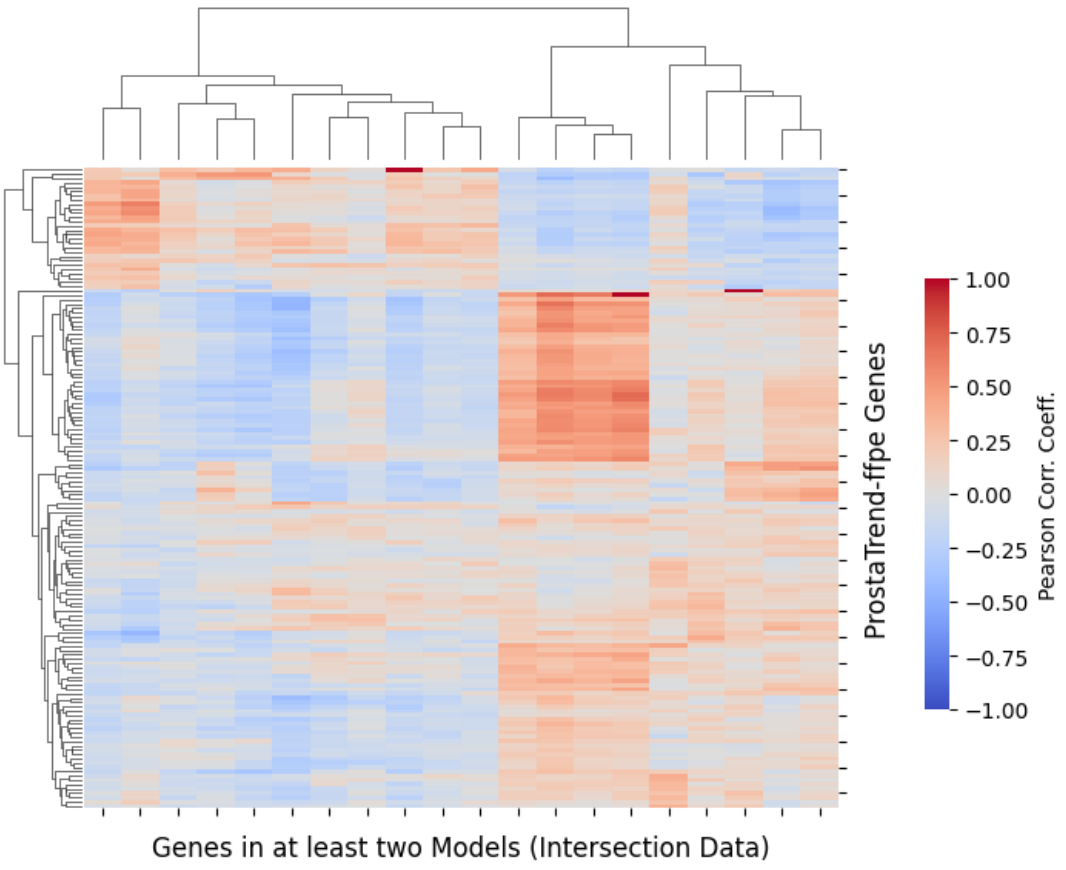

Figure 19: Clustered heatmap for genes in ProstaTrend-ffpe and genes selected in at least two models (Intersection Data).

The first question of interest is how the selected genes across models relate to the genes present in ProstaTrend-ffpe. Similar to Section 4.2, since the ProstaTrend-ffpe Score is obtained only using Gene Data, the models purely trained on Gene Data are used

for comparison. In addition, to avoid any uncertainties potentially introduced by the imputation process only the models trained on the Intersection Genes are examined. All in all 1068 genes are selected across the three models, 20 of which are contained in the ProstaTrend-ffpe Score. Only 20 genes are selected across at least two models, and only three of them are contained in the ProstaTrend-ffpe Score. Whilst this seems like a relatively small overlap between the methods, it has to be noted that from the beginning, only 149 of ProstaTrend-ffpe's 204 genes are contained in Intersection Genes, which of course limits the direct comparability of results. However, when examining the general relation between the genes selected in at least two models and the 149 genes present in the ProstaTrend-ffpe Score, Figure 19 shows that based on the Pearson correlation coefficient some clusters of genes form between the two genes sets. This indicates, that whilst not the exact same genes are selected, they potentially share general functional relationships, expressed by those clusters of high positive (red) and high negative correlation (blue). A version of this heatmap with the corresponding gene names can be found in the Appendix (see A.5). Regarding models only trained on clinical data, table 9 shows that Gleason Score is not only consistently selected across models, but also has the highest rank across all models. Preoperative PSA is also selected across all three models, ranking on position two/three importance wise.

