Hochschule Darmstadt

Fachbereich Mathematik
und Naturwissenschaften
& Informatik

Machine learning applied to right-censored survival data

Abschlussarbeit zur Erlangung des akademischen Grades
Master of Science (M. Sc.)
im Studiengang Data Science

vorgelegt von
Lukas Klein

Referentin: Prof. Dr. Antje Jahn-Eimermacher
Korreferent: Prof. Dr. Gunter Grieser

Ausgabedatum: 08.06.2022
Abgabedatum: 04.12.2022

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Chemnitz, den 3.12.2022 Lukas Klein

Abstract (deutsch)

Seit 2021 ist die deutsche Registerstelle für Organtransplantationen nach dem Transplantationsgesetz im regulären Betrieb (Nagel, 2022). Sie stellt Daten über potenzielle Empfänger, Spender, Organe, durchgeführte Transplantationen und Nachuntersuchungsergebnisse für die Qualitätssicherung und die Forschung bereit. Die erfassten Daten dieser Register eignen sich für Analysen mit Methoden der Ereigniszeitanalyse für Ereignisse wie Organversagen oder den Tod des Patienten. Trainierte Ereigniszeitanalyse-Modelle erlauben Vorhersagen und die Identifikation von relevanten Einflussgrößen auf den Erfolg von Transplantationen. Das erste Ziel dieser Arbeit war die Vorverarbeitung der Daten des deutschen Registerexports. Aus diesen wurden Zielvariablen für eine Ereigniszeitanalyse basierend auf Nierentransplantationen erstellt. Auf diesen Daten wurde dann maschinelles Lernen für die Ereigniszeitanalyse angewendet und die daraus resultierenden Modelle evaluiert als ein „Proof-of-Concept". Für Nierentransplantationen konnte ein Datensatz erstellt werden, der 20325 erste Nierentransplantationen mit 261 Prädiktoren beschreibt. Ungefähr 15000 dieser konnten für die Ereigniszeit Modellierung benutzt werden. Die besten Ergebnisse wurden von Gradient Boosting Modellen und Survival Random Forest Modellen erreicht. Für die Vorhersage von Organversagen oder dem Patiententod nach 10 Jahren erreichte das beste Gradient Boosting Modell einen C-Index von 0.74 und das beste Random Forest Survival Modell 0.73.

Abstract (english)

Since 2021, the national German registry for organ transplantation in accordance with the German transplantation law has been fully operational (Nagel, 2022). It provides data on potential recipients, donors, organs, performed transplantations and follow-up results for quality assurance and research. The data-exports can be used for survival time analysis with events like graft failure or patient death. Trained models can be used to predict graft and patient survival and to identify factors for treatment success. The first goal of this thesis is the preprocessing of the exported data from the German registry. For kidney transplantation, survival analysis targets are created. Survival Analysis machine learning is being applied to these targets and resulting models evaluated as a proof of concept to use the data for research. For the kidney transplantations, a dataset with 20325 first-time transplantations with 261 features was created. Approximately 15000 of these could be used for survival analysis modelling. The best results were achieved by gradient boosting models and survival random forests across all targets. For the prediction of graft failure or patient death over a time span of 10 years, the best gradient boosting model achieved a C-index of 0.74 and the best survival random forest 0.73.

In dieser Arbeit wurden die an das nationale Transplantationsregister übermittelten transplantationsmedizinischen Daten der Jahre 2006 bis 2016 (sogenannte Altdaten) und Daten der Jahre 2017 bis zu dem Exportzeitpunkt 11.02.2022 ausgewertet. Die Daten wurden in anonymisierter Form von der Transplantationsregisterstelle zu Forschungszwecken gemäß § 15g Absatz 1 TPG zur Verfügung gestellt.

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

1 Introduction

According to the “Bundeszentrale für gesundheitliche Aufklärung” (BZGA) in the year 2021, 1992 kidney transplantations were performed in Germany. Over the same time period, 2653 new patients were registered on the waiting list for a donor organ, which had a length of 6593 entries at the end of that year (Vieth, 2022). With such an imbalance between supply and demand, it is easy to see how important optimal matching between donor and recipient is. Factors for a successful transplantation can be identified with systematic data analysis. There is an increasing push to make data available in the healthcare sector. On the first of November 2016, the TxRegG law was passed in Germany (Nagel, 2022) with the goal of establishing a central registry for transplantation data. Data from all important institutions that are part of the organ transplant processes in Germany are combined (IQTIG, Eurotransplant and DSO[1]). The United States and the United Kingdom provide similar national services for researchers.

The increasing prevalence of digital data registries in the healthcare sectors gives rise to a need for computer aided analysis of the patient data and machine learning methods might be able to leverage increasing data dimension and recognize patterns on which physicians will be able to make decisions in the future. Potential applications are waitlist prioritization, organ allocation, identification of potential donors, prediction of overall survival, potential complications, and pharmacokinetic analysis (Gotlieb et al., 2022). This thesis focuses on the application for the prediction of overall survival and prediction of graft failure complications. Most clinical registries have similar data patterns: Information from patients is collected only at regular follow-up visits, patients are only observed up to a limited time and some patients are lost to follow-up. This also applies to the German central registry for transplantation data. These data patterns are complicating the analysis of patients’ risks, such as the risk of graft failure, organ rejection, or death of a patient over time after the transplantation.

Graft failure and patient death are events for which monitoring starts after the transplantation. The goal of modelling is to predict either the time until such an event may occur based on a set of predicting covariates or the risk patients are at for an event compared to other patients. Data used for this modeling task are called time-to-event data or survival data.

[1] IQTIG: https://iqtig.org/ DSO: https://dso.de/ Eurotransplant: https://www.eurotransplant.org/

1.1 Survival Data

The data collection only ends when either the outcome of interest - like graft failure - is observed, the patient is lost to follow-up, or the study has ended. This means that for some patients only the information that an event has not occurred for a certain amount of time is available. The remaining time to the event is censored in the data, leading to the term "right censoring". The field of survival analysis provides methods and models for censored time-to-event data.

1.2 Censoring types

Other possible types of censoring besides right censoring are left censoring and interval censoring. Left censoring can be observed if an event occurred before the subjects' events were monitored. The start time of monitoring is sometimes referred to as enter time. When both right and left censoring occurs, it is called double censoring (Turkson et al., 2021). Interval censoring is the most prevalent type of censoring when looking at follow-up data. Subjects are observed only at certain time points during visits. When an event for a subject is registered in this case, the actual event time is only known to be between the current follow-up visit and the last.

Censoring can be further classified based on the study design. Studies where each subject is monitored for a predetermined time are considered to have type I censoring. Type II happens when subjects are observed until a certain fraction of the subjects experiences events. These types of systematic censoring are different from random censoring, where censoring happens as a random event. It can also occur in Type I and II studies. Depending on the type of censoring, different quantities are fixed and others are random variables. The number of subjects is always assumed to be fixed (Moore, 2016; Turkson et al., 2021).

The following table gives a short overview of observable types of censoring and their affected random and fixed variables (Moore, 2016; Turkson et al., 2021).

Table 1 Censoring Source Types

Type of censoring	*Random Variables*	*Fixed Variables*
Type I	Number of uncensored events	Maximum observation time
Type II	Maximum observation time	Number of uncensored events
Random	Subjects enter and exit times	Total observation time

In addition to censoring, survival data can also suffer from truncation. Left truncation happens when there is a delay between a subject being at risk of a loss to follow-up and the start of monitoring. If a subject experiences an event before they can be monitored, they will not appear in the data. In contrast, with censoring a subject is known to have been at risk, but it has been too early to register, while with truncation the truncated subjects do not even appear in the data (Moore, 2016). Truncation needs to be detected in the study design, as its effect on the data might not be visible. It is important to note that sometimes censoring is also called truncation in some publications.

Most models only support right censoring, as it is the most common case and even interval censored data is often treated as right censored data. Therefore, correction methods might be needed to avoid bias.

1.3 Modeling

The time to an event can be described as a random variable T^* and, in the case of right censoring, the time to the censoring as the random variable C. Observable values are only the time until whichever event comes first $T = \min(T^*, U\ )$, and a boolean indicator whether an event was observed $\delta = I(T^* \leq U)$. A right censored observed sample consists of ordered values from the time ordered pairs (T_i, δ_i) with $i \in \{1, \dots, n\}$ and n being the number of subjects observed. T_i and δ_i are respectively assumed to be independent and identically distributed (i.i.d). The observed subjects are ordered by their registered time $t_1 \leq t_2 \leq \cdots \leq t_n$. If there are predictors $X = \left(X_1, X_2, \dots, X_p\right)^T$ available for the time to event T^*, most models assume the time to censoring U and time to event to be independent conditionally on predictors and the influence of the predictors on T^* to be constant over time.

Commonly, two functions are used to describe the time to event distribution. These are the survival function and the hazard function. The survival function reports the probability that a subject has not experienced an event up to a given time point t $S(t) = P(T^* > t)$ with $0 < t < \infty$. It is the complement to the cumulative distribution function of T^*: $F(t) = P(T^* \leq t) = 1 - S(t)$. The corresponding density distribution describes the probability of an event occurring at an infinitesimal small time window after t:

$$f(t) = \frac{d}{dt}F(t) = -\frac{d}{dt}S(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T^* < t + \Delta t)}{\Delta t}.$$

While the density can be used to describe the rate of events overall, often it is desired for applications to describe the rate of events for subjects which have not yet experienced an event up to a certain point. This is called the hazard function and it can be derived from the survival function (Moore, 2016):

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T^* < t + \Delta t |\, T^* > t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{P(t \leq T^* < t + \Delta t\,, T^* > t)}{\Delta t \cdot P(T^* > t)}$$

$$= \lim_{\Delta t \to 0} \frac{P(t \leq T^* < t + \Delta t\,)}{\Delta t} \cdot \frac{1}{S(t)} = \frac{f(t)}{S(t)} = \frac{-S'(t)}{S(t)}.$$

Commonly reported is the cumulative hazard function $H(t) = \int_0^t h(x)dx$. The cumulative hazard can be transformed into the survival function and vice versa, provided they are assumed to be continuous:

$$H(t) = \int_0^t h(x)dx = \int_0^t \frac{-S'(x)}{S(x)} dx = -\int_0^t \frac{S'(x)}{S(x)} dx = -\ln S(t) + \ln S(0)$$

$$= -\ln S(t) + \ln 1 = -\ln S(t).$$

2 Methods

To correctly assess model performance, inspect for potential problems and extract insights from the data, it is important to consider the structure of a model and its training algorithm.

2.1 Statistical Models

The following model types are very common in survival analysis and have stronger connections to parametric and non-parametric methods present in the statistical learning field.

2.1.1 Parametric Models

When the hazard is constant $h_{exp}(t) \equiv \lambda$, the exponential distribution can be used to describe the time to event $T^* \sim Exp(\lambda)$ $(\text{Moore}, 2016;\ \text{Sestelo}, 2017)$:

$$H_{exp}(t) = \lambda t \quad S_{exp}(t) = e^{-\lambda t} \quad E(T^*) = \frac{1}{\lambda} \quad Var(T^*) = \frac{1}{\lambda^2}$$

Other parametric distributions used are for example the Gamma and Weibull distributions, of which the exponential distribution is a special case, and the logarithmic versions of the normal and logistic distributions (Kartsonaki, 2016).

To estimate the parameter set γ for a distribution, ignoring censoring in the data, the maximum likelihood estimator based on the density can be used:

$$L(\gamma; t_1^*, \dots, t_n^*) = \prod_{i=1}^{n} f(t_i^*, \gamma).$$

When right censoring occurs, it is only known that the subject experienced no event up until the given time point. Therefore, for these subjects, the survival function is used instead of the density function (Moore, 2016):

$$L(\gamma; t_1, \dots, t_n; \delta_1, \dots, \delta_n) = \prod_{i=1}^{n} f(t_i, \gamma)^{\delta_i} \cdot S(t_i, \gamma)^{1-\delta_i}$$

$$= \prod_{i=1}^{n} (h(t_i, \gamma) \cdot S(t_i, \gamma))^{\delta_i} \cdot S(t_i, \gamma)^{1-\delta_i} = \prod_{i=1}^{n} h(t_i, \gamma)^{\delta_i} \cdot S(t_i, \gamma).$$

To incorporate multivariate predictors, a common model is the linear accelerated failure time model (AFT):

$$\ln T_i^* = x_i \beta + \epsilon_i.$$

In this model x_i is the vector of predictors, β is a vector of coefficients and ϵ is an i.i.d. residual random variable. As can be seen, this results in a linear model. The influence on the time to event from the residual term and the linear predictor is multiplicative on the original time scale; this gives the model its name. When the expected value of the residual term is assumed to be $E(\epsilon|X) = 0$, the least squares estimator can be used, minimizing the following residual sum of squares (RSS) (Orbe et al., 2002):

$$RSS^* = \sum_{i=1}^{n} (\ln t_i^* - x_i\beta)^2$$

To adjust the estimator for censoring, weights as determined by the Kaplan-Meier estimator (see next chapter for W_{KM_i}) are used:

$$RSS = \sum_{i=1}^{n} W_{KM_i} (\ln t_i - x_i\beta)^2$$

With those weights, the least squares estimator can be written in a matrix-based notation with T being the observed times as a vector, X being the predictor matrix and W being a diagonal matrix with the Kaplan-Meier weights (Orbe et al., 2002):

$$\hat{\beta} = (X'W'X)^{-1}X'W \ln T.$$

With $\epsilon_i \sim_{iid} N(0, \sigma^2)$, the time to event T_i^* follows a lognormal distribution (Orbe et al., 2002). If it is not possible to construct the least squares estimator due to the assumption $E(\epsilon|X) \neq 0$, the maximum likelihood with iterative fitting algorithms using the survival and density functions described before can be used.

In some linear AFT models, the residual term is modeled by $\epsilon_i = \sigma W_i$, with W_i also being an i.i.d. random variable, a scaling factor $\sigma > 1$, and a constant offset μ:

$$\ln T_i^* = \mu + x_i\beta + \sigma W_i.$$

Common distributions for W_i are members of the gamma distribution family. An AFT model models the survival function based on the distribution of the underlying random variable (Faruk, 2018):

$$S_i(t) = P(T^* > t) = P(\ln T_i^* > \ln t) = P(\mu + x_i\beta + \sigma W_i > \ln t)$$
$$= P\left(W_i > \frac{\ln t - \mu - x_i\beta}{\sigma}\right).$$

This calculation of the prediction of the survival function is therefore based on the quantile function of the random variable W_i, producing survival functions at fixed survival rates but variable times.

2.1.2 Non-Parametric Models

Without censoring, a simple naive survival function non-parametric estimator can be constructed as follows:

$$\widehat{S_{unc}}(t) = \frac{1}{n}\sum_{i=1}^{n} I(t_i^* > t)\,.$$

Because survival times of the subjects are assumed to be independent from each other and because the resulting survival function has discrete steps it is possible to also use the previous survival rate and the discrete hazard $\hat{\lambda}(t_i)$ to construct the value at the next survival time observed with $t_0 = 0$ and $S(0) = 1$ (Clark et al., 2003; Sestelo, 2017):

$$\widehat{S_{unc}}(t_i) = \widehat{S_{unc}}(t_{i-1}^*) \cdot \left(1 - \widehat{\lambda_{unc}}(t_i^*)\right)$$

$$\widehat{\lambda_{unc}}(t) = \frac{\sum_{i=1}^{n} I(t_i^* = t)}{\sum_{i=1}^{n} I(t_i^* \geq t)}.$$

$$\widehat{S_{unc}}(t) = \prod_{i:t_i \leq t} \left(1 - \widehat{\lambda_{unc}}(t_i^*)\right)$$

The discrete hazard is the number of subjects which experienced an event at the specified time divided by the number of subjects who have not experienced one yet. For the discrete version, the equality is included for the survival rate. Censoring can be accounted for by correcting the number of subjects at risk, which is the divisor for the discrete hazard by only counting subjects which have not been censored or have not registered with an event (Clark et al., 2003; Sestelo, 2017):

$$\hat{\lambda}(t) = \frac{\sum_{i=1}^{n} I(t_i = t, \delta_i = 1)}{\sum_{i=1}^{n} I(t_i \geq t)}.$$

This is the Kaplan-Meier estimator, also called the product-limit. The Kaplan-Meier estimator can be also formulated as weights for the naive survival estimator ignoring censoring (Orbe et al., 2002; Sestelo, 2017):

$$\hat{S}(t) = 1 - \sum_{i=1}^{n} W_{KM_i} I(t_i \leq t)$$

$$W_{KM_i} = \widehat{F_{unc}}\left(\ln T_{(i)}\right) - \widehat{F_{unc}}\left(\ln T_{(i-1)}\right) = \frac{\delta_{(i)}}{n - i + 1} \prod_{j=1}^{i-1} \left(\frac{n - j}{n - j + 1}\right)^{\delta_{(j)}}.$$

Another important non-parametric estimator is the Nelson-Aalen estimator (Moore, 2016) for the cumulative hazard function:

$$\widehat{H}(t) = \sum_{i:t_i \leq t} \hat{\lambda}(t_i) = \sum_{i:t_i \leq t} \frac{\sum_{i=1}^{n} I(t_i = t, \delta_i = 1)}{\sum_{i=1}^{n} I(t_i \geq t)}.$$

Both the Kaplan-Meier and the Nelson-Aalen estimator can also be employed for the cumulative hazard and the survival function using the previous established negative natural exponential relationship between them. The survival function estimator based on the Nelson-Aalen estimator is called the Breslow estimator (SAS Institute Inc, 2019). Fleming and Harrington noted that while the Kaplan Meier estimator results in the same estimator with tied times and those ties resolved by moving them infinitesimally to the left, the Breslow estimator gives different results (Fleming & Harrington, 1984; SAS Institute Inc, 2019). Their proposed estimator correcting this behavior is used by many software packages instead of the Nelson-Aalen estimator:

$$\widehat{H}(t) = \sum_{i:t_i \le t} \sum_{j=0}^{d_i - 1} \frac{1}{\left(\sum_{i=1}^{n} I(t_i \ge t)\right) - j} \qquad d_i = \sum_{k=1}^{n} I(t_k = t_i, \delta_k = 1)$$

Different methods are used to estimate standard errors and therefore confidence intervals for these estimators. Most are based on the Greenwood formula which uses the delta method for variance estimation with different transformations. The delta method is based on the Taylor series approximation and uses the assumption, that a transformed random variable with a constant c close to the mean of the random variable, expected value and variance can be approximated (Hosmer et al., 2008):

$$E\big(f(X)\big) \approx f(c) + f'(c) \cdot (E(X) - c) \qquad Var\big(f(X)\big) \approx \big(f'(c)\big)^2 \cdot Var(X)$$

Usually, the logarithmic or double logarithmic transformation of the estimator with the random binomial variable being $d_i \sim B(\sum_{i=1}^{n} I(t_i \ge t), \hat{\lambda}(t_i))$. With the log-log transformation, the confidence intervals will stay between 0 and 1. The variance of the Kaplan-Meier estimator with an assumed normal distribution would be (Moore, 2016):

$$Var\left(\ln\left(-\ln \hat{S}(t)\right)\right) \approx \left(\log \hat{S}(t)\right)^{-2} \sum_{i:t_i \le t} \frac{d_i}{\left(\sum_{i=1}^{n} I(t_i \ge t)\right) \cdot \left(\left(\sum_{i=1}^{n} I(t_i \ge t)\right) \cdot d_i\right)}$$

These non-parametric estimators can be used for discrete predictors, with each possible covariate combination having its own survival estimator.

2.1.3 Semi-Parametric Models

One of the core assumptions for survival analysis estimators is based on a Lehmann alternative hypothesis (Davies, 1971) for the cumulative density functions F, G of two random variables:

$$H_0: F = G \qquad H_1: G = 1 - (1 - F)^{\frac{1}{\eta}} \qquad \eta > 1.$$

Adopted to two survival functions with an inverted $\psi = \frac{1}{\eta}$, it is the proportional hazard assumption (Moore, 2016):

$$S(t) = \big(S_0(t)\big)^{\psi} \Leftrightarrow h_1(t) = \psi \cdot h_0(t).$$

The hazard function $h_0(t)$ is called the baseline hazard. With $\psi(x) = e^{x\beta}$, x being a vector of predictors and β being a vector of coefficients, it becomes possible to use covariates for the estimation of survival functions. The proportional hazard based model assumes a linear relationship for the logarithmic ratio of the hazard and the baseline hazard, as shown by the following equation:

$$\ln \frac{h_1(t)}{h_0(t)} = x\beta.$$

Some AFT models, for example the ones based on the Weibull and exponential distribution, also have a proportional hazard interpretation (Orbe et al., 2002). In 1972 Cox introduced the partial likelihood to estimate the coefficients in β even without knowing the baseline hazard function. Assuming the indices of subjects are sorted by their failure time, ignoring ties for now and assigning each event experiencing subject their own hazard function, the probability that the subject was the one observed in the data is (Cox, 1972; Moore, 2016):

$$p_k = \frac{P(T_k^* = t_k)}{\sum_{i:t_i \geq t_k} P(T_i^* = t_k)} = \frac{h_k(t_k)}{\sum_{i:\, t_i \geq t_k} h_i(t_k)} \quad \forall k: \delta_k = 1.$$

The probabilities $P(T_i^* = t_i^*)$ could not be known as they might be censored, but under the proportional hazard assumption, only the factors ψ remain and the time points are no longer relevant:

$$p_k = \frac{h_k(t_k)}{\sum_{i:\, t_i \geq t_k} h_i(t_k)} = \frac{h_0(t_k) \cdot \psi(x_k)}{\sum_{i:\, t_i \geq t_k} h_0(t_k) \cdot \psi(x_i)} = \frac{\psi(x_k)}{\sum_{i:\, t_i \geq t_k} \cdot \psi(x_i)} \quad \forall k: \delta_k = 1.$$

From these probabilities the partial likelihood can be constructed and maximized to find the coefficients:

$$L(\beta) = \prod_{k:\delta_k = 1} \frac{\psi(x_k)}{\sum_{i:\, t_i \geq t_k} \psi(x_i)}$$

After the coefficients have been determined, the baseline hazard function needs to be estimated. A slightly modified version of the Nelson-Aalen estimator with the divisor not using the number of subjects at risk but the sum of the ψ, can be used to determine it. This calculation produces a baseline where all covariates would be zero:

$$h_0(t_i) = \frac{\sum_{j=1}^{n} I\big(t_j = t_i, \delta_j = 1\big)}{\sum_{j:\, t_j \geq t_i} \psi(x_j)}$$

In the likelihood function, ties are resolved by summing the likelihoods for all possible orderings of tied times. For large datasets with a lot of ties, the combinatoric evaluations can become a challenge. Commonly, approximate methods for the partial likelihood are used, which evaluate the likelihood at each time point an event was observed. They all share the sum of the $\psi(x)$ values for each subject with the event at a time point in the numerator, but have different divisors (Hertz-Picciotto & Rockhill, 1997):

$$L_{Breslow}(\beta) = \prod_{t_k \in \{t_i : \delta_i = 1\}} \frac{\sum_{j: t_j = t_k \wedge \delta_j = 1} \psi(x_j)}{\left(\sum_{i:\, t_i \geq t_k} \psi(x_i)\right)^{d_k}} \qquad d_k = \sum_{j: t_j = t_k \wedge \delta_j = 1} 1$$

$$L_{Efron}(\beta) = \prod_{t_k \in \{t_i : \delta_i = 1\}} \frac{\sum_{j: t_j = t_k \wedge \delta_j = 1} \psi(x_j)}{\prod_{g=1}^{d_k} \left(\sum_{i:\, t_i \geq t_k} \psi(x_i) - \frac{g-1}{d_k} \sum_{j: t_j = t_k \wedge \delta_j = 1} \psi(x_j)\right)}$$

$$L_{Kalbfleisch,Prentice}(\beta) = \prod_{t_k \in \{t_i : \delta_i = 1\}} \frac{\sum_{j: t_j = t_k \wedge \delta_j = 1} \psi(x_j)}{\sum_{q \in Q_k} \psi(s_q)} \qquad s_q = \sum_{j \in q} x_j.$$

The Kalbfleisch-Prentice approximation uses all possible subsets Q_k, each with a size d_k, which where the number of patiens at risk for an event. Cox himself also proposed an estimation based on the similarity of his method to a logistic model. It has been shown that the Efron method has lower bias, especially for smaller samples compared to the Breslow method (Hertz-Picciotto & Rockhill, 1997). In the R package survival (Therneau, 2022), the Efron version is used by default, while the scikit-survival Python package (Pölsterl, 2020) uses the Breslow estimator.

2.1.4 Machine Learning Models

It is difficult to draw the line between methods of regression and classification employed in applied statistics and those used in machine learning. In some circles statistical and machine learning are considered synonyms. All statistical learning algorithms like the ones previously described with either extensions or non-linear estimators are considered machine learning algorithms for this thesis.

2.1.4.1 AFT with Shrinkage

Regression methods like ridge, regression trees or least absolute shrinkage and selection operator (LASSO) regression can be adapted using assumptions of an AFT model with a residual distribution based on the assumption that the residual term has the expected value of zero:

$$\ln T_i^* = x_i \beta + \epsilon_i \quad E(\epsilon_i) = 0$$

Censoring is considered by using the inverse probability of censoring weighting (IPCW). A Kaplan-Meier type estimator is used to estimate the probability of the subject being censored at the time it experienced the event time or later. It is important to note that usually the estimator is used to estimate the probability of an event, not censoring. This can be done by treating the number of censored subjects as the number of events and treating each event as a dropout due to censoring. The inverse of the censoring probability is used to give weights to the subjects. Censored subjects receive a weight of zero. The class “IPCRidge” in scikit-survival inherits from the “Ridge” class in scikit-learn (Pedregosa et al., 2011) and adds the weights and the logarithmic transformation while using an intercept for the model. These are minimal changes easily adaptable to other regression algorithms with sample weighting available (Vock et al., 2016).

For this ridge model, the loss function with p being the number of predictors is:

$$RSS_{Ridge} = \sum_{i=1}^{n} W_{IPCW_i}(\ln t_i - \beta_0 + x_i\beta)^2 + \alpha \sum_{j=1}^{p} \beta_j^2$$

In addition to the loss by errors in the prediction, large coefficients lead to an increase in the loss. Ridge uses the L2 norm of the coefficients. Because the coefficients are summed after squaring it is important that the predictors are standardized so that the coefficients are of similar magnitude if they have a similar influence on the prediction. With the increasing α, the effective number of degrees of freedom of the model decreases. This can help avoid overfitting and multicollinearity. LASSO regression is a similar shrinkage technique. It uses the absolute values of the coefficients and is therefore able to set coefficients to zero (Hastie et al., 2009), hence eliminating them from the model. This is the L1 norm of the coefficients and has a computational advantage over ridge. The tuning parameter α needs to be set for the model in advance either with hyperparameter optimization or by using an algorithm to determine possible α values and selecting from them (Hastie et al., 2009).

2.1.4.2 Cox Regression with Shrinkage

Commonly used is the combination of the ridge and LASSO regression to the elastic net, which uses a weighted average of the L1 and L2 norm as an additional loss:

$$RSS_{ElasticNet,Linear} = \sum_{i=1}^{n} (y_i - \beta_0 + x_i\beta)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2$$

The tuning parameters are often combined to a mixing parameter $\alpha = \frac{\lambda_2}{\lambda_1+\lambda_2}$ (Zou & Hastie, 2005). The elastic net penalty can also be used on a Cox regression model. This is done by adding the elastic net to the partial (log)likelihood as a constraint $\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha)\sum_{j=1}^{p} \beta_j^2 \leq c$ (Simon et al., 2011):

$$LL(\beta) = \ln\left(\prod_{k:\delta_k=1} \frac{\psi(x_k)}{\sum_{i:\, t_i \geq t_k} \psi(x_i)}\right) = \sum_{k:\delta_k=1} x_k\beta - \ln\left(\sum_{i:\, t_i \geq t_k} \psi(x_i)\right)$$

The constraint can be added using the Lagrange multiplier λ:

$$\hat{\beta} = \operatorname{argmax} \frac{2}{n}\left(\sum_{k:\delta_k=1} x_k\beta - \ln\left(\sum_{i:\, t_i \geq t_k} \psi(x_i)\right)\right) - \lambda\left(\alpha\sum_{j=1}^{p}|\beta_j| + \frac{1}{2}(1-\alpha)\sum_{j=1}^{p}\beta_j^2\right)$$

Different algorithms can be used to solve this problem. The scikit-survival package (Pölsterl, 2020) in the class "CoxnetSurvival" uses a coordinate descent algorithm[2] (Simon et al., 2011). This algorithm is an iterative approach which repeatedly reweights a penalized least squares problem. It is performed at a path of values for λ. The method estimates a maximum λ which would, at a fixed α, completely shrink the coefficients β and minimum λ, which in turn is set to 5% of the maximum if $n < p$ or else 0.01%. By default, 100 λ are tested between the minimum and maximum in scikit-survival. In the original paper, the best λ is chosen by an adopted leave-one-out cross validation estimation incorporating the loglikelihood difference with and without a subject i based on coefficients estimated without it (Simon et al., 2011):

$$\widehat{CV}_i(\lambda) = L\big(\beta_{-i}(\lambda)\big) - L_{-i}(\beta_{-i}(\lambda))$$

2.1.4.3 Survival Trees

A decision tree-based algorithm partitions the feature space during the training phase. The produced model consists of hierarchical connected nodes, with each containing a univariate splitting rule until a final node is reached. In each of these final nodes, which are also called leaves, a set of training instances are grouped. If a prediction is requested, a path along the tree is followed until a leaf is reached. For classification trees, the frequencies of the target classes in this leaf are returned. Regression trees, on the other hand, return the mean and variance of the target variable in the leaf.

