LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Statistical Learning in Survival Models Combining Scan and Clinical Data

Master Thesis

Author: Katharina Hechinger (11352058)
Supervisor: Prof. Dr. Göran Kauermann
Elite Program Data Science
Date: October 8, 2020

Declaration of Independence

I hereby confirm that I have written the accompanying thesis

"Statistical Learning in Survival Models Combining Scan and Clinical Data"

by myself, without contributions from any sources other than those cited in the text and acknowledgements. This applies also to all graphics, drawings, maps and images included in the thesis.

Munich, October 8, 2020

Katharina Hechinger

Contents of the USB stick

The USB stick contains the following files:

- Thesis as pdf format
- Graphics as pdf formats
- R scripts:
  - data_preparation.R: for preparing and preprocessing the dataset
  - data_modelling.R: for the concrete model setup
  - summary.R: for a summary of the results
  - graphics.R: for the creation of the included graphics

Contents

Abstract

The goal of this thesis is the comparison of classical methods of survival analysis and statistical learning approaches adapted for lifetime random variables. With the rise of big data and the resulting challenges for data analysis, traditional models suffer under high dimensional data sets and the lack of flexibility. As machine learning models have proven to overcome these issues for standard classification and regression problems, they are also gaining more and more popularity in the field of survival analysis. In the recent years, many statistical learning approaches have been adapted for handling censored data and predicting the time to events or risks of events.

Survival analysis is very popular in biostatistics, mainly focusing on the death of patients as events. In this work, a number of methods is studied in theory and afterwards applied to a real-life dataset. It consists of information about patients suffering from a rare type of cancer, called soft tissue sarcomas. Hereby, clinical variables were combined with features extracted from their medical scan images, which are also referred to as radiomics. This dataset was analysed using classical survival models as well as statistical learning based methods with the goal of predicting the patients' risks of events and identifying risk factors. The comparison of predictive performances showed an advantage of the more flexible machine learning approaches over traditional survival methods.

1 Introduction

Survival Analysis is a field of statistics that is often used in biomedical applications. Its goal is the analysis of time-to-event data, i.e. the length of time from a starting point to an event of interest. In this work, such events are the timepoints of death of individuals. Therefore, the focus lies on the survival time of patients. The main goals of survival analysis are the analysis of event patterns, the comparison of survival times in different patient groups and the identification of factors that affect the survival time. In this way, it is possible to assess and predict the risk of an event or death for patients and select appropriate treatment accordingly.
The analysis of time-to-event data poses additional challenges to statisticians, due to its unique property of censored observations. In order to handle instances without information about the actual event time, a number of methods have been developed over the last decades.

While the traditional approaches gained great popularity and are still widely used today, the Big Data era asks for novel ideas. With the rise of machine learning and high computational capacities, datasets, especially in the medical fields, are becoming more and more high dimensional. Due to their nature, time-to-event datasets are usually rather small in terms of observations, i.e. events or deaths. When combined with a large number of features, this results in the well known curse of dimensionality.
Another limitation of traditional methods is the lack of flexibility. While most of those approaches rely on strict assumptions, modern applications often demand for a more flexible modelling approach.

These challenges are of course not limited to survival analysis. In the last years, a great effort has been made in the research community to develop novel methods for dealing with high dimensional data in flexible ways. Their foundation lies in Statistical Learning. This field focuses on the prediction of a random variable Y based on covariates $X = x$ using a

random sample of data from a population and assessing the quality of the prediction afterwards. Many popular methods rose from this structure and became part of the general toolkit of data analysis.
Whereas methods like e.g. penalized regression and random forest are already widely used for the analysis of regular data, their application in the field of survival analysis is still rather rare. Adapting the approaches of statistical learning for censored data offers new possibilities for the analysis of time-to-event data.

The goal of this work is the comparison of numerous methods for the analysis of survival data. Hereby, traditional methods are evaluated as well as approaches from the field of statistical learning. After some definitions and explaining the basic theory of survival analysis and statistical learning, a real-life dataset was analysed. The dataset contains clinical data of patients with features extracted from their scan images, as well as information about their survival or censoring times. Using this information, this work aims for the prediction of risk of death using various methods and the identification of special risk factors among a large number of variables.

2 Survival Analysis

As mentioned in the introduction, Survival Analysis is a popular tool in classical Statistics for handling time-to-event data. As expressed by Kleinbaum and Klein (2010), survival or lifetime data analysis is a collection of statistical procedures to investigate data, where the outcome variable of interest is the time until an event of interest occurs. Utilizing this random variable, risks of events can be constructed for any point in time and any patient. These risks or survival probabilities are of major interest in the field of survival analysis. Due to the special data generating process, it contains unique challenges for an analyst. The main and most popular issue is the handling of data instances, of which the progression is only partially known, i.e. the information about them is incomplete. These observations are called **censored**. For achieving satisfying and valid outcomes, censoring should not be equated with ordinary missing data. Therefore, the analysis of such datasets requires specialized methods and models.
In the following, basic theory and the most common methods of survival analysis are explained. Due to convenience, this work will focus on clinical studies, where an event of interest typically equals death of an individual due to a certain condition. Hence, one is interested in the survival probability and risk of death for patients.

2.1 Basic Theory

Statistical analysis always revolves around a certain random variable of interest. For time-to-event data, the focus lies on a special kind of random variable T, which describes the time until an event occurs, i.e. the survival time of a patient. It starts at a particular time point that is defined beforehand, e.g. beginning of the study or a specific surgery related to the study. The start point can of course be different from patient to patient. Therefore, it is necessary to shift the viewpoint of time from calender to duration time, as visualized in Figure 1.
Additionally, the random variable T is of course non-negative, as it describes a duration.

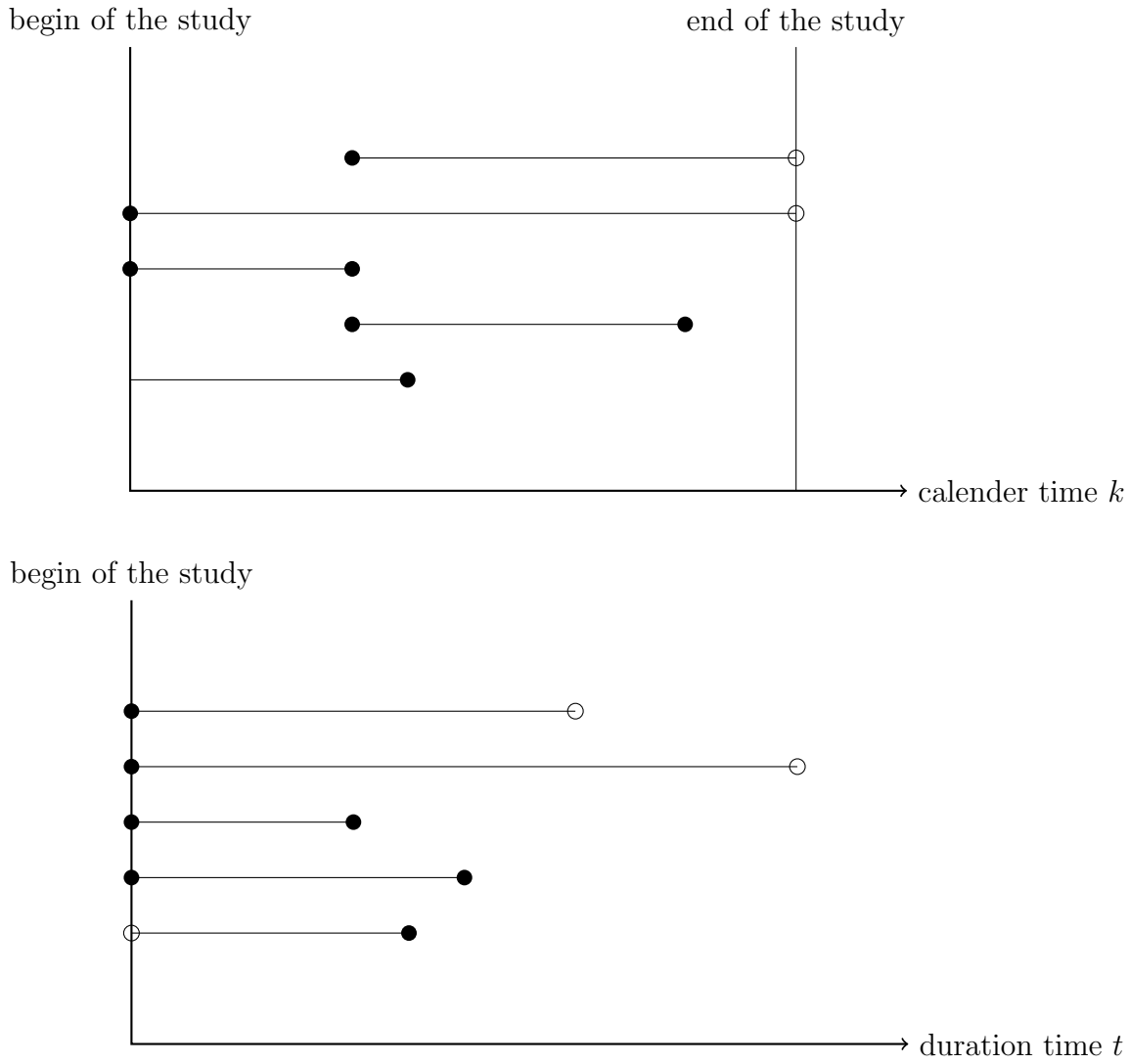


Figure 1: The development of instances can be transferred from calender time to duration time. Each vertical line symbolizes an individual. A transparent dot displays individuals that completed the study without an event or were unobservable in the beginning (transparent dot). In contrast, a solid dot shows that an individual's start and event time could be observed during the study. The instances without event during the study are known as censored observations. Whereas the two individuals at the top of the graph are right censored, the down most observation is left censored.

2.1.1 Foundations

Any non-negative random variable T can be described uniquely by several functions related to the survival of an instance. Of course, known properties of random variables like density and distribution function can be used. But in addition, one also has specific functions at hand to characterize random variables in the context of survival analysis. With density $f_T(t)$ and cumulative distribution function $F_T(t) = P(T \leq t)$, the **survivor function** can be defined as

$$S_T(t) = S(t) := P(T > t) = 1 - F(t).$$

It describes the probability of surviving beyond time t. Another central function in survival analysis is the **hazard rate**, defined as

$$h_T(t) = h(t) := \lim_{\Delta t \to 0} \frac{1}{\Delta t} P(t \leq T < t + \Delta t | T \geq t).$$

The hazard is not a density or a probability but a measure of risk: Loosly speaking, $h_T(t)$ is the rate, at which the survivors in the population at time point T are "falling off the cliff". Alternatively, the hazard function can be represented in terms of the **cumulative hazard rate**

$$\Lambda_T(t) = \Lambda(t) = \int_0^t h(u)du,$$

which in fact resembles accumulated hazards of different time points up to time t.
Using these additional functions to describe random variables has several advantages in the context of survival analysis applications. A well-known example to demonstrate the benefits of the hazard rate over the common density is the age of death in Germany. Looking at the density, it peaks at the age of $\sim$80 and drastically decreases afterwards, which is of course due to the small number of people older than 80. As one is interested in the random variable "age of death", this might not be intuitive though it is of course correct. The hazard rate gives more meaningful insights in this context. It grows exponentially after the age of 80, which indicates a highly increased risk of death for citizens over 80. Therefore, the hazard rate serves as a more intuitive way of describing risks and survival

probabilities. Additionally, using $h(t)$ leads to analytic simplifications, i.e. easier analyses in the presence of censoring.

When the data are subject to right censoring, hazard function representations often lead to easier analyses. For example, one can consider a cohort of N patients, who just turned 50 years old and are followed for one year. If d individuals die during this follow-up period, one can nicely estimate the (discrete) hazard function of the random variable T = age of death. It simply results from the ratio d/N, which is a quite useful property.

Using the above quantities, the distribution of a random variable T can be uniquely described through the following relationships:

$$h(t) = \frac{f(t)}{1 - F(t)} = \frac{f(t)}{S(t)} \tag{1}$$

$$S(t) = \exp(-\Lambda(t)) = \exp(-\int_0^t h(u)du) \tag{2}$$

$$f(t) = -\frac{dS(t)}{dt}. \tag{3}$$

For sequential time points $t_0 = 0 < t_1 < t_2 < ... < t_K$, the survivor function at a certain time t_{k^*} can be formulated as a product of functions of previous time points:

$$\begin{aligned} S(t_{k^*}) &= \exp(-\Lambda(t_{k^*})) = \exp\left(-\int_0^{t_{k^*}} h(u)du\right) \\ &= ... \\ &= \prod_{k=0}^{k^*-1} P(T \geq t_{k+1} | T \geq t_i). \end{aligned}$$

Just as for ordinary random variables, T can follow a number of classical distributions for duration time. Exemplary, the exponential distribution with rate λ describes the time between the occurrence of events in a Poisson process, i.e. events occurring continuously and independently at a constant rate. $Exp(\lambda)$ is a very simple distribution but generally serves as a baseline assuming a (unrealistic) constant hazard. It can be described by the following functions:

Hazardfunction: $h(t) = \lambda$

Survivorfunction: $S(t) = \exp(-\int_0^t h(u)du) = \exp(-\lambda t)$.

Density and distribution function as well as the moments equal those of a "normal" exponential distribution.
Of course, there is a variety of different distribution families for random variables describing survival time. But for most applications, knowing the exact or even approximate distribution of a quantity of interest is rare.

2.1.2 Censoring

As already mentioned, in survival analysis one encounters different problems than in ordinary data analysis. Often, the time of an event is unknown for patients, for example due to the end of a study or other reasons of absence, or the patient's development was not observed from the very start of the study. In this case, the variable T is not fully observed, i.e. the information about the patient is incomplete. This phenomenon is called **censoring**.
In order to handle this issue, an additional random variable C is defined. It describes the maximal observable time of a patient. In any case, either T or C is observed and documented, resulting in

$$T_i^* = \min\{T_i, C_i\}.$$

Based on those two quantities a censoring indicator can be constructed as

$$\delta_i = 1\{T_i \leq C_i\},$$

which takes value 1 for instances, where an event is observed, and 0 for censored observations. This indicator is often used as "survival status", as it also describes whether the patient died during the study or not.

Naturally, one can distinguish two types of censoring. Left truncation of observations, i.e. missing data at the beginning of the observation period, is relatively rare. In contrast, most datasets contain many cases of right censored data, that occurs naturally because of drop-outs of patients or simply the end of a fixed-time study. This work will therefore focus on the second type: right censoring. There are four different forms of right truncation that all demand for different treatment. Censoring of type I means that a deterministic maximum duration time c_i can be observed for individual i, which results in $T_i^* = \min(T_i, c_i)$. Typically, $c_i = c, \forall i = 1, ..., n$ holds in this case. For type II censoring, the duration time in the study ends if a given number of events have been observed. Type III is also called random censoring and refers to censoring times C_i that are independent to the duration times T_i for all instances. Lastly, type IV, also known as non-informative censoring, describes situations, where the distribution of T provides no information about the distribution of C, and vice versa.

In order to perform unbiased data analysis despite censored observations, it is crucial that censoring is non-informative and not caused by systematic circumstances. For a more extensive discussion about the impact of censoring, see e.g. the introductory book about lifetime data analysis by Kleinbaum and Klein (2010).

2.2 Traditional Methods

The two most common tools in the field of survival analysis are the Kaplan-Meier estimate and Cox Regression. Generally, both can be used to investigate the association between survival times of patients and one or more predictors. In contrast to other popular methods in the field of survival analysis, these two approaches belong to the family of non-parametric and semi-parametric models. They offer a data based modelling approach instead of a priori assumptions about the model structure. This is often desired due to complex and high dimensional data sets that do not seem to follow any specific prespecified distribution. Therefore, the flexibility of non- or semiparametric approaches is a preferable property in survival analysis.

2.2.1 Kaplan-Meier Estimate

The Kaplan-Meier Estimate is a nonparametric method for survival analysis and was proposed by Kaplan and Meier (1958). It is one of the simplest methods for estimating the survival over time. It assesses the probability of an event for a subject during a specified time interval in the presence of censoring.

