Norwegian University
of Life Sciences

Master's Thesis 2023 30 ECTS
Faculty of Science and Technology

Comparison of Pre-processing Methods and Various Machine Learning Models for Survival Analysis on Cancer Data

Haris Karovic
Data Science

Preface

This thesis marks the end of my 8 years at the Norwegian University of Life Sciences, and my 5 year study in Data Science. I am immensely grateful for the knowledge, skills and friends I have acquired during my study. I offer my sincerest gratitude to my supervisor Professor Oliver Tomic and co-supervisor Professor Cecilia Maria Futsæther, both for the inspiring courses held at the university and for the immense help and fruitful discussions about the research conducted in this thesis.
Furthermore I would like to thank my parents and sister for always believing in me and supporting me during my studies and in life. Many thanks goes towards my girlfriend who has supported me a lot and whom I have had little time for the last 6 months. Additionally I would like to personally thank Dr Kroepelien and his professional team at Volvat Storo for reacting quickly and planning the surgery I had to undergo at the start of this project at a suitable time. You managed to fix me up well just in time, so I would be able to write my thesis this semester.

Haris Karovic
Ås, June 15, 2023

[This page is intentionally left blank]

Abstract

Colorectal cancer and cancers in the head and neck region still pose a big problem in medicine and in the healthcare sector. In 2021 alone 11 121 deaths could be accounted for due to various cancers [1], with colorectal and head and neck cancer being among the more common types [2] [3]. In today's digital age, hospitals and researchers are collecting more data than ever before. Many studies have patients where the follow-up or study has ended before an event of interest occurs. Instead of discarding those patients from observed data when applying machine learning methods and subsequently losing valuable information, survival analysis can be applied. Survival analysis utilizes the information from the censoring variable that tells whether or not the event of interest has taken place before the study has ended.

In this thesis several pre-processing techniques were utilized, such as removal of outliers, feature distribution transformations and feature selection techniques. These techniques were applied together with multiple machine learning algorithms from the scikit-learn and scikit-survival library. The survival algorithms used were Regularized Cox model with elastic net (Coxnet), random survival forest, tree based gradient boosting and gradient boosting with partial least squares as base learner. These algortihms take into account the information from the censoring variable in addition to the survival time. Other machine learning algorithms used were linear regression, ridge regression and Partial least squares regression (PLSR), where the last three algorithms only use the survival time as the target and do not account for the censoring variable. Two datasets were used in this thesis, one with patients diagnosed with colorectal cancer, and the second with patients diagnosed with various head and neck cancers. Furthermore, two experiments were carried out separately and validated by the use of repeated stratified k-fold cross validation. In the first experiment the models were fitted to different feature transformations of the datasets in combination with feature selection techniques. The second experiment involved hyperparameter tuning for the survival models. There was little difference in performance between the transformations, with no improvement on the head and neck dataset, however for the high dimensional colorectal cancer dataset, powertransformation led to a very small increase of 0.02 in the concordance index. The feature selection techniques did improve the performance of four of the models, which were Linear Regression, Ridge Regression, PLSR and Coxnet. For the more advanced survival models which were Gradient Boosted and Random Survival Forest, the feature selection did in general not improve metrics, as they might have benefited from greedily selecting features and updating feature weights on their own. The best model in the first experiment for OxyTarget was Random Forest with powertransform applied before, and all features available. This resulted in a concordance index of 0.83. For the head and neck dataset both Component Wise gradient boosting, Coxnet and PLSR were able to achieve the highest concordance index with 0.77, with Coxnet able to achieve that score across all three transformations.

In the second experiment, all the survival models were tuned for different hyperparameters to see if the various metrics would improve. A small performance increase could be seen for several models. However, for the dataset with colorectal cancer, a Coxnet model tuned with a low regularization strength and low l1_ratio penalty yielded a large increase in the concordance index and resulted in the best model with a score of 0.827. For the head and neck dataset, parameter tuning the Random Survival Forest algorithm for min_weight_fraction_leaf and max_depth resulted in the best model, and a concordance of 0.787 was achieved. The research and the framework created to conduct the aforementioned experiments show that more promising ranking results while maintaining robust models can be achieved through the use of pre-processing techniques and through the utilization of all data using repeated stratified k-fold cross validation. However, as the research conducted shows, there is no universal best algorithm or method to conduct survival analysis for cancer data, as it depends on the data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Cancer is a complex disease that continues to pose a significant health challenge globally, as it is one of the most common sources leading to death. It can be characterized as a disease where abnormal cells have an uncontrolled growth due to genetic mutations. These mutations can form malignant tumors and spread to surrounding areas, potentially causing harm to tissue and organs [4].
Only in Norway, there were 327 101 humans which have had a cancer diagnosis or are living with cancer in 2022, and 11 121 deaths in 2021 alone are due to cancer [1]. On a global level near 10 millions deaths in 2020 could be accounted to cancer, where colorectal cancer is one of the most common types [2]. There are various risk factors associated with rectal cancer, such as medical history of the family, certain gene changes, personal medical history of ulcerative colitis or Crohns for prolonged time and alcohol and tobacco usage [5].
Cancer in the head and neck region account for around 325 000 deaths globally and is the seventh most prevalent form of cancer with the numbers of incidents increasing within the last decades [3]. There are various risk factors that cause cancers in the head and neck region. Radiation exposure, occupational exposure to certain materials, HPV infection, Epstein Barr virus, and alcohol and tobacco consumption being the most important risk factors [6].
There are various treatments for rectal, head and neck cancer, but the most common types are often a form of surgery or therapy including medical treatments such as radiotheraphy, chemotherapy, targeted therapy and immunotherapy. Some patients may undergo more than one therapy, for example using a therapy method after surgery to make sure all remaining cancer cells are removed if further or more extensive surgery can pose a great risk to the patient.
As an example, survival for patients diagnosed with rectal cancer has increased from 71.4 to 71.8 percent and 72.3 to 73.4 percent for males and females, respectively [1]. A great effort is done globally to understand, counteract and improve the treatment of cancer. Hospitals, the government, medical staff and researchers have put a tremendous effort into capturing, curating and supplying clinical data to study various cancer forms. With the rise of technology for the last two decades, artificial intelligence and machine learning can be utilized to efficiently analyze clinical data and forecast future outcomes. The use of machine learning models in survival analysis, can help us get a better understanding of risk factors, treatment types, clinical recordings and potential outcomes and hopefully lead to an increase in survival for multiple forms of cancer.
Both datasets investigated in this thesis have never been analyzed with Survival Analysis before, therefore it is in itself of interest to apply survival analysis to these data because it takes into account that parts of the data are censored. Previous studies have focused on whether an event occurs or not after a certain time period through classification, not when.

1.2 Aims of the master thesis

The primary focus of this thesis will be a comparison of performance between machine learning algorithms used for survival analysis, where several widely used metrics for survival analysis will be used. Additionally, conventional linear regression models which are not able to capture information regarding censoring will be used to estimate time of survival, where ranking of samples will be evaluated towards the survival models and accuracy will be evaluated between the conventional regression models. Can modern machine learning algorithms achieve better performance than classical survival models and classical regression models? Additionally, as one performance metric only captures one side of performance, are we able to get a more comprehensive picture of the performance of each model by assessing several performance metrics in survival analysis?
A secondary objective will be to assess and evaluate the effect different feature selection methods together with different scaling and transformation methods of the data have on the selected models. As two different datasets will be used, performance on high-dimensional vs regular-sized data will be assessed to some extent.
Chapter 2 will explain some theory revolving around regression, survival analysis, the models used and their methods of predicting the time of survival. Chapter 3 will give an introduction to the datasets used for the experiments. Chapter 4 will present the methodology used before the results are presented in Chapter 5. The results will be discussed in Chapter 6, while Chapter 7 will contain the conclusion and further work that could have been done.

1.3 Objectives

Hopefully, by identifying which models and techniques are able to perform good by assessing the resulting metrics, several improvements can be made in the future regarding cancer treatment. By benchmarking different machine learning models and regression techniques, researchers can determine which models provide the most accurate predictions for cancer survival rates, to effectively predict patient outcomes and guide towards the right treatment options. With more reliable estimates hopefully in the future, clinicians can use modern data science to better plan and personalize treatments effectively. Survival and regression models can be used further to analyze recurrence rates and progression as well. Accurate models and feature selection techniques can assist in identifying important features and their impact on survival, and potentially help researchers understand underlying biological mechanisms that impact the outcome of patients. Hopefully, new biomarkers can be discovered and therapeutic targets for effectively improving cancer treatment. Hopefully, this thesis can facilitate better interpretability of some machine learning models and by sharing the benchmarking results and methods can lead to further collaboration and advancements in cancer treatment. In today's age where many governmental healthcare providers are facing an overload of work, machine learning can hopefully efficiently help patients get help sooner in the future.

1.4 Related Work

As one of the clinical datasets used in this thesis, OxyTarget might be classified as high dimensional data, a key idea was to explore if feature selection might lead to better performing survival models and conventional regression models on survival data. A great inspiration was the article "A comparison of machine learning methods for survival analysis of high-dimensional clinical data for dementia prediction" by Spooner, et al [7]. This thesis will use some of the same models and feature selection models implemented in this article.

Chapter 2

Theory

2.1 Survival analysis

Survival analysis is a statistical method applied in multiple fields such as healtchare, engineering, economics, where the goal is to analyze time-to-event data. The primary focus of survival analysis on time-to-event data is the study of when specific events occur [8]. For example, patients are studied from when they are diagnosed with a disease until their death. Data from such a study will usually contain the time of survival. The key difference which is essential for survival analysis is censoring. Censoring can for example occur when the event of interest yet has to happen by the end of the study or simply because the sample or patient is lost to follow-up [9]. Survival models are able to use the information of these censored samples. Through the use of statistical survival models and survival machine learning algorithms the probability of an event happening over time can be estimated, such as a patient dying, a mechanical part breaking down, or the time until a loan is paid back.
Survival analysis has been an important statistical method in cancer studies for several decades. Some methods can be tracked back to the 50's, such as the Kaplan-Meier estimator which was developed by Kaplan and Meier [10] and the infamous Cox Proportional Hazards Model developed in 1972 by D.R. Cox [11]. Around the millennia and up until today new methods have emerged with machine learning algorithms that have been adapted for use in survival analysis. Both nonlinear tree based models and new linear models have been adapted, such as random survival forest [12] and gradient boosted models [13].

2.1.1 Censoring

One of the key differences between survival analysis from conventional regression is the event indicator. The target values for the model to train on consists of both the length of survival and the event indicator, telling whether an event has occurred or not. In survival analysis a crucial analytical element which has to be considered is censoring, which is when there is no specific known time to event for a sample, illustrated in figure 2.1.

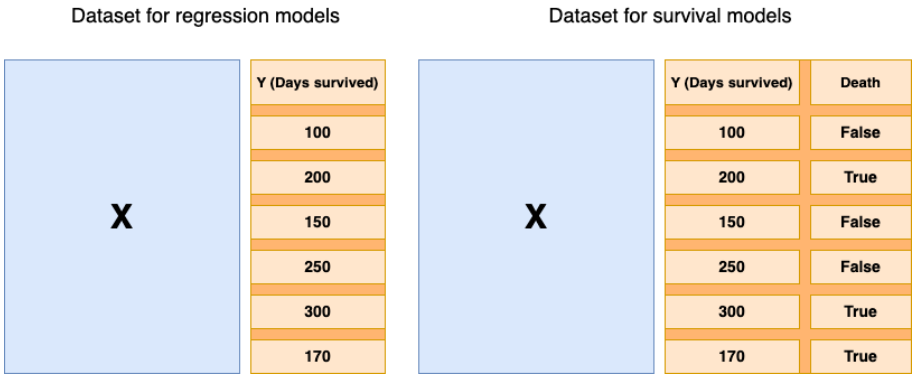


Figure 2.1: Dataset examples with and without censoring variable

This thesis will utilize data which is right censored, meaning the data contains samples which had not yet experienced an event when the data observation period ended, but will experience it afterwards. In general there are three main types of censoring, which are left censoring, interval censoring and right censoring. Each censoring type is not exclusive to each sample either, a sample can for example be both left and right censored. Typically censoring occurs because a sample has yet to experience the event before the observational period for the study ends, is lost to follow up or simply withdraws from the study [9].

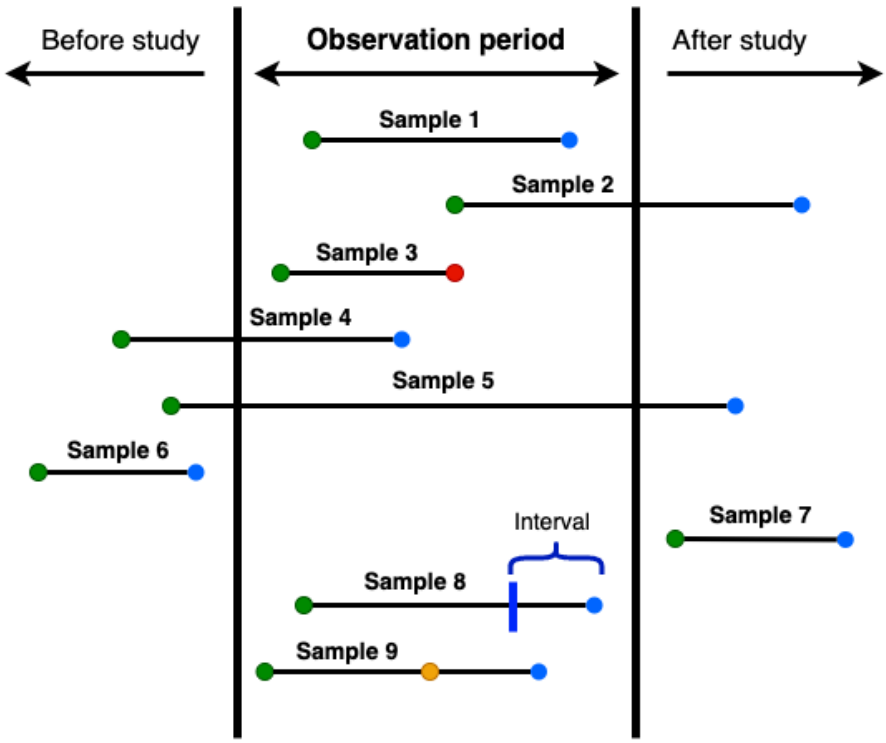


Figure 2.2: Examples of different censoring types for a number of samples. Green illustrates diagnosis, blue illustrates event and red illustrates withdrawal from study.

Figure 2.2 illustrates a variety of different occurrences of censoring for nine different samples. As this thesis revolves around healthcare data, we can explain the figure using patients diagnosed with a medical condition and let death be the event indicator. Sample 1 is in fact not censored, both the medical condition was diagnosed and the patient died within the observation period.

Sample 2 was diagnosed within the observational period and experienced the event after the observation period, therefore the patient is right censored. Right censoring as illustrated with sample 2 is the most common form of censoring for a lot of popular and conventional datasets available to the public wanting to learn survival analysis. Sample 3 is right censored, as the patient has withdrawn from the study, consequently leading the sample to be right censored at the time of withdrawal. Sample 4 is left censored which means the time of origin for the medical condition is unknown, it is only known that it happened sometime before the study started. Interestingly for sample 5, the time of origin for the medical condition is unknown as it happened before the observational period and the patient was alive during the full length of the observation period, leading to the sample being both left and right censored. Sample 6 is fully left censored, while sample 7 is fully right censored. Fully left censored implies that the time of origin is unknown and both origin and event take place before the observations have started. Similarly, for a fully right censored sample, the time of origin and event takes place after the observation period has ended. Fully right and left censored samples will not impact or have any effect on analysis when conducted, but it might have a negative impact on how well a model will generalize [14]. Sample 8 is interval censored, a typical example is when it is known a patient has endured an event in the observational period, but the exact time is not known. For example, it is known that the patient died between the last checkup at the hospital and when the patient was found dead, which is a time interval. The interval for sample 8 is the timeframe between the blue line and the blue dot. Sample 9 is commonly referred to as a double interval censoring. As the figure illustrates, there is an orange dot and a blue dot highlighting two different events. The final event, which is death is the blue dot and has occurred as a possible effect of the initial event which is the orange dot.

Censoring assumptions

Additionally, there are three main assumptions which have to be taken into consideration about censoring. Censoring is independent, non informative and random.
Basic notation needs to be introduced, where T is a random variable for the survival time for a sample and "t = specific value for T [9]" To give an example, assessing five year survival can noted as: $T > t = 5$ [9]. Random censoring implies that all samples censored at time t should be representative of all samples still at risk at time t within the same risk set. Furthermore, this implies that the failure rates for both a sample censored at time t and a sample still at risk at time t are equal within the same risk set [9]. A risk set is a group of samples which have survived for at least a specific value of time $t_{(f)}$. For example considering a study of 10 patients, and the risk set containing 5 patients that have survived for at least 4 weeks. Considering this risk set, a patient from this subgroup is censored after 11 weeks of survival. At time 11 weeks, the failure rate for this patient is equal to the ones still alive in the risk set.
The second assumption, Independent censoring, is in essence that censoring and likelihood of experiencing the event of interest are independent [9]. To give an example, a study containing a group of 20 patients diagnosed with cancer studying the rate of survival after 1 year can be considered. After 1 year 10 of the patients are still alive. The researchers want to continue the study for another year to study the rate of survival at 2 years, but 5 of the samples have withdrawn. Of the 10 patients in the risk set that has survived for 1 year, we can divide the withdrawn patients into subgroup A and the 5 patients still a part of the study into subgroup B. After 2 years when the study has ended, of the 5 patients remaining in the study, 4 survived. Assuming censoring is independent and random, the researchers estimate that 4 out of 5 patients in subgroup A also survived after 2 years time. To further add to the first assumption, even if the patients did not withdraw at random from the risk set of patients alive after 1 year, the survival rate can be expected to be the same for subgroups A and B as if the patients did withdraw at random from the study at year 1 ($t_{(1)}$).
The third assumption regarding censoring is non-informative censoring. There are two depen-

dencies that have to be taken into account before it is determined whether the censoring is informative or non-informative. The dependencies are the distribution of time-to-event and time-to-censorship [9].
When the distribution of survival times for the data reveals no information about the distribution of times of censoring, then censoring is non-informative. Opposite, if the distribution of survival times reveals information about the distribution of times of censoring, then censoring is informative. Simplified, this means that non-informative censoring is when the probability of a sample being censored is not related to the samples survival time, after taking into account other observed features [15]. The importance of this assumption is highlighted in the fact that informative censoring can lead to a bias in the estimation of the hazard and survival of samples [16]. To ensure non-informative censoring for the data analyzed, the risk can be mitigated by including all relevant features in the analysis during observation which can be associated with either censoring or the event of interest.

2.1.2 Hazard

The hazard function is a fundamental concept in survival analysis which is known as the conditional failure rate and "gives the instantaneous potential per unit time for the event to occur, given that the individual has survived up to time t" [9]. The function is very useful in survival analysis as it captures the dynamic and instantaneous nature of an event of interest occurring over time. Hazard can as well help with explaining the relationship between covariates in the dataset in relation to the survival time. A comprehensive understanding of the time-to-event outcomes for subgroups and samples can be provided by examining the shape of the hazard function. Further on, risk factors which influence survival can be studied, and the impact treatments have becomes quantifiable. More simply Hazard can be said to be the intensity of failure at a given time, per unit time and can be compared to the speedometer of a train. The train can speed up and can speed down, but if the same speed is kept, it tells how far the train will travel. Essentially what the speedometer tells is how fast the train is going at any given time, which is the instantaneous potential, which is similar to the hazard function which tells the instantaneous event rate. The hazard function is expressed as [9]:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t < T \leq t + \Delta t | T \geq t)}{\Delta t} \tag{2.1}$$

As can be seen Equation 2.1, it is a limit of the probability of an event taking place within a time interval as the interval approaches 0, divided by the length of the time interval. The limit operator provides the instantaneous potential, as the change in time approaches 0 [9]. The numerator in the fraction is the probability of an event occurring in the time interval $[t, t + \Delta t]$, given that the event has not taken place before time t. Therefore as mentioned the hazard is a conditional failure rate. What is important to note, is that the hazard function can never be negative. To give an example, clinical data containing cancer recurrence can be considered. In turn defining the hazard function as the instantaneous recurrence of cancer at a given point in time. As known, cancer recurrence can never be negative, which means that in theory the hazard function can go to infinity.

2.1.3 Survival function

The hazard function is closely related to the survival function, which is crucial for survival analysis. The survival function can be obtained with the hazard function, and vice versa. The most simple way of expressing the survival function is in the form of [9]:

$$S(t) = P(T > t) \tag{2.2}$$

Where T is a continuous random variable representing the time-to-event [9], hence the expression can be described as the probability that the event of interest has yet to occur by time t. Considering a clinical dataset containing cancer patients, to asses the probability of survival by year 3, the expression can be written as $S(t) = P(T > 3)$. As the time variable goes from 0 to infinity, the probability goes from 1 to 0. To further explain the relationship between the hazard function and survival function, the survival function can be obtained by integrating the hazard function in Equation 2.1 from time 0 to t, which is shown in equation 2.3 [9].

$$S(t) = exp[-\int_0^t h(x)\,dx \tag{2.3}$$

The probability of that sample surviving beyond time t is the product of all the probabilities of surviving all the smaller time intervals given by Equation 2.3. The integral of the hazard function from time 0 to t is commonly known as the cumulative hazard function [9], which essentially represents the total risk of experiencing the event of interest before time t. The survival function is the exponential of the negative cumulative hazard function.

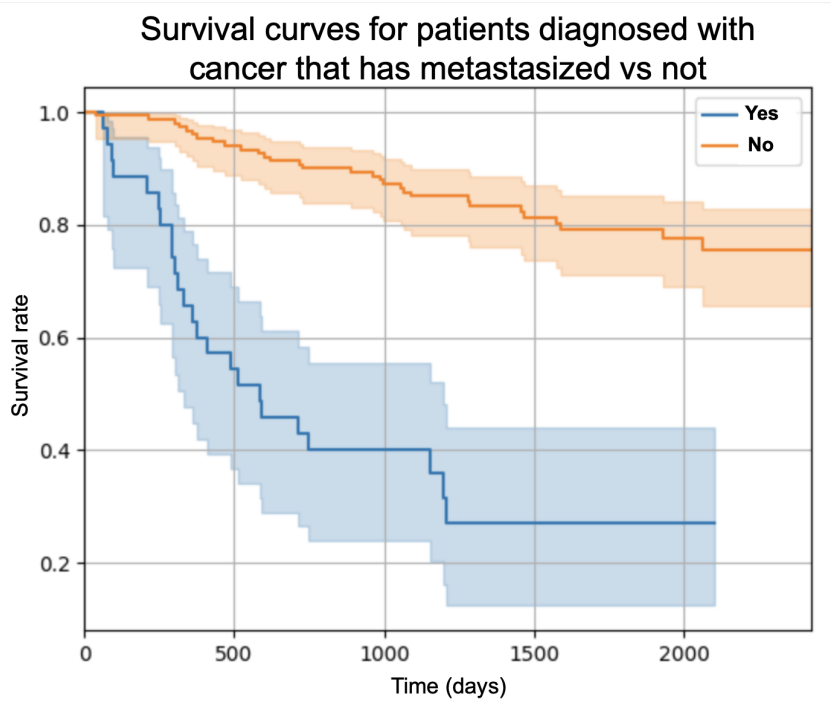


Figure 2.3: Example of survival curve

2.1.4 Cox proportional hazards

In a paper from 1972 David Cox suggested a new hazards model which is widely used for survival analysis [11]. An assumption made by the model is that the hazard function can be defined as the product of a time independent baseline hazard function $H_0(t)$, and a set of independent time varying covariates X [11]:

$$h(t|X) = h_0(t)e^{(\beta X)} \tag{2.4}$$

X is a vector consisting of multiple covariates $X_1, ..., X_n$. The baseline hazard function $H_0(t)$ represents the hazard at time t when all of the covariates in vector X equals 0. Whereas the beta β represents the effect of every covariate on the hazard function. In machine learning language it is the same as what is referred to as feature weights. The dot product of X and

regression coefficients beta is the total effect of all the covariates on the hazard function. As the last term containing the dot product is exponential the hazard function is positive at all times. Simplified, e^{β} can be considered the relative risk of the covariate on the hazard, for example an indication of "the relative risk of adverse event given by smoking over not smoking [17]" for a patient.
When it comes to applied data science, one has to figure out how to fit a model with the correct parameters. In order to estimate the correct regression coefficients beta, Cox´s proposition was to fit the model using maximum likelihood estimation. This is done by minimizing the negative log partial likelihood function. As can be seen from equation 2.5, the partial likelihood can be written as the product of the conditional probabilities of observed events given they have occurred and the probabilities of censoring for samples that have yet to experience said event.

$$L(\beta) = \prod_{i=1}^{n} \frac{\exp(\beta^T X_i)}{\sum_{j \in R(t_i)} \exp(\beta^T X_j)} \tag{2.5}$$

[11] [18]
In equation 2.5, n is the number of samples. X_i is the covariate vector for sample i, $R(t_i)$ is the individuals at risk at time t_i [18]. Further, using the partial likelihood function it is possible to derive the negative log partial likelihood 2.6, which is the objective function needed to optimize.

$$-l(\beta) = -\sum_{i=1}^{n} \delta_i \left(\beta^T X_i - \log \big(\sum_{j \in R_i} \exp(\beta^T X_j) \big) \right) \tag{2.6}$$

[11] [18]
For the negative log likelihood function 2.6 sigma (σ) indicates whether and event has taken place or not. Negative log partial likelihood is then used to find the values for the regression coefficients beta that minimizes the negative log partial likelihood. As the function is convex, gradient based methods can be used in order to find the correct values for the coefficient vector beta that minimizes the negative log likelihood [19].

2.2 Outlier detection

2.2.1 Z-Score

Z-score outlier detection is a statistical method which can be used to identify outliers in the dataset. The Z-score for a data point tells how many standard deviations the data point is away from the mean [20].
To calculate the Z-score, the mean and the standard deviation for the features have to be calculated. Next the Z-score for each sample has to be calculated with the formula in equation 2.7 which will tell the distance from the mean for the sample [20].

$$Z_i = \frac{|X - \mu|}{\sigma} \tag{2.7}$$

Further, a threshold value defining how far away from the mean an outlier is has to be specified. For example 3 or 4 standard deviations away from the mean. Then each sample which is further away than the threshold will be identified.

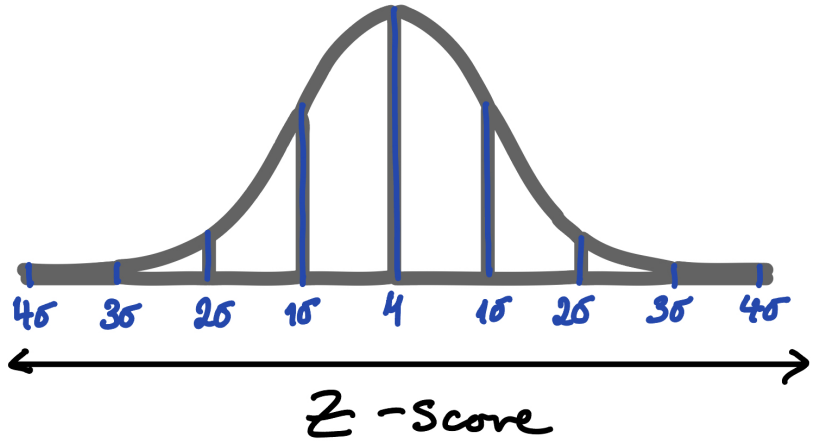


Figure 2.4: Example with Z-scores on a bell curve

Outlier detection with Z-score assumes the data is normally distributed [20]. This thesis uses Z-score outlier detection to remove samples with extreme values for continuous explanatory variables.

2.2.2 Isolation Forest

Isolation forest is an algorithm that detects anomalies by isolating the outlier samples. It works in a similar fashion to random forest and consist of an ensemble of isolations trees (iTrees) [21]. The iTrees randomly selects a feature for each split and selects a random value between the maximum and minimum observed value to make a split [22]. On the ensemble of the iTrees the anomalies are the samples which have a short average path length on the trees [21].

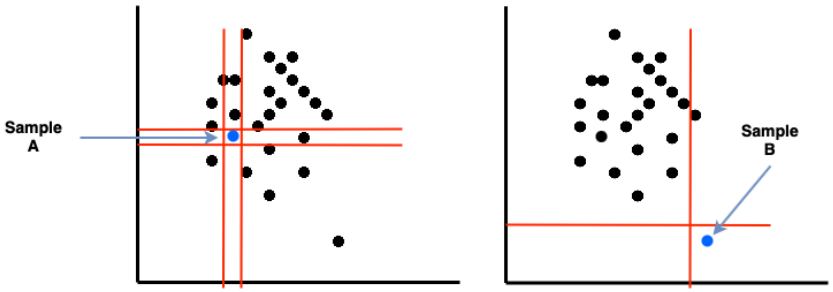


Figure 2.5: Example showing isolation of two samples, for an isolation tree.

As can be seen in Figure 2.5 the path length for a particular iTree is greater for sample A than sample B, which in this case is an outlier. As can be seen, outliers like sample B are more susceptible to become isolated as it requires fewer partitions to become isolated [21].

2.3 Feature selection

As data collection techniques have improved vastly over the last decades due to digitalization, more features are recorded and high dimensional data has gotten more accessible. Berkeley Statistics defines datasets used in high dimensional statistics as when "the number of features is of comparable size or larger than the number of observations" [23]. There are certain risks associated with working with high dimensional data in regards to predictive performance. To

counteract some of the risks associated with a large number of features, feature selection can be utilized. There are multiple possible benefits of reducing the feature space with feature selection. A smaller feature space can lead to noise reduction as high dimensional data can contain irrelevant or noisy features, which might reduce the overall performance of the analysis. Another factor to consider by including irrelevant or noisy features is the risk of poor generalization of the model on new data, as irrelevant or noisy features can lead to overfitting. An added benefit with a reduced feature space is also improved interpretability as it can be easier for the models to interpret relationships between the features in the subset.

2.3.1 Univariate Cox filter

The univariate Cox filter uses the Cox proportional hazards model to select features. By turn each feature is fitted to the model [24] and the resulting performance from the models is measured using Harrell's concordance index. The features are then ranked based on the concordance index [7]. The idea is to explore how well each feature univariately explains the survival outcome, and the better the score, the better the feature is at predicting the outcome. After the features are ranked, the user specifies how many of the top performing covariates to keep as a subset.

2.3.2 Random forest variable importance

One of the filter methods used to select features is the random forest variable importance. The features are selected by randomly permuting features, which in essence means that noise is added to a random feature. The error in prediction is measured before and after the noise is added [25]. If a noised up feature leads to a significant increase in prediction error, it might indicate that a feature is highly predictive.

2.3.3 RSF minimal depth

The feature selection method RSF minimal depth is in fact the minimal depth of a maximal subtree in a survival tree [26]. A maximal subtree for a variable is the greatest subtree where the variable was used for splitting at the root node. In other words the parent nodes of a maximal subtree for a variable is not split using the particular variable [26].

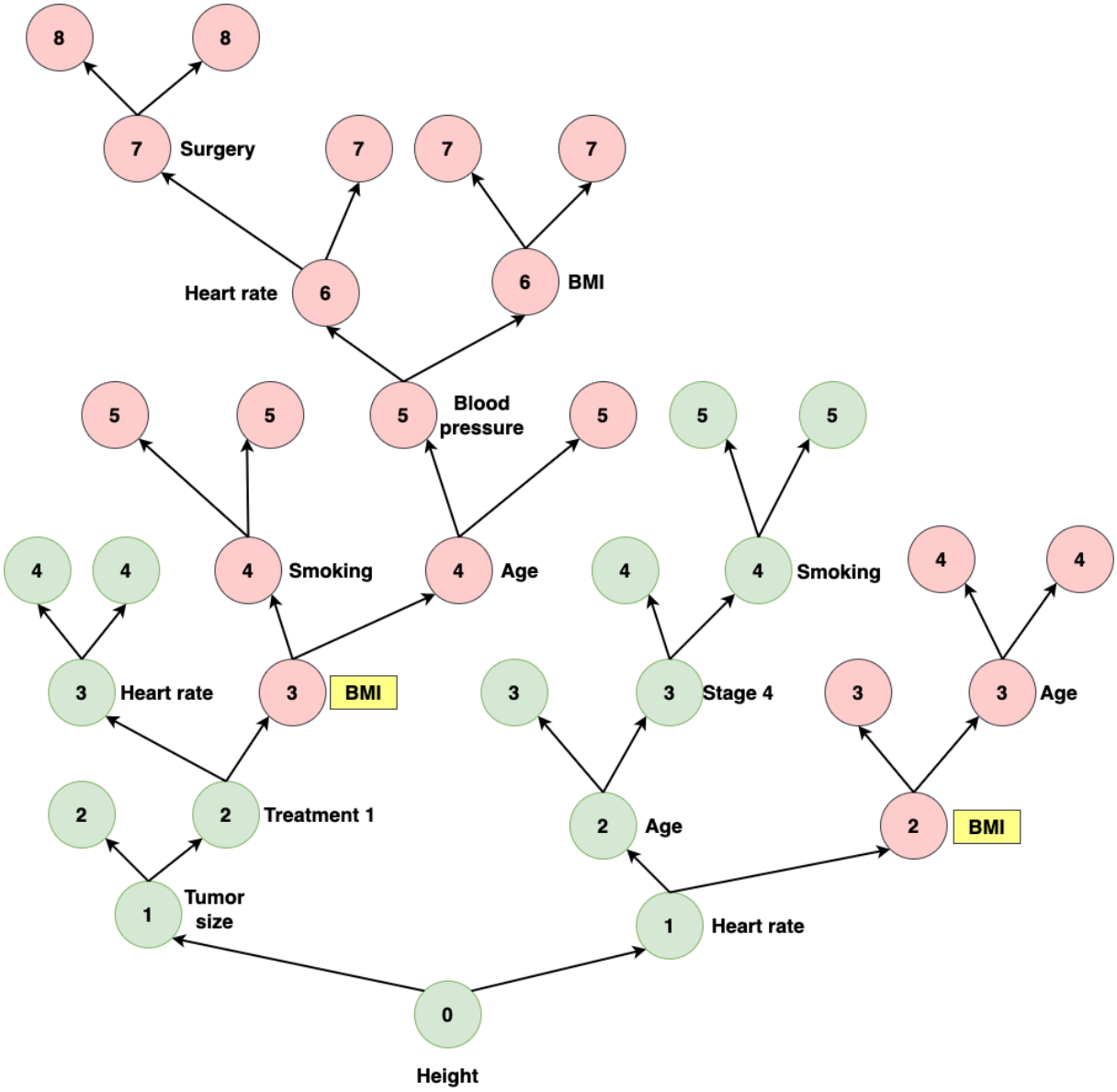


Figure 2.6: Survival tree highlighting the maximal subtrees for the variable BMI and its depths.