One way to implement decision trees for survival analysis would be to use IPCW with regression trees on the logarithmic time to event (Hothorn et al., 2005; Vock et al., 2016). For specialized survival trees, the Nelson-Aalen and/or Kaplan-Meier estimators are returned for the leaf that is reached during prediction (Ishwaran et al., 2008). The construction of decision trees is done in accordance with a recursive greedy algorithm which considers each possible split at a node and then is rerun on each resulting child until a terminating condition is met. These conditions are used for pre-pruning of a tree to reduce overfitting. Commonly, a maximum number of nodes or a minimum of the decrease in the impurity measurement are used.

[2] The model hyperparameter "alpha" is noted as λ in the paper and the mixing parameter as "l1_ratio" in the library.

Impurity measures are used to score possible splits. The split which achieves the highest weighted impurity decrease is selected. For a classification problem with k classes, commonly used measures are the entropy or Gini index (Pedregosa et al., 2011):

$$H_{Entropy} = -\sum_k p_k \log(p_k)$$

$$H_{Gini} = \sum_k p_k(1 - p_k).$$

Regression trees, on the other hand, often use the mean squared error or the mean absolute error for a median prediction. With a current partition D being split into g partitions, the information gain is calculated using the following equation:

$$IG(D) = H(D) - \sum_i^g \frac{|D_i|}{|D|} \cdot H(D_i).$$

Usually, the information gain ratio is used instead because the information gain prefers splits with more child nodes (Quinlan, 1986). It weights the information gain with the inverse entropy regarding the possible splits k of feature d:

$$GR(D) = \frac{IG(D)}{-\sum_{l \in k}\big(P(d = l) \cdot \log\big(P(d = l)\big)\big)}.$$

Assuming a binary tree with each inner node having two children, the absolute value of the log-rank test statistic can be used as the target to be maximized at each step. The log-rank statistic is derived by a comparison for a two-by-two contingency table. The comparison is between the number of subjects who experience an event d at a time t and those who are at risk and do not experience an event $n - d$ (Hothorn et al., 2005; Ishwaran et al., 2008; Ishwaran & Kogalur, 2007).

The following table visualizes the contingency table used for the log-rank statistic between two nodes together with the marginal counts.

Table 2 Contingency table for the log rank test

	Node 1	Node 2	Both
Subjects with event at t	$d_1(t)$	$d_2(t)$	$d(t)$
Subjects without event at t, but at risk	$n_1(t) - d_1(t)$	$n_2(t) - d_2(t)$	$n(t) - d(t)$
Subjects at risk	$n_1(t)$	$n_2(t)$	$n(t)$

A hypergeometric distribution can be assumed with $d_1(t)$ being the number of successes observed $n_1(t)$ being the number of potential successes and $d(t)$ being the number of draws. The total population size is $n(t)$ (Moore, 2016).

$$E\big(D_1(t)\big) = d(t)\frac{n_1(t)}{n(t)} \quad Var\big(D_1(t)\big) = d(t)\frac{n_1(t)}{n(t)} \cdot \frac{n(t) - n_1(t)}{n(t)} \cdot \frac{n_1(t) - d(t)}{n(t) - 1}$$

For each time point, the distributions are assumed to be independent. Therefore, both expected value and variance can be summed while assuming their covariance to be 0. The test statistic is the centered and standardized value of the observed count (Ishwaran & Kogalur, 2007; Moore, 2016):

$$L = \frac{\sum_t d_1(t) - \sum_t E\big(D_1(t)\big)}{\sqrt{\sum_t Var\big(D_1(t)\big)}} \sim_{\text{large sample}} N(0{,}1).$$

The log-rank statistic can also be written with estimators for the cumulative hazard like the Nelson-Aalen estimator (Fleming & Harrington, 2011). With an additional constant and weighting possibility, they are of the statistic class K (Fleming & Harrington, 2011), which is a class of statistics often used in survival analysis. The log-rank statistic uses weights of 1 as a special case. The numerator becomes:

$$G = \int_0^\infty w(t)\frac{n_1(t)n_2(t)}{n(t)}\big\{d\hat{H}_1(t) - d\hat{H}_2(t)\big\}.$$

A split on a continuous feature j requires splits to be estimated at each value s present in the training data. The log-rank statistic can also be updated based on a linear function from one split point to the next (Ishwaran et al., 2008; Ishwaran & Kogalur, 2007). This is based on the calculation of the score numerator with the help of the cumulative hazard estimate for the parent partition and the split (Ishwaran & Kogalur, 2007):

$$L * \sqrt{\sum_t Var\big(D_1(t)\big)} = \sum_t d_1(t) - \sum_t E\big(D_1(t)\big) = \sum_t d_1(t) - \sum_t I\big(x_j \le c\big)\hat{H}(t).$$

An important lemma for many survival analysis models is the “conversation of events”. The sum of the cumulative hazard estimates at all time points is equal to the number of observed events:

$$\sum_{i=1}^{n} \hat{H}(T_i) = \sum_{i=1}^{n} \delta_i.$$

This is the case for the Nelson-Aalen estimator and therefore also for the leaves of the built trees. It also is applicable to an entire survival tree summed over the leaves. This is only the case if considering all ordered samples. If only using the subjects up to a subject j, the lemma does not hold true anymore. By summing the differences and weighting by the number of subjects at risk $n(t)$, another impurity measure can be constructed. The value of the measure becomes closer to zero if the groups have very separate distributions of the event times but still show the conservation of events. Let the total number of subjects in a candidate split be n_z for an impurity measure C (Ishwaran et al., 2008; Ishwaran & Kogalur, 2007):

$$C = \frac{1}{n(t)} \sum_{z=1}^{2} \sum_{k=1}^{n_z} \left| \sum_{i=1}^{k} \widehat{H}(T_i) - \sum_{i=1}^{k} \delta_i \right|$$

To achieve a maximization goal, the inverse of this statistic plus one can be used. The log-rank statistic has been criticized to favor continuous features splits and splitting on the extreme ends of those (Ishwaran et al., 2008; Ishwaran & Kogalur, 2007). This conversation of events splitting approach avoids this bias. Still, the survival trees in the scikit-survival and ranger (Wright & Ziegler, 2017) library both use the log rank scoring method.

An alternative to pre-pruning of the tree is post-pruning. The classification and regression tree (CART) algorithm established minimal cost-complexity pruning. For a classification tree T, the classification error R for the whole tree is a weighted sum of the classification errors for a validation dataset at each leaf and a constant factor punishing the number of leaves in the tree $\tilde{T}$ (Breiman et al., 1984):

$$R_\alpha(T) = \sum_{h \in \tilde{T}} R(h) + \alpha |\tilde{T}|$$

It is possible to prove that each node t has a higher classification error than the errors of the leaves $\widetilde{T_t}$ summed for a subtree starting at node t (Breiman et al., 1984). An alpha can be found where the two errors are equal:

$$R(t) \cdot \alpha = R_\alpha(T_t)$$

Such an alpha can be found for each node. The nodes with the smallest alpha are pruned by having their subtree collapsed into a leaf. The same procedure is repeated subsequently on this new tree. This results in a strictly increasing path of alphas. Pruned trees for corresponding alphas can therefore be constructed by starting at the root tree and collapsing all nodes with an alpha below the given threshold. A smaller tree tends to show bias, which results in under-fitting, while a larger tree can overfit training data due to its high variance. The best alpha is usually selected by scoring each alpha along the path on a validation set (Breiman et al., 1984; Pölsterl, 2020).

Survival trees do not use the number of leaves $|\tilde{T}|$ in the loss. Instead the number of non-leaf nodes is used. As there is no classification error, the sum of the log-rank statistics for each node can be considered, but its behavior regarding interactions, signal noise and non-linearity make it unfit for the task. Usage of a test sample for a score, bootstrap sampling or permutation-based calculation may mitigate the effects. This pruning is not implemented in scikit-survival (LeBlanc & Crowley, 1993).

2.1.4.4 Survival Forest

A common meta model type in machine learning is the ensemble. For an ensemble, the output from different trained models is combined for a prediction. One possible ensemble model is stacking, where the outputs from several models are fit to a new model which is then trained to use these new inputs for the desired output. Another important group are bootstrapped aggregating models, also called bagging models. A bootstrap sample is generated by sampling with replacement from the obtained sample of the data. Usually, this sample has the same size n as the original sample. In this case, each bootstrap B is expected to use around 63.2% of the data (Hastie et al., 2009):

$$P(i \in B) = 1 - \lim_{n \to \infty} \left(1 - \frac{1}{n}\right)^n \approx 1 - e^{-1} \approx 0.632$$

This simulates the generation process of the sample and can be used to estimate variance of parameters of the data source through repeated estimations on the bootstraps (James, G., Witten, D., Hastie, T., Tibshirani, 2013).

On each generated bootstrap sample, a model can also be trained. This is what is called bagging and for regression, for example, the mean of all trained models is taken for a prediction. In a classification setting, the majority vote is used. The random forest model applies bagging with one additional step to decorrelate the trees. During training of the trees, each time a node is to be split, a random set of features to be tested is sampled. The features are redrawn at each split (James, G., Witten, D., Hastie, T., Tibshirani, 2013).

For a survival random forest, the predictions of each ensemble member survival tree need to be combined. For the cumulative hazard, for example, there are two ways to combine the tree outputs. Each tree produces a function estimator from the leaf, on which an observation i lands. The simple way is to just average across the N estimators (Ishwaran et al., 2008):

$$\widehat{H}_{\text{Forest}}(x_i, t) = \frac{\sum_{j=1}^{N} \widehat{H}_{\text{Tree } j}(x_i, t)}{N}$$

A second way is to use the out-of-bag estimate (OOB). This can be constructed for the instances which were passed to the algorithm during training. It only averages over the trees which do not have this sample present in their bootstrap sample. Scoring of such an estimator is an estimate of a threefold cross-validation score (Hastie et al., 2009; Ishwaran et al., 2008). By default, scikit-survival and ranger do not use this scoring method.

$$\widehat{H}_{\text{Forest, OOB}}(x_i, t) = \frac{\sum_{j=1}^{N} I(i \notin B_j) \widehat{H}_{\text{Tree } j}(x_i, t)}{\sum_{j=1}^{N} I(i \notin B_j)}$$

2.1.4.5 Gradient Boosted Models

Another way to construct an ensemble is a boosting algorithm. Instead of being able to fit estimators in parallel, a sequence of estimators is constructed. In the case of the AdaBoost (Hastie et al., 2009) algorithm, before each training process weights for the subjects are calculated based on the errors on the previously trained estimators. Each trained estimator also gets a weight assigned, which is used for the ensemble predictions. Many boosting algorithms use additive models. These do not use a weighted mean, but a weighted sum of base estimators.

Like other estimators, a boosting model aims to find an estimator which minimizes the expected loss. Instead of a single parameter set A, a set of M parameter sets for a base estimator h and M weights, also called expansion coefficients, are approximated (Friedman, 2002; Hastie et al., 2009):

$$\hat{F}(x) = \underset{F(x)}{\operatorname{argmin}} E\left(L\left(y, F_M(x)\right)\right) \Rightarrow \underset{\{\beta_1 \dots, \beta_M\} \cup \{A_1, \dots, A_M\}}{\operatorname{argmin}} E\left(L\left(y, \sum_{m=0}^{M} \beta_m h(x; A_m)\right)\right)$$

The initial estimator $F_0(x) = h(x; A_0)$ is directly fitted on the dataset. For each iteration m, the optimization goal is (Friedman, 2002; Hastie et al., 2009):

$$(\beta_m, A_m) = \underset{\beta, A}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \beta h(x_i; A)).$$

For the gradient boosting algorithm, the estimator is fit to pseudo-residuals based on the derivate of the loss function with, for example, the least squares method (Friedman, 2002):

$$\tilde{y}_{i,m} = -\left[\frac{\partial L\left(y, F(x_i)\right)}{\partial F(x_i)}\right]_{F(x) = F_{m-1}(x)}$$

$$A_m = \underset{A}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\tilde{y}_{i,m} - h(x_i; A)\right)^2$$

After the estimator for the iteration m has been determined, the estimator weight can be optimized (Friedman, 2002):

$$\beta_m = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} L\big(y_i, F_{m-1}(x_i) + \beta h(x_i; A_m)\big).$$

This use of pseudo-residuals reduces the complexity of the optimization problem (Friedman, 2002).

Gradient boosted regression trees are constructed using a regression tree with a fixed number of leaves. Each leaf contains a set of training instances inside a partition $R_{l,m}$. For each leaf l of the current tree, an offset from the current additive model up to this point is calculated, which minimizes the loss for these datapoints (Friedman, 2002):

$$\gamma_{l,m} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{m,l}} L(y_i, F_{m-1}(x_i) + \gamma).$$

If the loss is the mean squared error, this offset is the mean of the target variable values of the training instances inside the partition. The new estimator adds a fraction of these offsets to the output of the previous estimator. This is the learning rate ν. Smaller learning rates lead to slower fitting to the training data and can prevent overfitting. At each step the prediction can be made with the following equation (Friedman, 2002):

$$F_m(x) = F_{m-1}(x) + \nu \cdot \sum_{l=1}^{|\tilde{T}|} \gamma_{l,m} \cdot I(x \in R_{l.m}).$$

There are two loss functions for the gradient boosted survival training available with scikit-survival. One is based on the partial likelihood of the Cox regression (Ridgeway, 1999) and the other on AFT IPC weighted least-squared regression (Hothorn et al., 2005). The AFT IPCW gradient boosted trees model uses the previously described AFT model with regression trees while weighting the squared error according to the IPC.

The initial model for decision trees boosted with the Cox proportional hazard-based loss $F_0(x) = 0$ estimates the proportional risk for all subjects as 0. The used loss function is the negative log partial likelihood, and the pseudo residuals are calculated with the first derivative of this likelihood (Ridgeway, 1999):

$$L\big(t, \delta, F_m(x)\big) = -\left(\sum_{i:\delta_i=1} F_m(x_i) - \ln\left(\sum_{j:\, t_j \geq t_i} e^{F_m(x_j)}\right)\right)$$

$$\tilde{y}_{i,m} = \delta_i - \sum_{j:\, t_j \geq t_i} \delta_i \frac{e^{F_m(x_i)}}{\sum_{g:\, t_g \geq t_i} e^{F_m(x_g)}}$$

After a regression tree has been fitted on the $\tilde{y}_{i,m}$ values, the original paper (Ridgeway, 1999) describes the fitting of a linear Cox regression to the target values t, δ and the decision tree output as an input. The regression coefficient from this model is then used to scale the output from the regression tree multiplied with the learning rate. Scikit-survival appears to skip this step and directly uses the output from the tree. The output from the Cox loss-based model can be used to predict survival functions with a proportional hazard assumption (Ridgeway, 1999).

2.1.4.6 Survival Support Vector Machines

The proportional hazard simplifies the survival prediction problem from predicting exact survival probabilities over time to relative risks between subjects. It can be further simplified to the ranking of subjects in accordance with the relative risk. With censoring, two subjects are only comparable if both experienced an event, or one experienced an event before the other was censored. This limits the set of pairs which can be used for training with an upper bound $O(n^2)$ to (Pölsterl et al., 2015):

$$\mathcal{P} = \{(i,j) \mid t_i > t_j \wedge \delta_j = 1\}$$

Support vector machines can be adapted for ranking problems (Lee & Lin, 2014). Classification support vector machines use hyperplanes within the feature space to separate between classes. It can be shown that for a perfectly separatable binary classification problem, the best performing hyperplane is estimable by the maximal margin estimator for the hyperplane (Hastie et al., 2009). It places the hyperplane in such a way that it maximizes the distance between the instances, while separating the classes, in the feature space to it. This hyperplane can then be defined by the points which are at this maximum shortest distance from the hyperplane. They become the support vectors for the classifier (Hastie et al., 2009).

Non perfect separatable classes can be used with the soft margin classifier, which allows an error for the separation. Usually this allowed error is a hyperparameter which needs to be optimized. Instances within the margin on the wrong side of the hyperplane also become part of the support vectors. Kernel functions can be used to transform the feature space and result in nonlinear classifiers. Without external methods, support vector machines cannot be used to output probabilities for classification. Assuming $f(X) = \beta_1 X_1 + \cdots + \beta_d X_d$ is the hyperplane with no intercept for a binary classification with $Y_i \in \{-1,1\}$ as the target, a L2 support vector machine can be written as the solution for the following optimization with γ as a parameter influencing the softness of the margin (Hastie et al., 2009):

$$\underset{\beta_1,\dots,\beta_d}{\operatorname{argmin}}\left(\frac{\gamma}{2}\sum_{i=1}^{n}\max\bigl(0,1-y_i\cdot f(x_i)\bigr)^2+\frac{1}{2}\sum_{j=1}^{d}\beta_j^2\right).$$

All instances for which $1-y_i\cdot f(x_i)>0$ are support vectors. This can be transferred to evaluate ordering of pairs (Lee & Lin, 2014; Pölsterl et al., 2015) by assuming that for a pair (i,j) in the correct order the function output for the first element in the pair is higher by at least 1, $f(x_i)-1\geq f(x_j)$.

$$\underset{\beta_1,\dots,\beta_d}{\operatorname{argmin}}\left(\frac{\gamma}{2}\sum_{i,j\in\mathcal{P}}\max\left(0,1-\left(f(x_i)-f(x_j)\right)\right)^2+\frac{1}{2}\sum_{j=1}^{d}\beta_j^2\right).$$

With this full pair evaluation, the scoring becomes a problem with $O(n^{\,2})$ complexity. A possible optimization method is the Newton method. It requires the full Hessian matrix, which has a calculation iteration limit of $O(|\mathcal{P}|d^2+|\mathcal{P}|d+d)$ (Pölsterl et al., 2015). Instead, the so-called truncated Newton algorithm can be applied, which uses an iterative process to achieve an approximation of the gradient direction with a conjugate gradient. This calculation only has a complexity of $O(nd+|\mathcal{P}|+d)$ (Pölsterl et al., 2015). When calculating the partial derivatives during a Newton iteration, the complexity $O(n^2)$ is the result of the calculation of the matrix product. It can be simplified while limiting it to the support vectors, but still needs to be recalculated at every iteration:

$$A^T D_w A = A_w^T A_w \quad A\in\mathbb{R}^{p\times n} \quad D_w\in\mathbb{R}^{p\times p}$$

The matrix A is sparse and represents the pairs which are valid for training (Pölsterl et al., 2015):

$$A_{ki}=1 \quad A_{kj}=-1 \quad \text{if } (i,j)\in\mathcal{P} \quad \text{else} \quad 0$$

On the other hand, the matrix D_w indicates whether instances are support vectors. The pairs (i,j) in P are assumed have a fixed order and for each position k in this order:

$$(D_w)_{k,k}=\begin{cases}1 \quad if\ 1-\left(f(x_i)-f(x_j)\right)>0\\ \qquad 0, \quad else\end{cases}$$

The simplified representation matrix A_w represents A, but has been limited to support vector pairs $A_w\in\{-1,0,1\}^{p_w,n}$, with p_w being the number of support vectors. The values 1 and -1 are due to three conditions: that survival time is shorter or longer than some other subjects, the subject itself or the other is uncensored, and the pair is a support vector. For the two possible values, subsets of P can be constructed fulfilling these criteria for 1 and -1 and used to construct A_w. For the conjugate gradient, the required information from these sets are their size and the dot product sum of the vector from both the conjugate gradient and the feature vectors for their members. If these values are known, the complexity for an iteration would decrease to $O(nd + d)$. They can be constructed iteratively with an order statistic tree, which is a balanced binary search tree which keeps track of a size and sum for each subtree of each node (Pölsterl et al., 2015).

A similar approach with truncated Newton optimization can be used to adopt support vector regression models with the following loss function. Support Vector machines for regression assume that the target variable is another dimension to the feature space, and it tries to place a hyperplane which minimizes the prediction error (Pölsterl et al., 2015):

$$\underset{\beta_0,\beta_1,\dots,\beta_d}{\operatorname{argmin}} \left(\frac{\gamma}{2} \sum_{i=1}^{\mathrm{n}} \left(\xi(\beta, \mathrm{t_i}, \mathrm{x_i}, \delta_\mathrm{i}) \right)^2 + \frac{1}{2} \sum_{j=1}^{d} \beta_j^2 \right)$$

$$\xi(\beta, \mathrm{t_i}, \mathrm{x_i}, \delta_\mathrm{i}) = \begin{cases} \max\left(0, t_i - \mathrm{e}^{(f(x_i) - \beta_0)}\right) & if\ \delta_i = 0 \\ t_i - \mathrm{e}^{(f(x_i) - \beta_0)} & if\ \delta_i = 1 \end{cases}$$

The two described loss functions can also be combined. This leads to a hybrid model with a mixing parameter α moving the target between AFT regression and ranking (Pölsterl et al., 2015):

$$\underset{\beta_0,\beta_1,\dots,\beta_d}{\operatorname{argmin}} \left(\frac{\gamma}{2} \left(\alpha \sum_{i,j \in \mathcal{P}} \max\left(0{,}1 - \left(f(x_i) - f(x_j)\right)\right)^2 + (1-\alpha) \sum_{i=1}^{\mathrm{n}} \left(\xi(\beta, \mathrm{t_i}, \mathrm{x_i}, \delta_\mathrm{i}) \right)^2 \right) + \frac{1}{2} \sum_{j=1}^{d} \beta_j^2 \right)$$

2.2 Prediction Scoring

The hierarchy in the goals of survival predictions allows scoring at different levels of prediction. Ranking within group of subjects based on their risk is available for all models. Models using the AFT assumption can predict the expected survival time which can be negated in order to use it as a risk score. Models with the proportional hazard assumption can use the factor ψ. Survival trees predict either cumulative hazard functions and/or survival functions. Ordering scores can be constructed from these functions with the previously mentioned conservation of events lemma. A higher number of expected events equals a higher risk. Ordering is therefore based on the cumulative sum of the predicted cumulative hazard function. If only the survival function is available, then the exponential transformation could be used. If the gradient boosting model uses the IPCW squared error, then it also predicts on the time scale, while a boosting model with the Cox based loss predicts on the risk scale. For ranking survival support vector machines one can directly use their prediction values for ordering, while regression and mixed target survival support vector machines predict on the time scale and their prediction needs to be negated (Pölsterl, 2020).

To compare the predicted order based on the scores $\hat{y_i}$ against the real order of the subjects, the concordance index (C-index) can be used. It is the fraction of pairs from the observed data usable for order determination $\mathcal{P}$, which are predicted by the model to be in the right order (Harrell et al., 1996b). Subjects which are predicted to have the same risk score only get attributed by half to the count of concordant pairs:

$$C = \frac{\sum_{(i,j)\in\mathcal{P}} \mathrm{I}(\hat{y}_i \geq \hat{y}_j) - \left(0.5 \cdot \mathrm{I}(\hat{y}_i = \hat{y}_j)\right)}{|\mathcal{P}|}.$$

The range of the statistic is 0 to 1. With random predictions or the same score for all instances, the score is expected to be 0.5. A problem with this C-index is that it depends on the censoring distribution for the observation. By applying the IPCW based on the estimated censoring distribution (Uno et al., 2011), this dependency bias for comparing studies is corrected. This score needs an estimator of the censoring distribution over the entire range of times from the subjects being used as a scoring set. Because the passed subjects may have survival times higher than the maximal event time in the training set for the estimated censoring distribution $\hat{C}$, scoring on the test set is limited up to a maximal time τ, ignoring subjects where $t_i > \tau$.

$$C_{IPCW} = \frac{\sum_{(i,j)\in\mathcal{P}_\tau} w_i \left(\mathrm{I}(y_i \geq y_j) - \left(0.5 \cdot \mathrm{I}(y_i = y_j)\right)\right)}{\sum_{(i,j)\in\mathcal{P}_\tau} w_i}$$

$$\mathcal{P}_\tau = \{(i,j) \in \mathcal{P} \wedge t_i < \tau \wedge t_j < \tau\}$$

$$w_i = \begin{cases} \frac{1}{\hat{C}(t_i)^2} & if\ \delta_i = 1 \\ 0, & else \end{cases}$$

For binary classification problems, the receiver operating characteristic (ROC) curve is a tool to visualize the influence of a decision function when varying the threshold or different parameter configurations for a classifier. Assuming the target being $Y_i \in \{-1,1\}$, the decision function output being C and the decision boundary value as c, it plots the true-positive rate (TPR, sensitivity) against the false-positive rate (FPR, 1-specifity) (Pepe et al., 2009).

$$TPR(c) = P(C > c|Y = 1) \qquad FPR(c) = P(C > c|Y = -1)$$

A good model should have a ROC curve close to the upper left corner. A model predicting random classes will follow the bisector line. To report the results from the ROC curve, the area under the curve (AUC) of the ROC curve is often reported. It has a similar interpretation to the C-index with the range is between 0 and 1, but a model with an AUC under 0.5 is considered to have been incorrectly trained or scored and negating its prediction will result in a better model.

For survival prediction models, a time dependent ROC can be developed. One approach is the cumulative/dynamic ROC, which uses a cumulative TPR and a dynamic FPR. The decision function output is the previously described risk score and the target variable translated to have an event seen before or after the current time point (Lambert & Chevret, 2016).

$$TPR^C(c,t) = P(C > c|T \leq t) \qquad FPR^D(c,t) = P(C > c|T > t)$$

A ROC curve for this definition describes the distinguishing power between subjects who fail at a point in time t and before that. By evaluating the AUC for each point t, a time dependent plot or a mean AUC can be used as a scoring metric. Another possible implementation is the incident/dynamic ROC, where the incident TPR is the probability of the score being higher than a threshold for subjects failing at time t and the dynamic ROC (Lambert & Chevret, 2016).

$$TPR^I(c,t) = P(C > c|T = t) \qquad FPR^D(c,t) = P(C > c|T > t)$$

The cumulative/dynamic ROC has a higher value for clinical applications. A cumulative/dynamic AUC can be defined as (Lambert & Chevret, 2016):

$$AUC^{C,D}(t) = P(C_i > C_j|T_i \leq t, T_j > t)$$

To incorporate censoring correctly, IPCW weighting is used (Pölsterl, 2020):

$$\widehat{AUC}^{C,D}(t) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} I\left(t_j > t\right) I(t_i \leq t) w_i I(c_i \geq c_j)}{\left(\sum_{j=1}^{n} I\left(t_j > t\right)\right)\left(\sum_{i=1}^{n} I(t_i \leq t)\, w_i\right)}$$

Like with the weighted C-index, a training dataset for the estimation of the censoring distribution is necessary. Over the time range between τ_1 and τ_2 a weighted integral of the AUC considers the change of survival probability estimated with Kaplan-Meier from the test data along the time range (Lambert & Chevret, 2016):

$$AUC^{C,D}(\tau_1, \tau_2) = \frac{1}{\hat{S}(\tau_1) - \hat{S}(\tau_2)} \int_{\tau_1}^{\tau_2} \widehat{AUC}^{C,D}(t) d\hat{S}(t)$$

For models which can provide a cumulative hazard function as a prediction instead of a single score c, it can be used as a time dependent score in the AUC calculation.

A third metric type is the Brier score. This is another time dependent score, but it uses survival functions as the prediction output (Kantidakis et al., 2020a; Pölsterl, 2020):

$$B(t) = \frac{1}{n}\sum_{i=1}^{n} I(t_i \leq t \wedge \delta_i = 1)\frac{\left(0 - \widehat{S}_i(t)\right)^2}{\hat{C}(t_i)} + I(t_i > t)\frac{\left(1 - \widehat{S}_i(t)\right)^2}{\hat{C}(t)}$$

To account for censoring, IPC weighting is used. The Brier score behaves more like a loss. It is scaleless and a lower value close to 0 is better. A single score can be constructed using the integral over the time range while weighting later time points higher (Kantidakis et al., 2020a):

$$IBS(\tau_1, \tau_2) = \int_{\tau_1}^{\tau_2} B(t)d\frac{t}{\tau_2 - \tau_1}$$

2.3 Data Preparation

The data was extracted from the transplantation registry (Nagel, 2022) export tool on the 11th of February 2022. The focus of the analysis were the kidney transplantations. Files specific to other organs were ignored. Three institutions contribute data to the registry (TXReg). They are the Deutsche Stiftung Organtransplantation (DSO), Eurotransplant (ET), and Institut für Qualitätssicherung und Transparenz im Gesundheitswesen, (IQTIG). Th main identifiers are the encrypted versions from the ET system for the recipient, donor and performed transplant. Only the ET sourced data uses the transplantation identifier, while the others use the combination of recipient and donor or only one of the two when identifying transplants. All values except the identifiers have been lower cased and certain terms have been translated into English or more consistent versions. Clipping and replacement of invalid values for numeric columns were applied. Columns with different units were converted to the most frequent one, if possible, while text-based ordinals were converted to numbers. Data preparation was implemented in Python with the Pandas library (McKinney, 2010; McKinney & Team, 2015). Please refer to the notebooks in the digital attachments for more details on the process and to the environment specification files for exact software versions. All calculations were performed on an AMD EPYC 7402P processor with 256 GB memory and two disk based hard drives using ZFS mirroring with enabled memory caching on Debian 11.