Feature	**Count**	**Highest Rank**	**Lowest Rank**
Gleason Score	3	1	1
Preoperative PSA	3	2	3
Tissue Preservation	2	2	3
Age	2	4	4

Table 9: Feature importance Clinical Data.

Since the combination of clinical and Gene Data shows an increased predictive performance across model classes, the selected features for the combination of Intersection and Clinical Data is evaluated as well. Across all models 1313 genes are selected, whereas 47 are selected in at least two models. Regarding the clinical data the general tendency of importance consists, with Gleason Score being selected across two models, where it once has the highest and once the second highest importance rank. Preoperative PSA is selected only once, with the second highest rank in that model.

5 Conclusion

5.1 Summary of Results

During this statistical consulting project, the prognostic endpoint BCR is modeled using six machine learning and statistical models. The choice of models is based on the different challenges imposed by the present Gene Data, such as high dimensionality, complex interactions and cohort-wise missing data. For example, high dimensionality is i.a. addressed by sparsity inducing penalties (both in PrioLasso and Cox PH) and artificially created sparse layers, where the sparsity is based on predefined biological Pathway Data. The inclusion of Pathway Data extends the original scope of the statistical consulting project, with the Pathway Data being explicitly requested by the project team. Complex interactions and non-linearities are handled in a number of different model classes e.g., RSF, GBoost and the neural network-based approaches. The challenge of different gene availability across cohorts is addressed via PrioLasso, where the Block Data pattern is explicitly adapted to this use case. This challenge is also addressed during preprocessing, where different datasets are constructed via intersection, imputation, and dimensionality reduction.
All in all, these steps result in 31 model-dataset combinations being trained on Group A and evaluated on Group B, with a focus on the performance on Cohort 10. Here, the overall best performance is achieved by Cox PN closely followed by Cox PH and PrioLasso. Generally, it seems that model classes that explicitly induce sparsity (either artificially e.g. Cox PN or data driven e.g. PrioLasso, Cox PH) outperform less explicitly regularized/sparse approaches.
In addition, the performances of the models and the already existing ProstaTrend-ffpe Score are evaluated. To ensure a fair comparison, the cohort-wise performances of the models obtained during nested resampling on Group A are compared with the performance of the ProstaTrend-ffpe Score on the cohorts of Group A. While no entirely coherent picture arises, the models outperform the ProstaTrend-ffpe Score in the majority of cohorts.
Extending the original goals of the consulting project, some additional analyses are done. Firstly, an evaluation of the model performances across different training datasets is executed, which shows a clear benefit in combining both Gene and Clinical Data during modeling. Whilst no coherent picture arises regarding Gene Data vs. Clinical Data, the combination of both increases performance for nearly all model classes.
Another interesting aspect about both the models and the ProstaTrend-ffpe Score is, that their performances vary greatly across cohorts. While there is a correlation between method performance and censoring rates in the respective cohort, this is not necessarily the real cause for this discrepancy. Other potential explanations include missing covariates or limitations of the C-Index as evaluation metric.
Lastly the feature importance across a subset of models is evaluated. Only a very small subset of genes, that are selected in the majority of models are simultaneously contained in ProstaTrend-ffpe. However, these genes partly show a high correlation with one another, indicating that both the ProstaTrend-ffpe Scores and the models capture similar functional relationships even though different genes are considered important. Regarding the Clinical Data, Gleason Score and Preoperative PSA are selected as the most important covariates, which is in line with existing medical research.

5.2 Limitations and Further Work

Whilst the chosen methods address the imposed challenges, some limitations still remain. First of all, the differing data availability restricts the subsets of usable genes for the majority of models. Thus, a possible extension might be to adapt the logic behind PrioLasso to other model classes, to leverage information contained in the Gene Data to a higher degree. Another extension regards the methods used for imputation and dimensionality reduction during the preprocessing. For example, an interesting approach might be to use Autoencoders that include Pathway Data during the generation of latent representations, as introduced by da Costa Avelar et al. (2023).
In addition, one can argue that the modeling process itself has some weaknesses. The first one is the relatively small hyperparameter grid selection used during tuning, which might be especially relevant for models that are highly sensitive to their hyperparameters. However, these small grids are mostly caused by computational limitations, which are partly caused by the high dimensionality of the Gene Data. For example, tuning some of the models on local computers (i.e., 16GB RAM, no GPU) took up to a week, even with the relatively small grids used in this project. Thus, higher computational power would allow for tuning more comprehensive parameter grids, possibly providing better results. In addition, some model implementations such as DeepSurv, Cox PN, and GBoost explicitly allow the use of GPU, which can further enhance computational performance.
Another limitation of the modeling process is the use of the C-Index as an evaluation metric. Whilst it is easily applicable and especially suited for comparing model classes with different outputs, it does have some limitations, which might limit the comparability of results, especially across cohorts with very different target distributions. Thus, a sensible extension is to use other metrics, such as an integrated Brier score to gain a more precise understanding of the underlying weaknesses and strengths of the model classes.