As can be seen from the illustration 2.6, a variable can have more than one maximal subtree (subtrees marked in red colour). To find the minimal depth of the maximal subtree for a particular variable, one has to start looking at the maximal subtrees root nodes. The node distance from the maximal subtree(s) root node(s) to the root node of the survival tree is the depth of the maximal subtrees [26]. To find the minimal depth for a variable, one has to look at the aforementioned distance and find the maximal subtree with the smallest distance from the maximal subtree root node to the root node of the survival tree [26]. In Figure 2.6 the feature BMI has two maximal subtrees with a depth of 3 and 2, therefore the minimal depth for BMI is 2, whereas for the feature height, the minimal depth of its maximal subtree is 0, as the maximal subtree is the whole survival tree. The minimal depth will give an indication of how great the impact a feature has on prediction. In general, the smaller the minimal depth is, the more considerable the effect the feature has on predictions [26].

2.3.4 RSF variable hunting

Algorithm 1 RSF-VH Algorithm

1: **for** $b = 1$ to B **do**
2: Split the data into test and training data sets.
3: Select $P < p$ genes. Call this set of genes $\mathcal{G}_P$.
4: Fit a survival forest, $\mathcal{F}$, to the training data using $\mathcal{G}_P$.
5: Calculate the mean for D_v^* using $\mathcal{F}$. Let $\mathcal{G}$ be the subset of genes from $\mathcal{G}_P$ having minimal depth less than this threshold.
6: Let $\mathcal{V}$ be the joint VIMP for $\mathcal{G}$ from $\mathcal{F}$. Set $\Delta = \mathcal{V}$.
7: **while** $\Delta > 0$ **do**
8: Augment $\mathcal{G}$ to include the next gene in $\mathcal{G}_P$ with smallest minimal depth (if there is no such gene, then $\mathcal{G}$ is unchanged). Call this new set $\mathcal{G}^+$.
9: Let $\mathcal{V}^+$ be the joint VIMP for $\mathcal{G}^+$ from $\mathcal{F}$. Set $\Delta = \mathcal{V}^+ - \mathcal{V}$.
10: **if** $\Delta > 0$ **then**
11: Set $\mathcal{V} = \mathcal{V}^+$; $\mathcal{G} = \mathcal{G}^+$.
12: **end if**
13: **end while**
14: Fit a survival forest $\mathcal{F}^*$ to the training data using $\mathcal{G}$.
15: Calculate the prediction error of $\mathcal{F}^*$ over the test data.
16: **end for**

[26]

Figure 2.7: Random forest variable hunting algorithm

Random forest variable hunting is a regularized algorithm which uses the maximal subtrees of of the random forest algorithm in order to efficiently find the best subset of features [26]. The algorithm starts by randomly splitting the data given as input into a train and test set (line 2). Following the algorithm will subsample a set of features, resulting in a lower dimensionality (line 3). The subsampling of variables is done at random. In Figure 2.7 genes are used instead of features. The subsample will act as an initial model and features are subsequently added in order of minimal depth, meaning the feature with the smallest minimal depth is added to the initial subset first (line 8). The addition of features continues while the added joint variable importance (VIMP) for the nested models is greater than 0 (line 7) [27], resulting in a final set of features. Then this is repeated numerous times where a new initial set is selected at random and features are added until joint variable importance for the models reaches 0 or under (line 12). When the loop has finished the most important features are determined based on how many times they appear in the final subset in the loop.

2.3.5 mRMR

mRMR is a feature selection algorithm, which stands for minimum Redundancy Maximum Relevance [28]. It is a highly efficient algorithm where the goal is to remove the most redundant features and reduce the dimensionality to find the smallest set of relevant features [28]. The aim of the relevance criterion is to measure how much information a feature can provide about the target. Features with high correlation to the target are selected through the use of mutual information which measures the information given by a feature for a certain target [28]. The redundancy criterion is an evaluation of collinearity between features, which is overlapping information. Redundancy is calculated using mutual information [28]. The mutual information in this instance is simply how much one feature tells about the other feature. The goal is to select features with minimum redundancy, hence telling the least about each other. The mRMR algorithms score each feature by taking the difference between the average redundancy and relevance, and selecting the features with the highest score [28]. This is iteratively done

until the final number of features are reached. To give an example, the mutual information between two categorical features can be calculated using equation 2.8 [28].

$$I(x,y) = \sum_{i,j} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i p(y_j)} \tag{2.8}$$

The redundancy and relevance criterions are then applied as constraints to equation 2.8 to find the minimum redundancy and maximum relevance for a subset. For continuous variables the calculation is slightly different using F-tests [28]. The calculation for continuous variables and mathematical constraints can be found in the original paper "Minimum redundancy feature selection from microarray gene expression data" by Chris Ding and Hanchuan Peng, where mRMR was discovered [28].

2.4 Machine Learning algorithms for survival analysis

Machine learning is a subcategory of artificial intelligence (AI) [29]. Applied machine learning consists of computational self-learning algorithms that can capture complex relationships and patterns in data. Learning from the data, algorithms can build models with the ability to make decisions or predict on new observations. Machine learning can be divided into three main subcategories, supervised learning, unsupervised learning and reinforcement learning [29]. This thesis will use supervised learning which contains labelled data [29]. Supervised learning learns from data inputs where the outcome is known, and will be used to predict the outcome for observations where the outcome label is missing or not known using explanatory variables [30].

2.4.1 Random survival forest

Random Survival Forest builds on the random forest algorithm known from sci-kit learn. Random survival forest is a meta estimator that instead of decision trees utilizes multiple survival trees. The algorithm takes multiple bootstrap samples with replacement and fits a number of survival trees on each subset in order to maximize performance [31].
The goal of each survival tree is to have as pure nodes as possible when performing a split. In order to understand how the splits work, it is necessary to look at the splitting rule used. This thesis will use the default splitting rule, which is the log rank split. The formula for the log rank split can be seen below in formula 2.9.

$$L(x,c) = \frac{\sum_{i=1}^{N}(d_{i,1} - Y_{i,1}\frac{d_i}{Y_i})}{\sum_{i=1}^{N} \frac{Y_{i,1}}{Y_i}(1 - \frac{Y_{i,1}}{Y_i})(\frac{Y_i - d_i}{Y_i - 1})d_i} \tag{2.9}$$

[32]
Within a node there are a number of patients, denoted by l, and their survival times can be denoted by $[t_1 < t_2 < ... < t_N]$. As a result of the split at value c for feature x, the two daughter nodes j=1,2 will contain a number of patients at risk at time t_i, which is represented by $Y_{i,j}$. Following, the number of deaths at time t_i in a daughter node j is represented by $d_{i,j}$. Hence the the two daugther nodes can be expressed as $Y_{i,1} = T_l \geq t_i, x_l \leq c$ and $Y_{i,2} = T_l \geq t_i, x_l > c$ [32]. Considering a feature like "BMI", and setting the split value c to a value of 30, means that patients in the first node will have BMI equal to or less than 30, and the other node will have patients with BMI above 30.
L(x,c) is a measure of node separation, as the value for L(x,c) increases, the purer the node separation is [32]. To further simplify, it can be said that the survival curve at the two given daughter nodes and their respective subsets are compared, the greater the difference is between the two daughter nodes survival curves, the better the separation.

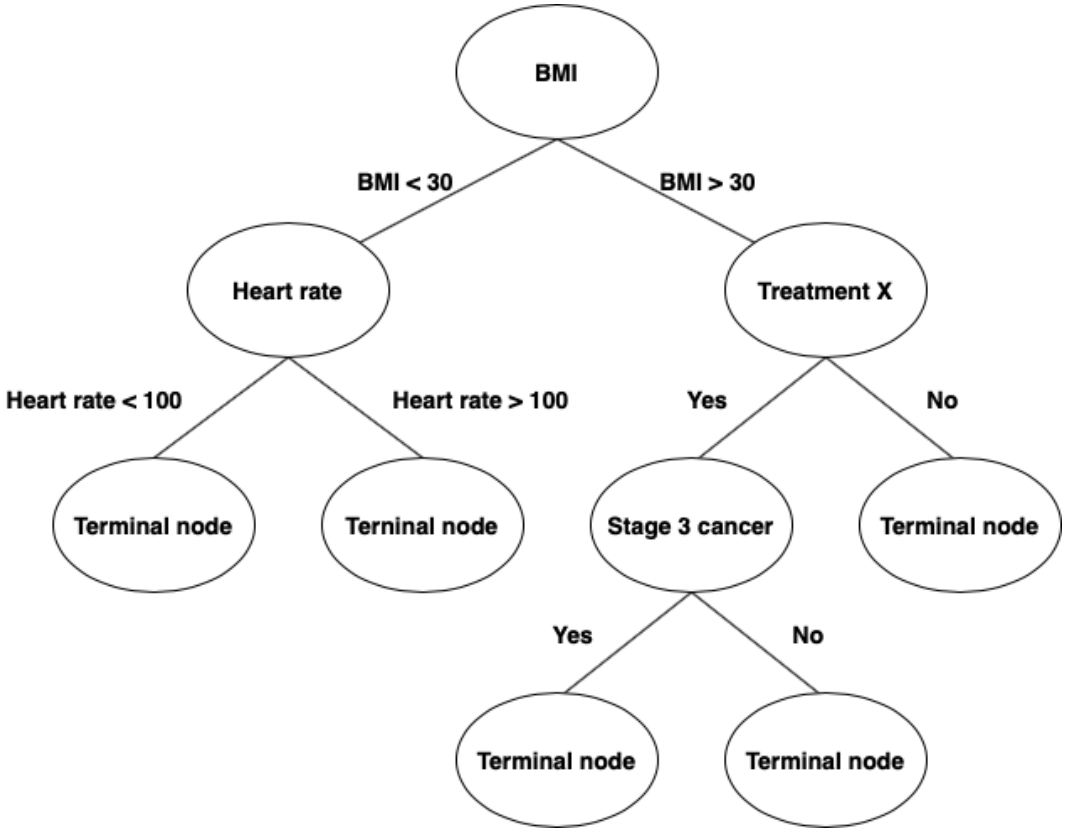


Figure 2.8: Survival tree caption write

For each bootstrap sample drawn a survival tree is grown on the subset. Then recursively at each node, a set number of features are tried for splits using log rank splitting. Of the number of features tried, the feature maximising the difference in survival between the daughter nodes is chosen, hence resulting in as pure nodes as possible. As mentioned, this process is recursively calculated until the tree is grown to full size, which means the terminal node does not contain more unique deaths than the nodesize or the maximum depth is reached. The model will grow a number of survival trees specified by the hyperparameter n_estimators, which has 100 survival trees as default and can be altered by the user. When all the trees are fully grown the model will "calculate an ensemble cumulative hazard estimator [32]" where the information from all the survival trees are assessed together, which results in one estimate for each sample in the original dataset. More specifically each terminal node in a survival tree will have a hazard estimate, which means that each survival tree will provide a sequence of hazard estimates for all its terminal nodes. The cumulative hazard estimate for a specific node (h) can be written as:

$$\hat{H}_h(t) = \sum_{t_{l,h} \leq t}^{\infty} \frac{d_{l,h}}{Y_{l_h}} \tag{2.10}$$

[32]
Similar to when explaining the logrank split, $d_{l,h}$ and $Y_{l,j}$ respectively accounts for events at node h and patients at risk in node h at the time point $t_{l,h}$ when calculating the cumulative hazard estimate at node h. The cumulative hazard function at node h will be the same for all samples at node h, as the goal is to create homogenous groups at the terminal nodes. If we consider a sample, noted as i and the corresponding predictor, noted as x_i, the cumulative hazard estimated at the terminal node for sample i is:

$$\hat{H}(t|x_i) = \hat{H}_h(t), \text{if } x_i \in h \tag{2.11}$$

[32]
This is done by simply dropping the predictor x_i downwards and towards the terminal node. An estimate is calculated for all samples in each tree in equation 2.11. These values will then be used in order to yield the average for the whole ensemble, in other words, the whole ensemble of survival trees. The result is the ensemble cumulative hazard estimate, which is shown in equation 2.12.

$$\hat{H}_e(t|x_i) = \frac{1}{n\_estimators} \sum_{b=1}^{n\_estimators} \hat{H}_b(t|x_i) \tag{2.12}$$

[32]
In equation 2.12, b is the range of number of estimators, also known as the number of survival trees, hence $\hat{H}_b$ stands for the cumulative hazard estimator for tree b. This ensemble cumulative hazard is calculated using the bootstrap subsets, also known as the in bag samples [12]. The last step for the model is to calculate an OOB (out of bag) error rate for the ensemble of estimators. In order for the model to do this, an OOB ensemble cumulative hazard has to be calculated using formula 2.13:

$$\hat{H}_e^*(t|x_i) = \frac{\sum_{b=1}^{n\_estimators} I_{i,b}\hat{H}_b(t|x_i)}{\sum_{b=1}^{n\_estimators} I_{i,b}} \tag{2.13}$$

[32]
$I_{i,b}$ determines whether the patient i is part of the OOB sample for estimator b. $I_{i,b} = 1$ means patient i is part of OOB sample for estimator b, while $I_{i,b} = 0$ means patient i is not part of OOB sample for estimator b. The error rate is calculated using Harells concordance index, as it does not have to calculate for a specific set time point chosen by the user of the model and takes censoring into account. The full overview of how the model works can be seen in Figure 2.9

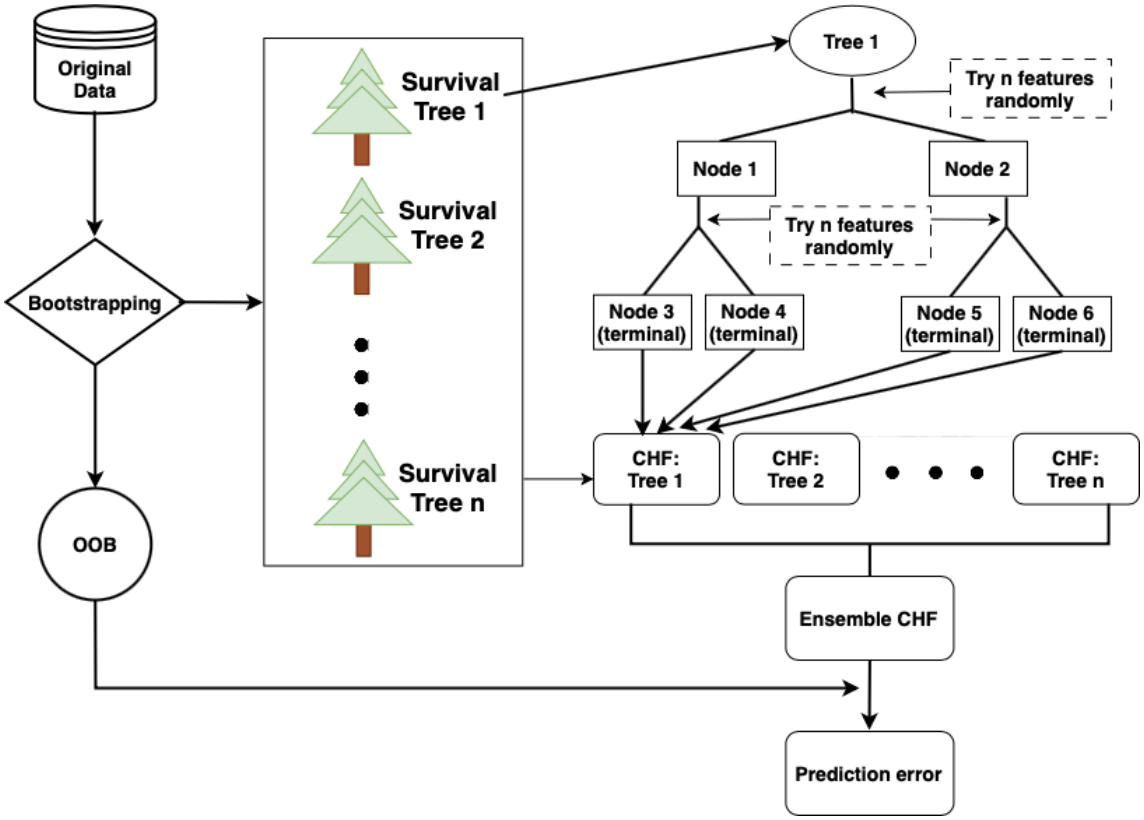


Figure 2.9: An overview of the inner workings of the Random Survival Forest algorithm from start to end.

2.4.2 Gradient boosting

Similar to Random Survival Forest, gradient boosting is an ensemble learner. The ensemble consists of multiple weak learners, and combining them will likely yield an acceptable result. What characterizes gradient boosting is the implementation of a functional gradient descent approach, which is a benefactor in minimizing the empirical risk[33]. In simple terms, this means that the model generalizes on the data domain we have for training as we do not know the true distribution of the data the model will predict on. Empirical risk will tell us how well the predicted values approximate the actual values in our dataset, and hence "is the average loss over the data points" [34]. The gradient boosting algorithm achieves this by sequentially fitting new weak learners to the residual errors of the preceding learner.
The base learner in gradient boosting from scikit-survival consists of multiple regression trees with the cox proportional hazards as its loss function [31], which means optimizing for the partial likelihood. As the negative log likelihood is a convex function, gradient based methods can be used [19], such as gradient boosting.
Survival gradient boosting using Cox proportional hazards is a modification of Friedman's Gradient boosting algorithm using regression trees, which was published in a paper in 2001 [35]. The modification was published in a paper by Greg Ridgeway in 2001 [36]. The goal for the function estimation is to find the regression function $\hat{F}(x)$ which will minimize a loss function $\psi(y, F)$ [36].
The gradient boosting algorithm will be explained followingly in a simplified fashion using an article from Greg Ridgeway [37]:
1. The model will start by initializing the terminal node predictions of the regression trees to 0, and step size p = 0 [37].

2. As the loss function $\psi(y, F)$ is set to the negative log likelihood, the model will then compute the gradient of the negative log likelihood and set that as the working response.
3. Next, the model randomly selects samples from the dataset, the number selected is the number of samples multiplied by the subsampling rate, which is a bag fraction.
4. The model then fits a regression tree on the residuals, with resulting K terminal nodes. The step size is then calculated for the algorithm in order to minimize the loss function which is the negative log partial likelihood.
5. The model then computes the predictions for the coefficients for all terminal nodes ($p1...pk$) [37].
6. Then the boosted estimate in step 1 is updated with the new estimate which is computed by taking the initial estimate and adding the new one multiplied by the learning rate. The algorithm then repeats from step 2 iteratively computing the gradient and fitting the regression trees on the new residuals until the model converges or a stopping criterion is met.
The concept is that each new weak learner, learns from the previous one and hence the goal is that the residuals (or working response) of the negative gradient direction will optimize for each step. The resulting output from the model is a risk score for each sample used for predictions.
Further extending the gradient boosting algorithm, there is a model called component wise gradient boosting [33], where the base learner is component wise least squares. Similar to the regular gradient boosted Cox model explained above, the negative log likelihood will be used as the loss function for this model as well. The use of least squares as the base learner results in a linear model, in contrast to the previously discussed model which is tree based. For each boosting iteration the base learners are fitted onto the working response of the previous model where the working response is component wise gradient of the negative log likelihood function [33]. The model its the base learners onto one feature at a time [38][39]. Within each boosting iteration the best update for a feature is selected, Which in essence leads to the update of only one of the elements for coefficients β [40]. This means that based on the number of iterations will determine the dimensionality of features used to predict, as the model initializes by setting all coefficients to zero, hence the result is a model which can prove to be highly efficient for high-dimensional data [33].

2.4.3 Penalized Cox model (Coxnet)

The Cox proportional hazards model has already been discussed in section 2.1.4. The coxnet algorithm is an extension of this model with the addition of the regularization technique known as elastic net. Elastic net consists of the two penalties known as ridge (L2) and lasso (L1). Although strictly not a machine learning algorithm, it uses concepts taught in a lot of machine learning courses at universities.
When working with high dimensional data where the number of features is close to or exceeds the number of samples, which often implies estimating many coefficients, the Cox proportional hazards model might not be the best suited model [31]. This is due to either unstable coefficient estimates or because the model fails to converge. In high dimensional data, the coefficients may as well go towards negative or positive infinity, because the models overfit due to fitting on the noise of the data. When the Cox proportional hazards model is fitting, the coefficients are calculated using maximum likelihood estimation. This step involves inverting a matrix, the Hessian matrix, which is a second partial derivative of the negative log likelihood function mentioned in section 2.1.4. This calculation step can be computationally expensive, especially when the number of features are large. A risk is the collinearity of features as well, which can lead to highly correlated estimates in the hessian matrix, which in turn might lead to the matrix becoming singular or ill-conditioned. As the Cox proportional hazard model uses this matrix to compute the variance and covariance among the regression estimates of the coefficients, a singular non-invertible matrix leads to problems in the computation. Therefore a non-desirable result might be poor regression coefficients.

Ridge (L2)

To deal with the potential problems in high dimensional clinical data, several researchers have addressed the problem with the implementation of elastic net [41]. One of the penalties in elastic net is called ridge (L1), which shrinks the coefficients towards zero [31]. Adding the L1-penalty to the likelihood function 2.5, the resulting objective function becomes:

$$\arg\max_{\beta} \quad \log \mathrm{L}(\beta) - \frac{\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \tag{2.14}$$

[32]
Features are noted as p and shrinkage is controlled with the hyperparameter α, which is a non-negative value. The L2 penalty in the optimization 2.14 is the sum of squared coefficients multiplied by the shrinkage parameter. Usually as the shrinkage parameter increases, the coefficient values in the vector $\beta_j, ..., \beta_p$ decreases. Dampening large coefficient estimates with the penalty can lead to less overfitting due to extreme coefficient estimates. The penalty will never set coefficient weights to 0, meaning the solution is not becoming sparse. It is rather an encouragement to keep the features which will lead to the features sharing information among them. This can be explained as ridge regression deals with highly correlated predictors by assigning them an equal weight [42]. Plotting the coefficient values against the shrinkage parameter it is easier to observe deviating paths for certain features, which might indicate the feature is of great importance. "The resulting objective is often referred to as ridge regression[31]".

LASSO (L1)

The second regularization technique is the L1-penalty known as LASSO penalty, which stands for Least Absolute Shrinkage and Selection Operator. In contrast to Ridge, lasso creates sparse solutions by setting several coefficients to 0 instead of shrinking them, which effectively implies excluding the particular features. similar to Ridge, LASSO can be applied to the likelihood function:

$$\arg\max_{\beta} \quad \log \mathrm{PL}(\beta) - \alpha \sum_{j=1}^{p} |\beta_j|. \tag{2.15}$$

[31]
In 2.15, alpha (α) is the regularization hyperparameter, and hence mathematically, the penalty consists of the regularization parameter multiplied with the sum of absolute values of the coefficients. The model optimizes the beta coefficients using coordinate descent [42], in essence, a form of log likelihood optimization discussed in section 2.1.4. In the coordinate descent, the coefficient estimates are updated one by one. The model utilizes a soft threshold operator which will shrink the coefficient estimate for a variable β_j to 0 based on the weighted sum of the predictor variable and the associated residuals, simply removing the feature. The coordinate descent continues until the number of coefficients either are small enough or a number of maximum iterations are reached. In essence, what Lasso does is "a type of continuous subset selection [31]". The number of features selected in the end depends a lot on the alpha hyperparameter, as it is a value between 1 and zero, the larger the parameter, the fewer features end up getting selected.

Elastic net

The Elastic Net penalty combines the best from both regularization techniques, Ridge and LASSO. As discussed the ridge penalty is good when dealing with collinear features, and as

such helps stabilize the Lasso penalty in some cases. The optimization problem when applying the elastic net penalty to Cox likelihood becomes:

$$\arg\max_{\beta} \quad \log \mathrm{PL}(\beta) - \alpha \left(r \sum_{j=1}^{p} |\beta_j| + \frac{1-r}{2} \sum_{j=1}^{p} \beta_j^2 \right), \tag{2.16}$$

[31]
Equation 2.16 is a combination of equation 2.14 and 2.15 with the addition of a relative weight, noted as r. The relative weight is a value between 0 and 1, and With alpha constant, as r moves from 0 to 1, the optimization problem is relatively weighed greater as an optimization with the lasso penalty rather than with the ridge penalty [13]. And only with a small relative weight between to ridge, such as 0.05 or 0.1, will help Lasso with removing very extreme correlations.

2.5 Regression

Regression analysis is a statistical method used for investigating the relationship between a dependent variable (the outcome) and independent features (explanatory variables) [43]. In this thesis more specifically referred to as the set of features and the length of survival as the dependent variable. By fitting a mathematical model the relationship between the variables can be explained, providing us with the ability to predict the outcome based on feature values. There are multiple types of regression, such as linear, logistic and polynomial regression, this thesis will look at linear regression models. The main difference from the survival models is that there is one target variable in the regression analysis performed in this thesis (the length of survival), whereas with survival models the target additionally consists of an event variable as well (is the patient deceased or censored).

2.5.1 Linear regression

Sci-kit learns linear regression model is ordinary least squares linear regression, also known as OLS. The aim of the linear regression model is to minimize the residual sum of squares when fitting the model to the coefficient vector $w = [w_1, w_2, ..., w_p]$
Simply explained, the model assumes that there is a linear relationship between the feature variables X and target Y [44]. OLS can be denoted as $Y = a + \beta X + e$. Where β represents the coefficients (also known as feature weights in machine learning language), a is the intercept and e is residuals (also known as error) [44]. When dealing with multiple features, the model is called multiple OLS regression, and a regression coefficient is estimated for each feature.

2.5.2 Ridge regression

Ridge regression is linear least squares regression which is L2 regularized [22], similar to the Coxnet model discussed previously. It is an extension of the ordinary least squares regression (OLS), where the penalty term is added to the objective function to regularize the complexity of the model. The squared sum of coefficients are multiplied with a tuning parameter (α).

$$||y - Xw||_2^2 + \alpha * ||w||_2^2, \tag{2.17}$$

[22]
The objective function can be seen in Figure 2.17, where w is the vector of coefficients calculated to minimize the squared sum of residuals. The α parameter controls how much the coefficient values are shrunk, where a higher value penalizes the coefficients more. As discussed previously the L2 penalty is great at shrinking highly correlated variables, which might lead to better generalisation of the model itself when predicting on unseen data.

To further explain how the coefficients are estimated with the penalty constraint, Figure 2.18 shows the objective function above written in matrix form.

$$RSS(\lambda) = (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta \tag{2.18}$$

[30]
The goal is to minimize $RSS(\lambda)$ where RSS is the residual sum of squares and lambda is the shrinkage parameter [30] instead of α. Deriving the objective function in matrix form yields the ridge regression estimator which can be seen in Figure 2.19

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T Y \tag{2.19}$$

[30]
Where the identity matrix I is of size p*p, where p is the number of features [30]. $(X^T X + \lambda I)^{-1}$ is often referred to as the shrinkage matrix, where the coefficients β are shrunk.

2.5.3 PLS Regression (PLSR)

PLS Regressions stands for Partial least squares regression. Partial least squares regression is a multivariate statistical method which models the relationships between a number of independent features (X) and a target variable (Y). Through a linear combination of the independent variables or features, a set of orthogonal components are computed which maximize covariance between X and Y, which are components that are best at predicting the target [45].
PLSR starts by centering the data around the mean value and scale it, hence transforming the variables to Z-scores [45]. This first step is optional. Sci-kit learns PLS regression uses the NIPALS algorithm to compute the components. To explain NIPALS in matrix form, the features and target can be considered as X variables and the target Y, as two matrices which are mean centered and scaled. Y is a matrix which can consist of multiple target variables, however as this thesis uses PLSR only one target variable (time of survival), PLSR with only one target variable will be explained.
NIPALS stands for Nonlinear iterative partial least squares and finds and calculates the components through an iterative process [46]. Some NIPALS algorithms initialize by setting a vector with random numbers u_a with the same length as number of samples [45]. In some cases the target column is set to u_a, or one of them, if there are multiple target variables. The algorithm proceeds with calculating weights w for the current component by a weighted linear regression between X and Y. The weights are normalized to have unit length in order to prevent relatively extreme values and are organized into matrix $W = [w_1, w_2..., w_A]$. Proceedingly the orthonormal scores are calculated by projecting the input features onto the weights, where the scores are a representation of new variables that capture the maximum covariance towards target Y. The orthonormal scores can be denoted as score matrix T containing the orthonormal score vectors for each component, such that $T = [t_1, t_2..., t_A]$, where A represents the number of computed components with $A = 1, ..., A$. Each orthonormal score vector contain the scores for each input variable.
Next, each and every feature or column is regressed with ordinary least squares regression (OLS) onto t_a, the orthonormal score vector in the current iteration [46]. OLS can simply be denoted as $y = \beta X$, where t_a replaces X such that the OLS regression becomes $y = \beta t_a$ [46]. The regression coefficients for each column in X are then stored in matrix P which are the projection loadings of X onto t_a [47]. P is also often referred to as the matrix of X-loadings. Then the deflation step proceeds by removing the component influence of the component from X. The last step in the iteration loop involves calculating the Y-loadings [47], which is found by projecting the orthonormal scores t_a onto u_a. The scores are then stored in matrix q^t. This iteration will stop until the desired numbered of components are reached or the model converges to a criteria in the iterative steps.

Finally, the regression coefficients are computed using the organized matrices of weights and score loadings (see equation 2.20) [47]. This enables the model to predict on unseen and new data.

$$\beta = W(P^T W)^{-1} q \tag{2.20}$$

[31]

for $a = 1 : A$, (A - the number of components to be extracted)

1. $\mathbf{v}_a = \mathbf{X}_{a-1}^t \mathbf{y}$
2. $\mathbf{w}_a = \mathbf{v}_a / \|\mathbf{v}_a\|$
3. $\tau_a = \mathbf{X}_{a-1} \mathbf{w}_a$
4. $\mathbf{t}_a = \tau_a / \|\tau_a\|$
5. $\mathbf{p}_a = \mathbf{X}_{a-1}^t \mathbf{t}_a$
6. $\mathbf{X}_a = \mathbf{X}_{a-1} - \mathbf{t}_a \mathbf{p}_a^t$ (the deflation step)
7. $q_a = \mathbf{t}_a^t \mathbf{y}$

end

Organize vectors and numbers into the matrices:
$\mathbf{T} = [\mathbf{t}_1\ \mathbf{t}_2 \dots \mathbf{t}_A]$ (the orthonormal scores),
$\mathbf{W} = [\mathbf{w}_1\ \mathbf{w}_2 \dots \mathbf{w}_A]$ (the orthonormal weights),
$\mathbf{P} = [\mathbf{p}_1\ \mathbf{p}_2 \dots \mathbf{p}_A]$ (the matrix of **X**-loadings),
$\mathbf{q}^t = [q_1\ q_2 \dots q_A]$ (the vector of **y**-loadings).

Finally compute the regression coefficients w.r.t. the original predictors of **X**:
$\mathbf{b} = \mathbf{W}(\mathbf{P}^t\mathbf{W})^{-1}\mathbf{q}$.

Algorithm 1: The NIPALS algorithm for computing an A-components PLS1 model with orthonormal scores and loadings.

Figure 2.10: NIPALS PLS1 pseudocode, posted with permission from Ulf Geir Indahl. Used for calculating PLSR with one target variable. [47]

There are various added benefits of using PLS for regression. The model is able to handle multicollinearity effectively, which is when multiple features are highly correlated. The latent variables or components which were introduced above capture the mutual information between features leading to a reduction in collinearity. Additionally, as the components which explain maximum covariance between X and Y are constructed, dimensionality is reduced, which might lead to more stable models when dealing with data of high dimensions.

2.6 Metrics

2.6.1 Harrells C-index

The most common metric in survival analysis when evaluating predictive accuracy of prognostic models is Harrell's concordance index, commonly referred to as C-index [48]. C-index assesses how well the model correctly ranks the samples, based on their predicted length of survival, this is done by a pairwise assessment of all observed samples. The metric ranges from 0 to 1, where a value of 1 is desirable which tells the model is perfectly ranking the predicted samples. A value of 0.5 indicates that the models performance is no better than a random guess.
As mentioned, a pairwise assessment is done. In order to understand how the C-index is calculated, it is necessary to define the three different types of pairs used; concordant, discordant, and tied. Additionally, it is necessary to define what makes a pair permissible as well. A pair can be permissible if both samples in the pair have experienced the event of interest within the

observational period illustrated in 2.2. A pair can also be permissible if one of the samples are censored, then the other pair must have a longer survival time without experiencing the event, simplified the censored sample has the shortest survival time and is the one experiencing the event.
A pair of samples is considered concordant if it is permissible and the sample with higher predicted survival time has the longer actual survival time, if it has not experienced the event of interest before the other observation. A pair of samples is considered discordant if the sample with highest predicted length of survival has the actual shortest survival time or experiences an event before the other sample in the pair. A pair is considered tied if the pairs have the exact same predicted survival time. When calculating the concordance, tied pairs are not taken into consideration and are excluded as they do not contribute to the ranking quality of the samples. In the article "A practical perspective on the concordance index for the evaluation and selection of prognostic time-to-event models", In equation 2.21 Harrells estimator has been defined by Longato, et al [49], which calculates the concordance index.

$$\widehat{C} = \frac{\sum_{i=1}^{N} \Delta_i \sum_{j=i+1}^{N} \Big[I\Big(T_i^{\text{obs}} < T_j^{\text{obs}}\Big) + (1-\Delta_j) I\Big(T_i^{\text{obs}} = T_j^{\text{obs}}\Big) \Big] \Big[I\Big(M_i > M_j\Big) + \frac{1}{2} I\Big(M_i = M_j\Big) \Big]}{\sum_{i=1}^{N} \Delta_i \sum_{j=i+1}^{N} \Delta_i \Big[I\Big(T_i^{\text{obs}} < T_j^{\text{obs}}\Big) + (1-\Delta_j) I\Big(T_i^{\text{obs}} = T_j^{\text{obs}}\Big) \Big]} \tag{2.21}$$

In equation 2.21 I is 1 if the argument is true and 0 if the arguments is false [49]. T is the time for an observation and M is the predicted risk score predicted by the model. δ_i is a binary variable, which equals to 0 if sample is censored before T_i and equals to 1 if event of interest takes place at time T_i [49]. Tied times and risk scores for observation i and j are respectively noted as $T_i^{\text{obs}} = T_j^{\text{obs}}$ and $M_i = M_j$. As seen by the estimator, Harrells C-index computes the C-index by computing a percentage of concordant pairs relative to permissible pairs, with the exception of tied events which are weighted by half. The equation can be further simplified to give a quick understanding of how popular Python packages scikit-survival and Lifelines compute the C-index as seen in 2.22.

$$C - index = \frac{\#ConcordantPairs + \#Tiedriskpairs * 0.5}{\#PermissiblePairs} \tag{2.22}$$

2.6.2 Uno´s C-statistic

As mentioned when introducing Harrells concordance index, it does only calculate for the proportion of concordant pair within the permissible pairs of samples. As an affect of this the concordance might have an optimistic attitude and cause an upwards bias when datasets with a high proportion of censoring are analyzed. To deal with this upwards bias, a new method of calculating the C-index was proposed. The method is commonly known as Uno´s estimator and solves the problem with the concept of inverse probability of censoring weights (IPCW) [31]. Uno´s C-statstic on the other hand does not rely on permissible pairs for computing the score, as the proposed statistic is free of censoring, meaning it does not depend on the censoring distribution [50]. Simply put, it can be defined as the probability that, given two randomly selected samples at a specified time point, the sample experiencing event of interest first, also is predicted with a higher risk score by the model. When scoring, each pair gets a weight as some pairs might be more informative than others. The applied weights are calculated using the concept of inverse probability of censoring weighting, which is gotten by the Kaplan-Meier estimate of the censoring distribution [50]. In order to compute the statistic using the scikit-survival package the range of survival times in the test portion of the data has to lie within

the train portions range of survival times. Therefore a truncated version of the statistic was defined, where the truncation time is selected with the hyperparameter τ [31]. In the end, when calculating the score for the truncated C-statistic, differently to Harrells concordance index, the weights for all concordant pairs are summed and divided by the sum of weights associated to all pairs [50]. The statistic considers a fully censored pair, although the contribution to the score is weighted as 0.