Given ordered and non-censored event time points $t_{(k)}, k = 1, ..., K$, one can define d_k as the number of events or deaths at time point $t_{(k)}$. The number of individuals at risk just before the particular point in time is denoted with n_k as the cardinality of the risk set $R(t_k)$. Being "at risk" means that the patient did not have an event yet and also was not censored before or at $t_{(k)}$. One can use d_k/n_k as an estimate for the hazard rate h_k. Then the Kaplan-Meier Estimate is defined as

$$\hat{S}(t) = \begin{cases} 1, & t < t_{(1)} \\ \prod_{t_{(k)} \leq t}(1 - d_k/n_k), & t \geq t_{(1)}, \end{cases} \tag{4}$$

which resembles a step function that jumps at each time point $t_{(1)}, ..., t_{(k)}, ..., t_{(m)}$. This estimate can serve as a useful tool in combination with specific testing routines for comparing two groups of instances and their statistical differences in survival probability.

Goel et al. (2010) provide a more in-depth explanation and demonstration of the Kaplan-Meier estimate and its practical use. Though it is a very popular method, it is not possible to include multiple and possibly continuous predictors. Therefore, it is only of limited utility in most more complex applications.

2.2.2 Cox Regression

Another frequently used and possibly more versatile tool in classical survival analysis is the **Proportional-Hazard** or **Cox Model**, as proposed by Cox (1972).

It belongs to the family of semiparametric models and therefore offers some advantages to nonparametric approaches, while still maintaining a lot of their flexibility. This approach

can be derived using a time-discrete logit model. The detailed derivation and motivation can be found in the Appendix A. It results in the multiplicative hazard model

$$h(t,x) = h_0(t)\exp(x^T\beta) = h_0(t)\cdot\exp(\beta_1 x_1)\cdot\ldots\cdot\exp(\beta_p x_p). \tag{5}$$

It contains the baseline hazard rate $h_0(t)$, which is identical for all individuals. Therefore, it cancels out upon comparison of the hazard of different patients. It can be a function of arbitrary form and does not have to be specified. As its name already implies, the central assumption of the model is the proportionality of hazard rates. Assuming two individuals with covariate vectors x_1 and x_2, the model follows

$$\log h(t,x_1) - \log h(t,x_2) = (x_1 - x_2)^T\beta,$$

which implies parallel or proportional survival curves. For example, one can think of the comparison of treatment groups. One group receives a new treatment, whereas the other is treated in a traditional way or serves as a placebo group. The proportional hazard assumption states that the ratio between the hazard for an individual in group 1 and the hazard for an patient in the second group remains constant over time.

In order to estimate the parameters of such a model, Cox proposed a partial likelihood approach for survival data. For a setting like

$$X \sim f_x(x|\theta), \theta^T = (\phi^T, \beta^2),$$

i.e. a distribution with two parameters, where β is of primary interest, the idea is to decompose the likelihood. This results in two parts, where the first component depends on β but not on ϕ, and the second one vice versa.
This idea can be applied to the cox model with $\phi = h_0(t)$. Assuming no ties, it results in

the decomposition

$$Li(\beta, h_0(t)) = \prod_{i=1}^{k} \frac{\exp(x_{(i)}^T \beta)}{\sum_{j \in R(t_{(i)})} \exp(x_j^T \beta)} \cdot \left\{ \sum_{j \in R(t_{(i)})} h_0(t_{(i)}) \exp(x_j^T \beta) \cdot \prod_{i=1}^{n} S_0(t_i) \exp(x_i^T \beta) \right\}.$$

$R(t)$ is the set of instances at risk at time point t. Note that the first part does not depend on the unknown parameter $h_0(t_i)$. Therefore, the partial likelihood can be defined as

$$Li_p(\beta) = \prod_{i=1}^{k} \frac{\exp(x_{(i)}^T \beta)}{\sum_{j \in R(t_{(i)})} \exp(x_j^T \beta)}. \tag{6}$$

The parameter of interest β can now be estimated in the usual manner, i.e.

$$\hat{\beta} = \arg\max_{\beta} \log Li_p(\beta).$$

Maximum likelihood inference leads to

$$\hat{\beta} \sim N(\beta, \hat{V}(\hat{\beta})).$$

This estimate is asymptotically unbiased. Note that some information gets lost during this estimation process, as the model does not use the exact event times but only their chronological order. Also, handling of ties can be problematic.

As explained in the beginning, the central assumption of this model are proportional hazards. This implies coefficients β should be constant over time. If this requirement cannot be fulfilled, the regression coefficients have to be modelled over time. There are numerous ways to extend the classical proportional hazard model in order to estimate coefficients in the presence of ties or for modelling time-varying covariates. Also, the cox model can be extended by semiparametric additive predictors. Another useful extension are frailty models. Here, heterogeneity between individuals, that cannot be assessed through the observed covariates, is also taken into account.

Generally speaking, the cox model can be used to estimate the influence of numerous variables on the survival time and can be extended in various ways according to the individual situation.

2.2.3 Limitations

Traditional survival analysis methods have been extensively used during the last decades and still remain popular and useful tools for some areas of application. But today's era of huge data sets poses new challenges and demands for specific adaptations.

A common issue of the cox model is the violation of the proportional hazard assumption. While this is not a new problem and does not necessarily prevent the analyst from using the model, it should of course not be neglected in the first place. A non-proportional hazard corresponds to an interaction of the independent variable, i.e. time to event, with time. In this case, it might be necessary to modify the cox model accordingly. To do so, Borucka et al. (2014) describes two possible methods: introducing time-variable interactions into the model or using a stratifying variable to re-estimate the model for two separate subgroups.
Another frequently occurring problem of cox regression are high-dimensional datasets. While most censored data sets in the medical domain contain a large number of features, the actual number of observations, i.e. the number of events, is often small. This results in a common problem of modern statistics, the curse of dimensionality. Though it is possible to perform variable selection as a separate step before model building, this is often not desirable due to the complexity of the variables and their lack of interpretability.
Lastly, many datasets contain complicated non-linear relationships between the survival time and its predictors. Simple regression models are limited to linear combinations of feature values and coefficients and are therefore not able to capture complex structures.

These aspects show the need for more advanced modelling procedures in the area of survival

analysis. Adapting known statistical methods to the special case of duration time analysis can help to overcome some of the mentioned problems.

3 Statistical Learning

3.1 Basic Theory

Statistical Learning theory is an important foundation of machine learning and is of course all about learning from data. Numerous famous books have been published on this topic, e.g. by Friedman et al. (2001) or Bishop (2006). The most essential ideas will be sketched in the following.

In the general setting of statistical learning, one is interested in making predictions for a random variable Y based on covariates $X = x$ using a random sample of data from the population $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$. To do so, one has to use some kind of prediction procedure that is applied to the features x_i of each instance $i = 1, ..., n$ in order to get the corresponding target value y_i. The goal is to quantify how accurately any possible procedure predicts and to choose the one that performs best.

Any statistical learning method can be decomposed in three main parts: hypothesis space, risk and optimization.

A **hypothesis space** is the family of functions that can be used for prediction. It specifies the general form of the model but not its particular shape. The learning algorithm has to pick a function or model from this set. Consider for example the space of linear models: The hypothesis space is the family of linear functions

$$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...\beta_p x_p.$$

All possible models have to be of this particular form and can only differ in the coefficient vector β, which is the parameter that one is interested in.

The **risk** is a metric that is used to evaluate how good a model performs and to compare different models of the same hypothesis space.

In order to find the optimal model, the risk has to be minimal over the hypothesis space. Therefore, one has to use an **optimization method**, like e.g. gradient descent.

In the following, the focus will lie on the second part of statistical learning: the risk function. The intuitive goal of the procedure is to find a model $f \in \mathcal{H} : \mathcal{X} \to \mathbb{R}^n$ that fulfils

$$y \approx f(x)$$

for $(x, y) \in P_{xy}$. In order to measure the difference between the true value and the prediction delivered by the model, another function is needed. Therefore, one defines the loss function as

$$L : Y \times \mathbb{R}^n \to R.$$

It compares the prediction $f(x)$ and the true value y in a pointwise manner, i.e. for each observation separately. Common loss functions for a continuous random variable Y are the squared error loss or the absolute loss:

$$L_{squared}(y, f(x)) = (y - f(x))^2$$
$$L_{absolut}(y, f(x)) = |y - f(x)|.$$

Both functions measure the deviation between truth and prediction. They set different emphasis on particular statistical features and also differ in their computational and optimization properties. For example, the squared loss offers convenient optimization due to its squared form. In contrast, the absolute loss is not as easily influenced by outliers and of a more stable nature.

The loss function is the main component in (theoretical) **Risk Minimization**. As it is of limited usage to compute the loss function only for the given data points, one wants to take its general expectation. This is defined as theoretical risk

$$R(f) = E_{xy}(L(y, f(x))) = \int L(y, f(x)) dP_{xy}.$$

In order to "learn", the goal is to find the function $f(x) \in \mathcal{H}$ that minimizes $R(f)$. Of course, this is not feasible in practice, as P_{xy} is unknown and can only be estimated with great effort. Such an estimation procedures would require rigorous assumptions on its distributional form, i.e. one would have to sacrifice the desired flexibility of machine learning.

Therefore, **Empirical Risk Minimization** is mostly used in practice. Given a dataset $\mathcal{D}$ with observation pairs $(x_i, y_i) \sim P_{xy}, i.i.d.$, one tries to approximate the theoretical risk $R(f)$ by a version based on the data, the empirical risk defined as

$$R_{emp}(f) := \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)). \tag{7}$$

The goal is to find

$$\hat{f} = \arg\min_{f \in \mathcal{H}} R_{emp}(f).$$

If f is parameterized by θ, like e.g. the family of linear models by the vector of coefficients β, it can be directly written as

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i|\theta)).$$

The empirical risk is of course only an approximation of the true function of interest $R(f)$. However, it is good enough if $\mathcal{D}$ is an unbiased, independent and a large enough sample from the unknown distribution P_{xy}.

Using this general framework of statistical learning, the goal of this work is now to incorporate its advantages and foremost its flexibility into survival analysis. Therefore, it is crucial to adapt the three components of general learning to the specific problem of analysing duration time data. Specifically, the defining feature of survival analysis, namely censoring, is a challenge and demands for changes in traditional methods of statistical learning. In particular, censoring has an influence on the computation of the loss function as the actual loss of censored instances cannot be calculated. There are several approaches to incorpo-

rate the presences of censoring into the process of statistical learning. An option is inverse probability weighting, as proposed by Robins and Rotnitzky (1992). Schemper and Henderson (2000) used imputation methods for handling censored data. Another challenge is the loss function itself, which has to be appropriate for survival data. Several properties of possible functions have been presented in the literature, for a deeper review, see Henderson (1995).

In the last years, a lot of work and research has been conducted to incorporate statistical learning into survival analysis. This work will only focus on a few exemplary methods. A very recent general overview and machine learning framework for the analysis of survival data is e.g. provided by Bender et al. (2020).

3.2 Methods

The following subsection presents different methods that are very popular and commonly used in machine and statistical Learning. They are all based on the three parts described before. In addition to defining the general principles, we will also discuss where and how adaptions have to be incorporated for survival data.

3.2.1 Penalized Regression

A common problem of regression models in general is **Overfitting**. It refers to a model that almost perfectly fits the training data but is not capable of generalizing to previously unseen data. Therefore, the test error will be large. Overfitting can occur due to several reasons: First, a dataset can contain not enough data or data that is too noisy for generalization. Another aspect is the model complexity as the used model can be too complex, i.e. it contains too many parameters. Lastly, an aggressive loss optimization approach can also lead to overfitting if it is not stopped early enough. Whereas the first aspect might be easily solved by collecting more and better data, it is often not feasible in practice. Therefore, it is useful to concentrate on the second and third point in order to avoid overfitting. The contradicting goals are to maximizing the fit and to minimize the complexity of the model at

the same time, which of course results in a trade-off. In order to find a sweet spot between those objectives, regularization is a useful tool in machine learning and statistics in general.

Regularized Empirical Risk Minimization

In order to account for problems like overfitting, it is helpful to adapt the empirical risk defined in (7). Instead, the function

$$R_{reg}(f) = R_{emp}(f) + \lambda \cdot J(f) = \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)) + \lambda \cdot J(f) \tag{8}$$

is used. $J(f)$ is the complexity penalty and can take various forms that will be discussed in the following. The constant $\lambda > 0$ is a tuning parameter for continuously controlling the complexity of the model. One is now facing an optimization problem with two contradicting criteria: Minimization of the empirical risk for a good model fit and minimization of the complexity penalty, in order to keep the models as simple as possible. As shown in formula (8), this is achieved with the help of the tuning parameter λ by using a weighted sum. As λ approaches 0, the regularized risk is reduced to the empirical risk, i.e. optimization results in a complex model without penalization. In contrast, if λ goes to infinity, the models become as simple as possible and the fit is neglected. In order to explain and understand the concrete methods of regularization, it is easier to focus on a concrete model with an appropriate loss function. For now, a linear regression model and the squared loss are used for demonstration purposes. This results in the following regularized risk:

$$R_{reg}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i\right)^2 + \lambda \cdot J(\boldsymbol{\theta}).$$

Ridge Regression

The first possible type of regularization is L_2- or ridge penalization, as proposed by Hoerl

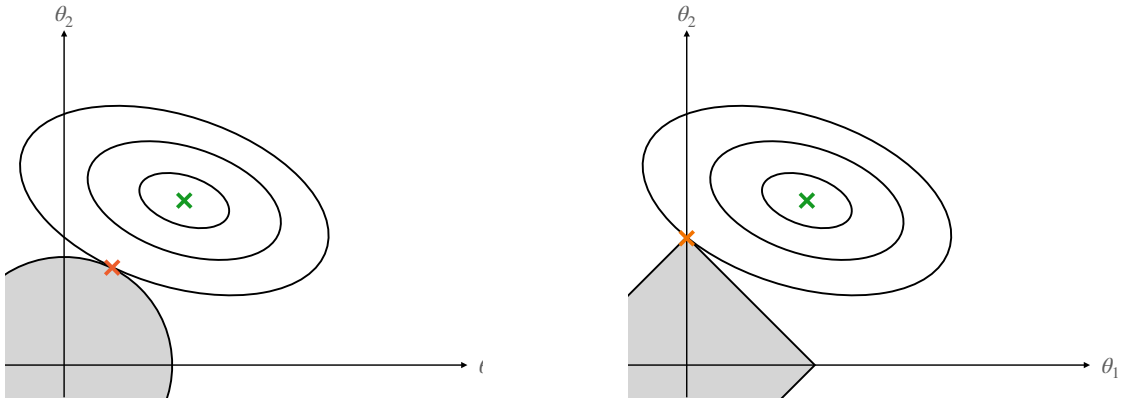


Figure 2: The figure shows the solution of a regression model $\theta = (\theta_1, \theta_2)$, marked as green cross. On the left hand side, the ridge penalty is visualized as a circle around the origin. The solution $\hat{\boldsymbol{\theta}}_{Ridge}$ is visualized as orange cross. The right hand side shows the lasso penalty. The solution $\hat{\boldsymbol{\theta}}_{Lasso}$ lies on the axis and is again symbolized by the orange cross.

and Kennard (1970):

$$\hat{\boldsymbol{\theta}}_{Ridge} = \arg\min_{\theta} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i\right)^2 + \lambda ||\boldsymbol{\theta}||_2^2. \tag{9}$$

This optimization problem with two components can also be reformulated as the constrained problem

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i|\boldsymbol{\theta}))^2 \quad s.t. \quad ||\boldsymbol{\theta}||_2^2 \leq t.$$

Using this form, it is easier to understand and visualize this form of penalization. The goal is still to optimize the empirical risk but the solution has to fulfill the constraint. The solution $\hat{\boldsymbol{\theta}}_{Ridge}$ will have a smaller parameter norm than the unregularized solution. As all features remain in the model and the result is still a dense vector of coefficients, this method is useful if many variables are influential.