A Appendix

A.1 Proportional Hazards Assumption

Dataset	P-Value Gleason Score	P-Value Global
Clinical Data	0.0013	0.0050
Intersection Genes + Clinical Data	0.0100	0.3567
Common Genes + Clinical Data	0.0031	0.1511
Autoencoder + Clinical Data	0.0096	0.3712

Table 10: Schoenfeld Residual Test p-values for different clinical data combinations.

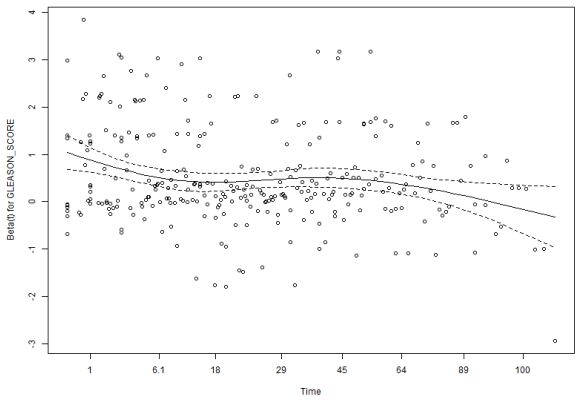

Figure 20: Schoenfeld residuals for Gleason Score (Clinical Data)

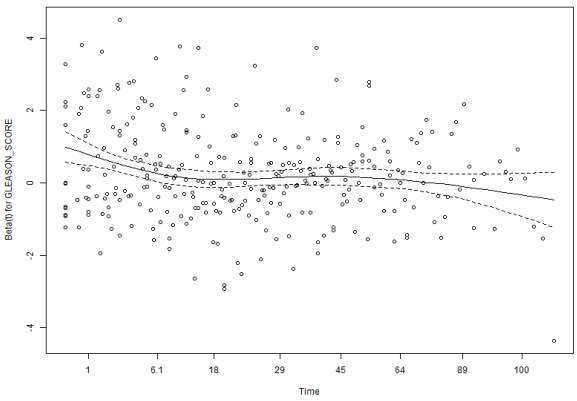

Figure 21: Schoenfeld residuals for Gleason Score (Clinical Data and Intersect. Genes).

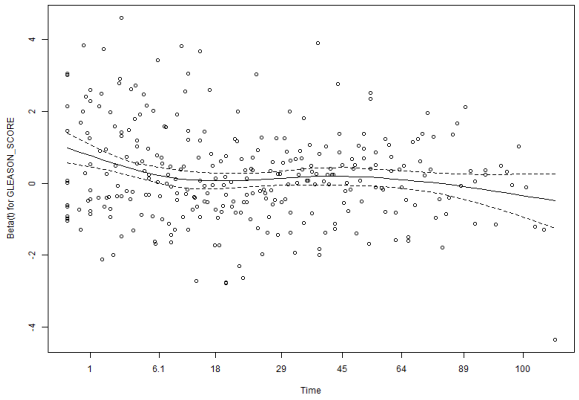

Figure 22: Schoenfeld residuals for Gleason Score (Clinical Data and Common Genes)

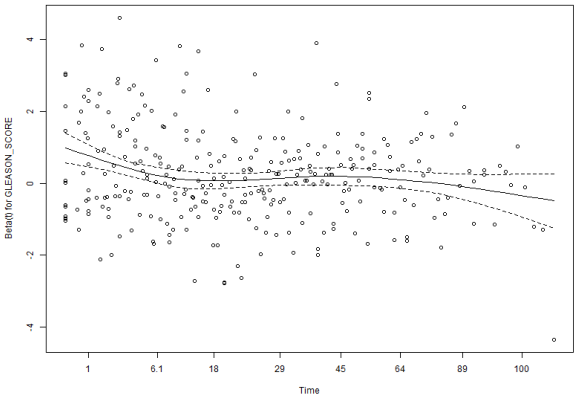

Figure 23: Schoenfeld residuals for Gleason Score (Clinical Data and AE Data).