The following figure shows the process of a kidney transplantation excluding living donors and data files receiving values during this process. There are some cases where columns in a file receive updated values later in the lifecycle. Waiting list states denote whether a patient is *ready to receive a transplant, received one, or was rejected from the list. For the deceased donors, the crossmatch result file was excluded for the visualization.*

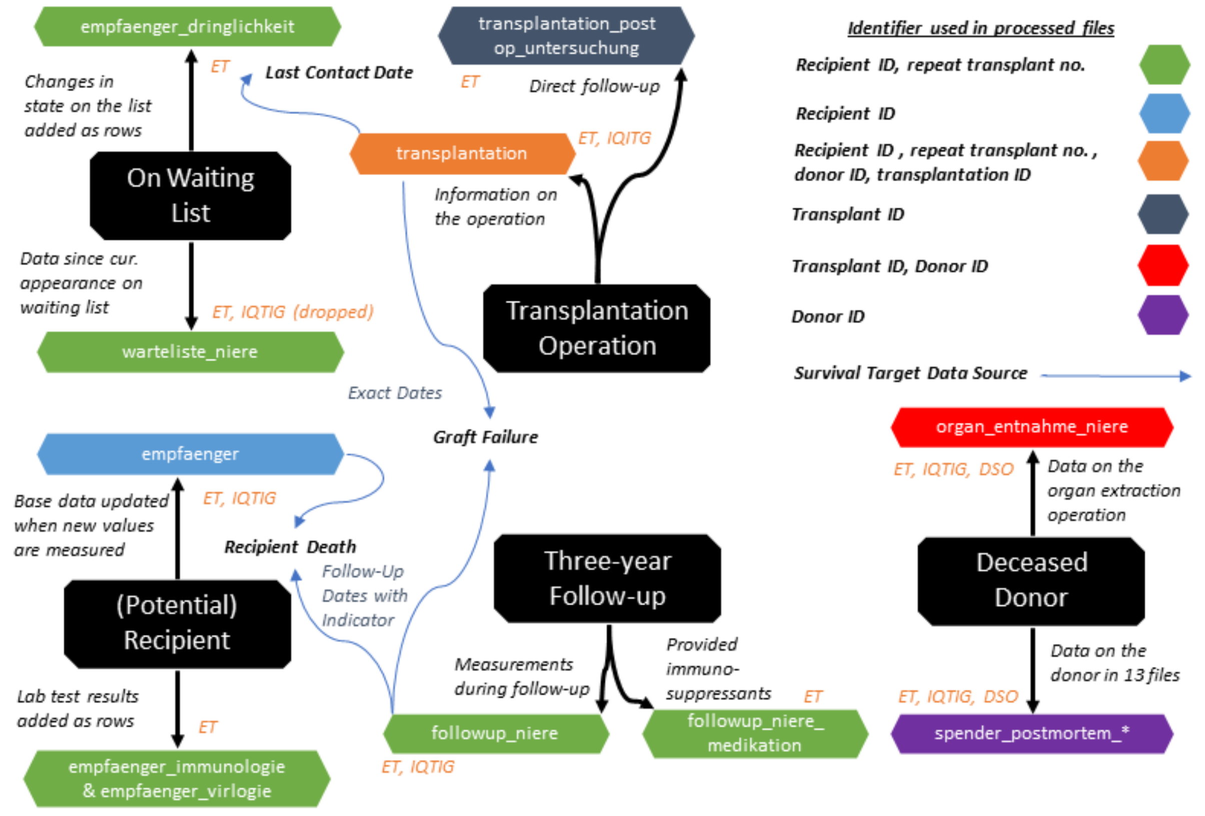

empfaenger_dringlichkeit
ET
Last Contact Date
Changes in state on the list added as rows
On Waiting List
Data since cur. appearance on waiting list
ET, IQTIG (dropped)
warteliste_niere
transplantation_post op_untersuchung
ET
Direct follow-up
transplantation
ET, IQTIG
Information on the operation
Transplantation Operation
Exact Dates
Graft Failure
Identifier used in processed files
Recipient ID, repeat transplant no.
Recipient ID
Recipient ID, repeat transplant no., donor ID, transplantation ID
Transplant ID
Transplant ID, Donor ID
Donor ID
Survival Target Data Source
empfaenger
Base data updated when new values are measured
ET, IQTIG
Recipient Death
Follow-Up Dates with Indicator
(Potential) Recipient
Lab test results added as rows
ET
empfaenger_immunologie & empfaenger_virlogie
Three-year Follow-up
Measurements during follow-up
Provided immuno-suppressants
ET
followup_niere
ET, IQTIG
followup_niere_medikation
organ_entnahme_niere
ET, IQTIG, DSO
Data on the organ extraction operation
Deceased Donor
Data on the donor in 13 files
ET, IQTIG, DSO
spender_postmortem_*

Figure 1 Files processed from the registry export and used identifiers after processing

2.3.1 Recipient data

The ET sourced file "empfaenger_dringlichkeit" contains the waiting state changes for every past and potential future recipient for all tracked organs. In total 47436 potential kidney recipients (1645 reappearances) could be identified by a column specifying the requested organ, while 27618 (645 reappearances) received a transplant. This file was also used to find patients who received multiple kidney transplants and an approximate date of these transplantations. The number of state changes in the waiting history and the duration of the recipient stay on the waiting list were extracted as additional features from this file.

More information relevant to the kidney recipient waiting list and matching process is available in the file "warteliste_niere". Some columns in this file are sourced from the ET, others from the IQTIG. The IQTIG columns were dropped because they appeared to contain redundant information to the ET columns. This file contained 245 duplicates. These are caused by additional entries for patients which were removed from the waiting list, either due to unfitness for transplantation or incomplete registration. The ET data also has a date column and waiting state, which were used to associate a number to patients who received multiple transplants over time. All recipients' data were kept, even if their newest state was not set to "transplanted". This was done to consider potential missing state changes to "transplanted". The newest values for every column for each recipient and repeat transplant number pair was kept removing the mentioned repeats. In the resulting table, 47439 (1596 reappearances) potential recipients are listed of which 27621 (645 reappearances) received an organ. These number slightly differ from the ones for the "empfaenger_dringlichkeit", suggesting potential inconsistencies between files.

Base data like age, height, biological sex is presented with the "empfaenger" file. This file also contains columns from IQTIG and ET, but the rows have already been joined by the registry. If the same data appeared in ET and IQTIG columns, their mean was calculated for numeric columns. For other column data types, the result was the missing value specifier in the case of a disagreement between the columns. The "empfaenger" file also reports an exact death date for the recipient, which was used for the survival data later in the processing pipeline. The data on 81390 potential recipients for any organ was retained.

Immunologic testing data can be found in the "empfaenger_immunologie" file. Each row is a test result communicated to the registry by the ET. There are different types of tests which reuse the columns between them. The transplantation dates from the kidney waiting history were used to associate the tests with the correct transplant by keeping all tests with dates equal or earlier to the transplantation date. The tests were then separated based on column values and presence of non-missing values in columns. The results were stored in new columns for each test. Again, only the newest test results were kept for each of the 47150 potential kidney recipients and repeat transplant number pairs in this file.

In the following table the presence of the test types is presented relative to all subjects present in the processed recipient immunology data file.

Table 3 Presence of immunologic tests in the processed "empfaenger_immunologie" file

Test	Availability of test results in the processed file
PRA percent	99.4%
HLA Phenotyping	66.0%
DTT Crossmatch	35.3%
Auto Antibodies	34.8%
Unacceptable Antigens	20.4%
vPRA percent	12.2%
Not Cytotoxic Antibody Presence	7.2%
Donor Frequency HET	2.7%

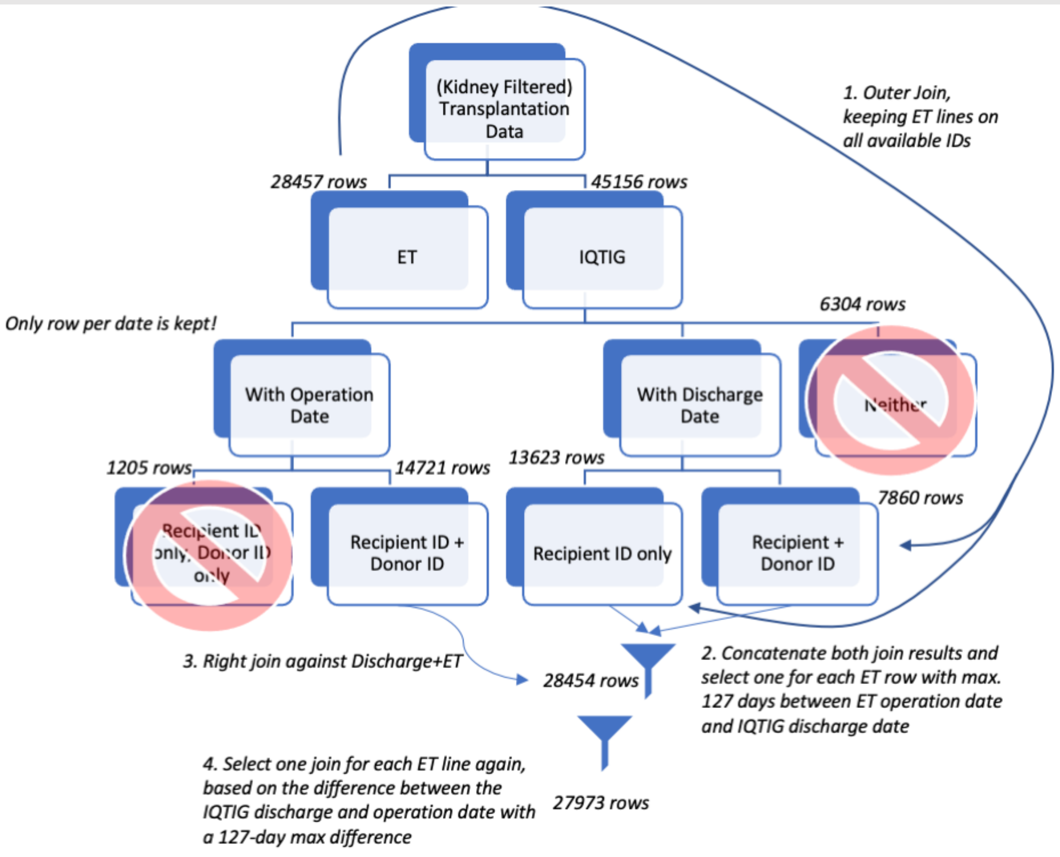
Donor Frequency ETKAS	2.7%
Acceptable Antigens	2.1%

The same procedure is used on the virology testing data in 'empfaenger_virologie' from the ET. While different tests are represented by different rows, each test result already has its own column in this file. Common pathogens, antigens, and antibodies test results are communicated for 44356 potential kidney recipients and repeat transplant number pairs.

2.3.2 Transplantation data

Information on the actual transplantation of the organ is provided by IQTIG and ET in "transplantation". While the ET provides one row per transplant together with a transplantation identifier, IQTIG provides data using multiple rows, sometimes only specifying recipient or donor. The ET rows have an organ column specifying which organ was transplanted. This is not the case for the IQTIG rows. Instead, the rows with the ICD10 prefix N0 to N39 and OPS codes beginning with 5-55 which are all kidney related, were kept. Some IQTIG columns are also contained kidney specific values. If any of those rows were used, the row was also kept.

The following figure illustrates the splitting and joining steps for the transplantation data file. It is important to note that every set was reduced by removing reoccurring dates by collapsing the columns together while only the newest values according to the file order were kept.



(Kidney Filtered) Transplantation Data
1. Outer Join, keeping ET lines on all available IDs
28457 rows
45156 rows
ET
IQTIG
6304 rows
Only row per date is kept!
With Operation Date
With Discharge Date
Neither
1205 rows
14721 rows
13623 rows
7860 rows
Recipient ID only, Donor ID only
Recipient ID + Donor ID
Recipient ID only
Recipient + Donor ID
3. Right join against Discharge+ET
28454 rows
2. Concatenate both join results and select one for each ET row with max. 127 days between ET operation date and IQTIG discharge date
4. Select one join for each ET line again, based on the difference between the IQTIG discharge and operation date with a 127-day max difference
27973 rows

Figure 2 Transplantation Data Split and Join Process

In the IQTIG set, rows either have a discharge date, an operation date, or neither. The 6304 rows with neither were ignored. If operation date rows only contained recipient information, they appeared to be mostly describing liver transplants. Those 57 rows with an operation date were ignored as well. The 1148 rows with an operation date and only donor information were also removed, as the recipients are the subjects of the analysis and correctly joining them is difficult. For both operation and discharge date rows, only the newest values for each column were kept, discarding repeating rows.

The 28457[3] ET rows were used for a left join against the 7860 IQTIG discharge date containing rows with both a donor and recipient identifier, but also against the 13623 rows with only the recipient identifier. This results in a reoccurrence of transplants. To find the correct associations between rows, the difference between the discharge date from IQTIG and the operation date from the ET data has been used to drop lines where the discharge would be before the operation date and more than 127 after it. The 127-day limit was chosen by testing different offsets on whether they correctly filter matching IQTIG and ET rows. This was done by manually inspecting the results through the joining process. Then, for every ET line joined, only the result joins with the smallest date difference are kept.

The 28454-row result was right joined with the IQTIG operation date rows on the recipient identifier. The difference between the IQTIG operation date and the discharge date was then used for the same procedure to keep only one ET line and its join partner.

The repeat transplant number was joined based on the smallest difference from the operation date communicated by ET to the state change date in the waiting state file. This resulted in one row being removed from the processed transplantation data. This transplantation file also reports exact failure dates, again also used later for the survival analysis. Furthermore, it also reports the last time the ET has had any follow-up with the transplant patient and whether the ET considers the patient lost to follow-up, but this lost-to follow-up column is sparsely populated.

Shortly after the transplantation, there is one direct follow-up examination. Measurements from this checkup are stored by ET in “transplantation_postop_untersuchung”. It contains multiple rows associated with one of the 52603 transplant identifiers. The values from these

[3] 2 additional ET lines had graft failure dates before the operation date. They were considered incorrectly joined or incorrectly registered and deleted.

measurements were added to the general follow-up described later. When the organ is extracted from the donor, parameters are recorded by ET, IQTIG and DSO in “organ_entnahme_niere”. While the DSO and ET data are already joined in this file, IQTIG data lines are separate. There are sometimes multiple rows for a single donor, therefore the joining process was another N:M join on the donor identifier. After consolidating the same measurement variables from different institutes, the information was added to the donor id in the index, which of the two kidneys had been extracted as reported from DSO and ET. After this addition, only 21 rows were not distinct. Those rows seem to be the results of repeated operations. The 31659 lines with the highest sum of all date columns for each donor id and kidney side were kept in order to only keep the newest results.

2.3.3 Follow-Up Data

Long-term health development of the recipients is observed with follow-up visits. Data from these visits are communicated by ET and IQTIG in “followup_niere”. Identification is done with the recipient identifier and the transplant identifier from the ET. To associate the follow-up visits with the pair of recipient identifier and repeat transplant number, the state change date was moved seven days back to accommodate for a date registration lag, which seems to appear sometimes. From this N:M join, only those rows were kept where the date of measurement was after the date of transplantation but before the next one if there was another kidney transplantation for this recipient. 27048 transplants remained with follow-up data after this process.

Values which appeared only once for all patients were used as scalar features. For repeating measurements only, the first value was used as a feature to prevent values, which are only measured long after the date of transplantation, to leak survival times to the models. These first values are measured shortly after the transplantation. During follow-up, immunosuppressants are administered to the patients, which is recorded by the ET for only 4133 transplant identifiers in “followup_niere_medikation”. IQTIG also records data from follow-up visits with living donors, but in this project, there was a focus on deceased donor transplants, as there is more data is available.

2.3.4 Deceased donor data

Many files record data on deceased donors. Base data is communicated by all institution in “postmortem” and are already joined on 15327 donors. When a donor dies, the lead-up and exact cause of death are recorded by ET and DSO in “spender_postmortem_diagnosen”. These

rows are not joined for each donor and use the columns in this file differently. However, lead-up and death diagnosis features were created and joined to one row for 13937 donors.
Data on blood gases (“spender_postmortem_labor_blutgase”, 11950 donors), blood typing (“spender_postmortem_labor_blutgruppe”, 14334 donors), microbiological testing (“spender_postmortem_labor_mikrobiologie”, 4423 donors), toxicologic testing (“spender_postmortem_labor_toxikologie”, 1633 donors), urine lab tests (“spender_post-mortem_labor_urin”, 11638 donors) and virologic tests (“spender_postmortem_labor_virol-ogie”, 13920 donors) were sourced from the ET and the DSO. Only the latest results were kept and the same measurement variables between institutes were consolidated with the arithmetic mean after unit conversions. Prior to a transplant, crossmatch testing is performed between the donors and potential recipients by the DSO. The results from these tests are in “spender_postmortem_labor_crossmatch”, with lines for all evaluated pairs of recipient and donor, sometimes with repeated evaluations. Only the latest comparison for all 162446 pairs was kept.
Human leukocyte antigen (HLA) test results for deceased donors are stored by the DSO and ET in “spender_postmortem_labor_hla”. The DSO provides the results by HLA groups and whether the test was performed with DNA sequencing or serological testing, while the ET communicates all antigens present in a single column. The latest results in this column were kept, resulting in data for 13718 donors. Other lab tests results are submitted by the ET, DSO and IQTIG in “spender_postmortem_labor_klinische_chemie”. ET and DSO results are already joined sometimes. After taking only the newest results for each column based on the sampling date, the mean was used for already joined rows and redundant columns were collapsed to a table with data on 15224 donors.
For deceased donor organs, pathologic analysis of different organs is performed. DSO communicates the results as separate rows with a quality parameter and text comment in “spender_postmortem_labor_pathologie”. The latest quality observation for each organ is extracted as a feature. However, that was only available for 4932 donors. For examinations prior the death of the patient by DSO, which are reported in “spender_postmortem_unter-suchungen”, the same procedure was applied. Diagnosis data is available for 12209 donors. If the donor received antibiotics or any other medications before their death, that is also noted by the ET and DSO in ‘spender_postmortem_medikation’. The start dates were consolidated, and features were created for each of the 13871 donors. For the 13947 deceased donors who

were under medical monitoring before their death, data by ET and DSO in ‘spender_postmortem_monitoring’ was also joined and consolidated.

2.3.5 Dataset Joins and Survival Target

All the recipient data including the scalar follow-up data were joined on the recipient identifier and the repeated transplant number if available. Only recipients who appeared on the waiting list files were kept. This resulted in a file with data on 48726 potential kidney recipients respecting repeats. All deceased donor files except for the crossmatch results were outer joined on the donor identifier. Data for 15822 deceased donors were available. The crossmatch results were instead right joined with the transplant data and then left joined with the organ removal data. This transplant table then contained 27972 rows.

After these file groups were connected, the transplant and recipient data were right joined based on the recipient identifier and the repeated transplant number. To these results the donor data was inner joined on the donor identifier. Only rows with an operation date by ET were kept and it was used to center all date columns. Overall, this resulted in 20870 transplant rows of which only the 20325 first time transplants were kept for the raw dataset.

The following figure shows the count of rows which contain information from a specific file in the raw dataset. As the availability of data in files was used to filter rows, all rows contain information from “empfaenger”, “empfaenger_dringlichkeit”, “transplantation” and “warteliste_niere”. “followup_niere” refers to the single time data points.

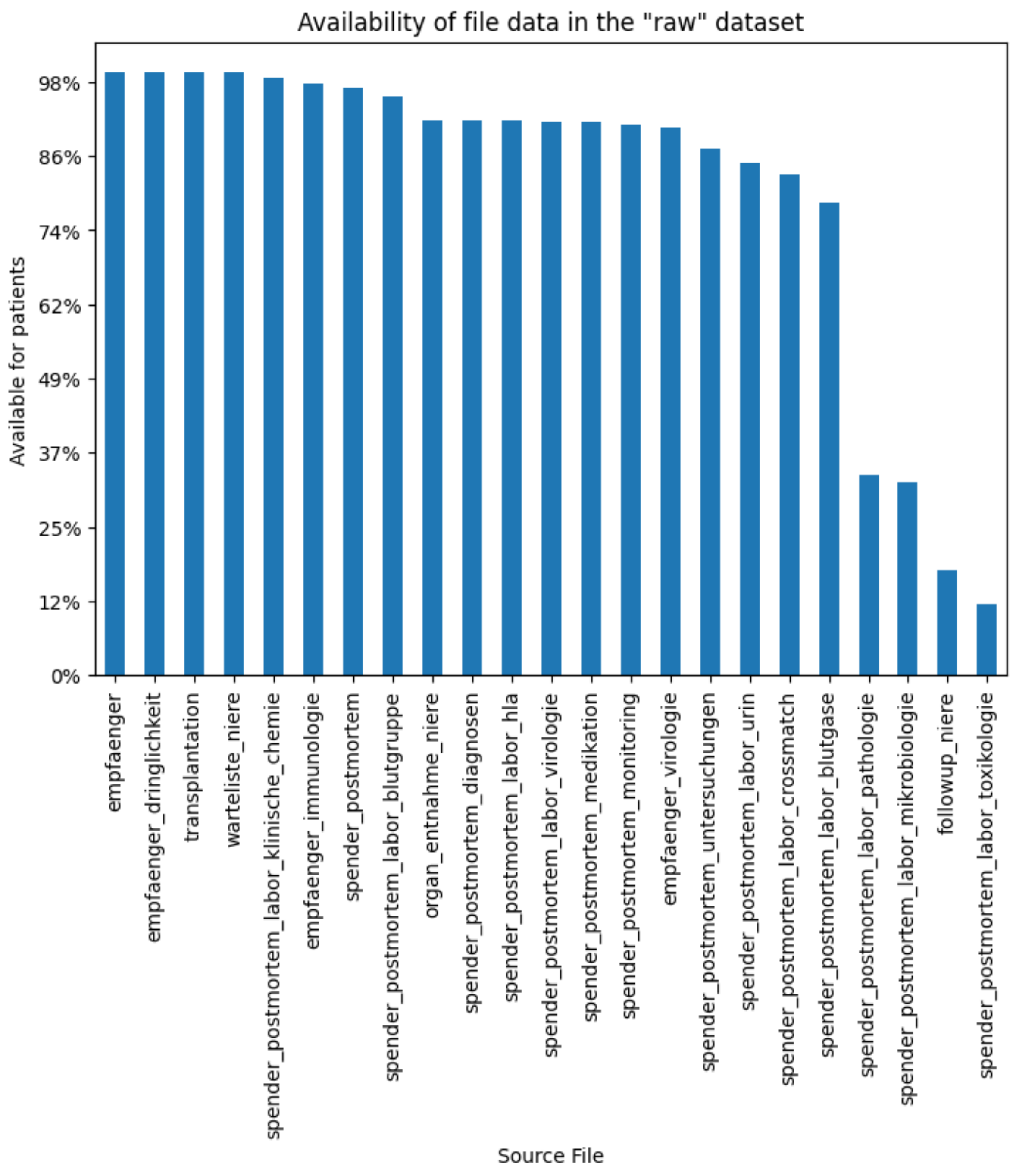


Figure 3 Presence of the raw dataset rows in the input data files.

After the join, HLA mismatch features were calculated based on the procedure outlined by the ET guidelines (Tieken, de Boer, & Hagenaars, 2022; Tieken, de Boer, Heidt, et al., 2022). First, all antigens from both serological and DNA sequencing results were mapped to broad and splits antigens[4]. The mismatches for the broad and split categories for the A, B, CW, DQ and

[4] As some antigens were missing from the ET manual, a mapping file was created manually. It can be found in the digital attachments.

DR classes were calculated. In accordance with the ET procedures for A, B and CW, the broad mismatches were used instead of the split mismatches if available, while for DQ and DR the split results were used, if available, over the broad results.
Patient death and graft failure were identified as the targets for the survival analysis. All target variants are listed in Table 4. Whether a patient died, or their graft experienced a failure event, is tracked for follow-up visits in the "followup_niere" file. However, as can be seen in Figure 1, an exact death date is also reported in "empfaenger" and exact failure date in "transplantation" as well. The follow-up information reports the dates of the follow-up visits and an indicator, whether an event was registered. From this follow-up data, a target of the last follow-up visits and an indicator variable was created for the patient death (DF) and the graft failure (FF).

The exact death dates and failure dates were also used to create targets. To incorporate censoring information, patients who were present in the corresponding follow-up target (DF, FF), but did not have an exact date registered, were considered censored at their last follow-up visit. This combination resulted in exact date event targets for patient death (DE) and for graft failure (FE).
A final fifth survival target (EE) is based on the last-seen column from the ET data in "transplantation". This was used as the censoring date. Event times were based on the exact event dates as either the graft failure date from "transplantation" or the patient death date from "empfaenger", whichever came earlier (Nagel, 2020). When the last contact date was unrealistically large (sometimes hundreds of years) only patients with observed times up to 10 years were retained, which dropped 5% of patients. This truncation was performed instead of censoring, as those were considered invalid data points. To allow AFT models to take the logarithm of the time points, the time for all targets was incremented by one day.

The following table shows the creation method for each of the survival targets in context of the previously defined random variables for right censored data.

Table 4 Construction methodology for the five different targets

Target	Event	T^*	C	T	δ
FF	Graft failure	-	-	Date of the last follow-up visit when a value was reported	Value reported in the latest follow-up visit
FE	Graft failure	Exact failure date reported in "transplantation"	Date of the latest follow-up visit (T_{FF})	T_{FE}^* if it was set, C_{FE} otherwise	Was $T = T_{FE}^*$?
DF	Patient death	-	-	Date of the last follow-up visit when a value was reported	Value reported in the latest follow-up visit
DE	Patient death	Exact death date reported in "empfaenger"	Date of the latest follow-up visit (T_{DF})	T_{DE}^* if it was set, C_{DE} otherwise	Was $T = T_{DE}^*$?
EE	Patient death and graft failure	$\min(T_{DE}^*, T_{FE}^*)$	Latest contact date reported in "transplantation"	T_{EE}^* if it was set, C_{EE} otherwise	Was $T = T_{EE}^*$?

The following figure shows the number of patients in the five survival targets. The variants of the graft failure and death target patient sets were combined for easier visualization.