2.6.3 Brier score

The Brier score is another widely used metric in survival analysis to assess the predictive accuracy of models. It functions more as a supplementary metric rather than a replacement for Harrell's concordance index. The Brier score is the average of the squared error at a given point in time [31]. In the paper from 1950 where Brier introduced the score for weather forecasting, the score was defined as an average squared difference between the forecast which consisted of the probability of rain or no rain and the outcome, determined by a binary value [51]. As the survival function determines the probability of not experiencing an event, the Brier score was later amended for use in survival analysis, as the squared difference between prediction of not experiencing the event of interest and the actual event observation, in other words the Brier score calculates how well the model is calibrated [52]. The Brier score ranges from 0 to 1, where a score as close to 0 is desireable. A model with perfectly accurate predictions that match the actual observations will have a score of 0. A score close to 1 indicates that the model fails to predict the observed outcomes.
When applied with the scikit-survival package the time dependent Brier score looks at the predictions for survival for all sample predictions towards the relative frequency of events at time t. The relative sample predictions should closely match the relative frequency of events in order for a model to be well calibrated. A simple example can be described as 20 percent of samples experiencing the event of interest at time=10 months, Then the predictions in overall for the samples predicted should be near an overall failure rate of 20 percent. Therefore the Brier score is useful in addition to Harrell's concordance index, as it is a direct quantifiable measure on the accuracy itself and not solely the measure of a models ranking capabilities. The Brier score can as well give a better insight into how well the model will generalize. Exactly how the Brier score can is calculated in scikit-survival can be seen in equation 2.23 where censoring is incorporated [31].

$$\mathrm{BS}^c(t) = \frac{1}{n}\sum_{i=1}^{n} I(y_i \leq t \wedge \delta_i = 1)\frac{(0 - \hat{\pi}(t|\mathbf{x}_i))^2}{\hat{G}(y_i)} + I(y_i > t)\frac{(1 - \hat{\pi}(t|\mathbf{x}_i))^2}{\hat{G}(t)}, \tag{2.23}$$

The models predicted probability of a sample not experiencing the event up to a specific time t for the feature vector is noted as $\hat{\pi}(t|\mathbf{x})$ [31]. The feature vector is noted as x. The article which incorporated censoring with the Brier score described the the time dependent Brier score with three categories 2.23.

$$\begin{aligned}
&\text{Category 1: } y_i \leq t \ \ \& \ \ \delta_i = 1 \\
&\text{Category 2: } y_i > t \ \ \& \ \ \delta_i = 0 \text{ or } \delta_i = 1 \\
&\text{Category 3: } y_i \leq t \ \ \& \ \ \delta_i = 1
\end{aligned}$$

The first category is for observations with survival time less or equal to t, and experiencing the event within that time frame, meaning they are uncensored. Due to the indicator function ($I(y_i > t)$ becoming zero, the second addend is nulled and the contribution to the score of all observations in the first category which can be seen from equation 2.23 is $(0 - \hat{\pi}(t|\mathbf{x}))^2$ [52].

For the second category, it is known that all observations have a greater survival time than t, nulling the first addend, and contributing $(1 - \hat{\pi}(t|\mathbf{x}))^2$ to the score. Observations for category 3 have no known time-to-event as the observations are censored before time t, meaning the observations cannot contribute to the score [52]. The time dependent Brier score compensates for censoring by weighting the different categories, "($\frac{1}{\hat{G}(t)}$ is an inverse probability of censoring weight [31]" and is actually the Kaplan-Meier estimation of the censoring distribution [52]. All observations will have some impact on the weights although only observations in the first two categories have some impact on the probability of not experiencing the event of interest.

Integrated Brier Score

To assess how well the model is calibrated in general, the time dependent brier score can be extended to the Integrated Brier Score 2.24. The Integrated Brier score is the time dependent Brier score integrated over the time interval for all times, which is from t_1 to t_{max} with respect to a weighting function. Scikit-survival's implementation of the Integrated Brier score uses the weighting function $w(t) = \frac{t}{t_{max}}$ [31].

$$\text{IBS} = \int_{t_1}^{t_{\max}} \text{BS}^c(t) dw(t) \tag{2.24}$$

Similar to the time dependent Brier score, the estimates for the samples event free probabilities in relation to the score contribution has to be considered based on the censoring time. As the integrated score integrates for time, each sample's contribution of estimated event-free probabilities, if censored, are calculated up to the time of censoring [52].

2.6.4 RMSE

RMSE stands for root mean squared error, and is a common metric for evaluating regression models [53]. The desired value for RMSE is 0, meaning the model's predicted values are perfect. the higher value the worse the predictions are.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.25}$$

Equation 2.25 shows how RMSE is calculated for n number of sample predictions $\hat{y}_i$ and observations y_i, where $i = 1, 2, ..., n$ samples [53]. There are some advantages to RMSE when evaluating regression models. RMSE measures the average difference between the observations and predictions, which means that the number quantifies the overall prediction accuracy and gives the user a sense of how well the model aligns with the true observations. RMSE is very sensitive to extreme values as the errors are squared, a few outliers can increase RMSE substantially, which can prove to be valuable in both detecting and measuring the effect outliers have on the model. And lastly RMSE has the same units as the target variable Y [53], yielding a quantifiable measure which is easy to interpret.
By taking the square root of the mean of squared errors,

2.7 Data scaling and transformation techniques

Several standardization techniques were used in the analyses performed in this thesis. Standardization is a technique used in machine learning where the goal is to transform continuous feature values into a standardized scale. By ensuring that all continuous features are on the same scale, the model might have an improvement in accuracy. This is a consequence of some machine learning algorithms requiring standardized data as input to perform well.

2.7.1 Standard scaling

Standard scaling is a standardization technique where the mean is taken for each feature and then divided by the standard deviation of the feature for each sample. The values are then transformed and the new resulting distribution for each feature has a mean of 0 and with a standard deviation of 1. The equation for standard scaling is the same as the equation 2.7 when calculating the z-scores.

$$Z = \frac{X - mean}{std} \tag{2.26}$$

Equation 2.26 is the equation for calculating the new transformed value for a sample for a particular feature. x is the original value for a sample, the mean is the mean of all feature values for that particular feature, and the standard deviation is the standard deviation for all values in the feature. This results in the Z-score which is the new value for the sample. It is calculated on a feature to feature basis due to the nature of different scales. Considering the feature age in a dataset with a minimum and maximum value of 15 to 78 and heart rate which ranges from 20 to 200, the scales are different and not directly comparable. Features of comparable scales might ensure each feature contributes equally when fitting the model, as very large values can have a tendency to dominate features with a smaller range of values.

2.7.2 Min-max scaling

Min-max scaling is another normalization technique which scales continuous feature values to a specific range. The features are re-scaled to lie between 0 and 1, leading the transformed features to have a common scale.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.27}$$

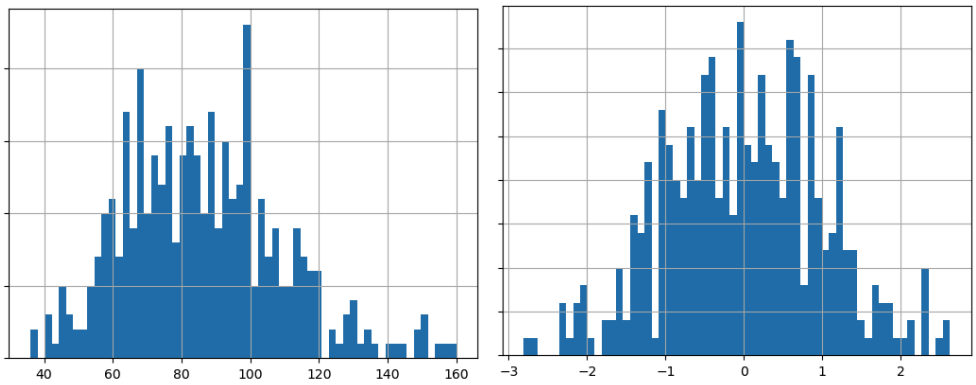
Equation 2.27 is the formula for calculating a new value for a sample within a feature. x is the original feature value for a sample, whereas x_{min} and x_{max} are the minimum and maximum values for the particular feature. x_{norm} is the normalized value. Using min-max scaling can as well help models perform better by restricting features with large values to have a dominant influence when calculating the coefficient weights. Normalization can lead to better interpretability between features as it might be easier to compare while the data might still be robust towards outliers as the relative distance between values are preserved.

2.7.3 Yeo-Johnson

Yeo-Johnson is not strictly a standardization technique, but can be regarded as a transformation technique which can be used for standardization of data. Yeo-johnson is a power transformation which is applied to continuous feature values in order to make the data distribution more normally distributed. Unlike standard scaling and min-max scaling, values are not transformed to a specific range or distribution but to a Gaussian like distribution. A great strength with Yeo-Johnson power transform is its ability to transform skewed data into a more symmetric form [54], it does also tend to perform well on data with varying levels of variance across the range of the data.
In order to understand how the power transform transforms the values it is necessary to introduce the correction parameter λ, which controls the direction and degree of correction regarding skewness. The formula for the Yeo-Johnson can be noted as [54]:

$$y_i^{(\lambda)} = \begin{cases} ((y_i+1)^{\lambda}-1)/\lambda, & \text{if } \lambda \neq 0, y_i \geq 0 \\ \ln(y_i+1), & \text{if } \lambda = 0, y_i \geq 0 \\ -((-y_i+1)^{2-\lambda}-1)/(2-\lambda), & \text{if } \lambda \neq 2, y_i < 0 \\ -\ln(-y_i+1), & \text{if } \lambda = 2, y_i < 0 \end{cases}$$

Where y_i represent the value i in the data, and the lambda parameter is a value that determines how the value is transformed. There are four ways a value can be transformed depending on the sign of y_i and the value of λ.



(a) Before (b) After

Figure 2.11: Yeo-Johnson power transformer from scikit-learn applied on feature 'heart rate' from the Worchester heart attack study dataset, with observed feature values on the x-axis and actual risk scored on the y-axis.

Figure 2.11 is an example of feature values before and after the Yeo-Johnson power transformer has been applied. As can be seen, the data is zero centered with a more even (symmetrical) spread on both sides. The zero centering is an integral part of Yeo-Johnson transformation in the scikit-lean package, Yeo-johnson itself does not address centering in the first place.

2.8 Imputation of data

A common problem with clinical data and in general where data is entered manually, in this case by doctors, is missing observed values within several features. There are several reasons as to why data can go missing, such as measurement error, data entry errors, and missing responses from surveys. The dangers of incomplete data is that it can lead to biased, inefficient or inaccurate analyses. When fitting models using scikit-learn it is crucial that the data of all features are complete, as the models will not accept missing values. To solve this, a commonly used approach is imputing the data. Imputation is the process of estimating the missing value. Several approaches exist to impute data, such as taking the mean of features or using distance based metrics such as the KNN-imputer.
The KNN-imputer from scikit-learn learn uses the K-nearest neighbors algorithm to impute missing values. The missing values are imputed by taking the mean from the K closest neighbors. The number of closest neighbors to look at is specified by the user through the hyperparameter n_neighbors. The algorithm starts by calculating the distance to all other data points in the dataset using a metric such as euclidean distance seen in equation 2.28, this is a hyperparameter and can be changed by the user. Consider sample A which has a missing value in the first feature, then the KNN-imputer will look for neighbors with the closest euclidean distance for feature 2 to N (N is total number of featues) and take the mean of those neighbors feature 1 values[55].

After K-number of data points with the smallest distance to the missing value is chosen, which is the nearest neighbors. Then the mean of the values of these K-neighbors is calculated and set instead of the missing value. The user can specify how the neighbors contributions are weighted, for example inverse proportion to the distance or uniform weighting.

$$\textbf{Euclidean distance } = \sqrt{\sum_{i=1}^{nsamples} (x_i - y_i)^2} \tag{2.28}$$

Euclidean distance between data points x and y where x_i and y_i are the i-th feature value for the data points.

Chapter 3

Datasets

The first step, before conducting any work consisted of some initial data exploration. Data exploration is the process to examine and understand how the data is observed and structured. It is necessary to understand the characteristics of the data before any analysis techniques are applied as it can help to detect critical errors or anomalies in the datasets.

3.1 Datasets

3.1.1 Oxytarget

The OxyTarget dataset has its origins from the OxyTarget study (ClinicalTrials.gov ID: NCT01816607)[56] and consists of data from patients diagnosed with colorectal cancer. 192 patients were observed over a five year period from 2013 to 2018. Throughout the observational period some manually has been manually entered by doctors and staff, such as observing patients recording and looking at MRI scans. One of the inclusion criteria to enter the study is that rectal cancer is confirmed and that the patient is scheduled for surgery. The raw data has a shape of 192 samples and 100 variables.

3.1.2 Head-Neck

The headneck data has its origins from Oslo University Hospital and consists of 197 patients of which clinical and PET data were extracted between 2007 and 2013 [57]. "Inclusion criteria were: squamous cell carcinoma of the oral cavity, oropharynx, hypopharynx and larynx treated with curatively intended radio(chemo)therapy and available radiotherapy plans based on FDG PET/CT. [57]". The data was received in three seperate files, one containing the clinical data, one containing PET data and the third containing response of disease free survival. The clinical data combined with PET data resulted in 15 variables which will be processed and used for evaluating models.

3.2 Data quality issues

For the OxyTarget dataset, most of the variables were incomplete. Several values were missing from each feature. One of the target values which was the survival time had missing values for all patients still at risk. Therefore a survival time had to be calculated for these patients by subtracting the datetime value of last registered alive, which is the datetime value of censoring or event with the inclusion datetime variable. This resulted in a survival time in months, as the number is easier to comprehend than days. Additionally the missing values were marked

in different ways and some values had obstructing symbols such as comma, which had to be removed.
There was a wider variety in datatypes for all the columns as well, most were of type object, and some of float64 and datetime. Looking at the variable names and values, it was immediately determined that some of the datatypes were wrong. In the Oxytarget data multiple of the features also has to be removed in order to fit a survival or regression model as they are not relevant for such analyses and will lead to error an error when fitting the model. Such features can be comments about the patient or datetime variables. To deal with this, many of the features were manually sorted into lists whether they were categorical, numerical or contained dates. Pre-processing of the data will be discussed further in the methods section.

3.3 Principal component analysis

3.3.1 PCA for Oxytarget

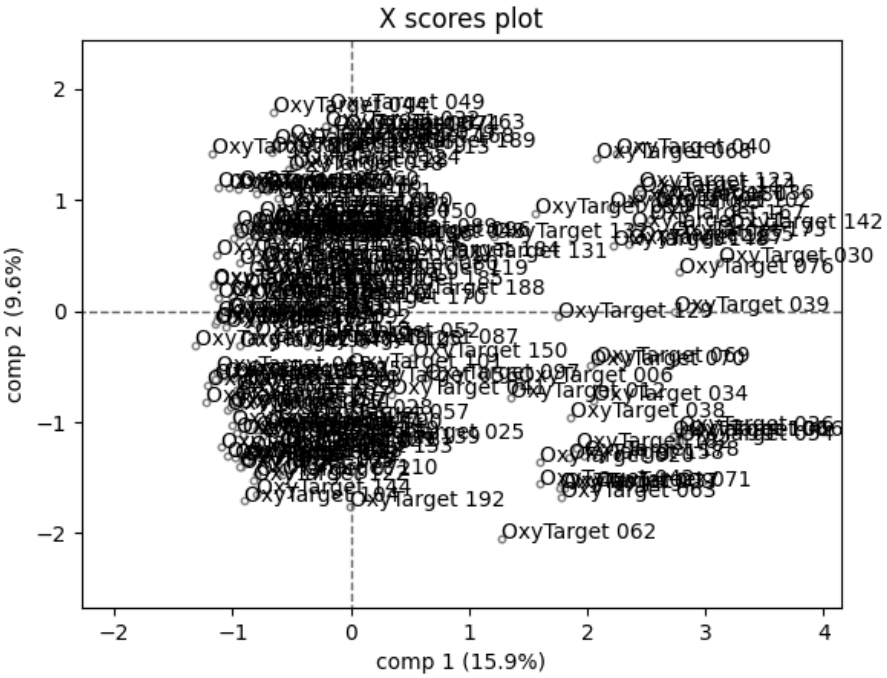


Figure 3.1: X score loadings plot from principal component 1 and 2 for OxyTarget

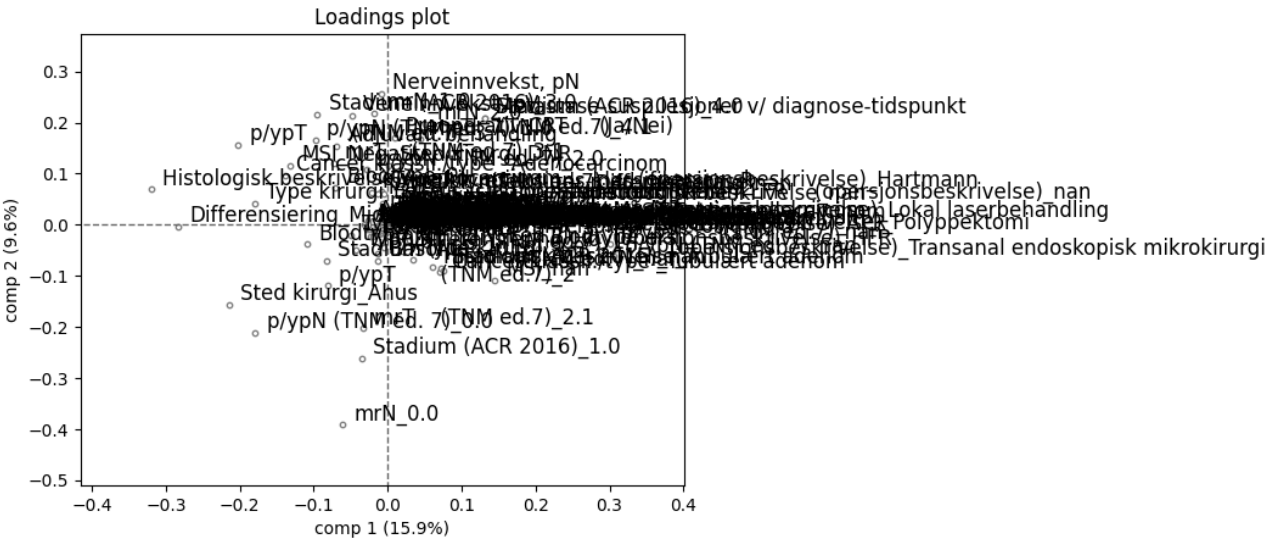


Figure 3.2: Loadings plot of principal component 1 and 2 for OxyTarget

3.3.2 PCA for headneck

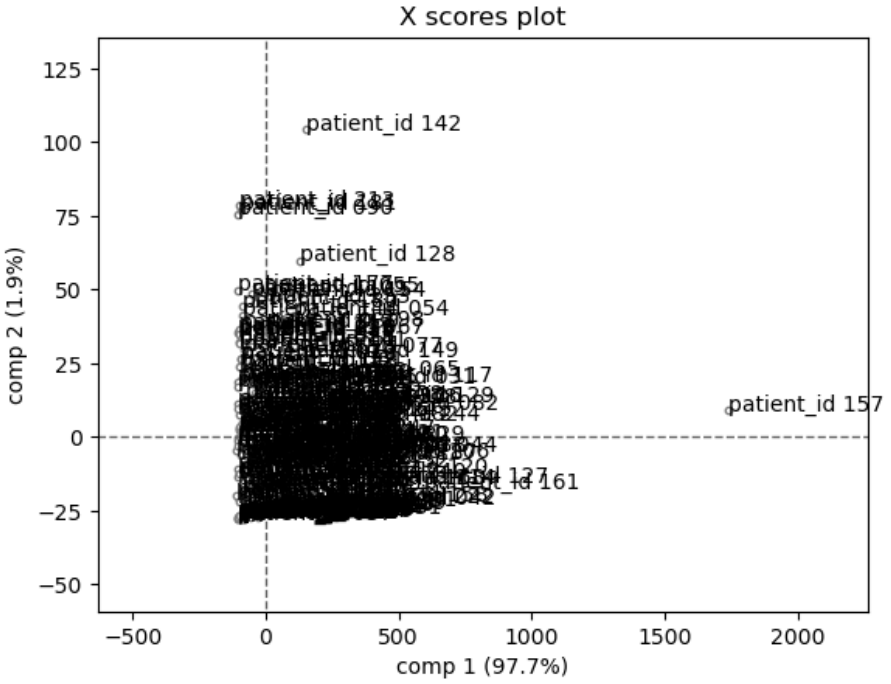


Figure 3.3: X score loadings plot from principal component 1 and 2 for headneck

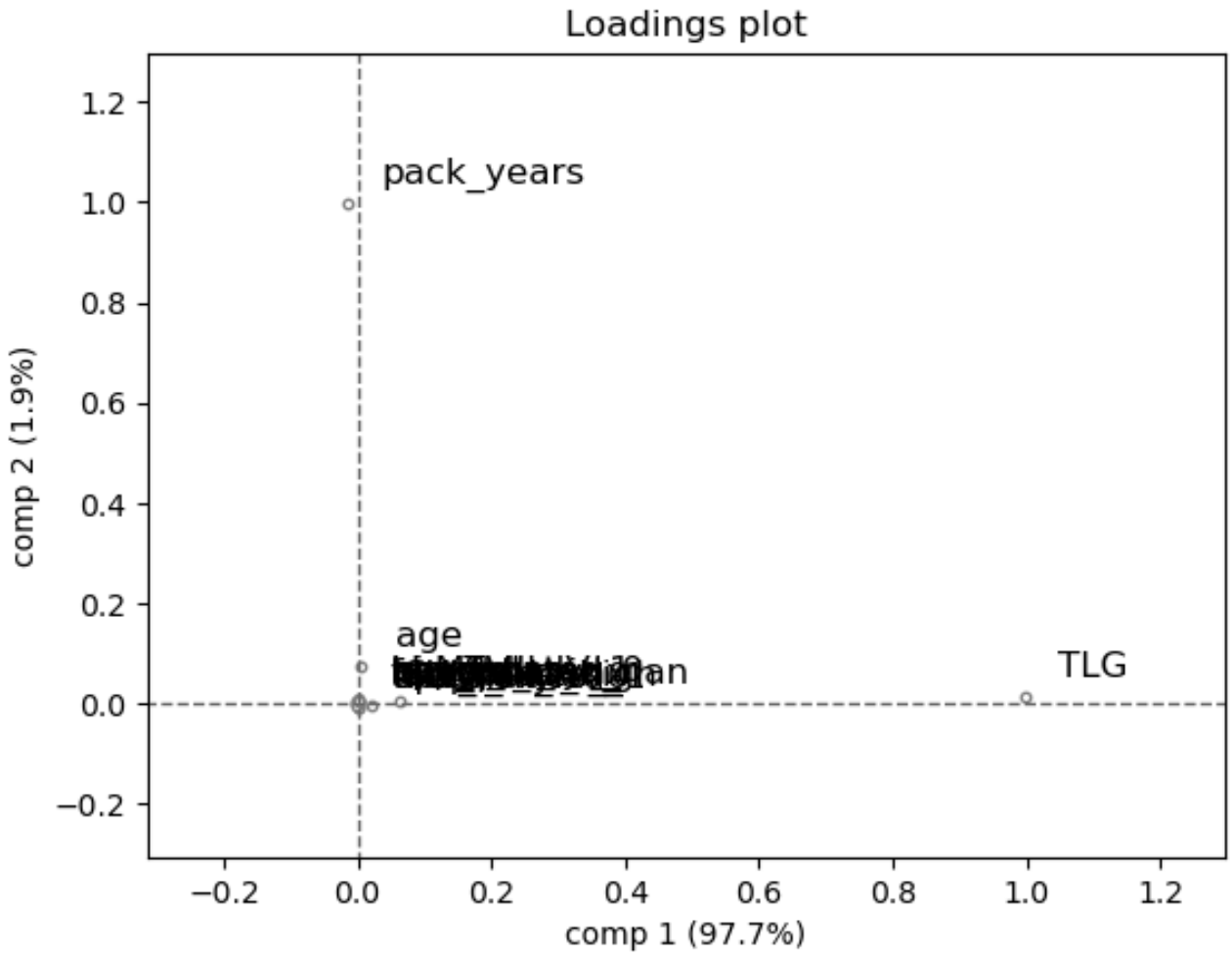


Figure 3.4: Loadings plot of principal component 1 and 2 for headneck

Chapter 4

Method

4.1 Pre-processing of the data

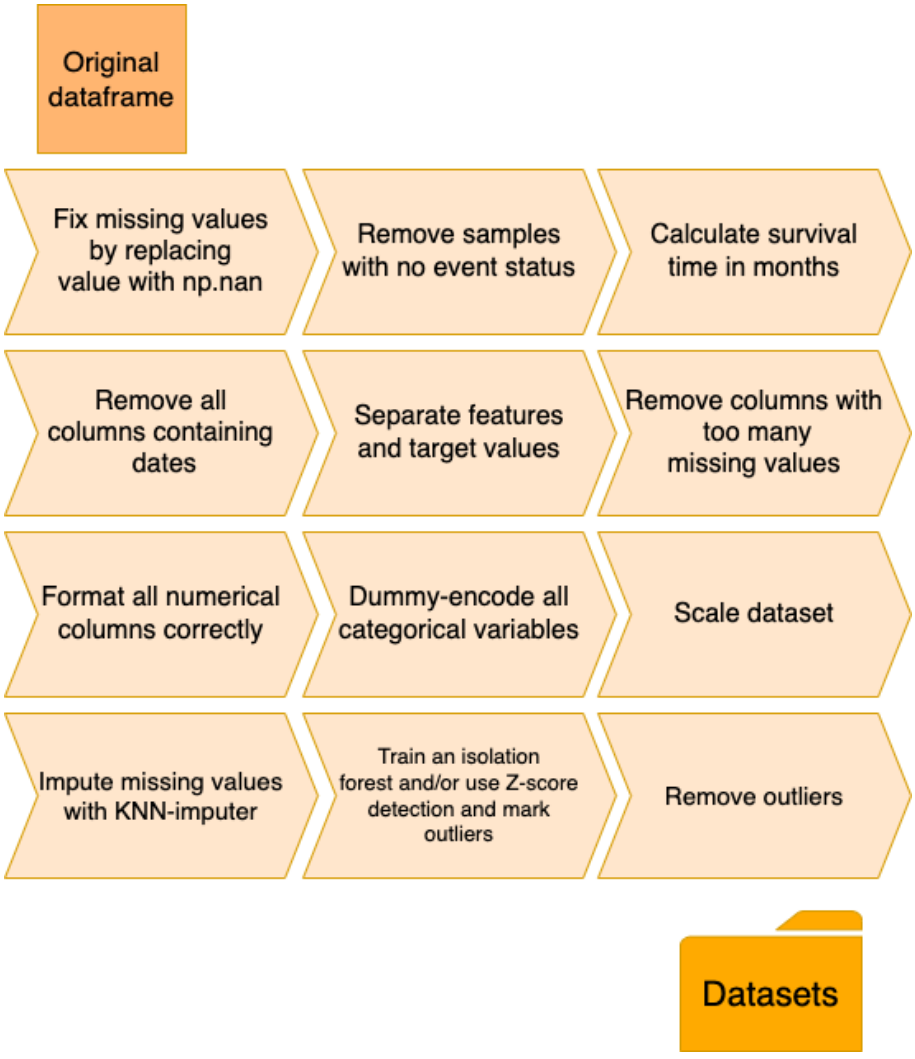


Figure 4.1: An overview of the consecutive pre-processing steps from start to end.

4.1.1 Formatting missing data

As mentioned in chapter 3, most of the columns in the OxyTarget data had missing values, and the missing values were marked either with "-", "nf" and "NF". In order to be able to deal with missing values further on in the process easily, all these values were changed to np.nan which is the way to represent undefined entries with Python NumPy package. Changing the values to np.nan is a crucial step, as imputation methods from sci-kit learn would not work with string values such as "NF".
The headneck dataset only had two variables with missing values, they were already formatted to np.nan, and were dummy encoded later on. Therefore nothing was done in particular on the dataset in this step.

4.1.2 Removing samples with no event status

Samples with no record of event status is not useful for survival analysis, as it is not known whether the patient is censored or the event of interest has occurred. For OxyTarget 7 samples with no known event indicator was removed. Some of these 7 patients were patients that had withdrawn their consent from the study. The headneck dataset had all responses intact.

4.1.3 Calculating the survival time

The survival time was not defined for any patients that were censored in the original OxyTarget dataset. In survival analysis the event indicator and survival time which is defined as a unit of time, like seconds, weeks and months is necessary in order to perform the analyses and predictions. Two of the initial features contained were the inclusion time and the date the patients were last registered alive. Therefore the timedelta in months between inclusion time and last registered alive was calculated and set as the target variable survival time. For the headneck dataset, the survival time in months was already calculated for all samples.

4.1.4 Sorting features and removing date columns

In the pre-processing phase, all columns for the Oxytarget data had to be sorted due to later processing steps. The columns were sorted based on whether they were continuous and numerical, categorical or datetime columns. After the sorting all the datetime columns were removed as they their values were irrelevant or had too revealing information. Then the target data was seperated from the dataframe to not be affected by the later processing steps. Additionally, irrelevant features such as doctors comments were removed as well. The headneck dataset contained only categorical and numerical variables, the non-binary categorical variables were noted down, for imputation later.

4.1.5 Columns with a high proportion of missing values

As previously mentioned, nearly all columns were to some degree incomplete as they had missing values. There are several risks to consider when there is a high proportion of missing values within a variable. First, it can lead to a biased analysis as the variable might not be representative of the true population. A high presence of missing values might in turn lead to reduced prediction accuracy of the model as the model might struggle to capture the true relationships between the features. Therefore a threshold was defined and all features with a proportion of missing values above 30 percent were removed in their entirety from the dataset. For the headneck dataset, as there were no missing values, none of the variables were removed.

4.1.6 Formatting numerical columns

The next step involved checking the datatype and values for columns that could be converted to a numerical datatype. A script was made where all the values in the dataframe is looped through. If the value is not not an np.nan value, commas are replaced with dots, as comma is not accepted for use with datatype floats. Then the ratio of numeric entries is calculated for a column, determining whether the column is numeric if 80 percent or above of the entries can be converted to float. If the column passes the threshold all values are then converted to floats. A script did also check the amount of unique numerical entries for all numerical columns, and all features that were below the threshold of 10 unique numerical entries were categorised and sorted as categorical features with numerical values. For the Oxytarget dataset, the feature "Hb baseline" was a bit troublesome and were altered manually.

4.1.7 Encoding categorical variables

The categorical variables had to be encoded as the models used are not able to interpret string entries for a feature. The solution which was used was sci-kit learns dummy encoding tool get_Dummies. Dummy encoding is a process where categorical variables are converted into binary variables. Each category is represented by a column, where 1 indicates the presence of the category for a sample, and 0 indicates and absence. 14 categorical columns were encoded as dummy variables, such as "Cancer classification type", "MSI" and "Type of surgery". As a last step column names were cleaned, by replacing and blank spaces such that all column names had a uniform format.
The headneck dataset had 2 non-binary categorical variables that needed to be dummy encoded, "hpv_related" and "uicc8_III_IV". As 'NaN' means "oropharyngeal tumors with unknown hpv status" for both of these variables, and the same set of patients did have 'NaN' for both variables, the 'NaN' values from both variables were encoded into a single column.

4.1.8 Scaling transformation of data

The datasets were scaled using 3 techniques, Yeo-Johnson Power Transform, standardscaling, and min-max scaling. The purpose of using three different scaling and transformation techniques is to check whether one of the three aforementioned techniques leads to any significant improvement in predictive performance over the other two. The second reason to implement scaling in general is that the KNN-imputer find the nearest neighbours by euclidean distance. Unscaled data where some variables have values that tend to be very large relative to the other variables might dominate and have a large influence on which neighbours that are chosen. Scaling and transformation techniques can solve this problem as the variables will have a common scale. It is also worth mentioning that the Yeo-Johnson Power Transform from sci-kit learn will center the data as well.

4.1.9 Imputing missing values

As the models are not able to handle missing values, as they would simply fail to fit and return an error, the missing values have to be handled. One approach is to simply remove samples containing missing values. For OxyTarget this would mean that all samples have to be removed and quite a lot for the headneck data as well, which is not feasible in this case. Additionally, the goal is to keep as much of the raw data as possible. The reason behind the high occurrence of missing values is due to a lot of the variables being related to individual treatments for patients and treatment types. Therefore as some treatments or medical examination might have been done on some patients, some might not have any value recorded. Some of the values might also not have been entered manually by the medical personnel at the hospital as well.

The missing values where imputed using sci-kit learns KNN-imputer and was applied on the whole OxyTarget dataset. The distance was to neighbors was chosen to be measured in euclidean distance and the number of neigbours was set to 5. The weighting was set to uniform as well, meaning that all the 5 neighbors contribute equally as much, and are not weighted relative to the distance. Each of the scaled datasets were imputed by the KNN-imputer. As there were no missing variables after dummy-encoding for the headneck dataset, there was no need to use the KNN-imputer.

4.1.10 Outlier removal

As the datasets have few patients partaking in the study, the goal is to maintain as much data for training and scoring as possible, only removing the worst samples which can influence the models poorly. Therefore both Z-score and Isolation forest was utilized on all three scalings (standard scaling, min_max scaling, and powertransform) for OxyTarget, removing only those samples which were classified as an outlier by the Isolation forest and Z-score for at least two of the three scaled datasets. Then the same samples were removed from all three variations of the scaled sets, in order for the different datasets to have the same dimensionality.
For the headneck dataset outliers were removed only using Z-score for the numerical values. Isolation forest suggested removing too many samples, even with the contamination parameter lowered drastically. Removing more than 15 samples would imply loss of too much information.