A.2 Further Model Performance Comparisons

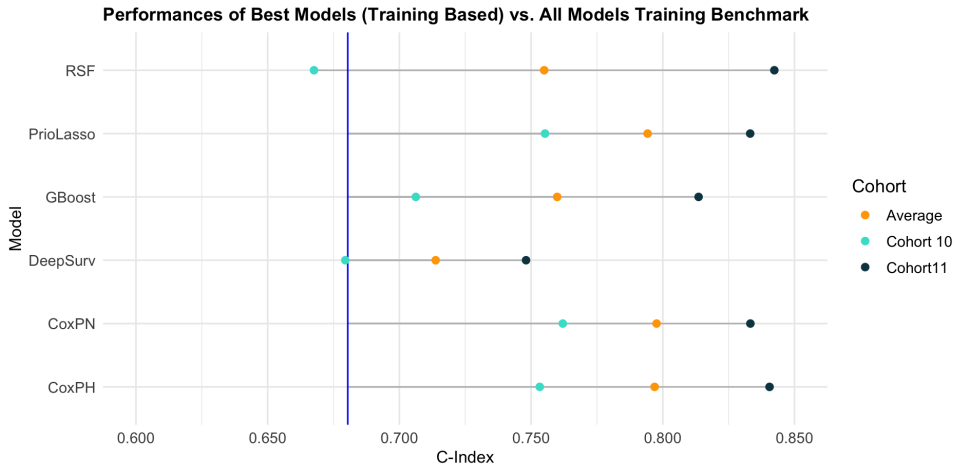

Figure 24: Best model-dataset combination per model class (based on nested resampling) evaluated on Group B. Benchmark: Mean perf. during nested resampling.

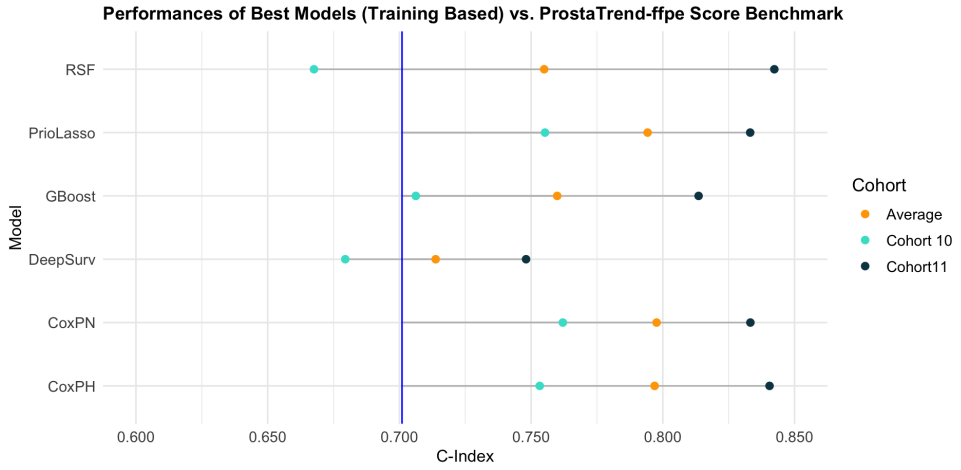

Figure 25: Best model-dataset combination per model class (based on nested resampling) evaluated on Group B. Benchmark: ProstaTrend-ffpe Score.

A.3 Further Performance Comparison Across Group A Cohorts.

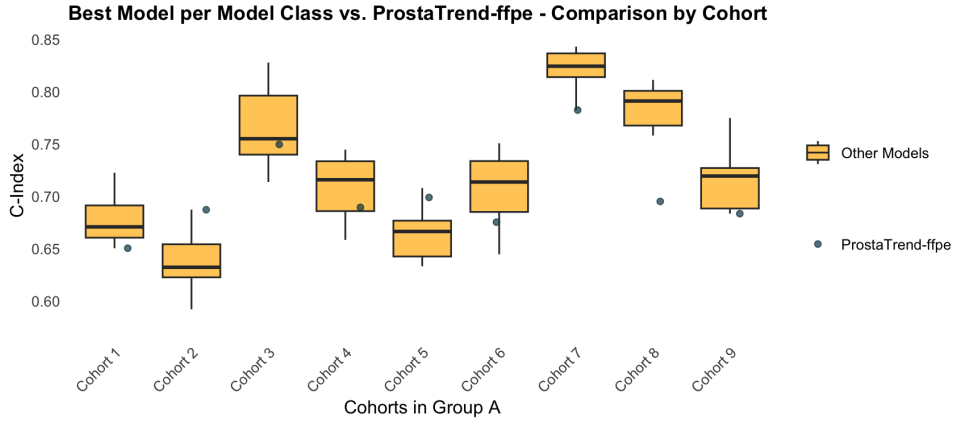

Figure 26: Comparison of all model performances with ProstaTrend-ffpe Scores on Cohorts 1-9.