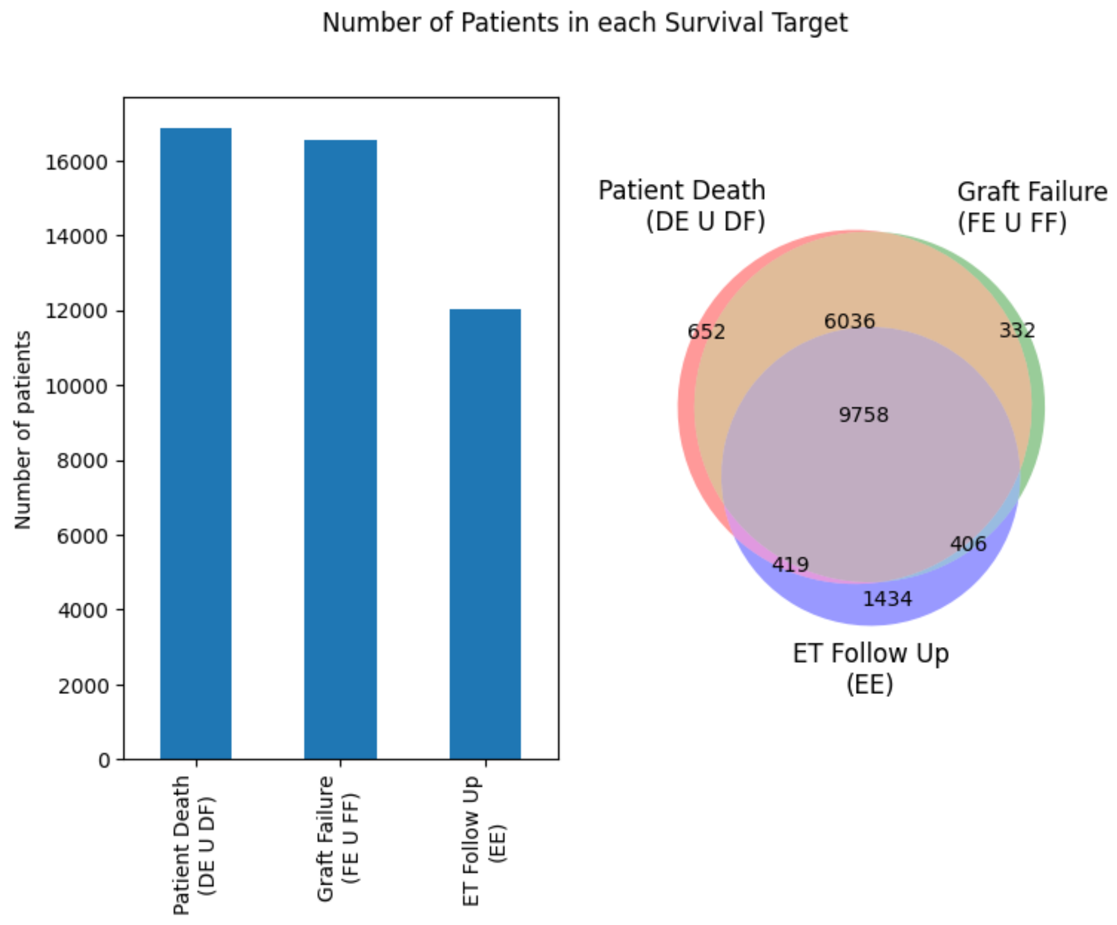


Figure 4 Transplant counts for each survival analysis target

2.4 Feature preprocessing

The raw dataset was further processed prior the training and test split. This transformed the raw dataset into the prepared feature dataset. The feature preprocessing was implemented with scikit-learn pipelines (Pedregosa et al., 2011) and custom written data transformers. Only features with at most 49.9% missing values and more than one value besides the missing-value indicator were kept. Categorical features defined, as text containing features with at most nine distinct values, were one-hot-encoded, while categories with less than 1% occurrence were collapsed. Missing values were treated as their own category. To avoid a singular feature matrix, the first category of every encoded feature was removed.

Other textual features were dropped. Columns with a variance of zero were also dropped. Afterwards, absolute Pearson correlation between the feature was calculated. When the absolute correlation was higher than 0.9 in a pair of features, then alphabetically first feature was removed.

The following figure shows the number of features present in the dataset after each of the described steps between the raw dataset and the preprocessed feature dataset.

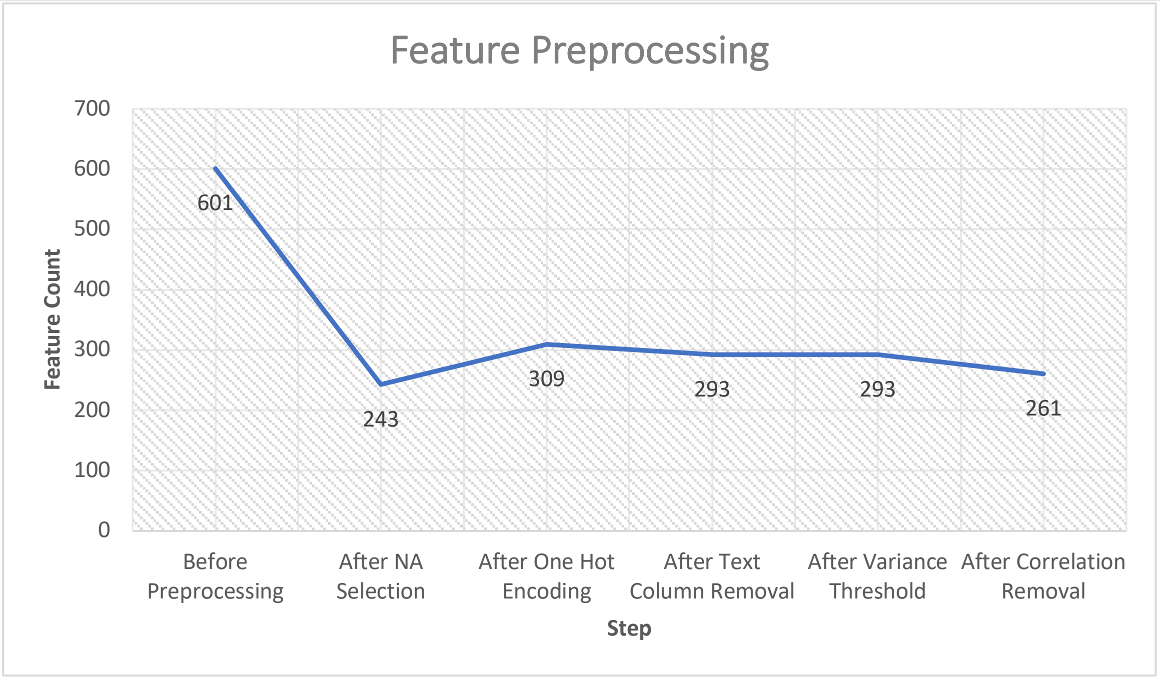


Figure 5 Number of remaining features through the preprocessing pipeline

2.5 Model Candidates

If any of the models used a random number generator, a seed value was set. For the Cox regression, the implementation in scikit-survival “CoxPHSurvivalAnalysis” (Pölsterl, 2020) with the allowed iterations increased to 1024 was fitted. In addition, the adoption of elastic net penalties to the Cox regression in “CoxnetSurvivalAnalysis”, also in scikit-survival, was used. The possible alpha values were determined by the automatic algorithm. For the elastic net mixing parameter, the values tested were 0, 0.25, 0.5, 0.7, and 1. A baseline model using only a non-parametric estimate with the Kaplan-Meier estimator was implemented with the “survfit” function from the R survival package (Terry M. Therneau & Patricia M. Grambsch, 2000; Therneau, 2022), ignoring features in the class “RCoxphNull”.

Survival trees in “RangerTreeSurvival” from the Python ranger package (Wright & Ziegler, 2017) were fitted. They use the log-rank impurity measure. Tree depths limited to 3, 26 and 50 were compared. The minimum leaf size was set to 3. Otherwise default parameters were used. The random forest from ranger in “RangerForestSurvival” had the maximum number of features considered at a split set to the square root of the total number of feature inputs. For all ensemble models, like the random forest, 10, 25 and 50 were set as the number of ensemble members.

With the function “survreg” from the R survival package parametric models with either a Weibull, exponential or log-normal distribution were tested with the class “RSurvreg”, with the number of maximum iterations increased to 1024. Their survival function was calculated using the quantile function with 1024 points between the 0.01 and 0.99 quantile of the survival distribution. Cumulative hazard functions were constructed using the negative exponential relationship.

A linear survival support vector machine (SVM) from scikit-survival “FastSurvivalSVM” with the alpha being with four values along a natural logarithmic scale between exponent -6 to -0.5 was also trained. More complex SVMs were not considered due to excessively long run times. The same alpha values were also used for a ridge regression accelerated failure model implemented by scikit-survival in ‘IPCRidge’.

A gradient boosting model implemented by the scikit-survival package in the class ‘GradientBoostingSurvivalAnalysis’ was the final model. Both losses based on the Cox proportional hazard and IPCW weighted least squared accelerated failure model were used. For the regression tree parameters, the maximum number of features and the number of estimators, the same configuration as for the survival random forest were used. The maximum tree depth was increased from the default three to six, the same value used by another boosting algorithm, XGBoost (Chen & Guestrin, 2016). For the learning rate, three values from ten to the power of zero to minus five were searched.

2.6 Model Scoring

For scoring methods, the implementations of the metrics in scikit-survival (Pölsterl, 2020) were used. When calculating the metrics, the passed test data was truncated to the time range of the training data to prevent problems with metrics using the training data to estimate the censoring distribution. For the cross-validation, all patients in the training partition were used to clip the test data and estimate the censoring distribution. This was done, as the scoring function during the grid-search has no access to the training data passed to the model.

With the risk scores returned by the model, the concordance index (Harrell et al., 1996a) along the truncated times from the test partition is calculated. Weighing based on IPCW of the concordance index (Uno et al., 2011) was done with τ set to the last time an observation was censored in the training data. If the survival function was available from the model, the integrated Brier score (Schumacher et al., 2003) along 1024 linearly spaced points between the start and end point of the training data was calculated. Not all models provide predictions for the entire training time range. Parametric models for example use the quantile function of the survival distribution for time estimation. Their survival times were clipped to the minimum and the maximum of their predicted survival functions and cumulative hazard, which results in expanding the first and last values for them to the left and right, respectively.

Based on the single risk score, the mean of the cumulative dynamic AUC (Lambert & Chevret, 2016) was calculated along the same 1024 time points. Plots of the Brier score and the risk score based dynamic AUC were also constructed for the best models with the test partition as test data. If the cumulative hazard function was available for a model, it was also used for another dynamic AUC plot using the same times and potential truncation.

2.7 Model Selection

The following steps were performed on all five targets. Predictor data was selected by keeping all patients, with a value for the follow-up column of the corresponding target or the exact date column. For the model prediction on the FE, FF, DE and DF targets the data was limited to a three-year and 30 days horizon after the transplantation, as this was the longest follow-up period for most patients observed. Patients with a date after this horizon were considered censored at this maximum date. For evaluation, the data was split based on a shuffled 70/30 split into a training and test partition.

The following training pipeline was used by all models and was refitted whenever the model was trained. First all features were clipped to the 0.01 and 99.99 percentile to remove outliers. Missing values were filled with the mean of the column. Subsequently, the features were standardized and centered around the mean. For the potentially following dimension reduction, either a PCA or recursive feature agglomeration with the Ward linkage and Euclidean distance was applied. The number of components or number of clusters being tested were 64, 128 and 256.

All combinations of model hyperparameters and dimension reductions options were examined with a threefold cross validation grid-search on the training data. For each model the best configuration with the lowest average C-index on the test partitions was selected and re-trained on the entire training partition. This model was then scored against the test partition. With permutation feature importance testing, the feature importance on the test partition with five shuffles was calculated for those best performing candidates.

Because collinearity in the data without dimension reduction caused problems for these models, “IPCRidge”, “RSurvreg” and “CoxPHSurvivalAnalysis” were always trained with dimension reduction. Because features are irrelevant to it, “RCoxphNull” was also added to this group to reduce the number of unnecessary evaluations. Models using the IPCW method from scikit-survival crashed with a division-by-zero exception[5]. When subjects at the last time point show events and censorings. This is caused by the probability of these events to be censored being zero. During runtime, a patched method is used to overwrite the function in scikit-survival which censors all events at the last timestep if there are also tied censorings.

[5] This is an issue tracked here: https://github.com/sebp/scikit-survival/issues/322

3 Results

3.1 Feature Data

A first preprocessing pipeline was run on the raw data. This raw data was the result of joining the processed registry export files. This pipeline included automatic feature selection, one-hot-encoding and correlation filtering. Remaining features at each step are shown in Figure 5. This pipeline produced a prepared feature dataset.

In the prepared feature dataset 20325 subjects with 261 features are present. Features were allowed to have up to 50% missing values during processing. Of these 261 features 33% have no missing values. There are no features with no missing values, but as can be seen in the empirical distribution of the missing value fractions (Figure 6a), 69% of rows have 10% or less values missing. Overall, only 12% of values in this preprocessed feature dataset are missing. Most features are binary features with 127 features having two distinct values, excluding missing values (Figure 6b). Some also only contain only a few possible values, because only textual features were one-hot-encoded. As many binary features contained missing values, most are encoded as floating numbers to represent these missing values (Figure 6c). Most features came from the "spender_postmortem" file, which produced 51 features. The second highest number of features was 28, extracted from "spender_postmortem_labor_klinische_chemie".

The following figure contains three plots describing the prepared feature dataset. The first one (a) has been calculated by taking the empirical cumulative density of the fraction of missing values across the features and rows respectively. The second plot (b) shows the empirical cumulative density of the number of distinct values across features ignoring missing values. Finally, the third plot (c) lists data types present in the data after an automated assignment. It is important to note that missing values can only be presented with a floating number datatype.

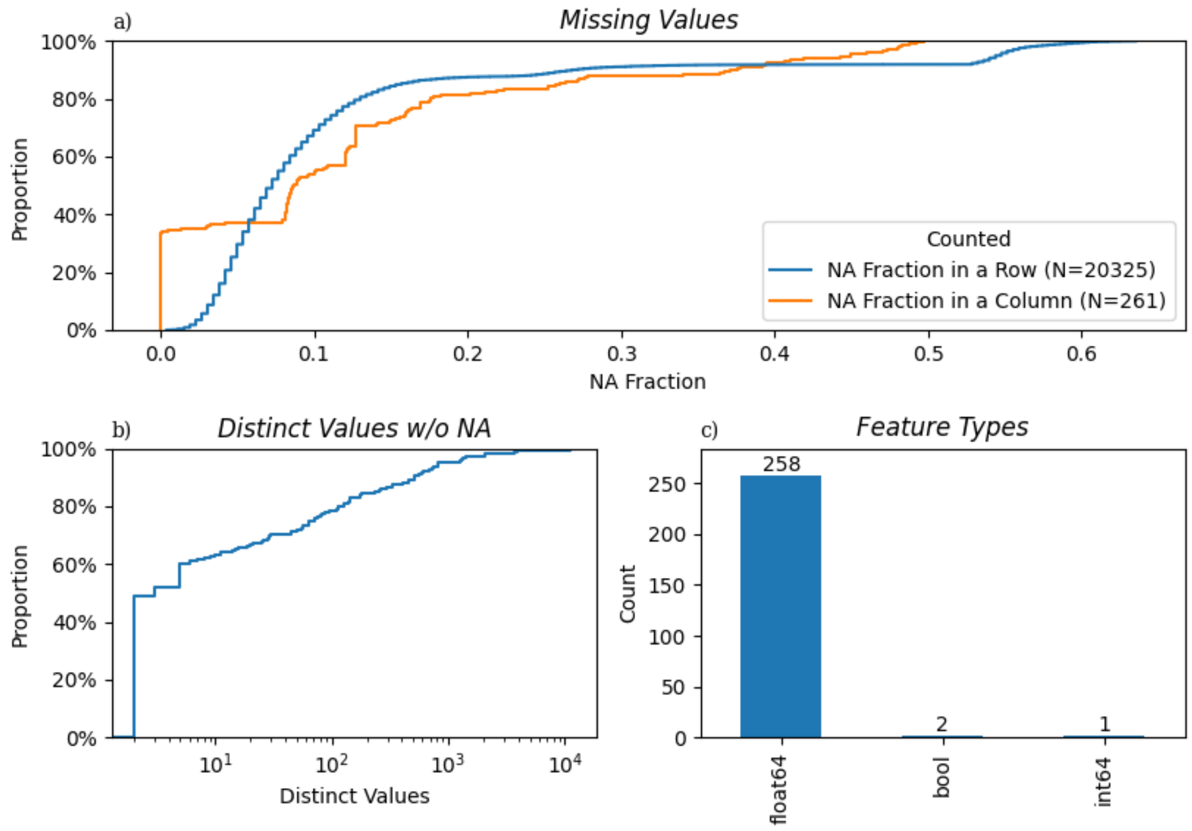


Figure 6 Missing value distribution (a), number of distinct values (b) in features, and feature data types (c) in the prepared feature dataset

To illustrate the preprocessing steps within model candidate pipelines, the percentile clipping has been applied, missing values filled with the mean and the features standardized on the entire prepared feature dataset. This is the same procedure as applied for the models, but there it is fitted on a training partition. For the figures this dataset will be called the intermediate model dataset.

Feature clustering, which is also performed in feature agglomeration, was also applied to this intermediate model dataset. The distances of the features with the Euclidean metric have been calculated and the Ward linkage between clusters used.

For the first plot in the following figure (a), the dendrogram of the features in the intermediate model dataset after feature clustering shown and cut to produce 32 clusters. Coloring was based on 70% of the maximum height. The correlation between features before clustering was calculated and is shown in the second plot (b) as a heatmap. In the third plot (c), the empirical cumulative distribution of the absolute correlations ignoring self-correlation is shown.

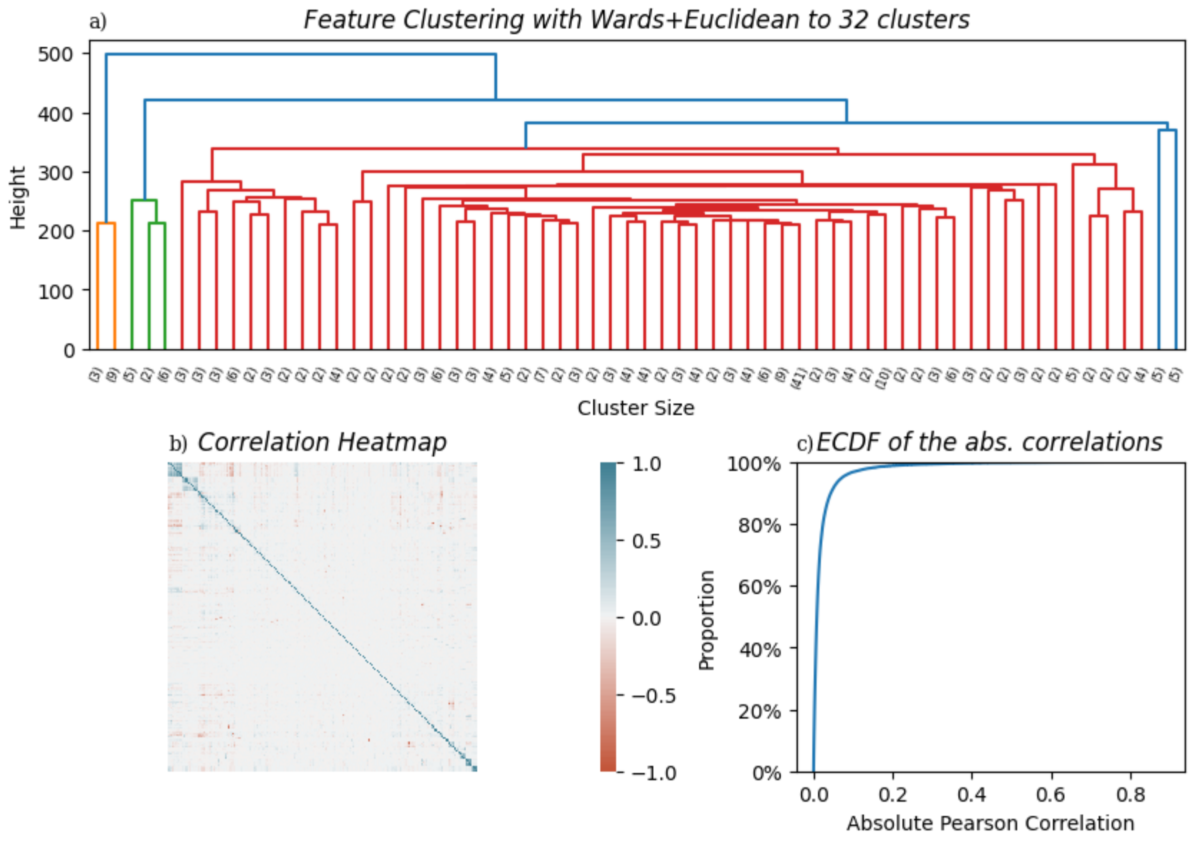


Figure 7 Feature Agglomeration example (a), Correlation Heatmap (b) and distribution of absolute correlation (c) in the intermediate model dataset.

Percentile-based clipping affected 101 of the features. 99% of the absolute correlations ignoring self-correlation are equal or below 0.230 (Figure 7b). The highest absolute correlation between columns was 0.89, which matches the filtering during preprocessing. Within the 32 clusters, the largest cluster contains 41 features. In size below that is a cluster with 10 features (Figure 7a).

While many correlated features are clustered close together, visible in the upper left-hand corner of the correlation heatmap (Figure 7b) and the right side of the dendrogram (Figure 7a), these features also appear to correlate with many other features in the dataset. They appear to be one-hot-encoded missing values for transformed textual, categorical features mainly from deceased donor data.

A PCA ran on the scaled and filled features lead to the three first components being able to explain 7.3% of the variance (Figure 8a, b). The Minka's maximum likelihood estimator (Minka, 2001) estimates the best number of components to be all components. The scatter plot of the three first components suggests a separation into two clusters of subjects (Figure 8a). The feature with the highest positive weights for the first component were also the columns which can be found in the clusters previously described on the left side of the dendrogram. For the second component, the features with the highest contribution were those specifying whether different organs of the donor were legally allowed to be transplanted and the age of the donor and the recipient. This suggests that the separation of the two clouds might be based on whether certain values were available.

For the following figure a PCA was run on the intermediate model dataset. The first plot (a) shows a scatter plot of the first three components and the second plot (b) the cumulative explained variance of all components.

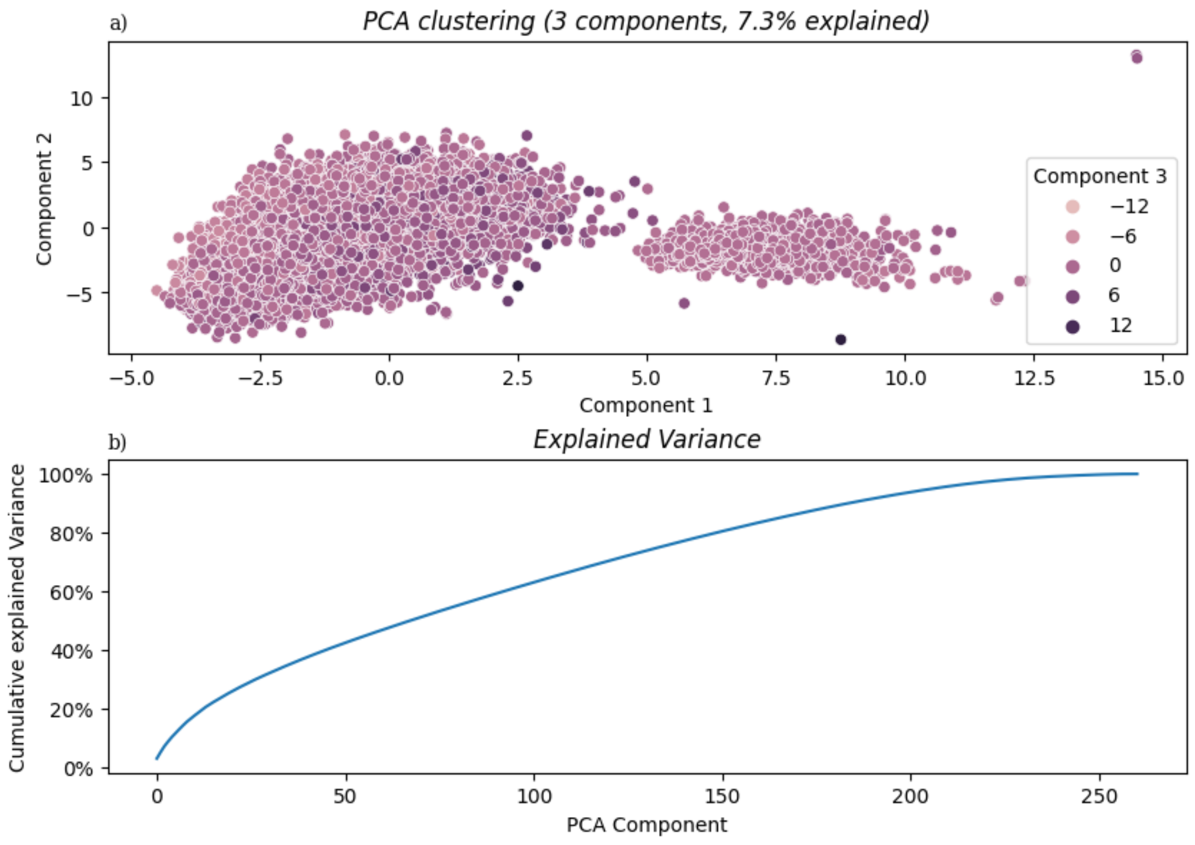


Figure 8 The PCA results. The first plot (a) is a scatter plot of the first three components and the second (b) the cumulative explained variance of the components

3.2 Survival Targets

As shown in Table 4 five survival targets were created. The targets which only used data sourced from the “followup” file (DF and EF) report events at the time of planned follow-up visits. This is different from the targets using dates reported by the other files, those are exact dates for the events.

The exact dates reported for patient death and graft failure by the “empfaenger” and “transplantation” respectively only represent the random variable T^*, as they are only assigned values, if an event was registered. This means that they are not providing censoring information on their own. The respective censoring dates were taken from the corresponding follow-up target and their last recorded follow-up date.

The following table contains a constructed example based on the data observed, which was used to build the survival targets.

Table 5 Constructed example of target construction inputs

ET Follow-Up	**Patient Death**			**Graft Failure**		
Last Follow-Up Date	Exact Death Date	Last Follow-Up Date	Event Indicator	Exact Failure Date	Last Follow-Up Date	Event Indicator
3290.0	[NA]	1103.0	0.0	[NA]	1103.0	0.0
[NA]	[NA]	[NA]	[NA]	999.0	[NA]	[NA]
3187.0	[NA]	[NA]	[NA]	[NA]	[NA]	[NA]
1334.0	[NA]	1096.0	0.0	[NA]	1096.0	0.0
[NA]	1105.0	1097.0	0.0	[NA]	1097.0	0.0

The constructed exempt above highlights challenges during the construction of the survival targets. For example, for the last patient in the table above an exact death date has been registered. In the DE target they would appear with $(T_{DE}, \delta_{DE}) = (1105, 1)$, while in DF it would appear with $(T_{DF}, \delta_{DF}) = (1097, 0)$.

From the shown excerpt data, the following survival targets would be constructed.

Table 6 Constructed targets from the input example

ET Follow-Up	**(EE)**	**Patient Death Exact**	**(DE)**	**Patient Death Follow-Up**	**(DF)**	**Graft Failure Exact**	**(FE)**	**Graft Failure Follow-Up**	**(FF)**
T_{EE}	δ_{EE}	T_{DE}	δ_{DE}	T_{DE}	δ_{DF}	T_{DF}	δ_{DF}	T_{FF}	δ_{FF}
3290.0	0	1103.0	0	1103.0	0	1103.0	0	1103.0	0
999.0	1	-----	-----	-----	-----	999.0	1	-----	----

3187.0	0	-----	-----	-----	-----	-----	-----	-----	----
1334.0	0	1096.0	0	1096.0	0	1096.0	0	1096.0	0
1105.0	1	1105.0	1	1097.0	0	1097.0	0	1097.0	0

This way of constructing the exact date targets results in the follow-up and exact date targets sharing their censoring information. The exact events however are overwriting the follow up date and latest value, if available. As was shown in Figure 4 9758 subjects are present in all three targets, not separating by exact and follow-up sources.