Oxytarget

Below are the figures displaying whether the samples are classified as an outlier by the Z-score or Isolation Forest. The threshold was set to four standard deviations for Z-score, in order to only remove the samples with the most extreme values.

Outlier	Z-Score	Isolation Forest
OxyTarget 030	True	True
OxyTarget 039	True	True
OxyTarget 056	False	True
OxyTarget 066	False	True
OxyTarget 113	True	True
OxyTarget 131	False	True
OxyTarget 180	False	True
OxyTarget 188	False	True

Figure 4.2: Outliers detected for the min-max transformation of OxyTarget dataset using Z-score and Isolation Forest

Outlier	Z-Score	Isolation Forest
OxyTarget 030	True	True
OxyTarget 039	True	True
OxyTarget 041	False	True
OxyTarget 054	False	True
OxyTarget 059	True	True
OxyTarget 084	True	True
OxyTarget 090	True	True
OxyTarget 097	False	True
OxyTarget 124	False	True
OxyTarget 142	False	True
OxyTarget 178	True	True
OxyTarget 188	False	True

Figure 4.3: Outliers detected for the standardscaled transformation of OxyTarget dataset using Z-score and Isolation Forest

Outlier	Z-Score	Isolation Forest
OxyTarget 030	False	True
OxyTarget 084	False	True
OxyTarget 096	False	True
OxyTarget 119	False	True
OxyTarget 178	True	True

Figure 4.4: Outliers detected for the powertransformed Oxytarget dataset using Z-score and Isolation Forest

The samples removed are OxyTarget 030, 039 and 178, as each of these samples are classified as outliers by both methods on at least two sets. This results in 182 samples which will be retained for model training and testing.

head and neck cancer dataset

Z-score outlier detection was applied on all three transformed subsets for the head and neck dataset (headneck). The threshold was set to four standard deviations, in order to only remove the samples with the most extreme values. The min_max scaled dataset and standardscaled dataset both detected the same four patients which can be seen in Figure 4.5. No outliers were detected on the powertransformed dataset. All four samples were removed for all three sets to keep the same dimensions. The outliers detected are to some degree consistent with the observations from the X scores plot when performing PCA in Figure 3.3, where it can be seen that patient id 157 and 142 and 157 deviate a lot from the rest of the samples.

Outliers: patient_id
patient_id 108
patient_id 157
patient_id 142
patient_id 082

Figure 4.5: Outliers detected using Z-score for the min_max and standardscaled transformation of the headneck dataset

4.2 Methodological framework

As one of the goals of this thesis is to explore to what extent scaling, transformation and feature selection have on survival and regression metrics on clinical data a methodogical framework had to be built to efficiently extract results. There are many widely known feature selection methods in scikit-learn for Python, however most of the methods are not compatible with right censored survival data with the target consisting of both a censoring variable and a time variable. Due to the ease of use of several feature selection techniques in R for survival data, the idea was to incorporate these feature selection methods from known R libraries. A database-like structure was adapted such that relating different scripts from R and Python could be executed easily, the overview of the framework can be seen in Figure 4.6.

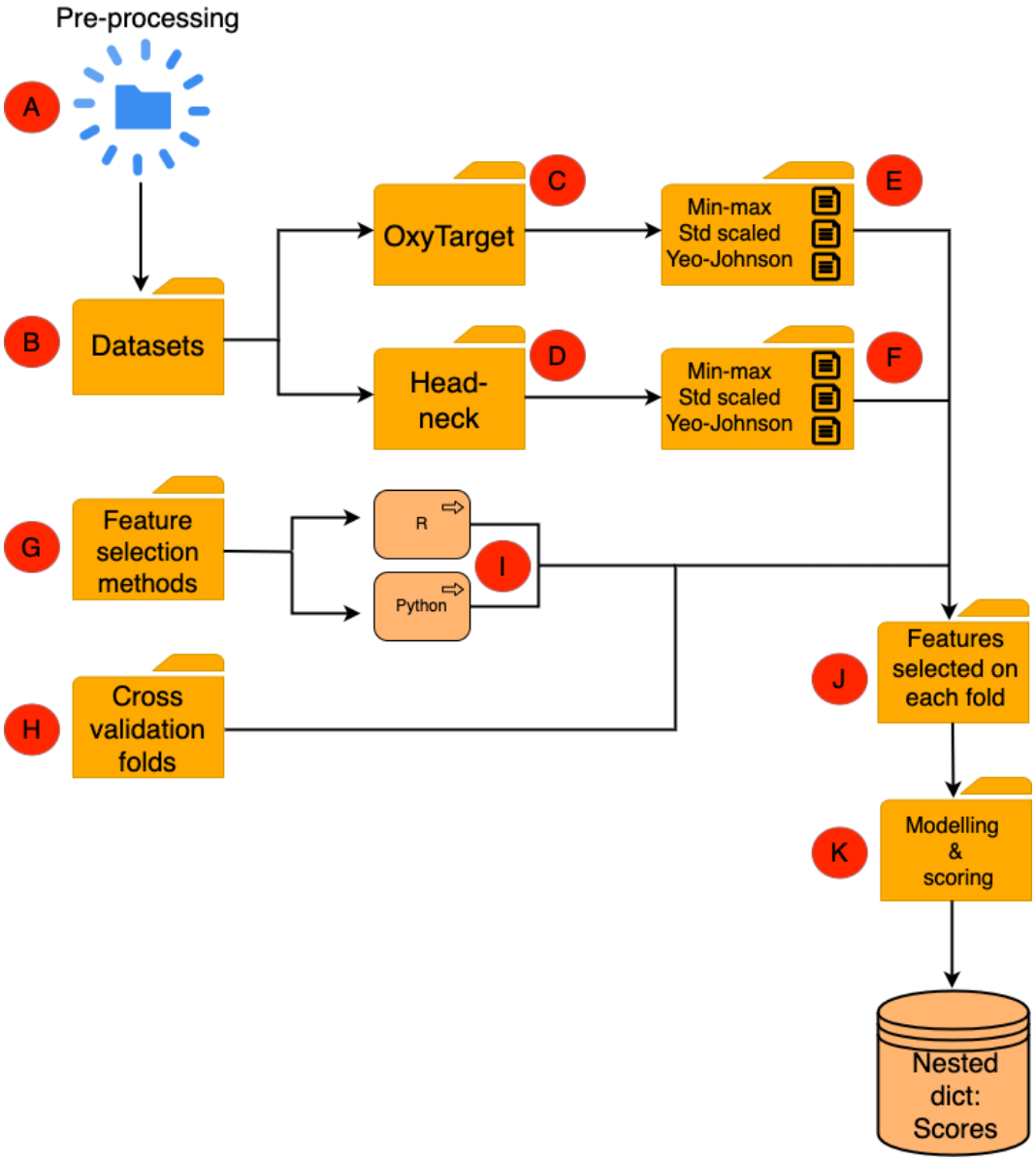


Figure 4.6: An overview of the methodological framework for feature selection and scoring of the models.

After pre-processing (step A), the differently scaled datasets are stored and structured into folders based on their main dataset (steps C and D). In step H a script is run to split the data for training and testing of models using repeated stratified k-fold cross validation. It is an improvement over the traditional k-fold and stratification ensures there is a near equal proportion of censored and uncensored data within each fold and split. Each fold will have a proportion of censored and uncensored samples which will be representative of the training data [29]. The added benefits of using a repeated stratified k-fold is that it reduces bias in performance estimation and indicates the robustness of a model against variability of within the data, as metrics from different repeats will indicate the performance across a variety of splits [29], which is essential when assessing and comparing different models.

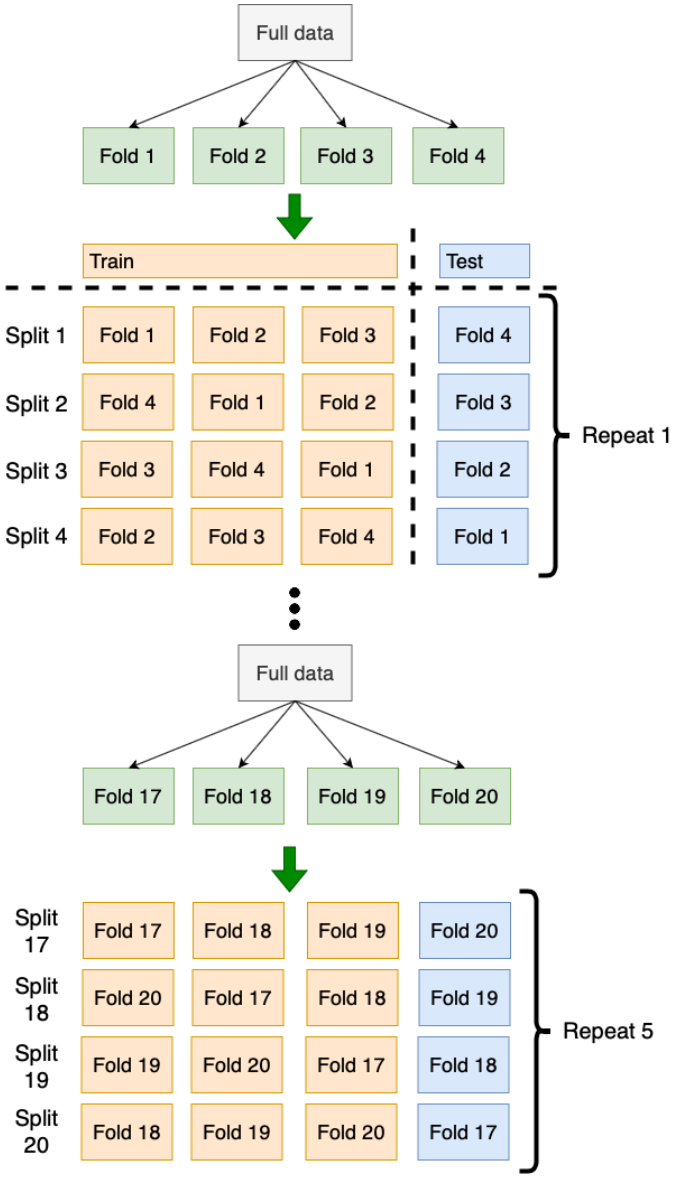


Figure 4.7: Example of repeated stratified k-fold with 4 splits and 5 repeats

The repeated stratified k-fold splits the data into a number of folds repeated for a number of times, in this case split into four folds, and repeated five times which results in 20 splits the data can train and test on. For each repeat and each split the model is fitted to three of the folds where the remaining fold is used for testing and scoring, this is done until all folds within the repeat has been used as a test set. Hence for the configuration illustrated in Figure 4.7, there will be 20 predictions for the 20 test folds, of which the average performance can be taken. In step H in Figure 4.6 the indexes for all the folds are saved to a CSV file, which will be used in steps G and J, meaning all models will be scored on the exact same folds. Step G is a script for feature selection, which will utilize the folds CSV file from step H and use these with the differently scaled datasets in order to perform feature selection. The script in step G executes two scripts in R and Python respectively in step I, where the R script contains the feature selection methods "Univariate Cox filter", "Random Forest variable importance", "Random Forest minimal depth" and "Random forest variable hunting". The Python script in step I contains the feature selection method "mRMR". The parameters and packages used when feature selecting in step I can be seen in table 4.1. The same hyperparameters used by

Spooner et al (2020) were used for the feature selection methods [7].

Table 4.1: Feature selection methods with corresponding packages and parameters

Feature selection method	package	Hyper-parameters
Univariate Cox filter	R: mlr and survival	predict.type="response" method = "univariate.model.score"
Random Forest variable importance	R: randomForestSRC	importance="permute", block.size=1
Random Forest variable hunting	R: randomForestSRC.var.select	method="vh" nstep = 1, ntree=1000, nodesize=3 nsplit=10, splitrule="logrank"
Random forest minimal depth	R: randomForestSRC.var.select	method="md", ntree=1000, nsplit=10, nodesize=3, splitrule="logrank"
mRMR	Python: mrmr_selection: mrmr_regression	-

After step G finishes running the scripts in step I, it saves the features selected for each fold and each dataset, and concatenates them into a CSV file for every transformed dataset for the two main datasets (step J). The result might be a different feature set for each fold, but it ensures there is no information leakage to other folds or the test fold when selecting features. Simply applying feature selection on the full dataset would reveal information for test samples and would not provide valid metrics when scoring.
Lastly, in step K, the features selected for each fold for the various datasets from step J and the folds file in step H containing indexes are used when fitting and scoring models. The various metrics Harrell's Concordance, UNOs concordance, truncated UNO concordance, IBS and RMSE are stored in a nested dictionary, such that the scores for various models and feature selection methods can be extracted easily. In step K all the models used are baseline models without any parameter tuning.

4.3 Methodology when tuning the models

To further explore how well the models compare performance wise, it is interesting to see how much better each model can perform when tuned. The machine used for running all the code is a 2020 macbook pro with an 8th generation Intel i5 processor which has four cores of 2 GHz base speed. The computer has 16 GB of RAM. To conserve time usage in regards to this thesis and due to system limitations the parameter tuning was only performed for subsets where all features were available.
Repeated stratified k-fold cross validation was used as well, similar to the example in Figure 4.7, only with 4 splits and 4 repeats, meaning the test proportion is 25 percent each time fitting the models. As there were few samples, a train test split was not made before fitting, all parameter combinations were fitted on all splits on all repeats. Repeated stratified k-fold is beneficial when parameter tuning as repeating the estimation enables the selection of parameters that perform consistently across a variety of splits. This ensures confidence in the results and evaluation across the different models.
The metrics on all folds were grouped and averaged for each parameter combinations. Scores for parameter combinations were sorted by Harrell's concordance index, then by UNOs concordance and Integrated Brier Score and will be discussed in results. The full tables can be found in Appendix B.

Chapter 5

Results

This chapter will present the discoveries and relevant results of the various metrics when assessing scaling, transformation, feature selection methods, base models and hyperparameters for the different models from this analysis.

5.1 Distribution of censorship

After outliers are removed, the distribution over time in months for censoring was plotted. The datasets share similar characteristics in when in comes to censoring, although the OxyTarget dataset has a higher proportion of censoring later on in the observational period. In the Oxytarget dataset 72% of samples are censored, whereas for the headneck dataset 62,2% of samples are censored.

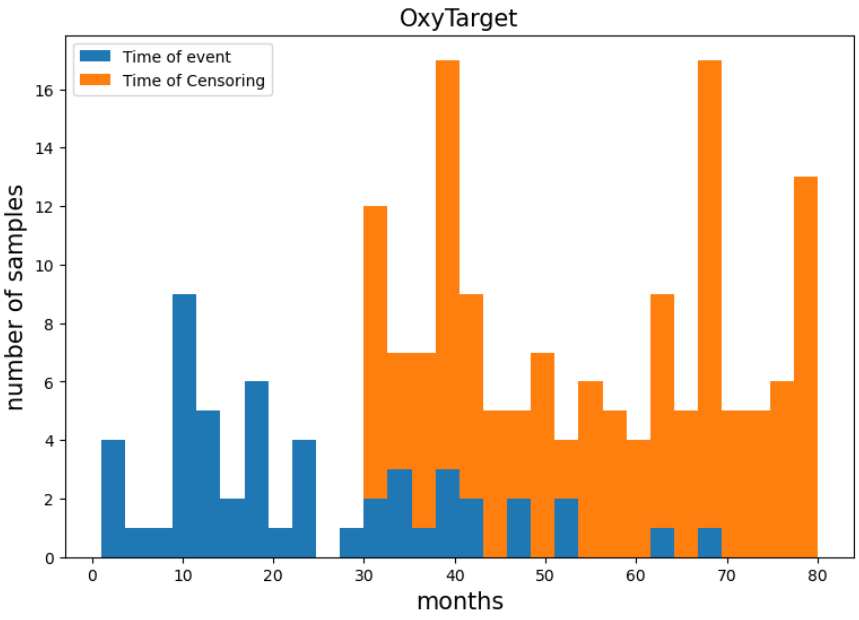


Figure 5.1: Distribution of censoring over time in months for Oxytarget, where 72 percent of samples are censored

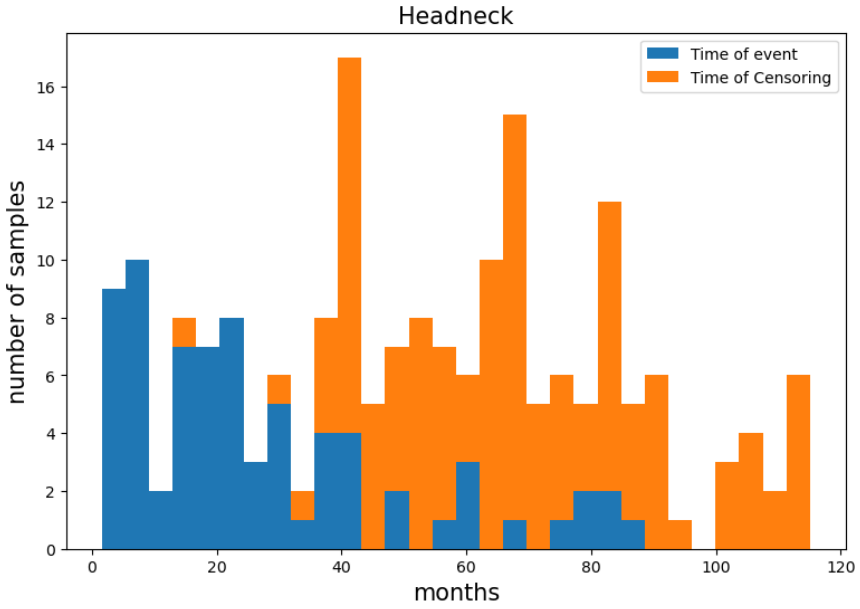


Figure 5.2: Distribution of censoring over time in months for headneck, where 62,2 percent of samples are censored

5.2 Scaling and feature selection

Three scaling or transformations techniques were applied, standardscaling, min_max scaling and the Yeo-Johnson powertransform. These scaling techniques were combined with several feature selection techniques and were assessed using various algorithms.

5.2.1 Most frequently selected features for OxyTarget dataset

	mrmr	var_hunting	coxfilter
1	mrt_tnm_ed_7_4.1	albumin	cea_baseline
2	mrn_0_0	bilirubin	metastase...
3	mrt_tnm_ed_7_4	natrium	stadium_acr_2016_4_0
4	stadium_acr_2016_4_0	eosinofile	p_eller_ypn_tnm_ed_7_0_0
5	metastase...	avstand_fra_analaapn...	alp
6	alp	cea_baseline	antall_lymfeknuter_med_metastaser
7	sted_kirurgi_nan	kirurgisk_reseseksjons...	minste_avstand_til_crm_mm
8	crp_baseline	monocytter	mrn_0_0
9	monocytter	hoeyde_cm	asat
10	sted_kirurgi_ahus	avstand_fra_analaapning_paa_mr	sted_kirurgi_ahus

	min_depth	rsf_vimp
1	cea_baseline	stadium_acr_2016_4_0
2	alp	antall_lymfeknuter_med_metastaser
3	metastase...	cea_baseline
4	stadium_acr_2016_4_0	alp
5	antall_lymfeknuter_med_metastaser	metastase...
6	kirurgisk_reseseksjons...	kirurgisk_reseseksjons...
7	crp_baseline	p_eller_ypn_tnm_ed_7_0_0
8	natrium	sted_kirurgi_nan
9	minste_avstand_til_crm_mm	minste_avstand_til_crm_mm
10	sted_kirurgi_nan	type_kirurgi_oper.._nan

Figure 5.3: Most frequently selected features on OxyTarget dataset by each feature selection method

	feature	times selected
1	metastase_minussusp_lesjoner_v_eller_diagnose_...	232
2	stadium_acr_2016_4_0	230
3	alp	223
4	cea_baseline	212
5	crp_baseline	202
6	sted_kirurgi_nan	185
7	kirurgisk_reseksjons_minusklassifikasjon_r_	184
8	type_kirurgi_opersjonsbeskrivelse_nan	178
9	antall_lymfeknuter_med_metastaser	176
10	monocytter	163
11	mrt_tnm_ed_7_4.1	153
12	sted_kirurgi_ahus	151
13	mrt_tnm_ed_7_4	147
14	mrn_0_0	144
15	minste_avstand_til_crm_mm	144

Figure 5.4: Most frequently selected features on OxyTarget dataset by all methods counted

5.2.2 Most frequent selected features for headneck dataset

	mrmr	var_hunting	coxfilter	min_depth	rsf_vimp
1	MTV	SUVpeak	hpv_related_0	pack_years	age
2	SUVpeak	hpv_related_0	ecog	ecog	ecog
3	uicc8_III_IV_0	female	pack_years	uicc8_III_IV_1	hpv_related_0
4	pack_years	uicc8_III_IV_1	uicc8_III_IV_1	hpv_related_0	oropharynx
5	oropharynx	TLG	oropharynx	age	pack_years
6	larynx	hpv_related_nan	age	TLG	uicc8_III_IV_1
7	hypopharynx	hypopharynx	hpv_related_1	MTV	TLG
8	hpv_related_nan	ecog	uicc8_III_IV_0	cavum_oris	MTV
9	hpv_related_1	oropharynx	TLG	SUVpeak	cavum_oris
10	hpv_related_0	pack_years	charlson	oropharynx	hpv_related_1

Figure 5.5: Most frequently selected features on headneck dataset by each feature selection method

	feature	times selected
1	pack_years	249
2	hpv_related_0	247
3	uicc8_III_IV_1	247
4	ecog	245
5	age	234
6	TLG	224
7	oropharynx	210
8	MTV	203
9	SUVpeak	183
10	cavum_oris	177
11	hpv_related_1	169
12	uicc8_III_IV_0	165
13	charlson	143
14	larynx	118
15	hpv_related_nan	105

Figure 5.6: Most frequently selected features on headneck dataset by all methods counted

5.2.3 Harrell's concordance index

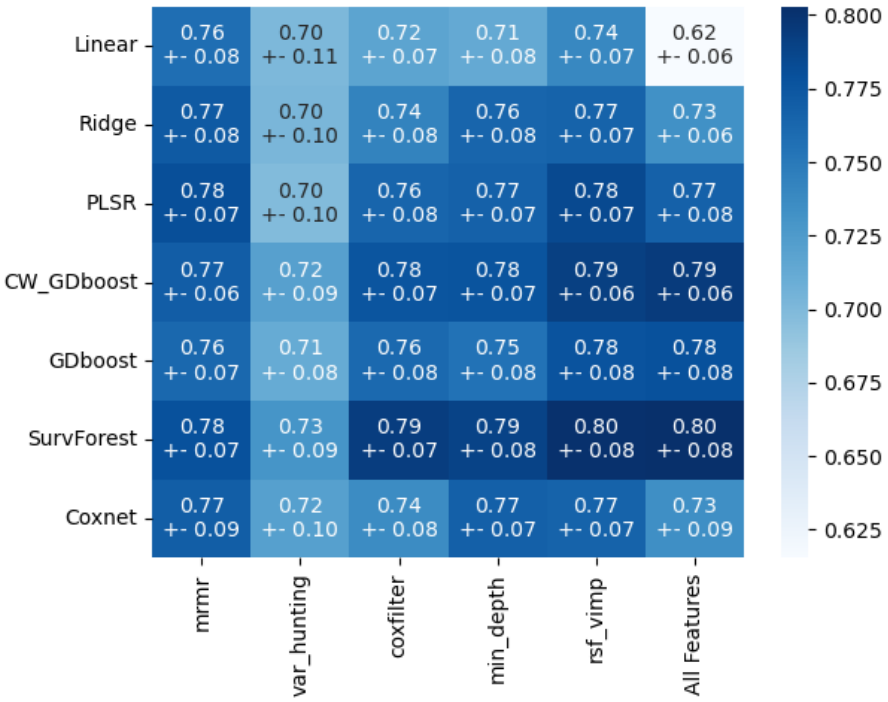


Figure 5.7: Harrell's concordance index for Oxytarget dataset with min_max scaling applied.

The heatmaps for standardscaled and powertransformed concordance index for OxyTarget can be found in Appendix A in Figure A.1 and A.2. All of the models fitted, are base models with default parameter settings. Taking the mean of all the results for each scaling there is little difference, the mean concordance index for min_max is 0.75, for standardscaled data it is 0.76 and for powertransformed it is 0.77. The same trend can be seen across all three figures, with random forest variable hunting performing poorly for all models. Linear regression and Ridge regression performs the worst in terms of ranking with all features available and Coxnet performing the worst out of all the machine learning models. In general feature selection helps with performance for linear regression and some for ridge regression. In terms of ranking Random survival forest is performing the best across all three subsets for OxyTarget, with some performance loss due to some feature selection techniques.

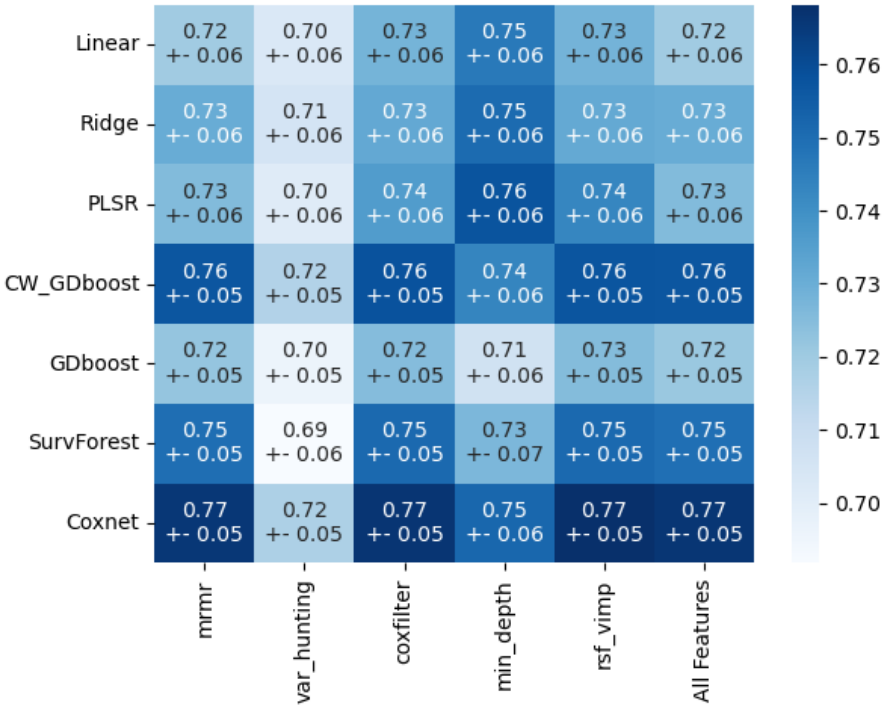


Figure 5.8: Harrell's concordance index for headneck dataset with min_max scaling applied.

The heatmaps for standardscaled and powertransformed concordance index for headneck subset can be found in Appendix A in Figure A.3 and A.4. All of the models fitted, are base models with default parameter settings. The mean concordance between scalings, similarly to OxyTarget show very little difference. The mean concordance for the whole table is 0,74 for min_max scaled and powertransformed subsets, and 0,73 for standardscaled subset. For headneck Random Survival Forest, Coxnet and PLSR are strong performers. For Linear Regression and Rigdge regression the increase in performance is not as strong as it was for OxyTarget.

5.2.4 UNO's C-statistic

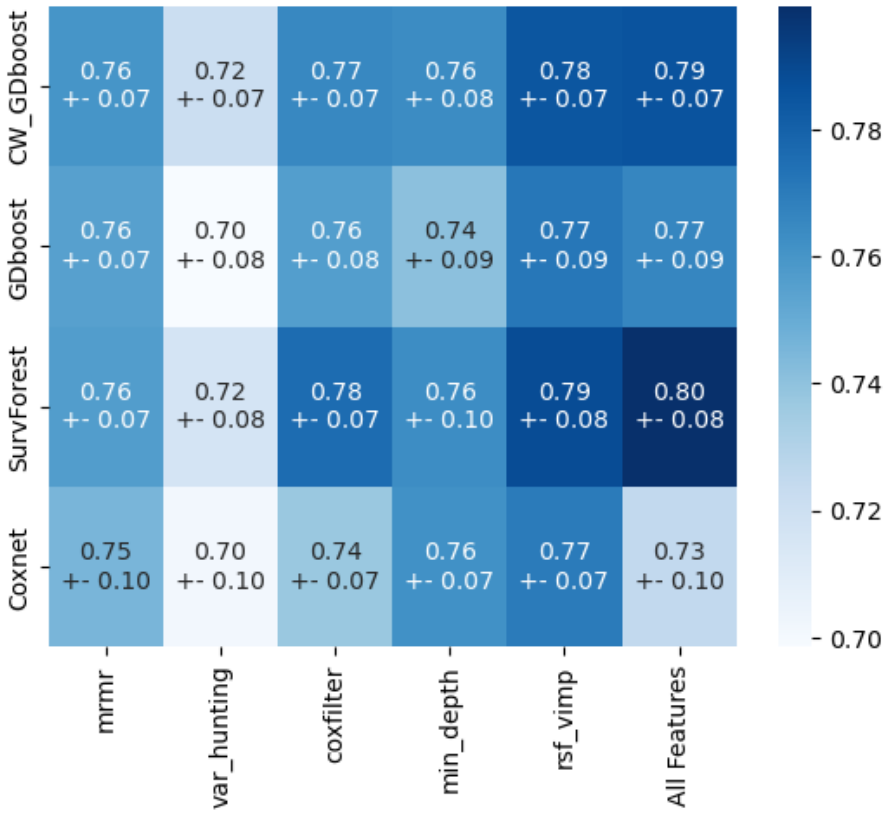


Figure 5.9: Scores for UNO's C-statistic for OxyTarget dataset with min_max scaling applied.

the heatmaps containing UNO's C-statistic for the standardscaled and powertransformed subsets for OxyTarget can be found in Appendix A in Figure A.5 and A.6. UNO's C-statistic is only calculated for the survival models as it is not possible to calculate the statistic for regression models. There is little difference across the different subsets with the min_max scaled subset having a mean score of 0.75 while the powertransformed and standardscaled have a score of 0.76.

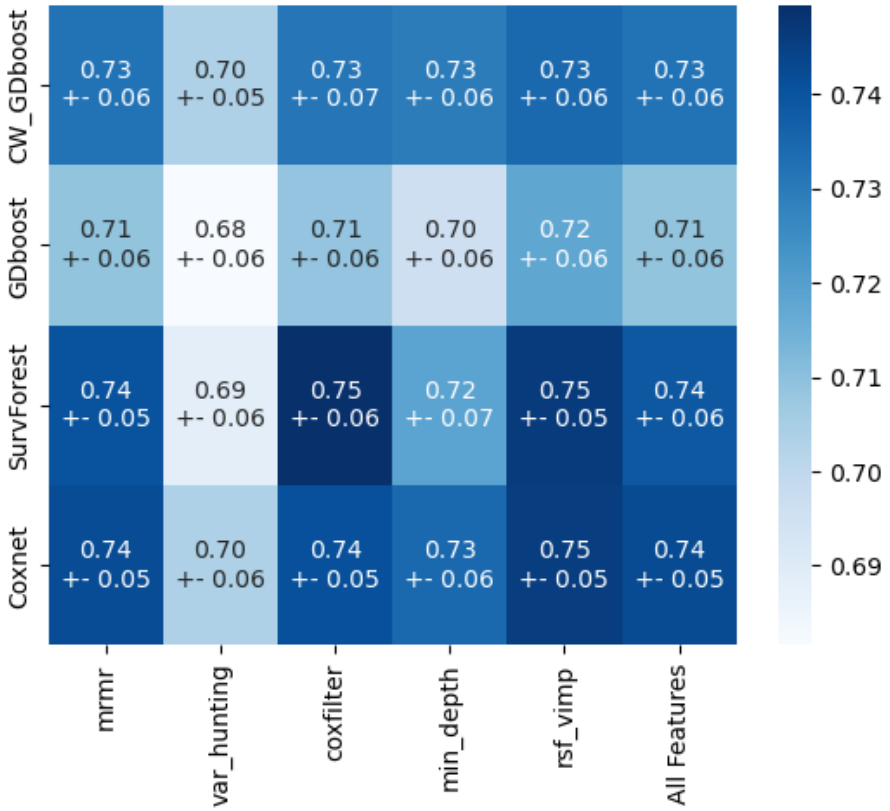


Figure 5.10: Scores for UNO's C-statistic for headneck dataset with min_max scaling applied.

the heatmaps containing UNO's C-statistic for the standardscaled and powertransformed subsets for headneck can be found in Appendix A in Figure A.7 and A.8. There is little difference across the different subsets with all three having a mean score of 0.72.

5.2.5 UNO's C-statistic with truncation

The heatmaps containing UNO's C-statistic with truncation for the standardscaled and powertransformed subsets can be found in Appendix A in Figure A.9 and A.10. The standardscaled subset had a slightly higher mean score of 0.78 while the min_max scaled and powertransformed respectively have a score of 0.76 and 0.77.

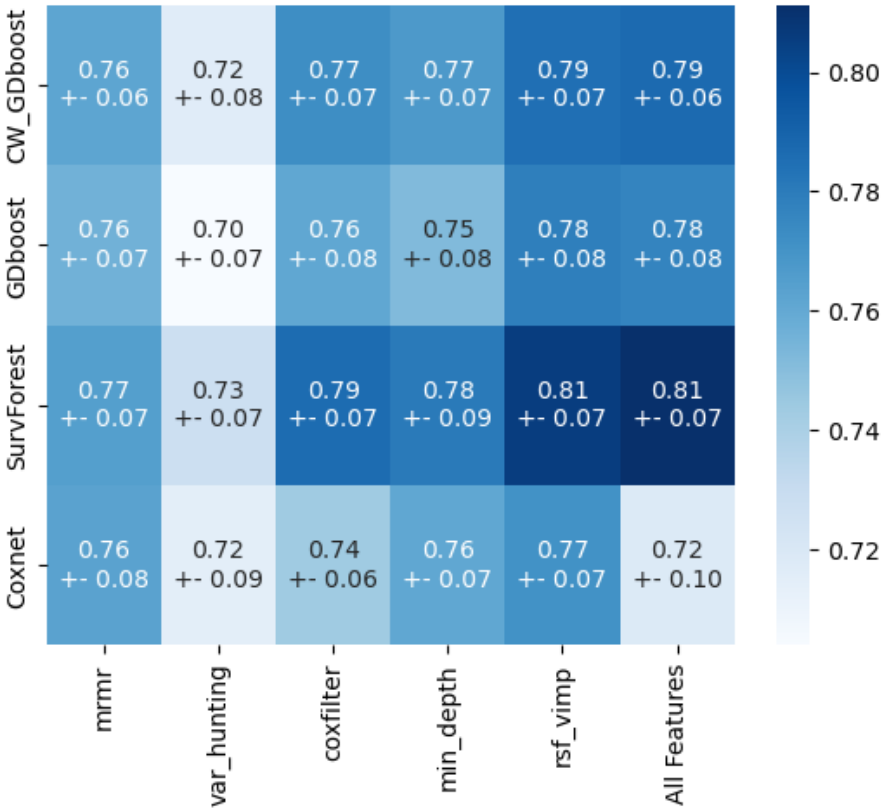


Figure 5.11: Scores for Uno's C-statistic with truncation for OxyTarget dataset with min_max scaling applied

The heatmaps containing UNO's C-statistic with truncation for the standardscaled and powertransformed subsets can be found in Appendix A in Figure A.11 and A.12. There was no significant difference across the subsets, as all had a mean score of 0.71.