A.4 Performance Across Different Training Datasets

Dataset	Mean	SD	C-Index Cohort 10	C-Index Cohort 11
Autoencoder	0.7076	0.0393	0.7518	0.7891
Clin. Data + Autoencoder	0.7254	0.0640	0.5996	0.6967
Common	0.7161	0.0517	0.7388	0.8151
Clin. Data + Common	0.7280	0.0613	0.7533	0.8405
Intersection	0.7166	0.0521	0.7380	0.8130
Clin. Data + Intersection	0.7275	0.0611	0.7424	0.8396
Clin. Data	0.7052	0.0847	0.7039	0.7735

Table 11: Cox PH: Overview of performance. Mean & standard deviation of nested resampling performance, and performances on Cohort 10& 11. (CI = C-Index).

Dataset	Mean	SD	C-Index Cohort 10	C-Index Cohort 11
Autoencoder	0.6497	0.0856	0.7045	0.7986
Clin. Data + Autoencoder	0.6711	0.0456	0.6630	0.6801
Common Genes	0.6556	0.0583	0.7349	0.8035
Clin. Data + Common Genes	0.6738	0.0661	0.7416	0.8035
Intersect. Genes	0.6928	0.0614	0.7062	0.8136
Clin. Data + Intersect. Genes	0.6673	0.0633	0.7366	0.8215
Clin. Data	0.6869	0.0862	0.6959	0.7746

Table 12: GBoost: Overview of performance. Mean & standard deviation of nested resampling performance, and performances on Cohort 10& 11. (CI = C-Index).

Dataset	Mean	SD	C-Index Cohort 10	C-Index Cohort 11
Autoencoder	0.6604	0.1062	0.6711	0.7968
Clin. Data + Autoencoder	0.7002	0.0882	0.6676	0.8423
Common Genes	0.6691	0.1072	0.6154	0.7669
Clin. Data + Common Genes	0.6668	0.1053	0.6558	0.7757
Intersect. Genes	0.6796	0.0956	0.6179	0.7834
Clin. Data + Intersect. Genes	0.6672	0.0991	0.6548	0.7727
Clin. Data	0.6767	0.0886	0.6317	0.7672

Table 13: RSF: Overview of performance. Mean & standard deviation of nested resampling performance, and performances on Cohort 10& 11. (CI = C-Index).

Dataset	Mean	SD	C-Index Cohort 10	C-Index Cohort 11
Autoencoder	0.6188	0.1021	0.6218	0.6887
Clin. Data + Autoencoder	0.5397	0.0513	0.6247	0.4313
Common Genes	0.6865	0.0567	0.6627	0.8173
Clin. Data + Common Genes	0.6848	0.0478	0.6979	0.7683
Intersect. Genes	0.6742	0.0514	0.6601	0.7019
Clin. Data + Intersect. Genes	0.6913	0.0459	0.6795	0.7481
Clin. Data	0.6732	0.0502	0.6464	0.7447

Table 14: DeepSurv: Overview of performance. Mean & standard deviation of nested resampling performance, and performances on Cohort 10& 11. (CI = C-Index).

Dataset	Mean	SD	C-Index Cohort 10	C-Index Cohort 11
Pathway Data	0.6821	0.0769	0.7620	0.8333

Table 15: Cox PN: Overview of performance. Mean & standard deviation of nested resampling performance, and performances on Cohort 10& 11. (CI = C-Index).

Dataset	Mean	SD	C-Index Cohort 10	C-Index Cohort 11
Clin. Data + Block Data	0.7033	0.0797	0.7553	0.8332
Blocks Data	0.6951	0.0586	0.7444	0.7796

Table 16: PrioLasso: Overview of performance. Mean & standard deviation of nested resampling performance, and performances on Cohort 10& 11. (CI = C-Index).

A.5 Heatmap Feature Importance

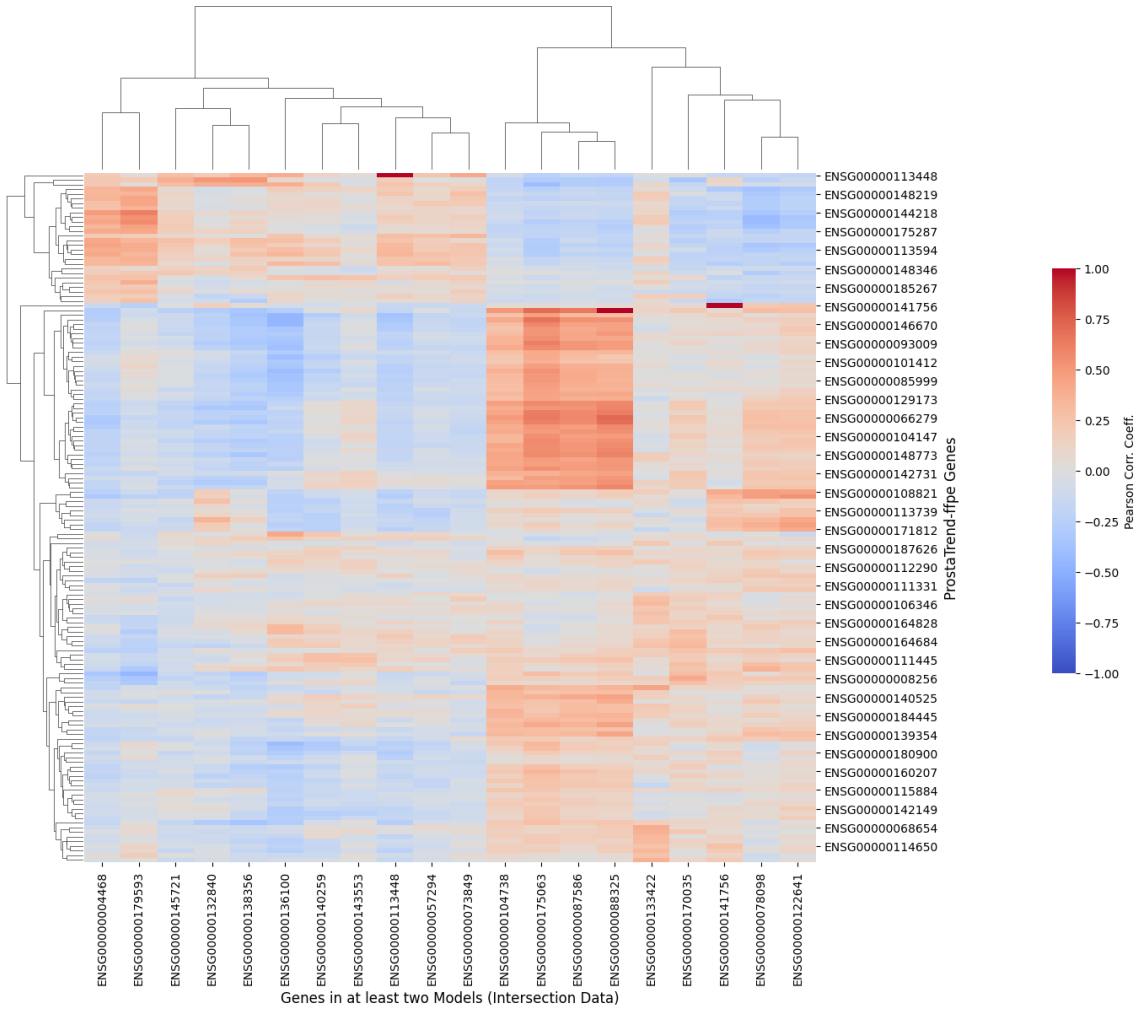

Figure 27: Clustered heatmap for genes in ProstaTrend-ffpe and selected in at least two models (Intersection Data).

A.6 Baseline Hazard

For predicting the survival via a DeepSurv model, the baseline hazard has to be additionally estimated.
To do so the Breslow estimator discussed by Lin (2007) is used. At an event time t, the estimator is given by

$$\lambda_0(t) = \frac{d(t)}{R(t)}, \tag{1}$$

where $d(t)$ is the number of events at time t, and $R(t)$ is the sum of the risk scores for all patients who are at risk at time t.
The cumulative baseline hazard function is then defined as

$$\Lambda_0(t) = \int_0^t \lambda_0(s)\, ds.$$

B Electronic Appendix

B.1 GitHub

The code used for this project, as well as CSV-files containing the model results can be found on **GitHub**.

References

Alsalem, M., Zaidan, A., Zaidan, B., Hashim, M., Madhloom, H., Azeez, N. and Alsyisuf, S. (2018). A review of the automated detection and classification of acute leukaemia: Coherent taxonomy, datasets, validation and performance measurements, motivation, open challenges and recommendations, *Computer Methods and Programs in Biomedicine* **158**: 93–112.
URL: *https://www.sciencedirect.com/science/article/pii/S0169260717314761*

Austin, P. C., White, I. R., Lee, D. S. and van Buuren, S. (2021). Missing data in clinical research: a tutorial on multiple imputation, *Canadian Journal of Cardiology* **37**(9): 1322–1331.

Bischl, B., Mersmann, O., Trautmann, H. and Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation, *Evolutionary computation* **20**: 249–75.

Bradburn, M. J., Clark, T. G., Love, S. B. and Altman, D. G. (2003). Survival analysis part ii: Multivariate data analysis – an introduction to concepts and methods, *British Journal of Cancer* **89**(3): 431–436.
URL: *https://doi.org/10.1038/sj.bjc.6601119*

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B. and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project, *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122.

CatBoost (n.d.).
URL: *https://catboost.ai/docs/en/concepts/fstrregular-feature-importance*

Christensen, N. J., Demharter, S., Machado, M., Pedersen, L., Salvatore, M., Stentoft-Hansen, V. and Iglesias, M. T. (2022). Identifying interactions in omics data for clinical biomarker discovery using symbolic regression, *bioRxiv* .
URL: *https://www.biorxiv.org/content/early/2022/05/24/2022.01.14.475226*

Cygu, S., Dushoff, J. and Bolker, B. M. (2021). pcoxtime: Penalized cox proportional hazard model for time-dependent covariates.
URL: *https://arxiv.org/abs/2102.02297*

da Costa Avelar, P. H., Wu, M. and Tsoka, S. (2023). Incorporating prior knowledge in deep learning models via pathway activity autoencoders.
URL: *https://arxiv.org/abs/2306.05813*

Dorogush, A. V., Ershov, V. and Gulin, A. (2018). Catboost: gradient boosting with categorical features support.
URL: *https://arxiv.org/abs/1810.11363*

Egevad, L., Micoli, C., Samaratunga, H., Delahunt, B., Garmo, H., Stattin, P. and Eklund, M. (2024). Prognosis of gleason score 9–10 prostatic adenocarcinoma in needle biopsies: a nationwide population-based study, *European Urology Oncology* **7**(2): 213–221.

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

Gretzer, M. B. and Partin, A. W. (2002). Psa levels and the probability of prostate cancer on biopsy, *European Urology Supplements* **1**(6): 21–27.

Hanspers, K., Kutmon, M., Coort, S. L., Digles, D., Dupuis, L. J., Ehrhart, F., Hu, F., Lopes, E. N., Martens, M., Pham, N. and et al. (2021). Ten simple rules for creating reusable pathway models for computational analysis and visualization, *PLOS Computational Biology* **17**(8).

Hao, J., Kim, Y., Mallavarapu, T., Oh, J. H. and Kang, M. (2019). Interpretable deep neural network for cancer survival analysis by integrating genomic and clinical data, *BMC Medical Genomics* **12**(10): 189.
URL: *https://doi.org/10.1186/s12920-019-0624-2*

Hastie, T., Tibshirani, R., Narasimhan, B. and Chu, G. (2024). *impute: impute: Imputation for microarray data*. R package version 1.80.0.
URL: *https://bioconductor.org/packages/impute*

Hornung, R., Nalenz, M., Schneider, L., Bender, A., Bothmann, L., Bischl, B., Augustin, T. and Boulesteix, A.-L. (2023). Evaluating machine learning models in non-standard settings: An overview and new findings.
URL: *https://arxiv.org/abs/2310.15108*

In Junyong, L. D. K. (2019). Survival analysis: part ii – applied clinical data analysis, *Korean J Anesthesiol* **72**(5): 441–457.
URL: *http://ekja.org/journal/view.php?number=8532*

Ishwaran, H. and Kogalur, U. (2007). Random survival forests for r, *R News* **7**(2): 25–31.
URL: *https://CRAN.R-project.org/doc/Rnews/*

Ishwaran, H., Kogalur, U. B., Blackstone, E. H. and Lauer, M. S. (2008). Random survival forests, *The Annals of Applied Statistics* **2**(3).
URL: *http://dx.doi.org/10.1214/08-AOAS169*

Ishwaran, H., Lauer, M. S., Blackstone, E. H., Lu, M. and Kogalur, U. B. (2021). randomForestSRC: random survival forests vignette, `http://randomforestsrc.org/articles/survival.html`. [accessed date].
URL: *http://randomforestsrc.org/articles/survival.html*

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T. and Kluger, Y. (2018). Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network, *BMC medical research methodology* **18**: 1–12.

Klau, S., Hornung, R., Bauer, A. and Hagenberg, J. (2019). *prioritylasso: Analyzing Multiple Omics Data with an Offset Approach*. R package version 0.3.2, commit 478e853b15af2d22a071ebe895835c898dc6c59c.
URL: *https://github.com/laefrost/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): 322.
URL: *https://doi.org/10.1186/s12859-018-2344-6*

Knipper, S., Pecoraro, A., Palumbo, C., Rosiello, G., Luzzago, S., Deuker, M., Tian, Z., Shariat, S. F., Saad, F., Tilki, D. et al. (2020). The effect of age on cancer-specific mortality in patients with prostate cancer: a population-based study across all stages, *Cancer Causes & Control* **31**: 283–290.

Kreuz, M., Otto, D. J., Fuessel, S., Blumert, C., Bertram, C., Bartsch, S., Loeffler, D., Puppel, S.-H., Rade, M., Buschmann, T. et al. (2020). Prostatrend—a multivariable prognostic rna expression score for aggressive prostate cancer, *European Urology* **78**(3): 452–459.

Lin, D. Y. (2007). On the breslow estimator, *Lifetime Data Analysis* .

Longato, E., Vettoretti, M. and Di Camillo, B. (2020). A practical perspective on the concordance index for the evaluation and selection of prognostic time-to-event models, *Journal of biomedical informatics* **108**: 103496.

Makrodimitris, S., Pronk, B., Abdelaal, T. and Reinders, M. (2023). An in-depth comparison of linear and non-linear joint embedding methods for bulk and single-cell multi-omics, *Briefings in Bioinformatics* **25**(1): bbad416.
URL: *https://doi.org/10.1093/bib/bbad416*

Michelucci, U. (2022). An introduction to autoencoders, *arXiv preprint arXiv:2201.03898* .

Otero-Carrasco, B., Ugarte Carro, E., Prieto-Santamaría, L., Diaz Uzquiano, M., Caraça-Valente Hernández, J. P. and Rodríguez-González, A. (2024). Identifying patterns to uncover the importance of biological pathways on known drug repurposing scenarios, *BMC Genomics* **25**(1): 43.
URL: *https://doi.org/10.1186/s12864-023-09913-1*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library, *Advances in Neural Information Processing Systems* **32**.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**: 2825–2830.

Pölsterl, S. (2020). scikit-survival: A library for time-to-event analysis built on top of scikit-learn, *Journal of Machine Learning Research* **21**(212): 1–6.
URL: *http://jmlr.org/papers/v21/20-729.html*

Pölsterl, S. (n.d.). Gradient boosted models.
URL: *https://scikit-survival.readthedocs.io/en/stable/user$_g$uide/boosting.html*

Rade, M., Kreuz, M., Borkowetz, A., Sommer, U., Blumert, C., Füssel, S., Bertram, C., Löffler, D., Otto, D. J., Wöller, L. A. et al. (2024). A reliable transcriptomic risk-score applicable to formalin-fixed paraffin-embedded biopsies improves outcome prediction in localized prostate cancer, *Molecular Medicine* **30**(1): 19.

Scholz, M., Fraunholz, M. and Selbig, J. (2008). Nonlinear principal component analysis: neural network models and applications, *Principal manifolds for data visualization and dimension reduction*, Springer, pp. 44–67.

Shtivelman, E., Beer, T. M. and Evans, C. P. (2014). Molecular pathways and targets in prostate cancer, *Oncotarget* **5**(17): 7217–7259.
URL: *https://www.oncotarget.com/article/2406/*

Simon, N., Friedman, J., Tibshirani, R. and Hastie, T. (2011). Regularization paths for cox's proportional hazards model via coordinate descent, *Journal of Statistical Software* **39**(5): 1–13.

Taunk, K., De, S., Verma, S. and Swetapadma, A. (2019). A brief review of nearest neighbor algorithm for learning and classification, *2019 international conference on intelligent computing and control systems (ICCS)*, IEEE, pp. 1255–1260.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D. and Altman, R. B. (2001). Missing value estimation methods for dna microarrays, *Bioinformatics* **17**(6): 520–525.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net, *Journal of the Royal Statistical Society Series B: Statistical Methodology* **67**(2): 301–320.
URL: *https://doi.org/10.1111/j.1467-9868.2005.00503.x*