In the following figure two plots are shown. For the first plot (a) 500 subjects were sampled which had at least one of $\mathrm{C_{EE}}$*,* $\mathrm{T_{DF}}$ *or* $\mathrm{T_{FF}}$ *available and those dates then were plotted. The second plot (b) had another 500 subjects sampled, but only from those who had both* δ_{DF} *and* δ_{FF} *set to any value. Coloring was based on these values and the dates* $\mathrm{T^*_{DE}}$*,* $\mathrm{T_{DF}}$*,* $\mathrm{T_{FF}}$ *and* $\mathrm{T^*_{FE}}$ *drawn.*

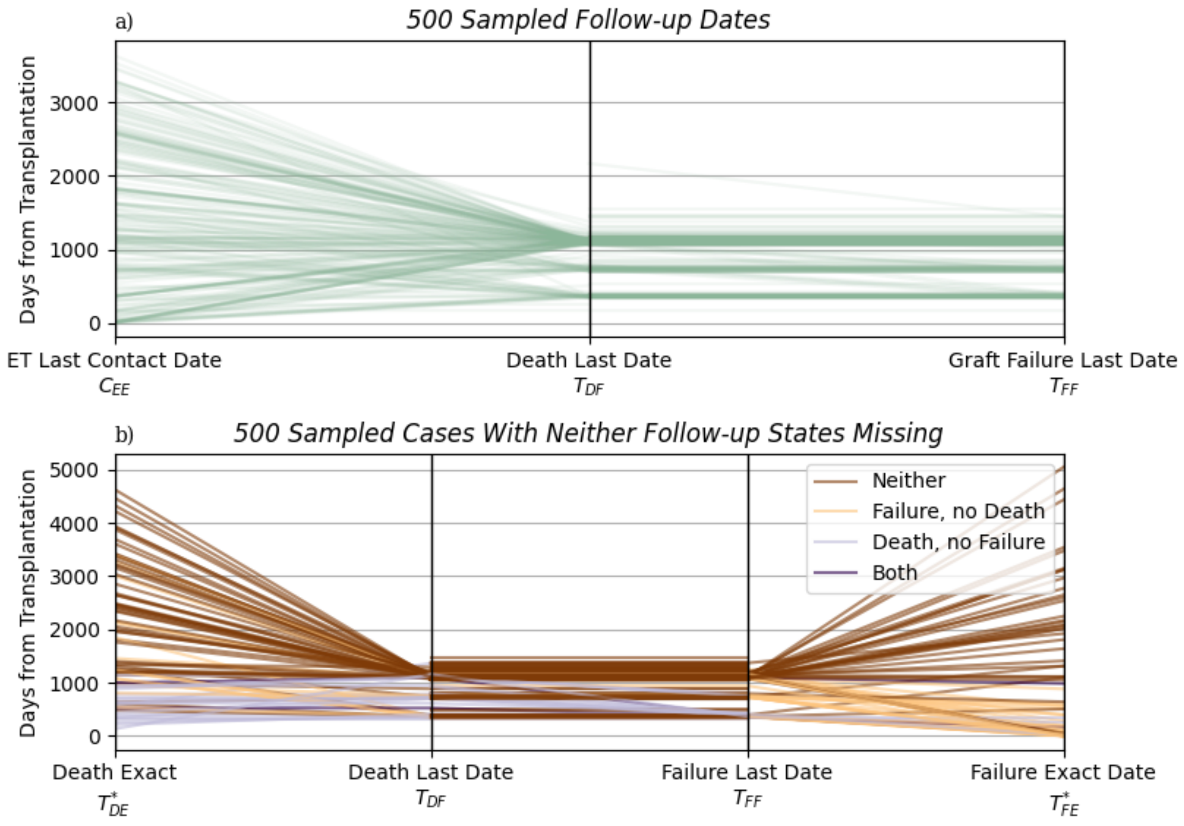


Figure 9 Parallel coordinate plot of follow-up dates relationships overall (a) and when a final state is set (b)

The latest follow-up times between the patient death $\mathrm{T_{DF}}$ and graft failure $\mathrm{T_{FF}}$ are mostly equal with the most common values being one, two and three years (Figure 9a). This pattern is not visible for the latest follow-up date reported by the ET in "transplantation" C_{EE}, where a wider range of dates is visible. In the second part of the figure (Figure 9b) only subjects, who had both targets failure δ_{FF} and death δ_{DF} set to either zero or one are shown. The latest follow-up dates ($\mathrm{T_{DF}}$, $\mathrm{T_{FF}}$) for the sample appear to be bounded to a maximum of three years. There are only very few patients who both registered with a graft failure and a death event ($\delta_{FF} = \delta_{DF} = 1$). Exact event dates ($T^*_{DE}, T^*_{FE}$) after three years have their corresponding latest follow-up date (T_{DF}, T_{FF}). If there was an exact event date (T^*_{DE}, T^*_{FE}) after three years, it did not register as an event in the follow-up data (δ_{DF}, δ_{FF}).

The following table shows the cross counts between the event indicators in the FE and DE targets to the FF and DF target. Only subjects who had both event indicators not missing in the FF and DF target were counted.

Table 7 Contingency table of exact dates and last follow-up states excluding missing follow-up states

Patient Death Event Indicator $\boldsymbol{\delta_{DF}}$		**0**	**0**	**1**	**1**
Graft Failure Event Indicator $\boldsymbol{\delta_{FF}}$		0	1	0	1
Patient Death Exact Death Date T^*_{DE} set	Graft Failure Exact Failure Date $\boldsymbol{T^*_{FE}}$ set				
False	False	10918	153	84	10
False	True	789	755	2	8
True	False	1242	51	403	27
True	True	376	209	115	50

Almost 11000 patients are censored in targets DF and FF and censored when looking at the exact date targets DE and FE (Table 7). The accuracy of the last follow-up value compared to the exact dates is 79.8%.

3.2.1 Patient Death

Inspecting the dates of previous follow-up visits, in contrast to only the latest date T_{DF}, shows that values were recorded only about once a year, with some exceptions (Figure 10a). These exceptions mainly appear when looking at the list of all follow-up dates and not only the last one. While some patients have more than three years of follow-up time and more than three visits, 99.3% of patients have three or less visits registered and 59.0% of them exactly three. Only 7.7% of patients were not censored. On the DF (T_{DF}, δ_{DF}) and DE (T_{DE}, δ_{DE}) targets survival functions based on the Kaplan-Meier estimator with a 0.95 confidence interval with the Greenwood formula and a single log transformation were constructed (Figure 10c).

The Kaplan-Meier estimator based on the follow-up dates (T_{DF}, δ_{DF}) stays almost constant at almost 100% until it reaches the one-year mark, then it drops almost instantaneously by 2-5%. This behavior of an almost constant level with drops of the same height at multiples of a year is repeated until the three-year mark. After the drop of the three years mark, the survival estimate constantly sinks to 70%, where it remains constant with a high confidence interval until nine years after the transplant. Based on the exact date data for patient death (T_{DE}, δ_{DE}), the red Kaplan-Meier (Figure 10a) estimator starts with a strong exponential decline which becomes slower over time, until the decrease remains constant after one year. After three years, simultaneously with the follow-up date-based estimator, it shows another strong exponential decline which continues for 14 years when the survival drops to zero. This suggests that only events happened after a certain time and no censoring. The follow-up sourced event indicators in the death follow-up set (δ_{DF}) show missing values as last values (Figure 10b). They indicate cases, where there is a missing value at the last visit reported. Those were removed from DF.

The event ($\sum \delta_{DF}, \sum \delta_{DE}$) and censor indicator ($\sum I(\delta_{DF} = 0), \sum I(\delta_{DE} = 0)$) counts over time are relevant for the survival function estimate. They are shown as cumulative counts in Figure 10d). Events T_{DF}^{*} and censorings C_{DF} in the follow-up only DF target almost exclusively are registered at the one-, two- or three-year mark. A huge number of almost 10000 censorings occur at the three-year mark. Because for the exact targets DE (and FE), their censorings C_{DE} are taken from the corresponding follow-up information T_{DF}, they share almost the same cumulative count plot. The visible decrease between in the counts of censorings is caused by subjects who have an exact death date $\delta_{DE} = 1$ but are censored in the follow up table $\delta_{DF} = 0$. Exact dates for deaths also occur after the three-year mark, but the event rate seems to be highest shortly after the transplant.

For the first plot (a) in the following figure the dates for which a patient death indicator was reported (δ_{DF} at previous visits) are plotted with a density estimate and jittering points for the date of each patient visit (T_{DF} at previous visits). The second plot (b) indicates the number of follow-up visits where a death indicator was registered separated by the final value of δ_{DF}. In the third plot (c) the Kaplan-Meier estimates for the DF and DE targets with a 95% confidence interval based on a logarithmic transformation are shown. The final plot (d) shows the cumulated counts of events and censorings across both targets.

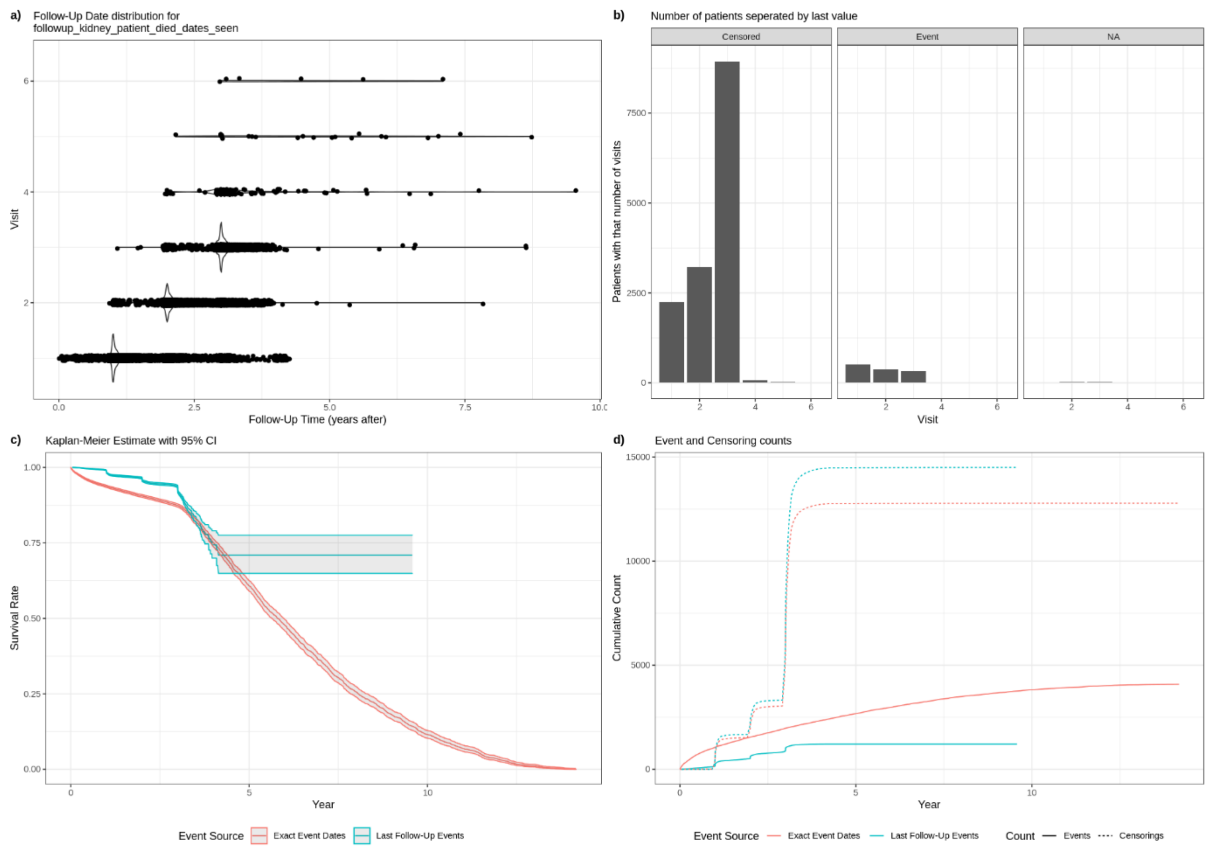


Figure 10 Time of follow-up visits where death event indicators were registered for DF (a), at which visit the final indicator value was registered for DF (b), the Kaplan-Meier estimates with a 95% confidence interval (c) for the DE and DF and cumulative events and censorings (d) in DE and DF

Almost no censoring information was available beyond the three-year boundary. Therefore, the data for survival prediction was limited to a three year and 30-day horizon. Patients with censorings or events after this horizon were considered censored at three years and thirty days.

$$T_{art.cens} = \min(T, 365.25 \cdot 3 + 30)$$

$$\delta_{art.cens} = \begin{cases} \delta, & T < 365.25 \cdot 3 + 30 \\ 0, & T \geq 365.25 \cdot 3 + 30 \end{cases}$$

Keeping all patients is important as just removing them changes the survival estimate and would equate to left truncation in the dataset. With artificial censoring, the survival estimate is just a time limited view of the full estimate (Figure 11a, c). A visible difference is in the cumulative censoring counts at the last time point (Figure 11b, d). The count has an instant jump, so that the absolute number of subjects is maintained. The final artificially censored follow-up death date target ($T_{art.cens,DF}$, $\delta_{art.cens,DF}$) was available for 15712 subjects with 7.0% experiencing events and the artificially censored exact death date target ($T_{art.cens,DE}$, $\delta_{art.cens,DE}$) was available for 16865 subjects with 11.9% experiencing an event.

The artificially censored patient death targets DE and DF are plotted in the following figure. The Kaplan-Meier estimates and cumulative counts were constructed in the same way as the full estimate (Figure 10c, d).

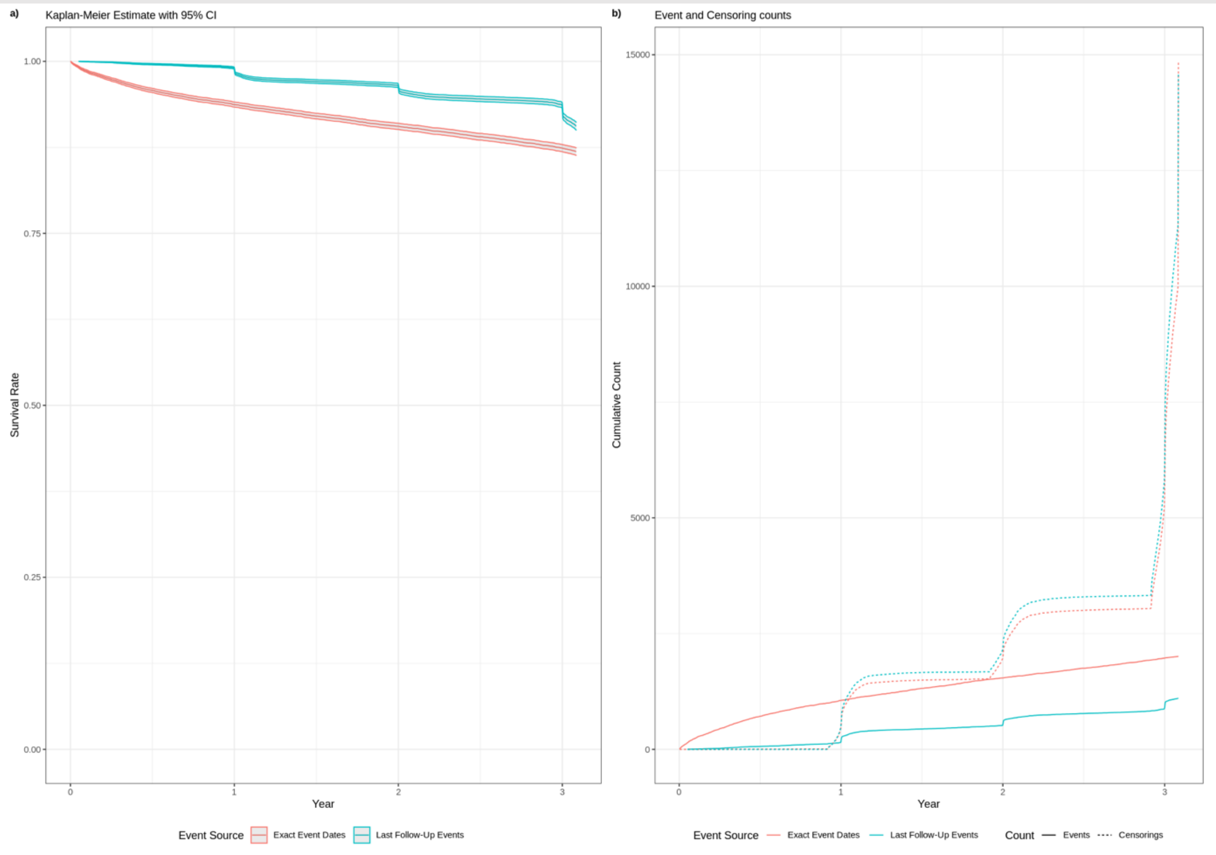


Figure 11 Kaplan-Meier estimates for the artificially censored DF and DE targets to three years and 30 days (a) and the corresponding cumulative counts (b)

3.2.2 Graft Failure

The graft failure targets share observed patterns with the patient death events. For the follow-up-based graft failure (FF) the first three years show the same curve as the death event estimate (DF). One notable difference is in the larger drop-off after these three years. For exact failure dates (FE) the main difference is at the time of the transplant. There is an immediate drop by almost 10% of the survival estimate. This is caused by unsuccessful transplants which might have been rejected or were never functional in the first place. These may be missing from the follow-up data and be a case of truncation in the follow-up datasets. Otherwise, the estimator behaves like the exact death date estimator, only offset by the drop-off at the start. There are no cases where there is no failure state set at a follow-up visit. This is different from the death observations.

The same plots as produced for Figure 10 are shown in Figure 12, but instead applied to the graft failure targets FF and FE. Please refer to the other figure for the construction method.

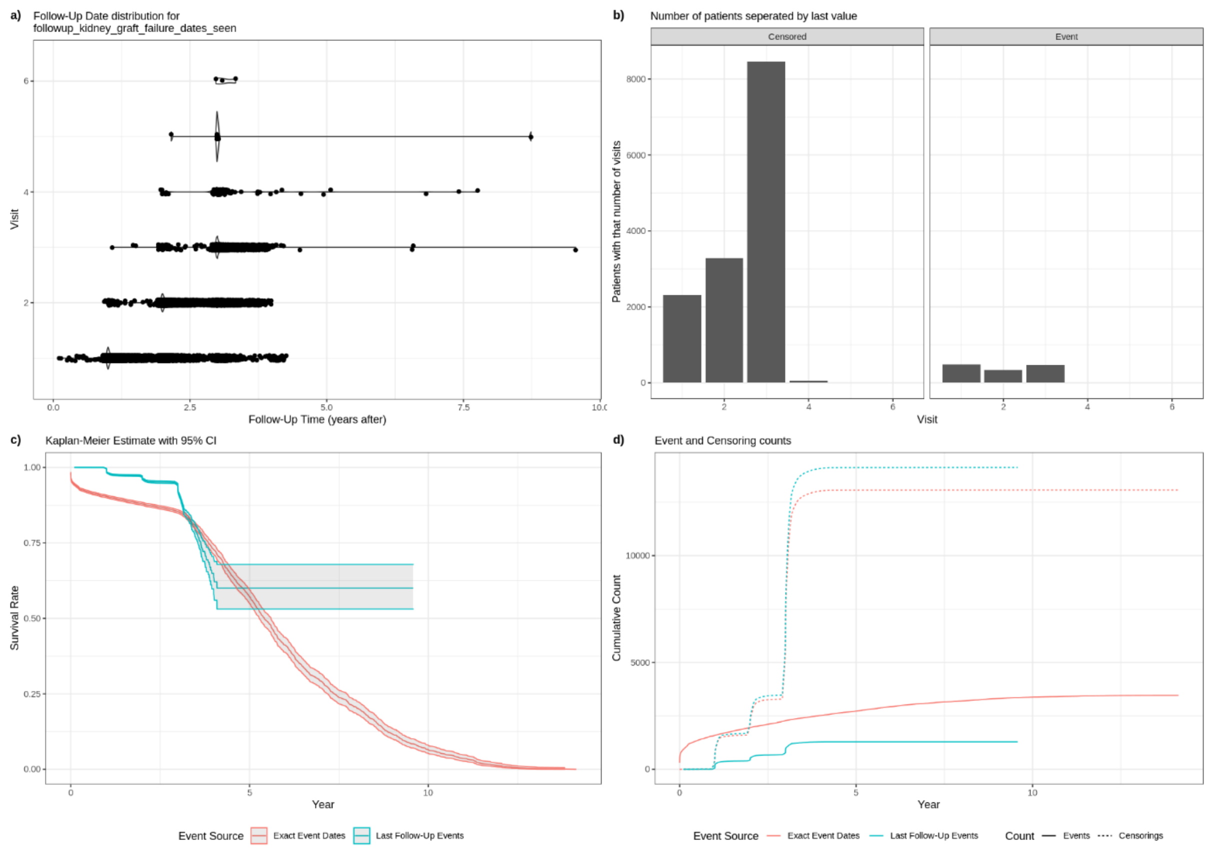


Figure 12 Time of follow-up visits where death event indicators were registered for FF (a), at which visit the final indicator value was registered for FF (b), the Kaplan-Meier estimates with a 95% confidence interval (c) for the FE and FF and cumulative events and censorings (d) in FE and FF

The same artificial censoring as applied to the death targets was applied to the graft failure. Due to the same pattern showing almost no censorings $\delta_{FF} = 0$ after three years being observed here. For the FF target ($T_{art.cens,FF}$, $\delta_{art.cens,FF}$) 15413 patients were available with 7.2% showing events. On the other hand, for the FE target ($T_{art.cens,FE}$, $\delta_{art.cens,FE}$) 16532 patients were available with 13.9% having an event.

To illustrate the effect of the artificial censoring, the next figure was constructed in the same way as Figure 11, but showing the Kaplan-Meier estimator and cumulative event and censoring counts for the graft failure targets FF and FE.

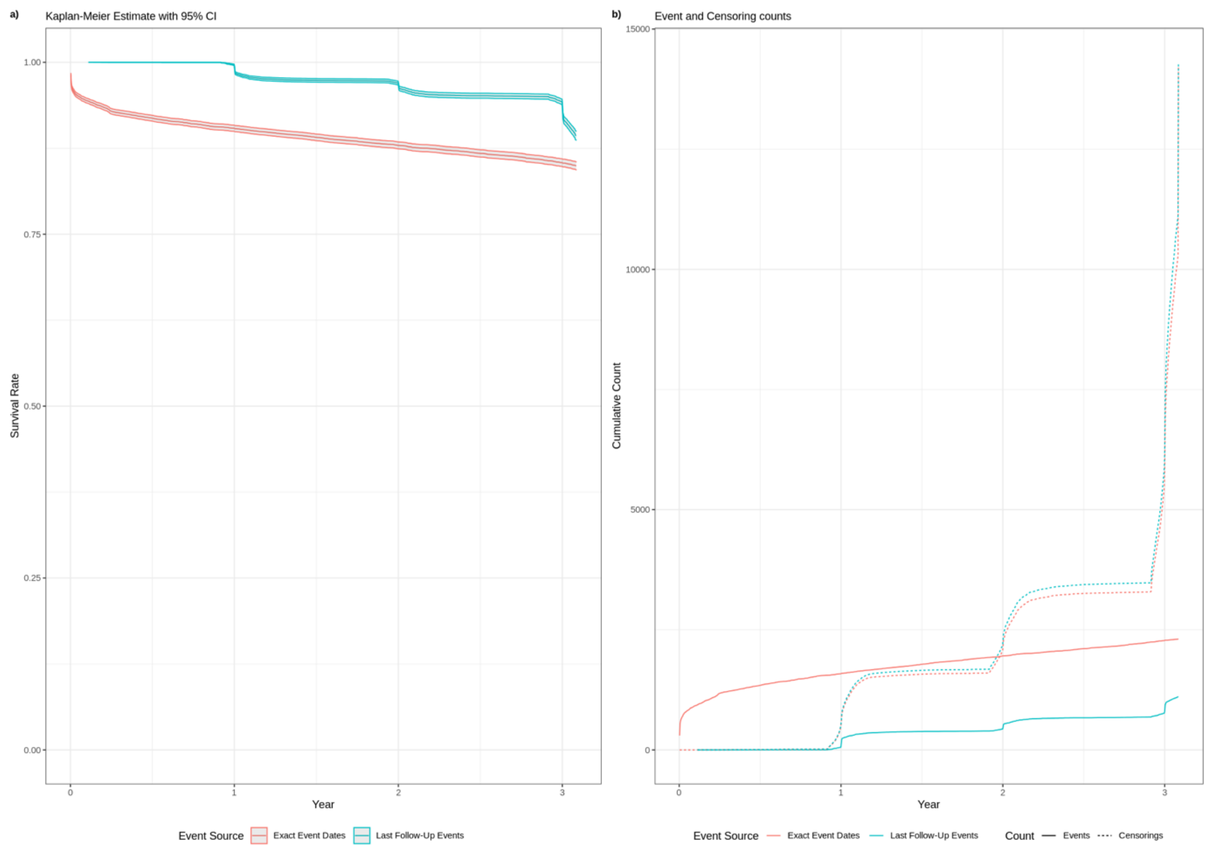


Figure 13 Kaplan-Meier estimates for the artificially censored FF and FE targets to three years and 30 days (a) and the corresponding cumulative counts (b).

3.2.3 ET reported follow-up

The combination of the exact failure dates T^*_{FE}, the death dates T^*_{DE} and the last contact date C^*_{EE} reported in the "transplantation" file (EE) shows a different pattern from the other targets. It has the immediate drop at the start caused by the failure dates. No yearly periodicity is visible in the survival estimate, but a less strong pattern compared to the other targets of the censorings is visible in the cumulative counts. After the immediate drop of the survival estimate at the start, the decline of the survival rate almost becomes linear after a short exponential decline over four years. However, after five years the hazard for a patient appears to increase and after eight years clearly decreases exponentially.

No artificial censoring was necessary, as both events and censorings occur over the entire time range. The pairs (T_{EE}, δ_{EE}) were available for 12017 patients. It was the smallest target, but it had the highest maximum survival times T_{EE}. The overall naïve risk was also the highest in this target with 29.4% experiencing an event.

For the constructed target from the direct ET follow-up dates and either death or failure dates (T_{EE}, δ_{EE}) the following figure shows the 95% confidence interval Kaplan-Meier estimator (a) and the cumulative counts (b).

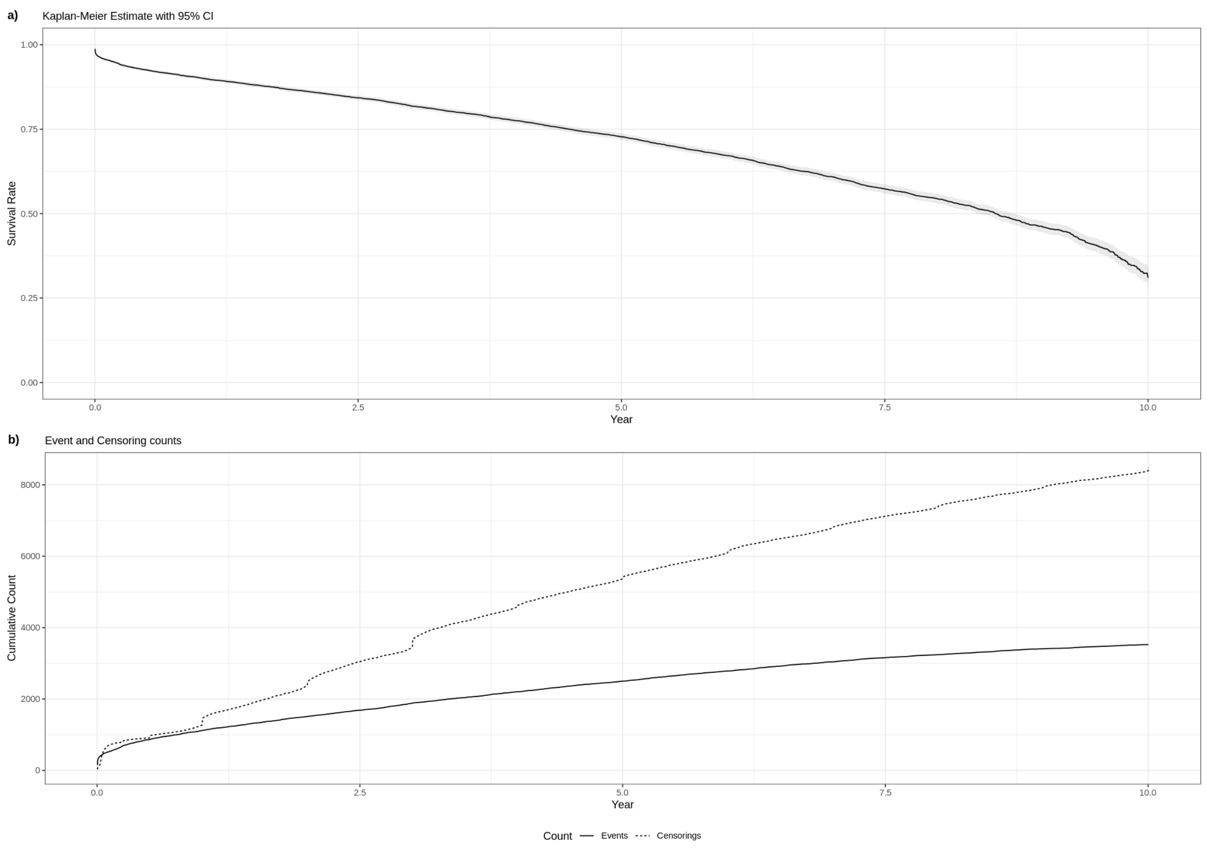


Figure 14 Kaplan-Meier estimates with 95% CI (a) for the EE target and the corresponding cumulative counts (b)

3.3 Comparing Model performance

An important factor for models is the run time. Caching for the preprocessing steps was enabled and the first model tested was the survival forest (“RangerForestSurvival”). This means its longest run time could be caused by needing to run the precomputing steps and the other configurations reusing its results. Timing results from the DE target are shown in Figure 15. The fastest models were the Kaplan-Meier model (“RCoxphNull”) and the AFT model with ridge regression (“IPCRidge”). Even though it is a Cox regression model with an additional elastic net penalty and has a higher maximum number of iterations (10000) by default, the “CoxnetSurvivalAnalysis” model is faster than the unmodified Cox Regression (“CoxPHSurvivalAnalysis”). The linear survival support vector machine (“FastSurvivalSVM”) shows a large range of training times. Due to the number of tested hyperparameters, the most configurations were evaluated for gradient boosting models (“GradientBoostingSurvivalAnalysis”).