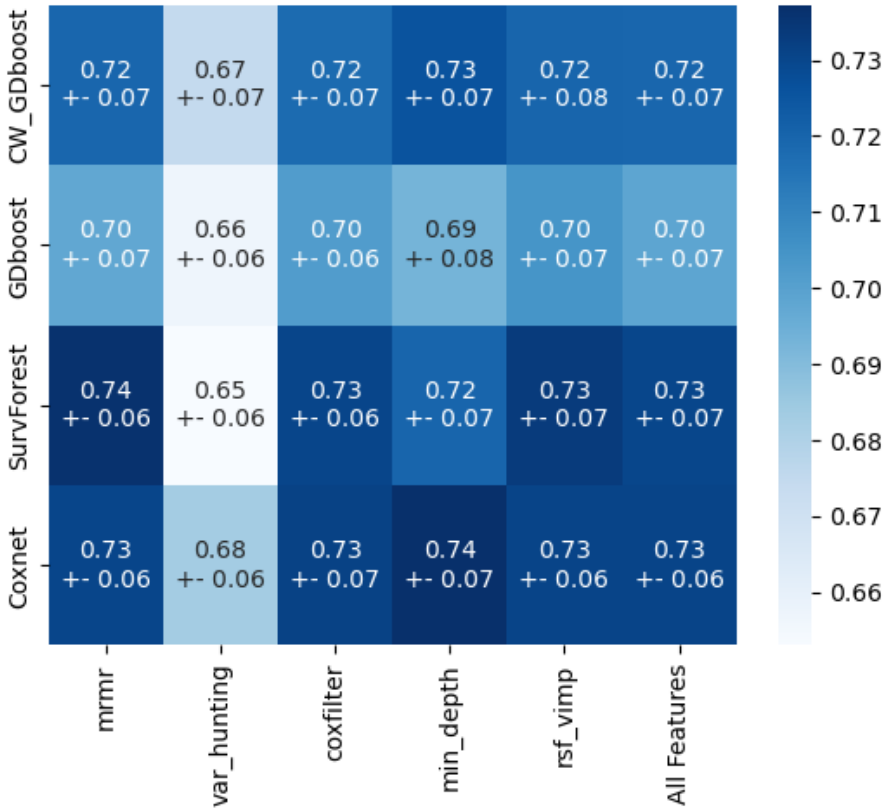


Figure 5.12: Scores for Uno's C-statistic with truncation for headneck dataset with min_max scaling applied

5.2.6 Integrated Brier Score

The heatmaps containing Integrated Brier Score for the standardscaled and powertransformed subsets were not able to be retrieved fort OxyTarget as certain combinations of folds, subsets, feature selection methods, and models led to an error in the prediction of the survival curve. Only metrics for min_max subset was retrieved for OxyTarget. IBS was truncated at month 52 for OxyTarget. As can be seen from Figure 5.13 the best calibrated models overall are the Component wise gradient boosted model and Survival Random Forest. Random forest variable hunting yields worse IBS scores for all models, except for Coxnet, which has a very high IBS score when it has access to all features.

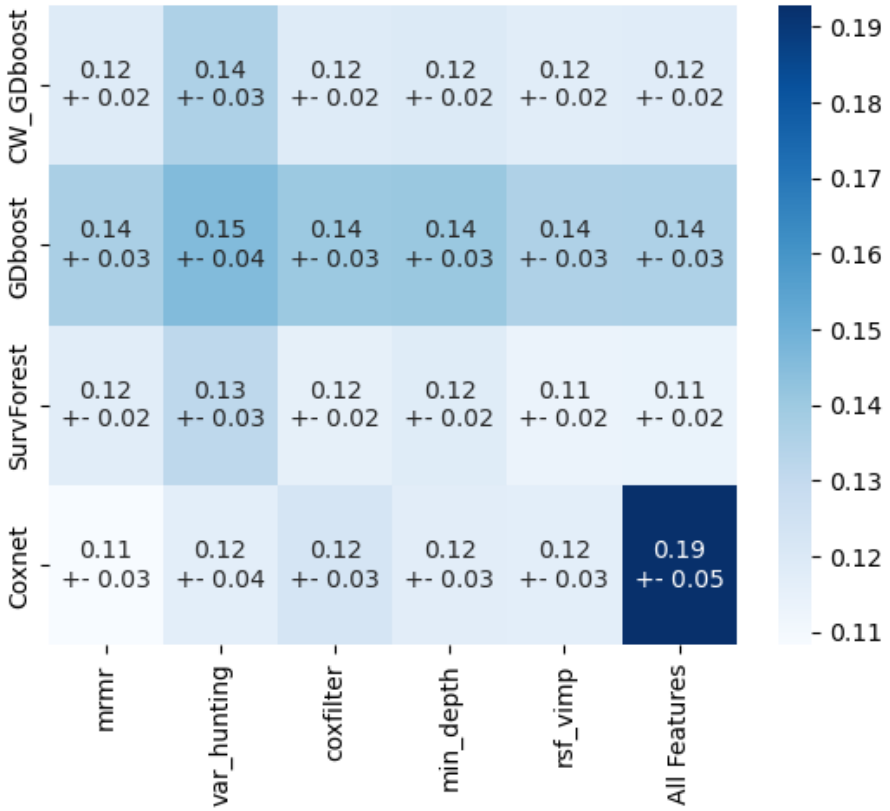


Figure 5.13: Integrated Brier Score for OxyTarget dataset with min_max scaling applied

The heatmaps containing Integrated Brier Score for the standardscaled and powertransformed subsets for headneck dataset can be found in Appendix A in Figure A.13 and A.12. IBS was truncated at month 60 for headneck. There was no difference across the subsets, as all had a mean score of 0.16. The same trend applies to all subsets, where the gradient boosted tree based model and Random Forest variable hunting feature selection yielded the worst scores.

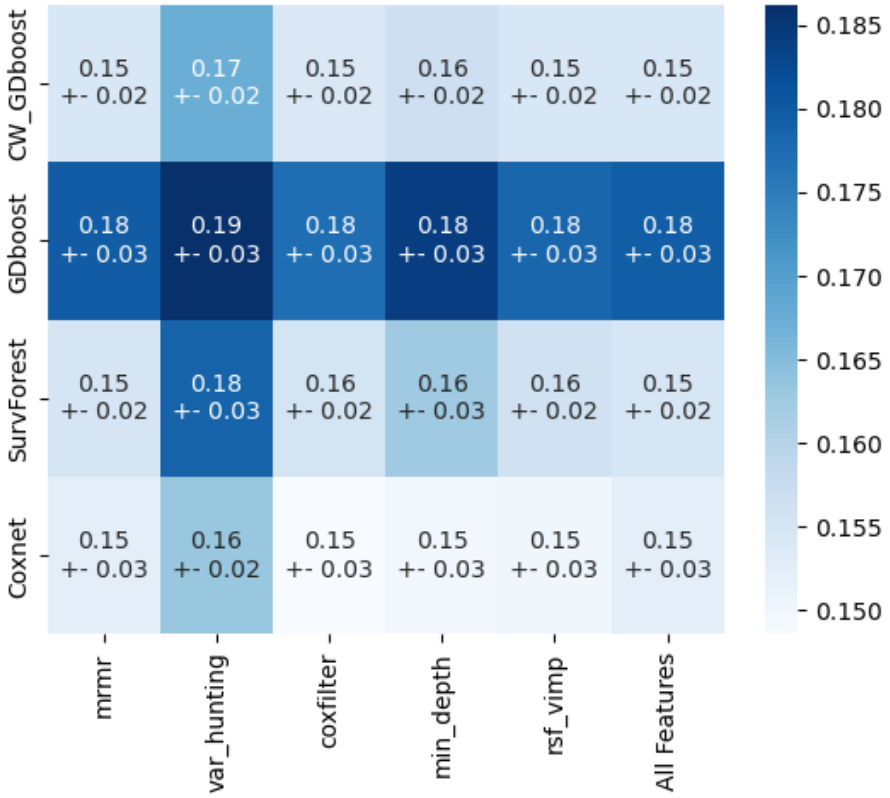


Figure 5.14: Integrated Brier Score for headneck dataset with min_max scaling applied

5.2.7 RMSE

The heatmaps containing RMSE for the standardscaled and powertransformed subsets for OxyTarget can be found in Appendix A in Figure A.15 and A.16. Feature selection has a great effect for Ridge regression and Linear regression in general except for Random Forest min_depth for the min_max scaled and standardscaled subsets, where RMSE is very large. Feature selection has a very slight effect for the better on PLSR. The best scores are achieved by powertransforming the data where all RMSE scores are below 20, except for linear regression with all features available. From the results, both PLSR and Ridge perform very close and better than Linear Regression.

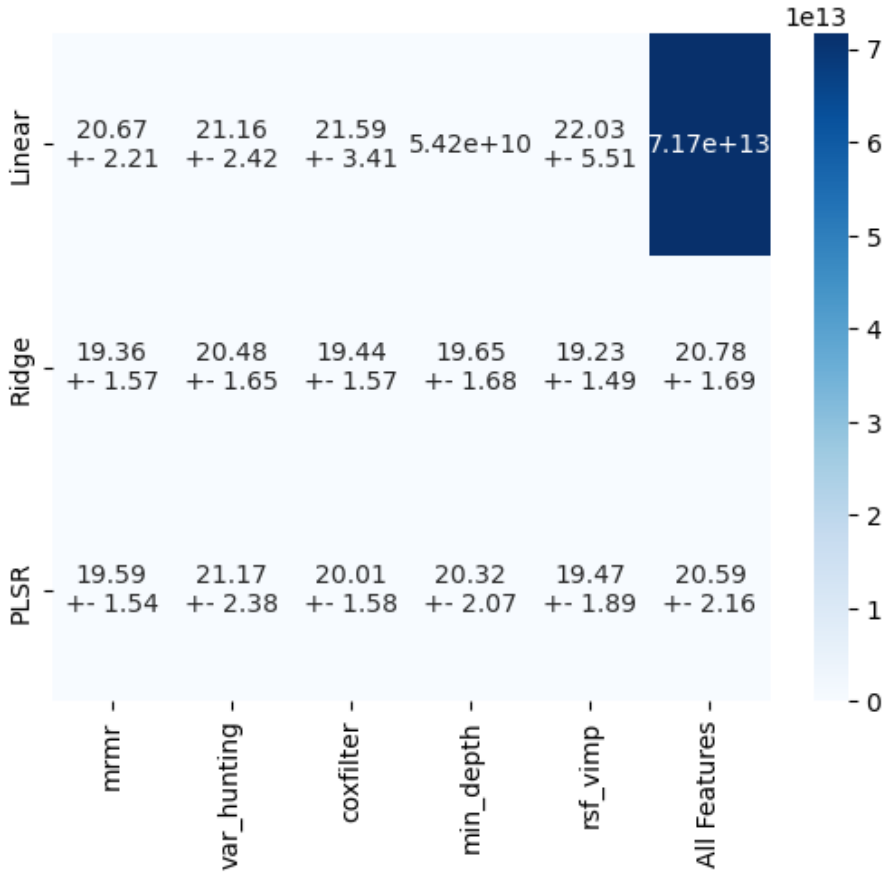


Figure 5.15: RMSE for OxyTarget dataset with min_max scaling applied

The heatmaps containing RMSE for the standardscaled and powertransformed subsets for headneck can be found in Appendix A in Figure A.17 and A.18. RMSE metrics were worse for the headneck dataset compared to OxyTarget the difference between the subsets were miniscule. Interestingly enough Ridge regressions seems to be the model with best RMSE scores, although the difference is so small compared to Linear regression and PLSR that it is negligible.

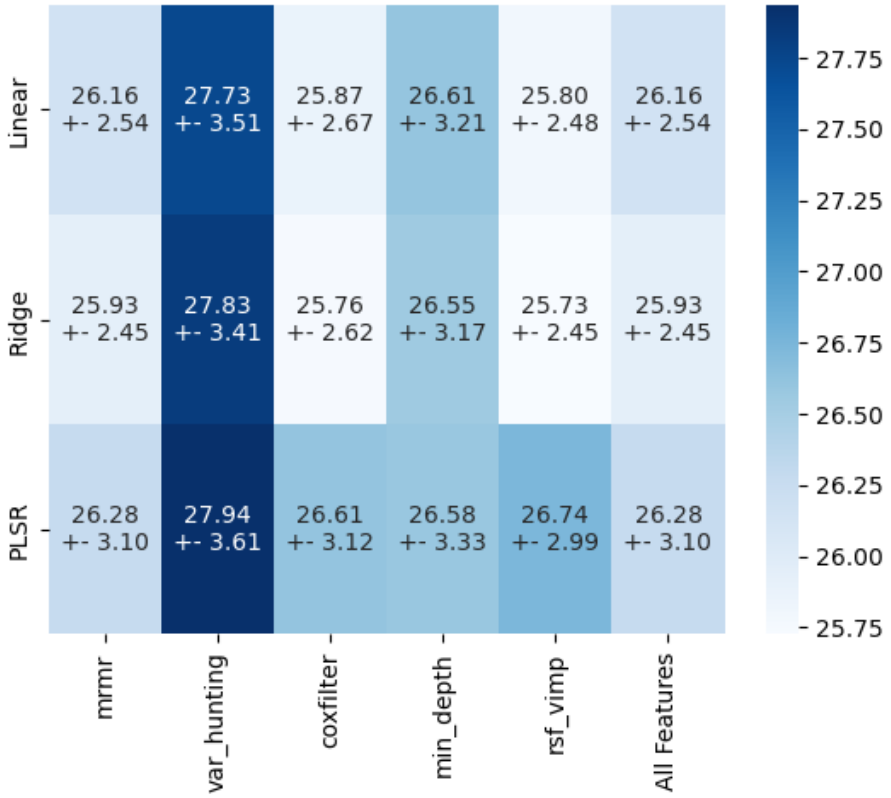


Figure 5.16: RMSE for headneck dataset with min_max scaling applied

5.3 Parameter tuning

5.3.1 Gradient boosting

Parameter tuning was done in two rounds for the tree based gradient boosting. First the number of estimators tuned was between 100 and 500, with increments of 100, whereas learning rate was tuned between 0.1 and 1 with an incremental increase of 0.1. The second round of tuning was for parameters number of estimators, learning rate, min_weight_fraction_leaf and max_depth. Loss was set to cox proportional hazards.

Tuning for number of estimators and learning rate

The full tables can found in Appendix B and Figures B.7 and B.9.

estimators	learning rate	Harrell-C	UNO-C	IBS
300	1.0	0.769	0.757	0.175
400	1.0	0.769	0.756	0.176
200	1.0	0.769	0.755	0.174
500	1.0	0.766	0.754	invalid score
100	1.0	0.766	0.749	0.168

Table 5.1: 5 best models for OxyTarget dataset using tree based gradient boosting and tuning for number of estimators and learning rate.

estimators	learning rate	Harrell-C	UNO-C	IBS
100	0.1	0.742	0.733	0.173
100	0.2	0.734	0.728	0.189
200	0.1	0.730	0.724	0.190
300	0.1	0.726	0.721	0.200
100	0.4	0.724	0.722	0.209

Table 5.2: 5 best models for headneck dataset using tree based gradient boosting and tuning for number of estimators and learning rate.

Tuning for number of estimators, learning rate, weight_fraction and max_depth

The full tables can found in Appendix B and Figures B.8 and B.10.

estimators	learning rate	weight_fraction	max_depth	Harrell-C	UNO-C	IBS
100	0.4	0.4	2	0.775	0.784	0.124
100	0.4	0.4	3	0.775	0.784	0.124
100	0.4	0.4	4	0.775	0.784	0.124
100	0.4	0.4	5	0.775	0.784	0.124
100	0.4	0.4	10	0.775	0.784	0.124

Table 5.3: 5 best models for OxyTarget dataset using tree based gradient boosting and tuning for number of estimators, learning rate, weight_fraction and max_depth.

estimators	learning rate	weight_fraction	max_depth	Harrell-C	UNO-C	IBS
100	0.1	0.3	2	0.768	0.747	0.148
100	0.1	0.3	3	0.768	0.747	0.148
100	0.1	0.3	4	0.768	0.747	0.148
100	0.1	0.3	5	0.768	0.747	0.148
100	0.1	0.3	10	0.768	0.747	0.148

Table 5.4: 5 best models for headneck dataset using tree based gradient boosting and tuning for number of estimators, learning rate, weight_fraction and max_depth.

5.3.2 Component wise gradient boosting

Parameter tuning was done in two rounds for componentwise gradient boosting. First the number of estimators tuned was between 100 and 500, with increments of 100, whereas the learning rate was tuned between 0.1 and 1 with an incremental increase of 0.1. The second round of tuning was for parameters learning rate and subsample, where the values for subsample ranged from 0.1 to 1 with an incremental increase of 0.1

Tuning for number of estimators and learning rate

The full tables can be found in Appendix B and Figures B.11 and B.13.

estimators	learning rate	Harrell-C	UNO-C	IBS
200	0.6	0.813	0.809	0.121
500	0.3	0.812	0.813	0.122
300	0.5	0.812	0.812	0.123
200	0.8	0.812	0.811	0.123
100	0.6	0.812	0.808	0.119

Table 5.5: 5 best models for OxyTarget dataset using componentwise gradient boosting and tuning for the number of estimators and learning rate.

estimators	learning rate	Harrell-C	UNO-C	IBS
100	0.9	0.769	0.750	0.149
200	0.5	0.768	0.749	0.149
500	0.2	0.768	0.750	0.149
100	0.8	0.768	0.750	0.149
200	0.3	0.768	0.749	0.149

Table 5.6: 5 best models for headneck dataset using componentwise gradient boosting and tuning for the number of estimators and learning rate.

Tuning for estimators, learning rate and subsample

The full tables can found in Appendix B and Figures B.12 and B.14.

estimators	learning rate	subsample	Harrell-C	UNO-C	IBS
300	0.4	0.1	0.815	0.814	0.121
300	0.4	0.4	0.814	0.809	0.121
300	0.4	0.8	0.814	0.811	0.121
300	0.4	0.6	0.813	0.811	0.121
300	0.4	0.2	0.813	0.810	0.120

Table 5.7: 5 best models for OxyTarget dataset using componentwise gradient boosting and tuning for number of estimators, learning rate and subsample.

estimators	learning rate	subsample	Harrell-C	UNO-C	IBS
300	0.4	0.1	0.771	0.752	0.148
300	0.3	0.1	0.768	0.751	0.148
300	0.3	0.2	0.768	0.752	0.149
300	0.3	0.8	0.768	0.750	0.149
200	0.4	0.6	0.768	0.751	0.149

Table 5.8: 5 best models for headneck dataset using componentwise gradient boosting and tuning for number of estimators, learning rate and subsample.

5.3.3 Random survival forest

Parameter tuning was done in two rounds for random survival forest. First the number of estimators (trees) tuned was between 100 and 500, with increments of 100, whereas max_depth

was tuned between 5 and 25 with an incremental increase of 5. The second round of tuning was for parameters min_weight_fraction_leaf and max_depth. The weight fraction ranging from 0.005 to 0.5 and max depth ranging 3 to 25.

Tuning for number of estimators (trees) and max_depth

The full tables can found in Appendix B and Figures B.3 and B.5.

estimators	max_depth	Harrell-C	UNO-C	IBS
300	5	0.807	0.788	0.116
400	5	0.807	0.787	0.117
500	5	0.806	0.786	0.117
500	10	0.804	0.779	0.116
300	10	0.804	0.779	0.116

Table 5.9: 5 best models for OxyTarget dataset using random survival forest and tuning for number of estimators and max_depth.

estimators	max_depth	Harrell-C	UNO-C	IBS
400	3	0.766	0.766	0.149
500	3	0.765	0.765	0.150
300	3	0.764	0.764	0.149
200	3	0.763	0.763	0.150
100	3	0.762	0.762	0.150

Table 5.10: 5 best models for headneck dataset using random survival forest and tuning for number of estimators and max_depth.

Tuning for min_weight_fraction_leaf and max_depth

The full tables can found in Appendix B and Figures B.4 and B.6.

min_weight_fraction_leaf	max_depth	Harrell-C	UNO-C	IBS
0.005	7	0.800	0.776	0.117
0.01	7	0.800	0.776	0.117
0.005	5	0.799	0.776	0.117
0.01	5	0.799	0.776	0.117
0.1	7	0.798	0.800	0.124

Table 5.11: 5 best models for OxyTarget dataset using random survival forest and tuning for min_weight_fraction_leaf and max_depth.

min_weight_fraction_leaf	max_depth	Harrell-C	UNO-C	IBS
0.3	10	0.787	0.769	0.158
0.3	20	0.783	0.768	0.159
0.3	5	0.780	0.763	0.158
0.3	25	0.780	0.763	0.158
0.3	15	0.778	0.763	0.159

Table 5.12: 5 best models for headneck dataset using random survival forest and tuning for min_weight_fraction_leaf and max_depth.

5.3.4 Coxnet

For the Coxnet algorithm, the most essential hyperparameters for the elastic net penalty were used to hyperparameter tune the models, which is the alpha value and l1_ratio. For OxyTarget the value for alpha when performing hyperparameter tuning was between 20 and 0.005 and for l1_ratio it was between 0.01 and 0.9. The same values for alpha were used when tuning on the headneck dataset, but for l1_ratio the values were between 0.01 and 0.95.

Tuning for alpha value and l1_ratio

The full tables can found in Appendix B and Figures B.1 and B.2.

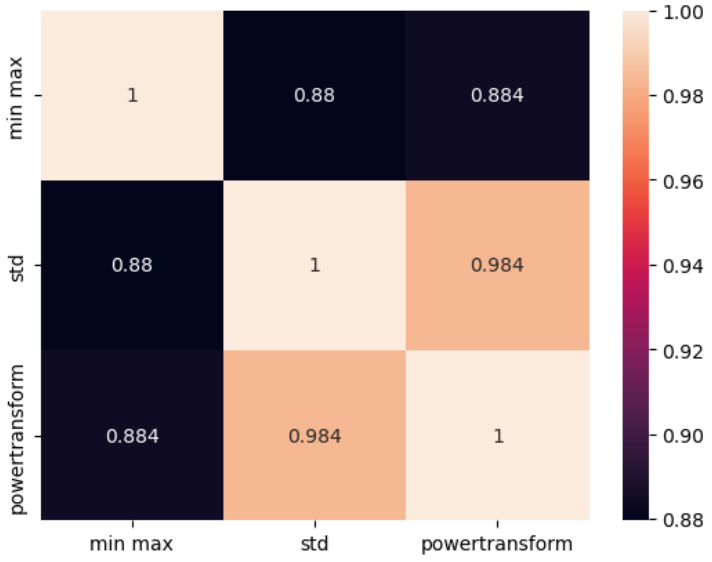
alpha	l1_ratio	Harrell-C	UNO-C	IBS
0.01	0.1	0.827	0.818	0.114
0.01	0.05	0.827	0.817	0.115
0.01	0.01	0.826	0.817	0.115
0.01	0.2	0.825	0.811	0.113
0.01	0.3	0.823	0.811	0.112

Table 5.13: 5 best models for OxyTarget dataset using Coxnet and tuning for alpha value and l1_ratio.

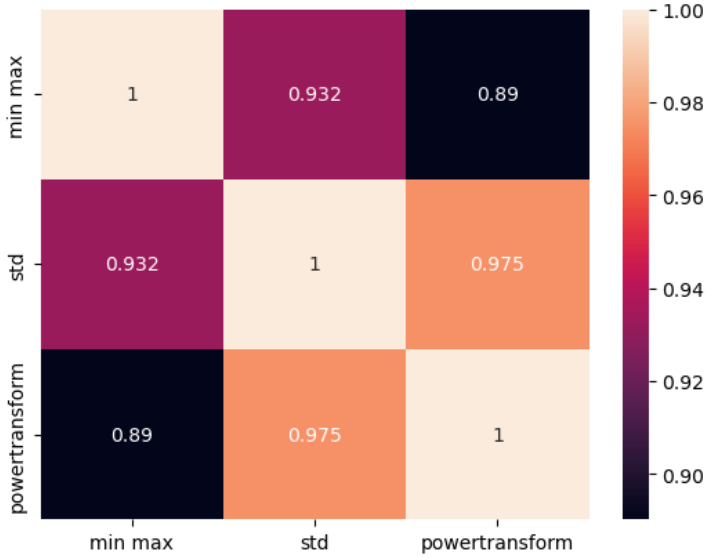
alpha	l1_ratio	Harrell-C	UNO-C	IBS
0.005	0.01	0.771	0.752	0.149
0.005	0.05	0.770	0.752	0.149
0.005	0.2	0.770	0.752	0.149
0.01	0.1	0.770	0.749	0.148
0.005	0.1	0.770	0.752	0.149

Table 5.14: 5 best models for headneck dataset using Coxnet and tuning for number of alphas and l1_ratio.

5.4 RV values for different subsets



(a) OxyTarget



(b) headneck

Figure 5.17: RV computation of different subsets for OxyTarget and headneck dataset.

Figure 5.17, present the computation of RV values between the different subsets of datasets OxyTarget and headneck. RV values give an indication of the similarity in the information given by the matrix [58].

Chapter 6

Discussion

This chapter provides a discussion about the methodology, the results of the models together with the implications faced. Relevant literature using similar feature selection techniques, cross validation techniques and models will be used comparatively to asses the feature selection techniques, scaling, methods and various metrics.

6.1 Different scalings and transformations

The different scaling and transformation methods were used on both datasets, standardscaling, min max scaling and the Yeo-Johnson Powertransform. This resulted in three transformed datasets for each dataset. As seen from the results there was not a significant improvement between any of the subsets on a general level, the biggest difference was seen for Harrell's concordance index where the powertransformed subset had a mean concordance score of 0.77 and the min max scaled subset has a mean score of 0.75 for the OxyTarget dataset.
There might be various reasons as to why the subsets perform similarly in terms of the metrics used. The first is inherent robustness. Survival algorithms like random survival forest, cox proportional hazards and gradient boosting are designed to handle a variety of scales and various distributions in the input data provided. Models which are based on non-parametric or semi-parametric methods are usually not greatly affected by either scale or distribution. Non-parametric models do not have any distributional assumptions from the input data [8]. For example, the tree based models make decisions based on relative feature values and split values, instead of absolute values, which leads to less impact from distributions and scales on performance [30].
Looking at the RV-values which comparatively asses the information similarity between the different transformations in Figure 5.17, it is apparent that the different transformations provide similar information.

6.2 Feature selection and features selected

The most frequently selected features for OxyTarget are the categorical features telling whether patients had suspected metastasized lesions and whether it was a stage 4 cancer (Figure 5.4). These are medical conditions that are critical to be diagnosed with. Metastasis in colon cancer is found to extremely reduce the survival probability [59]. Looking at the thesis by Engesæth (2021) [60], where repeated elastic net (RENT) was used for feature selection on OxyTarget, some similar discoveries where made. In Engesæth's thesis the best performing model to classify 5 year survival, cea_baseline and metastasized lesions were the top selected features [60]. Cea_baseline was the fourth most selected feature overall, and the most selected for coxfilter

and min_depth selection techniques in the experiments conducted in this thesis. CEA stands for carcinoembryonic antigen, which is mostly produced before birth, but often higher levels of CEA are prevalent in colorectal cancers [61]. A study by Hall et al (2019), found higher levels of CEA can often be linked to advanced cancer. Additionally it was discovered that higher levels of CEA combined with stage one to three cancer before operation led to a 62 percent probability increase of death comparative to normal CEA levels [61]. However as concluded in the study, CEA levels alone are not sufficient to predict survival for patients [61]. Additionally, ALP was the third most selected feature. ALP stands for alkaline phosphatase elevation, and elevated levels was found to significantly reduce survival over time compared to normal levels for colorectal cancer patients [62].
The most frequently selected features for headneck are pack_years, hpv_related_0 and uicc8_III_IV_1 (Figure 5.6). Pack years is a measure of how many years the patient has smoked a pack of cigarettes each day, hpv_related_0 indicates that the cancer is not hpv related and uicc8_III_IV_1 indicates it is a stage three or four cancer. Some of the same factors which were declared to be some of the largest risk factors by the National Cancer Institute of the United States government [6]. Of the location based features, oropharynx was the most frequently selected, followed by cavum_oris. This is interesting, since 39 out of the 73 deceased patients had oropharyngal cancer (53.4 %). Although considering that 144 patients were classified as having oropharyngal cancer, 39 were confirmed to be deceased (27.1 %). Considering that 10 out of 12 patients (83.3 %) with oral cavity cancer (cavum_oris), 9 out of 15 (60 %) with hypopharyngeal cancer (hypopharynx) and 15 out of 21 (71.4 %) with laryengeal cancer (larynx) were confirmed deceased, the number for patients with oropharyngal cancer that are confirmed deceased are in comparison proportionally not as high. On the contrary, in a study conducted by Leonici et al (2018) [63], the conditional 5-year survival rate for oropharyngal cancer was the second lowest with hypopharynx having the worst conditional survival rate. Laryngeal cancer had the best 5-year conditional survival rate, followed by oral cavity [63].
In common for both datasets, was that random forest variable hunting had a significantly worse performance than the other feature selection methods. For OxyTarget, the three top selected features by variable hunting, only one was selected in the top 10 features by one of the other selection methods. For headneck we can see some features ranked highly by variable hunting which is not that frequently selected in total by all methods. With the performance loss in mind, it might indicate these features are not as important as other features which were selected more frequently by the other selection methods.

6.2.1 Feature selection and performance

As seen from the results, feature selection in general did not improve any of the metrics of the survival models in most instances, except for Coxnet. An improvement was mostly seen for the regression models with only time as the target variable, especially linear regression and ridge regression. As most of the survival models used and PLSR already have built in feature selection capabilities.
Coxnet selects features and shrinks coefficient weights through the use of l1 and l2 penalization. As the default model with an l1_ratio of 0.5 was used, l1 and l2 was penalized equally. Ranking performance with Harells concordance index was better with three of the feature selection methods on OxyTarget (Figure 5.7) whereas for headneck there was no improvement with feature selection (Figure 5.8), although results were near equal for three feature selection techniques and the model having all features available. The same pattern was seen when evaluating ranking using UNO's C-statistic, feature selection drastically improved performance on OxyTarget (Figure 5.9). For headneck there was a slight improvement with random forest variable importance feature selection (Figure 5.10). For UNO's C-statistic with truncation, random forest minimal depth slightly improved the performance on the headneck dataset (Figure 5.12). Especially when evaluating how well the model is calibrated, having all features available on OxyTarget

led to a massive drop in the Integrated Brier Score for Coxnet (Figure 5.13). Looking at the results mentioned, the feature selection techniques used except for random forest variable hunting, might lead to equal or overall better performance for the Coxnet algorithm. Similarly in the findings by Spooner et al (2020) [7], the elastic net model did perform better with most external feature selection techniques for one of the datasets, whereas for the other dataset used, the feature selection techniques performed near equally as good.
Feature selection did not improve the performance of random survival forest. In the article from Spooner et.al(2020) [7], similar characteristics are found for the feature selection methods for Harrell's concordance index. The scores are very close for multiple of the methods, although there is a miniscule increase in performance when using random forest variable importance to select features beforehand. There are multiple possible reasons that can account for the lack of performance increase when applying feature selection. Random survival forest perform feature selection by selecting a random subset of features at each split within the trees. Randomization prevents correlation between the survival trees [64], which in result counteracts overfitting. The diverse set of subsets containing features for each split, reduces the need for external feature selection methods [64]. To further assess the lack of added performance with feature selection for random survival forest, one can look towards the splitting criterion used for each survival tree, which is the log-rank splitting criterion. Log-rank considers both the time-to-event information as well as the survival distribution when it determines what the best possible split is [65]. This means that applying external feature selection methods before fitting the random survival forest algorithm might not be necessary, as the survival aspects relevant to each feature already are incorporated when the splits are calculated.
The tree based gradient boosting algorithm did not have any increase in ranking performance with external feature selection, except for random forest variable importance which slightly increased Harrell's concordance index and UNO's C-statistic on the headneck dataset. The increase using random forest variable importance in the aforementioned scenarios was only 0.01. Similarly Spooner et al (2020) [7] did not observe any improvement for a gradient boosted cox model by adding feature selection, except for a slight increase with random forest minimum depth feature selection. Some of the same reasoning when evaluating feature selection performance with random forest can be used to explain the lack of performance increase with added feature selection methods for the tree based gradient boosting algorithm. The regression trees also select a random subset of features for each tree, increasing the model robustness [66]. Tree based models are capable of capturing complex interactions between features automatically, and many external feature selection methods may not consider these interactions. Relevant features or features having combined effects might have been removed. Tree based gradient boosted models are good at handling noisy features, and therefore external feature selection methods are oftentimes not needed. Although not survival data was used, in a study by Chen et al.(2016) [67], it was shown that tree based boosting models benefited from greedily using information from the full dataset rather than fitting subsets of smaller dimensions.
Similarly there is no performance increase seen by combining feature selection methods with the componentwise gradient boosted model using partial least squares as the base learner. There is one exception, where the truncated UNO C-statistic is increased by 0.01 using the headneck data with random forest minimal depth selection. This is such a small increase for only one metric, meaning it can be considered to perform near equally as good. The componentwise gradient boosted model already has feature selection as an integral part of the algorithm, as mentioned in section 2.4.2, for each boosting iteration the coefficient for one feature gets its weights updated. This implies that irrelevant features and noisy features might get very low weights or might not be weighted at all.

6.3 Hyperparameter tuning of survival models

All of the survival models were tuned with different hyperparameters to check whether it could increase the performance. As the feature selection methods used in the first experiment yielded no or marginal increase in performance, the models were tuned with all features available on the min_max scaled subsets. Similar to the first experiment, the models were tuned with repeated stratified k-fold cross validation, where the average performance metrics were computed across all splits. When performing hyperparameter tuning each split contained 4 folds and was repeated 4 times, meaning the test proportion was set to 25 percent of the full dataset for each split. A similar configuration to when feature selection techniques were applied, only with one less repeat.