Lasso Regression

Lasso stands for Least Absolute Shrinkage and Selection Operator and was originally pro-

posed by Tibshirani (1996). It is another shrinkage method that works similarly but uses an L_1 penalty on the parameter. This results in the form

$$\hat{\boldsymbol{\theta}}_{Ridge} = \arg\min_{\theta} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i\right)^2 + \lambda ||\boldsymbol{\theta}||_1. \tag{10}$$

Of course, it can again be re-written as a constrained optimization problem:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i|\boldsymbol{\theta}))^2 \quad s.t. \quad ||\boldsymbol{\theta}||_1 \leq t.$$

In addition to smaller parameter norms, the lasso method also induces sparsity. As the optimal solution is likely to touch the edges of the diamond shape, some coefficients do not only get smaller but are shrunken to zero. For highly dimensional data set with a high number of dimensions in combination with a relatively small number of observations, i.e. $p > n$, the lasso regularization method selects at most n variables and neglects all others. This is a form of automatic feature selection. In cases of only few influential variables, this can be beneficial. Therefore, lasso regression is in this case often preferred over ridge regression. For handling correlated predictors on the other hand, the L_1 norm does not perform in the desired way. To gain robustness of the model, highly correlated features should receive the same weight. With lasso regularization, this is not the case. Instead, only one of the variables will be included whereas the others are shrunken to zero.

The two penalties can also be combined in the so called **Elastic Net** penalty:

$$\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \leq t.$$

This version contains both penalty terms and α serves as a regulation parameter. Depending on the size of α, solutions can be similar to the ones found by either lasso or ridge regression. Using this technique, it is possible to select more than n features even if $p > n$. Correlated predictors are then either selected together or collectively shrunken to zero.

Therefore, the elastic net combines the advantages of both approaches.

L_0-Regularization

Another way of regularization that is mainly used in the area of neural networks is L_0 penalization:

$$R_{reg}(f) = R_{emp}(f) + \lambda \cdot ||\boldsymbol{\theta}||_0 := R_{emp}(f) + \lambda \cdot \sum_{j=1}^{n} |\theta_j|^0. \tag{11}$$

This norm only counts the number of non-zero parameters in the model and induces even more sparsity in the parameter vector than the L_1 penalty. As the L_0-regularized risk does not have mathematically desirable properties for optimization, it is computationally hard to handle.

Penalized Cox Regression

The regularization approach can of course not only be applied to ordinary linear regression models. In order to tackle the problem of highly dimensional data in survival analysis, several methods for incorporating penalization in the traditional cox regression model have been proposed.

Before diving into the topic of regularized cox regression, it is important to notice the connection between loss functions, which are typically used in machine learning methods, and the likelihood. As mentioned in section 2, in ordinary cox regression the parameters of interest can be obtained by maximizing the partial likelihood Li_p or its logarithm l_p. In contrast, most modern approaches in the field of statistical learning concentrate on minimizing a loss function, i.e. most optimization routines minimize a given function. For practical purposes, it is therefore useful to reformulate the problem. Using the equivalence

$$\arg\max_x(x) = \arg\min_x(-x),$$

which holds for arbitrary x, a maximum likelihood estimate can also be obtained by minimizing the negative log (partial) likelihood. Therefore, $-\log(Li)$ or $-\log(Li_p)$ are special cases of loss functions. Depending on the chosen hypothesis space, this approach based on the likelihood is equivalent to explicit loss functions. For example, the negative log-likelihood in linear regression resembles the mean squared error, i.e. the squared loss function.

Li and Luan (2003) have proposed an L_2 penalization approach for the proportional hazard model in high-dimensional settings with a small number of observations. While this might serve as a useful tool in some applications, the ridge estimate still uses all features for the prediction and thus does not provide variable selection, as explained in the previous part. For medical and biological problems, one expects that only a small number of features might influence the time until an event of interest occurs. Therefore, the focus of this work lies on approaches based on lasso regularization.

The L_1 penalization method for regression models was extended by Tibshirani (1997) for the cox proportional hazard model. He proposed a method that minimizes the partial likelihood with the constraint that the sum of the absolute values of the parameters should be smaller than a constant, following the original regularized model. As described in section 2.2.2, the parameter θ in the proportional hazard model corresponds to the vector of coefficients β and can be estimated through maximization of the partial likelihood in (6). Denoting the log partial likelihood with $l_p(\boldsymbol{\beta})$, the parameter of interest β can be estimated via the criterion

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} l_p(\boldsymbol{\beta}), \quad s.t. \quad \sum_{j=1}^{p} |\beta_j| \leq t. \tag{12}$$

In order to solve this optimization problem, an iterative Newton-Raphson procedure is used. It replaces the weighted least squares step by a constrained weighted least squares procedure. The design matrix of features is denoted by $\boldsymbol{X}$ and the linear predictor by $\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta}$. Using this notation, one can define $\boldsymbol{u} = \partial l_p / \partial \boldsymbol{\eta}$, $\boldsymbol{A} = -\partial^2 l_p / \partial \boldsymbol{\eta}\boldsymbol{\eta}^T$ and $\boldsymbol{z} =$

$\boldsymbol{\eta} + \boldsymbol{A}^{-1}\boldsymbol{u}$. A more detailed explanation of those quantities can be found in Hastie and Tibshirani (1990). With the one-term Taylor series expansion for the partial log-likelihood $l(\beta)$

$$(\boldsymbol{z} - \boldsymbol{\eta})^T \boldsymbol{A} (\boldsymbol{z} - \boldsymbol{\eta})$$

we can solve the problem (12) by using the following algorithm:

Lasso Estimation for Cox Regression (by Tibshirani, 1997)

1. Fix t and initialize $\hat{\boldsymbol{\beta}} = 0$.
2. Compute $\boldsymbol{\eta}, \boldsymbol{u}, \boldsymbol{A}, \boldsymbol{z}$ based on the current value of $\hat{\boldsymbol{\beta}}$.
3. Minimize $(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})^T \boldsymbol{A} (\boldsymbol{z} - \boldsymbol{X}\boldsymbol{\beta})$ subject to $\sum_j |\beta_j| \leq t$.
4. Repeat steps 2 and 3 until $\hat{\boldsymbol{\beta}}$ does not change.

As in an ordinary lasso regression setting, the penalty parameter t is either chosen by the user or determined by the data itself. Note that the cox model does not contain an intercept and therefore, step 3 also does not require the intercept. The minimization procedure is done using a quadratic programming approach as described in Tibshirani (1997). However, as stated by Gui and Li (2005), applying quadratic programming procedures is not possible in settings with $p >> n$, which is often the case in medical or biological areas of application.

In practice, a popular approach is the algorithm developed by Simon et al. (2011), which provides a fast and efficient way to fit the cox model with elastic net penalties. As mentioned before, this implies the maximization of the partial likelihood subject to the combination of ridge and lasso penalty. Setting $\alpha = 1$ of course results in lasso regression. The algorithm works similar to the one presented by Tibshirani (1997) but uses cyclical coordinate descent in the minimization step. This approach was originally proposed for linear regression and extended for generalized regression models later by Friedman et al.

(2007). With the objective

$$M(\beta) = \frac{1}{n}\sum_{i=1}^{n} w(\eta)_i (z(\eta)_i - x_i^T\beta)^2 + \lambda(\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1}{2}(1-\alpha)\sum_{j=1}^{p} \beta_j^2)$$

one can compute its derivative $\frac{\partial M}{\partial \beta_k}$. This results in the coordinate solution

$$\hat{\beta}_{j^*} = \frac{S(\frac{1}{n}\sum_{i=1}^{n} w(\eta)_i x_{i,k}) \cdot [z(\eta)_i - \sum_{j \neq j^*} x_{ij}\beta_j], \lambda\alpha)}{\frac{1}{n}\sum_{i=1}^{n} w(\eta)_i x_{ij^*}^2 + \lambda(1-\alpha)} \tag{13}$$

with $S(x, \lambda) = sgn(x)(|x| - \lambda)_+$. Equation (13) is applied to all elements of β in a cyclic fashion until convergence and the minimization of the overall objective is achieved.
A number of different algorithms is also available for combining regularization with the proportional hazard model. Optimization can e.g. also be done via a combination of gradient descent and Newton Raphson or the LARS algorithm. Simon et al. (2011) compared different approaches regarding runtime for various data situations. The algorithm based on cyclical coordinate descent was significantly faster, particularly in high dimensional settings.

3.2.2 Tree based Methods

The family of tree-based methods is a very widely used and flexible class of machine learning methods. Though they are also quite known for the application in classification tasks, this work will focus on their usage in a regression context, i.e. for the prediction of a continuous variable.

Regression Stumps

The simplest version of a tree-based machine learner is a one-level regression tree, namely a regression stumps. In a univariate setting with $\mathcal{X} \subset \mathbb{R}$ and $\mathcal{Y} = \mathbb{R}$, the idea is to assign the given training data to two subgroups $\mathcal{N}_1$ and $\mathcal{N}_2$ depending on the value of an input

variable:

$$(\boldsymbol{x}_i, y_i) \in \mathcal{N}_1 \Leftrightarrow \boldsymbol{x}_i \leq t$$
$$(\boldsymbol{x}_i, y_i) \in \mathcal{N}_2 \Leftrightarrow \boldsymbol{x}_i > t.$$

This results in two nodes $\mathcal{N}_1$ and $\mathcal{N}_2$ that contain all training observations. The nodes are chosen based on a split variable $x_j = (x_{1j}, ..., x_{nj})$ and a particular value it takes, called split point t. For each of these nodes, a constant prediction can be made:

$$(\boldsymbol{x}_i, y_i) \in \mathcal{N}_1 \Leftrightarrow f(\boldsymbol{x}_i) = c_1$$
$$(\boldsymbol{x}_i, y_i) \in \mathcal{N}_2 \Leftrightarrow f(\boldsymbol{x}_i) = c_2.$$

Therefore, regression stumps can predict only two values c_1 or c_2 and can be described as regression trees that only use a single split.
Using this simple method, one aims to find an optimal splitting variable x_j and splitting point t, such that the risk will be minimal. For quantifying the risk of a possible split, the empirical risks of both nodes are simply summed up: $R(j,t) = R(\mathcal{N}_1) + R(\mathcal{N}_2)$. The risk of a node is then calculated as

$$R(\mathcal{N}) = \sum_{(\boldsymbol{x},y)\in\mathcal{N}} L(y, c)$$

and is minimal for a constant $c = \arg\min_c R(\mathcal{N})$, with respect to a previously chosen loss function. For regression settings, the most common choice is the $L2$ loss, which results in

$$R(\mathcal{N}) = \sum_{(\boldsymbol{x},y)\in\mathcal{N}} (y - c)^2.$$

Regression Trees

Extending this concept to splitting nodes not only once but further in a recursive manner, one ends up with a more sophisticated model: Regression Trees. Though there are nu-

merous methodologies of trees, one of the most popular algorithms is the one by Breiman et al. (1984) named **CART** (Classification and Regression Trees). In a greedy top-down approach, binary splits are constructed to build the tree. Thereby, split variables and split points are selected by exhaustive search, i.e. the algorithm involves iterations over all features and over all possible split points for each feature. Formally, a tree divides the features space into M rectangles R_m and fits a constant model in each of them, which results in a constant prediction

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{I}(x \in R_m),$$

where c_m is the predicted response.

The central question for regression trees is the choice of an appropriate split criterium, as it determines the choice of x_j and t and therefore the final tree. The impurity of the data contained in a node $\mathcal{N}$ can be measured by a function $I(\mathcal{N})$. If a node $\mathcal{N}$ should be split, potential choices for $\mathcal{N}_1$ and $\mathcal{N}_2$ can be evaluated by impurity reduction

$$I(\mathcal{N}) - \frac{|\mathcal{N}_1|}{|\mathcal{N}|} I(\mathcal{N}_1) - \frac{|\mathcal{N}_2|}{|\mathcal{N}|} I(\mathcal{N}_2).$$

For continuous target variables, one usually chooses the mean-squared error or the variance in a node

$$I(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{(\boldsymbol{x},y)\in\mathcal{N}} (y - \bar{y}_{\mathcal{N}})^2,$$

where $\bar{y}_{\mathcal{N}}$ is the mean value of target variable y in node $\mathcal{N}$. The impurity of a node is therefore measured by the variance of y. This is equivalent to the popular ANOVA method. Choosing the split with the minimal impurity resembles maximizing the between-group variance, i.e. sum of squares in a very simple analysis of variance. The remaining error of a node is then the variance of the contained y.

In theory, the CART algorithm could continue the splitting process until each node only contains a single observation. This would again result in an overfitting problem, as ex-

plained in section 3.2.1. The complexity of a regression tree depends on a number of factors and hence can also be controlled by those. There are several ways to control the risk of overfitting: First, it is possible to specifying a minimal number of observations per node for another split. Another way is to set a minimal number of observations in an end leaf- Lastly,the user can also determine a maximum number of tree levels.

Regression trees in general have a number of advantages. Not only are they easy to comprehend and offer an intuitive graphical representation, they are also quite efficient and scale well with larger data sets. In addition, they provide built-in feature selection, which can be in particular of advantage in high-dimensional settings. Nevertheless, single trees are very unstable and highly depend on the data, i.e. small changes can lead to completely different results. Also, the prediction function is a step function and thus not smooth, which sets some mathematical obstacles.

Random Forest

In order to overcome the problems of single regression trees and achieve a better and more stable performance, one often focuses on ensemble methods. In particular, the random forest algorithm proposed by Breiman (2001) is a very popular ensemble approach for trees. One form of training ensemble methods is Bagging (Bootstrap Aggregation). First, M bootstrap samples of size n are created using the training data. On each of those samples, one fits the chosen base learner.The base learners are simple models like e.g. regression trees, that can be easily fit but on their own suffer under high variation. A final prediction is then obtained by aggregating the predictions obtained by the M fitted base learners. This can be either done by averaging or a majority voting. Bagging is a useful method of unstable learners such as classification and regression trees or also neural networks as it reduces the learner's variance. At the same time, it increases the bias in return and complicates the interpretation of the model.

For random forests, a modified version of bagging is used and bootstrapped decorrelated

trees are constructed. The correlation between trees obtained by the bagging procedure is reduced by randomization, i.e. not all but only $mtry$ variables are chosen for fitting a base learner. At each split, $mtry \leq p$ random candidates for the split are drawn. The concrete algorithm proposed by Breiman (2001) works as follows:

Random Forest

Input: Dataset $\mathcal{D}$ of n observations, M, $mtry$

For $m = 1, ..., M$ **do:**

1. Draw a bootstrap sample from $\mathcal{D}$: $\mathcal{D}^{[m]}$
2. Grow tree $b^{[m]}(x)$ using $\mathcal{D}^{[m]}$
3. For each split, only consider $mtry$ random features
4. Grow the tree without pruning or early stopping

End for: Aggregate the predictions of M estimators to predict on new data.

The hyperparameters M and $mtry$ have to be chosen in advance and highly influence the performance of the algorithm.

A useful feature of many random forest implementations is the out of bag (OOB) score or estimate of the misclassification error. It can be calculated easily and is often used for model validation. In bootstrap samples, observations are drawn with replacement, i.e. instances of the original sample can be left out in the bootstrapped versions. These are called out of bag samples and will not be used for training the respective base learner. Therefore, it is possible to use the resulting base model to predict the outcome of the OOB observations. Repeating this procedure for every base learner and its bootstrapped training data, one can define the OOB score as the number of correctly forecasted instances from the OOB sample. In theory, roughly 36.8% of the original dataset are part of the OOB

sample. The probability of not picking n rows in random draws is $(\frac{n-1}{n})^n$ and with growing n, this equals

$$\lim_{n\to\infty}(1-\frac{1}{n})^n = \exp(-1) = 0.368. \tag{14}$$

This way of model validation is useful if the dataset is not large enough to form a separate validation set. The OOB and validation error are not equivalent and thus, should not be compared directly. Nevertheless, the OOB estimate of the error can serve as a good alternative.

Random Forests overcome a lot of the challenges of ordinary regression trees. They are also easy to implement and parallelize and are very flexible, thus can be applied to basically any data. But at the same time, a lot of desirable interpretability, especially for feature interactions, is lost. Also, the procedure is relatively rigid and does not offer possibilities to adapt it according to a specific problem.

Tree based methods for Survival Analysis

Survival trees and forests became very popular non-parametric alternatives to the traditional tools of duration time analysis. Due to their flexibility and ability to identify complex non-linear relationships, (almost) no assumptions have to be made beforehand. Single trees also offer a natural distinction of the population in prognostic groups and can be combined into an ensemble method to form a powerful predictive method. Though the basic principle is always the same, a lot of different approaches have been developed by many groups of researchers. Bou-Hamad et al. (2011) provide an extensive overview over the development of tree-based models for the analysis of censored survival data.

For building a single regression tree for survival data, it is crucial to choose an appropriate splitting criteria, i.e. the function that should be optimized. Over the years, several choices have been proposed and tested in practice. Generally, those criteria can again be divided in

two groups: functions that minimize the risk within one nodes and functions that maximize the separation between two nodes. Shimokawa et al. (2015) provide a comparison of nine different split functions for the application in survival trees. For example, the exponential log-likelihood loss can be used as a criterion, as proposed by Davis and Anderson (1989). As the optimal split here minimizes the loss among all possible splits, this approach aligns well with the general structure of this work and will therefore be discussed exemplary. Let N denote a node in a tree. The sample $\mathcal{D} = \{(t_i^*, \delta_i, x_i), i = 1, ..., n\}$ that is used for training, contains the observation time $T^* = \min(T, C)$, the censoring indicator $\delta = 1\{T \leq C\}$ and the covariates X. The hazard of a terminal node can be defined as $h(x|N) = h_N$, i.e. is constant within the node N. In order to estimate h_N, the maximum likelihood procedure is used:

$$\hat{h}_N = \frac{\sum_{i \in D_N} \delta_i}{\sum_{i \in D_N} x_i}.$$

The exponential log-likelihood loss of a node N is then defined as

$$-\log Li(\hat{h}_N) = \sum_{i \in D_N} \delta_i - \sum_{i \in D_N} \delta_i \log(\hat{h}_N).$$

Thus, an optimal split should minimize this criterion.

As discussed before, single regression or survival trees are of limited utility in practical settings. Therefore, this work will focus instead on an ensemble method, namely **Random Survival Forests** as proposed by Ishwaran et al. (2008). They extend Breiman's random forest model to right censored survival data. The high-level algorithm works as follows:

Random Survival Forest

1. Draw M bootstrap samples from the original data. Each sample excludes on average 37% of the data.
2. Grow a survival tree for each bootstrap sample. At each node, select randomly p candidate variables. The node is split using the candidate variable that maximizes the survival difference between the child nodes.
3. Grow the tree to full size under the constraint that a terminal node should have no less than $d_0 > 0$ unique deaths.
4. Calculate a cumulative hazard function (CHF) for each tree and average them to obtain the ensemble CHF.
5. Calculate the prediction error for the ensemble CHF.

The first central element of the algorithm is growing a single survival tree (see step 2). Just like ordinary binary trees, a survival tree is grown by recursively splitting its nodes. Therefore, a survival criterion is used. The goal is to maximize the survival difference between two child nodes, i.e. to find split variable x_j and split point t, such that dissimilar cases are separated. As discussed before, a large number of possible splitting criteria are available. The `R` package `randomSurvivalForest` in particular offers four splitting rules. An optimal split can be for example found by maximizing the log-rank test statistic. Let therefore $t_1 < ... < t_K$ be the distinct event times in parent node g. Additionally, $d_{i,h}$ and $n_{i,h}$ describe the number of deaths and patients at risk at time t_i in possible child nodes $h = 1$ and $h = 2$. For a split at value t for predictor x_j the log-rank test is defined as

$$LR(x_j, t) = \frac{\sum_{k=1}^{K}(d_{k,1} - n_{k,1}d_k/n_k)}{\sqrt{\sum_{k=1}^{K} \frac{n_{k,1}}{n_k} \cdot (1 - \frac{n_{k,1}}{n_k})(\frac{n_k - d_k}{n_k - 1})d_k}}.$$

Maximizing the log-rank statistic $|LR(x_j, t)|$ leads then to the best split. Alternatively,

the standardized version of the log-rank test can be used. Another possibility is to find the children closest to the conservation-of-events principle and split the node accordingly. The last option provided by the software package is to select a random split for each of the candidate variables in a node and use the variable with the maximum log-rank statistic for the final split.

The next important question is the prediction in the terminal nodes, i.e. how to compute the cumulative hazard estimate for a single terminal node of a tree and the ensemble CHF (see step 4). For one terminal node h and its distinct event times $t_{1,h} < t_{2,h} < ... < t_{K(h),h}$, the CHF estimate is defined as

$$\hat{\Lambda}_h(t) = \sum_{t_{k,h} \leq t} \frac{d_{k,h}}{n_{k,h}}.$$

Hereby, $d_{k,h}$ and $n_{k,h}$ describe the number of deaths and individuals at risk at time $t_{k,h}$. The cumulative hazard function is the same for all cases within the terminal node h. Every new observation with covariates $\boldsymbol{x}^*$ will fall into a unique terminal node h. Therefore, the prediction based on a single tree results as

$$\Lambda(t|\boldsymbol{x}^*) = \hat{\Lambda}_h(t). \tag{15}$$

To compute an ensemble prediction instead, one averages over M survival trees. $\Lambda^*_m(t|\boldsymbol{x})$ is defined as the CHF for a single tree based on one bootstrap sample. Then, the bootstrap ensemble CHF for an observation i can be estimated as

$$\Lambda^*_e(t|\boldsymbol{x}_i) = \frac{1}{M} \sum_{m=1}^{M} \Lambda^*_m(t|\boldsymbol{x}_i). \tag{16}$$

The last step of the algorithm is the calculation of the prediction error. Therefore, Harrell's C-index is a suitable choice. Proposed by Harrell et al. (1982), it estimates the concordance

probability as

$$c = \frac{\#\text{concordant pairs}}{\#\text{concordant pairs} + \#\text{discordant pairs}}. \tag{17}$$

If the model predicts well, patients who had shorter times-to-event should have higher risk scores than patients with a longer event time. The c-index is a popular tool for measuring survival performance, as it accounts for censoring and does not depend on a fixed time for evaluation.

The standard method of random survival forests has recently been extended by Utkin et al. (2019) to a weighted version. The authors replace the averaging that is used for estimating the forest survival function by a weighted averaging to achieve better performance.

3.2.3 Neural Network Approaches

Over the past years, Neural Networks and Deep Learning became very popular approaches for solving many real-life problems. More and more deep learning technologies have been developed and the computational capacities have been extended enormously. Nowadays, it is possible to build any kind of network architecture and achieve outstanding performances, even for very complex problems.

Simple Network Architectures

In order to understand how neural networks work, it is useful to consider a single neuron and its graphical representation as nodes first. The features of $\boldsymbol{x}$ are represented by the nodes of the input layer. Those are connected to a neuron in the output layer by edges that represent weights $\boldsymbol{w}$. The hypothesis space for a single neuron is

$$\mathcal{H} = \left\{ f_w : \mathbb{R}^p \to \mathbb{R} | f_w(\boldsymbol{x}) = \tau \left(\sum_{j=1}^{p} w_j x_j + b \right) \right\}.$$

The function $f_w(.)$ depends on the weights of the neural network. The goal now is to minimize the empirical risk $R_{emp}(w, b) = \sum_{i=1}^{n} L(y^{(i)}, f_w(\boldsymbol{x}^{(i)}))$ with respect to these weights

$w_1, ..., w_p$ and the bias term b. Therefore, a suitable loss function $L(y, f(\boldsymbol{x}))$ has to be defined. The global minimum and therefore the solution to the problem can be found by using optimization methods. A very popular tool for optimizing neural networks, i.e. for finding the best parameters, is gradient descent. After initialization of the parameters, the algorithm minimizes the objective function $R_{emp}(\theta))$ w.r.t $\theta = (w, b)$ by updating them iteratively. Intuitively, it does so by computing the "best" direction along which the weights and the bias should be changed in order to reach the optimum. Therefore, the gradient of the objective function $\bigtriangledown_\theta R_{emp}(\theta)$ is used, as the negative gradient points in the direction of the steepest descent. This leads to the following update rule for the parameters at optimization step t:

$$\theta_{(t+1)} = \theta_{(t)} - \alpha \bigtriangledown_\theta R_{emp}(\theta_{(t)}).$$

Here, the tuning parameter α is used control the magnitude of a step and is also called learning rate. If the rate is too high, the algorithm oscillates. On the other hand, if α is too small, it takes many iterations to reach the optimum. It is often useful in practice to not choose a fixed learning rate but to reduce its value as the training progresses. In modern frameworks for neural networks, many variants of the gradient descent algorithm are available for optimization. Therefore, the procedure can be adapted depending on the practical needs.

Returning to the simplest architectures using only one neuron, the intuition behind neural networks becomes obvious. As the hypothesis space shows, this is just another (more complicated) way of representing ordinary regression models. Substituting $\tau()$ with the identity function leads to linear regression, whereas using the logit function results in logistic regression. Integrating additional hidden layers in the network allows to construct extremely complex and flexible hypothesis space, as visualized in Figure 3.
For simplicity, let's consider a single hidden layer with nodes $z_1, ..., z_M$. Given the input $\boldsymbol{x}$

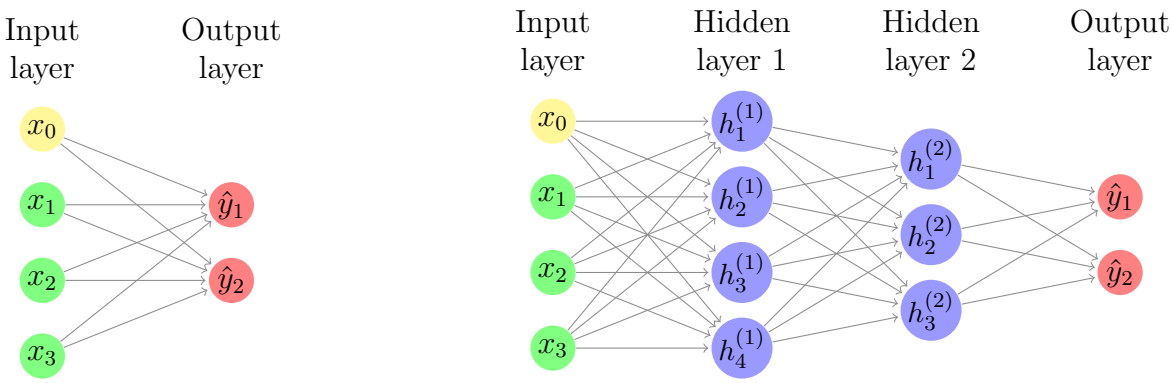


Figure 3: The network on the left side only consists of input and output layer. By adding hidden layers to the network, as shown on the right, it is possible to connect and convert the input values in very flexible ways.

of dimensions $p \times 1$, one needs a weight matrix with dimensions $p \times M$:

$$\boldsymbol{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{p,1} & w_{p,2} & \vdots & w_{p,m} \end{pmatrix}.$$

The value of a hidden node z_m are then obtained by $z_m = \sigma(W_m^T \boldsymbol{x} + b_m)$, where b_m is the bias of the first hidden neuron and σ is an activation function.

Feed-Forward Neural Networks

The first adaption of neural networks for lifetime data analysis was done by Faraggi and Simon (1995). Their goal was to model censored survival data with a very simple feed-forward neural network and its input-output relationship as a basis for a non-linear proportional hazards model. They used a basic architecture with only a single output node. In a feed-forward network, each input is connected to all but one node in the hidden layer. The output of a single hidden layer neural network with M hidden nodes and a given input

vector $\boldsymbol{x}$ can be represented as

$$g(\boldsymbol{x}, \theta) = b_0 + \sum_{m=1}^{M} b_m \phi(w_m^T \boldsymbol{x}) = b_0 + \sum_{m=1}^{M} \frac{b_m}{1 + \exp(-w_m^T \boldsymbol{x})}.$$

For simplicity, θ denotes the vector of all unknown parameters, including weights and bias terms. The product of input values and weights is handed to an activation function is $\phi()$. As for any statistical learning procedure, the term "training" refers to the calculation of the parameter values such that the final predicted outputs $g(x, \hat{\theta})$ are as close as possible to the true outputs. This is done by an iterative procedure called **backpropagation** via gradient descent.

In order to adapt this technology for survival analysis, Faraggi and Simon (1995) proposed to simply replace the linear functional $x_i^T \beta$ in the partial likelihood (6) by the outputs $g(x_i, \theta)$ of the neural network. The proportional hazards model would then be

$$h(t, x_i) = h_0(t) \exp(g(x_i, \theta))$$

with the partial log likelihood

$$l_p(\theta) = \prod_{i=1}^{k} \frac{\exp\{\sum_{h=1}^{H} \alpha_h / (1 + \exp(-w_h^T x_{(i)}))\}}{\sum_{j \in R(t_{(i)})} \exp\{\sum_{h=1}^{H} \alpha_h / (1 + \exp(-w_h^T x_{(j)}))\}}. \tag{18}$$

Using the Newton-Raphson method, the partial log likelihood can be maximized and one obtains the optimal network parameters.
This is a very basic approach that only uses simplest neural network architectures. As experiments and research showed, the proposed network did not necessarily outperfom the linear cox proportional hazards model. Nevertheless, it serves as a base for numerous more sophisticated methods in this direction.

Deep Neural Networks

Nowadays, neural networks can contain hundreds of hidden layers, hence the name "deep" neural networks. A great milestone achieved in the last years was overcoming the challenges of stacking many layers. Increased computational power, huge amounts of data and novel techniques of regularization made it possible to finally implement deep neural network architectures. Those models are able to handle data of highly complex shape and achieve great performances.

Of course, there have also been attempts to use deep neural networks in the context of survival analysis. A recent example is the **DeepSurv** architecture proposed by Katzman et al. (2018). The authors use a cox proportional hazards deep neural network for the application of personalized treatment recommendations.
DeepSurv is constructed as a deep feed-forward architecture, which predicts the effect of covariates on the hazard rate of patients. The hazard is parameterized by θ, the weights of the network. The network itself consists of fully-connected layers of nodes that are each followed by dropout layers. Given an input x, the output of the network $\hat{h}(x, \theta)$ estimates the log-risk function in the cox model. Comparable to (18), the objective function is the average negative log partial likelihood but in this case with additional L2-regularization:

$$l_p(\theta) = \frac{1}{N_{E=1}} \sum_{i:E_i=1} \left(\hat{h}(x_{(i)}, \theta) - \log \sum_{j \in R(t_{(i)})} \exp\{\hat{h}(x_{(j)}, \theta)\} \right) + \lambda \cdot ||\theta||_2^2.$$

Here, $N_{E=1}$ denotes the number of patients, where an event was observed. In the particular use case, DeepSurv was able to outperform a number of other survival methods due to its highly flexible nature.
Nevertheless, the success of deep neural networks also lies in large sample sizes. As explained in the beginning, a large amount of event observations is rare in survival analysis. In most application, only a low number of non-censored samples is given, whereas the dimensionality of the dataset is often high. Therefore, deep architectures might not always be the best choice and other neural network algorithms should not be neglected.

Extreme Learning Machine

Another direction of neural network algorithms is the extreme learning machine (ELM). Wang and Zhou (2018) recently provided the theoretical framework and a software toolkit for adapting ELM to the high dimensional setting of survival data.

ELM was introduced by Huang et al. (2006) for single- and multi-hidden layer feed-forward neural networks (SLFNs). For such networks, one usually uses a backpropagation based algorithm for computing gradients. However, it is known that such procedures have a number of bottlenecks, which the ELM approach tries to overcome. Its particularity is that the parameters of hidden nodes do not need to be tuned. Instead, it is possible to assign values to hidden nodes randomly or to inherit the parameter values from ancestor nodes without affecting the performance negatively. This learning method is comparably simple and can be summarized in the following algorithm:

Extreme Learning Machine (by Huang et al., 2006)

Given: Training data set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in \mathbb{R}^p, \boldsymbol{y}_i \in \mathbb{R}^n, i = 1, ..., N\}$, an activation function $\phi(x)$ and hidden node number $\tilde{N}$

1. Randomly assign input weight $\boldsymbol{w}_i$ and bias b_i, $i = 1, ..., \tilde{N}$.
2. Calculate the hidden layer output matrix $\boldsymbol{G}$.
3. Calculate the output weights $\boldsymbol{\beta}$ of the hidden nodes.

Calculate $\beta = \boldsymbol{G}\boldsymbol{y}$ with the training data target matrix $\boldsymbol{y} = (y_1, ..., y_N)^T$, which contains the true values for the training inputs $x_1, ..., x_N$.

Using the training data set $\mathcal{D}$, the output function in ELM with L hidden neurons is then

$$f_L(x) = \sum_{l=1}^{L} \beta_l h_l(x) = g(\boldsymbol{x})\boldsymbol{\beta}.$$

Here, $g(x)$ is the output vector of the hidden layer with respect to the input $\boldsymbol{x}$, i.e. the

function that maps the data from the input space to the ELM feature space. Using the least squares solution $\beta = G^T(\frac{1}{C} + GG^T)^{-1}y$, the output function of the ELM algorithm is

$$f(x) = g(x)G^T(\frac{1}{C} + GG^T)^{-1}y.$$

If this feature map is unknown, one can make now use of a kernel based version of the extreme learning machine. Following Mercer's conditions, the kernel matrix of ELM is given by $M = GG^T : m_{ij} = g(x_i)g(x_j) = k(x_i, x_j)$ and the output function of KELM results as

$$f(x) = [k(x, x_1), ..., k(x, x_N)](\frac{1}{C} + M)^{-1}y.$$

Wang et al. (2017) proposed a way to combine (kernel) extreme learning machines with methods for analysing right censored data. Their method uses the Buckley-James estimator for "uncensoring" the data. This estimator assumes that the transformed survival time follows a linear regression with normally distributed error terms. Censored survival times can then be imputed accordingly. This corresponds to **Accelerated Failure Time models** (AFT models), which are another class of traditional methods for survival analysis. They provide a parametric alternative to the proportional hazards model if the assumption of constant hazards is violated. Instead, the logarithm of the survival time is expected to follow a linear regression model with normally distributed errors, as explained before. A deeper discussion of AFT models and the comparison to the proportional hazards model can be found in Wei (1992).

In ELM the parameters in the hidden layers have to be chosen randomly and the kernel matrix has to be specified by the user. This induces instability. To overcome this problem, ensemble methods are a popular choice. Using imputed estimates for censored data and an unstable base learner like KELM leads to a lot of diversity and thus enables a good performance of the ensemble learner. Based on this research, Wang and Zhou (2018) published a software package named `SurvELM` that contains six survival models based on kernel ELMs. First, they use the Buckley-James estimator to impute the survival times

of censored observations before applying ELM learning to survival analysis. This is implemented as a single model as well as an ensemble. Secondly, they combine the approach with cox regression. The linear combination of covariates in the cox model is replaced by the non-linear output function of a neural network based on ELM. This can be seen as an extension of the early approach for inducing non-linearity in the cox model by Faraggi and Simon (1995). To achieve more stability, a random forest ensemble of this method is also provided. Thirdly, the authors also propose to use ELM within the boosting framework. It is possible to use ELM combined with gradient boosting or likelihood based boosting. In their experimental results, the authors found these methods to be useful alternatives to traditional models, due to their high efficiency and accuracy.

4 Application

The main focus of this work is of course not the theoretical discussion of and reasoning behind the proposed methods. In order to compare classical and modern approaches to survival analysis it is crucial to apply them to real-life datasets and test them in specific use cases. In general, survival analysis plays a huge role in medicine and especially in the observation of patients and their condition from the time of treatment to time of death. In this area of application, censoring is common and therefore, has to be incorporated adequately in the analysis of lifetime data.
The following chapter will first describe the application of the methods discussed in Chapter 3 to a real-life dataset and then discuss the results and individual performances of models.

4.1 Data

The patients in the study suffer from soft tissue sarcomas. This is a rare type of cancer that begins to sprawl in the tissues. Such tissues connect and support surrounding body structures and can occur anywhere in the body. Soft tissue sarcomas are usually treated by surgical resection, often in combination with radiation or chemo therapy. The particularity of this disease is its huge variety. The spectrum of processes comprises long patient survival to highly aggressive variants that spread in the entire body rapidly. Therefore, it is difficult but of course crucial to choose appropriate treatment for individual patients. Personalized medication in general is an active field of research and can benefit greatly from data analysis and data science.
As found by recent studies, images of the tumor and a number of biomarkers that can be extracted from the pictures are related to the patient's outcome in terms of survival. Analysing structural and quantitative features from tomography images can therefore support the selection of appropriate individual treatment.
Combining the features extracted from tomography pictures with clinical patient data forms a useful dataset. Whereas the former is very abstract and hardly interpretable for non-experts, the latter simply refers to general aspects about patients and their condition.

It comprises broad information about the various disease process of different instances. One goal of this work is to use the provided dataset to assess different methods for predicting survival probabilities. On the other hand, it is also of interest to identify features that are important in terms of prediction, in order to provide interpretability and usability of the proposed methods.

4.1.1 Radiomics

In recent studies, researchers used a novel method called Radiomics for the examination of radiographic medical images. In general, radiomics is a technique for mining quantitative features from standard medical images, that gained a lot of importance in the field of cancer research in the last years. It is able to extract a large number of features, which capture disease characteristics that cannot be seen by the human eye. The overall goal of radiomics is the prediction of prognosis and therapeutic response for different patient conditions. Thus, it provides valuable information for personalized therapy.
A large number of studies has been conducted to show the possibilities offered by the usage of radiomic features. The extracted variables are for example capable of discriminating tumor stages and clinical outcomes. Though this technique offers a lot of possibilities in modern medicine, a lot of work still has to be done in the area. As mentioned by Lambin et al. (2017), standardized data collection, evaluation criteria, and reporting guidelines are still to be found for radiomics.

In particular, the technique is also used for soft tissue sarcomas. The extraction of radiomic features from medical tumor images enables the analysis of important biomarkers, which can be insightful for diagnosis and treatment selection, as explained in the previous subsection. The researchers used the implementation in the python package `pyradiomics` for image preprocessing and obtaining the actual quantities. This results in a large number of features describing shape, intensity and texture, either extracted from the original image or a reconstructed version of it.

4.1.2 Dataset

Data Collection

The provided dataset comprises information from patients of two cohorts suffering from biopsy-proven soft tissue sarcomas. The patients have been treated with surgery in curative intent,, radiation therapy, or chemotherapy based on multidisciplinary review at two medical centers, namely at the Technical University of Munich (TUM) and the University of Washington/Seattle Cancer Care Alliance (UW). The variable of interest is the overall survival time, which was calculated from the initial pathologic diagnosis to the time point of death or censoring. Combining clinical patient information with radiomic features leads to a high-dimensional dataset, which sets the basis for the following work.

Feature Reduction and Selection

Following the work of Peeken et al. (2019), several reduction methods were applied to the dataset before starting model training.

On the one hand, it is advisable to only use a subset of the provided patients. By only including patients with AJCC staging classes 3, 4 and 5, one obtains a more meaningful and robust cohort.

On the other hand, the provided dataset is very high dimensional with only few observations and thus, a preceding feature selection procedure is useful. Two steps of unsupervised reduction were applied in order to guarantee a meaningful dataset. First, the intraclass correlation coefficient ICC(3,1) was calculated for each feature. In general, this statistic describes how strongly units that belong to the same group resemble each other. In this case, the dataset contains three resegmentations or reconstructions of each patient image. Thus, it is useful to quantify the size of the differences between the three image versions and the extracted features. A higher ICC value corresponds to a higher level of agreement between the raters or, in this case, of similarity between the re-segmentations. Therefore, features with an ICC value below 0.8 were deleted and not used in further analyses.

Next, inter-correlated features were removed. The Pearson Correlation Coefficient ϱ measures linear correlation between two variables and ranges between -1 and 1. Calculating the coefficient for each pair of features F_1 and F_2 makes it possible to assess the pairwise similarity. An absolute value of ϱ_{F_1,F_2} above 0.9 across all patients is an indicator for highly correlated variables. As such features are of limited use for the following analysis, feature F_2 was removed from the dataset if $\varrho_{F_1,F_2} > 0.9$.

Another important preprocessing step of the features is standardization. The most popular technique to standardizing is substracting the mean and dividing by the variance for each feature. As this method is not very robust against outliers, another standardization approach was used in this work. For each feature, the median was calculated and substracted from its values, before dividing each result by the interquartile range.

After all described steps of data preparation, the final dataset contained 127 observations and 1189 features. 93 patients were still alive at the end of the observation period and therefore, only provide right-censored information. For 34 patients, the time of death could be observed. A more detailed description of the patient characteristics can be found in Table 1. Here, the target variables are OS (survival time) and survival status, which refers to the observation of an event (= 1) and censoring (= 0). The quantities Age, TNMSuff, TNMT, TNMN, TNMM and Grading are the available clinical information about the instances. The abbreviations refer to a tumor grading system that is often used to assess the severity of disease and choose appropriate treatment by hand. Additionally, Figure 4 shows the cumulative survival distribution function over time, indicating events and censored observations in different colours.

4.2 Model

The following part describes the application of all methods mentioned in section 3. The goal is to model the survival time of patients (OS). Therefore, experiments with tradi-

continuous	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
OS	2.267	21.551	40.145	47.107	74.097	105.519
Age	17	43	57	56.37	69	88

categorical	Level	Count
Survival Status	0	93
	1	34
TNMSuff	a	9
	b	118
TNMT	1	21
	2	104
	k.A.	2
TNMN	0	124
	1	3
TNMM	0	127
Grading	2	50
	3	77

Table 1: Descriptive statistics of the clinical variables as well as the target variables.

tional methods of survival analysis as well as modern statistical learning approaches were conducted.

4.2.1 Model Setup

The most important chunks of `R` code for all models and their configurations described in the following can be found in Appendix B.

Cox Regression

The most popular method for analysing survival data is cox regression, as introduced in section 2.2.2 and implemented in the `R` package `survival`. This method suffers strongly from the curse of dimensionality and thus cannot be used on the dataset without further feature selection.

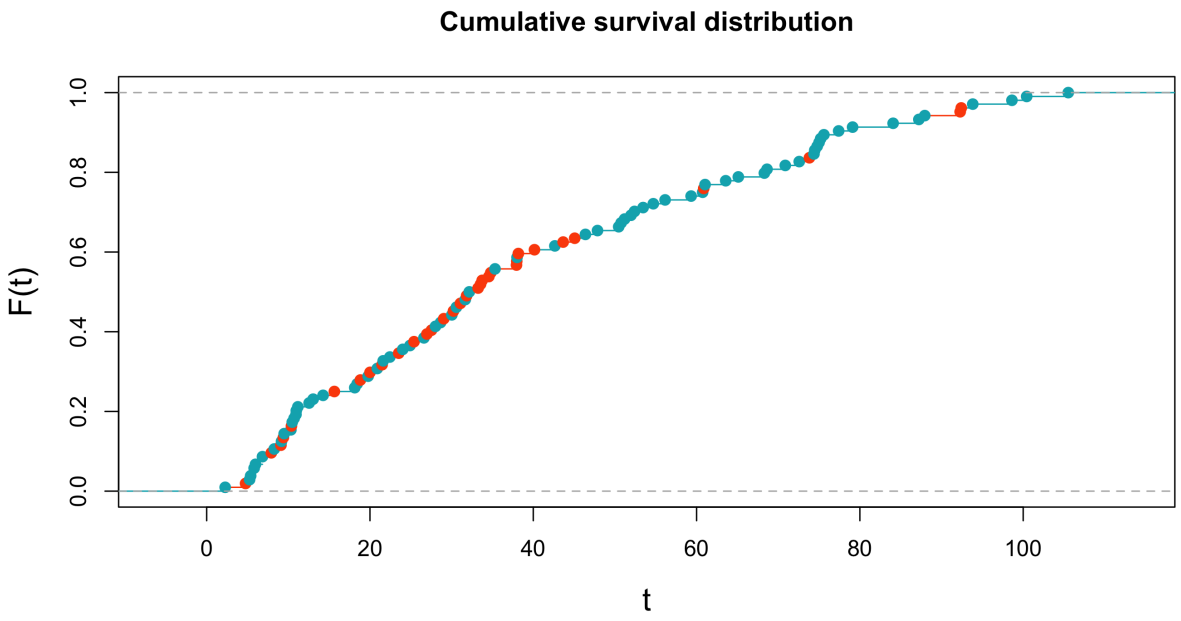


Figure 4: The plot shows the cumulative survival distribution of the patients over time. The red dots indicate events, the blue observations are censored.

Therefore, univariate regression models were built for each feature first. All but one variables were removed and the survival time as target variable was predicted only using this single feature. Examining all regression outputs and all p-values, it was possible to assess the predictive power of all variables in a univariate setting. In order to perform a meaningful multivariate cox regression, the dimensionality of the dataset must not be larger than the actual number of observed deaths ($n = 34$). Therefore, only features with a p-value smaller than 0.0001 were used for a more sophisticated version of the Cox regression. The final model uses the clinical variable `Age` and 14 radiomics features, see Table 2, which comprises hazard ratios, 95% confidence intervals and the respective p-values.

Looking at the large p-values and very wide 95% confidence intervals, it is obvious that some selected variables do not contribute much to the model. Therefore, the features were further restricted. Again only selecting significant features, a Cox regression model was built only using the variables `Age` as clinical information and the radiomic features

Feature	HR	95% CI	p-value
Age	1.07	1.03, 1.10	<0.001
log_sigma_1_0_mm_3D_glcm_Imc2	0.28	0.09, 0.83	0.022
log_sigma_1_0_mm_3D_glrlm_RunEntropy	2.49	0.17, 37.2	0.5
log_sigma_1_0_mm_3D_glszm_GrayLevelNonUniformityNormalized	1.31	0.14, 12.5	0.8
log_sigma_2_0_mm_3D_firstorder_Entropy	1370	0.02, 87586972	0.2
log_sigma_2_0_mm_3D_glcm_JointEntropy	0.05	0.00, 3864	0.6
log_sigma_2_0_mm_3D_glcm_InverseVariance	17.9	0.28, 1152	0.2
log_sigma_2_0_mm_3D_glrlm_GrayLevelNonUniformityNormalized	7.96	0.47, 136	0.2
log_sigma_2_0_mm_3D_glrlm_RunEntropy	0.59	0.07, 5.26	0.6
log_sigma_5_0_mm_3D_glcm_MaximumProbability	1.27	0.27, 5.96	0.8
log_sigma_5_0_mm_3D_glrlm_LongRunEmphasis	0.59	0.00, 93.0	0.8
log_sigma_5_0_mm_3D_glrlm_RunVariance	1.71	0.01, 329	0.8
log_sigma_5_0_mm_3D_glszm_ZoneEntropy	0.72	0.30, 1.73	0.5
wavelet_LLH_glszm_SmallAreaHighGrayLevelEmphasis	0.57	0.36, 0.91	0.018
wavelet_HHL_glszm_SizeZoneNonUniformity	1.47	0.90, 2.38	0.12

Table 2: The table displays the variables in the multivariate cox model based on the univariate p-values, along with the respective hazard ratios, p-values and confidence intervals.

`log_sigma_1_0_mm_3D_glcm_Imc2`,
`wavelet_LLH_glszm_SmallAreaHighGrayLevelEmphasis` and
`wavelet_HHL_glszm_SizeZoneNonUniformity`. This model resulted in a similar performance using only four features. Hazard ratios, confidence intervals and p-values are displayed in Table 3.

Feature	HR	95% CI	p-value
Age	1.06	1.03, 1.08	<0.001
log_sigma_1_0_mm_3D_glcm_Imc2	0.50	0.30, 0.82	0.007
wavelet_LLH_glszm_SmallAreaHighGrayLevelEmphasis	0.54	0.37, 0.78	0.001
wavelet_HHL_glszm_SizeZoneNonUniformity	1.35	1.01, 1.80	0.045

Table 3: The table shows the output of a Cox regression model that only uses the four variables with significant coefficients. Here, the confidence intervals as well as the hazard rate look more reasonable.

Penalized Cox Regression

As an extension of regular cox regression, many `R` packages also provide a penalized version of the regression model. Here, only two variants will be described further.

The package `penalized` by Goeman (2010) enables the user to build generalized linear models with penalized estimation. It does not only support the cox proportional hazards models but also linear, logistic and poisson regression models. With the two tuning parameters λ_1 and λ_2, estimation can be penalized with the lasso, ridge or elastic net approach. The basic function `penalized()` performs a regularized regression for fixed values of λ_1 and λ_2. The focus of this work lies on lasso penalization, as it provides built-in feature selection. The resulting simplicity of the model is desirable for interpretation and identification of risk factors.

As it is often difficult to specify the penality parameters in advance, the package also offers cross validation approaches in order to find optimal values. Using the option `steps = n`, it is possible to visualize the effect of changing the values of λ_1 on the final regression coefficients. At step 1, the algorithm fits the model with the maximal λ_1, i.e. the value that leads to maximal penalization and shrinks all coefficients to zero. In the next steps, models with a successively decreasing penalty parameter are fitted. The algorithm stops as soon as the specified value of λ_1 is reached. The result of this procedure is a list of objects of class *penfit* and the user can plot the estimated coefficients of each step by using the function `plotpath()`, as shown in Figure 5.

For building a cox proportional hazards model, the survival time is used as response to be predicted, i.e. `Surv(time, event)`.

Another implementation of penalized cox regression is offered in the package `glmnet`, as introduced by Simon et al. (2011). Its function `glmnet()` fits the Cox proportional hazards model by using the elastic net penalty, i.e. a combination of L_1 and L_2 regularization. As most applications of survival analysis suffer under high dimensionality in combination with only few observations of actual deaths, this function focuses on underdetermined models and only selects a small number of covariates. In order to perform cox regression, the user specifies `family = "cox"` and uses the survival time as response. The package also offers cross validation approaches and in case of high dimensional datasets, a large number of

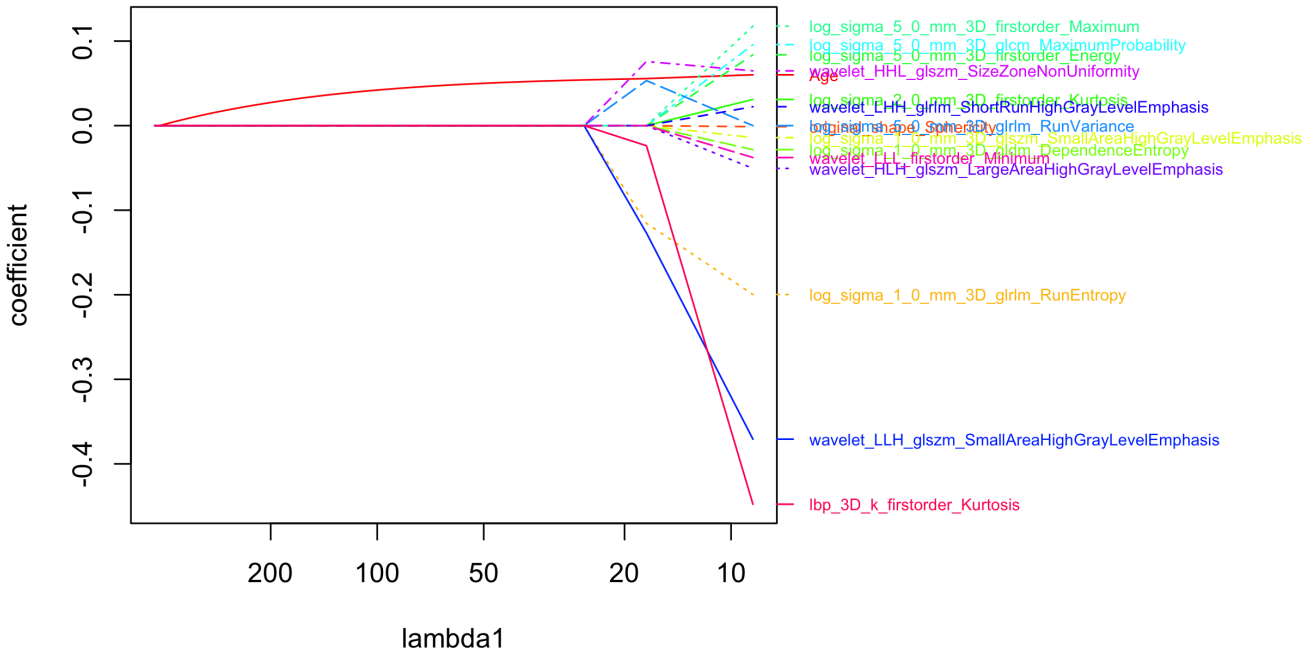


Figure 5: The figure displays fitting steps of a L1 regularized regression. As λ_1 gets smaller, more variables are included into the model. For large values, i.e. strong penalization, only one variable has a non-zero coefficient. At approx. $\lambda_1 \approx 30$, the model gets more complex.

iterations is required. After fitting, an optimal value for λ_1 is returned. In order to evaluate the model, it is possible to visualize the development of performance. A number of different metrics is available, e.g. the concordance index for cox regression. In this case, a higher value equals better results. Choosing the maximum (or minimum for different performance metrics) of this curve, the optimal model and its active covariates can be easily determined.

It is to note here that the functions for performing cross validated lasso regression provided by the two packages deliver different results in terms of the tuning parameter λ as well as performance metrics. This instability of lasso regression combined with cross validation was reported for various applications and does not seem to be a novel issue. Whereas both methods maximize the penalized log-likelihood with penalty parameter λ, the estimation methods differ. A possible explanation can be found in the respective documentations by Goeman et al. (2012) and Friedman et al. (2009). The package `penalized` uses the full gradient algorithm and the Newton Raphson method for finding an estimate for the coefficients. For `glmnet` however, coordinate descent is applied, which is much faster in comparison. The package also provides better documentation and easier usability. For the comparison of performance metrics, only the `glmnet` implementation of lasso regression will be considered in order to avoid confusion.

Tree based Methods

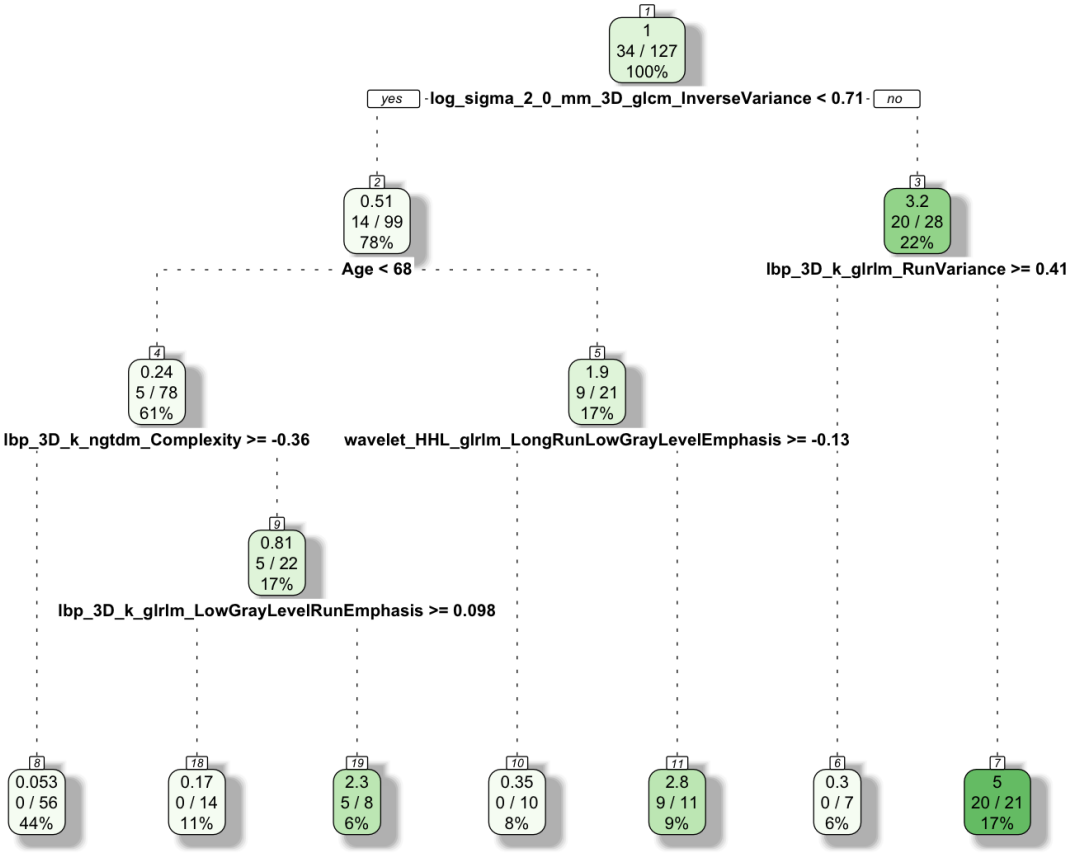
A very popular approach for analysing data without distributional assumptions and flexibility restrictions are tree based methods in all variations, as introduced in section 3.2.2.

As a first step, a survival tree was built using the package `rpart`, which is based on the work of Breiman et al. (1984). Though the single tree might not suffice for actual prediction of the survival time, it can be easily visualized and interpreted and can therefore serve as a good starting point. The function `rpart(formula, data)` builds a full survival tree for the target variable survival time. The resulting *rpart* object can then be pruned using `prune()` in order to avoid overfitting. This function "snips" off the least important splits based on a

tuning parameter *cp* that measures the complexity of the model. It then generates a nested sequence of subtrees. If a split does not decrease the overall lack of fit by a factor of *cp*, it is not attempted anymore. This is not only a way of avoiding overfitting, but also saves computing time. If the parameter is set to a larger value, the resulting tree will be smaller, i.e. less splits will be attempted. Of course, the main advantage of single trees is their interpretability and the option to visualize them. Figure 6 shows a the full version as well as two pruned version of the final tree. This method also enables the user to inspect the differences of the groups that are formed by the terminal nodes. It is possible to inspect separate Kaplan Meier curves for the survival probability in each group and compare them.

In order to get a more sophisticated model, the package `ipred` can be used to perform bagging of survival trees. Using the parameter *nbagg* in the function `bagging()`, the analyst can specify the number of bootstrap samples that should be aggregated. For this usecase, 100 base learners were combined. By setting `coob = TRUE`, the function also returns the out-of-bag estimate for the misclassification error. The package only provides a very basic implementation of the bagging principle and is very slow in terms of runtime, compared to the other methods.

Another common package for random forest is `randomForestSRC`. Its main function `rfsrc()` is able to handle a variety of data settings, not only including classification and regression but also survival analysis with right censored observations and competing risks. The user can specify a number of tuning parameters, like e.g. the number of trees *ntree*. Another important information is the splitting rule, as already discussed in theory in section 3.2.2. For survival settings, the package offers three splitrules: `logrank` (implements log-rank splitting), `bs.gradient` (gradient-based brier score splitting) and `logrankscore` (log-rank score splitting).
Again, 100 base learners were trained, specified through `ntree = 100`.



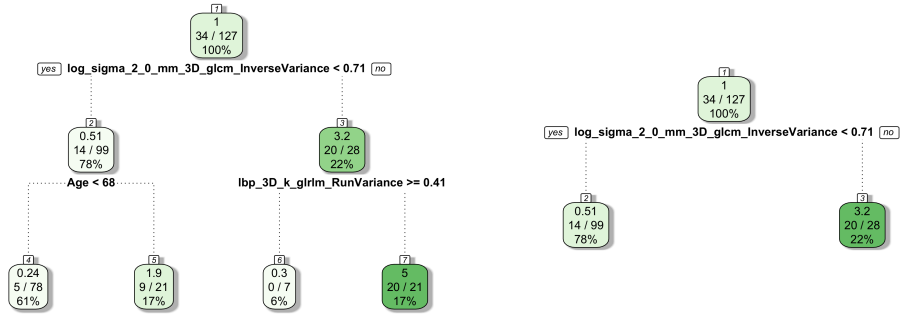


Figure 6: The upper figure shows the survival tree produced by the function **`rpart`**. *Split variables as well as the node sizes are displayed in this graph. Adjusting the complexity parameter and setting it to* $cp = 0.1$ *or* $cp = 0.2$ *leads to a smaller tree, displayed in the two bottom visualizations.*

Neural Network Approach

Lastly, the dataset was analysed using two approaches based on neural networks.

First, the package `SurvELM`, as described in theory before, was used. It provides six different methods, which are all implemented in the fashion of "black boxes". Additionally to specifying target vector and feature matrix and the algorithm, the user can only choose between four kernel types. In this work, the function `ELMCox` was used, i.e. a combination of extreme learning machine and classical cox regression. It produces an output of the same type as the function `glmnet()` that was used for penalized regression before. Therefore, performance measures as well as predicted values could be extracted analogously for the ELM version. Though Wang and Zhou (2018) stated that the methods perform well in their experimental studies, they struggled to achieve comparable performances on the STS dataset. This might be due to the very small number of events.

Additionally, a very simple feed forward network architecture was implemented from scratch, following the work of Drysdale (2018). The goal is to predict hazards with a network consisting of L hidden layers with a particular number of nodes. Therefore, several functions had to be defined beforehand. First, **forward propogation** was needed to pass the information about the weights through the network. As the prediction is a nested function of L activation functions, a for-loop was implemented. Second, the derivative of the weights of each layer with respect to the loss function was calculated. This could be done in a partial manner, by deriving partial derivatives of the activations, the linear inputs and the weights. Starting with the terminal layer and iterating back one layer at a time, the information, i.e. the derivatives of the weights, are stored in a cache.With the help of the chain rule for derivation, **back propogation** is performed to get the derivatives. These two main functions were combined with a weight initialization and a parameter update into a wrapper method. The user specifies the number of layers and nodes per layer, as well as the features and targets, i.e. survival times and censoring indicators. The network results in final weights, that can again be used to make hazard predictions for a dataset.

The implemented functions only provide basic functionalities and can be extended in various ways, e.g. by adding regularization or multiple output nodes for multi-class tasks.

4.2.2 Model Validation

In machine learning, model validation and tuning of parameters is usually done by a technique called K-fold **Cross Validation**. It generally works as follows: The available data is divided into K folds. Then, the model is fit on $K-1$ of these folds and evaluated on the remaining fold in terms of prediction accuracy. This process is repeated K times, i.e. the observations in each fold are used as test set once and K times as training set. Whereas the procedure is straightforward in normal statistical analysis, it gets more complicated for cox regression and in particular penalized versions. For cox regression, it is not quite obvious how to quantify prediction accuracy on a test set. The estimated coefficients only allow to set the risk of each patient in relation to other observations. A very basic approach to assess the model's predictive accuracy is to calculate the partial likelihood based on the observations in the test set. Then, the risk for each patient can be set into relation to the risks of other members of the test set. However, this approach is only of limited use if the dataset does not contain many events. To overcome this issue, several versions of cross validation for survival methods have been proposed and implemented, e.g. by Simon et al. (2011) in the software package `glmnet` or more recently by Dai and Breheny (2019).

In this work, general train and test splits were not used for model validation. The dataset does not provide a sufficient number of observations and therefore, further reducing the events by splitting the dataset is not advisable. 10-fold cross validation was only used for determining the complexity parameters in penalized cox regression. The packages described above already provide implementations and options for determining optimal parameters. When plotting the results of e.g. `cv.glmnet`, one can assess the achieved performance for different values of the tuning parameter. Figure 7 shows the concordance index for a changing λ.

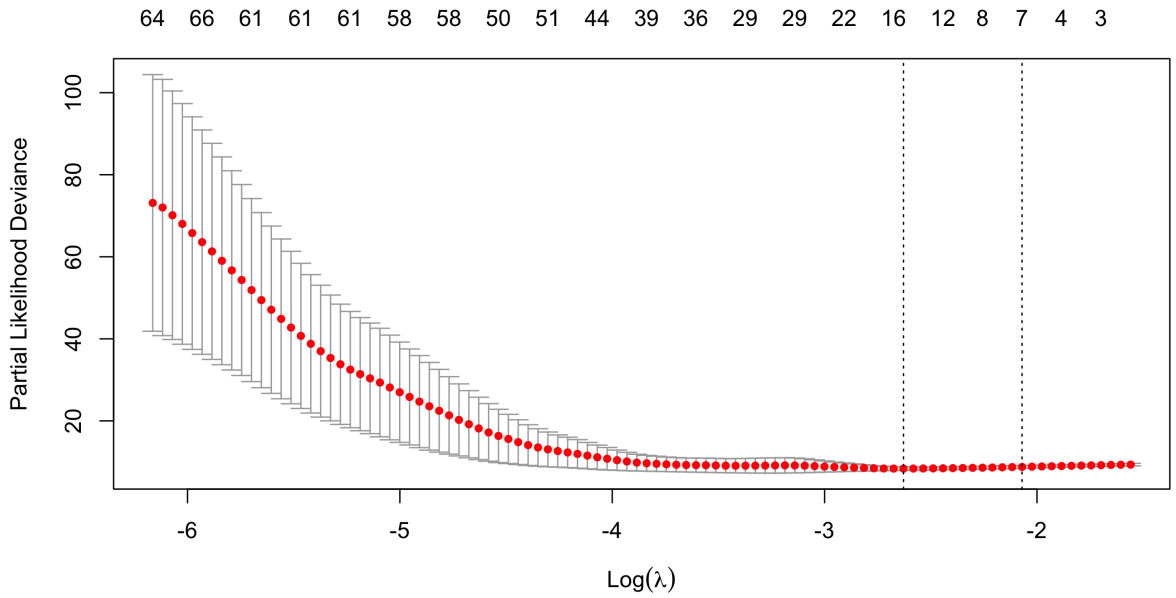


Figure 7: The figure shows the development of the partial likelihood deviance and its confidence intervals for different values of the tuning parameter $log(\lambda)$.

Nevertheless, it is important to assess the internal performance and stability of the models. Therefore, a classical bootstrap procedure was used in this work. As explained in section 3.2.2 for the bagging technique, bootstrapping refers to randomly drawing observations with replacement from the original dataset. Repeating this process several times results in B bootstrap samples of size n, that can contain duplicate observations. One can build the models based on each bootstrap samples, which of course results in B slightly different versions. These can be used to calculate performance metrics based on either the bootstrap sample, which would result in a training performance, or the whole sample, i.e. a dataset containing seen and unseen observations. By using all available data, it is possible to indicate an upper limit to the expected model performance in other settings and avoid the over-optimism of the pure training performance. Averaging the B bootstrap performance metrics leads to a more honest estimate for the model. It is also possible to inspect the variation of performance for the B different model versions. A low variance indicates more

stability. For more details on this approach, the work of Efron and Tibshirani (1994) provides an intensive introduction to the bootstrap.

4.2.3 Evaluation Metrics

As seen in the previous sections, numerous methods for handling survival data are available. Therefore, it is crucial to evaluate their performance, i.e. their ability to predict future data. An overview of possible methods and metrics is for example provided by Graf et al. (1999) or more recently by Chen et al. (2012). It is possible to divide the population into groups and test for significant differences in those groups in order to compare them. Additionally, metrics that focus on the relationship between the risk score and the survival are available.

A very general approach is of course to look at suitable loss functions, as introduced in Chapter 3. The inaccuracy can be measured in a pointwise manner by simply adding the values of the loss function for each true data point and its prediction. Whereas this is a very flexible and general approach, not all methods provide a straight-forward loss function. Therefore, it is useful to take a look at specifically defined measures that can be calculated for arbitrary models.

In order to assess the performance, the models generated from the training data were used to estimate the risk scores. Using these fitted values, metrics can be calculated for each model for the comparison of their predictive performance. In the following, the most important and central measures will be explained shortly.

Hazard Ratio and R^2

Of course, it is possible to compare the predictive performance of models in terms of the estimated hazard ratio and the reported coefficient of determination, R^2. The latter refers to the proportion of explained variation by the cox model fitted on the training data and of course can be used as a goodness-of-fit measure between the predicted risk scores and the patients' true survival times. Though this measure is reported as R^2 in most model outputs,

the values are not the same as in ordinary regression outputs and are therefore often called "pseudo-R^2". In the literature, several types of these pseudo version have been proposed. The R package `coxph` calculates the reported R^2 values as the improvement in the (partial) likelihood when comparing the fitted model against the model without any predictors.

Brier Score

The Brier Score is a score function that measures the accuracy of any probabilistic prediction and can be thought of as a cost function. It was originally proposed by Brier (1950). In survival analysis, it is used to assess the accuracy of a predicted survival function at a specific time point. The Brier score is defined as the average squared distance between the true survival status and the predicted probability of survival. It can take any value between 0 and 1, with 0 being the best value that is possible. In the absence of censoring and given a dataset $\mathcal{D} = \{(t_i^*, \delta_i, x_i), i = 1, ..., n\}$ and the predicted survival function $\hat{S}(t, x_i)$ at some point in time t, the Brier Score is generally calculated as

$$BS(t) = \frac{1}{n}\sum_{i=1}^{n}(1_{t_i^*>t} - \hat{S}(t, x_i))^2.$$

Of course, if the dataset contains right-censored observations, the formula has to be adapted. By estimating the conditional survival function of the censoring times C by $\hat{G}(t) = P(C > t)$, the Brier score results as

$$BS_c(t) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{(0 - \hat{S}(t, x_i))^2 \cdot 1_{t_i^*>t,\delta_i=1}}{\hat{G}(T_i^-)} + \frac{(1 - \hat{S}(t, x_i))^2 \cdot 1_{t_i^*>t}}{\hat{G}(t)}\right).$$

It is important to note that this version of the Brier score only considers a single time point.

In order to assess the model performance at all possible times, the **integrated Brier score** is used:

$$IBS(t_{max}) = \frac{1}{t_{\max}}\int_0^{t_{\max}} BS(t)dt.$$

Though the Brier score is a very popular measure, it has its shortcomings in situations with very rare or frequent events, see Benedetti (2010). In such cases, small forecast changes are significant for rare events but the score is not able to discriminate between them.

C-Index

Another very popular tool for the comparison of survival models and their performances is the concordance index or c-index. Generally speaking, it refers to the agreement between the prediction and the truth and was proposed by Harrell et al. (1982). This index can be seen as a generalization of the area under the ROC curve (AUC) that is able to account for censoring. It compares concordant and discordant observations. In survival analysis, two observations are called concordant if the risk prediction for an event and the time of the actual event align. Therefore, the predicted risk should be lower for an observation with a later event time. As defined in (17), the index relates the number of pairs and results in values between 0 and 1. The latter corresponds to a perfect model forecast, a value of 0.5 indicates a random prediction.

The c-index is very popular and widely used as it is easy to compute and interpret, but of course it also has its disadvantages. With an increasing amount of censoring, the measure tends to be too optimistic and hence, induces an upward bias. In order to overcome this issue, solutions and variants have been proposed in the literature, see e.g. Uno et al. (2011).

4.3 Results

The first goal of this work is the identification of risk factors. Due to the massive amount of radiomic variables, it is essential to determine, which of them actually influence the survival of patients. Therefore, one can extract the variables used in specific models. Most of the mentioned and implemented methods perform internal feature selection and therefore exclude features with a low predictive power automatically. Table 4 shows how many of 1189 available variables were actually included in each model. The cox regression model contains 15 features that were selected in the univariate procedure described before. As

seen already in Table 3, only four out of 15 coefficients are significantly different from zero. For the penalized regression models with L1 regularization, 19 and 6 features were selected. The parameter λ_1 was in both variants determined by an implemented cross validation procedure with 10 folds. The single survival tree used 34 different split variables. Due to their definition, the bagging methods contain more features for the construction of 100 base learners. Nevertheless, the model built with `ipred` only contains 36 variables that are utilized by five or more single trees. For the random survival forest, 226 features were used and achieved a variable importance score larger than zero. This suggests large variability of the bagging approaches, which is desirable to achieve a stable ensemble performance.

Method	Number of included features
cox	15
cox.penalized	19
cox.glmnet	7
rpart	34
ipred	36
rsf	226

Table 4: The table shows the number of features included in each model. Note that the methods `ipred` *and* `rsf` *perform bagging with nbagg* $= 100$.

In addition, it is also useful to take an explicit look at the chosen columns and the associated variable importance score. For a huge amount of features and a small number of observations, it is possible that selection of variables does not happen because of a valuable relationship with the outcome variable. Instead, features might just be selected by chance and do not actually contain information about the survival of patients. Therefore, it is useful to inspect the number of times a feature was used as well as its importance within the model. For all methods and the utilized variables, these importance scores were standardized to take a value between 0 and 1, in order to achieve comparable dimensions. In Table 5, features that were chosen by more than two models are shown. In addition to the number of occurrences, the table also shows the importance score the variable received

from each model. 237 features were chosen by one method only, 28 were selected by two methods and four features occurred in three or four models, respectively. Three variables were included in four out of the six models and four of them, including the only clinical variable ”Age”, were even selected by five methods.

Variable	Sum	Count	cox	penalized	glmnet	rpart	ipred	rsf
log_sigma_2_0_mm_3D_glcm_InverseVariance	1.704	3	0.013	0	0	1.000	0	0.691
log_sigma_2_0_mm_3D_glrlm_RunEntropy	0.794	3	0.0004	0	0	0	0.549	0.245
wavelet_LLH_glszm_HighGrayLevelZoneEmphasis	0.568	3	0	0.012	0	0	0.514	0.042
original_glrlm_RunPercentage	0.445	3	0	0	0	0.426	0	0.019
log_sigma_1_0_mm_3D_glcm_Imc2	0.831	4	0.0002	0	0.124	0	0.631	0.076
log_sigma_2_0_mm_3D_glrlm_GrayLevelNonUniformityNormalized	1.404	4	0.006	0	0.038	0.841	0	0.519
wavelet_HHL_glszm_SizeZoneNonUniformity	0.194	4	0.001	0.099	0.086	0	0	0.007
Age	1.589	5	0.0007	0.103	0.0003	0.485	0	1.000
log_sigma_1_0_mm_3D_glrlm_RunEntropy	1.210	5	0.002	0.156	0.475	0	0.528	0.049
wavelet_LLH_glszm_SmallAreaHighGrayLevelEmphasis	3.025	5	0.0003	0.715	1.000	0	0.769	0.540
lbp_3D_k_firstorder_Kurtosis	1.863	5	0	1.000	0.045	0.309	0.462	0.047

Table 5: The table contains features that were used in more than two of the presented models and their respective times of occurrences.

The second aspect to consider is of course the predictive performance of the individual models. Overall model performance was assessed using Harrell's c-index and the integrated Brier score. In order to internally validate the models, a bootstrap procedure was performed. Therefore, all models were fit based on $B = 200$ randomly generated bootstrap samples of the same size as the original data. The results could then be used to evaluate the performance on the bootstrap data set and also the full data set (containing in-bag and out-of-bag samples). This results in a more honest estimation of performance and indicates the upper limit to the expected performance in other settings, as explained before already. In Table 6, the mean performance measures based on the bootstrap samples are shown for all models.

Model	Harrell's concordance index	Integrated Brier Score
cox	0.8584269	0.08018208
glmnet	0.9618095	0.04668102
rpart	0.8864362	0.07502253
ipred	0.9095640	0.14457626
rsf	0.9146984	0.06999434
coxnn	0.8595210	0.08278964
ELMCox	0.8215034	0.13724042

Table 6: The table shows the performance measures for different models for internal validation purposes.

First, higher values of the c-index indicate better performance, as already discussed. The metric shows that the penalized cox regression model built with `glmnet` achieved the best results, followed by the random survival forest and the bagging of survival trees. Nevertheless, all values are comparably high. Second, an integrated Brier score close to zero is desirable. The smallest scores were again achieved by the `glmnet` approach and the random survival forest. The bagging procedure based on `ipred` on the other hand performed worst in terms of the IBS. Summarizing these aspects, it is to note that the penalized regression routine from the quite popular package `glmnet` performs best, followed by more

sophisticated tree based methods. The approaches based on neural networks only managed performances comparable to the classic Cox regression model.

The bootstrap procedure also offers the inspection of the variance of the performance measures through the 200 random samples. This can be easily visualized in form of boxplots. The borders of the box indicates the 0.25 and 0.75 quantiles, the line inside symbolizes the median. Figure 8 shows the boxplots for C-index and integrated Brier score, for the full data set as well as the bootstrap data. Whereas all boxplots for the ELMCox model are rather large, the other models show smaller performance variance on the bootstrap samples than on the full data set. Again, one can observe that the penalized regression based on the `glmnet` package performs consistently best.

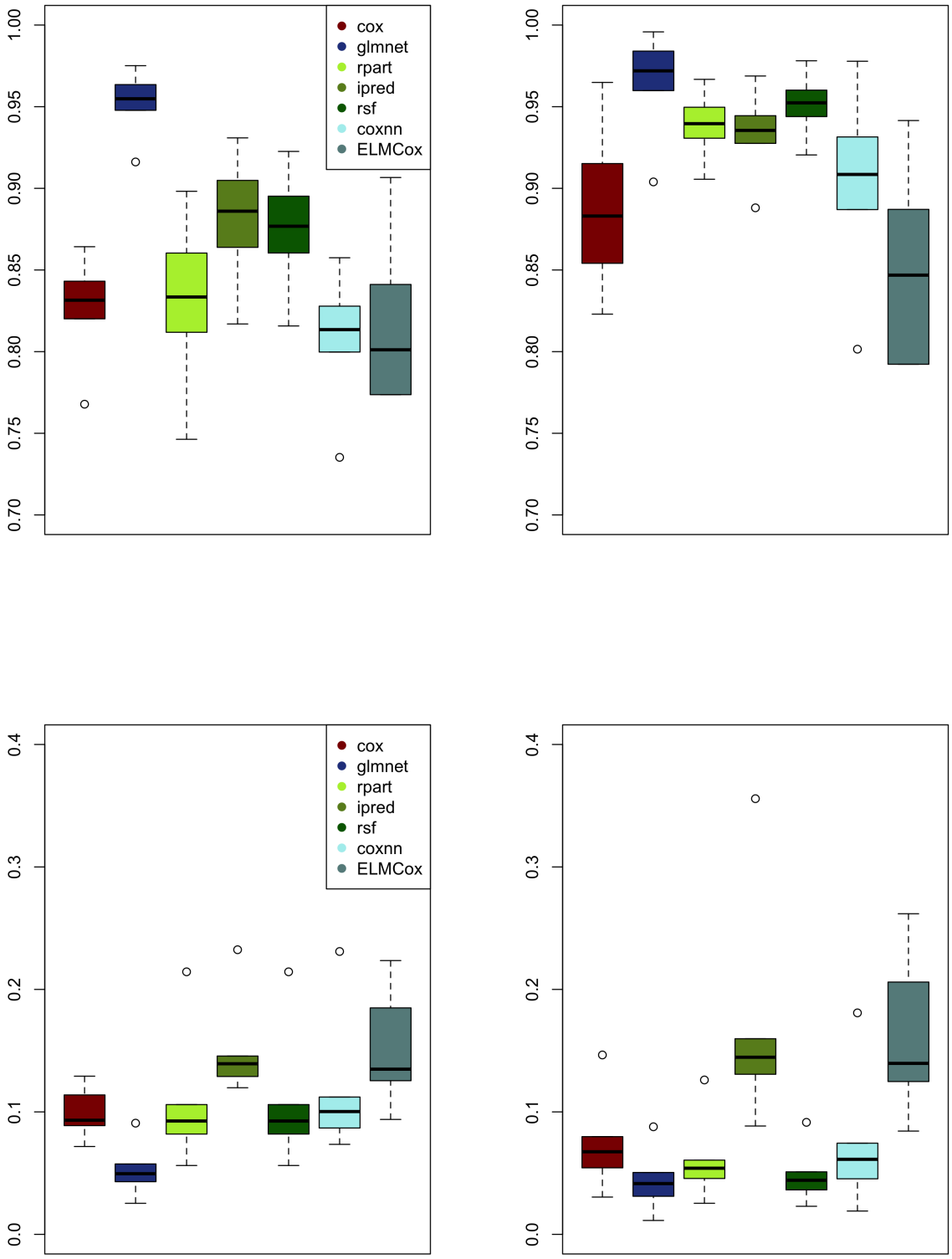


Figure 8: The upper row shows boxplots for the bootstrapped c-index values for each model. The left plot contains the values for the full sample, whereas the values on the right side were calculated based only on the bootstrap sample. The bottom row shows the same plots for the integrated Brier score.

5 Conclusion

5.1 Summary

The goal of this thesis was the comparison of different methods for the analysis of survival data. Whereas classical approaches like the Cox proportional hazards model are very popular in this field and have been widely used for decades already, the recent developments in the field of statistics and data science ask for modifications and more flexible approaches. Methods of statistical learning are already quite famous for their performances in classification and regression problems but did not receive the same attention in the analysis of lifetime data. This work aimed to compare some of these novel approaches to the classical procedures.

First, the basics of survival analysis were introduced in Chapter 2, including the theory behind the Cox regression model. Chapter 3 discussed the idea of statistical learning in general, as well as selected popular methods. After explaining their general functionalities, adaptions for survival analysis were introduced. In Chapter 4, the models were applied to a real-life dataset to predict the risk of death for patients suffering from a special kind of cancer. Along with some clinical variables, a large number of features extracted from medial images was available. The highly dimensional dataset served as a good basis for comparing the different methods. Whereas classical Cox regression suffers from such settings, statistical learning approaches are able to handle many variables due to regularization or internal variable selection. The comparison of performance metrics showed that regularized regression models and random survival forests are superior to other methods for this kind of use case.

Apart from overall performance of the models, another relevant aspect was the identification of risk factors, i.e. features with a high variable importance. Therefore, the number of occurrence of features within the different methods and their respective importance scores were examined. Starting out with 1189 variables, most models only used a much smaller

number of features. 11 quantities were included in at least half of the discussed methods, which indicates an relevance for risk prediction. The only important clinical variable was the age of patient, all other identified risk factors are radiomic features. It is left to the biomedical expertise to further interpret those results but the results can serve as a starting point for continuing investigation.

5.2 Outlook

Though a lot of work has already been done in the direction of incorporating statistical learning in the field of survival analysis, there is still a lot to come in this particular field. Whereas regularized regression and tree based methods are already available in various packages in `R`, suitable implementations of neural network approaches are still rare and to be extended in the future. Most of the time, the main bottleneck is the size of the datasets. With typically only few observations or events, deep learning and other popular neural network approaches are not suitable for the analysis of survival data. This is a topic of extensive research at the moment, which offers great possibilities for future work.
Nevertheless, the discussed methods already provide great improvements in terms of flexibility and high dimensional datasets. In the light of the developments during the last years, this greatly enhances modern survival analysis.

A Derivation of the Cox proportional hazard model

The Cox regression model can be motivated utilizing a time-discrete logit model. For a small additional time Δt, one has

$$h(t,x)\Delta t = P(t \leq T < t + \Delta t | T \geq t, x) + o(\Delta t),$$

where the latter part is irrelevant. The continuous duration time T can be converted to a discrete grid, namely $0 = t_0 < t_1 < ... < t_j < ... < t_J$, which leads to the discrete random variable

$$T^* = t_j \Leftrightarrow t_j \leq T < t_{j+1}, j = 1, ..., J.$$

For T^*, a discrete hazard can be formulated in terms of conditional probabilities:

$$\begin{aligned} h_j^*(x) &= P(T^* = t_j | T^* \geq t_j, x) = P(t_j \leq T \leq t_{j+1} | T \geq t_j, x) \\ &= \frac{P(t_j \leq T \leq t_{j+1}, x)}{P(T \geq t_j, x)} = \frac{P(t_j \leq T, x) - P(t_{j+1} \leq T, x)}{P(T \geq t_j, x)} \\ &= 1 - \frac{\exp(-\int_0^{t_{j+1}} h(t,x)dt)}{\exp(-\int_0^{t_j} h(t,x)dt)} \\ &= 1 - \exp(-\int_{t_j}^{t_{j+1}} h(t,x)dt). \end{aligned}$$

Let then $h_{j0}^* = h_j(x = 0)$ be the baseline at $x = 0$, i.e. the intercept. The proportional hazards assumption now states that the odds of $h_j^*(x)$ are proportional to the odds of h_{j0}^* for all j, which implies for all $j = 1, ..., J$

$$\frac{1 - h_j^*(x)}{h_j^*(x)} = \frac{1 - h_{j0}^*}{h_{j0}^*} g(x).$$

Now, the function $g(x)$ can be replaced by $\exp(x^T \beta)$ and one gets

$$\frac{1 - h_j^*(x)}{h_j^*(x)} = \exp(\log \frac{1 - h_{j0}^*}{h_{j0}^*} + x^T \beta).$$

Therefore, survival can be interpreted as the sequence of binary events. If Δt approaches zero, this generalizes to the proportional hazards model:

$$\frac{h(t,x)}{h_0(t)} = \exp(x^T\beta) \Leftrightarrow h(t,x) = h_0(t)\exp(x^T\beta).$$

B Implementation Details

This section of the appendix contains a more detailed discussion of the concrete implementation of various methods. Packages as well as specific model settings will be described in order to provide a reusable framework for future extensions and further research. All analyses were done in the statistical programming language `R`, which offers numerous different packages for all kinds of survival analysis and is the standard tool for statistics in the biological sector.

Cox Regression

As described in section 4.2.1, it was necessary to reduce the number of dimensions drastically before performing cox regression. The data frame is called `data.STS` and the vector `cox_uni$var` contains the names of variables that were chosen from the univariate regression models and should be used to build a multivariate prediction model.

```
library(survival)

# generate formula of chosen variables
formula_cox <- formula(paste("Surv(OS,statusSurvival)~",
paste(as.character(cox_uni$var), collapse="+")))
# model
mod.cox <- coxph(formula_cox, data=data.STS, x=TRUE)
# results
summary(mod.cox)
```

Penalized Cox Regression

In order to perform a penalized version of the classic cox proportional hazards model, the implementation of two different packages was used. Both of them provide the option to perform cross validation to determine suitable values for the penalization parameters,

which is demonstrated below. L1 regularization was used to shrink parameter values to zero and therefore provide built-in variable selection. This time, all variables were included in the model formula as `all_var`.

```
formula <- formula(paste("Surv(OS, statusSurvival == 1) ~ ",
paste(as.character(all_var), collapse=" + ")))

## Version 1: package penalized ---
library(penalized)

# model with 10 fold cross validation
mod.penalized.cv <- optL1(formula, data=data.STS, lambda2=0, fold=10)

# inspect steps of lambda
profL1(formula, data=data.STS, fold=cox.penalized.cv.1$fold, steps = 50)

## Version 2: glmnet package ---
library(glmnet)

# suitable data format
x <- model.matrix( ~ .,  data.STS[,8:1189]) # clinical and radiomics predictors
y <- data.STS[,c(2,5)] # survival times and status

# model
mod.glmnet.cv <- cv.glmnet(x, Surv(data.STS$OS,
data.STS$statusSurvival == 1), family="cox",
type.measure="deviance", nfolds=10)
```

Tree based methods

Three different version of survival trees or ensembles of such were used in this work. Again, the trees choose from all available variables. The bagging model of the package `ipred` is based on the implementation of single trees in `rpart` and therefore offers the same configurations. The random survival forest presents more options for specific survival settings.

```
formula <- formula(paste("Surv(OS, statusSurvival == 1) ~ ",
paste(as.character(all_var), collapse=" + ")))

## Version 1: rpart ---
libarary(rpart)

# model; parameter "cp" for adjusting pruning
mod.rpart <- rpart(formula, data=data.STS, method='exp')
mod.rpart.02 <- rpart(formula, data=data.STS, method='exp', cp=0.2)

## Version 2: ipred (bagging of rpart) ---
library(ipred)

# model; 100 bootstrap iterations
mod.ipred <- bagging(formula=formula, data=data.STS, coob=TRUE,
nbagg=100, control=rpart.control(minsplit=2,minbucket=1,cp=0, xval=0),
method="exp")

# Version 3: random survival forest ---
library(randomForestSRC)

# model; 100 bootstrap iterations
mod.rsf <- rfsrc(formula=formula, data = data.STS, ntree=100,
```

```
ensemble='oob', splitrule='logrank', block.size=1,
node.size=1, var.used='all.tree')
```

Neural Network methods

Though R offers packages for building general neural networks, the offer of survival specific approaches is quite small. In addition to a fairly new package called SurvELM providing black box models, a small and very simple feed forward neural network was implemented from scratch. The details will not be discussed here but a similar approach can be found at http://www.erikdrysdale.com/neuralnetsR/. The implemented function allows the user to specify the number of layers and hidden units for each layer, as well as common network tuning parameters.

```
# Version 1: SurvELM ---
library("devtools")
devtools::install_github("whcsu/SurvELM")
library(SurvELM)

# data format
n <- dim(data.STS)[1]
x1 <- model.matrix( ~ .,  data.STS[,6:1189])

# model
mod.survCox <- ELMCox(x1,Surv(data.STS$OS,data.STS$statusSurvival),
alpha=1, maxit=10000)

# Version 2: Feed forward neural network ---

# data fromat
```

```
x2 <- t(data.matrix(data.STS[,c(6,8:1189)]))
Surv <- Surv(time=data$OS, event=data$statusSurvival)

# call own function: one hidden layer with 10 nodes
cox.net <- survnetfit(c(10,1), x2, Surv[,2], Surv[,1],
alpha = 0.001,niter=10000,verbose=T,ll=1/2,leaky=0.1)
```

List of Figures

List of Tables

References

Bender, A., D. Rügamer, F. Scheipl, and B. Bischl (2020). A general machine learning framework for survival analysis. *arXiv preprint arXiv:2006.15442*.

Benedetti, R. (2010). Scoring rules for forecast verification. *Monthly Weather Review 138*(1), 203–211.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Borucka, J. et al. (2014). Extensions of cox model for non-proportional hazards purpose. *Ekonometria* (45), 85–101.

Bou-Hamad, I., D. Larocque, H. Ben-Ameur, et al. (2011). A review of survival trees. *Statistics surveys 5*, 44–71.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen (1984). *Classification and regression trees*. CRC press.

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review 78*(1), 1–3.

Chen, H.-C., R. L. Kodell, K. F. Cheng, and J. J. Chen (2012). Assessment of performance of survival prediction models for cancer prognosis. *BMC medical research methodology 12*(1), 102.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological) 34*(2), 187–202.

Dai, B. and P. Breheny (2019). Cross validation approaches for penalized cox regression. *arXiv preprint arXiv:1905.10432*.

Davis, R. B. and J. R. Anderson (1989). Exponential survival trees. *Statistics in Medicine 8*(8), 947–961.

Drysdale, E. (2018). Building a survival-neuralnet from scratch in base r. *http://www.erikdrysdale.com/neuralnetsR/*.

Efron, B. and R. J. Tibshirani (1994). *An introduction to the bootstrap*. CRC press.

Faraggi, D. and R. Simon (1995). A neural network model for survival data. *Statistics in medicine 14*(1), 73–82.

Friedman, J., T. Hastie, H. Höfling, R. Tibshirani, et al. (2007). Pathwise coordinate optimization. *The annals of applied statistics 1*(2), 302–332.

Friedman, J., T. Hastie, and R. Tibshirani (2001). *The elements of statistical learning*, Volume 1. Springer series in statistics New York.

Friedman, J., T. Hastie, and R. Tibshirani (2009). glmnet: Lasso and elastic-net regularized generalized linear models. *R package version 1*(4).

Goel, M. K., P. Khanna, and J. Kishore (2010). Understanding survival analysis: Kaplan-meier estimate. *International journal of Ayurveda research 1*(4), 274.

Goeman, J., R. Meijer, N. Chaturvedi, and M. Lueder (2012). penalized: L1 (lasso and fused lasso) and l2 (ridge) penalized estimation in glms and in the cox model. *URL http://cran. r-project. org/web/packages/penalized/index. html*.

Goeman, J. J. (2010). L1 penalized estimation in the cox proportional hazards model. *Biometrical journal 52*(1), 70–84.

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

Gui, J. and H. Li (2005). Penalized cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data. *Bioinformatics*, 3001–3008.

Harrell, F. E., R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati (1982). Evaluating the yield of medical tests. *Jama 247*(18), 2543–2546.

Hastie, T. and R. Tibshirani (1990). *Generalized Additive Models*. Chapman and Hall.

Henderson, R. (1995). Problems and prediction in survival-data analysis. *Statistics in medicine 14*(2), 161–184.

Hoerl, A. E. and R. W. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*(1), 55–67.

Huang, G.-B., Q.-Y. Zhu, and C.-K. Siew (2006). Extreme learning machine: theory and applications. *Neurocomputing 70*(1-3), 489–501.

Ishwaran, H., U. B. Kogalur, E. H. Blackstone, M. S. Lauer, et al. (2008). Random survival forests. *The annals of applied statistics 2*(3), 841–860.

Kaplan, E. L. and P. Meier (1958). Nonparametric estimation from incomplete observations. *Journal of the American statistical association 53*(282), 457–481.

Katzman, J. L., U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger (2018). Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology 18*(1), 24.

Kleinbaum, D. G. and M. Klein (2010). *Survival analysis*. Springer.

Lambin, P., R. T. Leijenaar, T. M. Deist, J. Peerlings, E. E. De Jong, J. Van Timmeren, S. Sanduleanu, R. T. Larue, A. J. Even, A. Jochems, et al. (2017). Radiomics: the bridge between medical imaging and personalized medicine. *Nature reviews Clinical oncology 14*(12), 749–762.

Li, H. and Y. Luan (2003). Kernel cox regression models for linking gene exxpression profiles to censored survival data. *Pacific Symposium on Biocomputing 8*, 65–76.

Peeken, J. C., M. Bernhofer, M. B. Spraker, D. Pfeiffer, M. Devecka, A. Thamer, M. A. Shouman, A. Ott, F. Nüsslin, N. A. Mayr, et al. (2019). Ct-based radiomic features predict tumor grading and have prognostic value in patients with soft tissue sarcomas treated with neoadjuvant radiation therapy. *Radiotherapy and Oncology 135*, 187–196.

Robins, J. M. and A. Rotnitzky (1992). Recovery of information and adjustment for dependent censoring using surrogate markers. In *AIDS epidemiology*, pp. 297–331. Springer.

Schemper, M. and R. Henderson (2000). Predictive accuracy and explained variation in cox regression. *Biometrics 56*(1), 249–255.

Shimokawa, A., Y. Kawasaki, and E. Miyaoka (2015). Comparison of splitting methods on survival tree. *The International Journal of Biostatistics 11*(1), 175 – 188.

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

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*(1), 267–288.

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

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

Utkin, L. V., A. V. Konstantinov, V. S. Chukanov, M. V. Kots, M. A. Ryabinin, and A. A. Meldo (2019). A weighted random survival forest. *Knowledge-Based Systems 177*, 136–144.

Wang, H., J. Wang, and L. Zhou (2017, 09). A survival ensemble of extreme learning machine. *Applied Intelligence*.

Wang, H. and L. Zhou (2018). Survelm: an r package for high dimensional survival analysis with extreme learning machine. *Knowledge-Based Systems 160*, 28–33.

Wei, L.-J. (1992). The accelerated failure time model: a useful alternative to the cox regression model in survival analysis. *Statistics in medicine 11*(14-15), 1871–1879.