As an example, the average training times for the model pipelines with the slowest, best performing, and fastest hyperparameter configuration across the folds for the DE is shown in the following figure. The vertical lines show the standard deviation across the three folds. The *training time data from the exact patient death target is plotted (a) as well as the number of configurations (b), which was shared among all targets.*

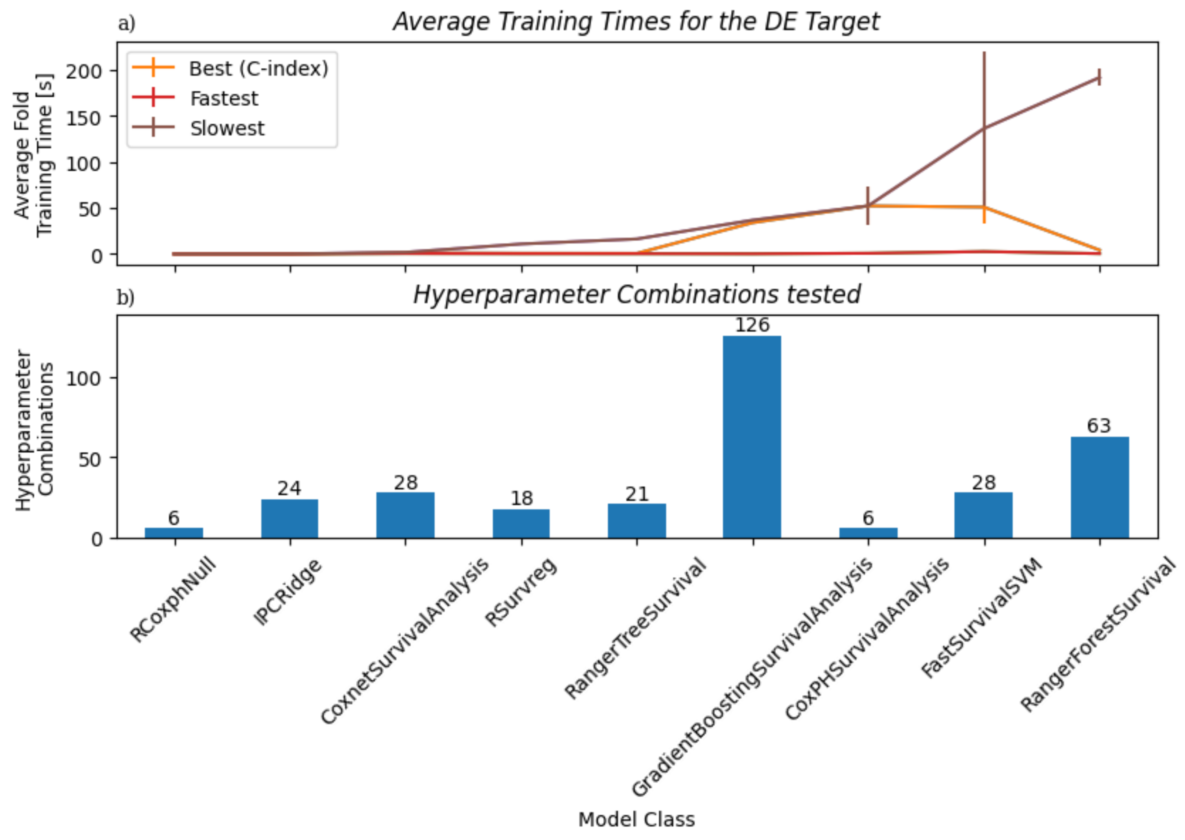


Figure 15 Average fold training times for the DE target (a) and the number of hyperparameter combinations tested for each model (b)

These results seen on the exact death date target translate to the other targets from the follow-up category as well. Comparing the slowest parameter configuration across all five targets, the ranking of the models stays similar. The “CoxPHSurvivalAnalysis” and “RangerForestSurvival” models are significantly slower on the exact date targets DE and FE. Because the EE target has less (T, δ) pairs available compared to the other targets, models are mostly faster or need similar time. Yet, Cox regression and the random forest are significantly slower on this dataset.

The training times for the slowest pipeline configuration of each model across all targets are shown in the following figure. Mean training time of the configuration refers to the mean across all folds. Ordering of the labels was based on the position on the FE target.

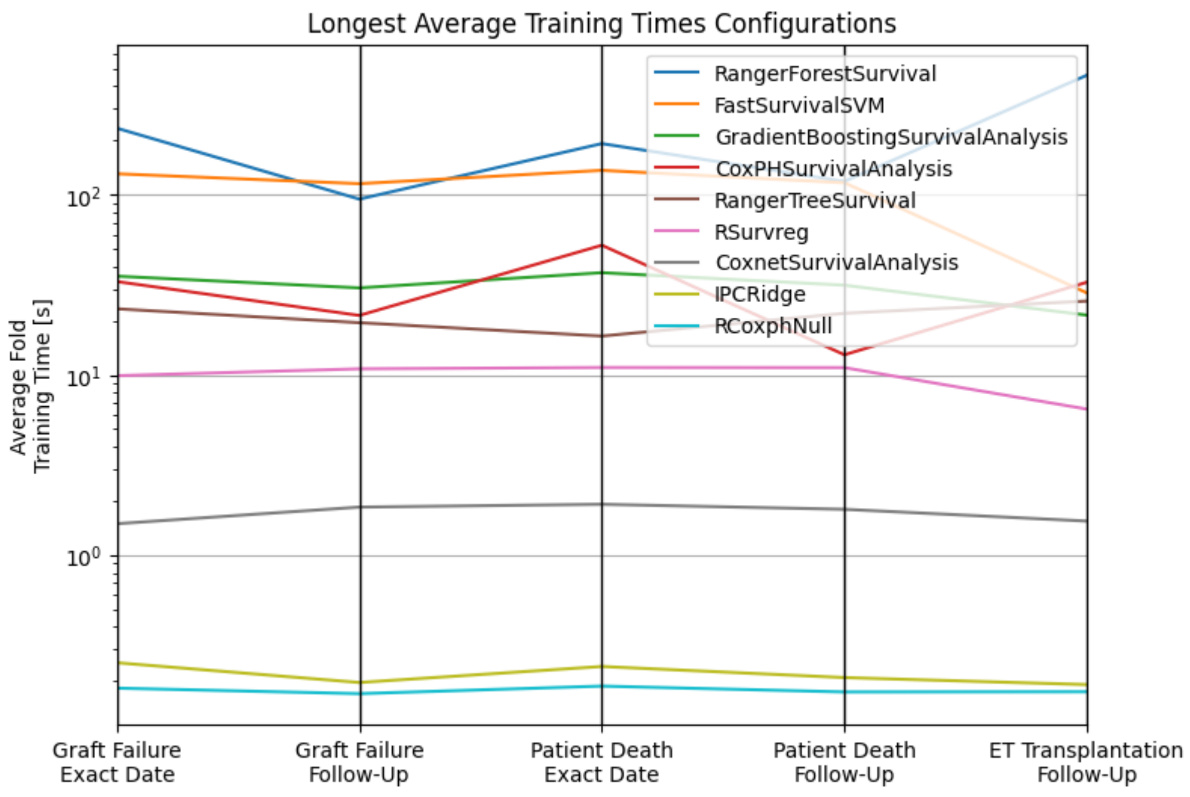


Figure 16 Average slowest configuration training times across all targets during grid search

Some hyperparameter configurations were unable to be trained or scored, because the training or scoring failed on one or more of the folds. This was the case for all five targets for the Cox regression using the feature agglomeration with 256 clusters. The AFT ridge regression model failed three times in overall with the alpha parameter set to 0.002479. The highest number of hyperparameter combinations failing was 39 for the survival regression AFT model "RSurvreg". If a certain dimension reduction and target combination had a failing model, in most cases, all three time-distributions Weibull, exponential and lognormal failed. Overall, 47 configurations of 1600 failed.

The most prevalent preprocessing options among the selected best for each model class, ignoring the selections for the Kaplan-Meier model, were no dimension reduction (9 times), feature agglomeration with 256 clusters (9 times), PCA with 256 components (8 times) and PCA with 128 components (7 times).

3.3.1 Prediction Scores

The risk scores usable for ordering subjects in accordance with their relative risk of failure is available for all models. It can be used to calculate a C-index score, IPCW C-index score, and an average cumulative, dynamic AUC over the subjects based on the relative risks. All models except the IPCW AFT ridge regression and survival support vector machines provide survival functions as predictions. If available, it was used to calculate an integrated Brier error as a loss. This integrated Brier loss was scaled to be comparable to the other scores.

3.3.1.1 Patient Death

Model performance on the exact death date (DE) target is better compared to the performance on the death follow-up target (DF). The ranking of the best models for the patient death targets seems to stay the same across all scores. An exception from this rule appears with the follow-up-based target (DF), where for the integrated Brier score the best model becomes the AFT survival regression ("RSurvreg").

For the following figure the prediction scores on the test partition (Test Scores) of the retrained best performing model configuration are plotted for the targets DE (a) and DF (b). The scores for the same model are connected. Ordering of the labels was based on the C-index.

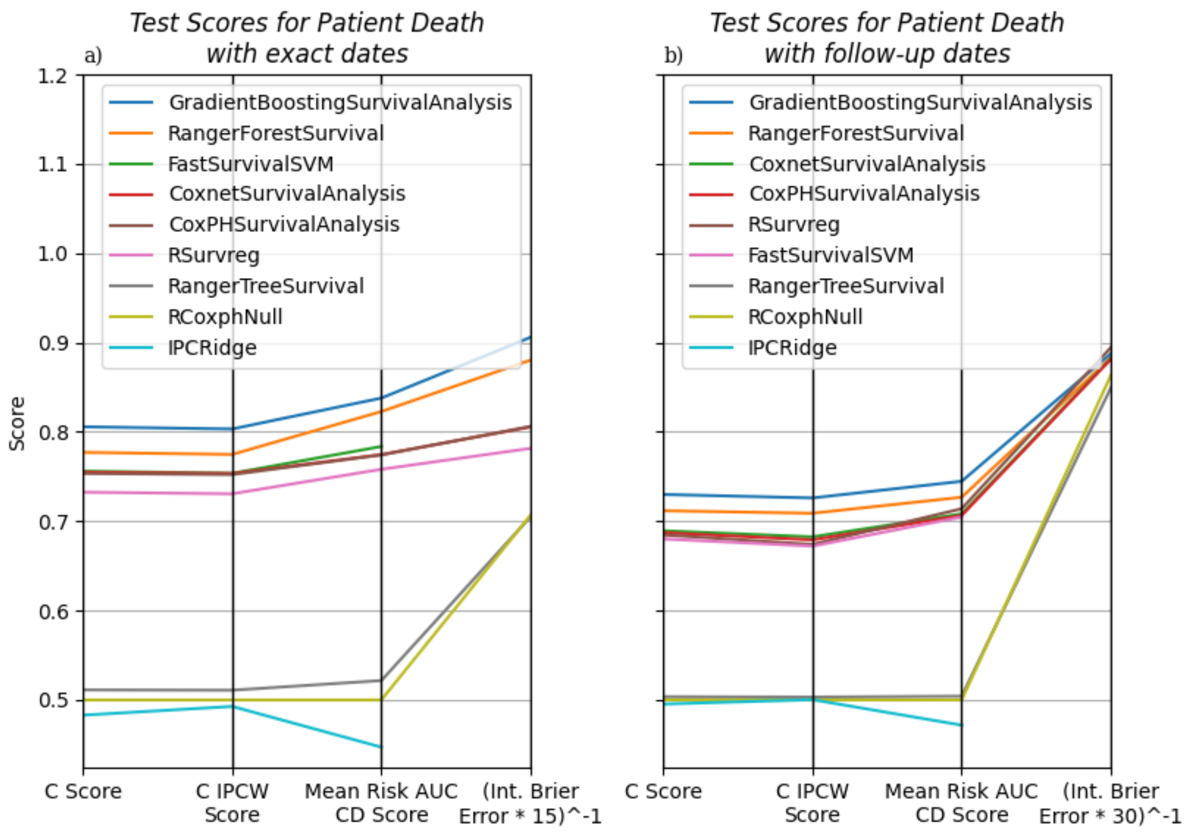


Figure 17 Test scores for the best models on the patient death targets DE (a) and DF (b)

Especially for the follow-up target (DF), the Cox regression, Cox regression with shrinkage, AFT survival regression, and the linear survival support vector machine have almost the same scoring results between 0.73 and 0.75 on the C-index. IPCW weighing of the test score has almost no impact for the exact dates, but slightly reduces the score on the follow-up target. The connection between the C-index and the cumulative, dynamic AUC is visible, as the scores are almost identical. Best scores are achieved by the gradient boosted model with a C-index of

0.73 and 0.81 on the follow-up and exact date target respectively. Shortly behind is the survival random forest with 0.71 and 0.78, respectively. For most models, the C-index score is between 0.73 to 0.81 on the exact date target. The performance of the Kaplan-Meier model with a C-index of 0.5 is to be expected, as it predicts the same relative risk for all tested subjects. The IPCW AFT ridge regression model and the survival tree are failing to fit the data. The best survival tree appears to almost make the same predictions as the Kaplan-Meier model. The IPCW AFT ridge regression model performs even worse than the baseline Kaplan-Meier model. It has a C-index below 0.5, which suggests that inverting its risk ordering would lead to better results, compared to the model output itself.

The cumulative, dynamic AUC on the test partition can be calculated with the risk scores and the cumulative hazard function scores if they are available. These variants over time are shown both for the best models on the exact date target (DE, a) and follow-up target (DF, b) in the following figure.

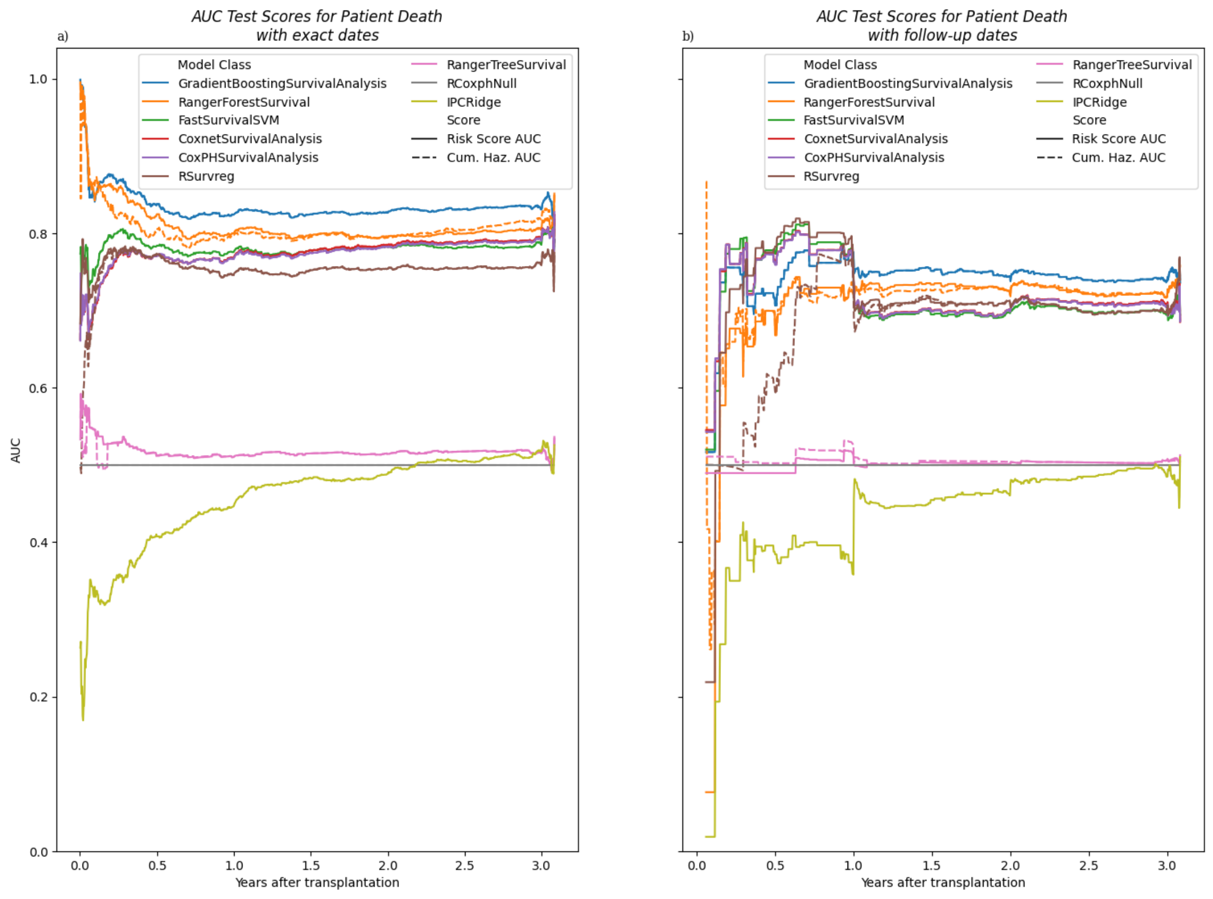


Figure 18 Cumulative, dynamic AUCs based on the risk scores and cumulative hazard functions of the best performing models on the DE (a) and DF (b) targets

Most scores start at a height close to the mean of the scores and then show a peak and drop back to a similar level. This might be caused by the small number of subjects with events occurring directly at the beginning. Scoring based on the cumulative hazard functions deviates only slightly from the risk scores for the survival random forest but overlaps the single risk score curve after one year again. After three years it starts to deviate again. The curve for the IPCW AFT ridge regression model stays below 0.5 again. This confirms that its risk scores are reversed. More deviation between cumulative hazard and single risk scores AUCs is visible for the follow-up target (DF). This is especially the case for the AFT regression models before the first-year mark and a little bit again for the random forest survival model. Before the first year mark the ordering of the scores is quite different from the average model performance. It is important to note that the average is weighted based on the change of survival function and therefore this part, where the survival function stays almost constant, is almost irrelevant for the average AUC. For this first year, models previously close to the Cox regression perform better compared to the other models. All models show a high variance of the AUC at the end and the beginning.

3.3.1.2 Graft Failure

The survival target for patient death and graft failure are overall very similar. Test scores for the exact failure date target (FE) are slightly below the corresponding model scores for the exact death date target (DE). However, two notable differences are the slightly better performing survival tree and that the survival random forest drops in the ranking of the integrated Brier loss to a similar error as the survival tree on the exact target (FE) of 0.11. Prediction performance on the follow-up dates is higher for the graft failure compared to the follow-up target for the death date. The “IPCRidge” model also fails to fit to the data with C-index scores of 0.47 and 0.46 on FE and FF, respectively. The gradient boosting model is the best model for this target as well with C-index scores of 0.78 and 0.77.

For the following figure the scores on the test partition of the best model configurations retrained on the entire training partition are shown just like in Figure 17, but on the graft failure targets FE (a) and FF (b).

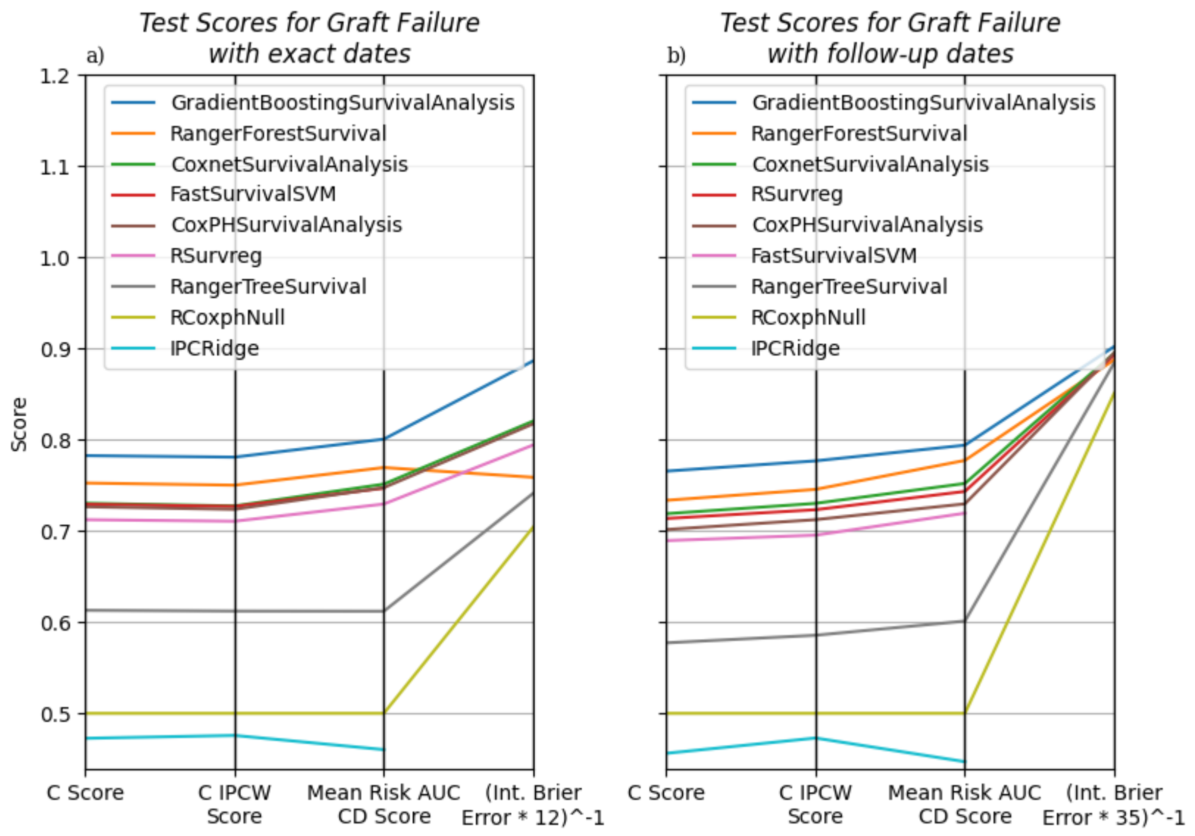


Figure 19 Test scores for the best models on the graft failure targets FE (a) and FF (b)

The most stable over time AUC score is produced by the exact failure date target (FE, Figure 20). All models appear to struggle modeling the strong decline immediately at the beginning. The targets show a higher score at exactly three years, where most events near the end are registered, shortly followed by many censorings at the end. For the FF target, there is a high variance of the scores after one year. It is important to note that almost no events are present in this target until the first year. This seems to lead to the estimation of the score before the first event to stay at the level of the first achieved score, which like the other targets shows a high variance.

In the following figure the same time-dependent AUC data based on the risk scores and cumulative hazard function predictions plotted in Figure 18 are shown for the FE and FF targets.

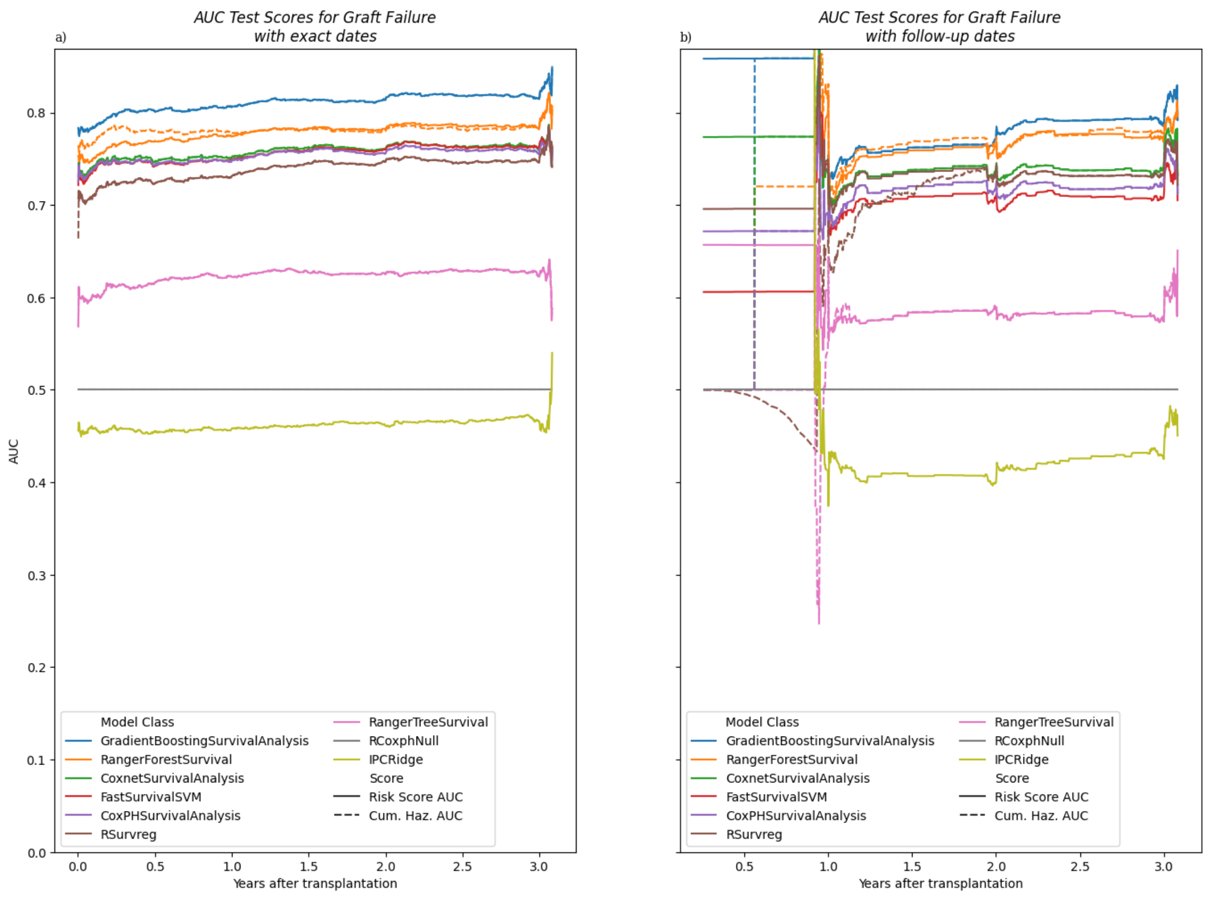


Figure 20 Cumulative, dynamic AUCs based on the risk scores and cumulative hazard functions of the best performing models on the FE (a) and FF (b) targets

3.3.1.3 ET reported follow-up

The ET “transplantation” constructed survival target EE has a much longer time range compared to the other targets, yet similar patterns in the model test scores are visible. The Cox regression models, the survival support vector machine, and “RSurvreg” achieve similar scores, like was already observed in the other targets. This target has the worst scores of all targets. On the integrated Brier error, the Kaplan-Meier model and the survival tree achieve almost the same score. The best model is again the gradient boosted model with a C-index of 0.74.

For the target EE, the test scores are shown in the following figure, constructed in the same way as Figure 17 and Figure 19.

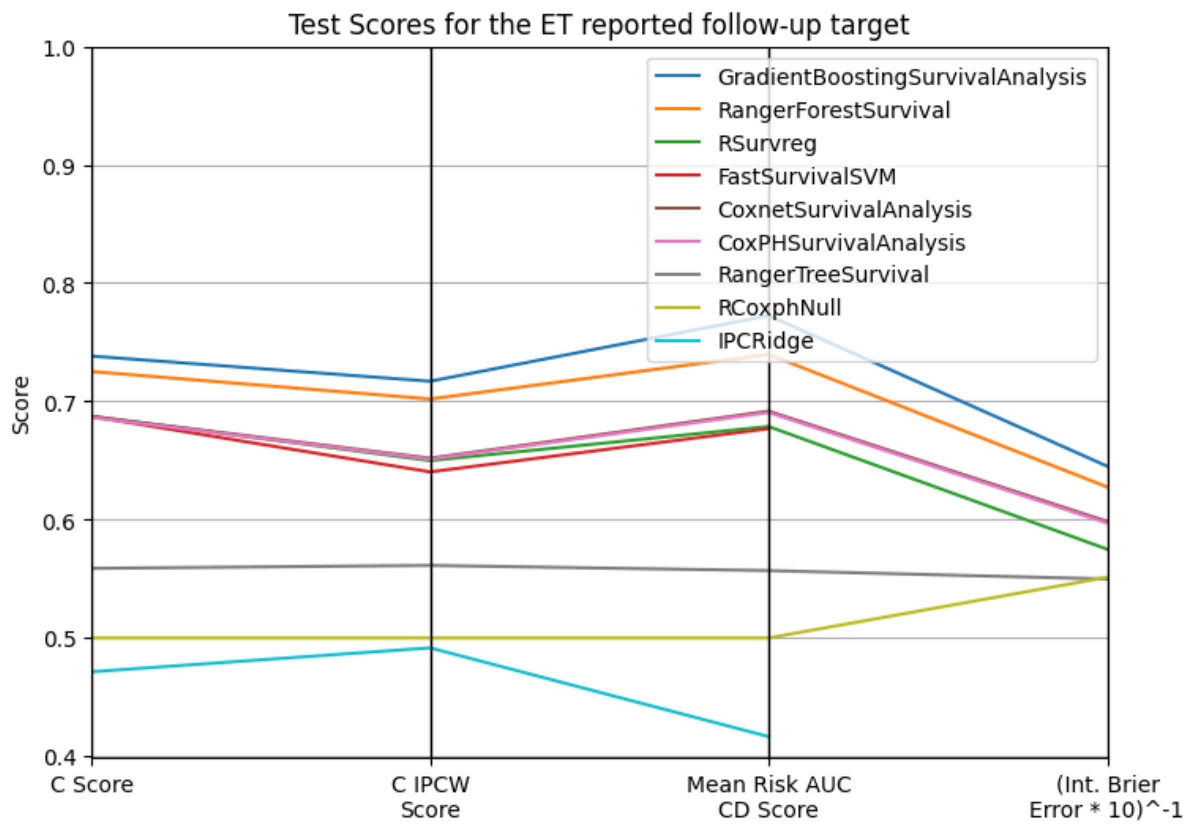


Figure 21 Test scores for the best models on the ET reported follow-up target EE

AUC scores are mostly stable through time, but the models with behavior close to the Cox regression stay almost identical and drop after four years. The linear survival support vector machine and AFT survival regression model stay close together while dropping slightly below the Cox Regression models.

In the following figure the time-dependent AUC for the test partition based on the risk scores and cumulative hazard function predictions is shown for the ten-year ET sourced follow-up target (EE).

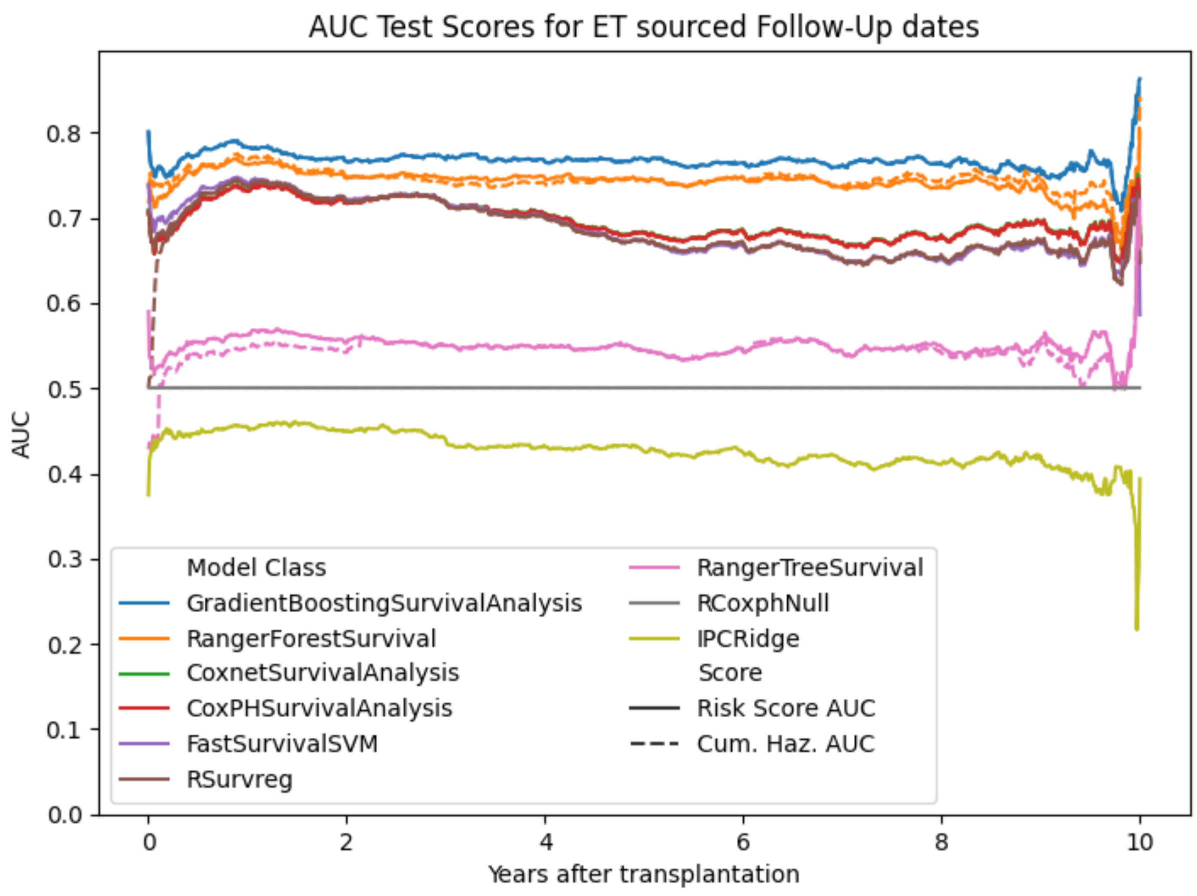


Figure 22 Cumulative, dynamic AUCs based on the risk scores and cumulative hazard functions of the best performing models on the EE target

4 Discussion

Results of this research project include the processing of the registry data, the targets, and the fitted model pipelines. These results can be further inspected and future potential to resolve challenges encountered identified.

4.1 Data and Preprocessing

Most of the work hours for this research project was spent on preparing the registry data. While there already is an export for scientific research, the necessary joins of the datasets, especially within files, with unclear matches made the development of the data preprocessing pipeline difficult. For this dataset, a pipeline with self-checks, continuous integration and reporting should be implemented to reduce the potential for data integrity problems. Because the transplantation registry does not provide a clear guide on how to cite it for research projects, it is difficult to find publications using its data. The challenges encountered by the author are general difficulties in using the data and might be fixed with updates to the export tool. Pseudonymization of some date columns also sometimes leaks the offsets for the pseudonymization of the date columns.

While some studies use slightly larger sample sizes of 30.000 with 50 features (Ayers et al., 2021), 50.000 with 40 features (Naqvi et al., 2021) or 60.000 with 100 features (Kantidakis et al., 2020b) for machine learning applications on transplantation data, they usually source their data from the UNOS registry, which is larger compared to the German registry. With a differing data base and a different region modeled, the German registry should however also be considered for further research. The cited research papers all also performed manual feature selection for clinical importance. This step was skipped in this work to test performance across all features, but that might have introduced useless features, increased variance, and features leaking target information leading to an overestimation of performance.

Among machine learning for transplantation studies, the use of survival machine learning is not entirely common. It appears the use of regression models and solely looking at the immediate failure probability or the survival probability after a fixed time still are very prevalent, but a thorough literature review might be necessary to establish measures of usage.

The goal of the scikit-survival library to adopt survival methods for the scikit-learn ecosystem is admirable, as it enables the support of feature preprocessing methods and integration of models into existing systems, but the instability of the library lead to problems during this project. In addition to new models, more preprocessing options specific to survival analysis might be introduced in the future, integrating the scikit-learn application programming interface (Buitinck et al., 2013).

4.2 Survival Targets

The follow-up-based targets behave more like interval censored data, and it is easy to see how their predictions are biased, especially for the time before the first follow-up. A future research project might look at methods to correct the bias caused by such a data pattern and how correction methods might be employed. For models like survival random forests and survival trees, which internally construct Kaplan-Meier estimators, smoothing methods might improve performance. Smoothing methods for the Kaplan-Meier estimator are well researched. An example is smoothing with Bèzier curves (Kim et al., 2003). They could be employed for survival trees, in which only a few instances are present in a leaf.

For the graft failure targets, the inclusion of immediately failing transplants might be incorrect, as this drop is not seen in quality reports for example of the DSO for 2019 and the estimate is lower compared to the report (Rahmel & Wadewitz, 2020). The EE target on the other hand seems to match expectations when comparing it to general predictions made for kidney transplant survival. A potential source of bias present in this target is caused by the last contact dates for censoring being used together with event detection which happens outside these tracked contacts. When comparing the last contact dates and the event dates, events are registered on average 154 days after the last follow-up date. Death dates are registered on average 501 days after the last contact and graft failures 37 days before the last contact.

Lesko et al. argue that captured events, which are registered even without follow-up contacts, and measured events should be treated differently for censoring. The exact date targets DE and FE have captured events, while the follow-up target DF and FF use measured events. For the ET sourced follow-up information in the target EE a mix of both seems to be present. They showed that for captured events, not the last contact date, but rather use a lost-to-follow-up (LTFU) definition should be used to avoid estimate bias. A LTFU application extends the censoring dates by the time window used to define when patients are LTFU and censors events at the time the LTFU time is reached if the event occurs after this window (Lesko et al., 2018). This correction might be extended for mixed targets like the EE and implemented like previously mentioned as a transformer class for scikit-survival and easily integrated in model pipelines.

4.3 Scoring

The C-index is the most commonly used score for survival analysis and closely connected to the other metrics. In this research project, the differences between the scoring methods were small, but the additional challenge of estimating the censoring distribution can be problematic, as the censoring distribution estimation needs to be applicable to the data used to calculate the scoring. This is not a problem for most applications, where the study is over a fixed time with events and censors evenly distributed through time. However, the application of cross-fold evaluation, where subsamples are taken, can lead to the mismatches between the time ranges. In the scoring method used for the test scores and cross-fold evaluation the test subjects were limited to the time range of the training data. This might be considered invalid, as the goal of the evaluation of test data is an estimate of the model performance on previous unseen data. For test scoring, this ignored around 1000 of approximately 5000 subjects in the test partition. This additional work however may be worth doing, as the weighted scores are comparable between studies (Uno et al., 2011).

While the time dependent AUC of the cumulative hazard and the Brier scores are not available for all models, they can provide insight into the model performance over longer time periods and how it may deteriorate. AUC scoring sometimes failed for the cross-fold evaluation, resulting from an exception within scikit-survival probably caused by edge cases where the true positive counts are zero, as their corresponding calculation throws the exception. Future research questions might include how censoring patterns and dependent censorings impact the scoring methods.

4.4 Model Inspection

Depending on the model architecture, different levels of inspection are possible for each model. Due to time constraints, the evaluated parameter spaces can be considered limited overall. In the field of deep learning, successive halving grid searches have shown potential for a faster optimization while sampling from parameter distributions (Li et al., 2018). They incrementally assign resources to promising model candidates while at the same time using resources to explore the parameter space further. Over time, less resources are spent on exploration and more resources on the best candidates. For models using a gradient descent optimization, the resources are iterations. For other model types, instances, or number of contained estimators can be used as resources. Halving grid searches with instances could be applied to survival models. However, a minimum number of subjects across the time range needs to be provided. The estimation of a minimum sample size might also be used for other model hyperparameters like subsampling and minimum number of subjects in tree leaves.

During follow-up visits, time varying features are measured. There are further adaptions of models to incorporate these (Zhang et al., 2018) and they could improve prediction power, especially over longer time ranges.

Only a simple missing value imputation was used. With caching of intermediate results during optimization, more advanced methods like the MissForest (Stekhoven & Buhlmann, 2012) method might improve results at a low computational cost.

4.4.1 AFT Modelling with IPCW and Ridge Regression

The models based on the class “IPCRidge” showed worse performance than the Kaplan-Meier model “RCoxphNull” with C-index scores below 0.5. This is caused by the scikit-survival model not correctly communicating the property that some models return prediction scores on the time and not the risk scale when they are part of a pipeline[6]. The same bug affected the gradient boosting model with the ‘IPCWLS’ error. It would also have affected survival support vector machines, where the rank target ratio would not have been set to 1, but these configurations were not tested in this project. The “IPCRidge” model and hyperparameter optimization was retrained on the EE target with a workaround against the library bug.

The best model with this workaround achieves an average C-index score across the test-folds of 0.55 and dimension reduction using feature agglomeration with 128 features and the smallest alpha tested of 0.002479. An explanation of the low performance compared to the AFT survival regression might be the use of the IPCW, as it discards censored subjects and their feature values entirely. For further research, IPCW might be evaluated for different censoring patterns.

4.4.2 AFT Survival Regression

The survival regression with different residual distributions was implemented in “RSurvreg”. All selected models for the targets except the EE target used the PCA dimension reduction with 128 components. The EE target AFT survival regression model used the feature agglomeration with 256 clusters. The feature agglomeration with 256 clusters behaves like no feature reduction, as most clusters will contain 1 feature. For the exact date targets (FE, DE) and follow-up target for patient death (DF) the best performing distribution family was the

[6] This issue is tracked here: https://github.com/sebp/scikit-survival/issues/324

lognormal distribution. The best model from this class on the EE target used an exponential distribution and for the FF target a Weibull distribution was chosen.

Predicted survival functions for the "RSurvreg" class of models are made by using the quantile function of the logarithmic residual distribution together with the linear predictor. The survival function therefore is not directly defined for time points, but for a survival fraction where a time can be defined, when this fraction would be reached.

As can be seen in Figure 23, for the exact date graft failure target FE, an immediate drop off to an almost constant level is predicted. The targets DE and EE share a similar starting behavior but deviate after the first year. For the entire three years the DF and FF target prediction follow the same line. Even though targets behaved similar, they did not necessarily share the distribution family. The distributions are scaled to match the data and different distributions may show similar behavior when scaled correctly.

In the following figure the predicted survival functions for each subject constructed by taking the mean across the entire set of subjects used for each target are shown. This plot has been limited to 25 years of predictions. The gray line shows the three-year-mark.

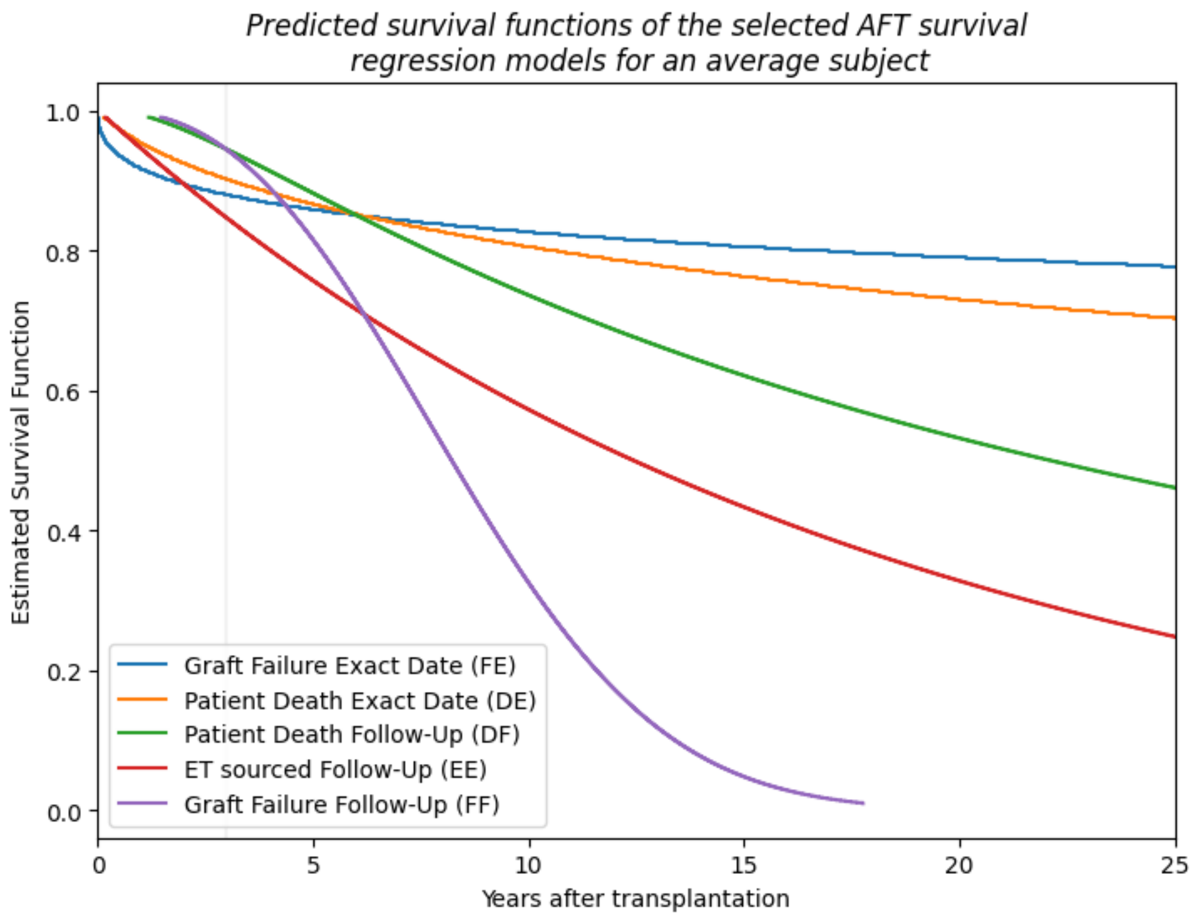


Figure 23 Prediction from the selected linear AFT models with different residual distributions

4.4.3 Cox Regression

On all targets except the graft failure follow-up target, the best performing preprocessing was the PCA with 256 components, the highest number of components tested. The models profited from the higher availability of data and as suggested by the explained variance plot, with less components too much variance is lost. The Cox regression appears to need the PCA transformation as compared to the feature agglomeration with 256 clusters, which mainly uses all features directly. For the FF target, the best performing model was the PCA with 128 components. On the folds, mean AUC scoring failed internally in scikit-learn for this configuration, but over the test data the mean AUC could be calculated. This could have been caused by the late occurrence of the first failures and the time ranges being estimated from the entire training partition, as the training folds are not available during scoring and the entire training partition giving a time range, for which scoring with the AUC is not possible. The Cox Regression uses the $x_i\beta$ results as the risk scores, which as previously mentioned are the logarithmic ratio between the hazard of the subject i and the baseline. Because each cumulative hazard function and survival function are scaled by this risk score, there is almost no difference between the AUC scores from the risk score and the cumulative hazard function. The exception from this is the constant start of the FF target. This could again be caused by differences in the training and testing partition on where the first events occur. The baseline hazard and survival function on the other hand can be defined at points before the first subject in the test partition.

When looking at the baseline survival functions for the Cox regression model, it appears that the baseline model for the DF target is not following the follow-up structure. Its coefficients however are larger compared to the other models, leading to stronger scaling of the survival baseline function. This is caused by it being the only model which only uses 128 components for the PCA instead of 256. The baseline in the Cox regression by scikit-learn in "CoxPHSurvivalAnalysis" is estimated for the features being the mean.

For the following figure the baseline survival functions for the best performing Cox regression models are shown on all targets (a) and the predicted survival function for a random subject from the dataset (b).

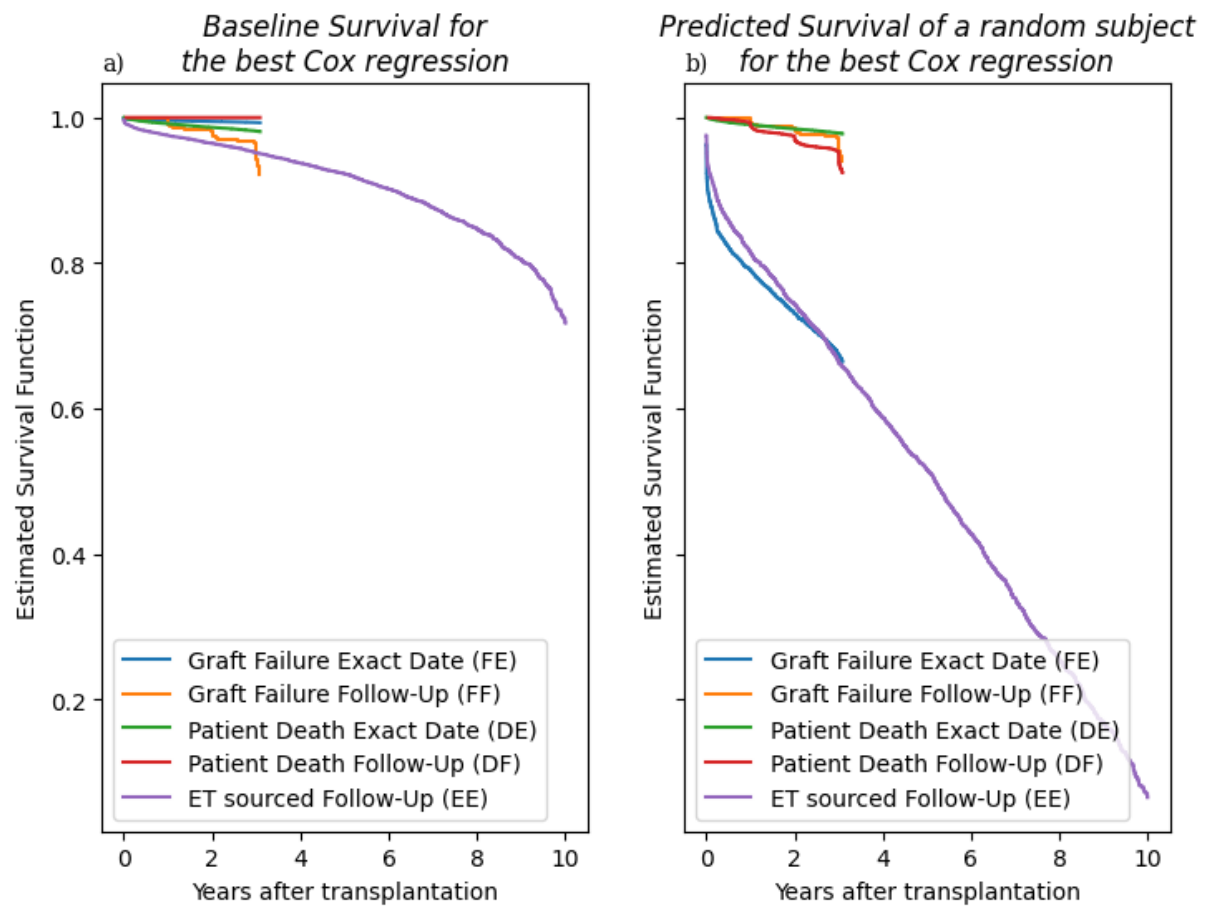


Figure 24 Baseline survival functions and random selected prediction for the best Cox regression models

The Cox Regression with elastic net penalties shows performance very close to the unmodified Cox regression. Shrinkage has the goal to reduce overfitting when applied with the right intensity, while too much shrinkage leads to a worse fitting mean model equivalent to the Kaplan-Meier null model. Too little shrinkage would lead to a Cox regression model behaving in the same way as a Cox regression model with no shrinkage. The results suggest this might be the case.

For the “CoxnetSurvivalAnalysis” models, 256 component PCA was selected for both patient death targets with a L1 ratio parameter of 0.1 and 0.4 for DE and DF, respectively. No dimension reduction was selected for the ET sourced follow-up data and a L1 ratio of one. The graft failure target with exact dates FE used feature agglomeration with 256 clusters and a L1 ratio of 0.7, while the graft failure target with follow-up dates FF again used a PCA with 128 components and a L1 ratio of 0.1. With L1 ratio of zero a L2 shrinkage is used and with the L1 ratio set to one, a L1 ratio is used. The testing of the values 0, 0.4, 0.7, and 1 is reasonable as they test enough different combinations of the different shrinkage methods.

Automatic selection of the alpha value as described is not implemented in the scikit-survival implementation. It uses the latest alpha in the solution path which results in the selection of the smallest alpha determined by the algorithm. This leads to the same behavior as having almost no shrinkage. The selected L1 ratios therefore also do not influence performance as almost no shrinkage is applied. This model should have a hyperparameter which refers to the alpha position, so that each possible value can be tested at a hyperparameter optimization or the described leave-one-out estimate for scoring. An alternative would be to determine the regularization path before hyperparameter optimization and evaluate each option, like post-pruning trees. While the following selection of the parameters in Table 7 would be invalid, as it is information leakage, the best alpha values selected on the test partition are only slightly better compared to the last value, which was used.

In the following table, the results from the new selected alphas are shown for the Cox regression models with elastic net penalties. It is important to note, while this selection of a hyperparameter overestimates its performance, but it was done to illustrate the effect of the chosen alpha.

Table 8 Performance of the Cox Regression models with Elastic Net penalties when choosing a new alpha

Target	*Preprocessing*	*Components / Clusters*	*L1 Ratio*	*Alpha Used*	*C-Index*	*Best performing Alpha on test partition*	*New C-index*
FE	Feature Agg.	256	0.7	0.0002	0.73	0.006	0.75
FF	PCA	128	0.1	0.001	0.72	0.010	0.72
DE	PCA	256	0.1	0.0007	0.75	0.011	0.76
DF	PCA	256	0.4	0.0001	0.69	0.007	0.70
EE	None		1.0	0.0002	0.69	0.004	0.69

While the effect of the alpha appears low, the chosen hyperparameter might have been different if the best alphas would have been used for each configuration.

4.4.4 Survival Trees

The survival trees showed performance only slightly better than the Kaplan-Meier estimator. This suggests that small trees with few nodes were selected whose estimates are close to the Kaplan-Meier estimator with all training instances. For the graft failure targets (FF, FE), the decision tree used no dimension reduction, while all other targets used feature agglomeration with 128 clusters. The suspicion that small trees were used is confirmed as all ranger survival tree models were selected to use a maximum depth of 3. They provided the best performance during the grid search.

The average C-index with one standard error in each direction on the test folds for the selected dimension reduction as previously listed is shown for each tested maximum tree depth is plotted for the next figure.

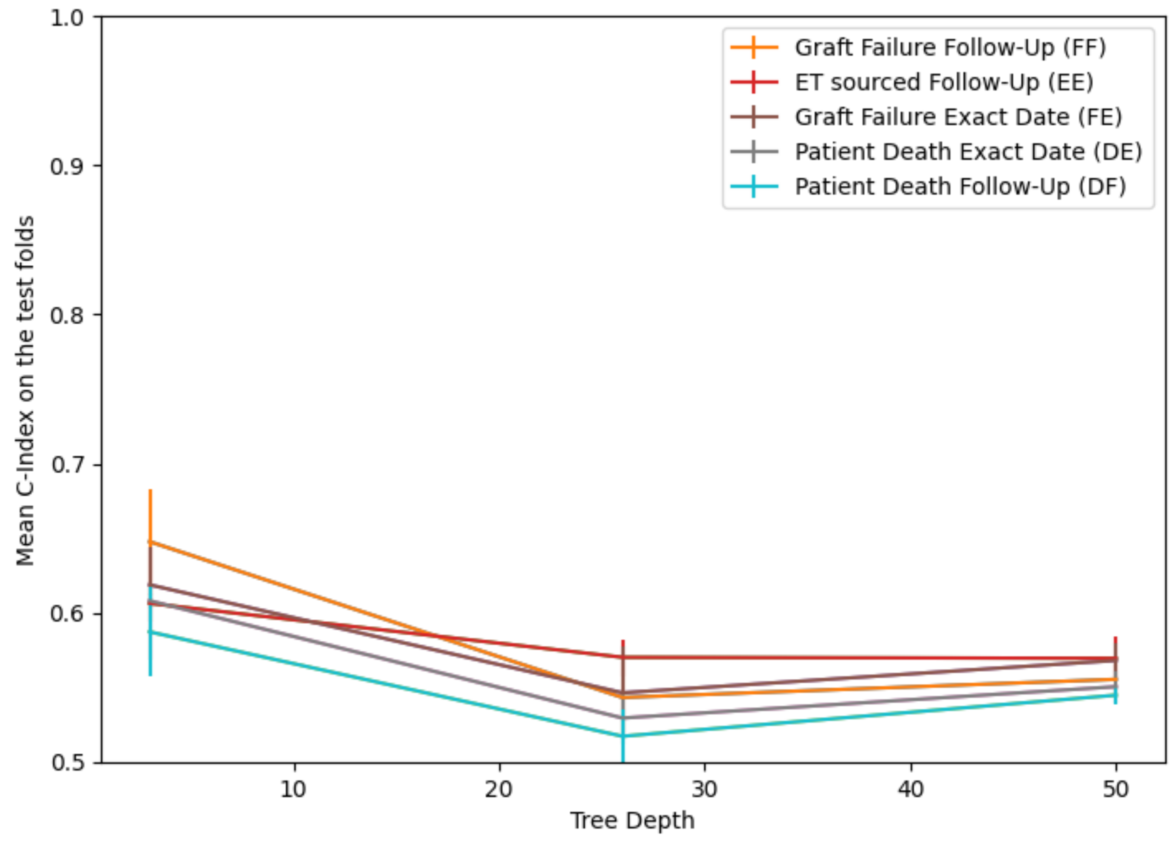


Figure 25 Average test fold performance for each tested depth with the selected dimension reduction

Trees with higher depth tend to overfit, while shallower trees tend to underfit. For survival trees an additional constraint to consider is that the training instances placed in the leaves are used to construct a Kaplan-Meier estimator which requires a certain number of subjects. Instead of using the tree depths, the minimum number of instances in the leaves might be a better parameter to optimize for survival trees. For further optimization of survival trees, the performance of trees with depths between 3 and 26 might be interesting.

Because the trees for graft failure targets used no dimension reduction, the used features can be directly extracted from the tree structure. For the FE target, the recipient age, donor age, warm ischemia time, relative discharge date, quality assignment of the kidney, the donor partial pressure in their blood, and an internal ordinal identifier used for joining ET and IQTIG data was used. The relative discharge date and the identifier, which was not deleted from the dataset, could be information leakage, as a long stay shortly after transplantation suggests immediate failure and the ordinal identifier may indicate order in the data files and therefore failure times. For the FF target, the kidney quality, presence of HIV antigens in the recipient, the missing value indicator for the heart state of the donor, alkaline phosphatase activity for the donor, the positive end-expiratory pressure (PEEP), general quality assignment of donor health and donor blood hemoglobin was used. Here the missing value indicator for the heart state of the donor could be an information leaking feature. While present through different features, the oxygen supply of the donor’s body and the kidney health appear as useful features.

4.4.5 Survival Random Forest

For survival random forests, only the patient death targets were selected to use a dimension reduction with a feature agglomeration with 256 clusters. Like the previously discussed models, AUC scoring on the folds of FF failed again. The feature agglomeration with 256 clusters mostly produces clusters with a single feature, therefore it is close to having no reduction. All forests used the highest number tested of 50 estimators. This is a small number of estimators; the default is 100 and usually random forests are trained with more estimators. Just like the single trees, for the graft failure FF targets and the exact death DE target a tree depth of three was selected. The death follow-up target DF used a maximum depth of 26 and the target EE a maximum depth of 50. It is important to note, that each node considered only a subsample of features. This may lead to bigger trees being selected in order to use certain features to achieve better performance. When counting the used features across the trees all forests with a tree depth of three used slightly more than 100 features, while the death exact date target and ET sourced follow-up target use slightly above 250 features. While they show no problematic overestimated performance on the test partition and tests folds, they are the only configurations where the model trained on the entire training partitions shows an almost perfect score on the training partition.

The best configurations retrained on the train partition are shown in the following table for the survival forest.

Table 9 Perfomance of the survival random forest models

Target	C-Index Test	C-Index Train	Used Features	Dimension Reduction	Clusters	Max Depth	Trees
Graft Failure Exact Date (FE)	0.75	0.76	103	None		3	50
Graft Failure Follow-Up (FF)	0.73	0.79	106	None		3	50
Patient Death Exact Date (DE)	0.78	0.98	252	Feature Agglomeration	256	26	50
Patient Death Follow-Up (DF)	0.71	0.77	116	Feature Agglomeration	256	3	50
ET sourced Follow-Up (EE)	0.73	0.95	253	None		50	50

4.4.6 Gradient Boosting

As previously mentioned, due to a bug in scikit-survival the "IPCWLS" loss was incorrectly scored for the gradient boosted models. This leads to the selected configurations always using the Cox regression loss, where the prediction is on the risk scale. All selected pipeline candidates either used no dimension reduction or Feature Agglomeration with 256 clusters, which again can be considered equivalent to no reduction. Like the random forests, they also used 50 estimators. Overall, the internally trained regression trees used 210 to 225 features. For all gradient boosted models, the estimated training error suggests that it is almost perfectly fitting to the training data. All candidates used the learning rate 1. Such a high learning rate would lead other model applications to heavily overfit the data.

The best configurations retrained on the train partition are shown in the following table for the gradient boosting model.

Table 10 Perfomance of the survival gradient boosting models

Target	C-Index Test	C-Index Train	Used Features	Dimension Reduction	Clusters	Learning rate	Trees
Graft Failure Exact Date (FE)	0.78	0.93	223	Feature Agglomeration	256	1.0	50
Graft Failure Follow-Up (FF)	0.77	0.94	224	None		1.0	50
Patient Death Exact Date (DE)	0.81	0.93	215	Feature Agglomeration	256	1.0	50
Patient Death Follow-Up (DF)	0.73	0.94	216	None		1.0	50
ET sourced Follow-Up (EE)	0.74	0.91	210	Feature Agglomeration	256	1.0	50

After applying the fix for the correct scoring of the IPCWLS loss to a new grid search on the EE target, the Cox regression-based loss again achieves better results with the same mean test fold score. With the fair scoring however, the best configuration for the IPCWLS loss achieves a 0.67 C-index on average across the test folds with the 256-cluster feature agglomeration compared to the old results of 0.48 with a PCA with 128 components.

4.4.7 Support Vector Machine

The support vector machines used no dimension reduction for the ET sourced follow-up target EE. Feature agglomeration with 256 clusters was used for the exact date graft failure target FE and the patient death follow-up target DF. A PCA with 64 components was selected for the graft failure follow-up target and 256 components for the exact patient death date target DE. Two of the three tested alpha values were observed. The smallest tested value of 0.002479 was chosen for the targets FF, DE and DF, while the largest value 0.606531 was used for the FE and EE target. More values should have been tested, as possible alpha values sometimes can also be larger than 1 or much smaller. The survival support vector machine showed similar performance to the Cox regression models and AFT regression models. With no kernel applied, a survival support vector machine has almost an identical model definition as a linear AFT model.

4.5 Feature Importance

To consider the results from a linear model and a non-linear model, the feature importance on the C-index for the Cox regression model and the gradient boosting model will be used to identify the most important features for the targets on the test partitions. An important feature will be defined as decreasing the C-index by at least 0.01 on average when the feature is shuffled.

For the Cox regression on the FE target, important features were the number of any organ transplants the patient ever had, the discharge date from the hospital and the age of the donor. The number of transplants could be higher for patients who received a transplant, and the discharge date may indicate whether there were early problems with the transplantations. These two features should be removed for future analysis. The corresponding gradient boosting model had 7 important features. They included the allowed maximum donor age allowed for a recipient, the urine production volume of the transplant, the cold ischemia time, and whether the recipient has German citizenship. This model also used the discharge date and the total number of transplants in addition to the previously mentioned ordinal identifier. These features may leak information to the model. The other features appear to be clinically important. For the FF target, no very important features could be identified for the Cox

regression and for the corresponding gradient boosting model, the discharge date was identified.
On the DE target the Cox regression model used the donor age, the discharge date, the number of previous transplantations, whether the left kidney of the donor was legally allowed to be transplanted, the numeric representation of the hospital discharge code, and the age of the recipient. The gradient boosting model on the other hand used the warm and cold ischemia times, the ordinal id, the discharge date, the number of any transplants, the recipient age, and the discharge reason code as its most important features. For the Cox regression on the DF target 255 features were identified as important. This might be connected to the follow-up pattern not being visible in the baseline survival functions. It is also using larger linear predictor values compared to the other models. The gradient boosting model identified the age, the cold ischemia time, the discharge date, and the recipient age as important features.
For the EE target the Cox regression identified 262 important features. The reason might be similar to the DF target, where large linear predictors are present. For this target, the gradient boosting model identified the urine production during transplantation, the discharge date, the cold ischemia time, and the warm ischemia time as the most important features.

5 Conclusion

The goal of the research project was overall reached. Different models were trained and evaluated on the constructed targets and data from the transplantation registry. With more time invested in data preprocessing, model engineering, and model optimization better results might be possible. Some information leaking features were passed to the models. Leading to overestimated model performance. For future projects the features should be manually inspected more closely for more reliable results.
Promising future research areas are the improvement of the data preprocessing pipeline as well as the reduction of bias in the estimates due to the censoring patterns in the different targets.

6 Appendix

The following table lists the tested hyperparameters.

Table 11 Tested hyperparameters

Parameter	Values
Dimension Reduction	Feature agglomeration, PCA, None
Clusters/Components for Dimension Reduction	64, 128, 256
Residual Distribution for Linear AFT Model	Weibull, exponential, lognormal
Alpha Value for IPCW Ridge Model and Survival SVM	0.00247875, 0.01550385, 0.09697197, 0.60653066
Ensemble Members for Random Forest and Gradient Boosting	10, 25, 50
Tree Depth for Random Forest and Survival Tree	3, 26, 50
Loss Functions for Gradient Boosting	Partial Likelihood, IPCW weighted squared error
Learning Rate for Gradient Boosting	1.0e-05, 3.16227766e-03, 1
L1 Ratios for Cox Regression with elastic net penalty	0.1, 0.4, 0.7, 1

In the following table the prediction performance of the best configuration for each model and target on the test partition is listed.

Table 12 Prediction performance on the test partitions

Target	Model Class	C-In-dex	IPCW C-In-dex	Integrated Brier Error	Mean C,D AUC	Min. Test Time	Max. Test Time	Test Rows	Test Rows Used
EE	GradientBoostingSurvivalAnalysis	0,74	0,72	0,155	0,77	1	3653,5	3966	3943
EE	RangerForestSurvival	0,73	0,70	0,159	0,74	1	3653,5	3966	3943
EE	RSurvreg	0,69	0,65	0,174	0,68	1	3653,5	3966	3943
EE	FastSurvivalSVM	0,69	0,64		0,68	1	3653,5	3966	3943
EE	CoxnetSurvivalAnalysis	0,69	0,65	0,167	0,69	1	3653,5	3966	3943
EE	CoxPHSurvivalAnalysis	0,69	0,65	0,167	0,69	1	3653,5	3966	3943
EE	RangerTreeSurvival	0,56	0,56	0,182	0,56	1	3653,5	3966	3943
EE	RCoxphNull	0,50	0,50	0,181	0,50	1	3653,5	3966	3943
EE	IPCRidge	0,47	0,49		0,42	1	3653,5	3966	3943
FE	GradientBoostingSurvivalAnalysis	0,78	0,78	0,094	0,80	1	1126,75	5456	4147
FE	RangerForestSurvival	0,75	0,75	0,110	0,77	1	1126,75	5456	4147
FE	CoxnetSurvivalAnalysis	0,73	0,73	0,102	0,75	1	1126,75	5456	4147
FE	FastSurvivalSVM	0,73	0,73		0,75	1	1126,75	5456	4147
FE	CoxPHSurvivalAnalysis	0,73	0,72	0,102	0,75	1	1126,75	5456	4147
FE	RSurvreg	0,71	0,71	0,105	0,73	1	1126,75	5456	4147
FE	RangerTreeSurvival	0,61	0,61	0,112	0,61	1	1126,75	5456	4147
FE	RCoxphNull	0,50	0,50	0,118	0,50	1	1126,75	5456	4147
FE	IPCRidge	0,47	0,48		0,46	1	1126,75	5456	4147
FF	GradientBoostingSurvivalAnalysis	0,77	0,78	0,032	0,79	41	1126,75	5087	4010
FF	RangerForestSurvival	0,73	0,75	0,032	0,78	41	1126,75	5087	4010
FF	CoxnetSurvivalAnalysis	0,72	0,73	0,032	0,75	41	1126,75	5087	4010

FF	RSurvreg	0,71	0,72	0,032	0,74	41	1126,75	5087	4010
FF	CoxPHSurvivalAnalysis	0,70	0,71	0,032	0,73	41	1126,75	5087	4010
FF	FastSurvivalSVM	0,69	0,70		0,72	41	1126,75	5087	4010
FF	RangerTreeSurvival	0,58	0,59	0,032	0,60	41	1126,75	5087	4010
FF	RCoxphNull	0,50	0,50	0,034	0,50	41	1126,75	5087	4010
FF	IPCRidge	0,46	0,47		0,45	41	1126,75	5087	4010
DE	GradientBoostingSurvivalAnalysis	0,81	0,80	0,074	0,84	1	1126,75	5566	3946
DE	RangerForestSurvival	0,78	0,77	0,076	0,82	1	1126,75	5566	3946
DE	FastSurvivalSVM	0,76	0,75		0,78	1	1126,75	5566	3946
DE	CoxnetSurvivalAnalysis	0,75	0,75	0,083	0,77	1	1126,75	5566	3946
DE	CoxPHSurvivalAnalysis	0,75	0,75	0,083	0,77	1	1126,75	5566	3946
DE	RSurvreg	0,73	0,73	0,085	0,76	1	1126,75	5566	3946
DE	RangerTreeSurvival	0,51	0,51	0,095	0,52	1	1126,75	5566	3946
DE	RCoxphNull	0,50	0,50	0,094	0,50	1	1126,75	5566	3946
DE	IPCRidge	0,48	0,49		0,45	1	1126,75	5566	3946
DF	GradientBoostingSurvivalAnalysis	0,73	0,73	0,038	0,74	19	1126,75	5185	4086
DF	RangerForestSurvival	0,71	0,71	0,038	0,73	19	1126,75	5185	4086
DF	CoxnetSurvivalAnalysis	0,69	0,68	0,038	0,71	19	1126,75	5185	4086
DF	CoxPHSurvivalAnalysis	0,69	0,68	0,038	0,71	19	1126,75	5185	4086
DF	RSurvreg	0,68	0,67	0,037	0,71	19	1126,75	5185	4086
DF	FastSurvivalSVM	0,68	0,67		0,70	19	1126,75	5185	4086
DF	RangerTreeSurvival	0,50	0,50	0,039	0,50	19	1126,75	5185	4086
DF	RCoxphNull	0,50	0,50	0,039	0,50	19	1126,75	5185	4086
DF	IPCRidge	0,50	0,50		0,47	19	1126,75	5185	4086

The following table shows the best results for each target and model during the grid search.

Table 13 Best grid search results

Target	Model Class	Dimension Reduction Class	Clusters / Components	Used Hyperparameters	Average Training Tim in s	Std. of Training Times in s	Average Test Fold C-Index	Std. Test Fold C-Index
DE	GradientBoostingSurvivalAnalysis	Feature Agglomeration	256	n_estimators=50.0, learning_rate=1.0, loss=coxph	34,1	0,1	0,79	0,00
DE	RangerForestSurvival	Feature Agglomeration	256	max_depth=26.0, n_estimators=50.0	4,4	0,2	0,76	0,01
DE	FastSurvivalSVM	PCA	256	alpha=0.0024787521766663	51,0	18,1	0,74	0,00
DE	CoxnetSurvivalAnalysis	PCA	256	l1_ratio=0.1	1,3	0,0	0,73	0,01
DE	CoxPHSurvivalAnalysis	PCA	256		52,4	20,3	0,73	0,01
DE	RSurvreg	PCA	128	dist=lognormal	0,9	0,0	0,72	0,01
DE	RangerTreeSurvival	Feature Agglomeration	128	max_depth=3.0	0,8	0,1	0,61	0,01
DE	IPCRidge	PCA	64	alpha=0.6065306597126334	0,2	0,0	0,46	0,02
DF	GradientBoostingSurvivalAnalysis			n_estimators=50.0, learning_rate=1.0, loss=coxph	28,7	0,1	0,73	0,02
DF	CoxnetSurvivalAnalysis	PCA	256	l1_ratio=0.4	1,3	0,1	0,71	0,02
DF	RangerForestSurvival	Feature Agglomeration	256	max_depth=3.0, n_estimators=50.0	0,8	0,0	0,71	0,02
DF	CoxPHSurvivalAnalysis	PCA	256		12,9	1,0	0,71	0,02
DF	FastSurvivalSVM	Feature Agglomeration	256	alpha=0.0024787521766663	22,8	4,3	0,70	0,01
DF	RSurvreg	PCA	128	dist=lognormal	0,9	0,0	0,70	0,03
DF	RangerTreeSurvival	Feature Agglomeration	128	max_depth=3.0	0,8	0,1	0,59	0,03

DF	IPCRidge	PCA	128	alpha=0.0155038535990093	0,2	0,0	0,49	0,02
EE	GradientBoostingSurvivalAnalysis	Feature Agglomeration	256	n_estimators=50.0, learning_rate=1.0, loss=coxph	19,5	0,1	0,71	0,01
EE	RangerForestSurvival			max_depth=50.0, n_estimators=50.0	12,3	0,7	0,70	0,01
EE	RSurvreg	Feature Agglomeration	256	dist=exponential	4,6	0,0	0,68	0,01
EE	CoxnetSurvivalAnalysis			l1_ratio=1.0	1,4	0,1	0,67	0,01
EE	FastSurvivalSVM			alpha=0.6065306597126334	22,5	10,3	0,67	0,01
EE	CoxPHSurvivalAnalysis	PCA	256		32,7	28,0	0,67	0,01
EE	RangerTreeSurvival	Feature Agglomeration	128	max_depth=3.0	0,8	0,1	0,61	0,04
EE	IPCRidge	Feature Agglomeration	64	alpha=0.6065306597126334	0,1	0,0	0,47	0,02
FE	GradientBoostingSurvivalAnalysis	Feature Agglomeration	256	n_estimators=50.0, learning_rate=1.0, loss=coxph	32,4	0,1	0,76	0,01
FE	RangerForestSurvival			max_depth=3.0, n_estimators=50.0	1,2	0,1	0,75	0,02
FE	CoxnetSurvivalAnalysis	Feature Agglomeration	256	l1_ratio=0.7	1,4	0,1	0,72	0,01
FE	FastSurvivalSVM	Feature Agglomeration	256	alpha=0.6065306597126334	43,4	2,9	0,72	0,01
FE	CoxPHSurvivalAnalysis	PCA	256		32,9	24,5	0,71	0,01
FE	RSurvreg	PCA	128	dist=lognormal	0,9	0,1	0,70	0,01
FE	RangerTreeSurvival			max_depth=3.0	0,3	0,0	0,62	0,03
FE	IPCRidge	PCA	64	alpha=0.6065306597126334	0,2	0,0	0,50	0,02
FF	GradientBoostingSurvivalAnalysis			n_estimators=50.0, learning_rate=1.0, loss=coxph	27,7	0,1	0,74	0,01

FF	RangerForestSurvival			max_depth=3.0, n_estima-tors=50.0	0,8	0,1	0,73	0,01
FF	RSurvreg	PCA	128	dist=weibull	1,4	0,2	0,70	0,01
FF	CoxnetSurvivalAnalysis	PCA	128	l1_ratio=0.1	1,5	0,8	0,70	0,01
FF	CoxPHSurvivalAnalysis	PCA	128		2,2	0,3	0,70	0,01
FF	FastSurvivalSVM	PCA	64	alpha=0.0024787521766663	3,5	0,7	0,70	0,01
FF	RangerTreeSurvival			max_depth=3.0	0,3	0,0	0,65	0,04
FF	IPCRidge	PCA	256	alpha=0.096971967864405	0,2	0,0	0,48	0,02

7 References

Ayers, B., Sandholm, T., Gosev, I., Prasad, S., & Kilic, A. (2021). Using machine learning to improve survival prediction after heart transplantation. *Journal of Cardiac Surgery*, *36*(11), 4113–4120. https://doi.org/10.1111/jocs.15917

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification And Regression Trees*. Routledge. https://doi.org/10.1201/9781315139470

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). *API design for machine learning software: experiences from the scikit-learn project*.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*. https://doi.org/10.1145/2939672.2939785

Clark, T. G., Bradburn, M. J., Love, S. B., & Altman, D. G. (2003). Survival Analysis Part I: Basic concepts and first analyses. *British Journal of Cancer*, *89*(2), 232–238. https://doi.org/10.1038/sj.bjc.6601118

Cox, D. R. (1972). Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, *34*(2). https://doi.org/10.1111/j.2517-6161.1972.tb00899.x

Davies, R. B. (1971). Rank Tests for "Lehmann's Alternative." *Journal of the American Statistical Association*, *66*(336), 879. https://doi.org/10.2307/2284246

Faruk, A. (2018). The comparison of proportional hazards and accelerated failure time models in analyzing the first birth interval survival data. *Journal of Physics: Conference Series*, *974*, 012008. https://doi.org/10.1088/1742-6596/974/1/012008

Fleming, T. R., & Harrington, D. P. (1984). Nonparametric estimation of the survival distribution in censored data. *Communications in Statistics - Theory and Methods*, *13*(20), 2469–2486. https://doi.org/10.1080/03610928408828837

Fleming, T. R., & Harrington, D. P. (2011). Weighted Logrank Statistics. In *Counting Processes and Survival Analysis* (pp. 255–285). https://doi.org/10.1002/9781118150672.ch7

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, *38*(4). https://doi.org/10.1016/S0167-9473(01)00065-2

Gotlieb, N., Azhie, A., Sharma, D., Spann, A., Suo, N.-J., Tran, J., Orchanian-Cheff, A., Wang, B., Goldenberg, A., Chassé, M., Cardinal, H., Cohen, J. P., Lodi, A., Dieude, M., & Bhat, M. (2022). The promise of machine learning applications in solid organ transplantation. *Npj Digital Medicine*, *5*(1), 89. https://doi.org/10.1038/s41746-022-00637-2

Harrell, F. E., Lee, K. L., & Mark, D. B. (1996a). Multivariable prognostic models: Issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, *15*(4). https://doi.org/10.1002/(SICI)1097-0258(19960229)15:4<361::AID-SIM168>3.0.CO;2-4

Harrell, F. E., Lee, K. L., & Mark, D. B. (1996b). Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, *15*(4), 361–387. https://doi.org/10.1002/(SICI)1097-0258(19960229)15:4<361::AID-SIM168>3.0.CO;2-4

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer New York. https://doi.org/10.1007/978-0-387-84858-7

Hertz-Picciotto, I., & Rockhill, B. (1997). Validity and Efficiency of Approximation Methods for Tied Survival Times in Cox Regression. *Biometrics*, *53*(3), 1151. https://doi.org/10.2307/2533573

Hosmer, D. W., Lemeshow, S., & May, S. (2008). *Applied Survival Analysis*. John Wiley & Sons, Inc. https://doi.org/10.1002/9780470258019

Hothorn, T., Bühlmann, P., Dudoit, S., Molinaro, A., & van der Laan, M. (2005). Survival ensembles. *Biostatistics*, *7*(3), 355–373. https://doi.org/10.1093/biostatistics/kxj011

Ishwaran, H., & Kogalur, U. B. (2007). Random survival forests for R. *R News*, *7*(2), 25–31.

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

James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An Introduction to Statistical Learning - with Applications in R | Gareth James | Springer. In *Book*.

Kantidakis, G., Putter, H., Lancia, C., Boer, J. de, Braat, A. E., & Fiocco, M. (2020a). Survival prediction models since liver transplantation - comparisons between Cox models and machine learning techniques. *BMC Medical Research Methodology*, *20*(1), 277. https://doi.org/10.1186/s12874-020-01153-1

Kantidakis, G., Putter, H., Lancia, C., Boer, J. de, Braat, A. E., & Fiocco, M. (2020b). Survival prediction models since liver transplantation - comparisons between Cox models and machine learning techniques. *BMC Medical Research Methodology*, *20*(1), 277. https://doi.org/10.1186/s12874-020-01153-1

Kartsonaki, C. (2016). Survival analysis. *Diagnostic Histopathology*, *22*(7), 263–270. https://doi.org/10.1016/j.mpdhp.2016.06.005

Kim, C., Park, B. U., Kim, W., & Lim, C. (2003). Bezier curve smoothing of the Kaplan-Meier estimator. *Annals of the Institute of Statistical Mathematics*, *55*(2), 359–367. https://doi.org/10.1007/BF02530504

Lambert, J., & Chevret, S. (2016). Summary measure of discrimination in survival models based on cumulative/dynamic time-dependent ROC curves. *Statistical Methods in Medical Research*, *25*(5). https://doi.org/10.1177/0962280213515571

LeBlanc, M., & Crowley, J. (1993). Survival Trees by Goodness of Split. *Journal of the American Statistical Association*, *88*(422), 457. https://doi.org/10.2307/2290325

Lee, C. P., & Lin, C. J. (2014). Large-scale linear rankSVM. In *Neural Computation* (Vol. 26, Issue 4). https://doi.org/10.1162/NECO_a_00571

Lesko, C. R., Edwards, J. K., Cole, S. R., Moore, R. D., & Lau, B. (2018). When to Censor? *American Journal of Epidemiology*, *187*(3), 623–632. https://doi.org/10.1093/aje/kwx281

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18*.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*.

McKinney, W., & Team, P. D. (2015). Pandas - Powerful Python Data Analysis Toolkit. *Pandas - Powerful Python Data Analysis Toolkit*.

Minka, T. P. (2001). Automatic choice of dimensionality for PCA. *Advances in Neural Information Processing Systems*.

Moore, D. F. (2016). *Applied Survival Analysis Using R*. Springer International Publishing. https://doi.org/10.1007/978-3-319-31245-3

Nagel, R. (2020, July 17). *Technische Spezifikation - Registerdatenbank*. Transplantations-Register.De.

Nagel, R. (2022). *Über das Transplantationsregister*. Transplantations-Register.De. https://transplantations-register.de/ueber-das-transplantationsregister

Naqvi, S. A. A., Tennankore, K., Vinson, A., Roy, P. C., & Abidi, S. S. R. (2021). Predicting Kidney Graft Survival Using Machine Learning Methods: Prediction Model Development and Feature Significance Analysis Study. *Journal of Medical Internet Research*, *23*(8), e26843. https://doi.org/10.2196/26843

Orbe, J., Ferreira, E., & Núñez-Antón, V. (2002). Comparing proportional hazards and accelerated failure time models for survival analysis. *Statistics in Medicine*, *21*(22), 3493–3510. https://doi.org/10.1002/sim.1251

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

Pepe, M., Longton, G., & Janes, H. (2009). Estimation and Comparison of Receiver Operating Characteristic Curves. *The Stata Journal*, *9*(1), 1.

Pölsterl, S. (2020). scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn. *Journal of Machine Learning Research*, *21*(212), 1–6. http://jmlr.org/papers/v21/20-729.html

Pölsterl, S., Navab, N., & Katouzian, A. (2015). Fast training of support vector machines for survival analysis. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9285*. https://doi.org/10.1007/978-3-319-23525-7_15

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/BF00116251

Rahmel, A., & Wadewitz, J. (2020). *Grafiken zum Tätigkeitsbericht 2019 veröffentlicht durch die Deutsche Stiftung Organtransplantation*. https://www.dso.de/BerichteTransplantationszentren/Grafiken%20D%202019%20Niere.pdf

Ridgeway, G. (1999). The state of boosting. *Computing Science and Statistics*, *31*.

SAS Institute Inc. (2019, February 13). *The LIFETEST Procedure - SAS Help Center*. Https://Documentation.Sas.Com/Doc/En/Pgmsascdc/9.4_3.3/Statug/Statug_lifetest_overview.Htm.

Schumacher, M., Graf, E., & Gerds, T. (2003). How to Assess Prognostic Models for Survival Data: A Case Study in Oncology. *Methods of Information in Medicine*, *42*(5). https://doi.org/10.1055/s-0038-1634384

Sestelo, M. (2017). *A short course on Survival Analysis applied to the Financial Industry*. Bookdown.Org. https://bookdown.org/sestelo/sa_financial/

Simon, N., Friedman, J., Hastie, T., & Tibshirani, R. (2011). Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. *Journal of Statistical Software*, *39*(5). https://doi.org/10.18637/jss.v039.i05

Stekhoven, D. J., & Buhlmann, P. (2012). MissForest--non-parametric missing value imputation for mixed-type data. *Bioinformatics*, *28*(1), 112–118. https://doi.org/10.1093/bioinformatics/btr597

Terry M. Therneau, & Patricia M. Grambsch. (2000). *Modeling Survival Data: Extending the Cox Model*. Springer.

Therneau, T. M. (2022). *A Package for Survival Analysis in R*. https://CRAN.R-project.org/package=survival

Tieken, C. M., de Boer, J., & Hagenaars, J. (2022). Eurotransplant Manual Chapter 4. In *https://www.eurotransplant.org/*. Eurotransplant Foundation .

Tieken, C. M., de Boer, J., Heidt, S., & Doxiadis, I. (2022). Eurotransplant Manual Chapter 10. In *https://www.eurotransplant.org/*. Eurotransplant Foundation .

Turkson, A. J., Ayiah-Mensah, F., & Nimoh, V. (2021). Handling Censoring and Censored Data in Survival Analysis: A Standalone Systematic Literature Review. *International Journal of Mathematics and Mathematical Sciences*, *2021*, 1–16. https://doi.org/10.1155/2021/9307475

Uno, H., Cai, T., Pencina, M. J., D'Agostino, R. B., & Wei, L. J. (2011). On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in Medicine*, *30*(10), 1105–1117. https://doi.org/10.1002/sim.4154

Vieth, F. (2022). *Die Nierentransplantation*. Organspende Info. https://www.organspende-info.de/organspende/transplantierbare-organe/nierentransplantation/

Vock, D. M., Wolfson, J., Bandyopadhyay, S., Adomavicius, G., Johnson, P. E., Vazquez-Benitez, G., & O'Connor, P. J. (2016). Adapting machine learning techniques to censored time-to-event health record data: A general-purpose approach using inverse probability of censoring weighting. *Journal of Biomedical Informatics*, *61*, 119–131. https://doi.org/10.1016/j.jbi.2016.03.009

Wright, M. N., & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, *77*(1). https://doi.org/10.18637/jss.v077.i01

Zhang, Z., Reinikainen, J., Adeleke, K. A., Pieterse, M. E., & Groothuis-Oudshoorn, C. G. M. (2018). Time-varying covariates and coefficients in Cox regression models. *Annals of Translational Medicine*, *6*(7), 121. https://doi.org/10.21037/atm.2018.02.12

Zou, H., & Hastie, T. (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, *67*(2), 301–320. http://www.jstor.org/stable/3647580