6.3.1 Coxnet

For Coxnet, the elastic net hyperparameters alpha and l1_ratio were tuned. l1_ratio is the mixing parameter for the elastic net penalties Ridge (L2) and Lasso (L1), the smaller the l1_ratio, the smaller the influence of the L1 penalty is compared to L2 penalty. Hence, the smaller the l1_ratio is, there is a stronger tendency for feature shrinkage and less tendency for dimensionality reduction. Alpha on the other hand is the penalization or regularization strength, a greater value for alpha results in greater shrinkage of feature weights and leads to a more sparse set of features fitted onto the model. The regularization strength alpha is the same as r in equation 2.16. Before the search for the optimal combination of hyperparameter values, the value of alpha that would set all feature weights to zero was unknown. Therefore multiple increasing alpha values were tried until the value of alpha that would set all feature weights to zero, independently of the l1_ratio. Then the final list of alpha values to tune for was set by choosing the alpha value which resulted in complete shrinkage of weights and adding more values in decreasing order until feature weights would be too large for the algorithm to handle. From the results in Appendix B and Figure B.1 and B.2, alpha values of 1 and above tend to score 0.5 for concordance which means the model guesses randomly. This is can possibly be an effect due to the L1 penalty setting all feature weights to 0. Furthermore, hyperparameter tuning for lower values of alpha and l1_ratio resulted in a large increase in performance for OxyTarget compared to the base model without tuning. With all features available, the performance for Harrell's concordance index increased from 0.73 to 0.827 (Figure 5.7 & Table 5.13), a large increase was seen for UNO's C-statistic as well and the Integrated Brier Score. For OxyTarget the best hyperparameter combination is a small alpha value of 0.01 and an l1_ratio of 0.1. This indicates that the penalization strength is not very high and that the Ridge penalty has highest influence, which might indicate that shrinkage of a few feature weights is able to remove the most degenerate behaviour leading to the performance increase.
With regards to the hyperparameters, the same trend was seen on the headneck dataset, where a small value for alpha and l1_ratio yielded the best performance. With the best hyperparameter combination being 0.005 for alpha and 0.01 for l1_ratio. However, compared to the default models when feature selection was tested, from the results it is clear that the tuned model have very similar performance to the model having all features available and for some of the models with external feature selection applied beforehand. The best performance after hyperparameter tuning for Coxnet was 0.771 for Harrell's concordance index (Table 5.14). Compared to OxyTarget, the headneck dataset is much more sparse, and shrinking feature weights might not be as effective as for some high-dimensional datasets.

6.3.2 Random Survival Forest

Hyperparameter tuning for number of estimators and maximum depth yielded a very small increase in terms of ranking when looking at Harrell's concordance index. For OxyTarget UNO's

C-statistic was slightly worse for the model with the highest Harrell concordance index. For headneck UNO's C-statistic was slightly increased. Tuning for these parameters no significant change in terms of Integrated Brier Score was observed. The best hyperparameter combinations for both dataset were with over 300 estimators and a small maximum depth.
Tuning for min_weight_fraction_leaf and maximum depth yielded no better performance for OxyTarget, but lead to an increase in ranking performance for the headneck dataset. Integrated Brier Score was slightly worse for the best hyperparameters in terms of ranking for headneck. For OxyTarget the best hyperparameter combinations consisted of a small weight fraction and a small maximum depth. Whereas for headneck the best combinations consisted of a weight fraction of 0.3, where the best model had a maximum depth of 10. With hyperparameter tuning, Random Survival Forest ended up as the best performing model for the headneck dataset overall. Scikit-survival documentation explains min_weight_fraction_leaf as "The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. [31]". In other words, a threshold for minimum cumulative sample weight to be present at each leaf node. A lower value will in turn lead to deeper tree growth, imposing a risk of overfitting. Although as can be seen from the best hyperparameter combinations, a small maximum depth is favoured. Despite yielding slightly better ranking performance for the headneck dataset, The Integrated Brier Score increased slightly as well, resulting in a model which is slightly worse at generalizing.

6.3.3 Gradient boosting with regression trees as base learner

Hyperparameter tuning for the number of estimators and learning rate did not yield any significant increase in performance on OxyTarget. The best combination of hyperparameters is with 300 estimators and a learning rate of 1. The best hyperparameter combinations when tuned for Harrell's concordance index had a learning rate of 1. While it increases ranking performance, the Integrated Brier Score gets larger. Looking at Figure B.7 in Appendix B, it is evident that a better Integrated Brier Score can be achieved by setting the learning rate lower.
On the headneck dataset, the hyperparameter tuning yielded a significant increase in terms of ranking metrics. Where the best hyperparameter combinations consisted of a small learning rate and a small number of estimators. The best hyperparameter combination of 100 estimators and a learning rate of 0.1 achieved a slightly better Integrated Brier Score.
Hyperparameter tuning was additionally extended by adding to more parameters, hence tuning for number of estimators, learning rate, min_weight_fraction_leaf and maximum depth. This increased the performance slightly further. For OxyTarget the ranking metrics did not improve when compared to the first experiment, but the Integrated Brier Score did. The best parameter combinations consisted of 100 estimators, a learning rate of 0.4, a min_weight_fraction_leaf of 0.4 and max depth between 2 and 10.
On the headneck dataset tuning for the same four parameters resulted in a significant increase in ranking metrics compared to the first experiment and the previous tuning attempt. The model performed slighty worse in terms of Integreated Brier Score with a minuscule increase of the score. The best hyperparameters consisted of a small number of estimators, a small learning rate, a min_weight_fraction_leaf of 0.3 and a max depth between 2 and 10. What is apparent from the results, is that introducing a min_weight_fraction_leaf of 0.3 improves the performance a lot.

6.3.4 Componentwise gradient boosting with partial least squares as base learner

Hyperparameter tuning the number of estimators and learning rate resulted in better ranking performance for both datasets. Both Harrell's concordance index and UNO's C-statistic is higher while Integrated Brier Score is equally good when compared to the first experiment. The

best parameter combination for OxyTarget consisted of 200 estimators and a learning rate of 0.6, while for headneck it was 100 estimators and a learning rate of 0.9. There is no clear trend among the best hyperparameter combinations. Additionally tuning the subsample proportion, yielded a very small increase in performance as well, in terms of ranking. The best tuned models for both datasets contained a subsample value of 0.1. In general, for both experiments and all models, hyperparameter tuning the componentwise gradient boosted model resulted in overall best performance for OxyTarget.
Setting the subsample parameter to a low value means that each base learner is trained on a small proportion of the data. Introducing this diversity of samples to the base learner can help with better performance. The base learner might be able to capture different aspects and patterns within the data when the subsampling is low. The effect can tend to reduce variability in the final boosted model [66]

6.4 The performance of the models used

Of the boosted models the best performing model observed Spooner et al (2020) [7] was likelihood based boosting with cox models as the base learner. Although for one of the datasets, she observed near identical performance using gradient boosting with linear models as base learners. In the experiments conducted in this thesis, a better performance was overall achieved using gradient boosting with partial least squares as the base learner. Spooner et al (2020) [7] did observe a greater performance relative to the other models using elastic net, whereas the experiments in this thesis showed that an elastic net penalized cox model turned out to be the worst performer among the survival models. What is clear from the results achieved in this thesis, is that there is no clear best algorithm to use. The best performers for the headneck dataset is hyperparameter tuned Random Survival Forest models or Coxnet models, which both achieved a concordance of 0.771. For the OxyTarget dataset when hyperparameter tuning on the min_max scaled subset the best performing model was with the Coxnet algorithm tuned for l1_ratio and regularization strength parameter alpha, which achieved a concordance of 0.827. The second best was a componentwise gradient boosted model using partial least squares, which resulted in a concordance index of 0.815. Based on these results the component wise gradient boosting algorithm with partial least squares or the Coxnet algorithm are preferred for the high-dimensional OxyTarget data, whereas random survival forest is preferred for the more sparse headneck dataset as it was able to score 0.787 for Harrell's concordance index. Although if we look at the other subsets for OxyTarget without any hyperparameter tuning, the best performing model is Random Survival Forest on the powertransformed subset with a concordance index of 0.83 (Figure A.2). This yielded the overall highest concordance index for OxyTarget together with Coxnet. Although considering UNO's C-statistic and UNO's truncated C-statistic for the powertransformed subset, there is a slight drop to 0.81 for Random Survival Forest, which might indicate that there is a slight overestimation of Harrell's concordance index for Random Survival forest on the powertransformed subset.
In general, from the first experiment, it is apparent that some survival machine learning models which are able to handle censored data are better at ranking the survival times of samples better than the linear regression and Ridge regression. Nevertheless, Root mean square error (RMSE), show that Linear regression is imprecise in its prediction with all features available and with random forest minimal depth on OxyTarget data. The difference in RMSE for headneck and between PLSR and Ridge on OxyTarget is so small it is, negligible. In the findings of Spooner et al (2020) [7], there was a lack of added performance for many models in many instances when utilizing feature selection techniques. Similarly in this thesis, feature selection did not increase performance for most of the machine learning models.

6.5 Bias due to high proportion of censoring

Both UNO's C-statistic and UNO's C-statistic with truncation were used to assess all the survival models. With a high proportion of censoring in the test data, Harell's concordance index is known to be biased upwards [50], resulting in higher scores for the concordance index. When calculating Harrell's concordance index, all pairs where the sample with the shortest survival time is censored are ignored [68]. This might lead to an upwards bias when the censoring proportion is high in the test set. To counteract this bias to some extent UNO's C-statistic and truncated C-statistic can be used. The last part of the survival function can tend to be unstable [50], and therefore the truncated statistic was truncated to the last confirmed failure time. The last observed death for OxyTarget is at 68 months, whereas for the headneck dataset it is observed at 87.97 months. Truncation time τ should be set such that $P(D > \tau) > 0$, which implies that the probability to be censored after truncation is nonzero [50][31].
Both UNO's C-statistic with and without truncation is very close to the score for Harrells's concordance index for the OxyTarget dataset. However for the headneck dataset there is a slight drop in performance when looking at UNO's C-statistic. What is interesting is that the censoring proportion is higher for OxyTarget with 72 percent, compared to 62.2 percent for headneck.

6.6 Cross validation to evaluate feature selection techniques and method of searching for optimal hyperparameters

Repeated stratified K-fold cross validation (RSKF) was used when both assessing different feature selections techniques and when searching for optimal hyperparameters. One of the reasons to use RSKF is robustness against data variability as it reduces the impact of random data variations by repeating the cross validations [69]. It ensures that each data point has been in the training and test folds, hence making sure that all data is used and utilised to the largest possible extent. The robustness is important when searching for hyperparameters as it minimizes the influence a random data split can have. The optimal hyperparameters found will generalize better to unseen data as it mitigates the risk of overfitting for a particular train and test fold or subset of the data used [69]. As the hyperparameters are averaged from multiple splits the hyperparameter combinations provide a more representative estimate of the impact on the model used. When evaluating the feature selection techniques the RSKF was set up with 5 splits and 5 repeats, which is exactly the same configuration used by Spooner et al (2020) [7]. It is important to mention that when searching for the optimal hyperparameters, RSKF was configured with 4 splits and 4 repeats. This is the same configuration as illustrated for the first experiment in Figure 4.7, but one less repeat. The reason to use one less repeat was due to computation time and that Coxnet became very unstable for particular splits. As the number of splits was increased and due to more repeats, there were more splits which created problems for survival estimates from the Coxnet model. This further created difficulties when survival estimates were predicted in order to calculate the Integrated Brier Score.

6.7 Issues faced and limitations

6.7.1 Unstable predictions with Coxnet

Mostly the issues faced were with the Coxnet model providing very unstable survival estimates for OxyTarget at specific splits from the repeated stratified k-fold. The survival estimates for certain samples in certain splits would be estimated as infinitely low or infinitely large values as time increased. Therefore in order to have comparable Integrated Brier Scores, the scores were calculated between 12 and 48 months, which corresponds to between 1 year and 4 years.

Predictions were stable for the headneck dataset, therefore the Brier score was truncated to between 1 and 5 years, as 5-year survival is common to assess in clinical studies.

6.7.2 Other loss functions for gradient boosted models

The gradient boosted models and Coxnet all use cox proportional hazards as its loss function. The gradient boosted models from sci-kit survival do have the ability to use other loss functions such as squared loss or inverse probability of censoring weighted least squares (ipcwls). It would be interesting to investigate if performance could be improved using one of the other loss functions. The limitation with the gradient boosted models from scikit-survival is that the models are not able to predict a survival curve with loss set to squared or ipcwls. Therefore the Integrated Brier Score can not be calculated, to asses the calibration error compared to the other models used.

6.7.3 Features selected by the different algorithms

The most important features of the different algorithms were not assessed in this thesis due to limitations with scikit-survival. For the gradient boosted models in addition to coxnet it is possible to extract the feature weights, however for random survival forest implementation from scikit-survival it is not possible to do so.

6.8 Suggestions for future work

6.8.1 Impute right censored samples

As the regression models Linear Regression, Ridge Regression and PLSR do not take into account censoring, which might lead to an underestimation in the response (since the censored patients live longer than the response says). One approach could be to use conditional survival distributions (CondiS) which is a method derived from Kaplan-Meier [70], to impute the time of survival for right censored patients. This method allows the application of ML-models which do not handle censoring on the imputed data, without losing all the information from the censoring variable. In the article by Wang et al (2022) [70], where the method was discovered, the use of CondiS to impute time variables showed improvement in prediction error and concordance.

6.8.2 Different or additional approach to hyperparameter tuning

On the condition that there would be more samples, hyperparameter tuning could have been done differently, for example by splitting the full dataset into a training portion, validation portion and a test portion. For gradient boosting and random survival forest, different values for various hyperparameters over the number of estimators or trees could have been plotted for various metrics on the training data with stratified k-fold cross validation. And for the gradient boosted methods, an early stopping monitor could be implemented, assessing the average improvement across the boosting iterations. This means that after a number of boosting iterations, for example 20, if there was no improvement on average, the most suitable number of iterations could be determined.
For the Coxnet model, utilizing a pipeline and stratified k-fold cross validation on the train and validation portion of the data, values for alpha potentially could have been estimated more precisely by visualising the regularization path for various metrics. This can be achieved by using the hyperparameters alpha_min_ratio and setting a high number of alphas to be calculated, in order to find the maximum score for alpha. By using alpha_min_ratio the alpha needed to set all coefficients to zero is calculated first (maximum alpha) and then gradually decreased until until a percentage value of the inital alpha value where all coefficients are nulled is reached,

the percentage value is controlled by the hyperparameter. Alpha_min_ratio multiplied with the maximum number of alphas will determine the minimum number of alphas and the algorithm will compute for a number of equally spaced alphas in between the maximum and minimum alphas [31]. Alpha is the regularization parameter, and by visualising the regularization path, the strength of the regularization can be thoroughly assessed along the path between maximum and minimum alpha [31]. Additionally feature weights could be extracted improving the interpretability of the model for clinical use to researchers and doctors, which can further assess the feature weights.

Chapter 7

Conclusion

The goals of this thesis consisted of comparing various machine learning algorithms for various metrics on cancer data and testing several pre-processing tools such as different transformations and feature selection techniques. From the results and additional analysis done in this thesis, it can be concluded that there is no universal best approach for performing survival analysis on cancer data. However, the results indicate that some models have better predictive ability, both in terms of ranking performance and were better calibrated, indicating a higher ability to generalize to unseen data. The different trasformations yielded no increase in results for the headneck datasets, but a small increase in performance was seen with Yeo-Johnson powertransform on OxyTarget. For the models with default parameters, feature selection did not improve the performance of the survival models except for Coxnet. However feature selection did in most instances improve performance of Linear regression and Ridge regression significantly. Despite the lack of performance for several of the survival models, some of the most frequently selected features can be supported by published studies and articles. However, also backed by published literature and the findings of this thesis, is that advanced machine learning algorithms possibly can benefit from greedily selecting features rather than using feature selection techniques as part of pre-processing.
Random survival forest scored well for both datasets, the high dimensional OxyTarget dataset and the more sparse headneck dataset. For Harrell's concordance index the algorithm scored 0.83 with powertransform applied beforehand on OxyTarget and 0.787 for the headneck dataset with hyperparameter tuning. Secondly Coxnet performed as well for OxyTarget with a slightly higher score for UNO's C-statistic compare to Random Survival Forest, whereas for headneck the concordance index was 0.771. Thirdly the component wise gradient boosted model with partial least squares as base learner performed relatively well for both datasets with concordance index scores of 0.813 for OxyTarget and 0.769 for OxyTarget. Of the regression models with only time as the target variable, PLSR performed better than Ridge regression and Linear regression. Hyperparameter tuning the survival models did in most instances not change the Integrated Brier Score significantly, for the models with the best ranking performance.

Bibliography

[1] Cancer Registry of Norway, May 2022. URL: https://www.kreftregisteret.no/globalassets/cancer-in-norway/2022/cin_report-2022.pdf.

[2] URL: https://www.who.int/news-room/fact-sheets/detail/cancer.

[3] Mark Gormley et al. "Reviewing the epidemiology of head and neck cancer: definitions, trends and risk factors". In: *British Dental Journal* 233.9 (Nov. 2022), pp. 780–786. ISSN: 1476-5373. DOI: 10.1038/s41415-022-5166-x.

[4] Geoffrey M. Cooper. *The Cell: A Molecular Approach. 2nd edition.* eng. Sinauer Associates 2000, 2000. ISBN: 0-87893-106-6. URL: https://www.ncbi.nlm.nih.gov/books/NBK9963/.

[5] PDQ Adult Treatment Editorial Board. "Rectal Cancer Treatment (PDQ®): Health Professional Version". en. In: *PDQ Cancer Information Summaries*. Bethesda (MD): National Cancer Institute (US), 2002.

[6] URL: https://www.cancer.gov/types/head-and-neck/head-neck-fact-sheet.

[7] Annette Spooner et al. "A comparison of machine learning methods for survival analysis of high-dimensional clinical data for dementia prediction". en. In: *Scientific Reports* 10.1 (Nov. 2020), p. 20410. ISSN: 2045-2322. DOI: 10.1038/s41598-020-77220-w.

[8] Christiana Kartsonaki. "Survival analysis". In: *Diagnostic Histopathology* 22.7 (2016). Mini-Symposium: Medical Statistics, pp. 263–270. ISSN: 1756-2317. DOI: https://doi.org/10.1016/j.mpdhp.2016.06.005. URL: https://www.sciencedirect.com/science/article/pii/S1756231716300639.

[9] David G. Kleinbaum and Mitchel Klein. *Survival Analysis: A Self-Learning Text, Third Edition (Statistics for Biology and Health)*. Springer, 2012, pp. 5–18. ISBN: 978-1-4939-5018-8. URL: https://link.springer.com/book/10.1007/978-1-4419-6646-9.

[10] E. L. Kaplan and Paul Meier. "Nonparametric Estimation from Incomplete Observations". In: *Journal of the American Statistical Association* 53.282 (1958), pp. 457–481. ISSN: 01621459. URL: http://www.jstor.org/stable/2281868 (visited on 05/20/2023).

[11] D. R. Cox. "Regression Models and Life-Tables". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 34.2 (1972), pp. 187–220. ISSN: 00359246. URL: http://www.jstor.org/stable/2985181 (visited on 04/11/2023).

[12] H. Ishwaran et al. "Random survival forests". In: *Ann. Appl. Statist.* 2.3 (2008), pp. 841–860. URL: https://arXiv.org/abs/0811.1645v1.

[13] Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22. DOI: 10.18637/jss.v033.i01. URL: https://www.jstatsoft.org/index.php/jss/article/view/v033i01.

[14] Kwan-Moon Leung, Robert M. Elashoff, and Abdelmonem A. Afifi. "CENSORING ISSUES IN SURVIVAL ANALYSIS". In: *Annual Review of Public Health* 18.1 (1997). PMID: 9143713, pp. 83–104. DOI: 10.1146/annurev.publhealth.18.1.83. eprint: https://doi.org/10.1146/annurev.publhealth.18.1.83. URL: https://doi.org/10.1146/annurev.publhealth.18.1.83.

[15] J. D. Kalbfleisch and Ross L. Prentice. *The statistical analysis of Failure Time Data*. 2nd ed. John Wiley, 2002.

[16] Terry M. Therneau and Patricia M. Grambsch. "The Cox Model". In: *Modeling Survival Data: Extending the Cox Model*. New York, NY: Springer New York, 2000, pp. 39–77. ISBN: 978-1-4757-3294-8. DOI: 10.1007/978-1-4757-3294-8_3. URL: https://doi.org/10.1007/978-1-4757-3294-8_3.

[17] Nicolo Cosimo Albanese. *Survival Analysis: Optimize the Partial Likelihood of the Cox Model*. en. Dec. 2022. URL: https://towardsdatascience.com/survival-analysis-optimize-the-partial-likelihood-of-the-cox-model-b56b8f112401.

[18] Christopher M Wilson et al. "Fenchel duality of Cox partial likelihood with an application in survival kernel learning". en. In: *Artif Intell Med* 116 (Apr. 2021), p. 102077.

[19] Christopher M. Wilson et al. "Fenchel duality of Cox partial likelihood with an application in survival kernel learning". In: *Artificial Intelligence in Medicine* 116 (2021), p. 102077. ISSN: 0933-3657. DOI: https://doi.org/10.1016/j.artmed.2021.102077. URL: https://www.sciencedirect.com/science/article/pii/S0933365721000701.

[20] Charu C. Aggarwal. *Outlier Analysis*. 2nd. Springer Publishing Company, Incorporated, 2016. ISBN: 3319475770.

[21] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation-based anomaly detection". English. In: *ACM Transactions on Knowledge Discovery from Data* 6.1 (2012), pp. 1–39. ISSN: 1556-4681. DOI: 10.1145/2133360.2133363.

[22] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[23] URL: https://statistics.berkeley.edu/research/high-dimensional-data-analysis.

[24] Takeshi Emura and Yi-Hau Chen. "Gene selection for survival data under dependent censoring: A copula-based approach". en. In: *Stat Methods Med Res* 25.6 (May 2014), pp. 2840–2857.

[25] Hemant Ishwaran. "Variable importance in binary regression trees and forests". In: *Electronic Journal of Statistics* 1.none (2007), pp. 519–537. DOI: 10.1214/07-EJS039. URL: https://doi.org/10.1214/07-EJS039.

[26] Hemant Ishwaran et al. "High-Dimensional Variable Selection for Survival Data". In: *Journal of the American Statistical Association* 105.489 (2010), pp. 205–217. DOI: 10.1198/jasa.2009.tm08622. eprint: https://doi.org/10.1198/jasa.2009.tm08622. URL: https://doi.org/10.1198/jasa.2009.tm08622.

[27] Hong Wang and Gang Li. "A Selective Review on Random Survival Forests for High Dimensional Data". In: *Quantitative bio-science* 36.2 (2017), pp. 85–96. ISSN: 2288-1344. DOI: 10.22283/qbs.2017.36.2.85.

[28] Chris Ding and Hanchuan Peng. "Minimum redundancy feature selection from microarray gene expression data". en. In: *J Bioinform Comput Biol* 3.2 (Apr. 2005), pp. 185–205.

[29] Sebastian Raschka and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with python, scikit-learn, and TensorFlow 2*. 3rd ed. Birmingham, England: Packt Publishing, 2019. ISBN: 9781789955750.

[30] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The elements of Statistical Learning: Data Mining, Inference, and prediction, second edition.* 2nd ed. Springer, 2009. URL: https://link.springer.com/book/10.1007/978-0-387-84858-7.

[31] Sebastian Pölsterl. "scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn". In: *Journal of Machine Learning Research* 21.212 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/20-729.html.

[32] H. Ishwaran and U.B. Kogalur. "Random survival forests for R". In: *R News* 7.2 (Oct. 2007), pp. 25–31. URL: https://journal.r-project.org/articles/RN-2007-015/RN-2007-015.pdf.

[33] Torsten Hothorn et al. "Survival ensembles". In: *Biostatistics* 7.3 (July 2006), pp. 355–373. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxj011.

[34] Sanjay Lall and Stephen Boyd. *EE104 Empirical Risk Minimization.* Apr. 2023. URL: https://ee104.stanford.edu/lectures/erm.pdf.

[35] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine." In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451. URL: https://doi.org/10.1214/aos/1013203451.

[36] Greg Ridgeway. "The State of Boosting". In: *Comp Sci Stat* 31 (Dec. 2001).

[37] Greg Ridgeway. "Generalized Boosted Models: A guide to the gbm package". In: (2020). URL: https://cran.rproject.org/web/packages/gbm/vignettes/gbm.pdf (visited on 04/11/2023).

[38] Nam Phuong Nguyen. "The Paradox of OverfittingGradient Boosting for Survival Analysis with Applications in Oncology". MA thesis. University of South Florida, 2019. URL: https://digitalcommons.usf.edu/etd/8062/.

[39] Peter Bühlmann. "Boosting for high-dimensional linear models". In: *The Annals of Statistics* 34.2 (2006), pp. 559–583. DOI: 10.1214/009053606000000092. URL: https://doi.org/10.1214/009053606000000092.

[40] Harald Binder and Martin Schumacher. "Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models". In: *BMC Bioinformatics* 9.1 (Jan. 2008), p. 14. ISSN: 1471-2105. DOI: 10.1186/1471-2105-9-14.

[41] Noah Simon et al. "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent". en. In: *J Stat Softw* 39.5 (Mar. 2011), pp. 1–13.

[42] Noah Simon et al. "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent". en. In: *J Stat Softw* 39.5 (Mar. 2011), pp. 1–13.

[43] David G. Kleinbaum et al. *Applied regression analysis and other multivariable methods, 3rd ed.* Applied regression analysis and other multivariable methods, 3rd ed. Belmont, CA, US: Thomson Brooks/Cole Publishing Co, 1998, pp. xviii, 798. ISBN: 0-534-20910-6.

[44] Bozena Zdaniuk. "Ordinary Least-Squares (OLS) Model". In: *Encyclopedia of Quality of Life and Well-Being Research.* Ed. by Alex C. Michalos. Dordrecht: Springer Netherlands, 2014, pp. 4515–4517. ISBN: 978-94-007-0753-5. DOI: 10.1007/978-94-007-0753-5_2008. URL: https://doi.org/10.1007/978-94-007-0753-5_2008.

[45] Hervé Abdi. "Partial least squares regression and projection on latent structure regression (PLS Regression)". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (Jan. 2010), pp. 97–106. DOI: 10.1002/wics.51.

[46] Kevin Dunn. URL: https://learnche.org/pid/PID.pdf?10d109.

[47] Ulf Indahl. "The geometry of PLS1 explained properly: 10 key notes on mathematical properties of and some alternative algorithmic approaches to PLS1 modelling". In: *Journal of Chemometrics* 28 (Mar. 2014). DOI: 10.1002/cem.2589.

[48] F E Harrell Jr, K L Lee, and D B Mark. "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors". en. In: *Stat Med* 15.4 (Feb. 1996), pp. 361–387.

[49] Enrico Longato, Martina Vettoretti, and Barbara Di Camillo. "A practical perspective on the concordance index for the evaluation and selection of prognostic time-to-event models". In: *Journal of Biomedical Informatics* 108 (2020), p. 103496. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2020.103496. URL: https://www.sciencedirect.com/science/article/pii/S1532046420301246.

[50] Hajime Uno et al. "On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data". en. In: *Stat Med* 30.10 (Jan. 2011), pp. 1105–1117.

[51] GLENN W. BRIER. "VERIFICATION OF FORECASTS EXPRESSED IN TERMS OF PROBABILITY". en. In: *Monthly Weather Review* 78.1 (Jan. 1950), pp. 1–3. DOI: 10.1175/1520-0493(1950)078<0001:vofeit>2.0.co;2. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=feee6551179612b9691f021b583d8a99b81b9b86.

[52] E Graf et al. "Assessment and comparison of prognostic classification schemes for survival data". en. In: *Stat Med* 18.17-18 (1999), pp. 2529–2545.

[53] Timothy O. Hodson. "Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not". en. In: *Geoscientific Model Development* 15.14 (July 2022), pp. 5481–5487. ISSN: 1991-9603. DOI: 10.5194/gmd-15-5481-2022.

[54] In-Kwon Yeo and Richard A. Johnson. "A New Family of Power Transformations to Improve Normality or Symmetry". In: *Biometrika* 87.4 (2000), pp. 954–959. ISSN: 00063444. URL: http://www.jstor.org/stable/2673623 (visited on 04/23/2023).

[55] Olga Troyanskaya et al. "Missing value estimation methods for DNA microarrays". In: *Bioinformatics* 17.6 (June 2001), pp. 520–525. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/17.6.520.

[56] Kathrine Røe Redalen. *The oxytarget study – merging functional MRI and circulating biomarkers for biopsy-free detection of chemoradiotherapy resistant rectal cancer*. URL: https://forskningsprosjekter.ihelse.net/prosjekt/2013002.

[57] Jon Magne Moan et al. "The prognostic role of 18F-fluorodeoxyglucose PET in head and neck cancer depends on HPV status". en. In: *Radiother Oncol* 140 (June 2019), pp. 54–61.

[58] Oliver Tomic et al. "hoggorm: a python library for explorative multivariate statistics". In: *The Journal of Open Source Software* 4.39 (2019). DOI: 10.21105/joss.00980. URL: http://joss.theoj.org/papers/10.21105/joss.00980.

[59] Weixing Dai et al. "Prognostic and predictive value of radiomics signatures in stage I-III colon cancer". en. In: *Clin Transl Med* 10.1 (Jan. 2020), pp. 288–293.

[60] Lars Jetmund Svartis Engesæth. "Predicting Patient Outcome Using Radioclinical Features Selected With RENT for Patients With Colorectal Cancer". Elizabeth Stephansens v. 15, 1430 Ås: Norwegian University of Life Sciences, 2022. URL: https://nmbu.brage.unit.no/nmbu-xmlui/bitstream/handle/11250/3036071/Engesaeth2022_merged.pdf?sequence=1&isAllowed=y.

[61] Claire Hall et al. "A Review of the Role of Carcinoembryonic Antigen in Clinical Practice". en. In: *Ann Coloproctol* 35.6 (Dec. 2019), pp. 294–305.

[62] Hsin-Yuan Hung et al. "Preoperative alkaline phosphatase elevation was associated with poor survival in colorectal cancer patients". en. In: *Int J Colorectal Dis* 32.12 (Oct. 2017), pp. 1775–1778.

[63] Emanuele Leoncini et al. "Tumour stage and gender predict recurrence and second primary malignancies in head and neck cancer: a multicentre study within the INHANCE consortium". In: *European Journal of Epidemiology* 33.12 (Dec. 2018), pp. 1205–1218.

[64] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[65] Michael LeBlanc and John Crowley. "Survival Trees by Goodness of Split". In: *Journal of the American Statistical Association* 88.422 (1993), pp. 457–467. ISSN: 01621459. URL: http://www.jstor.org/stable/2290325 (visited on 05/24/2023).

[66] Jerome H. Friedman. "Stochastic gradient boosting". In: *Computational Statistics Data Analysis* 38.4 (2002). Nonlinear Methods and Data Mining, pp. 367–378. ISSN: 0167-9473. DOI: https://doi.org/10.1016/S0167-9473(01)00065-2. URL: https://www.sciencedirect.com/science/article/pii/S0167947301000652.

[67] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785. URL: https://arxiv.org/abs/1603.02754.

[68] M Shafiqur Rahman et al. "Review and evaluation of performance measures for survival prediction models in external validation settings". en. In: *BMC Med Res Methodol* 17.1 (Apr. 2017), p. 60.

[69] Sebastian Raschka. *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning*. 2020. arXiv: 1811.12808 [cs.LG].

[70] Yizhuo Wang et al. "CondiS: A conditional survival distribution-based method for censored data imputation overcoming the hurdle in machine learning-based survival analysis". In: *Journal of Biomedical Informatics* 131 (2022), p. 104117. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2022.104117. URL: https://www.sciencedirect.com/science/article/pii/S1532046422001332.

Appendix A

Heatmaps of metrics for feature selection methods

A.1 Harrells concordance index

A.1.1 OxyTarget

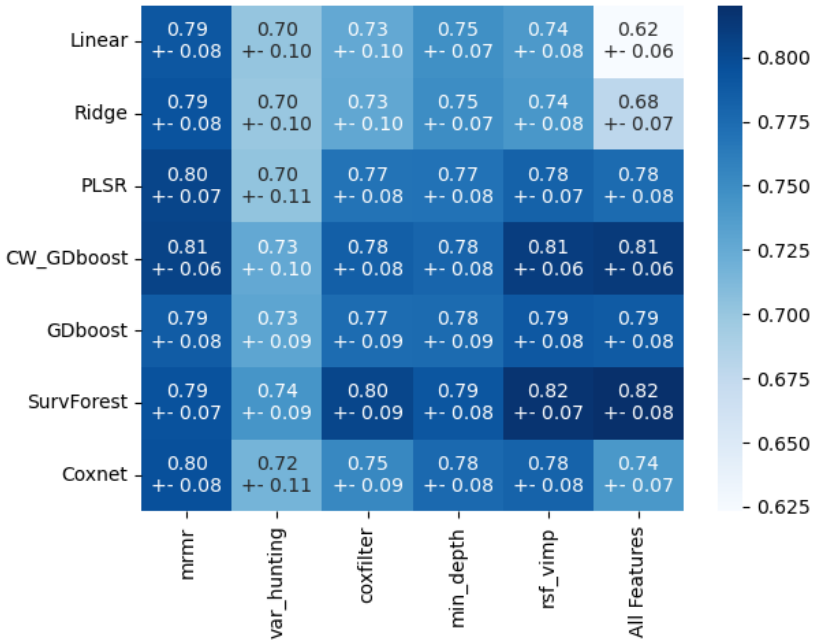


Figure A.1: Harrell's concordance index on Oxytarget dataset with standardscaling applied.

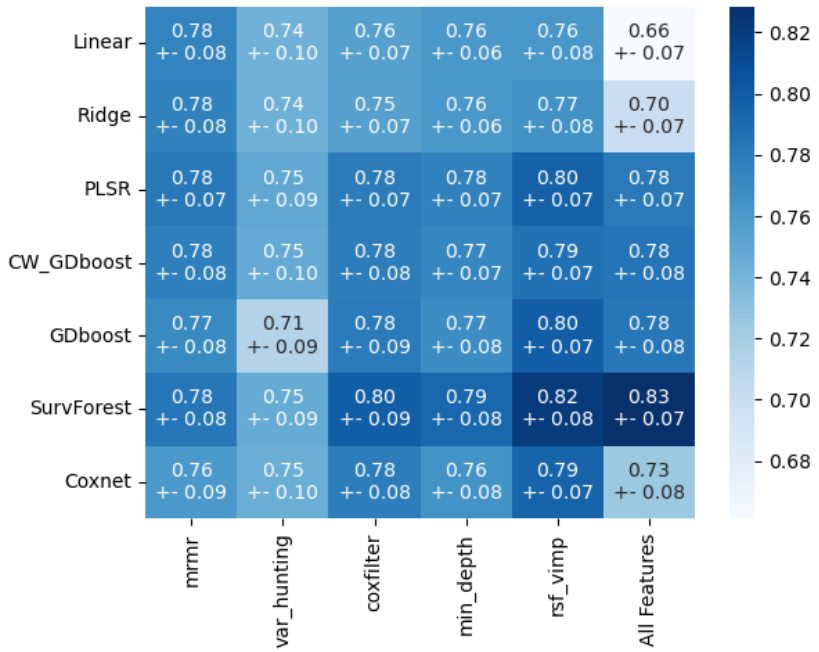


Figure A.2: Harrell's concordance index on Oxytarget dataset with powertransform applied.

A.1.2 headneck

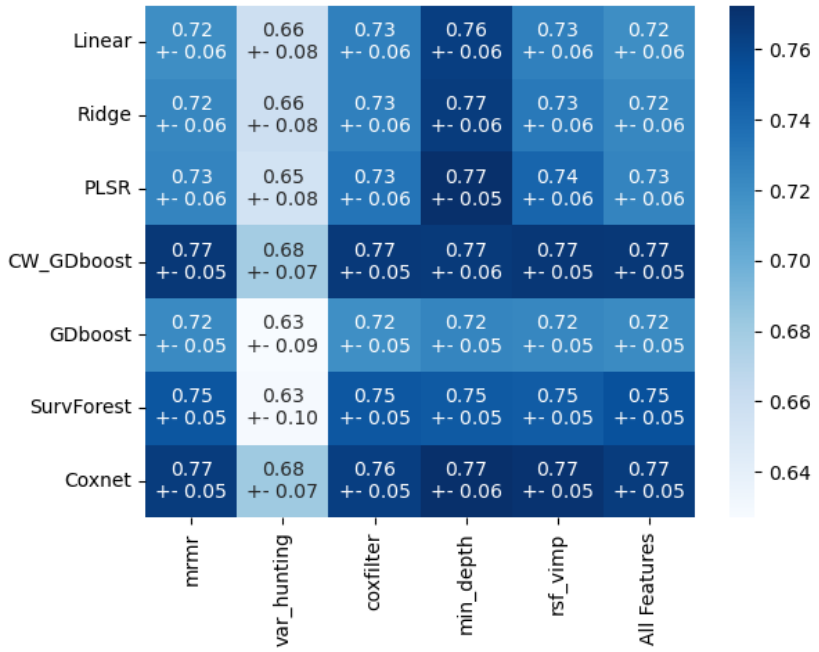


Figure A.3: Harrell's concordance index on headneck dataset with standardscaling applied.

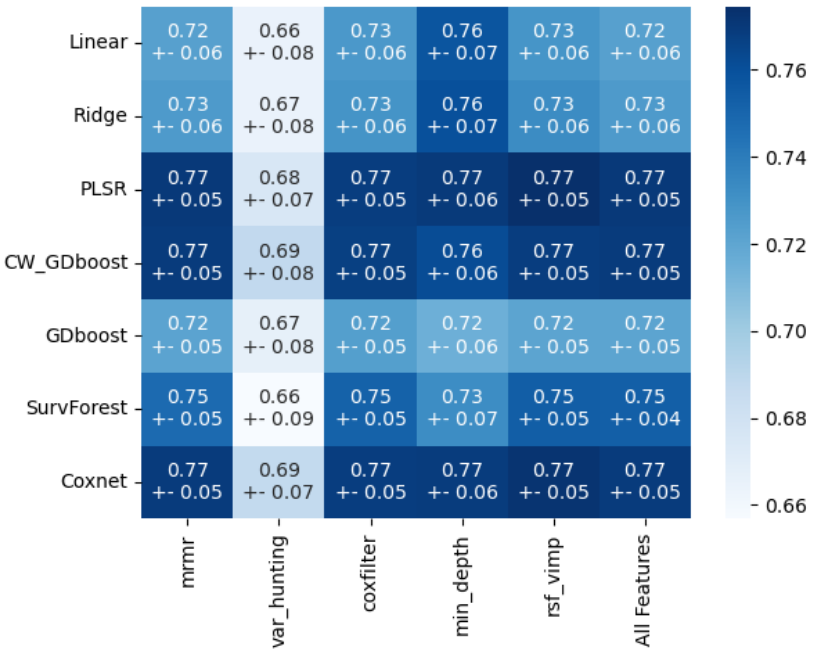


Figure A.4: Harrell's concordance index on headneck dataset with powertransform applied.

A.2 UNO's C-statistic

A.2.1 OxyTarget

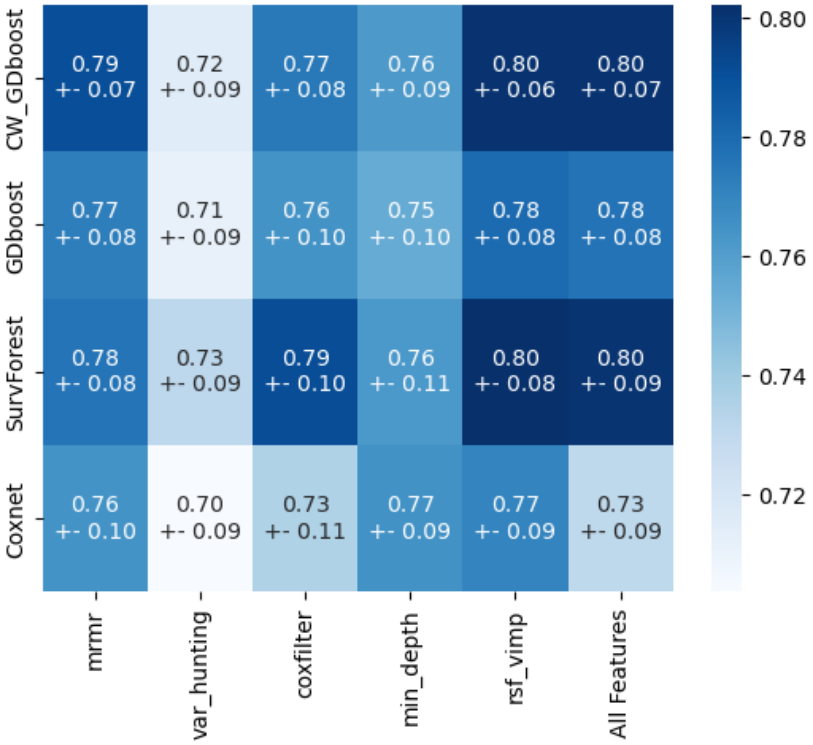


Figure A.5: Uno's C-statistic on OxyTarget dataset with standardscaling applied

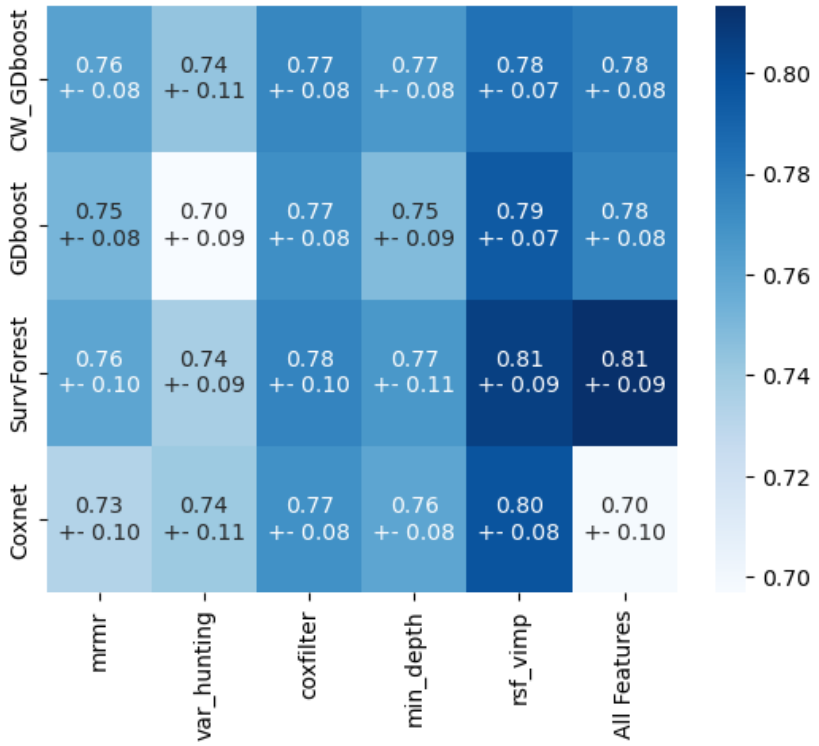


Figure A.6: Uno's C-statistic on OxyTarget dataset with powertransform applied

A.2.2 headneck

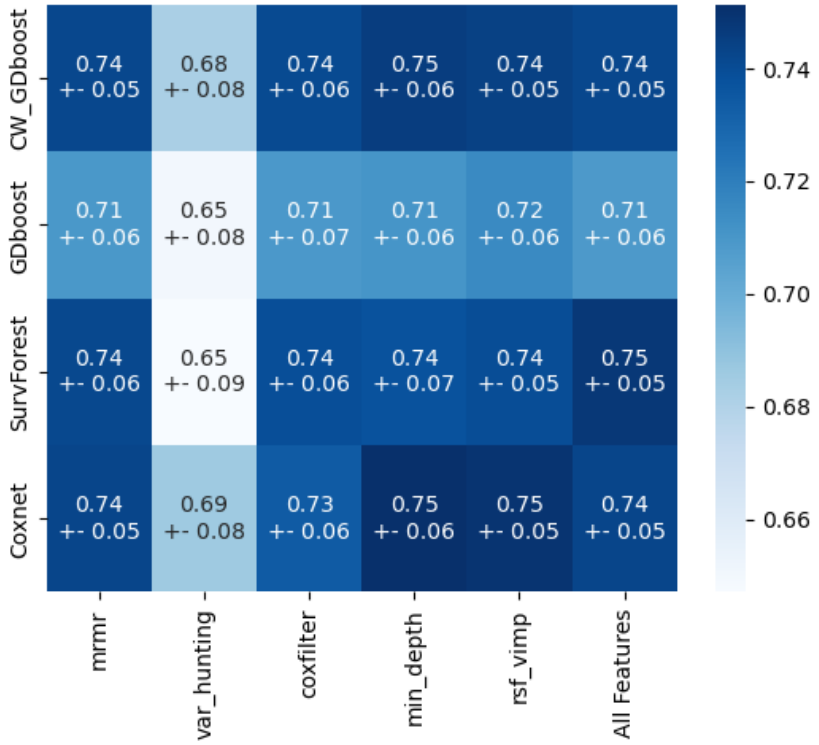


Figure A.7: Uno's C-statistic on headneck dataset with standardscaling applied

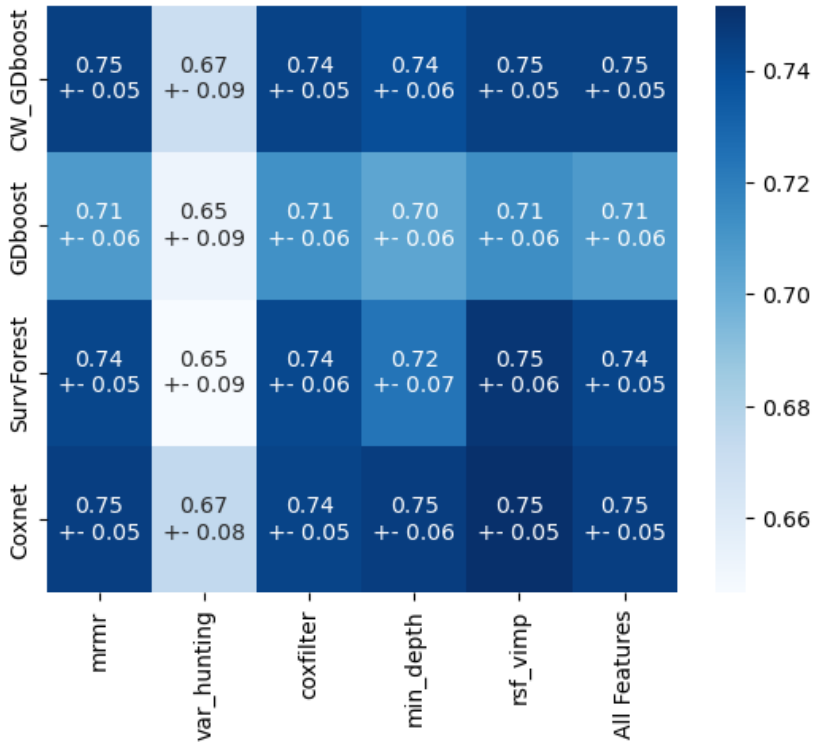


Figure A.8: Uno's C-statistic on headneck dataset with powertransform applied

A.3 UNO's C-statistic with truncation

A.3.1 OxyTarget

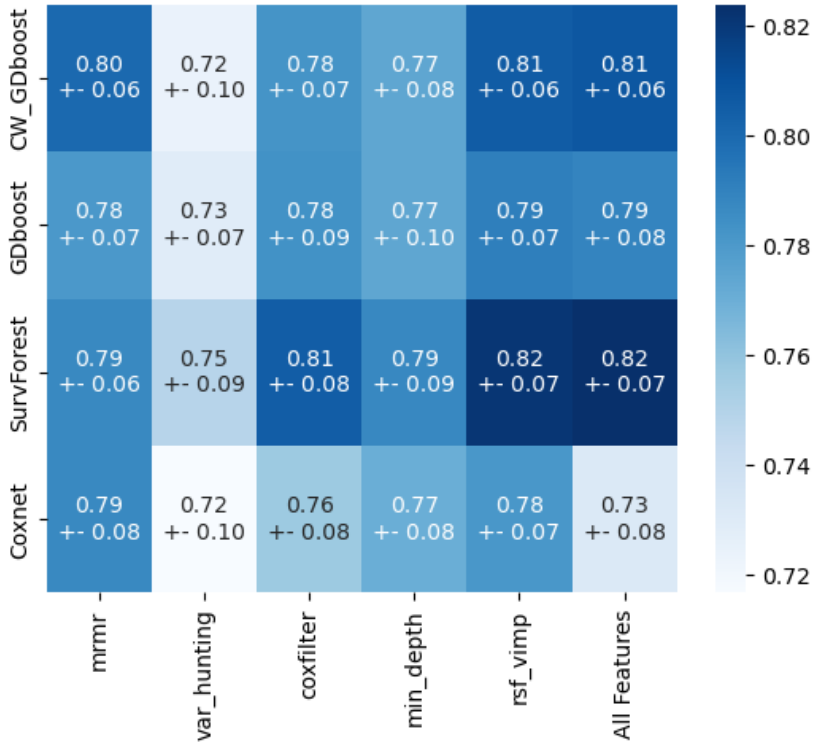


Figure A.9: Uno's C-statistic with truncation on OxyTarget dataset with standardscaling applied

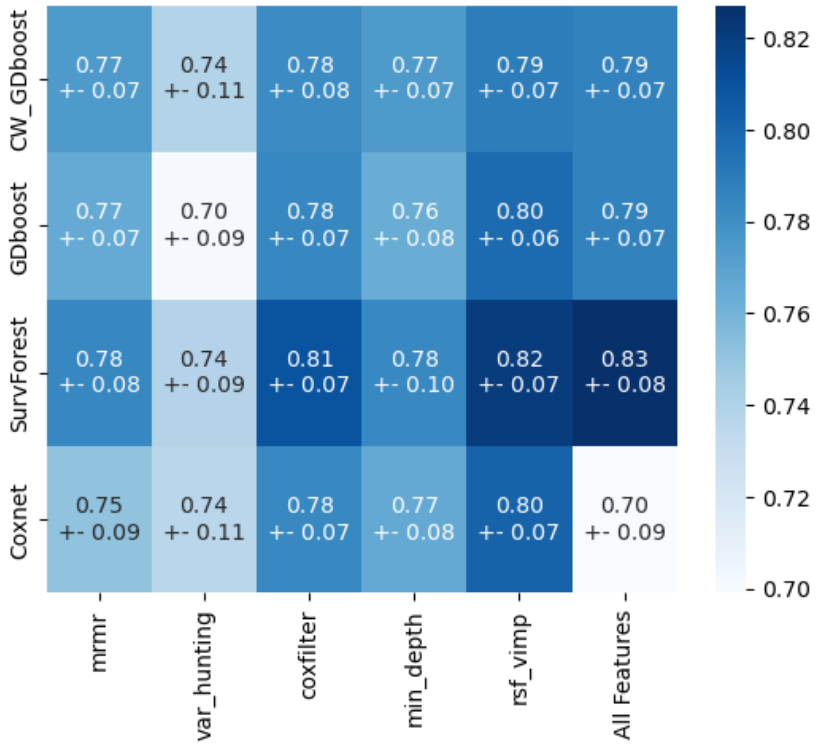


Figure A.10: Uno's C-statistic with truncation on OxyTarget dataset with powertransform applied

A.3.2 headneck

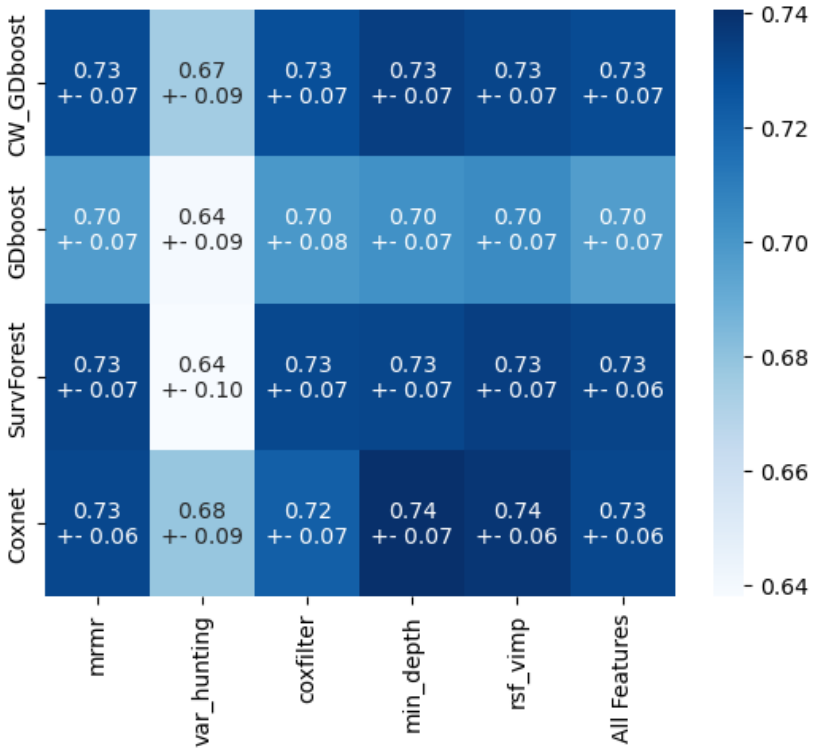


Figure A.11: Uno's C-statistic with truncation on headneck dataset with standardscaling applied

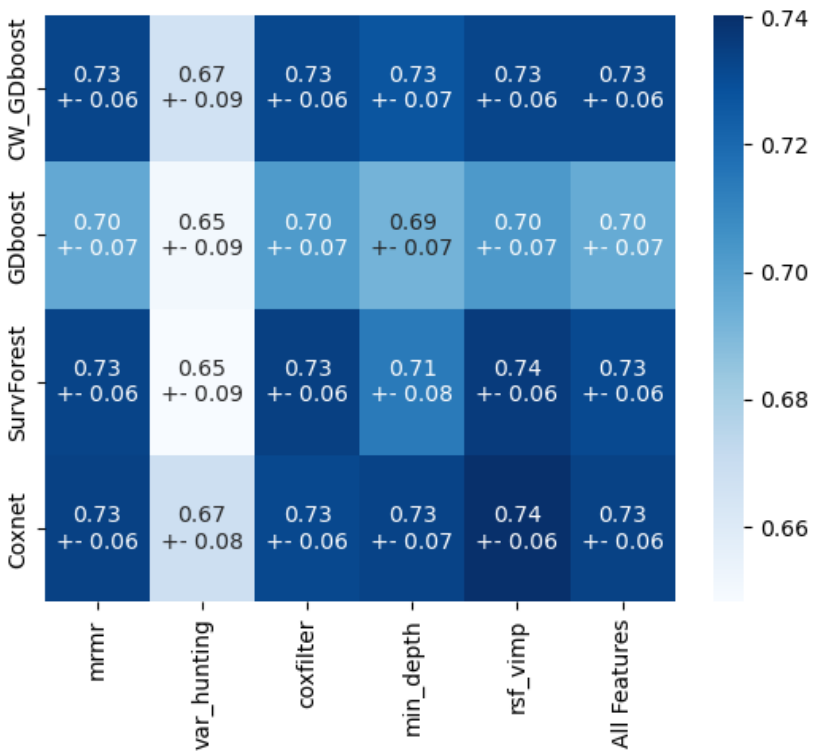


Figure A.12: Uno's C-statistic with truncation on headneck dataset with powertransform applied

A.4 IBS

A.4.1 Oxytarget

Heatmap with IBS metric for standardscaled and powertransformed data was not obtainable due to unstability in some models.

A.4.2 headneck

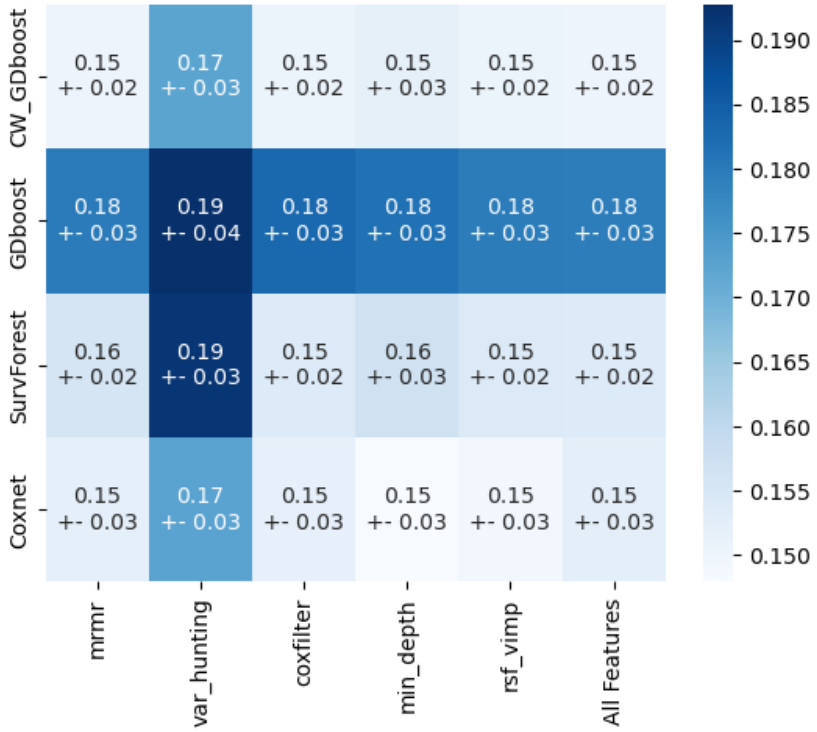


Figure A.13: Integrated Brier Score for headneck dataset with standardscaling applied

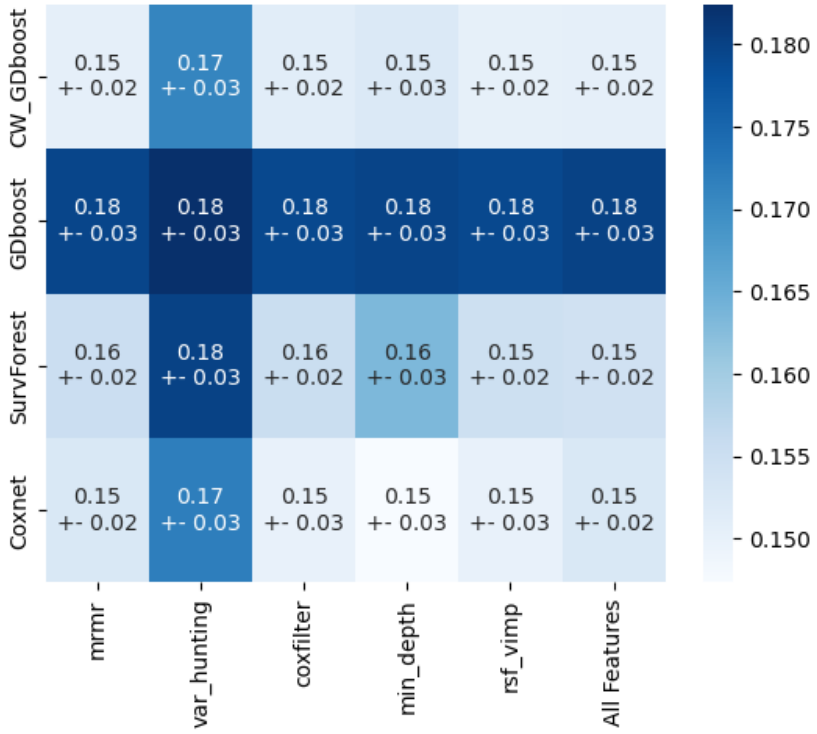


Figure A.14: Integrated Brier Score for headneck dataset with powertransform applied

A.5 RMSE

A.5.1 OxyTarget

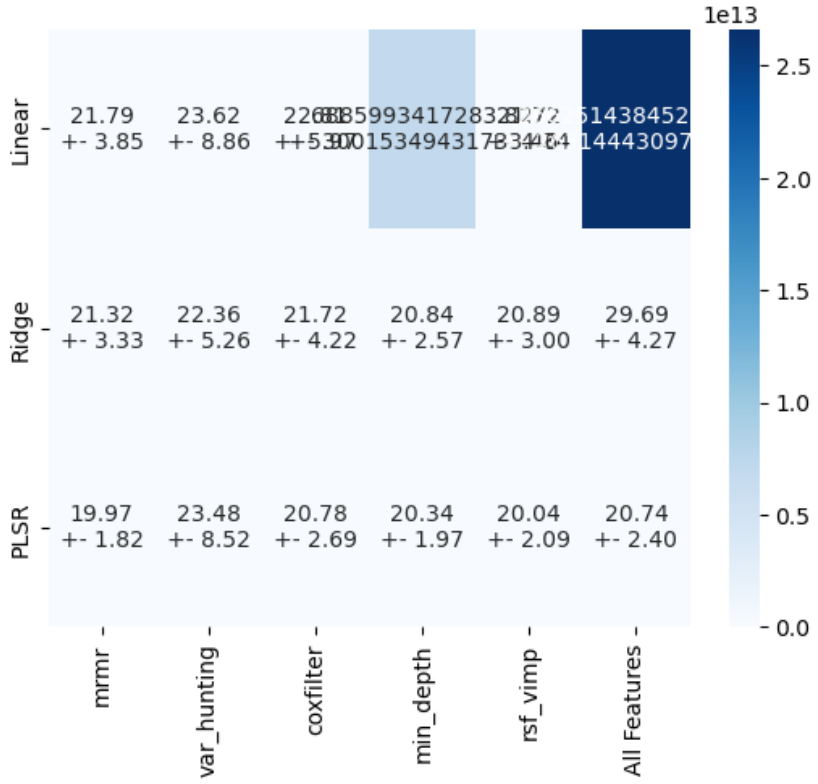


Figure A.15: RMSE for OxyTarget dataset with standardscaling applied

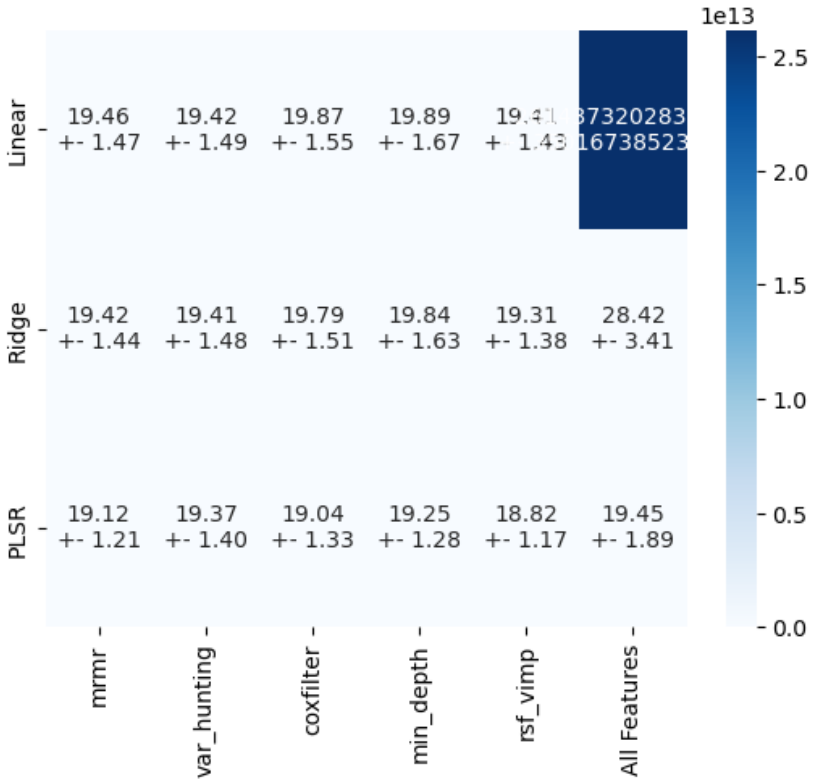


Figure A.16: RMSE for OxyTarget dataset with powertransform applied

A.5.2 headneck

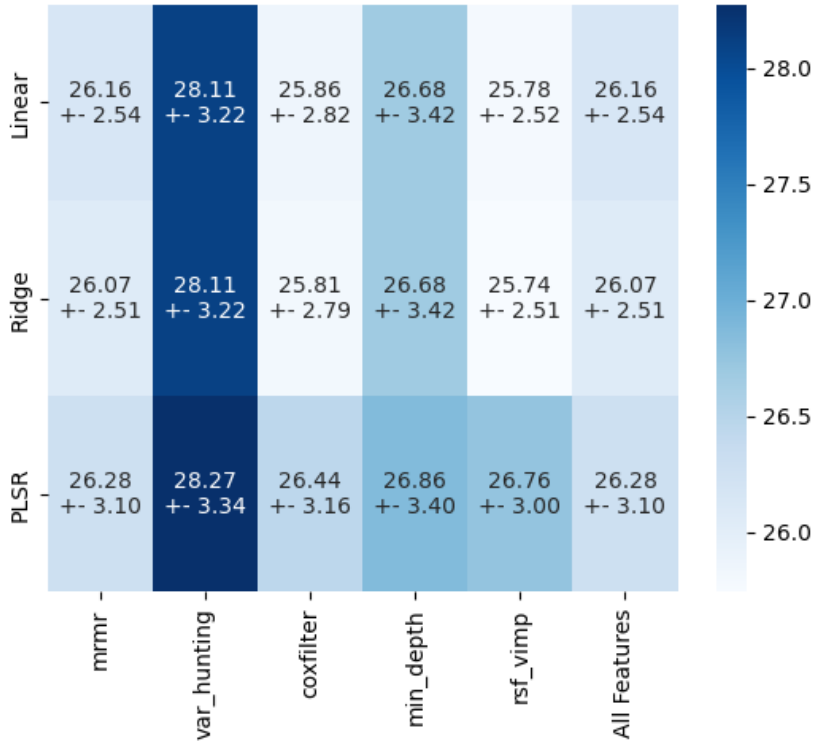


Figure A.17: RMSE for headneck dataset with standardscaling applied

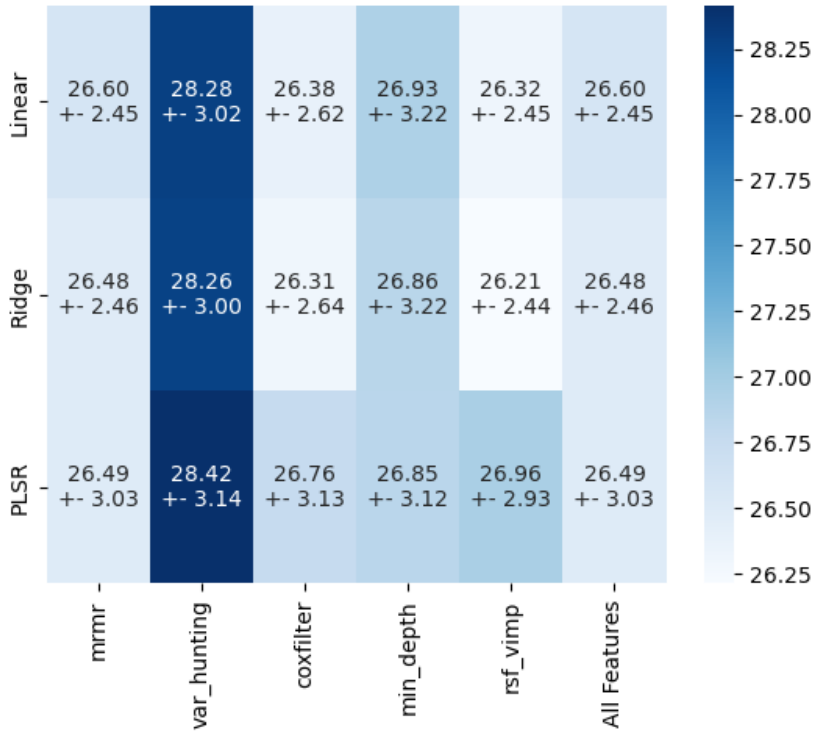


Figure A.18: RMSE for headneck dataset with powertransform applied

Appendix B

Parameter tuning with repeated k-fold

B.1 Coxnet

B.1.1 OxyTarget

	alpha	l1_ratio	Harrell-C	Uno-C	IBS
13	0.01	0.1	0.827	0.818	0.114
12	0.01	0.05	0.827	0.817	0.115
11	0.01	0.01	0.826	0.817	0.115
14	0.01	0.2	0.825	0.811	0.113
15	0.01	0.3	0.823	0.811	0.112
0	0.005	0.01	0.822	0.815	0.122
3	0.005	0.2	0.822	0.815	0.121
2	0.005	0.1	0.822	0.815	0.122
22	0.05	0.01	0.822	0.796	0.108
1	0.005	0.05	0.821	0.815	0.122
16	0.01	0.4	0.821	0.808	0.111
4	0.005	0.3	0.821	0.812	0.12
5	0.005	0.4	0.819	0.811	0.12
17	0.01	0.5	0.819	0.803	0.11
21	0.01	0.9	0.817	0.806	0.109
7	0.005	0.6	0.817	0.808	0.119
6	0.005	0.5	0.817	0.808	0.119
18	0.01	0.6	0.817	0.806	0.11
23	0.05	0.05	0.817	0.79	0.108
20	0.01	0.8	0.815	0.803	0.109
24	0.05	0.1	0.815	0.791	0.108
19	0.01	0.7	0.815	0.804	0.109
8	0.005	0.7	0.814	0.804	0.118
9	0.005	0.8	0.812	0.802	0.118
10	0.005	0.9	0.809	0.8	0.119
25	0.05	0.2	0.802	0.783	0.11
26	0.05	0.3	0.791	0.775	0.112
33	0.5	0.01	0.783	0.773	0.129
27	0.05	0.4	0.782	0.771	0.114
44	1.0	0.01	0.774	0.764	0.14
28	0.05	0.5	0.771	0.765	0.116
34	0.5	0.05	0.764	0.755	0.137
29	0.05	0.6	0.764	0.76	0.118
30	0.05	0.7	0.756	0.753	0.12
45	1.0	0.05	0.753	0.749	0.149
32	0.05	0.9	0.753	0.747	0.125
55	5.0	0.01	0.752	0.749	0.152
35	0.5	0.1	0.752	0.749	0.146
31	0.05	0.8	0.751	0.747	0.122
46	1.0	0.1	0.526	0.525	0.153
36	0.5	0.2	0.526	0.525	0.153
37	0.5	0.3	0.5	0.5	0.153
38	0.5	0.4	0.5	0.5	0.153
39	0.5	0.5	0.5	0.5	0.153
40	0.5	0.6	0.5	0.5	0.153
41	0.5	0.7	0.5	0.5	0.153
42	0.5	0.8	0.5	0.5	0.153
43	0.5	0.9	0.5	0.5	0.153
47	1.0	0.2	0.5	0.5	0.153
48	1.0	0.3	0.5	0.5	0.153
49	1.0	0.4	0.5	0.5	0.153
50	1.0	0.5	0.5	0.5	0.153
51	1.0	0.6	0.5	0.5	0.153
52	1.0	0.7	0.5	0.5	0.153
53	1.0	0.8	0.5	0.5	0.153
54	1.0	0.9	0.5	0.5	0.153
56	5.0	0.05	0.5	0.5	0.153
57	5.0	0.1	0.5	0.5	0.153
58	5.0	0.2	0.5	0.5	0.153
59	5.0	0.3	0.5	0.5	0.153
60	5.0	0.4	0.5	0.5	0.153
61	5.0	0.5	0.5	0.5	0.153
62	5.0	0.6	0.5	0.5	0.153
63	5.0	0.7	0.5	0.5	0.153
64	5.0	0.8	0.5	0.5	0.153
65	5.0	0.9	0.5	0.5	0.153
66	20.0	0.01	0.5	0.5	0.153
67	20.0	0.05	0.5	0.5	0.153
68	20.0	0.1	0.5	0.5	0.153
69	20.0	0.2	0.5	0.5	0.153
70	20.0	0.3	0.5	0.5	0.153
71	20.0	0.4	0.5	0.5	0.153
72	20.0	0.5	0.5	0.5	0.153
73	20.0	0.6	0.5	0.5	0.153
74	20.0	0.7	0.5	0.5	0.153
75	20.0	0.8	0.5	0.5	0.153
76	20.0	0.9	0.5	0.5	0.153

Figure B.1: Coxnet with repeated stratified k_fold tuned for alpha and l1_ratio on OxyTarget dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

B.1.2 headneck

	alpha	l1_ratio	Harrell-C	Uno-C	IBS
0	0.005	0.01	0.771	0.752	0.149
1	0.005	0.05	0.77	0.752	0.149
3	0.005	0.2	0.77	0.752	0.149
14	0.01	0.1	0.77	0.749	0.148
2	0.005	0.1	0.77	0.752	0.149
12	0.01	0.01	0.77	0.749	0.148
13	0.01	0.05	0.77	0.748	0.148
4	0.005	0.3	0.77	0.751	0.149
6	0.005	0.5	0.77	0.751	0.15
5	0.005	0.4	0.769	0.751	0.149
15	0.01	0.2	0.769	0.748	0.149
16	0.01	0.3	0.769	0.746	0.149
7	0.005	0.6	0.768	0.75	0.15
8	0.005	0.7	0.767	0.749	0.15
9	0.005	0.8	0.766	0.748	0.151
11	0.005	0.95	0.766	0.747	0.152
10	0.005	0.9	0.766	0.747	0.151
17	0.01	0.4	0.765	0.742	0.15
24	0.05	0.01	0.764	0.741	0.149
18	0.01	0.5	0.764	0.741	0.15
25	0.05	0.05	0.762	0.74	0.149
19	0.01	0.6	0.761	0.74	0.151
26	0.05	0.1	0.76	0.738	0.15
20	0.01	0.7	0.76	0.738	0.152
21	0.01	0.8	0.758	0.737	0.153
27	0.05	0.2	0.758	0.737	0.151
22	0.01	0.9	0.757	0.737	0.154
23	0.01	0.95	0.756	0.736	0.154
28	0.05	0.3	0.752	0.731	0.152
37	0.5	0.05	0.75	0.731	0.171
36	0.5	0.01	0.748	0.729	0.166
48	1.0	0.01	0.748	0.729	0.175
29	0.05	0.4	0.746	0.722	0.153
31	0.05	0.6	0.746	0.722	0.154
30	0.05	0.5	0.745	0.721	0.153
32	0.05	0.7	0.742	0.72	0.154
33	0.05	0.8	0.739	0.719	0.155
38	0.5	0.1	0.738	0.719	0.176
49	1.0	0.05	0.738	0.718	0.182
60	5.0	0.01	0.738	0.718	0.187
34	0.05	0.9	0.737	0.717	0.156
35	0.05	0.95	0.737	0.717	0.156
50	1.0	0.1	0.732	0.721	0.188
39	0.5	0.2	0.727	0.716	0.186
40	0.5	0.3	0.5	0.5	0.19
41	0.5	0.4	0.5	0.5	0.19
42	0.5	0.5	0.5	0.5	0.19
43	0.5	0.6	0.5	0.5	0.19
44	0.5	0.7	0.5	0.5	0.19
45	0.5	0.8	0.5	0.5	0.19
46	0.5	0.9	0.5	0.5	0.19
47	0.5	0.95	0.5	0.5	0.19
51	1.0	0.2	0.5	0.5	0.19
52	1.0	0.3	0.5	0.5	0.19
53	1.0	0.4	0.5	0.5	0.19
54	1.0	0.5	0.5	0.5	0.19
55	1.0	0.6	0.5	0.5	0.19
56	1.0	0.7	0.5	0.5	0.19
57	1.0	0.8	0.5	0.5	0.19
58	1.0	0.9	0.5	0.5	0.19
59	1.0	0.95	0.5	0.5	0.19
61	5.0	0.05	0.5	0.5	0.19
62	5.0	0.1	0.5	0.5	0.19
63	5.0	0.2	0.5	0.5	0.19
64	5.0	0.3	0.5	0.5	0.19
65	5.0	0.4	0.5	0.5	0.19
66	5.0	0.5	0.5	0.5	0.19
67	5.0	0.6	0.5	0.5	0.19
68	5.0	0.7	0.5	0.5	0.19
69	5.0	0.8	0.5	0.5	0.19
70	5.0	0.9	0.5	0.5	0.19
71	5.0	0.95	0.5	0.5	0.19
72	20.0	0.01	0.5	0.5	0.19
73	20.0	0.05	0.5	0.5	0.19
74	20.0	0.1	0.5	0.5	0.19
75	20.0	0.2	0.5	0.5	0.19
76	20.0	0.3	0.5	0.5	0.19
77	20.0	0.4	0.5	0.5	0.19
78	20.0	0.5	0.5	0.5	0.19
79	20.0	0.6	0.5	0.5	0.19
80	20.0	0.7	0.5	0.5	0.19
81	20.0	0.8	0.5	0.5	0.19
82	20.0	0.9	0.5	0.5	0.19
83	20.0	0.95	0.5	0.5	0.19

Figure B.2: Coxnet with repeated stratified k_fold tuned for alpha and l1_ratio on OxyTarget dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

B.2 Random survival forest

B.2.1 OxyTarget

	estimators	max_depth	Harrell-C	Uno-C	IBS
13	300.0	5.0	0.807	0.788	0.116
19	400.0	5.0	0.807	0.787	0.117
25	500.0	5.0	0.806	0.786	0.117
26	500.0	10.0	0.804	0.779	0.116
14	300.0	10.0	0.804	0.779	0.116
27	500.0	15.0	0.803	0.779	0.116
28	500.0	20.0	0.803	0.779	0.116
29	500.0	25.0	0.803	0.779	0.116
7	200.0	5.0	0.803	0.784	0.116
15	300.0	15.0	0.803	0.778	0.116
16	300.0	20.0	0.803	0.778	0.116
17	300.0	25.0	0.803	0.778	0.116
20	400.0	10.0	0.802	0.777	0.116
12	300.0	3.0	0.802	0.783	0.121
21	400.0	15.0	0.802	0.776	0.116
22	400.0	20.0	0.802	0.776	0.116
23	400.0	25.0	0.802	0.776	0.116
8	200.0	10.0	0.801	0.778	0.116
9	200.0	15.0	0.801	0.778	0.116
10	200.0	20.0	0.801	0.778	0.116
11	200.0	25.0	0.801	0.778	0.116
18	400.0	3.0	0.8	0.779	0.121
6	200.0	3.0	0.8	0.779	0.12
24	500.0	3.0	0.799	0.778	0.121
1	100.0	5.0	0.799	0.776	0.117
3	100.0	15.0	0.795	0.772	0.117
4	100.0	20.0	0.795	0.772	0.117
5	100.0	25.0	0.795	0.772	0.117
2	100.0	10.0	0.795	0.772	0.117
0	100.0	3.0	0.793	0.775	0.121

Figure B.3: Random survival forest on repeated stratified k_fold tuned for n_trees and max_depth on OxyTarget dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

	weight_fraction	max_depth	Harrell-C	Uno-C	IBS
2	0.005	7.0	0.8	0.776	0.117
9	0.01	7.0	0.8	0.776	0.117
1	0.005	5.0	0.799	0.776	0.117
8	0.01	5.0	0.799	0.776	0.117
16	0.1	7.0	0.798	0.8	0.124
17	0.1	10.0	0.798	0.8	0.124
18	0.1	15.0	0.798	0.8	0.124
19	0.1	20.0	0.798	0.8	0.124
20	0.1	25.0	0.798	0.8	0.124
15	0.1	5.0	0.798	0.8	0.124
28	0.3	3.0	0.798	0.8	0.142
29	0.3	5.0	0.798	0.8	0.142
30	0.3	7.0	0.798	0.8	0.142
31	0.3	10.0	0.798	0.8	0.142
32	0.3	15.0	0.798	0.8	0.142
33	0.3	20.0	0.798	0.8	0.142
34	0.3	25.0	0.798	0.8	0.142
4	0.005	15.0	0.795	0.772	0.117
5	0.005	20.0	0.795	0.772	0.117
6	0.005	25.0	0.795	0.772	0.117
11	0.01	15.0	0.795	0.772	0.117
12	0.01	20.0	0.795	0.772	0.117
13	0.01	25.0	0.795	0.772	0.117
3	0.005	10.0	0.795	0.772	0.117
10	0.01	10.0	0.795	0.772	0.117
14	0.1	3.0	0.793	0.788	0.126
0	0.005	3.0	0.793	0.775	0.121
7	0.01	3.0	0.793	0.775	0.121
21	0.2	3.0	0.792	0.79	0.136
22	0.2	5.0	0.792	0.79	0.136
23	0.2	7.0	0.792	0.79	0.136
24	0.2	10.0	0.792	0.79	0.136
25	0.2	15.0	0.792	0.79	0.136
26	0.2	20.0	0.792	0.79	0.136
27	0.2	25.0	0.792	0.79	0.136
35	0.4	3.0	0.786	0.788	0.147
36	0.4	5.0	0.786	0.788	0.147
37	0.4	7.0	0.786	0.788	0.147
38	0.4	10.0	0.786	0.788	0.147
39	0.4	15.0	0.786	0.788	0.147
40	0.4	20.0	0.786	0.788	0.147
41	0.4	25.0	0.786	0.788	0.147
42	0.5	3.0	0.616	0.619	0.157
43	0.5	5.0	0.616	0.619	0.157
44	0.5	7.0	0.616	0.619	0.157
45	0.5	10.0	0.616	0.619	0.157
46	0.5	15.0	0.616	0.619	0.157
47	0.5	20.0	0.616	0.619	0.157
48	0.5	25.0	0.616	0.619	0.157

Figure B.4: Random survival forest on repeated stratified k_fold tuned for min_weight_fraction_leaf and max_depth on OxyTarget dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

B.2.2 headneck

	estimators	max_depth	Harrell-C	Uno-C	IBS
18	400.0	3.0	0.766	0.757	0.149
24	500.0	3.0	0.765	0.756	0.15
12	300.0	3.0	0.764	0.753	0.149
6	200.0	3.0	0.763	0.755	0.15
0	100.0	3.0	0.762	0.756	0.15
19	400.0	5.0	0.76	0.755	0.151
1	100.0	5.0	0.759	0.752	0.15
7	200.0	5.0	0.759	0.752	0.151
25	500.0	5.0	0.759	0.755	0.152
20	400.0	10.0	0.758	0.754	0.153
26	500.0	10.0	0.758	0.754	0.152
8	200.0	10.0	0.758	0.754	0.153
13	300.0	5.0	0.758	0.751	0.152
21	400.0	15.0	0.757	0.752	0.152
16	300.0	20.0	0.757	0.752	0.152
15	300.0	15.0	0.756	0.755	0.153
28	500.0	20.0	0.756	0.753	0.152
22	400.0	20.0	0.756	0.751	0.152
23	400.0	25.0	0.756	0.751	0.152
27	500.0	15.0	0.756	0.751	0.152
5	100.0	25.0	0.756	0.75	0.153
9	200.0	15.0	0.756	0.75	0.153
10	200.0	20.0	0.756	0.752	0.152
11	200.0	25.0	0.756	0.753	0.153
29	500.0	25.0	0.755	0.753	0.153
3	100.0	15.0	0.755	0.748	0.153
14	300.0	10.0	0.755	0.749	0.154
2	100.0	10.0	0.754	0.747	0.153
17	300.0	25.0	0.754	0.75	0.153
4	100.0	20.0	0.752	0.747	0.154

Figure B.5: Random survival forest on repeated stratified k_fold tuned for n_trees and max_depth on headneck dataset. Sorted by decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

	weight_fraction	max_depth	Harrell-C	Uno-C	IBS
31	0.3	10.0	0.787	0.769	0.158
33	0.3	20.0	0.783	0.768	0.159
29	0.3	5.0	0.78	0.763	0.158
34	0.3	25.0	0.78	0.763	0.158
32	0.3	15.0	0.778	0.763	0.159
28	0.3	3.0	0.777	0.763	0.158
30	0.3	7.0	0.777	0.759	0.158
22	0.2	5.0	0.773	0.755	0.152
19	0.1	20.0	0.772	0.767	0.148
23	0.2	7.0	0.771	0.755	0.152
27	0.2	25.0	0.77	0.757	0.153
16	0.1	7.0	0.77	0.761	0.148
25	0.2	15.0	0.77	0.754	0.152
24	0.2	10.0	0.769	0.753	0.153
15	0.1	5.0	0.769	0.757	0.149
21	0.2	3.0	0.768	0.755	0.153
17	0.1	10.0	0.768	0.758	0.149
18	0.1	15.0	0.767	0.759	0.149
14	0.1	3.0	0.767	0.756	0.15
20	0.1	25.0	0.767	0.76	0.149
26	0.2	20.0	0.766	0.751	0.153
7	0.01	3.0	0.765	0.752	0.15
0	0.005	3.0	0.763	0.759	0.15
2	0.005	7.0	0.76	0.756	0.152
13	0.01	25.0	0.759	0.757	0.152
5	0.005	20.0	0.759	0.757	0.152
10	0.01	10.0	0.758	0.751	0.153
12	0.01	20.0	0.757	0.752	0.152
8	0.01	5.0	0.756	0.752	0.152
6	0.005	25.0	0.755	0.753	0.153
11	0.01	15.0	0.755	0.748	0.154
3	0.005	10.0	0.755	0.751	0.153
4	0.005	15.0	0.754	0.752	0.152
1	0.005	5.0	0.753	0.751	0.153
9	0.01	7.0	0.751	0.748	0.154
41	0.4	25.0	0.747	0.72	0.173
37	0.4	7.0	0.746	0.724	0.173
39	0.4	15.0	0.745	0.724	0.173
35	0.4	3.0	0.742	0.719	0.173
36	0.4	5.0	0.739	0.728	0.173
40	0.4	20.0	0.739	0.714	0.173
38	0.4	10.0	0.735	0.708	0.173
46	0.5	15.0	0.548	0.543	0.187
42	0.5	3.0	0.547	0.546	0.187
43	0.5	5.0	0.543	0.544	0.187
45	0.5	10.0	0.542	0.543	0.187
47	0.5	20.0	0.541	0.541	0.188
48	0.5	25.0	0.539	0.536	0.188
44	0.5	7.0	0.535	0.538	0.188

Figure B.6: Random survival forest on repeated stratified k_fold tuned for min_weight_fraction_leaf and max_depth on headneck dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

B.3 Gradient boosting with coxph as loss function

B.3.1 headneck dataset

B.3.2 OxyTarget

	estimators	learning_rate	Harrell-C	Uno-C	IBS
29	300.0	1.0	0.769	0.757	0.175
39	400.0	1.0	0.769	0.756	0.176
19	200.0	1.0	0.769	0.755	0.174
49	500.0	1.0	0.766	0.754	invalid
9	100.0	1.0	0.766	0.749	0.168
6	100.0	0.7	0.762	0.732	0.161
15	200.0	0.6	0.761	0.733	0.169
5	100.0	0.6	0.759	0.728	0.162
26	300.0	0.7	0.757	0.729	0.172
25	300.0	0.6	0.757	0.734	0.173
16	200.0	0.7	0.757	0.727	0.169
24	300.0	0.5	0.757	0.736	0.172
34	400.0	0.5	0.757	0.735	0.175
23	300.0	0.4	0.757	0.732	0.169
13	200.0	0.4	0.756	0.73	0.165
27	300.0	0.8	0.756	0.744	0.173
14	200.0	0.5	0.756	0.733	0.168
33	400.0	0.4	0.756	0.729	0.171
40	500.0	0.1	0.756	0.729	0.16
36	400.0	0.7	0.756	0.727	0.174
37	400.0	0.8	0.756	0.742	0.175
43	500.0	0.4	0.756	0.732	0.173
17	200.0	0.8	0.755	0.742	0.17
35	400.0	0.6	0.755	0.733	0.176
44	500.0	0.5	0.755	0.732	0.176
28	300.0	0.9	0.755	0.732	0.175
30	400.0	0.1	0.755	0.726	0.157
4	100.0	0.5	0.755	0.73	0.16
46	500.0	0.7	0.754	0.728	0.176
48	500.0	0.9	0.754	0.736	0.174
12	200.0	0.3	0.754	0.729	0.162
47	500.0	0.8	0.754	0.742	0.176
45	500.0	0.6	0.754	0.735	0.177
18	200.0	0.9	0.754	0.731	0.173
38	400.0	0.9	0.754	0.736	0.177
32	400.0	0.3	0.754	0.732	0.169
21	300.0	0.2	0.754	0.728	0.161
8	100.0	0.9	0.754	0.73	0.167
3	100.0	0.4	0.754	0.728	0.158
20	300.0	0.1	0.753	0.723	0.153
7	100.0	0.8	0.753	0.74	0.164
31	400.0	0.2	0.753	0.728	0.164
10	200.0	0.1	0.753	0.725	0.147
22	300.0	0.3	0.753	0.73	0.167
42	500.0	0.3	0.753	0.731	0.171
41	500.0	0.2	0.753	0.731	0.167
0	100.0	0.1	0.752	0.72	0.137
2	100.0	0.3	0.752	0.727	0.153
11	200.0	0.2	0.752	0.725	0.156
1	100.0	0.2	0.751	0.722	0.147

Figure B.7: Gradient boosting with coxph as loss function on repeated stratified k_fold tuned for n_estimators and learning_rate on OxyTarget dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

	estimators	learning_rate	weight_fraction	max_depth	Harrell-C	Uno-C	IBS
95	100.0	0.4	0.4	2.0	0.775	0.784	0.124
96	100.0	0.4	0.4	3.0	0.775	0.784	0.124
97	100.0	0.4	0.4	4.0	0.775	0.784	0.124
98	100.0	0.4	0.4	5.0	0.775	0.784	0.124
99	100.0	0.4	0.4	10.0	0.775	0.784	0.124
10	100.0	0.1	0.2	2.0	0.775	0.784	0.124
145	200.0	0.2	0.4	2.0	0.775	0.784	0.124
146	200.0	0.2	0.4	3.0	0.775	0.784	0.124
147	200.0	0.2	0.4	4.0	0.775	0.784	0.124
148	200.0	0.2	0.4	5.0	0.775	0.784	0.124
149	200.0	0.2	0.4	10.0	0.775	0.784	0.124
170	200.0	0.3	0.4	2.0	0.775	0.784	0.128
171	200.0	0.3	0.4	3.0	0.775	0.784	0.128
172	200.0	0.3	0.4	4.0	0.775	0.784	0.128
173	200.0	0.3	0.4	5.0	0.775	0.784	0.128
174	200.0	0.3	0.4	10.0	0.775	0.784	0.128
245	300.0	0.2	0.4	2.0	0.775	0.784	0.127
246	300.0	0.2	0.4	3.0	0.775	0.784	0.127
247	300.0	0.2	0.4	4.0	0.775	0.784	0.127
248	300.0	0.2	0.4	5.0	0.775	0.784	0.127

Figure B.8: Gradient boosting with coxph as loss function on repeated stratified k_fold tuned for n_estimators, learning_rate, min_weight_fraction_leaf and max_depth on OxyTarget dataset (top 20 hyperparameter combinations). Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

	estimators	learning_rate	Harrell-C	Uno-C	IBS
0	100.0	0.1	0.742	0.733	0.173
1	100.0	0.2	0.734	0.728	0.189
10	200.0	0.1	0.73	0.724	0.19
20	300.0	0.1	0.726	0.721	0.2
3	100.0	0.4	0.724	0.722	0.209
4	100.0	0.5	0.722	0.722	0.216
30	400.0	0.1	0.721	0.718	0.211
9	100.0	1.0	0.72	0.719	0.23
40	500.0	0.1	0.719	0.715	0.217
11	200.0	0.2	0.719	0.716	0.209
2	100.0	0.3	0.718	0.713	0.201
6	100.0	0.7	0.718	0.716	0.223
21	300.0	0.2	0.718	0.717	0.222
5	100.0	0.6	0.718	0.715	0.219
7	100.0	0.8	0.717	0.713	0.23
17	200.0	0.8	0.716	0.718	0.243
12	200.0	0.3	0.716	0.712	0.221
19	200.0	1.0	0.715	0.717	0.242
14	200.0	0.5	0.715	0.713	0.235
13	200.0	0.4	0.715	0.715	0.228
15	200.0	0.6	0.715	0.714	0.236
31	400.0	0.2	0.715	0.714	0.231
16	200.0	0.7	0.714	0.71	0.24
29	300.0	1.0	0.713	0.717	0.252
8	100.0	0.9	0.712	0.715	0.232
18	200.0	0.9	0.712	0.712	0.244
27	300.0	0.8	0.712	0.712	0.249
25	300.0	0.6	0.712	0.711	0.246
23	300.0	0.4	0.711	0.71	0.238
39	400.0	1.0	0.711	0.715	0.258
32	400.0	0.3	0.711	0.706	0.239
22	300.0	0.3	0.71	0.706	0.233
47	500.0	0.8	0.71	0.711	0.257
24	300.0	0.5	0.71	0.708	0.242
41	500.0	0.2	0.71	0.708	0.238
36	400.0	0.7	0.71	0.708	0.253
49	500.0	1.0	0.71	0.713	0.262
45	500.0	0.6	0.71	0.71	0.254
37	400.0	0.8	0.71	0.713	0.255
35	400.0	0.6	0.71	0.707	0.251
33	400.0	0.4	0.71	0.709	0.244
46	500.0	0.7	0.709	0.708	0.257
26	300.0	0.7	0.709	0.707	0.248
44	500.0	0.5	0.709	0.708	0.253
43	500.0	0.4	0.709	0.708	0.248
42	500.0	0.3	0.708	0.706	0.246
34	400.0	0.5	0.708	0.708	0.248
28	300.0	0.9	0.708	0.709	0.254
38	400.0	0.9	0.708	0.708	0.259
48	500.0	0.9	0.707	0.707	0.26

Figure B.9: Gradient boosting with coxph as loss function on repeated stratified k_fold tuned for n_estimators and learning_rate on headneck dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

	estimators	learning_rate	weight_fraction	max_depth	Harrell-C	Uno-C	IBS
15	100.0	0.1	0.3	2.0	0.768	0.747	0.148
16	100.0	0.1	0.3	3.0	0.768	0.747	0.148
17	100.0	0.1	0.3	4.0	0.768	0.747	0.148
18	100.0	0.1	0.3	5.0	0.768	0.747	0.148
19	100.0	0.1	0.3	10.0	0.768	0.747	0.148
115	200.0	0.1	0.3	2.0	0.763	0.745	0.151
116	200.0	0.1	0.3	3.0	0.763	0.745	0.151
117	200.0	0.1	0.3	4.0	0.763	0.745	0.151
118	200.0	0.1	0.3	5.0	0.763	0.745	0.151
119	200.0	0.1	0.3	10.0	0.763	0.745	0.151
40	100.0	0.2	0.3	2.0	0.762	0.743	0.151
41	100.0	0.2	0.3	3.0	0.762	0.743	0.151
42	100.0	0.2	0.3	4.0	0.762	0.743	0.151
43	100.0	0.2	0.3	5.0	0.762	0.743	0.151
44	100.0	0.2	0.3	10.0	0.762	0.743	0.151
215	300.0	0.1	0.3	2.0	0.76	0.743	0.153
216	300.0	0.1	0.3	3.0	0.76	0.743	0.153
217	300.0	0.1	0.3	4.0	0.76	0.743	0.153
218	300.0	0.1	0.3	5.0	0.76	0.743	0.153
219	300.0	0.1	0.3	10.0	0.76	0.743	0.153

Figure B.10: Gradient boosting with coxph as loss function on repeated stratified k_fold tuned for n_estimators, learning_rate, min_weight_fraction_leaf and max_depth on headneck dataset. (top 20 hyperparameter combinations). Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

B.4 Componentwise Gradient boosting with coxph as loss function

B.4.1 OxyTarget

	estimators	learning_rate	Harrell-C	Uno-C	IBS
15	200.0	0.6	0.813	0.809	0.121
42	500.0	0.3	0.812	0.813	0.122
24	300.0	0.5	0.812	0.812	0.123
17	200.0	0.8	0.812	0.811	0.123
5	100.0	0.6	0.812	0.808	0.119
23	300.0	0.4	0.811	0.809	0.121
12	200.0	0.3	0.811	0.806	0.119
16	200.0	0.7	0.811	0.81	0.122
32	400.0	0.3	0.811	0.808	0.121
6	100.0	0.7	0.811	0.807	0.119
33	400.0	0.4	0.811	0.812	0.123
43	500.0	0.4	0.811	0.81	0.124
25	300.0	0.6	0.81	0.811	0.124
4	100.0	0.5	0.81	0.804	0.119
41	500.0	0.2	0.81	0.808	0.121
14	200.0	0.5	0.81	0.808	0.12
21	300.0	0.2	0.81	0.805	0.119
34	400.0	0.5	0.81	0.809	0.124
13	200.0	0.4	0.81	0.807	0.12
22	300.0	0.3	0.809	0.809	0.12
9	100.0	1.0	0.809	0.806	0.121
7	100.0	0.8	0.809	0.806	0.12
40	500.0	0.1	0.809	0.803	0.119
18	200.0	0.9	0.809	0.809	0.124
35	400.0	0.6	0.808	0.806	0.125
30	400.0	0.1	0.808	0.802	0.119
31	400.0	0.2	0.808	0.806	0.12
26	300.0	0.7	0.808	0.808	0.125
8	100.0	0.9	0.808	0.805	0.12
11	200.0	0.2	0.808	0.801	0.119
19	200.0	1.0	0.808	0.807	0.125
3	100.0	0.4	0.807	0.8	0.119
44	500.0	0.5	0.807	0.806	0.126
45	500.0	0.6	0.806	0.804	0.127
27	300.0	0.8	0.806	0.806	0.126
28	300.0	0.9	0.804	0.803	0.126
20	300.0	0.1	0.804	0.794	0.119
36	400.0	0.7	0.803	0.801	0.127
2	100.0	0.3	0.802	0.793	0.119
29	300.0	1.0	0.802	0.801	0.127
37	400.0	0.8	0.801	0.799	0.128
46	500.0	0.7	0.8	0.797	0.128
38	400.0	0.9	0.799	0.797	0.128
39	400.0	1.0	0.798	0.794	0.129
48	500.0	0.9	0.797	0.795	0.13
1	100.0	0.2	0.797	0.787	0.119
10	200.0	0.1	0.797	0.786	0.119
49	500.0	1.0	0.797	0.792	0.13
47	500.0	0.8	0.797	0.794	0.129
0	100.0	0.1	0.779	0.762	0.122

Figure B.11: Parameter tuning for number of estimators and learning rate for componentwise gradient boosting using repeated stratified k-fold for OxyTarget dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

	estimators	learning_rate	subsample	Harrell-C	Uno-C	IBS
89	300.0	0.4	0.1	0.815	0.814	0.121
91	300.0	0.4	0.4	0.814	0.809	0.121
93	300.0	0.4	0.8	0.814	0.811	0.121
92	300.0	0.4	0.6	0.813	0.811	0.121
90	300.0	0.4	0.2	0.813	0.81	0.12
81	300.0	0.3	0.1	0.813	0.805	0.119
83	300.0	0.3	0.4	0.813	0.811	0.12
50	200.0	0.3	0.2	0.812	0.803	0.118
94	300.0	0.4	0.9	0.812	0.809	0.121
53	200.0	0.3	0.8	0.812	0.808	0.119
54	200.0	0.3	0.9	0.812	0.805	0.119
95	300.0	0.4	1.0	0.811	0.809	0.121
55	200.0	0.3	1.0	0.811	0.806	0.119
86	300.0	0.3	0.9	0.811	0.81	0.12
85	300.0	0.3	0.8	0.811	0.808	0.12
76	300.0	0.2	0.6	0.811	0.806	0.119
73	300.0	0.2	0.1	0.811	0.799	0.119
84	300.0	0.3	0.6	0.811	0.81	0.12
52	200.0	0.3	0.6	0.811	0.806	0.119
51	200.0	0.3	0.4	0.81	0.806	0.119

Figure B.12: Componentwise gradient boosting with coxph as loss function on repeated stratified k_fold tuned for n_estimators, learning_rate and subsample on OxyTarget dataset. (top 20 hyperparameter combinations). Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

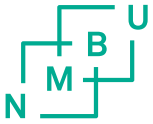
B.4.2 headneck dataset

	estimators	learning_rate	Harrell-C	Uno-C	IBS
8	100.0	0.9	0.769	0.75	0.149
14	200.0	0.5	0.768	0.749	0.149
41	500.0	0.2	0.768	0.75	0.149
7	100.0	0.8	0.768	0.75	0.149
12	200.0	0.3	0.768	0.749	0.149
21	300.0	0.2	0.768	0.749	0.149
6	100.0	0.7	0.768	0.749	0.15
32	400.0	0.3	0.768	0.75	0.149
22	300.0	0.3	0.767	0.749	0.149
13	200.0	0.4	0.767	0.749	0.15
40	500.0	0.1	0.767	0.75	0.15
23	300.0	0.4	0.767	0.75	0.149
9	100.0	1.0	0.767	0.749	0.149
4	100.0	0.5	0.767	0.75	0.15
11	200.0	0.2	0.767	0.75	0.15
30	400.0	0.1	0.767	0.75	0.15
31	400.0	0.2	0.767	0.748	0.15
5	100.0	0.6	0.767	0.748	0.149
15	200.0	0.6	0.767	0.748	0.149
42	500.0	0.3	0.766	0.749	0.149
33	400.0	0.4	0.766	0.749	0.15
18	200.0	0.9	0.766	0.748	0.15
3	100.0	0.4	0.766	0.75	0.15
16	200.0	0.7	0.766	0.748	0.149
24	300.0	0.5	0.766	0.748	0.149
17	200.0	0.8	0.765	0.747	0.149
25	300.0	0.6	0.765	0.747	0.15
43	500.0	0.4	0.765	0.747	0.15
20	300.0	0.1	0.765	0.748	0.15
2	100.0	0.3	0.764	0.749	0.15
19	200.0	1.0	0.764	0.748	0.15
34	400.0	0.5	0.764	0.747	0.15
26	300.0	0.7	0.764	0.747	0.15
44	500.0	0.5	0.764	0.747	0.15
27	300.0	0.8	0.764	0.749	0.15
28	300.0	0.9	0.764	0.746	0.151
35	400.0	0.6	0.764	0.746	0.15
29	300.0	1.0	0.764	0.746	0.151
36	400.0	0.7	0.763	0.745	0.151
45	500.0	0.6	0.763	0.746	0.151
37	400.0	0.8	0.762	0.745	0.151
46	500.0	0.7	0.762	0.745	0.152
1	100.0	0.2	0.762	0.745	0.151
47	500.0	0.8	0.762	0.745	0.152
10	200.0	0.1	0.762	0.745	0.151
39	400.0	1.0	0.762	0.746	0.152
38	400.0	0.9	0.762	0.745	0.152
48	500.0	0.9	0.761	0.745	0.153
49	500.0	1.0	0.76	0.743	0.153
0	100.0	0.1	0.759	0.744	0.155

Figure B.13: Parameter tuning for number of estimators and learning rate for componentwise gradient boosting using repeated stratified k-fold for headneck dataset. Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS

	estimators	learning_rate	subsample	Harrell-C	Uno-C	IBS
89	300.0	0.4	0.1	0.771	0.752	0.148
81	300.0	0.3	0.1	0.768	0.751	0.148
82	300.0	0.3	0.2	0.768	0.752	0.149
85	300.0	0.3	0.8	0.768	0.75	0.149
60	200.0	0.4	0.6	0.768	0.751	0.149
27	100.0	0.4	0.4	0.768	0.75	0.149
55	200.0	0.3	1.0	0.768	0.749	0.149
90	300.0	0.4	0.2	0.768	0.749	0.148
52	200.0	0.3	0.6	0.768	0.752	0.149
61	200.0	0.4	0.8	0.768	0.749	0.149
79	300.0	0.2	1.0	0.768	0.749	0.149
84	300.0	0.3	0.6	0.768	0.75	0.149
54	200.0	0.3	0.9	0.768	0.75	0.149
87	300.0	0.3	1.0	0.767	0.749	0.149
63	200.0	0.4	1.0	0.767	0.749	0.15
86	300.0	0.3	0.9	0.767	0.748	0.149
76	300.0	0.2	0.6	0.767	0.749	0.149
58	200.0	0.4	0.2	0.767	0.75	0.149
95	300.0	0.4	1.0	0.767	0.75	0.149
47	200.0	0.2	1.0	0.767	0.75	0.15

Figure B.14: Componentwise gradient boosting with coxph as loss function on repeated stratified k_fold tuned for n_estimators, learning_rate and subsample on headneck dataset (top 20 hyperparameter combinations). Sorted in decreasing order by: Harrell's concordance index, UNO's C-statistic and IBS



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway