Machine Learning in Survival Analysis

Getting Started

Main Title

Standard blurb goes here

%%Placeholder for Half title

Series page goes here (if applicable); otherwise blank

Machine Learning in Survival Analysis

Raphael Sonabend, Andreas Bender

Imprint page here; PE will provide text

Table of contents

References **123**

Preface

> "Everything happens to everybody sooner or later if there is time enough" - George Bernard Shaw
>
> "...but in this world nothing can be said to be certain, except death and taxes." - Benjamin Franklin

Together, the epigraphs for this book perfectly and succinctly set up the 'survival problem'. In maths perhaps we could be even more succinct: $\forall X(P(X) \to 1 \text{ as } t \to \infty)$. Of course, nothing is immortal and therefore there is never 'time enough'. Thus creating the predictive 'survival problem': given an individual (object), predict when this individual will experience an event of interest. The 'problem' in survival analysis is that we implicitly assume the individual *will* experience the event, which is of course not guaranteed (with the exception of death or taxes). To make matters worse, the vast majority of accessible models to make survival predictions, encode an even stricter assumption, that either the event of interest will occur, or, for some completely unrelated reason, it will not. For example, a model assumes either a marathon runner will finish a race, or they won't due to some arbitrary reason, e.g., someone else pushing them over. This assumption does not reflect the real-world, in practice the runner may not finish the marathon due to some related reasons, such as extreme exhaustion.

A note from RS: When I began writing my PhD thesis I was motivated by two questions: given the enormous power of survival analysis (who doesn't want to be able to predict the future?), and given the rapid rise in interest in machine learning, i) why are more data scientists not using machine learning? ii) and why do machine learning models fail to outperform simple, classical models that have been around for decades? The answer it transpired was fairly simple, and largely ignored by researchers (including myself), for decades. The survival analysis problem is simply impossible. In 1975, Anastasios Tsiatis published a paper as part of their PhD submission (Tsiatis 1975) demonstrating the 'non-identifiability problem' in survival analysis. In the abstract of this paper, Tsiatis sums up the problem we face today: "based on the assumption that [competing events] are independent, may have no resemblance to reality". In simpler terms, the majority of survival models encode an assumption that is very hard (/impossible) to justify in reality and in doing so, predictions made by these models "may have no resemblance to reality". Despite this, the majority of this book is going to focus on these models, and we will come back to why after a brief detour to the alternative treatment of events.

A note from AB (competing risks): Amet occaecat veniam consectetur irure labore sit nulla ex quis non id non. Tempor fugiat dolore eiusmod veniam dolore labore eu adipisicing aliqua quis culpa proident in aliqua. Mollit incididunt sunt mollit id fugiat est eu esse cillum enim. Occaecat culpa sint reprehenderit proident quis duis reprehenderit Lorem sint deserunt cupidatat nisi. Aliquip duis ullamco aliquip ullamco nisi non fugiat eu sunt cupidatat qui eu velit elit. Dolore occaecat id ad reprehenderit ea irure qui mollit nostrud.

So in this book, we will consider both the 'standard framework' and the competing risks

framework. Instead of treating these as two distinct parts to the book, we will interweave methods throughout the narrative, to highlight our belief that all practitioners should have an equally firm grasp on both settings. Where competing risk alternatives do not yet exist, readers are directed to Chapter 15 which includes a section on extending *any* survival model, to the competing risks setting. Some readers may be questioning why we event both including the non-competing framework at all, which is a reasonable question. The answer is threefold, firstly there are times when this setting is appropriate, if you can reasonably justify that the sole reason for an event not occurring is independent of the event itself (we will see examples of this later in the book), then this framework is valid and is the simpler setting; secondly, as mentioned above, any model in this framework can be extended to competing risks, and therefore knowing about these models is still important; thirdly, so that readers who may be reviewing other academic texts, can spot errors and help correct them when found in the wild.

Overview

We wrote this book as we felt there was a gap in introductory academic textbooks for machine learning that satisfactorily cover the survival setting. There are many good books for machine learning (e.g., James et al. (2013), Hastie, Tibshirani, and Friedman (2001), Bishop (2006)) and survival analysis (e.g., Collett (2014), J. D. Kalbfleisch and Prentice (1973)) but we hope this book bridges a gap between these two fields. We recommend reading this book alongside (or after) James et al. (2013) and Collett (2014) as we do not want to duplicate content from these books. Instead, we will introduce machine learning methods in the regression setting and then demonstrate how these can extend to survival analysis. In Chapter 1 we explain the motivation of this choice by demonstrating how survival analysis can be reduced to the regression and classification settings and we exploit these results further in Chapter 15.

This book may be useful for Masters or PhD students who are specialising in machine learning in survival analysis, or machine learning practitioners looking to work in the survival setting, or statisticians who are familiar with survival analysis but less so with machine learning. We include examples and exercises in R, Julia, and Python [FIXME].

This book is split into three parts:

Part I: Survival Analysis and Machine Learning We begin the book by introducing the basics of survival analysis and machine learning and unifying terminology between the two to enable us to meaningfully talk about 'machine learning in survival analysis' (MLSA). Sit cupidatat non enim quis amet anim velit incididunt.

Part II: MLSA Models In the second part of the book we begin our deep dive into machine learning models for solving survival analysis problems. We begin with 'classical' models that may not be considered 'machine learning' and then continue by exploring different classes of machine learning models including random forests, gradient boosting machines, and neural networks. Veniam laboris consequat commodo nostrud in ipsum occaecat consequat ea adipisicing fugiat Lorem nisi in.

Part III: Evaluating and Extending In the next part of the book we discuss how to evaluate the models covered in the previous part using measures of discrimination, calibration, and scoring rules. We then discuss how to further extend models and pipelines by making use of composition and reduction and the close links between survival analysis, regression, and classification. Ea ex adipisicing occaecat eu consectetur sunt culpa ad.

In the final chapters of the book we list recommended open-source software for survival

analysis and then conclude the book.

Citing this book

Whilst this book remains a work in progress you can cite it as

```
Sonabend. R, Bender. A. (2023). Machine Learning in Survival Analysis.
https://www.mlsabook.com.
```

```
@book{MLSA2023
    title = Machine Learning in Survival Analysis
    editor = {Raphael Sonabend, Andreas Bender},
    url = {https://www.mlsabook.com},
    year = {2023}
}
```


We hope you enjoy reading this book.

Raphael and Andreas

Authors

Raphael Sonabend is a Visiting Researcher at Imperial College London and an Open Source Manager at Genomics England. They hold a PhD in statistics, specializing in machine learning applications for survival analysis. They wrote the `mlr3` packages `mlr3proba` and `mlr3benchmark`.

Andreas Bender is...

Symbols and Notation

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

The most common symbols and notation used throughout this book are presented below; in rare cases where different meanings are intended within the book, this will be made clear.

A lower-case letter in normal font, x, refers to a single, fixed observation. When in bold font, a lower-case letter, $\mathbf{x}$, refers to a vector of fixed observations, and an upper-case letter, $\mathbf{X}$, represents a matrix. A letter in normal font with a single subscript, refers to a single element from a vector, for example x_i would be the ith element in vector $\mathbf{x}$, whereas two subscripts refer to a single element from a matrix, for example x_{ij} would be the element in the ith row and jth column of matrix $\mathbf{X}$. When referring to a row of a matrix, $\mathbf{X}$, then a subscript is used with a bold-face lower-case letter, for example the ith row would be $\mathbf{x}_i$, in contrast the jth column would be written as $\mathbf{x}_{:j}$. Calligraphic letters, $\mathcal{X}$, are used to denote sets.

A matrix will always be defined with its dimensions using the notation, $\mathbf{X} \in \mathcal{X}^{n \times p}$, or if $\mathcal{X}$ is the set of Reals, it may be written as "$\mathbf{X}$ is a $n \times p$ Real-valued matrix". By default, a 'vector' will refer to a column vector, which may be thought of as a matrix with n rows and one column, and may be represented as:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Vectors are usually defined using transpose notation, for example the vector above may instead be written as $\mathbf{x}^\top = (x_1\ x_2 \cdots x_n)$ or $\mathbf{x} = (x_1\ x_2 \cdots x_n)^\top$. Vectors may also be defined in a shortened format as, $\mathbf{x} \in \mathcal{X}^n$, which implies a vector of length n with elements as represented above.

Typically, a 'hat', $\hat{x}$, will refer to the prediction or estimation of a variable, x, with bold-face used again to represent vectors. A 'bar', $\bar{x}$, refers to the sample mean of $\mathbf{x}$. Capital letters in normal font, X, refer to scalar or vector random variables, which will be made clear from context. $\mathbb{E}(X)$ and $\text{Var}(X)$ are the expectation and variance of the random variable X respectively. A function f, will either be written as a formal map of domain to codomain, $f : \mathcal{X} \to \mathcal{Y}; (x, y) \mapsto f(x, y)$ (which is most useful for understanding inputs and outputs), or more simply and commonly as $f(x, y)$. Given a random variable, X, following distribution ζ (mathematically written $X \sim \zeta$), then f_X denotes the probability density function, and analogously for other distribution defining functions. In the survival analysis context (Chapter 4), a subscript "0" refers to a "baseline" function, for example, S_0 is the baseline survival function.

Common variables and acronyms used in the book are given in Table 0.1 and Table 0.2 respectively.

Table 0.1: Common variables used throughout the book.

Variable	Definition
$\mathbb{R}, \mathbb{R}_{>0}, \mathbb{R}_{\geq 0}, \bar{\mathbb{R}}$	Set of Reals, positive Reals (excl. zero), non-negative Reals (incl. zero), and Reals including $\pm\infty$.
$\mathbb{N}_{>0}$	Set of Naturals excluding zero.
$(\mathbf{X}, \mathbf{t}, \delta)$	Survival data where $\mathbf{X}$ is an $n \times p$ matrix of features, $\mathbf{t}$ is a vector of observed outcome times, and δ is a vector of observed outcome indicators.
β	Vector of model coefficients/weights.
η	Linear predictor, $X\beta$.
$\mathcal{D}, \mathcal{D}_{train}, \mathcal{D}_{test}$	Dataset, training data, and testing data.

Table 0.2: Common acronyms used throughout the book.

Acronym	Definition
AFT	Accelerated Failure Time
cdf	Cumulative Distribution Function
chf	Cumulative Hazard Function
CPH	Cox Proportional Hazards
GBM	Gradient Boosting Machine
GLM	Generalised Linear Model
IPC(W)	Inverse Probability of Censoring (Weighted)
ML	Machine Learning
pdf	Probability Density Function
PH	Proportional Hazards
(S)SVM	(Survival) Support Vector Machine
t.v.i.	Taking Values In

1

Introduction

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

TODO

- Mention somewhere that SA can be used to solve T-year prediciton problems (i.e., see if we can get classif users over to SA).
- Also SA can be used for censoring/truncation

Writing after a global pandemic, applications of survival analysis are more relevant than ever. Predicting the time from onset of COVID-19 symptoms to hospitalisation, or the time from hospitalisation to intubation, or intubation to death, are all time-to-event predictions that are at the centre of survival analysis. As well as morbid applications, survival analysis predictions may be concerned with predicting the time until a customer cancels their gym membership, or the lifetime of a lightbulb; any event that is guaranteed (or at least very likely) to occur can be modelled by a survival analysis prediction. As these predictions can be so sensitive, for example a model predicting when a child should be taken off breathing support (Data Study Group Team 2020), the best possible predictions, evaluated to the highest standard, are a necessity. In other fields of predictive modelling, machine learning has made incredible breakthroughs (such as AlphaFold), therefore applying machine learning to survival analysis is a natural step in the evolution of an important field.

Survival analysis is the field of Statistics focusing on modelling the distribution of an event, which may mean the time until the event takes place, the risk of the event happening, the probability of the event occurring at a single time, or the event's underlying probability distribution. Survival analysis ('survival') is a unique field of study in Statistics as it includes the added difficulty of 'censoring'. Censoring is best described through example: a study is conducted to determine the mortality rate of a group of patients after diagnoses with a particular disease. If a patient dies during this study then their outcome is 'death' and their time of death can be recorded. However if a patient drops-out of the study before they die, then their time of death (though guaranteed to occur) is unknown and the only available information is the time at which they left the study. This patient is now said to be *censored* at the time they drop out. The censoring mechanism allows as much outcome information (time and event) to be captured as possible for all patients (observations).

Machine learning (ML) is the field of Statistics primarily concerned with building models to either predict outputs from inputs or to learn relationships from data (Hastie, Tibshirani, and Friedman 2001; James et al. 2013). This book is limited to the former case, or

more specifically supervised learning, as this is the field in which the vast majority of survival problems live. Relative to other areas of supervised learning, development in survival analysis has been slow – the majority of developments in machine learning for survival analysis have only been in the past decade (see chapters (**?@sec-review**)-(Chapter 5)). This appears to have resulted in less interest in the development of machine learning survival models (**?@sec-review**), less rigour in the evaluation of such models (Chapter 5), and fewer off-shelf/open-source implementations (R. Sonabend et al. 2021). This book seeks to set the foundations for clear workflows, good practice, and precise results for 'machine learning survival analysis'.

1.1 Why is this book needed?

Firstly, whilst there are many books dedicated to regression and classification as machine learning problems (the 'bibles' of machine learning focus entirely on regression and classification only (Bishop 2006; Hastie, Tibshirani, and Friedman 2001; James et al. 2013)), there is a deficit of books covering the survival analysis setting. By writing this book we hope to fill this gap and enable more practitioners to use cutting-edge methods in survival analysis. Survival analysis has important applications in healthcare, finance, engineering and more, all fields that directly impact upon individual lives on a day-to-day basis, and should perhaps be considered as important as classification and regression. The result of this gap in interest, is the erroneous assumption that one field can be directly applied to another. For example there is evidence of researchers treating censoring as a nuisance to be ignored and using regression models instead (Schwarzer, Vach, and Schumacher 2000). Censoring is indeed a challenge and may contribute to making survival analysis less accessible than other fields, but this need not be the case; a clear unification of terminology and presentation of methods may help make 'machine learning survival analysis' more accessible. Added accessibility could lead to more academics (and non-academics) engaging with the field and promoting good standards of practice, as well as developing more novel models and measures.

Where survival models have been developed, these have skewed towards 'ranking models', which predict the relative risk of an event occurring (Section 4.3). In many applications these predictions are sufficient, for example in randomised control trials if assessing the increased/decreased risk of an event after treatment. However, there are many use-cases where predicting an individual's survival probability distribution is required. Take, for example, an engineer calculating the lifetime of a plane's engine.[1] There are three important reasons to replace a jet engine at the optimal time:

- financial: jet engines are very expensive and replacing one sooner than required is a waste of money;
- environmental: an engine being replaced too early is a waste of potential usage;
- safety: if the engine is replaced too late then there is a risk to passengers.

Now consider examples for the three possible 'prediction types' the engineer can make:

i. A 'relative risk prediction': This engine is twice as likely to fail as another.
ii. A 'survival time prediction': The engine is expected to fail in 30 days.

[1] In this engineering context, survival analysis is usually referred to as reliability analysis.

iii. A 'survival distribution prediction': The lifetime of the engine is distributed according to the probability distribution ζ.

The first prediction type is not useful as the underlying relative risk may be unknown and the engineer is concerned with the individual lifetime. The second prediction type provides a useful quantity for the engineer to work with however there is no uncertainty captured in this prediction. The third prediction type can capture the uncertainty of failure over the entirety of the positive Reals (though usually only a small subset is possible and useful). With this final prediction type, the engineer can create safe decisions: 'replace the engine at time τ, where τ is the time when the predicted probability of survival drops below 60%, $S(\tau) = 0.6$'. There are ethical, economic, and environmental reasons for a good survival distribution prediction and this book considers a distribution prediction to be the most important prediction type.

Evaluating predictions from survival models is of the utmost importance. This is especially important as survival models are often deployed in the public domain, particularly in healthcare. Physical products in healthcare, such as new vaccines, undergo rigorous testing and research in randomised control trials before being publically deployed; the same level of rigour should be expected for the evaluation of survival models that are used in life-and-death situations. Evaluation measures for regression and classification are well-understood with important properties, however survival measures have not undergone the same treatment. For example many survival models are still being evaluated solely with concordance indices that have been repeatedly criticised (Gönen and Heller 2005; Rahman et al. 2017; Schmid and Potapov 2012).

1.2 Reproducibility

This book includes simulations and figures generated in R, the code for any figures or experiments in this book are freely available at https://github.com/RaphaelS1/MLSA under an MIT licence and all content on this website is available under CC BY 4.0. We use `renv` to manage package versions, you can find our lockfile at https://github.com/RaphaelS1/MLSA/blob/main/book/renv.lock.

Part I

Survival Analysis and Machine Learning

2

MLSA From Start to Finish

TODO (150-200 WORDS)

> ⚠ Page coming soon!
>
> **We are working on this page and it will be available soon!**

3

Statistical Learning

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

TODO

- $\mathcal{D}$ - labeled data set with n observation
- Observations: $(\mathbf{x}^{(i)}, y^{(i)})$ where $\mathbf{x}^{(i)} \in \mathcal{X}$ is a p-dimensional feature vector and $y^{(i)} \in \mathcal{Y}$ is a label
- Data set: $\mathcal{D} = ((\mathbf{x}^{(1)}, y^{(1)}), ..., (\mathbf{x}^{(n)}, y^{(n)}))$
- Assume $\mathcal{D} \overset{i.i.d.}{\sim} (\mathbb{P}_{xy})^n$ (unknown distribution)
- ML model: $f : \mathcal{X} \to \mathbb{R}^g$ assigning a prediction in $\mathbb{R}^g$
- ML learners, $\mathcal{I}$, configured by hyperparameters $\lambda \in \Lambda$, are $\mathcal{I} : \mathbb{D} \times \Lambda \to \mathcal{H}, (\mathcal{D}, \lambda) \mapsto \hat{f}$ where $\mathcal{H}$ is the function space to which a model belongs and $\hat{f}$ is a trained model with learned hyperparameters $\hat{\theta} \in \mathcal{H}$.
- Models are evaluated with loss functions which measure the discrepancy between predictions and true values $L : \mathcal{Y} \times \mathbb{R}^g \to \mathbb{R}, (\hat{f}(\mathbf{x}), \mathbf{y}) \mapsto l(\hat{f}(\mathbf{x}), \mathbf{y})$
- To prevent overfitting, models are evaluated on unseen test data to ensure unbiased performance estimation and the generalization error $GE(\mathcal{I}, \lambda, n_{train}, l) := \lim_{n_{test} \to \infty} \mathbb{E}_{\mathcal{D}_{train}, \mathcal{D}_{test} \sim \mathbb{P}_{xy}}[l(\mathbf{y}_{test}, \mathbf{F}_{\mathcal{D}_{test}, \mathcal{I}(\mathcal{D}_{train}, \lambda)})]$ where $\mathbf{F}_{\mathcal{D}_{test}, \mathcal{I}(\mathcal{D}_{train}, \lambda)}$ is the matrix of predictions from a model trained on $\mathcal{D}_{train}$ and making predictions on $\mathcal{D}_{test}$.

3.1 Machine Learning

This section begins with a very brief introduction to machine learning and a focus on regression and classification; the survival machine learning task is then introduced (Section 4.4). Of the many fields within machine learning (ML), the scope of this book is narrowed to supervised learning. Supervised learning is the sub-field of ML in which predictions are made for outcomes based on data with observed dependent and independent variables. For example predicting someone's height is a supervised learning problem as data can be collected

for features (independent variables) such as age and sex, and outcome (dependent variable), which is height. Predictive survival analysis problems fall naturally in the supervised learning framework as there are identifiable features and (multiple types of) outcomes.

3.1.1 Terminology and Methods

Common supervised learning methods are discussed in a simplified setting with features X *t.v.i.* $\mathcal{X}$ and outcomes Y *t.v.i.* $\mathcal{Y}$; usually outcomes are referred to as 'targets' (a 'target for prediction'). Let $\mathcal{D}_0 = \{(X_1, Y_1), ..., (X_n, Y_n)\}$ be a (training) dataset where $(X_i, Y_i) \overset{i.i.d.}{\sim} (X, Y)$. The methods below extend naturally to the survival setting.

Strategies and Models

In order to clearly separate between similar objects, several terms for machine learning are now introduced and clearly distinguished.

Let $g : \mathcal{X} \to \mathcal{Y}$ be the true (but unknown) mapping from the features to outcomes, referred to as the *true prediction functional*. Let $\mathcal{G}$ be the set of *prediction functionals* such that $\forall \Upsilon \in \mathcal{G}, \Upsilon : \mathcal{X} \to \mathcal{Y}$. A *learning* or *fitting algorithm* is defined to be any function of the form $\mathcal{A} : \mathcal{X}^n \times \mathcal{Y}^n \to \mathcal{G}$. The goal of supervised learning is to *learn* g with a learning algorithm *fit* on (i.e. the input to the algorithm is) training data, $\hat{g} := \mathcal{A}(\mathcal{D}_{train}) \in \mathcal{G}$. Note that $\hat{g}$ may take hyper-parameters that can be set or tuned (see below). The learning algorithm is 'good' if $\hat{g}(X) \approx g(X)$ (see 'Evaluation' below).

The learning algorithm is determined by the chosen *learning strategy* and *model*, where a model is a complete specification of a learning strategy including hyper-parameters. These terms are more clearly illustrated by example:

i. Learning strategy – simple linear regression
ii. Model – $y = \beta_0 + \beta_1 x$ where $x \in \mathbb{R}$ is a single covariate, $y \in \mathbb{R}$ is the target, and $\beta_0, \beta_1 \in \mathbb{R}$ are model coefficients.
iii. Learning algorithm (model fitting) – Minimise the residual sum of squares: $(\hat{\beta}_0, \hat{\beta}_1) := \text{argmin}_{\beta_0, \beta_1} \{\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2\}$ for $(x_i, y_i) \in \mathcal{D}_{train}, i = 1, ..., n$.
iv. Prediction functional – $\hat{g}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$

To further illustrate the difference between learning strategy and model, note that the same learning strategy 'simple linear regression' could either utilise the model above or instead a model without intercept, $y = \beta x$, in which case the learning algorithm and prediction functional would also be modified.

The model in (ii) is called *unfitted* as the model coefficients are unknown and the model cannot be used for prediction. After step (iii) the model is said to be fit to the training data and therefore the model is *fitted*.[1] It is common to refer to the learning algorithm (and associated hyper-parameters) as the unfitted model and to refer to the prediction functional (and associated hyper-parameters) as the fitted model.

[1] The terms 'fitted' and 'unfitted' are used instead of 'fit' and 'unfit' to prevent confusion with words such as 'suitable' and 'unsuitable'.

Evaluation

Models are *evaluated* by evaluation measures called *losses* or *scores*,[2] $L : \mathcal{Y} \times \mathcal{Y} \to \bar{\mathbb{R}}$. Let $(X^*, Y^*) \sim (X, Y)$ be test data (i.e. independent of $\mathcal{D}_{train}$) and let $\hat{g} : \mathcal{X} \to \mathcal{Y}$ be a prediction functional fit on $\mathcal{D}_{train}$, then these evaluation measures determine how closely predictions, $\hat{g}(X^*)$, relate to the truth, Y^*, thereby providing a method for determining if a model is 'good'.[3]

Task

A machine learning *task* is a simple mechanism to outline the problem of interest by providing: i) the data specification; ii) the definition of learning; iii) the definition of success (when is a prediction 'good'?) (Király et al. 2021). All tasks in this paper have the same definitions of learning and success. For (ii), the aim is to learn the true prediction functional, g, by fitting the learning algorithm on training data, $\hat{g} := \mathcal{A}(\mathcal{D}_0)$. For (iii), a predicted functional is considered 'good' if the *expected generalization error*, $\mathbb{E}[L(Y^*, \hat{g}(X^*))]$, is low, where $(X^*, Y^*) \sim (X, Y)$ is independent of the training data $\mathcal{D}_0$, and L is some loss that is chosen according to the domain of interest (regression, classification, survival).

Resampling

Models are *tested* on their ability to make predictions. In order to avoid 'optimism of training error' (James et al. 2013) – overconfidence caused by testing the model on training data – models are tested on previously unseen or 'held-out' data. *Resampling* is the procedure of splitting one dataset into two or more for separated training and testing. In this paper only two resampling methods are utilised: *holdout* and *cross-validation*. Holdout is the process of splitting a primary dataset into training data for model fitting and testing data for model predicting. This is an efficient method but may not accurately estimate the expected generalisation error for future model performance, instead this is well-estimated by K-fold cross-validation (KCV) (Hastie, Tibshirani, and Friedman 2001). In KCV, data is split into $K \in \mathbb{N}_{>0}$ 'folds' such that $K - 1$ of the folds are used for model training and the final Kth fold for testing. The testing fold is iterated over all K folds, so that each at some point is used for testing and then training (though never at the same time). In each iteration the model is fit on the training folds, and predictions are made and evaluated on the testing fold, giving a loss $L_k := L(\hat{g}(X^k), Y^k)$, where (X^k, Y^k) are data from the kth fold. A final loss is defined by, $L^* := \frac{1}{K} \sum_{k=1}^{K} L_k$. Commonly $K = 5$ or $K = 10$ (Breiman and Spector 1992; Kohavi 1995).

Model Performance Benchmarking

Whilst *benchmarking* often refers to speed tests, i.e. the time taken to complete an operation, it can also refer to any experiment in which objects (mathematical or computational) are compared. In this report, a benchmark experiment will either refer to the comparison of multiple models' predictive abilities, or comparison of computational speeds and object sizes for model fitting; which of these will be clear from context.

[2]The term 'loss' is usually utilised to refer to evaluation measures to be minimised, whereas 'scores' should be maximised, this is returned to in (@sec-eval).

[3]Here evaluation refers specifically to predictive ability; other forms of evaluation and further discussion of the area are provided in (@sec-eval).

Model Comparison

Models can be analytically compared on how well they make predictions for new data. Model comparison is a complex topic with many open questions (Demšar 2006; Dietterich 1998; Nadeau and Bengio 2003) and as such discussion is limited here. When models are compared on multiple datasets, there is more of a consensus in how to evaluate models (Demšar 2006) and this is expanded on further in (R. E. B. Sonabend 2021). Throughout this book there are small simulation experiments for model comparison on single datasets however as these are primarily intended to aid exposition and not to generalise results, it suffices to compare models with the conservative method of constructing confidence intervals around the sample mean and standard error of the loss when available (Nadeau and Bengio 2003).

Hyper-Parameters and Tuning

A *hyper-parameter* is a model parameter that can be set by the user, as opposed to coefficients that are estimated as part of model fitting. A hyper-parameter can be set before training, or it can be tuned. *Tuning* is the process of choosing the optimal hyper-parameter value via automation. In the simplest setting, tuning is performed by selecting a range of values for the hyper-parameter(s) and treating each choice (combination) as a different model. For example if tuning the number of trees in a random forest (Section 13.1), m_r, then a range of values, say $100, 200, 500$ are chosen, and three models $m_{r100}, m_{r200}, m_{r500}$ are benchmarked. The optimal hyper-parameter is given by whichever model is the best performing. *Nested resampling* is a common method to prevent overfitting that could occur from using overlapping data for tuning, training, or testing. Nested resampling is the process of resampling the training set again for tuning.

3.1.2 Machine Learning in Classification and Regression

Before introducing machine learning for survival analysis, which is considered 'non-classical', the more standard classification and regression set-ups are provided; these are referenced throughout this book.

3.1.2.1 Classification

Classification problems make predictions about categorical (or discrete) events, these may be *deterministic* or *probabilistic*. Deterministic classification predicts which category an observation falls into, whereas probabilistic classification predicts the probability of an observation falling into each category. In this brief introduction only binary single-label classification is discussed, though the multi-label case is considered in **??**. In binary classification, there are two possible categories an observation can fall into, usually referred to as the 'positive' and 'negative' class. For example predicting the probability of death due to a virus is a probabilistic classification task where the 'positive' event is death.

A probabilistic prediction is more informative than a deterministic one as it encodes uncertainty about the prediction. For example it is clearly more informative to predict a 70 chance of rain tomorrow instead of simply 'rain'. Moreover the latter prediction implicitly contains an erroneous assumption of certainty, e.g. 'it will rain tomorrow'.

Classification Task

Box 3.1. *Let (X, Y) be random variables t.v.i. $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{Y} = \{0, 1\}$. Then,*

- *The* probabilistic classification task *is the problem of predicting the probability of a single event taking place and is specified by $g : \mathcal{X} \to [0, 1]$.*
- *The* deterministic classification task *is the problem of predicting if a single event takes place and is specified by $g : \mathcal{X} \to \mathcal{Y}$.*

The estimated prediction functional $\hat{g}$ is fit on training data $\backslash(X_1, Y_1), ..., (X_n, Y_n) \overset{i.i.d.}{\sim} (X, Y)$ and is considered 'good' if $\mathbb{E}[L(Y^*, \hat{g}(X^*))]$ is low, where $(X^*, Y^*) \sim (X, Y)$ is independent of $(X_1, Y_1), ..., (X_n, Y_n)$ and $\hat{g}$.
In the probabilistic case, the prediction $\hat{g}$ maps to the estimated probability mass function $\hat{p}_Y$ such that $\hat{p}_Y(1) = 1 - \hat{p}_Y(0)$.

3.1.2.2 Regression

A regression prediction is one in which the goal is to predict a continuous outcome from a set of features. For example predicting the time until an event (without censoring) occurs, is a regression problem.

Regression Task

Box 3.2. *Let (X, Y) be random variables t.v.i. $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{Y} \subseteq \mathbb{R}$. Let $\mathcal{S} \subset \text{Distr}(\mathcal{Y})$ be a convex set of distributions on $\mathcal{Y}$. Then,*

- *The* probabilistic regression task *is the problem of predicting a conditional distribution over the Reals and is specified by $g : \mathcal{X} \to \mathcal{S}$.*
- *The* deterministic regression task *is the problem of predicting a single continuous value in the Reals and is specified by $g : \mathcal{X} \to \mathcal{Y}$.*

The estimated prediction functional $\hat{g}$ is fit on training data $\backslash(X_1, Y_1), ..., (X_n, Y_n) \overset{i.i.d.}{\sim}$

> *(X, Y) and is considered 'good' if $\mathbb{E}[L(Y^*, \hat{g}(X^*))]$ is low, where $(X^*, Y^*) \sim (X, Y)$ is independent of $(X_1, Y_1), ..., (X_n, Y_n)$ and $\hat{g}$.*

Whilst regression can be either probabilistic or deterministic, the latter is much more common and therefore in this book 'regression' refers to the deterministic case unless otherwise stated.

4

Survival Analysis

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

TODO

- Make sure intro is clear about censoring/truncation and that metrics can't highlight if this is setup wrong - analogously to hypothesis testing not testing the result but hypothesis, p-hacking, etc.
- If measures for right-censoring used in parts of pipelines hard to discern biases if wrong type of measure used
- Same as dependent/independent censoring and measures problem

---

In their broadest and most basic definitions, survival analysis is the study of temporal data from a given origin until the occurrence of one or more events or 'end-points' (Collett 2014), and machine learning is the study of models and algorithms that learn from data in order to make predictions or find patterns (Hastie, Tibshirani, and Friedman 2001). Reducing either field to these definitions is ill-advised.

This chapter collects terminology utilised in survival analysis (Section 4.1) and machine learning (Section 3.1) in order that this book can cleanly discuss 'machine learning survival analysis' (Section 4.4). Once the mathematical setting is set up, the book scope is fully presented in (Section 4.2). Whilst the content of this chapter is not novel with respect to either survival analysis or machine learning separately, this does appear to be the first formulation of the survival analysis machine learning 'task' (Király et al. 2021).

4.1 Survival Analysis

Survival analysis is the field of Statistics concerned with the analysis of time-to-event data, which consists of covariates, a categorical (often binary) outcome, and the time until this outcome takes place (the 'survival time'). As a motivating example of time-to-event data, say 100 patients are admitted to a COVID-19 ward and for each patient the following

covariate data are collected: age, weight and sex; additionally for each patient the time until death or discharge is recorded. In the time-to-event dataset, which takes a standard tabular form, each of the 100 patients is a row, with columns consisting of age, weight, and sex measurements, as well as the outcome (death or discharge) and the time to outcome.

Survival analysis is distinct from other areas of Statistics due to the incorporation of 'censoring', a mechanism for capturing uncertainty around when an event occurs in the real-world. Continuing the above example, if a patient dies of COVID-19 five dies after admittance, then their outcome is exactly known: they *died* after five days. Consider now a patient who is discharged after ten days. As death is a guaranteed event they have a true survival time but this may be decades later, therefore they are said to be *censored* at ten days. This is a convenient method to express that the patient survives up to ten days and their survival status at any time after this point is unknown. Censoring is a unique challenge to survival analysis that attempts to incorporate as much information as possible without knowing the true outcome. This is a 'challenge' as statistical models usually rely on learning from observed, i.e. known, outcome data; therefore censoring requires special treatment.

Whilst survival analysis occurs in many fields, for example as 'reliability analysis' in engineering and 'duration analysis' in economics, in this book the term 'survival' will always be used. Moreover the following terminology, analogous to a healthcare setting, are employed: survival analysis (or 'survival' for short) refers to the field of study; the event of interest is the 'event', or 'death'; an observation that has not experienced an event is 'censored' or 'alive'; and observations are referred to as 'observations', 'subjects', or 'patients'.

Some of the biggest challenges in survival analysis stem from an unclear definition of a 'survival analysis prediction' and different (sometimes conflicting) common notations. This book attempts to make discussions around survival analysis clearer and more precise by first describing the mathematical setting for survival analysis in (Section 4.1.1) and only then defining the prediction types to consider in (Section 4.3).

4.1.1 Survival Data and Definitions

Survival analysis has a more complicated data setting than other fields as the 'true' data generating process is not directly modelled but instead engineered variables are defined to capture observed information. Let,

- X *t.v.i.* $\mathcal{X} \subseteq \mathbb{R}^p, p \in \mathbb{N}_{>0}$ be the generative random variable representing the data *features/covariates/independent variables.*
- Y *t.v.i.* $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$ be the (unobservable) *true survival time.*
- C *t.v.i.* $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$ be the (unobservable) *true censoring time.*

It is impossible to fully observe both Y and C. This is clear by example: if an observation drops out of a study then their censoring time is observed but their event time is not, whereas if an observation dies then their true censoring time is unknown. Hence, two engineered variables are defined to represent observable outcomes. Let,

- $T := \min\{Y, C\}$ be the *observed outcome time.*
- $\Delta := \mathbb{I}(Y = T) = \mathbb{I}(Y \leq C)$ be the *survival indicator* (also known as the *censoring* or *event* indicator).[1]

Together (T, Δ) is referred to as the *survival outcome* or *survival tuple* and they form the dependent variables. The survival outcome provides a concise mechanism for representing

[1] Indicators are usually named to reflect a positive condition in the function (in this case the event when $Y = T$), but counter to this convention the 'censoring indicator' is possibly the most common term.

the time of the *observed* outcome and indicating which outcome (death or censoring) took place.

Now the full generative template for survival analysis is given by $\setminus\ (X, \Delta, C, Y, T)$ *t.v.i.* $\mathcal{X} \times \{0,1\} \times \mathcal{T} \times \mathcal{T} \times \mathcal{T}$ and with $(X_i, \Delta_i, C_i, Y_i, T_i)$ jointly i.i.d. A *survival dataset* is defined by $\mathcal{D} = \{(X_1, T_1, \Delta_1), ..., (X_n, T_n, \Delta_n)\}$ where $(X_i, T_i, \Delta_i) \overset{i.i.d.}{\sim} (X, T, \Delta)$ and X_i is a p-vector, $X_i = (X_{i;1}, ..., X_{i;p})$. Though unobservable, the true outcome times are defined by $(Y_1, C_1), ..., (Y_n, C_n)$ where $(Y_i, C_i) \overset{i.i.d.}{\sim} (Y, C)$.

(1) exemplifies a random survival dataset with n observations (rows) and p features.

Table 4.1: Theoretical time-to-event dataset. (Y, C) are 'hypothetical' as they can never be directly observed. Rows are individual observations, X columns are features, T is observed time-to-event, Δ is the censoring indicator, and (Y, C) are hypothetical true survival and censoring times.

X	X	X	T	Δ	Y	C
X_{11}	$\cdots$	X_{1p}	T_1	Δ_1	Y_1	C_1
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
X_{n1}	$\cdots$	X_{np}	T_n	Δ_n	Y_n	C_n

(2) exemplifies an observed survival dataset with a modified version of the `rats` dataset (Therneau 2015).

Table 4.2: `rats` (Therneau 2015) time-to-event dataset with added hypothetical columns (Y, C). Rows are individual observations, X columns are features, T is observed time-to-event, Δ is the censoring indicator, and (Y, C) are hypothetical (here arbitrary values dependent on (T, Δ)) true survival and censoring times.

litter ($X_{\cdot;1}$)	**rx** ($X_{\cdot;2}$)	**sexF** ($X_{\cdot;3}$)	**time** (T)	**status** (Δ)	**survTime** (Y)	**censTime** (C)
1	1	1	101	0	105	101
1	0	1	49	1	49	55
1	0	1	104	0	200	104
2	1	0	91	0	92	91
2	0	0	104	1	104	104
2	0	0	102	1	102	120

Both datasets includes two extra columns, on the right of the triple vertical line, which imagine hypothetical data for the unobserved true survival and censoring times.

Finally the following terms are used frequently throughout this report. Let $(T_i, \Delta_i) \overset{i.i.d.}{\sim} (T, \Delta), i = 1, ..., n$, be random survival outcomes. Then,

- The *set of unique* or *distinct time-points* refers to the set of time-points in which at least one observation dies or is censored, $\mathcal{U}_O \coloneqq \{T_i\}_{i \in \{1,...,n\}}$.
- The *set of unique death times* refers to the set of unique time-points in which death (and not censoring) occurred, $\mathcal{U}_D \coloneqq \{T_i : \Delta_i = 1\}_{i \in \{1,...,n\}}$.

- The *risk set* at a given time-point, τ, is the set of subjects who are known to be alive (not dead or censored) just before that time, $\mathcal{R}_\tau := \{i : T_i \geq \tau\}$ where i is a unique row/subject in the data.
- The *number of observations alive* at τ is the cardinality of the risk set, $|\mathcal{R}_\tau|$, and is denoted by $n_\tau := \sum_i \mathbb{I}(T_i \geq \tau)$.
- The *number of observations who die* at τ is denoted by $d_\tau := \sum_i \mathbb{I}(T_i = \tau, \Delta_i = 1)$.
- The Kaplan-Meier estimate of the average survival function of the training data *survival distribution* is the Kaplan-Meier estimator (Section 11.1.1) fit (Section 3.1.1) on training data (T_i, Δ_i) and is denoted by $\hat{S}_{KM}$.
- The Kaplan-Meier estimate of the average survival function of the training data *censoring distribution* is the Kaplan-Meier estimator fit on training data $(T_i, 1 - \Delta_i)$ and is denoted by $\hat{G}_{KM}$.

Notation and definitions will be recapped at the start of each chapter for convenience.

4.1.2 Censoring

Censoring is now discussed in more detail and important concepts introduced. Given the survival generating process (X, T, Δ) with unobservable (Y, C), the event is experienced if $Y \leq C$ and $\Delta = 1$ or censored if $\Delta = 0$.

Censoring 'Location' {.unnumbered .unlisted}

Right-censoring is the most common form of censoring in survival models and it occurs either when a patient drops out (but doesn't experience the event) of the study before the end and thus their outcome is unknown, or if they experience the event at some unknown point after the study end. Formally let $[\tau_l, \tau_u]$ be the study period for some, $\tau_l, \tau_u \in \mathbb{R}_{\geq 0}$. Then right-censoring occurs when either $Y > \tau_u$ or when $Y \in [\tau_l, \tau_u]$ and $C \leq Y$. In the first case $T = C = \tau_u$ and censoring is due to the true time of death being unknown as the observation period has finished. In the latter case, a separate censoring event, such as drop-out or another competing risk, is observed.

Left-censoring is a rarer form of censoring and occurs when the event happens at some unknown time before the study start, $Y < \tau_l$. Interval-censoring occurs when the event takes place in some interval within the study period, but the exact time of event is unknown. (Figure 4.1) shows a graphical representation of right-censoring.

Censoring 'Dependence'

Censoring is often defined as *uninformative* if $Y \perp\!\!\!\perp C$ and *informative* otherwise however these definitions can be misleading as the term 'uninformative' appears to be imply that censoring is independent of both X and Y, and not just Y. Instead the following more precise definitions are used in this report.

Definition 4.1 (Censoring)**.** Let (X, T, Δ, Y, C) be defined as above, then

- If $C \perp\!\!\!\perp X$, censoring is *feature-independent*, otherwise censoring is *feature-dependent*.
- If $C \perp\!\!\!\perp Y$, then censoring is *event-independent*, otherwise censoring is *event-dependent*.
- If $(C \perp\!\!\!\perp Y)|X$, censoring is conditionally independent of the event given covariates, or *conditionally event-independent*.
- If $C \perp\!\!\!\perp (X, Y)$ censoring is *uninformative*, otherwise censoring is *informative*.

Non-informative censoring can generally be well-handled by models as true underlying patterns can still be detected and the reason for censoring does not affect model inference

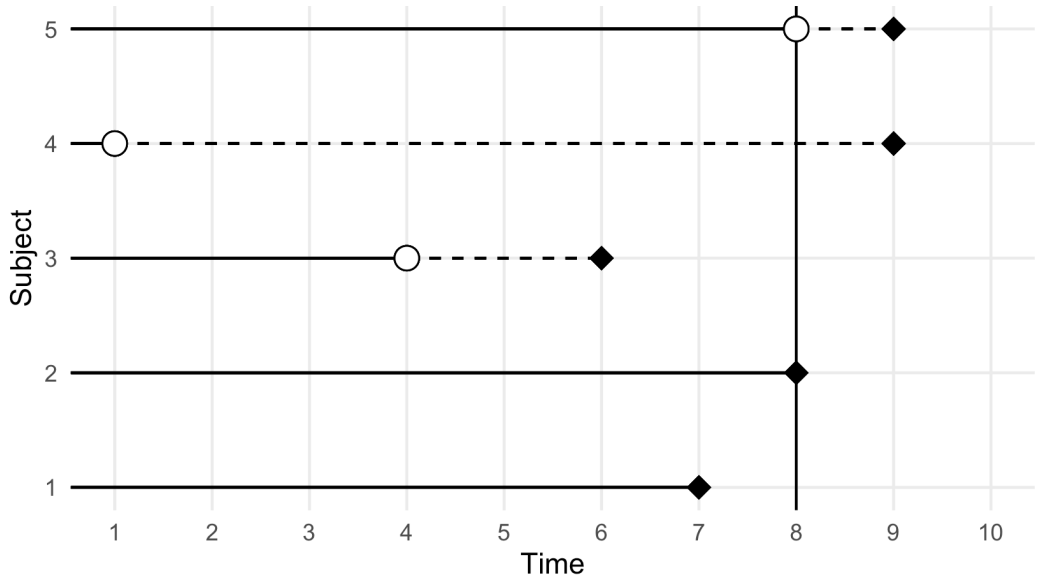


Figure 4.1: Dead and censored subjects (y-axis) over time (x-axis). Black diamonds indicate true death times and white circles indicate censoring times. Vertical line is the study end time. Subjects 1 and 2 die in the study time. Subject 3 is censored in the study and (unknown) dies within the study time. Subject 4 is censored in the study and (unknown) dies after the study. Subject 5 dies after the end of the study.

or predictions. However in the real-world, censoring is rarely non-informative as reasons for drop-out or missingness in outcomes tend to be related to the study of interest. Event-dependent censoring is a tricky case that, if not handled appropriately (by a competing-risks framework), can easily lead to poor model development; the reason for this can be made clear by example: Say a study is interested in predicting the time between relapses of stroke but a patient suffers a brain aneurysm due to some separate neurological condition, then there is a high possibility that a stroke may have occurred if the aneurysm had not. Therefore a survival model is unlikely to distinguish the censoring event (aneurysm) from the event of interest (stroke) and will confuse predictions. In practice, the majority of models and measures assume that censoring is conditionally event-independent and hence censoring can be predicted by the covariates whilst not directly depending on the event. For example if studying the survival time of ill pregnant patients in hospital, then dropping out of the study due to pregnancy is clearly dependent on how many weeks pregnant the patient is when the study starts (for the sake of argument assume no early/late pregnancy due to illness).

Type I Censoring

Type I and Type II censoring are special-cases of right-censoring, only Type I is discussed in this book as it is more common in simulation experiments. Type I censoring occurs if a study has a set end-date, or maximum survival time, and a patient survives until the end of the study. If survival times are dependent on covariates (i.e. not random) and the study start date is known (or survival times are shifted to the same origin) then Type I censoring will usually be informative as censored patients will be those who survived the longest.

4.2 Book Scope

Now that the mathematical setting has been defined, the book scope is provided. For time and relevance the scope of this book is narrowed to the most parsimonious setting that is genuinely useful in modelling real-world scenarios. This is the setting that captures all assumptions made by the majority of proposed survival models and therefore is practical both theoretically and in application. This setting is defined by the following assumptions (with justifications):

- Let p be the proportion of censored observations in the data, then $p \in (0, 1)$. This open interval prevents the case when $p = 0$, which is simply a regression problem (Section 3.1.2.2), or the case when $p = 1$, in which no useful models exist (as the event never occurs).
- Only right-censoring is observed in the data, no left- or interval-censoring. This accurately reflects most real-world data in which observations that have experienced the event before the study start (left-censoring) are usually not of interest, and close monitoring of patients means that interval-censoring is unlikely in practice. It is acknowledged that left-truncation is a common problem in medical datasets though this is often handled not by models but by data pre-processing, which is not part of the workflow discussed in this book.
- There is only one event of interest, an observation that does not experience this event is censored. This eliminates the 'competing risk' setting in which multiple events of interest can be modelled.
- The event can happen at most once. For example the event could be death or initial diagnosis of a disease however cannot be recurrent such as seizure. In the case where the event could theoretically happen multiple times, only the time to one (usually the first) occurrence of the event is modelled.
- The event is guaranteed to happen at least once. This is an assumption implicitly made by all survival models as predictions are for the time until the true event, Y, and not the observed outcome, T.

For both the multi-event and recurrent-event cases, simple reductions exist such that these settings can be handled by the models discussed in this paper however this is not discussed further here.

No assumptions are made about whether censoring is dependent on the data but when models and measures make these assumptions, they will be explicitly discussed.

The purpose of any statistical analysis is dependent on the research question. For example techniques are available for data analysis, imputation, exploration, prediction, and more. This book focuses on the predictive setting; other objectives, such as model inspection and data exploration can be achieved post-hoc via interpretable machine learning techniques (Molnar 2019).

Finally, the methods in this book are restricted to frequentist statistics. Bayesian methods are not discussed as the frequentist setting is usually more parsimonious and additionally there are comparatively very few off-shelf implementations of Bayesian survival methods. Despite this, it is noted that Bayesian methods are particularly relevant to the research in this book, which is primarily concerned with uncertainty estimates and predictions of distributions. Therefore, a natural extension to the work in this book would be to fully explore the Bayesian setting.

4.3 Survival Prediction Problems

This section continues by defining the survival problem narrowed to the scope described in the previous section. Defining a single 'survival prediction problem' (or 'task') is important mathematically as conflating survival problems could lead to confused interpretation and evaluation of models. Let (X, T, Δ) and $\mathcal{D}$ be as defined above. A general survival prediction problem is one in which:

- a survival dataset, $\mathcal{D}$, is split (Section 3.1.1) for training, $\mathcal{D}_{train}$, and testing, $\mathcal{D}_{test}$;
- a survival model is fit on $\mathcal{D}_{train}$; and
- the model predicts some representation of the unknown true survival time, Y, given $\mathcal{D}_{test}$.

The process of 'fitting' is model-dependent, and can range from simple maximum likelihood estimation of model coefficients, to complex algorithms. The model fitting process is discussed in more abstract detail in (Section 3.1) and then concrete algorithms are discussed in (**?@sec-review**). The different survival problems are separated by 'prediction types' or 'prediction problems', these can also be thought of as predictions of different 'representations' of Y. Four prediction types are discussed in this paper, these may be the only possible survival prediction types and are certainly the most common as identified in chapters (**?@sec-review**) and (Chapter 5). They are predicting:

- The *relative risk* of an individual experiencing an event – A single continuous ranking.
- The *time until an event* occurs – A single continuous value.
- The *prognostic index* for a model – A single continuous value.
- An individual's *survival distribution* – A probability distribution.

The first three of these are referred to as *deterministic* problems as they predict a single value whereas the fourth is *probabilistic* and returns a full survival distribution. Definitions of these are expanded on below.

Survival predictions differ from other fields in two respects. Firstly, the predicted outcome, Y, is a different object than the outcome used for model training, (T, Δ). This differs from, say, regression in which the same object (a single continuous variable) is used for fitting and predicting. Secondly, with the exception of the time-to-event prediction, all other prediction types do not predict Y but some other related quantity.

Survival prediction problems must be clearly separated as they are inherently incompatible. For example it is not meaningful to compare a relative risk prediction from one observation to a survival distribution of another. Whilst these prediction types are separated above, they can be viewed as special cases of each other. Both (1) and (2) may be viewed as variants of (3); and (1), (2), and (3) can all be derived from (4); this is elaborated on below.

Relative Risk/Ranking

This is perhaps the most common survival problem and is defined as predicting a continuous rank for an individual's 'relative risk of experiencing the event'. For example, given three patients, $\{i, j, k\}$, a relative risk prediction may predict the 'risk of event' as $\{0.1, 0.5, 10\}$ respectively. From these predictions, the following types of conclusions can be drawn:

- Conclusions comparing patients. e.g. i is at the least risk; the risk of j is only slightly higher than that of i but the risk of k is considerably higher than j; the corresponding ranks for i, j, k, are $1, 2, 3$.

- Conclusions comparing risk groups. e.g. thresholding risks at 1.0 places i and j in a 'low-risk' group and k in a 'high-risk' group

So whilst many important conclusions can be drawn from these predictions, the values themselves have no meaning when not compared to other individuals. Interpretation of these rankings has historically been conflicting in implementation, with some software having the interpretation 'higher ranking implies higher risk' whereas others may indicate 'higher ranking implies lower risk' **??**. In this book, a higher ranking will always imply a higher risk of event (as in the example above).

Time to Event

Predicting a time to event is the problem of predicting the deterministic survival time of a patient, i.e. the amount of time for which they are predicted to be alive after some given start time. Part of the reason this problem is less common in survival analysis is because it borders regression – a single continuous value is predicted – and survival – the handling of censoring is required – but neither is designed to solve this problem directly. Time-to-event predictions can be seen as a special-case of the ranking problem as an individual with a predicted longer survival time will have a lower overall risk, i.e. if t_i, t_j and r_i, r_j are survival time and ranking predictions for patients i and j respectively, then $t_i > t_j \to r_i < r_j$.

Prognostic Index

Given covariates, $x \in \mathbb{R}^{n \times p}$, and a vector of model coefficients, $\beta \in \mathbb{R}^p$, the linear predictor is defined by $\eta := x\beta \in \mathbb{R}^n$. The 'prognostic index' is a term that is often used in survival analysis papers that usually refers to some transformation (possibly identity), ϕ, on the linear predictor, $\phi(\eta)$. Assuming a predictive function (for survival time, risk, or distribution defining function (see below)) of the form $g(\varphi)\phi(\eta)$, for some function g and variables φ where $g(\varphi)$ is constant for all observations (e.g. Cox PH (Section 11.1.2)), then predictions of η are a special case of predicting a relative risk, as are predictions of $\phi(\eta)$ if ϕ is rank preserving. A higher prognostic index may imply a higher or lower risk of event, dependent on the model structure.

Survival Distribution

Predicting a survival distribution refers specifically to predicting the distribution of an individual patient's survival time, i.e. modelling the distribution of the event occurring over $\mathbb{R}_{\geq 0}$. Therefore this is seen as the probabilistic analogue to the deterministic time-to-event prediction, these definitions are motivated by similar terminology in machine learning regression problems (Section 3.1). The above three prediction types can all be derived from a probabilistic survival distribution prediction (Chapter 15).

A survival distribution is a mathematical object that is estimated by predicting a *representation* of the distribution. Let W be a continuous random variable t.v.i. $\mathbb{R}_{\geq 0}$ with probability density function (pdf), $f_W : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, and cumulative distribution function (cdf), $F_W : \mathbb{R}_{\geq 0} \to [0, 1]; (\tau) \mapsto P(W \leq \tau)$. The pdf, $f_W(\tau)$, is the likelihood of an observation dying in a small interval around time τ, and $F_W(\tau) = \int_0^\tau f_W(\tau)$ is the probability of an observation being dead at time τ (i.e. dying at or before τ). In survival analysis, it is generally more interesting to model the risk of the event taking place or the probability of the patient being alive, leading to other distribution representations of interest.

The survival function is defined as

$$S_W : \mathbb{R}_{\geq 0} \to [0,1]; \quad (\tau) \mapsto P(W \geq \tau) = \int_{\tau}^{\infty} f_W(u)\ du$$

and so $S_W(\tau) = 1 - F_W(\tau)$. This function is known as the survival function as it can be interpreted as the probability that a given individual survives until some point $\tau \geq 0$.

Another common representation is the hazard function,

$$h_W : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}; \quad (\tau) \mapsto \frac{f_W(\tau)}{S_W(\tau)}$$

The hazard function is interpreted as the instantaneous risk of death given that the observation has survived up until that point; note this is not a probability as h_W can be greater than one.

The cumulative hazard function (chf) can be derived from the hazard function by

$$H_W : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}; \quad (\tau) \mapsto \int_{0}^{\tau} h_W(u)\ du$$

The cumulative hazard function relates simply to the survival function by

$$H_W(\tau) = \int_0^{\tau} h_W(u)\ du = \int_0^{\tau} \frac{f_W(u)}{S_W(u)}\ du = \int_0^{\tau} -\frac{S'_W(u)}{S_W(u)}\ du = -\log(S_W(\tau))$$

Any of these representations may be predicted conditionally on covariates for an individual by a probabilistic survival distribution prediction. Once a function has been estimated, predictions can be made conditional on the given data. For example if n survival functions are predicted, $\hat{S}_1, ..., \hat{S}_n$, then $\hat{S}_i$ is interpreted as the predicted survival function given covariates of observation i, and analogously for the other representation functions.

4.4 Survival Analysis Task

The survival prediction problems identified in (Section 4.3) are now formalised as machine learning tasks.

Survival Task

Box 4.1. *Let (X, T, Δ) be random variables t.v.i. $\mathcal{X} \times \mathcal{T} \times \{0,1\}$ where $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$. Let $\mathcal{S} \subseteq \text{Distr}(\mathcal{T})$ be a convex set of distributions on $\mathcal{T}$ and let $\mathcal{R} \subseteq \mathbb{R}$. Then,*

- *The* probabilistic survival task *is the problem of predicting a conditional distribution over the positive Reals and is specified by $g : \mathcal{X} \to \mathcal{S}$.*
- *The* deterministic survival task *is the problem of predicting a continuous value in the positive Reals and is specified by $g : \mathcal{X} \to \mathcal{T}$.*
- *The* survival ranking task *is specified by predicting a continuous ranking in the Reals and is specified by $g : \mathcal{X} \to \mathcal{R}$.*

The estimated prediction functional $\hat{g}$ is fit on training data

$|(X_1, T_1, \Delta_1), ..., (X_n, T_n, \Delta_n) \overset{i.i.d.}{\sim} (X, T, \Delta)$ *and is considered 'good' if* $|\mathbb{E}[L(T^*, \Delta^*, \hat{g}(X^*))]$ *is low, where* $(X^*, T^*, \Delta^*) \sim (X, T, \Delta)$ *is independent of* $(X_1, T_1, \Delta_1), ..., (X_n, T_n, \Delta_n)$ *and* $\hat{g}$.

Any other survival prediction type falls within one of these tasks above, for example predicting log-survival time is the deterministic task and predicting prognostic index or linear predictor is the ranking task. Removing the separation between the prognostic index and ranking prediction types is due to them both making predictions over the Reals; their mathematical difference lies in interpretation only. In general, the survival task will assume that $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$, and the terms 'discrete' or 'reduced survival task' will refer to the case when $\mathcal{T} \subseteq \mathbb{N}_{>0}$. Unless otherwise specified, the 'survival task', will be used to refer to the probabilistic survival task.[2]

Survival Analysis and Regression

Survival and regression tasks are closely related as can be observed from their respective definitions. Both are specified by $g : \mathcal{X} \to \mathcal{S}$ where for probabilistic regression $\mathcal{S} \subseteq \text{Distr}(\mathbb{R})$ and for survival $\mathcal{S} \subseteq \text{Distr}(\mathbb{R}_{\geq 0})$. Furthermore both settings can be viewed to use the same generative process. In the survival setting in which there is no censoring then data is drawn from (X, Y) *t.v.i.* $\mathcal{X} \times \mathcal{T}, \mathcal{T} \subseteq \mathbb{R}_{\geq 0}$ and in regression from (X, Y) *t.v.i.* $\mathcal{X} \times \mathcal{Y}, \mathcal{Y} \subseteq \mathbb{R}$, so that the only difference is whether the outcome data ranges over the Reals or positive Reals.

These closely related tasks are discussed in more detail in (Chapter 15), with a particular focus on how the more popular regression setting can be used to solve survival tasks. In (**?@sec-review**) the models are first introduced in a regression setting and then the adaptations to survival are discussed, which is natural when considering that historically machine learning survival models have been developed by adapting regression models.

[2]These definitions are given in the most general case where the time variable is over $\mathbb{R}_{\geq 0}$. In practice, all models instead assume time is over $\mathbb{R}_{>0}$ and any death at $T_i = 0$ is set to $T_i = \epsilon$ for some very small $\epsilon \in \mathbb{R}_{>0}$. Analogously for the discrete survival task. This assumption may not reflect reality as a patient could die at the study start however models cannot typically include this information in training.

Part II

Evaluation

5

What are Survival Measures?

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

In this part of the book we discuss one of the most important parts of the machine learning workflow, model evaluation (Foss and Kotthoff 2024). In the next few chapters we will discuss different metrics that can be used to measure a model's performance but before that we will just briefly discuss why model evaluation is so important.

In the simplest case, without evaluation there is no way to know if predictions from a trained machine learning model are any good. Whether one uses a simple Kaplan-Meier estimator, a complex neural network, or anything in between, there is no guarantee any of these methods will actually make useful predictions for a given dataset. This could be because the dataset is inherently difficult for any model to be trained on, perhaps because it is very 'noisy', or because a model is simply ill-suited to the task, for example using a Cox Proportional Hazards model when its key assumptions are violated. Evaluation is therefore crucial to trusting any predictions made from a model.

5.1 Survival Measures

Evaluation can be used to assess in-sample and out-of-sample performance.

In-sample evaluation measures the quality of a model's 'fit' to data, i.e., whether the model has accurately captured relationships in the training data. However, in-sample measures often cannot be applied to complex machine learning models so this part of the book omits these measures. Readers who are interested in this are are directed to Collett (2014) and Hosmer Jr, Lemeshow, and May (2011) for discussion on residuals; Choodari-Oskooei, Royston, and Parmar (2012), Kent and O'Quigley (1988) and Royston and Sauerbrei (2004) for R^2 type measures; and finally Volinsky and Raftery (2000), Hurvich and Tsai (1989), and Liang and Zou (2008) for information criterion measures.

Out-of-sample measures evaluate the quality of model predictions on new and previously unseen (by the model) data. By following established statistical methods for evaluation, and ensuring that robust resampling methods are used (James et al. 2013), evaluation provides a method for estimating the 'generalisation error', which is the expected model performance

on new datasets. This is an important concept as it provides confidence about future model performance without limiting results to the current data. Survival measures are classified into measures of:

- Discrimination (aka 'separation') – A model's discriminatory power refers to how well it separates observations that are at a higher or lower risk of event. For example, a model with good discrimination will predict that (at a given time) a dead patient has a higher probability of being dead than an alive patient.
- Calibration – Calibration is a roughly defined concept (Collins et al. 2014; F. E. Harrell, Lee, and Mark 1996; Rahman et al. 2017; Van Houwelingen 2000) that generally refers to how well a model captures average relationships between predicted and observed values.
- Predictive Performance – A model is said to have good predictive performance (or sometimes 'predictive accuracy') if its predictions for new data are 'close to' the truth.

These measures could also be categorised into how they evaluate predictions. Discrimination measures compare predictions pairwise where pairs of observations are created and then the predictions for these pairs are compared within and across each other in some way. Calibration measures evaluate predictions holistically by looking at some 'average' performance across them to provide an idea of how well suited the model is to the data. Measures of predictive performance evaluate individual predictions and usually take the sample mean of these to estimate the generalisation error.

In the next few chapters we categorise measures by the type of survival prediction they evaluate, which is a more natural taxonomy for selecting measures, but we use the above categories when introducing each measure.

5.2 How are Models Evaluated?

As well as using measures to evaluate a model's performance on a given dataset, evaluation can also be used to measure future performance, to compare and select models, and to tune internal processes. In most cases, models should not be trained/predicted/evaluated on their own, instead a number of simpler reference models should be simultaneously trained and evaluated on the same data, which is known as a 'benchmark experiment'. This is especially important for survival models, as *all* survival measures depend on the censoring distribution and therefore cannot be interpreted out of context and without comparison to other models. Benchmark experiments are used to empirically compare models across the same data and measures, meaning that if one model outperforms another then that model can be selected for future experiments (though simpler models are preferred if the performance difference is marginal). A model is usually said to 'outperform' another if it has a lower generalisation error.

The process of model evaluation is dependent on the measure itself. Measures that are 'decomposable' (predictive performance measures) calculate scores for individual predictions and take the sample mean over all scores, on the other hand 'aggregate' measures (discrimination and calibration) return a single score over all predictions. The simplest method to estimate the generalisation error is 'holdout' resampling, where a dataset $\mathcal{D}$ is split into non-overlapping subsets for training $\mathcal{D}_{train}$ and testing $\mathcal{D}_{test}$. The model is trained on $\mathcal{D}_{train}$ and predictions, $\hat{\mathbf{y}}$ are made based on the features in $\mathcal{D}_{test}$. The model is evaluated by using a measure, L, to compare the predictions to the observed data in the test set, $L(\mathbf{y}_{test}, \hat{\mathbf{y}})$.

Where possible, (repeated) k-fold cross-validation (kCV) should be used for more robust estimation of the generalisation error and for model comparison. In kCV, the data is partitioned into k folds (often k is 5 or 10), which are non-overlapping subsets. A model is trained on $k-1$ folds and evaluated on the kth fold, this process is repeated until each of the k folds has acted as the test set exactly once, the computed loss from each iteration is averaged into the final loss, which provides a good estimate of the generalisation error.

For the rest of this part of the book we will introduce different survival measures, discuss their advantages and disadvantages, and in Chapter 10 we will provide some recommendations for choosing measures. We will not discuss the general process of model resampling or evaluation further but recommend Casalicchio and Burk (2024) to readers interested in this topic.

6

Discrimination Measures

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

The next measures discused are 'discrimination measures', which evaluate how well models separate observations into different risk groups. A model is said to have good discrimination if it correctly predicts that one observation is at higher risk of the event of interest than another, where the prediction is 'correct' if the observation predicted to be at higher risk does indeed experience the event sooner.

In the survival setting, the 'risk' is taken to be the continuous ranking prediction. All discrimination measures are ranking measures, which means that the exact predicted value is irrelevant, only its relative ordering is required. For example given predictions $\{100, 2, 299.3\}$, only their rankings, $\{2, 1, 3\}$, are used by measures of discrimination.

This chapter begins with time-independent measures (Section 6.1), which measure concordance between pairs of observations at a single observed time point. The next section focuses on time-dependent measures (Section 6.2), which are primarily AUC-type measures that evaluate discrimination over all possible unique time-points and integrate the results for a single metric.

6.1 Time-Independent Measures

The simplest form of discrimination measures are concordance indices, which, in general, measure the proportion of cases in which the model correctly ranks a pair of observations according to their risk. These measures may be best understood in terms of two key definitions: 'comparable', and 'concordant'.

Definition 6.1 (Concordance)**.** Let (i, j) be a pair of observations with outcomes $\{(t_i, \delta_i), (t_j, \delta_j)\}$ and let $r_i, r_j \in \mathbb{R}$ be their respective risk predictions. Then (i, j) are called (F. E. J. Harrell et al. 1984; F. E. Harrell, Califf, and Pryor 1982):

- *Comparable* if $t_i < t_j$ and $\delta_i = 1$;
- *Concordant* if $r_i > r_j$.

Note that this book defines risk rankings such that a higher value implies higher risk of event and thus lower expected survival time (Section 4.3), hence a pair is concordant if $\mathbb{I}(t_i < t_j, r_i > r_j)$. Other sources may instead assume that higher values imply lower risk of event and hence a pair would be concordant if $\mathbb{I}(t_i < t_j, r_i < r_j)$.

Concordance measures then estimate the probability of a pair being concordant, given that they are comparable:

$$P(r_i > r_j | t_i < t_j \cap \delta_i)$$

From this formula it may be seen why these measures are referred to as time *independent*, once observations are organised into comparable pairs, the observed survival times can be ignored.

While various definitions of a 'Concordance index' (C-index) exist (discussed in the next section), they all represent a weighted proportion of the number of concordant pairs over the number of comparable pairs. As such, a C-index value will always be between $[0, 1]$ with 1 indicating perfect separation, 0.5 indicating no separation, and 0 being separation in the 'wrong direction', i.e. all high risk observations being ranked lower than all low risk observations.

Concordance measures may either be reported as a value in $[0, 1]$, a percentage, or as 'discriminatory power', which refers to the percentage improvement of a model's discrimination above the baseline value of 0.5. For example, if a model has a concordance of 0.8 then its discriminatory power is $(0.8 - 0.5)/0.5 = 60\%$. This representation of discrimination provides more information by encoding the model's improvement over some baseline although is often confused with reporting concordance as a percentage (e.g. reporting a concordance of 0.8 as 80%). Representing measures as a percentage over a baseline is a common method to improve measure interpretability and closely relates to the ERV representation of scoring rules.

> **Learn more about baseline comparison**
>
> See Section 8.4 to learn more about calculating measures with respect to an arbitrary baseline.

6.1.1 Concordance Indices

Common concordance indices in survival analysis can be expressed as a general measure:

Let $\hat{r} = (\hat{r}_1\ \hat{r}_2 \cdots \hat{r}_m)^\top$ be predicted risks, $(\mathbf{t}, \delta) = ((t_1, \delta_1)\ (t_2, \delta_2) \cdots (t_m, \delta_m))^\top$ be observed outcomes, let W be some weighting function, and let τ be a cut-off time. Then, the time-independent ('ind') *survival concordance index* is defined by,

$$C_{ind}(\hat{\mathbf{r}}, \mathbf{t}, \delta | \tau) = \frac{\sum_{i \neq j} W(t_i) \mathbb{I}(t_i < t_j, \hat{r}_i > \hat{r}_j, t_i < \tau) \delta_i}{\sum_{i \neq j} W(t_i) \mathbb{I}(t_i < t_j, t_i < \tau) \delta_i}$$

The choice of W specifies a particular variation of the c-index (see below). The use of the cut-off τ mitigates against decreased sample size (and therefore high variance) over time due to the removal of censored observations (see Figure 6.1)). For τ to be comparable across

datasets, a common choice would be to set τ as the time at which 80%, or perhaps 90% of the data have been censored or experienced the event.

There are multiple methods for dealing with tied predictions and times. Strictly, tied times are incomparable given the definition of 'comparable' given above and hence are usually ignored in the numerator. On the other hand, ties in the prediction are more problematic but a common method is to set a value of 0.5 for observations when $r_i = r_j$ (Therneau and Atkinson 2020). Specific concordance indices can be constructed by assigning a weighting scheme for W which generally depends on the Kaplan-Meier estimate of the survival function of the censoring distribution fit on training data, $\hat{G}_{KM}$, or the Kaplan-Meier estimate for the survival function of the survival distribution fit on training data, $\hat{S}_{KM}$, or both. Measures that use $\hat{G}_{KM}$ are referred to as Inverse Probability of Censoring Weighted (IPCW) measures as the estimated censoring distribution is utilised to weight the measure in order to compensate for removed censored observations. This is visualised in Figure 6.1 where $\hat{S}_{KM}$, $\hat{G}_{KM}$, and $\hat{G}^{-2}_{KM}$ are computed based on the `whas` dataset (Hosmer Jr, Lemeshow, and May 2011).

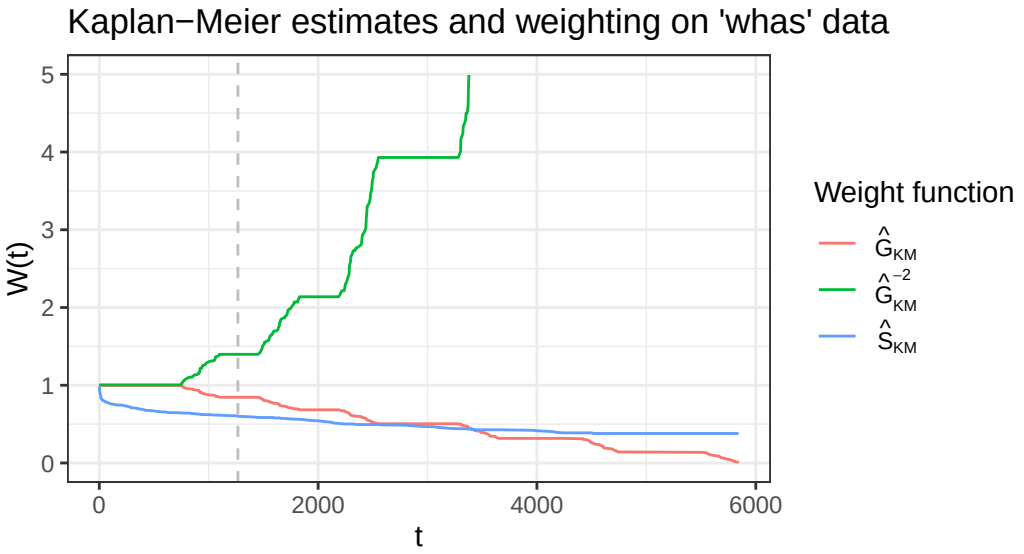


Figure 6.1: Weighting functions obtained on the `whas` dataset. x-axis is follow-up time. y-axis is outputs from one of three weighting functions: S(t), marginal survival function based on original `whas` dataset (blue), G(t), survival function based on the censoring distribution of the `whas` dataset (red), and 1/G(t)^2 (green). The vertical gray line at t=1267 represents the point at which G(t)<0.6.

The following weights have been proposed for the concordance index:

- $W(t_i) = 1$: Harrell's concordance index, C_H (F. E. J. Harrell et al. 1984; F. E. Harrell, Califf, and Pryor 1982), which is widely accepted to be the most common survival measure and imposes no weighting on the definition of concordance. The original measure given by Harrell has no cut-off, $\tau = \infty$, however applying a cut-off is now more widely accepted in practice.
- $W(t_i) = [\hat{G}_{KM}(t_i)]^{-2}$: Uno's C, C_U (Uno et al. 2011).
- $W(t_i) = [\hat{G}_{KM}(t_i)]^{-1}$
- $W(t_i) = \hat{S}_{KM}(t_i)$

- $W(t_i) = \hat{S}_{KM}(t_i)/\hat{G}_{KM}(t_i)$

All methods assume that censoring is conditionally-independent of the event given the features (Section 4.1.2), otherwise weighting by $\hat{S}_{KM}$ or $\hat{G}_{KM}$ would not be applicable. It is assumed here that $\hat{S}_{KM}$ and $\hat{G}_{KM}$ are estimated on the training data and not the testing data (though the latter may be seen in some implementations, e.g. Therneau (2015)).

6.1.2 Choosing a C-index

With multiple choices of weighting available, choosing a specific measure might seem daunting. Matters are only made worse by debate in the literature, reflecting uncertainty in measure choice and interpretation. In practice, when a suitable cut-of τ is chosen, all these weightings perform very similarly (Rahman et al. 2017; Schmid and Potapov 2012). For example, Table 6.1 uses the `whas` data again to compare Harrell's C with measures that include IPCW weighting, when no cutoff is applied (top row) and when a cutoff is applied when $\hat{G}(t) = 0.6$ (grey line in Figure 6.1). The results are almost identical when the cutoff is applied but still not massively different without the cutoff.

Table 6.1: Comparing C-index measures (calculated on the `whas` dataset using a Cox model with three-fold cross-validation) with no cut-off (top) and a cut-off when $\hat{G}(t) = 0.6$ (bottom). First column is Harrell's C, second is the weighting $1/\hat{G}(t)$, third is Uno's C.

	W=1	W=Gˆ-1	W=Gˆ-2
tau=Inf	0.74	0.73	0.71
tau=1267	0.76	0.75	0.75

On the other hand, if a poor choice is selected for τ (cutting off too late) then IPCW measures can be highly unstable (Rahman et al. 2017), for example the variance of Uno's C drastically increases with increased censoring (Schmid and Potapov 2012).

In practice, all C-index metrics provide an intuitive measure of discrimination and as such the choice of C-index is less important than the transparency in reporting. 'C-hacking' (R. Sonabend, Bender, and Vollmer 2022) is the deliberate, unethical procedure of calculating multiple C-indices and to selectively report one or more results to promote a particular model or result, whilst ignoring any negative findings. For example, calculating Harrell's C and Uno's C but only reporting the measure that shows a particular model of interest is better than another (even if the other metric shows the reverse effect). To avoid 'C-hacking':

i) the choice of C-index should be made before experiments have begun and the choice of C-index should be clearly reported;
ii) when ranking predictions are composed from distribution predictions, the composition method should be chosen and clearly described before experiments have begun.

As the C-index is highly dependent on censoring within a dataset, C-index values between experiments are not directly comparable, instead comparisons are limited to comparing model rankings, for example conclusions such as "model A outperformed model B with respect to Harrell's C in this experiment".

Learn about distribution to ranking compositions.

See Section 15.4 to learn more about creating ranking predictions from distribution predictions using composition.

6.2 Time-Dependent Measures

In the time-dependent case, where the metrics are computed based on specific survival times, the majority of measures are based on the Area Under the Curve, with one exception which is a simpler concordance index.

6.2.1 Concordance Indices

In contrast to the measures described above, Antolini's C (Antolini, Boracchi, and Biganzoli 2005) provides a time-dependent ('dep') formula for the concordance index. The definition of 'comparable' is the same for Antolini's C, however, concordance is now determined using the individual predicted survival probabilities calculated at the smaller event time in the pair:

$$P(\hat{S}_i(t_i) < \hat{S}_j(t_i) | t_i < t_j \cap \delta_i)$$

Note that observations are concordant when $\hat{S}_i(t_i) < \hat{S}_j(t_i)$ as at the time t_i, observation i has experienced the event and observation j has not, hence the expected survival probability for $\hat{S}_i(t_i)$ should be as close to 0 as possible (noting inherent randomness may prevent the perfect $\hat{S}_i(t_i) = 0$ prediction) but otherwise should be less than $\hat{S}_j(t_i)$ as j is still 'alive'. Once again this probability is estimated with a metric that could include a cut-off and different weighting schemes (though this is not included in Antolini's original definition):

$$C_{dep}(\hat{\mathbf{S}}, \mathbf{t}, \delta | \tau) = \frac{\sum_{i \neq j} W(t_i) \mathbb{I}(t_i < t_j, \hat{S}_i(t_i) < \hat{S}_j(t_i), t_i < \tau) \delta_i}{\sum_{i \neq j} W(t_i) \mathbb{I}(t_i < t_j, t_i < \tau) \delta_i}$$

where $\hat{S} = (\hat{S}_1 \ \hat{S}_2 \cdots \hat{S}_m)^\top$.

Antolini's C provides an intuitive method to evaluate the discrimination of a model based on distribution predictions without depending on compositions to ranking predictions.

6.2.2 Area Under the Curve

⚠ Warning

We are still discussing how to structure and write this section so the contents are all subject to change. The text below is 'correct' but we want to add more detail about estimation of AUC so the book can be more practical, otherwise we may remove the section completely, let us know your thoughts about what you'd like to see here!

AUC, or AUROC, measures calculate the Area Under the Receiver Operating Characteristic (ROC) Curve, which is a plot of the *sensitivity* (or true positive rate (TPR)) against $1 - \textit{specificity}$ (or true negative rate (TNR)) at varying thresholds (described below) for the predicted probability (or risk) of event. Figure 6.2 visualises ROC curves for two classification models. The blue line is a featureless baseline that has no discrimination. The red line is a decision tree with better discrimination as it comes closer to the top-left corner.

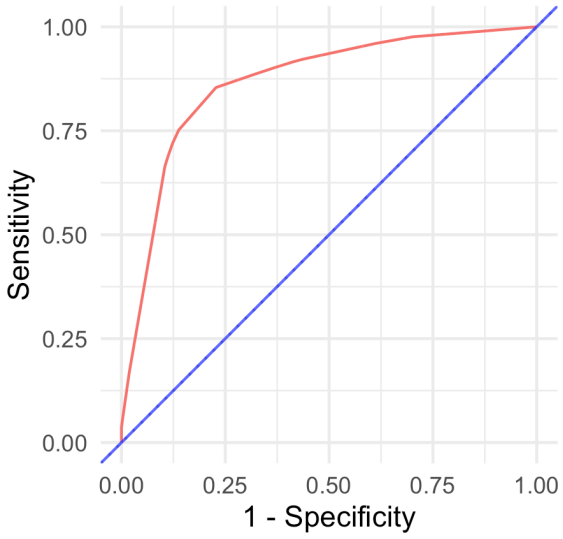


Figure 6.2: ROC Curves for a classification example. Red is a decision tree with good discrimination as it 'hugs' the top-left corner. Blue is a featureless baseline with no discrimination as it sits on $y = x$.

In a classification setting with no censoring, the AUC has the same interpretation as Harrell's C (Uno et al. 2011). AUC measures for survival analysis were developed to provide a time-dependent measure of discriminatory ability (Patrick J. Heagerty, Lumley, and Pepe 2000). In a survival setting it can reasonably be expected for a model to perform differently over time and therefore time-dependent measures are advantageous. Computation of AUC estimators is complex and as such there are limited accessible metrics available off-shelf. There is limited evidence of these estimators used in the literature, hence discussion of these measures is kept brief.

The AUC, TPR, and TNR are derived from the *confusion matrix* in a binary classification setting. Let $b_i, \hat{b}_i \in \{0, 1\}$ be the true and predicted binary outcomes respectively for some observation i. The confusion matrix is then given by:

	$b_i = 1$	$b_i = 0$
$\hat{b}_i = 1$	TP	FP
$\hat{b}_i = 0$	FN	TN

where $TN := \sum_i \mathbb{I}(b_i = 0, \hat{b}_i = 0)$ is the number of true negatives, $TP := \sum_i \mathbb{I}(b_i = 1, \hat{b}_i = 1)$ is the number true positives, $FP := \sum_i \mathbb{I}(b_i = 0, \hat{b}_i = 1)$ is the number of false positives, and $FN := \sum_i \mathbb{I}(b_i = 1, \hat{b}_i = 0)$ is the number of false negatives. From these are derived

$$TPR := \frac{TP}{TP + FN} \tag{6.1}$$

$$TNR := \frac{TN}{TN + FP} \tag{6.2}$$

In classification, a probabilistic prediction of an event can be *thresholded* to obtain a deterministic prediction. For a predicted $\hat{p} := \hat{P}(b = 1)$, and threshold α, the thresholded binary prediction is $\hat{b} := \mathbb{I}(\hat{p} > \alpha)$. This is achieved in survival analysis by thresholding the linear predictor at a given time for different values of the threshold and different values of the time. All measures of TPR, TNR and AUC are in the range $[0, 1]$ with larger values preferred.

Weighting the linear predictor was proposed by Uno *et al.* (2007) (Uno et al. 2007) and provides a method for estimating TPR and TNR via

$$TPR_U(\hat{\eta}, \mathbf{t}, \delta|\tau, \alpha) = \frac{\sum_{i=1}^{m} \delta_i \mathbb{I}(k(\hat{\eta}_i) > \alpha, t_i \leq \tau)[\hat{G}_{KM}(t_i)]^{-1}}{\sum_{i=1}^{m} \delta_i \mathbb{I}(t_i \leq \tau)[\hat{G}_{KM}(t_i)]^{-1}}$$

and

$$TNR_U(\hat{\eta}, \mathbf{t}|\tau, \alpha) \mapsto \frac{\sum_{i=1}^{m} \mathbb{I}(k(\hat{\eta}_i) \leq \alpha, t_i > \tau)}{\sum_{i=1}^{m} \mathbb{I}(t_i > \tau)}$$

where $\hat{\eta} = (\hat{\eta}_1 \ \hat{\eta}_2 \cdots \hat{\eta}_m)^\top$ is a vector of predicted linear predictors, τ is the time at which to evaluate the measure, α is a cut-off for the linear predictor, and k is a known, strictly increasing, differentiable function. k is chosen depending on the model choice, for example if the fitted model is PH then $k(x) = 1 - \exp(-\exp(x))$ (Uno et al. 2007). Similarities can be drawn between these equations and Uno's concordance index, in particular the use of IPCW. Censoring is again assumed to be at least random once conditioned on features. Plotting TPR_U against $1 - TNR_U$ for varying values of α provides the ROC.

The second method, which appears to be more prominent in the literature, is derived from Heagerty and Zheng (2005) (Patrick J. Heagerty and Zheng 2005). They define four distinct classes, in which observations are split into controls and cases.

An observation is a *case* at a given time-point if they are dead, otherwise they are a *control*. These definitions imply that all observations begin as controls and (hypothetically) become cases over time. Cases are then split into *incident* or *cumulative* and controls are split into *static* or *dynamic*. The choice between modelling static or dynamic controls is dependent on the question of interest. Modelling static controls implies that a 'subject does not change disease status' (Patrick J. Heagerty and Zheng 2005), and few methods have been developed for this setting (Kamarudin, Cox, and Kolamunnage-Dona 2017), as such the focus here is on *dynamic* controls. The incident/cumulative cases choice is discussed in more detail below.[1]

The TNR for dynamic cases is defined as

[1] All measures discussed in this section evaluate model discrimination from 'markers', which may be a *predictive* marker (model predictions) or a *prognostic* marker (a single covariate). This section always defines a marker as a ranking prediction, which is valid for all measures discussed here with the exception of one given at the end.

$$TNR_D(\hat{\mathbf{r}}, N|\alpha, \tau) = P(\hat{r}_i \leq \alpha | N_i(\tau) = 0)$$

where $\hat{r} = (\hat{r}_1\ \hat{r}_2 \cdots \hat{r}_n)^\top$ is some deterministic prediction and $N(\tau)$ is a count of the number of events in $[0, \tau)$. Heagerty and Zheng further specify y to be the predicted linear predictor $\hat{\eta}$. Cumulative/dynamic and incident/dynamic measures are available in software packages 'off-shelf', these are respectively defined by

$$TPR_C(\hat{\mathbf{r}}, N|\alpha, \tau) = P(\hat{r}_i > \alpha | N_i(\tau) = 1)$$

and

$$TPR_I(\hat{\mathbf{r}}, N|\alpha, \tau) = P(\hat{r}_i > \alpha | dN_i(\tau) = 1)$$

where $dN_i(\tau) = N_i(\tau) - N_i(\tau-)$. Practical estimation of these quantities is not discussed here.

The choice between the incident/dynamic (I/D) and cumulative/dynamic (C/D) measures primarily relates to the use-case. The C/D measures are preferred if a specific time-point is of interest (Patrick J. Heagerty and Zheng 2005) and is implemented in several applications for this purpose (Kamarudin, Cox, and Kolamunnage-Dona 2017). The I/D measures are preferred when the true survival time is known and discrimination is desired at the given event time (Patrick J. Heagerty and Zheng 2005).

Defining a time-specific AUC is now possible with

$$AUC(\hat{\mathbf{r}}, N|\tau) = \int_0^1 TPR(\hat{\mathbf{r}}, N|1 - TNR^{-1}(p|\tau), \tau)\ dp$$

Finally, integrating over all time-points produces a time-dependent AUC and as usual a cut-off is applied for the upper limit,

$$AUC^*(\hat{\mathbf{r}}, N|\tau^*) = \int_0^{\tau^*} AUC(\hat{\mathbf{r}}, N|\tau) \frac{2\hat{p}_{KM}(\tau)\hat{S}_{KM}(\tau)}{1 - \hat{S}^2_{KM}(\tau^*)}\ d\tau$$

where $\hat{S}_{KM}, \hat{p}_{KM}$ are survival and mass functions estimated with a Kaplan-Meier model on training data.

Since Heagerty and Zheng's paper, other methods for calculating the time-dependent AUC have been devised, including by Chambless and Diao (Chambless and Diao 2006), Song and Zhou (Song and Zhou 2008), and Hung and Chiang (Hung and Chiang 2010). These either stem from the Heagerty and Zheng paper or ignore the case/control distinction and derive the AUC via different estimation methods of TPR and TNR. Blanche *et al.* (2012) (Blanche, Latouche, and Viallon 2012) surveyed these and concluded ''regarding the choice of the retained definition for cases and controls, no clear guidance has really emerged in the literature'', but agree with Heagerty and Zeng on the use of C/D for clinical trials and I/D for 'pure' evaluation of the marker. Blanche *et al.* (2013) (Blanche, Dartigues, and Jacqmin-Gadda 2013) published a survey of C/D AUC measures with an emphasis on non-parametric estimators with marker-dependent censoring, including their own Conditional IPCW (CIPCW) AUC, which is not discussed further here as it cannot be used for evaluating predictions (R. E. B. Sonabend 2021).

Reviews of AUC measures have produced (sometimes markedly) different results (Blanche, Latouche, and Viallon 2012; Li, Greene, and Hu 2018; Kamarudin, Cox, and Kolamunnage-Dona 2017) with no clear consensus on how and when these measures should be used. The primary advantage of these measures is to extend discrimination metrics to be time-dependent. However, it is unclear how to interpret a threshold of a linear predictor and moreover if this is even the 'correct' quantity to threshold, especially when survival distribution predictions are the more natural object to evaluate over time.

7

Calibration Measures

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

Calibration measures evaluate the 'average' quality of survival distribution predictions. This chapter is kept relatively short as the literature in this area is scarce (Rahman et al. 2017), this is likely due to the meaning of calibration being unclear in a survival context (Van Houwelingen 2000). However the meaning of calibration is better specified once specific metrics are introduced. As with other measure classes, only measures that can generalise beyond Cox PH models are included here but note that several calibration measures for re-calibrating PH models have been discussed in the literature (Demler, Paynter, and Cook 2015; Van Houwelingen 2000).

Calibration measures can be grouped (Andres et al. 2018) into those that evaluate distributions at a single time-point, '1-Calibration' or 'Point Calibration' measures, and those that evaluate distributions at all time-points 'distributional-calibration' or 'probabilistic calibration' measures. A point-calibration measure will evaluate a function of the predicted distribution at a single time-point whereas a probabilistic measure evaluates the distribution over a range of time-points; in both cases the evaluated quantity is compared to the observed outcome, (t, δ).

7.1 Point Calibration

Point calibration measures can be further divided into metrics that evaluate calibration at a single time-point (by reduction) and measures that evaluate an entire distribution by only considering the event time. The difference may sound subtle but it affects conclusions that can be drawn. In the first case, a calibration measure can only draw conclusions at that one time-point, whereas the second case can draw conclusions about the calibration of the entire distribution. This is the same caveat as using prediction error curves for scoring rules.

> Learn more about prediction error curves
>
> See Section 8.3 to learn more about prediction error curves.

7.1.1 Calibration by Reduction

Point calibration measures are implicitly reduction methods as they use classification methods to evaluate a full distribution based on a single point only. For example, given a predicted survival function $\hat{S}$, one could calculate the survival function at a single time point, $\hat{S}\tau$ and then use probabilistic classification calibration measures. Using this approach one may employ common calibration methods such as the Hosmer–Lemeshow test (Hosmer and Lemeshow 1980). Measuring calibration in this way can have significant drawbacks as a model may be well-calibrated at one time-point but poorly calibrated at all others (Haider et al. 2020). To mitigate this, one could perform the Hosmer–Lemeshow test (or other applicable tests) multiple times with multiple testing correction at many (or all possible) time points, however this would be less efficient and more difficult to interpret than other measures discussed in this chapter.

> Learn more about reduction
>
> See Chapter 15 to learn more about reduction.

7.1.2 Houwelingen's α

As opposed to evaluating distributions at one or more arbitrary time points, one could instead evaluate distribution predictions at meaningful times. van Houwelingen proposed several measures (Van Houwelingen 2000) for calibration but only one generalises to all probabilistic survival models, termed here 'Houwelingen's α'. The measure assesses if the model correctly estimates the theoretical 'true' cumulative hazard function of the underlying data generating process, $H = \hat{H}$.

The statistic is derived by noting the closely related nature of survival analysis and counting processes, and exploiting the fact that the sum of the cumulative hazard function is an estimate for the number of events in a given time-period (Hosmer Jr, Lemeshow, and May 2011). As this result is often surprising result to readers, below is a short experiment using R that demonstrates how the sum of the cumulative hazard estimated by a Kaplan-Meier estimator is identical to the number of randomly simulated deaths in a dataset:

```
set.seed(42)
library(survival)

event = rbinom(100, 1, 0.7)
times = runif(100)
H = survfit(Surv(times, event) ~ 1)$cumhaz
sum(event) / sum(H)
#> [1] 1
```

Houwelingen's α is then defined by substituting H for the observed total number of deaths and summing over all predictions:

$$H_\alpha(\delta, \hat{\mathbf{H}}, \mathbf{t}) = \frac{\sum_i \delta_i}{\sum_i \hat{\mathbf{H}}_i(t_i)}$$

with standard error $SE(H_\alpha) = \exp(1/\sqrt{\sum_i \delta_i})$. A model is well-calibrated with respect to

H_α if $H_\alpha = 1$.

The next metrics we look at evaluate models across a spectrum of points to assess calibration over time.

7.2 Probabilistic Calibration

Calibration over a range of time points may be assessed quantitatively or qualitatively, with graphical methods often favoured. Graphical methods compare the average predicted distribution to the expected distribution, which can be estimated with the Kaplan-Meier curve, discussed next.

7.2.1 Kaplan-Meier Comparison

The simplest graphical comparison compares the average predicted survival curve to the Kaplan-Meier curve estimated on the testing data. Let $\hat{S}_1, ..., \hat{S}_m$ be predicted survival functions, then the average predicted survival function is the mixture: $\bar{\hat{S}} = \frac{1}{m}\sum_{i=1}^{m} \hat{S}_i(\tau)$. This estimate can be plotted next to the Kaplan-Meier estimate of the survival distribution in a test dataset (i.e., the true data for model evaluation), allowing for visual comparison of how closely these curves align. An example is given in Figure 7.1, a Cox model (CPH), random survival forest, and relative risk tree with distribution composition, are all compared to the Kaplan-Meier estimator. This figure highlights the advantages and disadvantages of this method. The relative risk tree is clearly poorly calibrated as it increasingly diverges from the Kaplan-Meier. In contrast, the Cox model and random forest cannot be directly compared to one another, as both models frequently overlap with each other and the Kaplan-Meier estimator. Hence it is possible to say that the Cox and forests models are better calibrated than the risk tree, however it is not possible to say which of those two is better calibrated and whether their distance from the Kaplan-Meier is significant or not at a given time (when not clearly overlapping).

This method is useful for making broad statements such as "model X is clearly better calibrated than model Y" or "model X appears to make average predictions close to the Kaplan-Meier estimate", but that is the limit in terms of useful conclusions. One could refine this method for more fine-grained information by instead using relative risk predictions to create 'risk groups' that can be plotted against a stratified Kaplan-Meier, however this method is harder to interpret and adds even more subjectivity around how many risk groups to create and how to create them (Royston and Altman 2013). The next measure we consider includes a graphical method as well as a quantitative interpretation.

7.2.2 D-Calibration

D-Calibration (Andres et al. 2018; Haider et al. 2020) evaluates a model's calibration by assessing if the predicted survival distributions follow the Uniform distribution as expected, which is motivated by the result that for any random variable X it follows $S_X(x) \sim \mathcal{U}(0, 1)$. This can be tested using a χ^2 test-statistic:

$$\chi^2 := \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

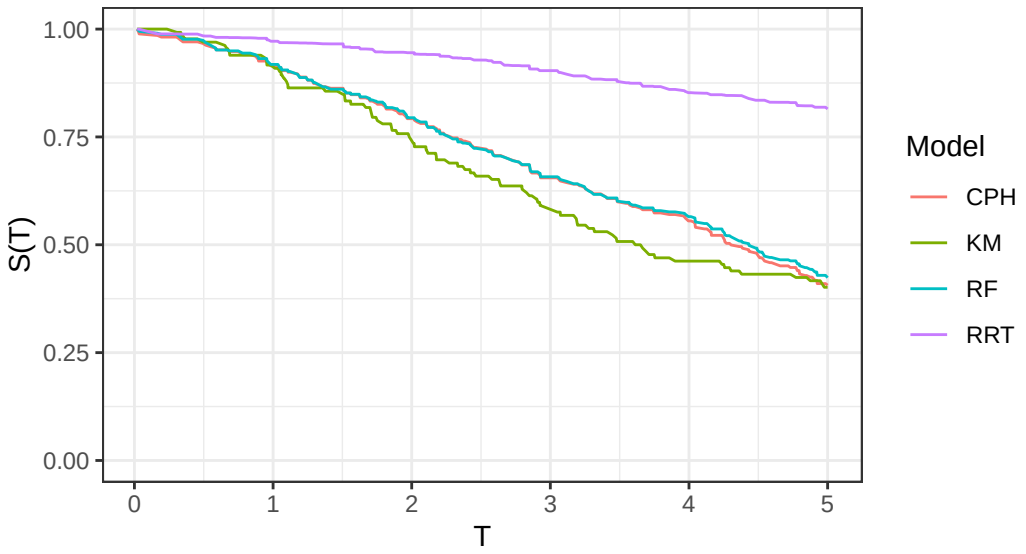


Figure 7.1: Comparing the calibration of a Cox PH (CPH), random forest (RF), and relative risk tree (RRT) to the Kaplan-Meier estimate of the survival function calculated on a test set. The calibration of RRT notably decreases over time whereas RF and CPH are closer to the Kaplan-Meier curve.

where $O_1, ..., O_n$ is the observed number of events in n groups and $E_1, ..., E_n$ is the expected number of events.

To utilise this test, the $[0, 1]$ codomain of S_i is cut into B disjoint contiguous intervals ('bins') over the full range $[0, 1]$. Let m be the total number of observations, then assuming a discrete uniform distribution as the theoretical distribution, the expected number of events in each bin is $E_i = m/B$ (as the probability of an observation falling into each bin is equal).

The observations in the ith bin, b_i, are defined by the set:

$$b_i := \{j = 1, \dots, m : \lceil \hat{S}_i(t_j)B \rceil = i\}$$

where $j = 1, \dots, m$ are the indices of the observations, $\hat{S}_i$ are observed (i.e., predicted) survival functions, t_i are observed (i.e., the ground truth) outcome times, and $\lceil \cdot \rceil$ is the ceiling function. The observed number of events in b_i is then the number of observations in that set: $O_i = |b_i|$.

The D-Calibration measure, or χ^2 statistic, is now defined by,

$$D_{\chi^2}(\hat{\mathbf{S}}, \mathbf{t}) := \frac{\sum_{i=1}^{B} (O_i - \frac{m}{B})^2}{m/B}$$

where $\hat{\mathbf{S}} = (\hat{S}_1 \ \hat{S}_2 \cdots \hat{S}_m)^\top$ and $\mathbf{t} = (t_1 \ t_2 \cdots t_m)^\top$.

This measure has several useful properties. Firstly, one can test the null hypothesis that a model is 'D-calibrated' by deriving a p-value from comparison to χ^2_{B-1}. Secondly, D_{χ^2} tends to zero as a model is increasingly well-calibrated, hence the measure can be used for model

comparison. Finally, the theory lends itself to an intuitive graphical calibration method as a D-calibrated model implies:

$$p = \frac{\sum_i \mathbb{I}(T_i \leq \hat{F}_i^{-1}(p))}{m}$$

where p is some value in $[0, 1]$, $\hat{F}_i^{-1}$ is the ith predicted inverse cumulative distribution function, and m is again the number of observations. In words, the number of events occurring at or before each quantile should be equal to the quantile itself, for example 50% of events should occur before their predicted median survival time. Therefore, one can plot p on the x-axis and the right hand side of the above equation on the y-axis. A D-calibrated model should result in a straight line on $x = y$. This is visualised in Figure 7.2 for the same models as in Figure 7.1. This figure supports the previous findings that the relative risk tree is poorly calibrated in contrast to the Cox model and random forest but again no direct comparison between the latter models is possible.

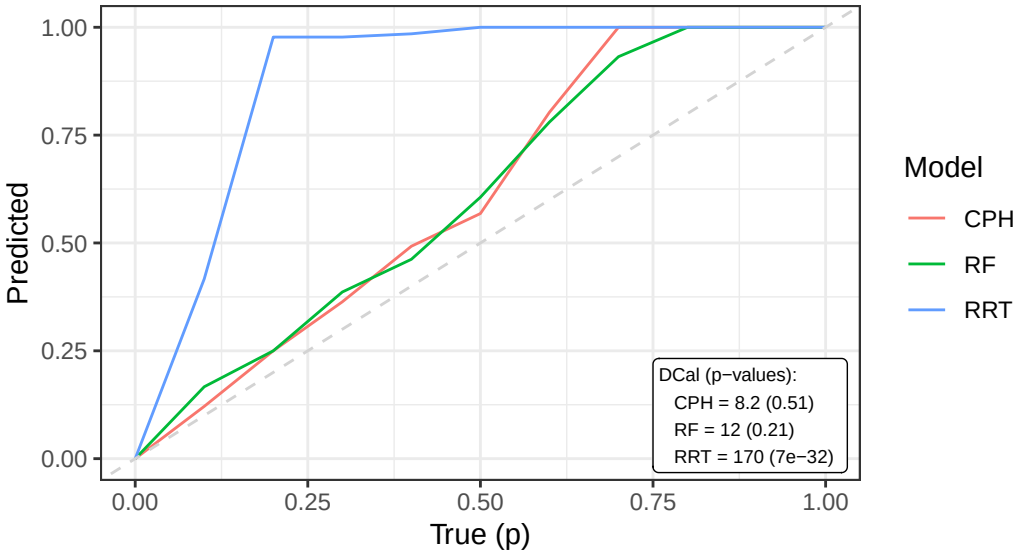


Figure 7.2: Comparing the D-calibration of a Cox PH (CPH), random forest (RF), and relative risk tree (RRT) to the expected distribution on y=x. As with Figure 7.1, the relative risk tree is clearly not D-calibrated (as supported by the figures in the bottom-right). The CPH and RF are closer to the y=x however neither follow it perfectly.

Whilst D-calibration has the same problems as the Kaplan-Meier method with respect to visual comparison, at least in this case there are quantities to help draw more concrete solutions. For the models in Figure 7.2, it is clear that the relative risk tree is not D-calibrated with $p < 0.01$ indicating the null hypothesis of D-calibration, i.e., the predicted quantiles not following a Discrete Uniform distribution, can be comfortably rejected. Whilst the D-calibration for the Cox model is smaller than that of the random forest, the difference is unlikely to be significant, as is seen in the overlapping curves in the figure.

The next chapter will look at scoring rules, which provides a more concrete method to analytically compare the predicted distributions from survival models.

8

Evaluating Distributions by Scoring Rules

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

Scoring rules evaluate probabilistic predictions and (attempt to) measure the overall predictive ability of a model in terms of both calibration and discrimination (Gneiting and Raftery 2007; Murphy 1973). In contrast to calibration measures, which assess the average performance across all observations on a population level, scoring rules evaluate the sample mean of individual predictions across all observations in a test set. As well as being able to provide information at an individual level, scoring rules are also popular as probabilistic forecasts are widely recognised to be superior to deterministic predictions for capturing uncertainty in predictions (A. P. Dawid 1984; A. Philip Dawid 1986). Formalisation and development of scoring rules has primarily been due to Dawid (A. P. Dawid 1984; A. Philip Dawid 1986; A. Philip Dawid and Musio 2014) and Gneiting and Raftery (Gneiting and Raftery 2007); though the earliest measures promoting "rational" and "honest" decision making date back to the 1950s (Brier 1950; Good 1952). Few scoring rules have been proposed in survival analysis, although the past few years have seen an increase in popularity in these measures. Before delving into these measures, we will first describe scoring rules in the simpler classification setting.

8.1 Classification Losses

In the simplest terms, a scoring rule compares two values and assigns them a score (hence 'scoring rule'), formally we'd write $L : \mathbb{R} \times \mathbb{R} \mapsto \bar{\mathbb{R}}$. In machine learning, this usually means comparing a prediction for an observation to the ground truth, so $L : \mathbb{R} \times \mathcal{P} \mapsto \bar{\mathbb{R}}$ where $\mathcal{P}$ is a set of distributions. Crucially, scoring rules usually refer to comparisons of true and predicted *distributions*. For example, let's construct a scoring rule as follows:

1. Let $y \in \{0, 1\}$ be the ground truth and let $\hat{p}$ be the predicted probability mass function such that $\hat{p}(y)$ is the probability of the observed event occurring;

2. Define $\hat{y} \coloneqq \mathbb{I}(\hat{p}(y) \geq 0.5)$, i.e., $\hat{y}$ is 1 if the predicted probability of event 1 is greater or equal than 0.5;
3. Then define our scoring rule such that we score 1 if $\hat{y}$ equals y or 0 otherwise: $SR \coloneqq \mathbb{I}(\hat{y} = y)$.

In practice, minimisation is often the goal in automated machine learning processes, so we usually talk about 'losses' (which are minimised) instead of scoring rules that are maximised, hence let's adapt SR slightly to the loss $L \coloneqq \mathbb{I}(\hat{y} \neq y))$, and putting all the above together we get

$$L_P(\hat{p}, y) = \mathbb{I}(y \neq \mathbb{I}(\hat{p}(y) \geq 0.5))$$

This loss is interpretable and has a real world meaning, in fact it's just the mean misclassification error after discretising a probabilistic classification prediction. Now consider the following loss:

$$L_I(\hat{p}, y) = 1 - L_P$$

This follows the definition of a scoring rule/loss as it maps a distribution and value to a real-valued number, but the loss is also terrible as it assigns lower scores to worse predictions!

The difference between these losses is that the first is 'proper' whereas the latter is 'improper'. A 'proper' loss is a loss that is minimised by the 'correct' prediction.

Another important property is *strict* properness. A loss is strictly proper if the loss is *uniquely* minimised by the 'correct' prediction. Let's modify L_P slightly to become the squared difference between the true value and predicted probability (in fact this is the widely used Brier score (Brier 1950)):

$$L_S(\hat{p}, y) = (y - \hat{p}(y))^2$$

Now if we compare L_P and L_S across different values of y and $\hat{p}_y$ (Table 8.1), we can easily see that whilst L_P provides some utility, this is limited as we'd have no way to know that some predictions are closer to the truth than others. On the other hand, L_S provides a quantitative method to compare predictions against the truth and between each other.

Table 8.1: Comparing proper (L_P) and strictly proper (L_S) scoring rules across different qualities of predictions.

	$y = 0$	$y = 1$
$\hat{p}_y = 0$	$L_P = 0; L_S = 0$	$L_P = 0; L_S = 1$
$\hat{p}_y = 0.3$	$L_P = 0; L_S = 0.09$	$L_P = 0; L_S = 0.49$
$\hat{p}_y = 0.6$	$L_P = 1; L_S = 0.36$	$L_P = 1; L_S = 0.16$
$\hat{p}_y = 1$	$L_P = 1; L_S = 1$	$L_P = 1; L_S = 0$

Mathematically, a classification loss $L : \mathcal{P} \times \mathcal{Y} \to \bar{\mathbb{R}}$ is *proper* if for any distributions p_Y, p in $\mathcal{P}$ and for any random variables $Y \sim p_Y$, it holds that $\mathbb{E}[L(p_Y, Y)] \leq \mathbb{E}[L(p, Y)]$. The loss is *strictly proper* if, in addition, $p = p_Y$ uniquely minimizes the loss.

Proper losses provide a method of model comparison as, by definition, predictions closest to the true distribution will result in lower expected losses. Strictly proper losses have additional important uses such as in model optimisation, as minimisation of the loss will result in the 'optimum score estimator based on the scoring rule' (Gneiting and Raftery 2007). Whilst properness is usually a minimal acceptable property for a loss, it is generally not sufficient on its own, for example consider the measure $L(\hat{p}_y, y) = 0$, which is proper as it is minimised by $L(y, y)$ but it is clearly useless.

Two widely used losses for classification are the Brier score (Brier 1950) and log loss (Good 1952), defined respectively by

$$L_{brier}(\hat{p}, y) \mapsto (y - \hat{p}(y))^2$$

and

$$L_{logloss}(\hat{p}, y) = -\log \hat{p}(y)$$

These losses are visualised in Figure 8.1, which highlights that both losses are strictly proper (A. Philip Dawid and Musio 2014) as they are minimised when the true prediction is made, and we can say that we converge to the minimum as predictions are increasingly improved.

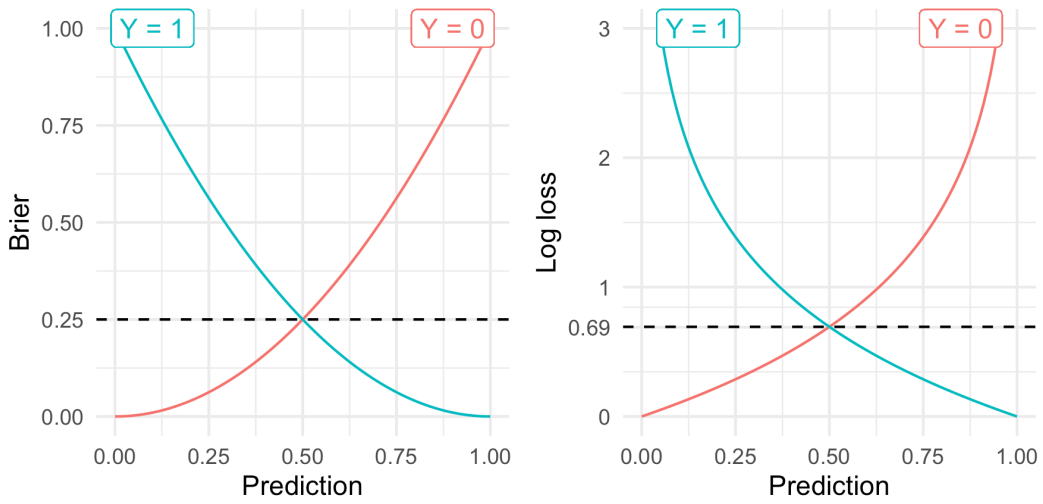


Figure 8.1: Brier and log loss scoring rules for a binary outcome and varying probabilistic predictions. x-axis is a probabilistic prediction in $[0, 1]$, y-axis is Brier score (left) and log loss (right). Blue lines are varying Brier score/log loss over different predicted probabilities when the true outcome is 1. Red lines are varying Brier score/log loss over different predicted probabilities when the true outcome is 0. Both losses are minimised with the correct prediction, i.e. if $\zeta.p(1) = 1$ when $y = 1$ and $\zeta.p(1) = 0$ when $y = 0$ for a predicted discrete distribution ζ.

8.2 Survival Losses

We are now ready to list common scoring rules in survival analysis and discuss some of their properties. As with other chapters, this list is likely not exhaustive but will cover commonly used losses.

8.2.1 Integrated Graf Score

The Integrated Graf Score (IGS) was introduced by Graf (Graf and Schumacher 1995; Graf et al. 1999) as an analogue to the integrated brier score (IBS) in regression. The loss is defined by

$$L_{IGS}(\hat{S}, t, \delta | \hat{G}_{KM}) = \int_0^{\tau^*} \frac{\hat{S}^2(\tau)\mathbb{I}(t \leq \tau, \delta = 1)}{\hat{G}_{KM}(t)} + \frac{\hat{F}^2(\tau)\mathbb{I}(t > \tau)}{\hat{G}_{KM}(\tau)} \, d\tau \tag{8.1}$$

where $\hat{S}^2(\tau) = (\hat{S}(\tau))^2$ and $\hat{F}^2(\tau) = (1 - \hat{S}(\tau))^2$, and $\tau^* \in \mathbb{R}_{\geq 0}$ is an upper threshold to compute the loss up to, and $\hat{G}_{KM}$ is the Kaplan-Meier trained on the censoring distribution for IPCW.

> Learn more about IPCW
>
> See Section 6.1 to learn more about IPCW.

To understand this loss, let's break it down and look at the computations at a single time-point, τ. At τ the loss will either be:

1. $\frac{\hat{S}^2(\tau)}{\hat{G}_{KM}(t)}$ – If the observation experiences the event before τ
2. 0 – If the observation is censored before τ
3. $\frac{\hat{F}^2(\tau)}{\hat{G}_{KM}(\tau)}$ – If the observation's outcome (censoring or event) is after τ

As we have no information about the true survival status of censored observations beyond their censoring times, it is sensible to not attempt to provide a meaningful score once censored, so their contribution is 0. For observations that are known to have experienced the event at τ, we would expect their survival probability to be zero as the event has occurred (and they cannot continue to survive) hence contributing $\hat{S}^2$ – the addition of $\hat{G}_{KM}(t)$ has the effect of placing more weight on the score at the observed event time if the proportion of censoring is lower at this time, the reason being that when the observations are alive ($t > \tau$) then their contributing the rest of the weighting after this time. Finally, for observations who are still alive, then we'd expect their survival probability to be as close to 1 as possible with inverse weighting at the current timepoint. As $\tau \to \infty$, then $\hat{G}_{KM}(\tau) \to 0$ as the number of observations in the dataset decreases, hence this weighting ensures that observations that are still in the data can contribute as if all observations were still in the data.

When censoring is uninformative, the IGS consistently estimates the mean square error $L(t, S|\tau^*) = \int_0^{\tau^*} [\mathbb{I}(t > \tau) - S(\tau)]^2 d\tau$, where S is the correctly specified survival function (Gerds and Schumacher 2006). However, despite these promising properties, the IGS is improper and must therefore be used with care (Rindt et al. 2022; R. Sonabend 2022).

The reweighted IGS is a strictly proper outcome-independent loss (R. Sonabend 2022) that reweights the IGS by removing censored observations and reweighting the denominator.

$$L_{RIGS}(\hat{S}, t, \delta|\hat{G}_{KM}) = \frac{\delta \int_{\mathcal{T}} (\mathbb{I}(t \leq \tau) - \hat{F}(\tau))^2 \, d\tau}{\hat{G}_{KM}(t)}$$

This loss removes all censored observations, which can be problematic if the proportion of censoring is high. For uncensored observations we expect the predicted survival probability to be 1 before any outcome is observed and 0 otherwise, which follows more closely to the integrated Brier score. By changing the weighting the interpretation of contributions at time-points changes slightly, in the original IGS we may think of this as "inverse weighting for as long as the observation remains in the data", which means the weight of a contribution at a time-point will be different for all observations and all time-points. On the other hand, for RIGS, we weight by the outcome time for each observation, which remains the same over time. Hence we instead inflate scores for observations whose outcome are later in the dataset, this is intuitive as it essentially places more importance on observations that are representative of being alive at those time points.

As the loss is strictly proper it may be 'safer' to use than the IGS in automated experiments, however this does come at the expense of removing censored observations.

8.2.2 Integrated Survival Log Loss

The integrated survival log loss (ISLL) was also proposed by Graf et al. (1999).

$$L_{ISLL}(\hat{S}, t, \delta|\hat{G}_{KM}) = -\int_0^{\tau^*} \frac{\log[\hat{F}(\tau)]\mathbb{I}(t \leq \tau, \delta = 1)}{\hat{G}_{KM}(t)} + \frac{\log[\hat{S}(\tau)]\mathbb{I}(t > \tau)}{\hat{G}_{KM}(\tau)} \, d\tau$$

where $\tau^* \in \mathcal{T}$ is an upper threshold to compute the loss up to.

Similarly to the IGS, there are three ways to contribute to the loss depending on whether an observation is censored, experienced the event, or alive, at τ. Whilst the IGS is routinely used in practice, there is no evidence that ISLL is used, and moreover there are no proofs (or claims) that it is proper.

The reweighted ISLL (RISLL) follows similarly to the RIGS and is also outcome-independent strictly proper (R. Sonabend 2022).

$$L_{RISLL}(\hat{S}, t, \delta|\hat{G}_{KM}) \mapsto -\frac{\delta \int_{\mathcal{T}} \mathbb{I}(t \leq \tau) \log[\hat{F}(\tau)] + \mathbb{I}(t > \tau) \log[\hat{S}(\tau)] \, d\tau}{\hat{G}_{KM}(t)}$$

8.2.3 Survival density log loss

Another outcome-independent strictly proper scoring rule is the survival density log loss (SDLL) (R. Sonabend 2022), which is given by

$$L_{SDLL}(\hat{f}, t, \delta|\hat{G}_{KM}) = -\frac{\delta \log[\hat{f}(t)]}{\hat{G}_{KM}(t)}$$

where $\hat{f}$ is the predicted probability density function. This loss is essentially the classification log loss ($-\log(\hat{p}(t))$) with added IPCW. Whilst the classification log loss has beneficial properties such as being differentiable, this is more complex for the SDLL, which is also only an approximate loss. A useful alternative to the SDLL which can be readily used in automated procedures is the right-censored log loss.

8.2.4 Right-censored log loss

The right-censored log loss (RCLL) is an outcome-independent strictly proper scoring rule (Avati et al. 2020) that does not make use of IPCW and is thus not considered to be an approximate loss. The RCLL is defined by

$$L_{RCLL}(\hat{S}, t, \delta) = -\log[\delta \hat{f}(t) + (1-\delta)\hat{S}(t)]$$

This loss is easily interpretable when we break it down into its two halves:

1. If an observation is censored at t then all the information we have is that they did not experience the event at the time, so they must be 'alive', hence the optimal value is $\hat{S}(t) = 1$ (which becomes $-log(1) = 0$).
2. If an observation experiences the event then the 'best' prediction is for the probability of the event at that time to be maximised, as pdfs are not upper-bounded this means $\hat{f}(t) = \infty$ (and $-log(t) \to \infty$ as $t \to \infty$).

8.2.5 Absolute Survival Loss

The absolute survival loss, developed over time by Schemper and Henderson (2000) and Schmid et al. (2011), is based on the mean absolute error is very similar to the IGS but removes the squared time:

$$L_{ASL}(\hat{S}, t, \delta | \hat{G}_{KM}) = \int_0^{\tau^*} \frac{\zeta.S(\tau)\mathbb{I}(t \leq \tau, \delta = 1)}{\hat{G}_{KM}(t)} + \frac{\zeta.F(\tau)\mathbb{I}(t > \tau)}{\hat{G}_{KM}(\tau)}\, d\tau$$

where $\hat{G}_{KM}$ and τ^* are as defined above. Analogously to the IGS, the ASL score consistently estimates the mean absolute error when censoring is uninformative (Schmid et al. 2011) but there are also no proofs or claims of properness. The ASL and IGS tend to yield similar results (Schmid et al. 2011) but in practice there is no evidence of the ASL being widely used.

8.3 Prediction Error Curves

As well as evaluating probabilistic outcomes with integrated scoring rules, non-integrated scoring rules can be utilised for evaluating distributions at a single point. For example, instead of evaluating a probabilistic prediction with the IGS over $\mathbb{R}_{\geq 0}$, instead one could compute the IGS at a single time-point, $\tau \in \mathbb{R}_{\geq 0}$, only. Plotting these for varying values of τ results in 'prediction error curves' (PECs), which provide a simple visualisation for how predictions vary over the outcome. PECs are especially useful for survival predictions as they can visualise the prediction 'over time'. PECs should only be used as a graphical guide

and never for model comparison as they only provide information at a limited number of points. An example is provided in Figure 8.2 for the IGS where the the Cox PH consistently outperforms the SVM.

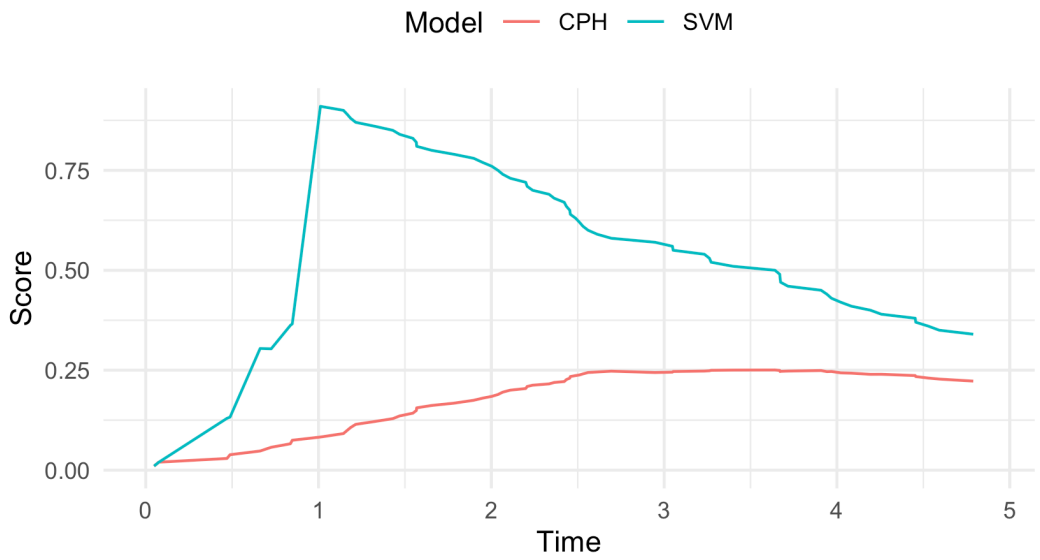


Figure 8.2: Prediction error curves for the CPH and SVM models from Chapter 7. x-axis is time and y-axis is the IGS computed at different time-points. The CPH (red) performs better than the SVM (blue) as it scores consistently lower. Trained and tested on randomly simulated data from **mlr3proba**.

8.4 Baselines and ERV

A common criticism of scoring rules is a lack of interpretability, for example, an IGS of 0.5 or 0.0005 has no meaning by itself, so below we present two methods to help overcome this problem.

The first method, is to make use of baselines for model comparison, which are models or values that can be utilised to provide a reference for a loss, they provide a universal method to judge all models of the same class by (Gressmann et al. 2018). In classification, it is possible to derive analytical baseline values, for example a Brier score is considered 'good' if it is below 0.25 or a log loss if it is below 0.693 (Figure 8.1), this is because these are the values obtained if you always predicted probabilties as 0.5, which is a reasonable basline guess in a binary classificaiton problem. In survival analysis, simple analytical expressions are not possible as losses are dependent on the unknown distributions of both the survival and censoring time. Therefore all experiments in survival analysis must include a baseline model that can produce a reference value in order to derive meaningful results. A suitable baseline model is the Kaplan-Meier estimator (Graf and Schumacher 1995; Lawless and Yuan 2010; Royston and Altman 2013), which is the simplest model that can consistently estimate the true survival function.

As well as directly comparing losses from a 'sophisticated' model to a baseline, one can also compute the percentage increase in performance between the sophisicated and baseline models, which produces a measure of explained residual variation (ERV) (Edward L. Korn and Simon 1990; Edward L. Korn and Simon 1991). For any survival loss L, the ERV is,

$$R_L(S, B) = 1 - \frac{L|S}{L|B}$$

where $L|S$ and $L|B$ is the loss computed with respect to predictions from the sophisticated and baseline models respectively.

The ERV interpretation makes reporting of scoring rules easier within and between experiments. For example, say in experiment A we have $L|S = 0.004$ and $L|B = 0.006$, and in experiment B we have $L|S = 4$ and $L|B = 6$. The sophisticated model may appear worse at first glance in experiment A (as the losses are very close) but when considering the ERV we see that the performance increase is identical (both $R_L = 33\%$), thus providing a clearer way to compare models.

9

Evaluating Survival Time

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

When it comes to evaluating survival time predictions, there are few measures available at our disposal. As a result of survival time predictions being uncommon compared to other prediction types (Section 4.3), there are limited survival time evaluation measures in the literature. To our knowledge, there are no specialised 'survival time measures', instead regression measures are used by ignoring censored observations.

Before presenting these measures, consider what happens when censored observations are discarded. If censoring is truly independent, occurs randomly, and is *very* limited in the data, then there is little harm in discarding observations and treating this as a regression problem. However, if censoring is not independent, then discarding censored observations will lead to missing valuable insights about the model. For example, say the task of interest is to predict the probability of death due to kidney failure and patients are censored if they receive a transplant - this is clearly a competing risk as receiving a transplant greatly reduces the probability of death. If one were to predict the time to death for all patients and to not evaluate the quality of prediction for censored patients, then it would only be possible to conclude about the model's performance for those who do not receive a transplant. On the surface this may appear to be of value, however, if at the time of prediction it is impossible to know who will receive a transplant (perhaps because the dataset omits relevant information such as time of hospital admission, wait on register, etc.), then for a given prediction for an observation, it would be impossible to know if the prediction is trustworthy - it would be if that patient does not receive a transplant, but would not be if they do not. In short, it is essential that predictions for individuals who end up being censored, are as good as those who are not, simply because there is no method to know which group observations will eventually fall into.

It is interesting to consider if IPCW strategies would compensate for this deficiency, however as we were unable to find research into this method, we have only included measures that we term 'censoring-ignored regression measures', which are presented in (P. Wang, Li, and Reddy 2019).

9.1 Distance measures

Survival time measures are often referred to as 'distance' measures as they measure the distance between the true, $(t, \delta = 1)$, and predicted, $\hat{t}$, values. These are presented in turn with brief descriptions of their interpretation.

Censoring-ignored mean absolute error, MAE_C

In regression, the mean absolute error (MAE) is a popular measure because it is intuitive to understand as it measures the absolute difference between true and predicted outcomes; hence intuitively one can understand that a model predicting a height of 175cm is clearly better than one predicting a height of 180cm, for a person with true height of 174cm.

$$MAE_C(\hat{\mathbf{t}}, \mathbf{t}, \delta) = \frac{1}{d} \sum_{i=1}^{m} \delta_i |t_i - \hat{t}_i|$$

Where d is the number of uncensored observations in the dataset, $d = \sum_i \delta_i$.

Censoring-ignored mean squared error

In comparison to MAE, the mean squared error (MSE), computes the squared differences between true and predicted values. While the MAE provides a smooth, linear, 'penalty' for increasingly poor predictions (i.e., the difference between an error of predicting 2 vs. 5 is still 3), but the square in the MSE means that larger errors are quickly magnified (so the difference in the above example is 9). By taking the mean over all predictions, the effect of this inflation is to increase the MSE value as larger mistakes are made.

$$MSE_C(\hat{\mathbf{t}}, \mathbf{t}, \delta) = \frac{1}{d} \sum_{i=1}^{m} \delta_i (t_i - \hat{t}_i)^2$$

Where d is again the number of uncensored observations in the dataset, $d = \sum_i \delta_i$.

Censoring-adjusted root mean squared error

Finally, the root mean squared error (RMSE), is simply the square root of the MSE. This allows interpretation on the original scale (as opposed to the squared scale produced by the MSE). Given the inflation effect for the MSE, the RMSE will be larger than the MAE as increasingly poor predictions are made; it is common practice for the MAE and RMSE to be reported together.

$$RMSE_C(\hat{\mathbf{t}}, \mathbf{t}, \delta) = \sqrt{MSE_C(\hat{\mathbf{t}}, \mathbf{t}, \delta)}$$

9.2 Over- and under-predictions

All of these distance measures assume that the error for an over-prediction ($\hat{t} > t$) should be equal to an under-prediction ($\hat{t} < t$), i.e., that it is 'as bad' if a model predicts an outcome time being 10 years longer than the truth compared to being 10 years shorter. In

the survival setting, this assumption is often invalid as it is generally preferred for models to be overly cautious, hence to predict negative events to happen sooner (e.g., predict a life-support machine fails after three years not five if the truth is actually four) and to predict positive events to happen later (e.g., predict a patient recovers after four years not two if the truth is actually three). A simple method to incorporate this imbalance between over- and under-predictions is to add a weighting factor to any of the above measures, for example the MAE_C might become

$$MAE_C(\hat{\mathbf{t}}, \mathbf{t}, \delta, \lambda, \mu, \phi) = \frac{1}{d} \sum_{i=1}^{m} \delta_i |(t_i - \hat{t}_i)[\lambda \mathbb{I}(t_i > \hat{t}_i) + \mu \mathbb{I}(t_i < \hat{t}_i) + \phi \mathbb{I}(t_i = \hat{t}_i)]|$$

where λ, μ, ϕ are any Real number to be used to weight over-, under-, and exact-predictions, and d is as above. The choice of these are highly context dependent and could even be tuned.

10

Choosing Measures

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

After reading this part of the book, evaluating survival analysis models may appear more daunting than regression and classification settings, which, in contrast, have fewer (common) measures to choose from. In regression problems, the RMSE and MAE are common choices for evaluating how far predictions are from the truth. In classification, the Brier score or logloss may be used to evaluate probabilistic predictions and the accuracy score or TPR/TNR/FPR/FNR are common for deterministic predictions. In contrast, there are many more measures in survival analysis which are necessarily more complex, due to the need to handle censoring with many possible methods for doing so. Therefore, this final chapter aims to provide some simple to follow guidelines for selecting measures for different types of experiments.

10.1 Defining the experiment

Experiments may be performed to make predictions for new data, compare the performance of multiple models ('benchmark experiments'), investigate patterns in observed data, or some combination of these. Each experiment requires different choices of measures, with different levels of strictness applied to measure assumptions.

10.1.1 Predictive experiments

In the real world, predictive experiments are most common. These are now daily occurrences as machine learning models are routinely deployed on servers to make ongoing predictions. In these cases, the exact task must be precisely stated before any model is deployed and evaluated. Common survival problems to solve include:

1. Identifying low and high risk groups in new data (for resource allocation);
2. Predicting the survival distribution for an individual over time; and
3. Predicting the survival probability for an individual at a specific time.

The first of these is a discrimination problem and it is therefore most important that the model optimises corresponding measures and that measure assumptions are justified. However, even this task may be more complex than it initially seems. For example, while some papers have shown flaws in Harrell's C (Gönen and Heller 2005; Rahman et al. 2017; Schmid and Potapov 2012; Uno et al. 2007), others have demonstrated that common alternatives yield very similar results (Rahman et al. 2017; Therneau and Atkinson 2020) and moreover some prominent alternatives may be harder to interpret due to high variance (Rahman et al. 2017; Schmid and Potapov 2012). In predictive experiment that may require more level of automation, it is important to be careful of C-hacking (Section 6.1.2) and to avoid overoptimistic results. Hence one should not compute a range of concordance indices and report the maximum but instead calculate a single discrimination measure and then establish a pre-defined threshold to determine if the deployed model is optimal, a natural threshold would be 0.5 as anything above this is better than a baseline model. Given Harrell's C to be increasingly over-optimistic with additional censoring (Rahman et al. 2017), it is advisable to use Uno's C instead.

If the task of interest is to predict survival distributions *over time*, then the choice of measure is more limited and only the RCLL and the proper Graf score are recommended. Both these measures can only be interpreted with respect to a baseline so use of the ERV representation is strongly recommended. As with the previous task, establishing a threshold for performance is essential prior to deployment and for ongoing evaluation. It is less clear in these cases what this threshold might be, but the simplest starting point would be to ensure that the model continues to outperform the baseline or a simpler gold-standard model (e.g., the Cox PH).

The final task of interest differs from the previous by only making predictions at a specific time. In this case, prediction error curves, and single-time point calibration measures can be used, as well as scoring rules with shorter cut-offs (i.e., the upper limit of the integral). It is imperative that model performance is never extrapolated outside of the pre-specified time.

10.1.2 Benchmark experiments

When conducting benchmark experiments, it is advisable to use a spread of measures so that results can be compared across various properties. In this case, models should be tested against discrimination, calibration, and overall predictive ability (i.e., with scoring rules). As models make different types of predictions, results from these experiments should be limited to metrics that are directly comparable, in other words, two models should only be compared based on the *same* metric. In benchmark experiments, models are compared across the same data and same resampling strategy, hence measure assumptions become less important as they are equally valid or flawed for all models. For example, if one dataset has particularly high amounts of censoring leading to an artificially higher concordance index, then this bias would affect all models equally and the overall experiment would not be affected. Hence, in these experiments it suffices to pick one or two measures for concordance, discrimination, and predictive ability, without having to be overly concerned with the individual metric.

This book recommends using Harrell's C and Uno's C for concordance as these are simplest to compute and including both enables more confidence in model comparison, i.e., if a model outperforms another with respect to both these measures then there can be higher confidence in drawing statements about the model's discriminatory power. For calibration, D-calibration is recommended as it can be meaningfully compared between models, and the RCLL is recommended for a scoring rule (which is proper for outcome-independent

censoring). No distance measure is recommended as these do not apply to the vast majority of models. All these measures can be used for automated tuning, in the case of discrimination tuning to Harrell's C alone should suffice (without also tuning to Uno's C).

10.1.3 Investigation

Investigating patterns in observed data is increasingly common as model interpretability methods have become more accessible (Molnar 2019). Before data can be investigated, any model that is trained on the data must first be demonstrated to be a good fit to the data. A model's fit to data can also be evaluated by resampling the data (Chapter 3) and evaluating the predictions. In this case, it is important to choose measures that are interpretable and have justified assumptions. Calibration measures are particularly useful for evaluating if a model is well fit to data, and any of the methods described in Chapter 7 are recommended for this purpose. Discrimination measures *may* be useful, however, given how susceptible they are to censoring, they can be difficult to interpret on their own, and the same is true for scoring rules. One method to resolve ambiguity is to perform a benchmark experiment of multiple models on the same data (ideally with some automated tuning) and then select the best model from this experiment and refit it on the full data (Becker, Schneider, and Fischer 2024) – this is a robust, empirical method that demonstrates a clear trail to selecting a model that outperforms other potential candidates. When investigating a dataset, one may also consider using different measures to assess algorithmic fairness (R. Sonabend et al. 2022), any measure that can be optimised (i.e., where the lowest or highest value is the best) may be used in this case. Finally, there are survival adaptations to the well-known AIC (Liang and Zou 2008) and BIC (Volinsky and Raftery 2000) however as these are generally only applicable to 'classical' models (Chapter 11), they are out of scope for this book and hence have not been discussed.

10.2 Conclusions

This part of the book focused on survival measures. Measures may be used to evaluate model predictions, to tune a model, or to train a model (e.g., in boosting or neural networks). Unlike other settings, there are many different choices of survival measures and it can be hard to determine which to use and when. In practice, like many areas of Statistics, the most important factor is to clearly define any experiment upfront and to be clear about which measures will be used and why. As a rule of thumb, good choices for measures are Harrell's C for evaluating discrimination, with Uno's C supporting findings, D-calibration for calibration, and the RCLL for evaluating overall predictive ability from distribution predictions. Finally, if you are restricted to a single measure choice (e.g., for automated tuning or continuous evaluation of deployed models), then we recommended selecting a scoring rule such as RCLL which captures information about calibration and discrimination simultaneously.

Part III

Models

11

Classical Models

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

TODO

- change chapter name (don't use "classical")
-

11.1 A Review of Classical Survival Models

This chapter provides a brief review of classical survival models before later chapters move on to machine learning models. 'Classical' models are defined with a very narrow scope in this book: low-complexity models that are either non-parametric or have parameters that can be fit with maximum likelihood estimation (or an equivalent method). In contrast, 'machine learning' (ML) models require more intensive model fitting procedures such as recursion or iteration. The classical models in this paper are fast to fit and highly interpretable, though can be inflexible and may make unreasonable assumptions. Whereas the ML models are more flexible with hyper-parameters however are computationally more intensive (both in terms of speed and storage), require tuning to produce 'good' results, and are often a 'black-box' with difficult interpretation.

As classical survival models have been studied extensively for decades, these are only discussed briefly here, primarily these are of interest as many of these models will be seen to influence machine learning extensions. The scope of the models discussed in this chapter is limited to the general book scope (Section 4.2), i.e. single event with right-censoring and no competing-risks, though in some cases these are discussed.

There are several possible taxonomies for categorising statistical models, these include:

- Parametrisation Type: One of non-, semi-, or fully-parametric. \ Non-parametric models assume that the data distribution cannot be specified with a finite set of parameters. In

contrast, fully-parametric models assume the distribution can be specified with a finite set of parameters. Semi-parametric models are a hybrid of the two and are formed of a finite set of parameters *and* an infinite-dimensional 'nuisance' parameter.

- Conditionality Type: One of unconditional or conditional. A conditional prediction is one that makes use of covariates in order to condition the prediction on each observation. Unconditional predictors, which are referred to below as 'estimators', ignore covariate data and make the same prediction for all individuals.
- Prediction Type: One of ranking, survival time, or distribution (Section 4.3).

Table 11.1 summarises the models discussed below into the taxonomies above for reference. Note that the Cox model is listed as predicting a continuous ranking, and not a survival distribution, which may appear inconsistent with other definitions. The reason for this is elaborated upon in Chapter 15. Though the predict-type taxonomy is favoured throughout this book, it is clearer to review classical models in increasing complexity, beginning with unconditional estimators before moving onto semi-parametric continuous ranking predictions, and finally conditional distribution predictors. The review is brief with mathematics limited to the model fundamentals but not including methods for parameter estimation. Also the review is limited to the 'basic' model specification and common extensions such as regularization are not discussed though they do exist for many of these models.

All classical models are highly transparent and accessible, with decades of research and many off-shelf implementations. Predictive performance of each model is briefly discussed as part of the review and then again in (R. E. B. Sonabend 2021).

Table 11.1: Table of models discussed in this literature review, classified by parametrisation, prediction type, and conditionality.

Model[1]	Parametrisation[2]	Prediction[3]	Conditionality
Kaplan-Meier	Non	Distr.	Unconditional
Nelson-Aalen	Non	Distr.	Unconditional
Akritas	Non	Distr.	Conditional
Cox PH	Semi	Rank	Conditional
Parametric PH	Fully	Distr.	Conditional
Accelerated Failure Time	Fully	Distr.	Conditional
Proportional Odds	Fully	Distr.	Conditional
Flexible Spline	Fully	Distr.	Conditional

* 1. All models are implemented in the R package **survival** (Therneau 2015) with the exception of flexible splines, implemented in **flexsurv** (Jackson 2016), and the Akritas estimator in **survivalmodels** (R. Sonabend 2020). * 2. Non = non-parametric, Semi = semi-parametric, Fully = fully-parametric. * 3. Distr. = distribution, Rank = ranking.

11.1.1 Non-Parametric Distribution Estimators

Unconditional Estimators

Unconditional non-parametric survival models assume no distribution for survival times and estimate the survival function using simple algorithms based on observed outcomes and no covariate data. The two most common methods are the Kaplan-Meier estimator (**KaplanMeier1958?**), which estimates the average survival function of a training dataset,

and the Nelson-Aalen estimator (Aalen 1978; Nelson 1972), which estimates the average cumulative hazard function of a training dataset.

The Kaplan-Meier estimator of the survival function is given by

$$\hat{S}_{KM}(\tau|\mathcal{D}_{train}) = \prod_{t \in \mathcal{U}_O, t \leq \tau} \left(1 - \frac{d_t}{n_t}\right) \tag{11.1}$$

As this estimate is so important in survival models, this book will always use the symbol $\hat{S}_{KM}$ to refer to the Kaplan-Meier estimate of the average survival function fit on training data (T_i, Δ_i). Another valuable function is the Kaplan-Meier estimate of the average survival function of the *censoring* distribution, which is the same as above but estimated on $(T_i, 1 - \Delta_i)$, this will be denoted by $\hat{G}_{KM}$.

The Nelson-Aalen estimator for the cumulative hazard function is given by

$$\hat{H}(\tau|\mathcal{D}_{train}) = \sum_{t \in \mathcal{U}_O, t \leq \tau} \frac{d_t}{n_t} \tag{11.2}$$

The primary advantage of these models is that they rely on heuristics from empirical outcomes only and don't require any assumptions about the form of the data. To train the models they only require (T_i, Δ_i) and both return a prediction of $\mathcal{S} \subseteq \text{Distr}(\mathcal{T})$ ((**box-task-surv?**)). In addition, both simply account for censoring and can be utilised in fitting other models or to estimate unknown censoring distributions. The Kaplan-Meier and Nelson-Aalen estimators are both consistent estimators for the survival and cumulative hazard functions respectively.

Utilising the relationships provided in (Section 4.3), one could write the Nelson-Aalen estimator in terms of the survival function as $\hat{S}_{NA} = \exp(-\hat{H}(\tau|\mathcal{D}_{train}))$. It has been demonstrated that $\hat{S}_{NA}$ and $\hat{S}_{KM}$ are asymptotically equivalent, but that $\hat{S}_{NA}$ will provide larger estimates than $\hat{S}_{KM}$ in smaller samples (Colosimo et al. 2002). In practice, the Kaplan-Meier is the most widely utilised non-parametric estimator in survival analysis and is the simplest estimator that yields consistent estimation of a survival distribution; it is therefore a natural, and commonly utilised, 'baseline' model (Harald Binder and Schumacher 2008; Herrmann et al. 2021; Huang et al. 2020; P. Wang, Li, and Reddy 2019): estimators that other models should be 'judged' against to ascertain their overall performance (Chapter 5).

Not only can these estimators be used for analytical comparison, but they also provide intuitive methods for graphical calibration of models (Section 7.2). These models are never stuidied for prognosis directly but as baselines, components of complex models (Chapter 15), or graphical tools (Habibi et al. 2018; Jager et al. 2008; Moghimi-dehkordi et al. 2008). The reason for this is due to them having poor predictive performance as a result of omitting explanatory variables in fitting. Moreover, if the data follows a particular distribution, parametric methods will be more efficient (P. Wang, Li, and Reddy 2019).

Conditional Estimators

The Kaplan-Meier and Nelson-Aalen estimators are simple to compute and provide good estimates for the survival time distribution but in many cases they may be overly-simplistic. Conditional non-parametric estimators include the advantages described above (no assumptions about underlying data distribution) but also allow for conditioning the estimation on the covariates. This is particularly useful when estimating a censoring distribution that

may depend on the data (Chapter 5). However predictive performance of conditional non-parametric estimators decreases as the number of covariates increases, and these models are especially poor when censoring is feature-dependent (Gerds and Schumacher 2006).

The most widely used conditional non-parametric estimator for survival analysis is the Akritas estimator (Akritas 1994) defined by[1]

$$\hat{S}(\tau|X^*, \mathcal{D}_{train}, \lambda) = \prod_{j:T_j \leq \tau, \Delta_j = 1} \Big(1 - \frac{K(X^*, X_j|\lambda)}{\sum_{l=1}^{n} K(X^*, X_l|\lambda)\mathbb{I}(T_l \geq T_j)}\Big)$$

where K is a kernel function, usually $K(x, y|\lambda) = \mathbb{I}(|\hat{F}_X(x) - \hat{F}_X(y)| < \lambda), \lambda \in (0, 1]$, $\hat{F}_X$ is the empirical distribution function of the training data, $X_1, ..., X_n$, and λ is a hyper-parameter. The estimator can be interpreted as a conditional Kaplan-Meier estimator which is computed on a neighbourhood of subjects closest to X^* (Blanche, Dartigues, and Jacqmin-Gadda 2013). To account for tied survival times, the following adaptation of the estimator is utilised (Blanche, Dartigues, and Jacqmin-Gadda 2013)

$$\hat{S}(\tau|X^*, \mathcal{D}_{train}, \lambda) = \prod_{t \in \mathcal{U}_O, t \leq \tau} \Big(1 - \frac{\sum_{j=1}^{n} K(X^*, X_j|\lambda)\mathbb{I}(T_j = t, \Delta_j = 1)}{\sum_{j=1}^{n} K(X^*, X_j|\lambda)\mathbb{I}(T_j \geq t)}\Big) \tag{11.3}$$

If $\lambda = 1$ then $K(\cdot|\lambda) = 1$ and the estimator is identical to the Kaplan-Meier estimator.

The non-parametric nature of the model is highlighted in (Equation 11.3), in which both the fitting and predicting stages are combined into a single equation. A new observation, X^*, is compared to its nearest neighbours from a training dataset, $\mathcal{D}_{train}$, without a separated fitting procedure. One could consider splitting fitting and predicting in order to clearly separate between training and testing data. In this case, the fitting procedure is the estimation of $\hat{F}_X$ on training data and the prediction is given by (Equation 11.3) with $\hat{F}_X$ as an argument. This separated fit/predict method is implemented in **survivalmodels** (R. Sonabend 2020). As with other non-parametric estimators, the Akritas estimator can still be considered transparent and accessible. With respect to predictive performance, the Akritas estimator has more explanatory power than non-parametric estimators due to conditioning on covariates, however this is limited to a very small number of variables and therefore this estimator is still best placed as a conditional baseline.

11.1.2 Continuous Ranking and Semi-Parametric Models: Cox PH

The Cox Proportional Hazards (CPH) (Cox 1972), or Cox model, is likely the most widely known semi-parametric model and the most studied survival model (Habibi et al. 2018; Moghimi-dehkordi et al. 2008; Reid 1994; P. Wang, Li, and Reddy 2019). The Cox model assumes that the hazard for a subject is proportionally related to their explanatory variables, $X_1, ..., X_n$, via some baseline hazard that all subjects in a given dataset share ('the PH assumption'). The hazard function in the Cox PH model is defined by

$$h(\tau|X_i) = h_0(\tau)\exp(X_i\beta)$$

where h_0 is the non-negative *baseline hazard function* and $\beta = \beta_1, ..., \beta_p$ where $\beta_i \in \mathbb{R}$ are coefficients to be fit. Note the proportional hazards (PH) assumption can be seen as

[1]Arguments and parameters are separated in function signatures by a pipe, '|', where variables to the left are parameters (free variables) and those to the right are arguments (fixed). In this equation, τ is a parameter to be set by the user, and $X^*, \mathcal{D}_{train}, \lambda$ are fixed arguments. This could therefore be simplified to $\hat{S}(\tau)$ to only include free variables.

the estimated hazard, $h(\tau|X_i)$, is directly proportional to the model covariates $\exp(X_i\beta)$. Whilst a form is assumed for the 'risk' component of the model, $\exp(X_i\beta)$, no assumptions are made about the distribution of h_0, hence the model is semi-parametric.

The coefficients, β, are estimated by maximum likelihood estimation of the 'partial likelihood' (Cox 1975), which only makes use of ordered event times and does not utilise all data available (hence being 'partial'). The partial likelihood allows study of the informative β-parameters whilst ignoring the nuisance h_0. The predicted linear predictor, $\hat{\eta} := X^*\hat{\beta}$, can be computed from the estimated $\hat{\beta}$ to provide a ranking prediction.

Inspection of the model is also useful without specifying the full hazard by interpreting the coefficients as 'hazard ratios'. Let $p = 1$ and $\hat{\beta} \in \mathbb{R}$ and let $X_i, X_j \in \mathbb{R}$ be the covariates of two training observations, then the *hazard ratio* for these observations is the ratio of their hazard functions,

$$\frac{h(\tau|X_i)}{h(\tau|X_j)} = \frac{h_0(\tau)\exp(X_i\hat{\beta})}{h_0(\tau)\exp(X_j\hat{\beta})} = \exp(\hat{\beta})^{X_i - X_j}$$

If $\exp(\hat{\beta}) = 1$ then $h(\tau|X_i) = h(\tau|X_j)$ and thus the covariate has no effect on the hazard. If $\exp(\hat{\beta}) > 1$ then $X_i > X_j \to h(\tau|X_i) > h(\tau|X_i)$ and therefore the covariate is positively correlated with the hazard (increases risk of event). Finally if $\exp(\hat{\beta}) < 1$ then $X_i > X_j \to h(\tau|X_i) < h(\tau|X_i)$ and the covariate is negatively correlated with the hazard (decreases risk of event).

Interpreting hazard ratios is known to be a challenge, especially by clinicians who require simple statistics to communicate to patients (Sashegyi and Ferry 2017; Spruance et al. 2004). For example the full interpretation of a hazard ratio of '2' for binary covariate X would be: 'assuming that the risk of death is constant at all time-points then the instantaneous risk of death is twice as high in a patient with X than without'. Simple conclusions are limited to stating if patients are at more or less risk than others in their cohort. Further disadvantages of the model also lie in its lack of real-world interpretabilitity, these include (Reid 1994):

- the PH assumption may not be realistic and the risk of event may not be constant over time;
- the estimated baseline hazard from a non-parametric estimator is a discrete step-function resulting in a discrete survival distribution prediction despite time being continuous; and
- the estimated baseline hazard will be constant after the last observed time-point in the training set (Gelfand et al. 2000).

Despite these disadvantages, the model has been demonstrated to have excellent predictive performance and routinely outperforms (or at least does not underperform) sophisticated ML models (Michael F. Gensheimer and Narasimhan 2018; Luxhoj and Shyur 1997; Van Belle et al. 2011) (and (R. E. B. Sonabend 2021)). It's simple form and wide popularity mean that it is also highly transparent and accessible.

The next class of models address some of the Cox model disadvantages by making assumptions about the baseline hazard.

11.1.3 Conditional Distribution Predictions: Parametric Linear Models

Parametric Proportional Hazards

The CPH model can be extended to a fully parametric PH model by substituting the unknown baseline hazard, h_0, for a particular parameterisation. Common choices for dis-

tributions are Exponential, Weibull and Gompertz (John D. Kalbfleisch and Prentice 2011; P. Wang, Li, and Reddy 2019); their hazard functions are summarised in ((**tab-survivaldists?**)) along with the respective parametric PH model. Whilst an Exponential assumption leads to the simplest hazard function, which is constant over time, this is often not realistic in real-world applications. As such the Weibull or Gompertz distributions are often preferred. Moreover, when the shape parameter, γ, is 1 in the Weibull distribution or 0 in the Gompertz distribution, their hazards reduce to a constant risk ((Figure 11.1)). As this model is fully parametric, the model parameters can be fit with maximum likelihood estimation, with the likelihood dependent on the chosen distribution.

Table 11.2: Exponential, Weibull, and Gompertz hazard functions and PH specification.

Distribution[1]	$h_0(\tau)$[2]	$h(\tau\|X_i)$[3]
Exp(λ)	λ	$\lambda \exp(X_i\beta)$
Weibull(γ, λ)	$\lambda\gamma\tau^{\gamma-1}$	$\lambda\gamma\tau^{\gamma-1} \exp(X_i\beta)$
Gompertz(γ, λ)	$\lambda \exp(\gamma\tau)$	$\lambda \exp(\gamma\tau) \exp(X_i\beta)$

* 1. Distribution choices for baseline hazard. γ, λ are shape and scale parameters respectively. * 2. Baseline hazard function, which is the (unconditional) hazard of the distribution. * 3. PH hazard function, $h(\tau|X_i) = h_0(\tau) \exp(X_i\beta)$.

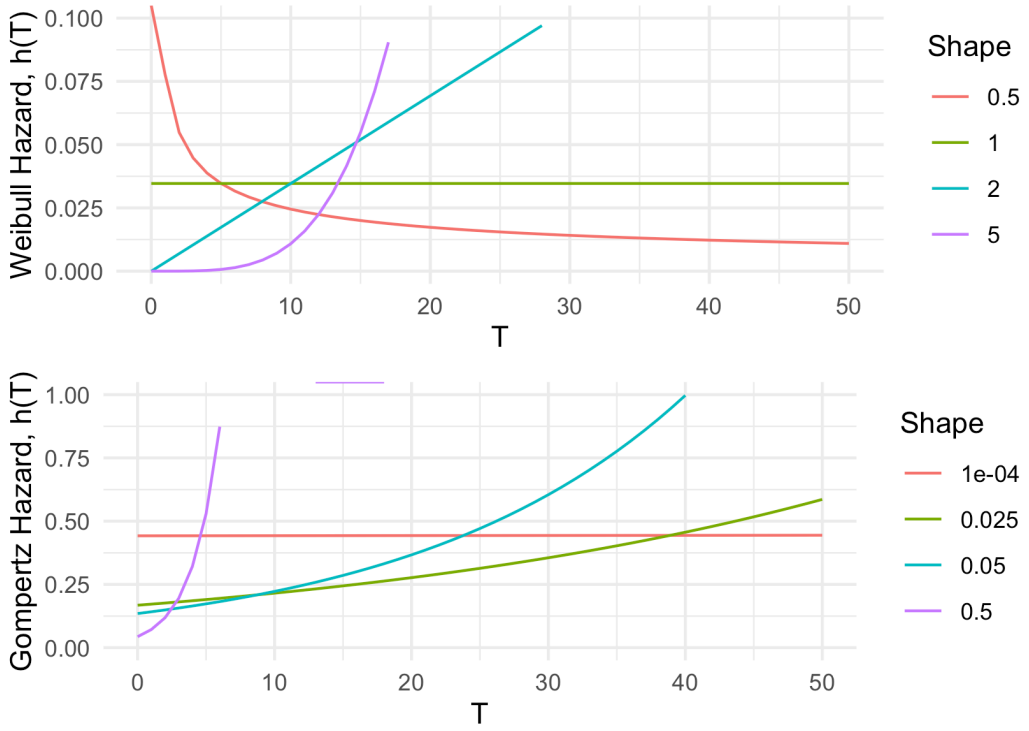


Figure 11.1: Comparing the hazard curves under Weibull and Gompertz distributions for varying values of the shape parameter; scale parameters are set so that each parametrisation has a median of 20. x-axes are time and y-axes are Weibull (top) and Gompertz (bottom) hazards as a function of time.

In the literature, the Weibull distribution tends to be favoured as the initial assumption for the survival distribution (Michael F. Gensheimer and Narasimhan 2018; Habibi et al. 2018; Hielscher et al. 2010; R. and J. 1968; Rahman et al. 2017), though Gompertz is often tested in death-outcome models for its foundations in modelling human mortality (Gompertz 1825). There exist many tests for checking the goodness-of-model-fit (**?@sec-eval-insample**) and the distribution choice can even be treated as a model hyper-parameter. Moreover it transpires that model inference and predictions are largely insensitive to the choice of distribution (Collett 2014; Reid 1994). In contrast to the Cox model, fully parametric PH models can predict absolutely continuous survival distributions, they do not treat the baseline hazard as a nuisance, and in general will result in more precise and interpretable predictions if the distribution is correctly specified (Reid 1994; Royston and Parmar 2002).

Whilst misspecification of the distribution tends not to affect predictions too greatly, PH models will generally perform worse when the PH assumption is not valid. PH models can be extended to include time-varying coefficients or model stratification (Cox 1972) but even with these adaptations the model may not reflect reality. For example, the predicted hazard in a PH model will be either monotonically increasing or decreasing but there are many scenarios where this is not realistic, such as when recovering from a major operation where risks tends to increase in the short-term before decreasing. Accelerated failure time models overcome this disadvantage and allow more flexible modelling, discussed next.

Accelerated Failure Time

In contrast to the PH assumption, where a unit increase in a covariate is a multiplicative increase in the hazard rate, the Accelerated Failure Time (AFT) assumption means that a unit increase in a covariate results in an acceleration or deceleration towards death (expanded on below). The hazard representation of an AFT model demonstrates how the interpretation of covariates differs from PH models,

$$h(\tau|X_i) = h_0(\exp(-X_i\beta)\tau)\exp(-X_i\beta)$$

where $\beta = (\beta_1, ..., \beta_p)$ are model coefficients. In contrast to PH models, the 'risk' component, $\exp(-X_i\beta)$, is the exponential of the *negative* linear predictor and therefore an increase in a covariate value results in a decrease of the predicted hazard. This representation also highlights how AFT models are more flexible than PH as the predicted hazard can be non-monotonic. For example the hazard of the Log-logistic distribution ((Figure 11.2)) is highly flexible depending on chosen parameters. Not only can the AFT model offer a wider range of shapes for the hazard function but it is more interpretable. Whereas covariates in a PH model act on the hazard, in an AFT they act on time, which is most clearly seen in the log-linear representation,

$$\log Y_i = \mu + \alpha_1 X_{i1} + \alpha_2 X_{i2} + ... + \alpha_p X_{ip} + \sigma\epsilon_i$$

where μ and σ are location and scale parameters respectively, $\alpha_1, ..., \alpha_p$ are model coefficients, and ϵ_i is a random error term. In this case a one unit increase in covariate X_{ij} means a α_j increase in the logarithmic survival time. For example if $\exp(X_i\alpha) = 0.5$ then i 'ages' at double the baseline 'speed'. Or less abstractly if studying the time until death from cancer then $\exp(X_i\alpha) = 0.5$ can be interpreted as 'the entire process from developing tumours to metastasis and eventual death in subject i is twice as fast than the normal', where 'normal' refers to the baseline when all covariates are 0.

Specifying a particular distribution for ϵ_i yields a fully-parametric AFT model. Common distribution choices include Weibull, Exponential, Log-logistic, and Log-Normal (John D.

Kalbfleisch and Prentice 2011; P. Wang, Li, and Reddy 2019). The Buckley-James estimator (Buckley and James 1979) is a semi-parametric AFT model that non-parametrically estimates the distribution of the errors however this model has no theoretical justification and is rarely fit in practice (Wei 1992). The fully-parametric model has theoretical justifications, natural interpretability, and can often provide a better fit than a PH model, especially when the PH assumption is violated (Patel, Kay, and Rowell 2006; Qi 2009; Zare et al. 2015).

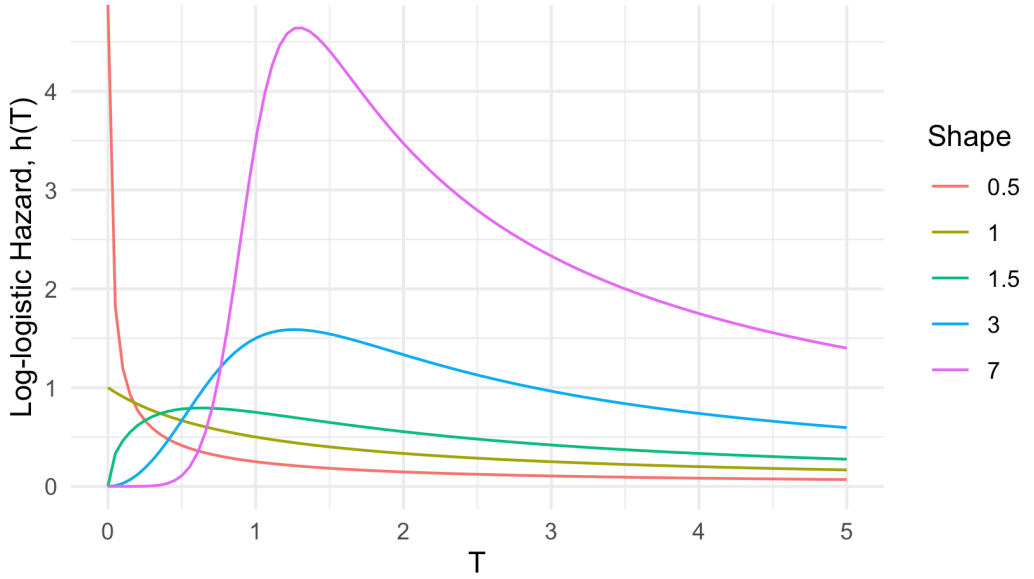


Figure 11.2: Log-logistic hazard curves with a fixed scale parameter of 1 and a changing shape parameter. x-axis is time and y-axis is the log-logistic hazard as a function of time.

Proportional Odds

Proportional odds (PO) models (Bennett 1983) fit a proportional relationship between covariates and the odds of survival beyond a time τ,

$$O_i(\tau) = \frac{S_i(\tau)}{F_i(\tau)} = O_0(\tau)\exp(X_i\beta)$$

where O_0 is the baseline odds.

In this model, a unit increase in a covariate is a multiplicative increase in the odds of survival after a given time and the model can be interpreted as estimating the log-odds ratio. There is no simple closed form expression for the partial likelihood of the PO model and hence in practice a Log-logistic distribution is usually assumed for the baseline odds and the model is fit by maximum likelihood estimation on the full likelihood (Bennett 1983).

Perhaps the most useful feature of the model is convergence of hazard functions (Kirmani and Gupta 2001), which states $h_i(\tau)/h_0(\tau) \to 1$ as $\tau \to \infty$. This property accurately reflects real-world scenarios, for example if comparing chemotherapy treatment on advanced cancer survival rates, then it is expected that after a long period (say 10 years) the difference in risk between groups is likely to be negligible. This is in contrast to the PH model that assumes the hazard ratios are constant over time, which is rarely a reflection of reality.

In practice, the PO model is harder to fit and is less flexible than PH and AFT models, both of which can also produce odds ratios. This may be a reason for the lack of popularity of the PO model, in addition there is limited off-shelf implementations (Collett 2014). Despite PO models not being commonly utilised, they have formed useful components of neural networks (**?@sec-surv-ml-models-nn**) and flexible parametric models (below).

Flexible Parametric Models – Splines

Royston-Parmar flexible parametric models (Royston and Parmar 2002) extend PH and PO models by estimating the baseline hazard with natural cubic splines. The model was designed to keep the form of the PH or PO methods but without the semi-parametric problem of estimating a baseline hazard that does not reflect reality (see above), or the parametric problem of misspecifying the survival distribution.

To provide an interpretable, informative and smooth hazard, natural cubic splines are fit in place of the baseline hazard. The crux of the method is to use splines to model time on a log-scale and to either estimate the log cumulative Hazard for PH models, $\log H(\tau|X_i) = \log H_0(\tau) + X_i\beta$, or the log Odds for PO models, $\log O(\tau|X_i) = \log O_0(\tau) + X_i\beta$, where β are model coefficients to fit, H_0 is the baseline cumulative hazard function and O_0 is the baseline odds function. For the flexible PH model, a Weibull distribution is the basis for the baseline distribution and a Log-logistic distribution for the baseline odds in the flexible PO model. $\log H_0(\tau)$ and $\log O_0(\tau)$ are estimated by natural cubic splines with coefficients fit by maximum likelihood estimation. The standard full likelihood is optimised, full details are not provided here. Between one and three internal knots are recommended for the splines and the placement of knots does not greatly impact upon the fitted model (Royston and Parmar 2002).

Advantages of the model include being: interpretable, flexible, can be fit with time-dependent covariates, and it returns a continuous function. Moreover many of the parameters, including the number and position of knots, are tunable, although Royston and Parmar advised against tuning and suggest often only one internal knot is required (Royston and Parmar 2002). A recent simulation study demonstrated that even with an increased number of knots (up to seven degrees of freedom), there was little bias in estimation of the survival and hazard functions (Bower et al. 2019). Despite its advantages, a 2018 review (Ng et al. 2018) found only twelve instances of published flexible parametric models since Royston and Parmar's 2002 paper, perhaps because it is more complex to train, has a less intuitive fitting procedure than alternatives, and has limited off-shelf implementations; i.e. is less transparent and accessible than parametric alternatives.

The PH and AFT models are both very transparent and accessible, though require slightly more expert knowledge than the CPH in order to specify the 'correct' underlying probability distribution. Interestingly whilst there are many papers comparing PH and AFT models to one another using in-sample metrics (**?@sec-eval-insample**) such as AIC (Georgousopoulou et al. 2015; Habibi et al. 2018; Moghimi-dehkordi et al. 2008; Zare et al. 2015), no benchmark experiments could be found for out-of-sample performance. PO and spline models are less transparent than PH and AFT models and are even less accessible, with very few implementations of either. No conclusions can be drawn about the predictive performance of PO or spline models due to a lack of suitable benchmark experiments.

12

Machine Learning Survival Models

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

12.1 A Survey of Machine Learning Models for Survival Analysis

These next sections provide a technical, critical survey of machine learning models proposed for survival analysis with the focus on the 'simpler' setup of non-competing risks. Models are separated into their different 'classes' (3), which exists as a natural taxonomy in machine learning. Each class review is then further separated by first discussing the simpler and more standard regression setting, before expanding into their survival framework. The focus is once again on the different predict types of the model, which enables clear exposition and discussion around how some areas have successfully dealt with the survival predictive problem, whereas others have fallen short.

This is not the first survey of machine learning models for survival analysis. A recent 2017 survey (P. Wang, Li, and Reddy 2019) focused on covering the breadth of machine learning models for survival analysis and this survey is recommended to the reader as a strong starting point to understand which ML models are available for survival analysis. However whilst this provides a comprehensive review and a 'big-picture' view, there is no discussion about how successful the discussed models are in solving the survival task.

A comprehensive survey of neural networks was presented by Schwarzer *et al.* (2000) (Schwarzer, Vach, and Schumacher 2000) in which the authors collected the many ways in which neural networks have been 'misused' in the context of survival analysis. This level of criticism is vital in the context of survival analysis and healthcare data as transparency and understanding are often prioritised over predictive performance. Whilst the survey in this book will try not to be as critical as the Schwarzer review, it will aim to discuss models and how well they actually solve the survival problem.

Historically, surveys have focused primarily on predictive performance, which is generally preferred for complex classification and regression tasks. However in the context of survival analysis, transparency is of the utmost importance and any model that does not solve the task it claims to, despite strong predictive performance, can be considered sub-optimal.

The survey will also examine the accessibility of survival models. A model need not be open-source to be accessible, but it should be 'user-friendly' and not require expert cross-domain knowledge. For example, a neural network may require knowledge of complex model building, but if set-up correctly could be handled without medical or survival knowledge. Whereas a Gaussian Process requires knowledge of the model class, simulation, (usually) Bayesian modelling, and also survival analysis.

(3) provides information about the models reviewed in this survey, including a model reference for use in the (R. E. B. Sonabend 2021) benchmark experiment, the predict types of the model, and in which R package it is implemented.

Table 12.1: Summarising the models discussed in (Section 12.1) by their model class and respective survival task.

Class[1]	Name[2]	Authors (Year)[3]	Task[4]	Implementation[5]
RF	RRT	LeBlanc and Crowley (1992) (LeBlanc and Crowley 1992)	Rank	**rpart** (Therneau and Atkinson 2019)
RF	RSDF-DEV	Hothorn *et al.* (2004) (Hothorn et al. 2004)	Prob.	**ipred** (Peters and Hothorn 2019)
RF	RRF	Ishwaran *et al.* (2004) (H. Ishwaran et al. 2004)	Rank	-
RF	RSCIFF	Hothorn *et al.* (2006) (Hothorn et al. 2005)	Det., Prob.	**party** (Hothorn, Hornik, and Zeileis 2006), **partykit** (Hothorn and Zeileis 2015)
RF	RSDF-STAT	Ishwaran *et al.* (2008) (B. H. Ishwaran et al. 2008)	Prob.	**randomForestSRC** (H. Ishwaran and Kogalur 2018), **ranger** (Wright and Ziegler 2017)
GBM	GBM-COX	Ridgeway (1999) (Ridgeway 1999) & Buhlmann (2007) (Buhlmann and Hothorn 2007)	Prob.	**mboost** (Hothorn et al. 2020), **xgboost** (T. Chen et al. 2020), **gbm** (Greenwell et al. 2019)
GBM	CoxBoost	Binder & Schumacher (2008) (Harald Binder and Schumacher 2008)	Prob.	**CoxBoost** (Harold Binder 2013)

Class[1]	Name[2]	Authors (Year)[3]	Task[4]	Implementation[5]
GBM	GBM-AFT	Schmid & Hothorn (2008) (Schmid and Hothorn 2008b)	Det.	**mboost**, **xgboost**
GBM	GBM-BUJAR	Wang & Wang (2010) (Z. Wang and Wang 2010)	Det.	**bujar** (Z. Wang 2019)
GBM	GBM-GEH	Johnson & Long (2011) (Johnson and Long 2011)	Det.	**mboost**
GBM	GBM-UNO	Mayr & Schmid (2014) (Mayr and Schmid 2014)	Rank	**mboost**
SVM	SVCR	Shivaswamy *et al.* (2007) (Shivaswamy, Chu, and Jansche 2007)	Det.	**survivalsvm** (Fouodo et al. 2018)
SVM	SSVM-Rank	Van Belle *et al.* (2007) (Van Belle et al. 2007)	Rank	**survivalsvm**
SVM	SVRc	Khan and Zubek (2008) (Khan and Bayer Zubek 2008)	Det.	-
SVM	SSVM-Hybrid	Van Belle (2011) (Van Belle et al. 2011)	Det.	**survivalsvm**
SVM	SSVR-MRL	Goli *et al.* (2016) (Goli, Mahjub, Faradmal, and Soltanian 2016; Goli, Mahjub, Faradmal, Mashayekhi, et al. 2016)	Det.	-
ANN	ANN-CDP	Liestøl *et al.* (1994) (Liestol, Andersen, and Andersen 1994)	Prob.	-
ANN	ANN-COX	Faraggi and Simon (1995) (Faraggi and Simon 1995)	Rank	-
ANN	PLANN	Biganzoli *et al.* (1998) (Biganzoli et al. 1998)	Prob.	-
ANN	COX-NNET	Ching *et al.* (2018) (Ching, Zhu, and Garmire 2018)	Prob.	**cox-nnet*** (Ching 2015)

Class[1]	Name[2]	Authors (Year)[3]	Task[4]	Implementation[5]
ANN	DeepSurv	Katzman *et al.* (2018) (Katzman et al. 2018)	Prob.	**survivalmodels** (R. Sonabend 2020)
ANN	DeepHit	Lee *et al.* (2018) (C. Lee et al. 2018)	Prob.	**survivalmodels**
ANN	Nnet-survival	Gensheimer & Narasimhan (2019) (Michael F. Gensheimer and Narasimhan 2019)	Prob.	**survivalmodels**
ANN	Cox-Time	Kvamme *et al.* (2019) (Kvamme, Borgan, and Scheel 2019)	Prob.	**survivalmodels**
ANN	PC-Hazard	Kvamme & Borgan (2019) (**Kvamme2019?**)	Prob.	**survivalmodels**
ANN	RankDeepSurv	Jing *et al.* (2019) (Jing et al. 2019)	Det.	**RankDeepSurv***,† (Jing et al. 2018)
ANN	DNNSurv	Zhao & Fend (2020) (Zhao and Feng 2020)	Prob.	**survivalmodels**

* 1. Model Class. RSF – Random Survival Forest; GBM – Gradient Boosting Machine; SVM – Support Vector Machine; ANN – Artificial Neural Network. There is some abuse of notation here as some of the RSFs are actually decision trees and some GBMs do not use gradient boosting. * 2. Model identifier used in this section and (R. E. B. Sonabend 2021). * 3. Authors and year of publication, for RSFs this is the paper most attributed to the algorithm. * 4. Survival task type: Deterministic (Det.), Probabilistic (Prob.), Ranking (Rank). * 5. If available in R then the package in which the model is implemented, otherwise '$*$' signifies a model is only available in Python. With the exception of DNNSurv, all ANNs in **survivalmodels** are implemented from the Python package **pycox** (Kvamme 2018) with **reticulate** (Ushey, Allaire, and Tang 2020). * † – Code available to create model but not implemented 'off-shelf'.

13

Tree-Based Methods

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

13.1 Random Forests

13.1.1 Random Forests for Regression

Random forests are a composite algorithm built by fitting many simpler component models, decision trees, and then averaging the results of predictions from these trees. Decision trees are first briefly introduced before the key 'bagging' algorithm that composes these trees to a random forest. Woodland terminology is used throughout this subsection.

Decision Trees

Decision trees are a common model class in machine learning and have the advantage of being (relatively) simple to implement and highly interpretable. A decision tree takes a set of inputs and a given *splitting rule* in order to create a series of splits, or branches, in the tree that culminates in a final *leaf*, or *terminal node*. Each terminal node has a corresponding prediction, which for regression is usually the sample mean of the training outcome data. This is made clearer by example, (Figure 13.1) demonstrates a decision tree predicting the miles per gallon (`mpg`) of a car from the `mtcars` (Henderson and Velleman 1981) dataset. With this tree a new prediction is made by feeding the input variables from the top to the bottom, for example given new data, $x = \{\grave{}wt\acute{} = 3, \grave{} disp\acute{} = 250\}$, then in the first split the right branch is taken as $\texttt{wt} = 3 > 2.32$ and in the second split the left branch is taken as $\texttt{disp} = 250 \leq 258$, therefore the new data point 'lands' in the final leaf and is predicted to have an `mpg` of 20.8. This value of 20.8 arises as the sample mean of `mpg` for the 11 (which can be seen in the box) observations in the training data who were sorted into this terminal node. Algorithmically, as splits are always binary, predictions are simply a series of conditional logical statements.

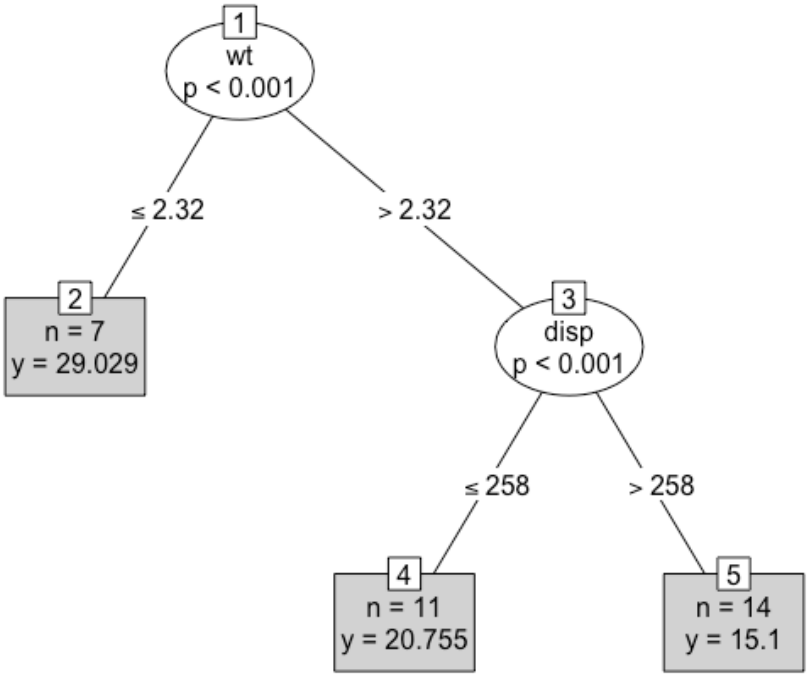


Figure 13.1: Demonstrating classification trees using the `mtcars` (Henderson and Velleman 1981) dataset and the **party** (Hothorn, Hornik, and Zeileis 2006) package. Ovals are leaves, which indicate the variable that is being split. Edges are branches, which indicate the cut-off at which the variable is split. Rectangles are terminal nodes and include information about the number of training observations in the node and the terminal node prediction.

Splitting Rules

Precisely how the splits are derived and which variables are utilised is determined by the splitting rule.[1] In regression, the most common splitting rule is to select the cut-off for a given variable that minimises the mean squared error in each hypothetical resultant leaf. The goal is to find the variable and cutoff that leads to the greatest difference between the two resultant leaves and thus the maximal homogeneity within each leaf. For all decision tree and random forest algorithms going forward, let L denote some leaf, then let L_{xy}, L_x, L_y respectively be the set of observations, features, and outcomes in leaf L. Let $L_{y;i}$ be the ith outcome in L_y and finally let $L_{\bar{y}} = \frac{1}{n}\sum_{i=1}^{n} L_{y;i}$. To simplify notation, $i \in L$ is taken to be equivalent to $i \in \{i : X_i \in L_X\}$, i.e. the indices of the observations in leaf L.

Let $c \in \mathbb{R}$ be some cutoff parameter and let $L^a_{xy}(j,c) := \{(X_i, Y_i)|X_{ij} < c, i = 1, ..., n\}, L^b_{xy}(j,c) = \{(X_i, Y_i)|X_{ij} \geq c, i = 1, ..., n\}$ be the two leaves containing the set of observations resulting from partitioning variable j at cutoff c. Then a split is determined by finding the arguments, (j^*, c^*) that minimise the sum of the mean squared errors (MSE) in both leaves (James et al. 2013),

$$(j^*, c^*) = \underset{j,c}{\operatorname{argmin}} \sum_{y \in L^a_y(j,c)} (y - L^a_{\bar{Y}}(j,c))^2 + \sum_{y \in L^b_y(j,c)} (y - L^b_{\bar{Y}}(j,c))^2 \tag{13.1}$$

This method is repeated from the first branch of the tree down to the very last such that observations are included in a given leaf L if they satisfy all conditions from all previous branches; features may be considered multiple times in the growing process. This is an intuitive method as minimising the above sum results in the set of observations within each individual leaf being as similar as possible, thus as an observation is passed down the tree, it becomes more similar to the subsequent leaves, eventually landing in a leaf containing homogeneous observations. Controlling how many variables to consider at each split and how many splits to make are determined by hyper-parameter tuning.

Decision trees are a powerful method for high-dimensional data as only a small sample of variables will be used for growing a tree, and therefore they are also useful for variable importance by identifying which variables were utilised in growth (other importance methods are also available). Decision trees are also highly interpretable, as demonstrated by (Figure 13.1). The recursive pseudo-algorithm in ((**alg-dt-fit?**)) demonstrates the simplicity in growing a decision tree (again methods such as pruning are omitted).

Algorithm 1 Fitting a decision tree.
Input Training data, $\mathcal{D}_{train}$. Splitting rule, SR.
Output Fitted decision tree, $\hat{g}$.

1: Compute (j^*, c^*) as the optimisers of SR (e.g. (@eq-dt-min)) to create the initial leaf and branches.
2: Repeat step 1 on all subsequent branches until a stopping rule is reached.
3: Return the fitted tree, $\hat{g}$, as the series of branches.

Stopping Rules

The 'stopping rule' in ((**alg-dt-fit?**)) is usually a condition on the number of observations in each leaf such that leaves will continue to be split until some minimum number of observations has been reached in a leaf. Other conditions may be on the 'depth' of the tree,

[1] Other methods for growing trees such as pruning are not discussed here as they are less relevant to random forests, which are primarily of interest. Instead see (e.g.) Breiman (1984) [@Breiman1984].

which corresponds to the number of levels of splitting, for example the tree in (Figure 13.1) has a depth of 2 (the first level is not counted).

Random Forests

Despite being more interpretable than other machine learning methods, decision trees usually have poor predictive performance, high variance and are not robust to changes in the data. As such, *random forests* are preferred to improve prediction accuracy and decrease variance. Random forests utilise bootstrap aggregation, or *bagging* (Breiman 1996), to aggregate many decision trees. A pseudo fitting algorithm is given in ((**alg-rsf-fit?**)).

Algorithm 2 Fitting a random forest.
Input Training data, $\mathcal{D}_{train}$. Total number of trees, $B \in \mathbb{N}_{>0}$.
Output Fitted random forest, $\hat{g}$.

1: **for** $b = 1, ..., B$ **do**
2: Create a bootstrapped sample of the data, D_b.
3: Grow a decision tree, $\hat{g}_b$, on D_b with (@alg-dt-fit).
4: **end for**
5: $\hat{g} \leftarrow \{\hat{g}_b\}_{b=1}^{B}$ **return** $\hat{g}$

Prediction from a random forest follows by making predictions from the individual trees and aggregating the results by some function σ ((**alg-rsf-pred?**)); σ is usually the sample mean for regression,

$$\hat{g}(X^*) = \sigma(\hat{g}_1(X^*), ..., \hat{g}_B(X^*)) = \frac{1}{B}\sum_{b=1}^{B} \hat{g}_b(X^*)$$

where $\hat{g}_b(X^*)$ is the terminal node prediction from the bth tree and B are the total number of grown trees (`$B$' is commonly used instead of`N' to note the relation to bootstrapped data).

Algorithm 3 Predicting from a random forest.
Input Testing data $X^* \sim \mathcal{X}$, fitted forest $\hat{g}$ with $B \in \mathbb{N}_{>0}$ trees, aggregation method σ.
Output Prediction, $\hat{Y} \sim \mathcal{Y}$.

1: **for** $b = 1, ..., B$ **do**
2: 'Drop' X^* down the tree $\hat{g}_b$ individually to return a prediction $\hat{g}_b(X^*)$.
3: **end for**
4: $\hat{Y} \leftarrow \sigma(\hat{g}_1(X^*), ..., \hat{g}_B(X^*))$ **return** $\hat{Y}$

Usually many (hundreds or thousands) trees are grown, which makes random forests robust to changes in data and 'confident' about individual predictions. Other advantages include having several tunable hyper-parameters, including: the number of trees to grow, the number of variables to include in a single tree, the splitting rule, and the minimum terminal node size. Machine learning models with many hyper-parameters, tend to perform better than other models as they can be fine-tuned to the data, which is why complex deep learning models are often the best performing. Although as a caveat: too many parameters can lead to over-fitting and tuning many parameters can take a long time and be highly intensive. Random forests lose the interpretability of decision trees and are considered 'black-box' models as individual predictions cannot be easily scrutinised.

13.1.2 Random Forests for Survival Analysis

Given time constraints and the scope of this book, this survey of random forests for survival analysis will primarily focus on 'traditional' decision trees and random forests and will not look at other sub-fields such as causal forests. A comprehensive review of random survival forests (RSFs) is provided in Bou-Hamad (2011) (Bou-Hamad, Larocque, and Ben-Ameur 2011), which includes extensions to time-varying covariates and different censoring types. In order to prevent overlap, this survey will focus primarily on methods that have off-shelf implementations, their prediction types, and how successfully these methods handle the problem of censoring. Random forests and decision trees for survival are termed from here as Random Survival Forests (RSFs) and Survival Decision Trees (SDTs) respectively.

Unlike other machine learning methods that may require complex changes to underlying algorithms, individual components of a random forest can be adapted without altering the fundamental algorithm. The principle random forest algorithm is unchanged for RSFs, the difference is in the choice of splitting rule and terminal node prediction, which both must be able to handle censoring. Therefore instead of discussing individual algorithms, the different choices of splitting rules and terminal node predictions are discussed, then combinations of these are summarised into five distinct algorithms.

13.1.2.1 Splitting Rules

Survival trees and RSFs have been studied for the past four decades and whilst the amount of splitting rules to appear could be considered "numerous" (Bou-Hamad, Larocque, and Ben-Ameur 2011), only two broad classes are commonly utilised and implemented (H. Ishwaran and Kogalur 2018; Pölsterl 2020; Therneau and Atkinson 2019; Wright and Ziegler 2017). The first class rely on hypothesis tests, and primarily the log-rank test, to maximise dissimilarity between splits, the second class utilises likelihood-based measures. The first is discussed in more detail as this is common in practice and is relatively straightforward to implement and understand, moreover it has been demonstrated to outperform other splitting rules (Bou-Hamad, Larocque, and Ben-Ameur 2011). Likelihood rules are more complex and require assumptions that may not be realistic, these are discussed briefly.

Hypothesis Tests

The log-rank test statistic has been widely utilised as the 'natural' splitting-rule for survival analysis (Ciampi et al. 1986; B. H. Ishwaran et al. 2008; LeBlanc and Crowley 1993; Segal 1988). The log-rank test compares the survival distributions of two groups and has the null-hypothesis that both groups have the same underlying risk of (immediate) death, i.e. identical hazard functions.

Let L^A and L^B be two leaves then using the notation above let h^A, h^B be the (true) hazard functions derived from the observations in the two leaves respectively. The log-rank hypothesis test is given by $H_0 : h^A = h^B$ with test statistic (Segal 1988),

$$LR(L^A) = \frac{\sum_{\tau \in \mathcal{U}_D} (d_\tau^A - e_\tau^A)}{\sqrt{\sum_{\tau \in \mathcal{U}_D} v_\tau^A}}$$

where d_τ^A is the observed number of deaths in leaf A at τ,

$$d_\tau^A := \sum_{i \in L^A} \mathbb{I}(T_i = \tau, \Delta_i = 1)$$

e_τ^A is the expected number of deaths in leaf A at τ,

$$e_\tau^A := \frac{n_\tau^A d_\tau}{n_\tau}$$

and v_τ^A is the variance of the number of deaths in leaf A at τ,

$$v_\tau^A := e_\tau^A \Big(\frac{n_\tau - d_\tau}{n_\tau}\Big)\Big(\frac{n_\tau - n_\tau^A}{n_\tau - 1}\Big)$$

where $\mathcal{U}_D$ is the set of unique death times across the data (in both leaves), $\setminus\ n_\tau = \sum_i \mathbb{I}(T_i \geq \tau)$ is the number of observations at risk at τ in both leaves, $\setminus\ n_\tau^A = \sum_{i \in L^A} \mathbb{I}(T_i \geq \tau)$ is the number of observations at risk at τ in leaf A, and $\setminus\ d_\tau = \sum_i \mathbb{I}(T_i = \tau, \Delta_i = 1)$ is the number of deaths at τ in both leaves.

Intuitively these results follow as the number of deaths in a leaf is distributed according to Hyper$(n_\tau^A, n_\tau, d_\tau)$. The same statistic results if L^B is instead considered. ((**alg-dt-fit?**)) follows for fitting decision trees with the log-rank splitting rule, SR, to be maximised.

The higher the log-rank statistic, the greater the dissimilarity between the two groups, thereby making it a sensible splitting rule for survival, moreover it has been shown that it works well for splitting censored data (LeBlanc and Crowley 1993).[2] When censoring is highly dependent on the outcome, the log-rank statistic does not perform well and is biased (Bland and Altman 2004), which tends to be true of the majority of survival models. Additionally, the log-rank test requires no knowledge about the shape of the survival curves or distribution of the outcomes in either group (Bland and Altman 2004), making it ideal for an automated process that requires no user intervention.

The log-rank *score* rule (Hothorn and Lausen 2003) is a standardized version of the log-rank rule that could be considered as a splitting rule, though simulation studies have demonstrated non-significant predictive performance when comparing the two (B. H. Ishwaran et al. 2008).

Alternative dissimiliarity measures and tests have also been suggested as splitting rules, including modified Kolmogorov-Smirnov test and Gehan-Wilcoxon tests (Ciampi et al. 1988). Simulation studies have demonstrated that both of these may have higher power and produce 'better' results than the log-rank statistic (Fleming et al. 1980). Despite this, these do not appear to be in common usage and no implementation could be found that include these.

Likelihood Based Rules {.unnumbered .unlisted} Likelihood ratio statistics, or deviance based splitting rules, assume a certain model form and thereby an assumption about the data. This may be viewed as an advantageous strategy, as it could arguably increase interpretability, or a disadvantage as it places restrictions on the data. For survival models, a full-likelihood can be estimated with a Cox form by estimating the cumulative hazard function (LeBlanc and Crowley 1992). LeBlanc and Crowley (1992) (LeBlanc and Crowley 1992) advocate for selecting the optimal split by maximising the full PH likelihood, assuming the cumulative hazard function, H, is known,

$$\mathcal{L} := \prod_{m=1}^{M} \prod_{i \in L^m} h_m(T_i)^{\Delta_i} \exp(-H_m(T_i))$$

[2]The results of this experiment are actually in LeBlanc's unpublished 1989 PhD thesis and therefore it has to be assumed that LeBlanc is accurately conveying its results in this 1993 paper.

where M is the total number of terminal nodes, h_m and H_m are the (true) hazard and cumulative hazard functions in the mth node, and again L^m is the set of observations in terminal node m. Estimation of h_m and H_m are described with the associated terminal node prediction below.

The primary advantage of this method is that any off-shelf regression software with a likelihood splitting rule can be utilised without any further adaptation to model fitting by supplying this likelihood with required estimates. However the additional costs of computing these estimates may outweigh the benefits once the likelihood has been calculated, and this could be why only one implementation of this method has been found (Bou-Hamad, Larocque, and Ben-Ameur 2011; Therneau and Atkinson 2019).

Other Splitting Rules

As well as likelihood and log-rank spitting rules, other papers have studied comparison of residuals (Therneau, Grambsch, and Fleming 1990), scoring rules (H. Ishwaran and Kogalur 2018), and distance metrics (Gordon and Olshen 1985). These splitting rules work similarly to the mean squared error in the regression setting, in which the score should be minimised across both leaves. The choice of splitting rule is usually data-dependent and can be treated as a hyper-parameter for tuning. However if there is a clear goal in prediction, then the choice of splitting rule can be informed by the prediction type. For example, if the goal is to maximise separation, then a log-rank splitting rule to maximise homogeneity in terminal nodes is a natural starting point. Whereas if the goal is to estimate the linear predictor of a Cox PH model, then a likelihood splitting rule with a Cox form may be more sensible.

13.1.2.2 Terminal Node Prediction

Only two terminal node predictions appear in common usage.

Predict: Ranking

Terminal node ranking predictions for survival trees and forests have been limited to those that use a likelihood-based splitting rule and assume a PH model form (H. Ishwaran et al. 2004; LeBlanc and Crowley 1992). In model fitting the likelihood splitting rule model attempts to fit the (theoretical) PH model $h_m(\tau) = h_0(\tau)\theta_m$ for $m \in 1, ..., M$ where M is the total number of terminal nodes and θ_m is a parameter to estimate. The model returns predictions for $\exp(\hat{\theta}_m)$ where $\hat{\theta}_m$ is the estimate of θ_m. This is estimated via an iterative procedure in which in iteration $j+1$, $\hat{\theta}_m^{j+1}$ is estimated by

$$\hat{\theta}_m^{j+1} = \frac{\sum_{i \in L^m} \Delta_i}{\sum_{i \in L^m} \hat{H}_0^j(T_i)}$$

where as before L^m is the set of observations in leaf m and

$$\hat{H}_0^j(\tau) = \frac{\sum_{i:T_i \leq \tau} \Delta_i}{\sum_{m=1}^{M} \sum_{\{i:i \in \mathcal{R}_\tau \cap L^a\}} \hat{\theta}_m^j}$$

which is repeated until some stopping criterion is reached. The same cumulative hazard is estimated for all nodes however $\hat{\theta}_m$ varies across nodes. This method lends itself naturally to a composition to a full distribution (Chapter 15) as it assumes a PH form and separately estimates the cumulative hazard and relative risk (**?@sec-surv-ml-models-ranfor-nov**), though no implementation of this composition could be found.

Predict: Survival Distribution

The most common terminal node prediction appears to be predicting the survival distribution by estimating the survival function, using the Kaplan-Meier or Nelson-Aalen estimators, on the sample in the terminal node (Hothorn et al. 2004; B. H. Ishwaran et al. 2008; LeBlanc and Crowley 1993; Segal 1988). Estimating a survival function by a non-parametric estimator is a natural choice for terminal node prediction as these are natural 'baselines' in survival, similarly to taking the sample mean in regression. The prediction for SDTs is straightforward, the non-parametric estimator is fit on all observations in each of the terminal nodes. This is adapted to RSFs by bagging the estimator across all decision trees (Hothorn et al. 2004). Using the Nelson-Aalen estimator as an example, let m be a terminal node in an SDT, then the terminal node prediction is given by,

$$\hat{H}_m(\tau) = \sum_{\{i:i\in L^m \cap T_i \leq \tau\}} \frac{d_i}{n_i} \tag{13.2}$$

where d_i and n_i are the number of events and observations at risk at time T_i in terminal node m. Ishwaran (B. H. Ishwaran et al. 2008) defined the bootstrapped Nelson-Aalen estimator as

$$\hat{H}_{Boot}(\tau) = \frac{1}{B}\sum_{b=1}^{B} \hat{H}_{m,b}(\tau), \quad m \in 1, ..., M \tag{13.3}$$

where B is the total number of bootstrapped estimators, M is the number of terminal nodes, and $\hat{H}_{m,b}$ is the cumulative hazard for the mth terminal node in the bth tree. The bootstrapped Kaplan-Meier estimator is calculated analogously. More generally these can be considered as a uniform mixture of B distributions (Chapter 15).

All implemented RSFs can now be summarised into the following five algorithms:

RRT {#mod-rrt}\ LeBlanc and Crowley's (1992) (LeBlanc and Crowley 1992) survival decision tree uses a deviance splitting rule with a terminal node ranking prediction, which assumes a PH model form. These 'relative risk trees' (RRTs) are implemented in the package **rpart** (Therneau and Atkinson 2019). This model is considered the least accessible and transparent of all discussed in this section as: few implementations exist, it requires assumptions that may not be realistic, and predictions are harder to interpret than other models. Predictive performance of the model is expected to be worse than RSFs as this is a decision tree; this is confirmed in (R. E. B. Sonabend 2021).

RRF {#mod-rrf}\ Ishwaran *et al.* (2004) (H. Ishwaran et al. 2004) proposed a random forest framework for the relative risk trees, which makes a slight adaptation and applies the iteration of the terminal node prediction after the tree is grown as opposed to during the growing process. No implementation for these 'relative risk forests' (RRFs) could be found or any usage in the literature.

RSDF-DEV {#mod-rsdfdev}\ Hothorn *et al.* (2004) (Hothorn et al. 2004) expanded upon the RRT by introducing a bagging composition thus creating a random forest with a deviance splitting rule, again assuming a PH form. However the ranking prediction is altered to be a bootstrapped Kaplan-Meier prediction in the terminal node. This is implemented in **ipred** (Peters and Hothorn 2019). This model improves upon the accessibility and transparency of the RRT by providing a more straightforward and interpretable terminal node prediction. However, as this is a decision tree, predictive performance is again expected to be worse than the RSFs.

RSCIFF {#mod-rsciff}\ Hothorn *et al.* (Hothorn et al. 2005) studied a conditional inference framework in order to predict log-survival time. In this case the splitting rule is based

on an IPC weighted loss function, which allows implementation by off-shelf classical random forests. The terminal node predictions are a weighted average of the log-survival times in the node where weighting is determined by the Kaplan-Meier estimate of the censoring distribution. This 'random survival conditional inference framework forest' (RSCIFF) is implemented in **party** (Hothorn, Hornik, and Zeileis 2006) and **partykit** (Hothorn and Zeileis 2015), which additionally includes a distribution terminal node prediction via the bootstrapped Kaplan-Meier estimator. The survival tree analogue (SDCIFT) is implemented in the same packages. Implementation of the RSCIFF is complex, which is likely why all implementations (in the above packages) are by the same authors. The complexity of conditional inference forests may also be the reason why several reviews, including this one, mention (or completely omit) RSCIFFs but do not include any comprehensive details that explain the fitting procedure (Bou-Hamad, Larocque, and Ben-Ameur 2011; H. Wang and Li 2017). In this regard, it is hard to claim that RSCIFFs are transparent or accessible. Moreover the authors of the model state that random conditional inference forests are for "expert user[s] only and [their] current state is rather experimental" (Hothorn and Zeileis 2015). Finally with respect to model performance, there is evidence that they can outperform RSDFs (below) dependent on the data type (Nasejje et al. 2017) however no benchmark experiment could be found that compared them to other models.

RSDF-STAT {#mod-rsdfstat}\ Finally Ishwaran *et al.* (2008) (B. H. Ishwaran et al. 2008) proposed the most general form of RSFs with a choice of hypothesis tests (log-rank and log-rank score) and survival measure (Brier, concordance) splitting rules, and a bootstrapped Nelson-Aalen terminal node prediction. These are implemented in **randomForestSRC** (H. Ishwaran and Kogalur 2018) and **ranger** (Wright and Ziegler 2017). There are several implementations of these models across programming languages, and extensive details for the fitting and predicting procedures, which makes them very accessible. The models utilise a standard random forest framework, which makes them transparent and familiar to those without expert Survival knowledge. Moreover they have been proven to perform well in benchmark experiments, especially on high-dimensional data (Herrmann et al. 2021; Spooner et al. 2020).

13.1.3 Conclusions

Random forests are a highly flexible algorithm that allow the various components to be adapted and altered without major changes to the underlying algorithm. The result is that relatively few R implementations of RSFs cover almost half a century's worth of developments. The only algorithm that does not seem to be implemented is the relative risk forest.

A lack of accessibility, transparency, or proven performance makes RRT and RSDF-DEV a poor choice for model fitting. RSCIFF is potentially a powerful method with promising results in benchmark experiments, but even the authors recognise its complexity prevents it from being accessible. Ishwaran's RSFs on the other hand are more user-friendly and suitable for model fitting and deployment. Simulation studies have demonstrated that RSFs can perform well even with high levels of censoring and there is evidence that on some datasets these can outperform a Cox PH (B. H. Ishwaran et al. 2008). Ishwaran's model is highly flexible, and its implementation in software packages reflects this. Therefore one can still confidently conclude that random forests are a powerful algorithm in regression, classification, and survival analysis.

14

Boosting Methods

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

14.1 Gradient Boosting Machines

14.1.1 Gradient Boosting Machines for Regression

Boosting is a machine learning strategy that can be applied to any model class. Similarly to random forests, boosting is an 'ensemble' method that creates a model from a 'committee' of learners. The committee is formed of 'weak' learners that make poor predictions individually, which creates a 'slow learning' approach (as opposed to 'greedy') that requires many iterations for a model to be a good fit to the data. Boosting models are similar to random forests in that both make predictions from a large committee of learners. However the two differ in how this committee is combined to a prediction. In random forest algorithms, each decision tree is grown independently and their predictions are combined by a simple mean calculation. In contrast, weak learners in a boosting model are fit sequentially and predictions are made by a linear combination of predictions from each learner. With respect to transparency, it is simpler to inspect 100 trees in a random forest, than it is to inspect 100 weak learners in a boosted model, though both are considered black-box models.

The best known boosting algorithm is likely AdaBoost (Freund and Schapire 1996), which is more generally a Forward Stagewise Additive Model (FSAM) with an exponential loss (Hastie, Tibshirani, and Friedman 2001). Today, the most widely used boosting model is the Gradient Boosting Machine (GBM) (J. H. Friedman 2001).

Training a GBM

Pseudo-code for training a componentwise GBM is presented in (7). The term 'componentwise' is explained fully below, only this variation of GBM is presented as it is the most common in implementation (Greenwell et al. 2019; Hothorn et al. 2020). Line 1: the initial function is initialized as $g_0 = 0$;[1] Line 2: iterate over boosting steps $m = 1, ..., M$ and; Line 3:

[1] Some algorithms may instead initialize g_0 by finding the value that minimises the given loss function, however setting $g_0 = 0$ appears to be the most common practice for componentwise GBMs.

randomly sample the training data, $\mathcal{D}_{train}$, to a smaller sample, $\mathcal{D}^*_{train}$, this may be ignored if $\phi = 1$; Line 4: for all training observations in the reduced dataset, $i \in \{i : X_i \in \mathcal{D}^*_{train}\}$, compute the negative gradient, r_{im}, of the differentiable loss function, L, with respect to predictions from the previous iteration, $g_{m-1}(X_i)$; Line 5: fit one weak learner for each feature, $j = 1, ..., p$, in the training data, where the feature, $X_{;j}$, is the single covariate and r_{im} are the labels; Line 6: select the optimal weak learner as the one that minimises the squared error between the prediction and the true gradient; Line 7: update the fitted model by adding the optimal weak learner with a shrinkage penalty, ν; Line 9: return the model updated in the final iteration as the fitted GBM.

Algorithm 4 Training a componentwise Gradient Boosting Machine.

Input Training data, $\mathcal{D}_{train} = \{(X_1, Y_1), ..., (X_n, Y_n)\}$, where $(X_i, Y_i) \overset{i.i.d.}{\sim} (X, Y)$. Differentiable loss, L. Hyper-parameters: sampling fraction, $\phi \in (0, 1]$; step-size, $\nu \in (0, 1]$; number of iterations, $M \in \mathbb{R}_{>0}$.
Output Boosted model, $\hat{g}$.

1: Initialize $g_0 \leftarrow 0$
2: **for** $m = 1, ..., M$ **do**
3: $\quad \mathcal{D}^*_{train} \leftarrow$ Randomly sample $\mathcal{D}_{train}$ w.p. ϕ
4: $\quad r_{im} \leftarrow -[\frac{\partial L(y_i, g_{m-1}(X_i))}{\partial g_{m-1}(X_i)}], i \in \{i : X_i \in \mathcal{D}^*_{train}\}$
5: $\quad$ Fit p weak learners, w_j to $(X_i, r_{im}), j = 1, .., p$
6: $\quad j^* \leftarrow \text{argmin}_{j=1,..,p} \sum_{i \in \{i: X_i \in \mathcal{D}^*_{train}\}} (r_{im} - w_j(X_i))^2$
7: $\quad g_m \leftarrow g_{m-1} + \nu w_{j^*}$
8: **end for**
9: $\hat{g} \leftarrow g_M$ **return** $\hat{g}$

Predicting with a GBM

In general, predictions from a trained GBM are simple to compute as the fitted model (and all individual weak learners) take the same inputs, which are passed sequentially to each of the weak learners. In (7), the fitted GBM is a single model, which is a linear combination of weak learners. Instead one could think of the returned model as a collection of the optimal weak learners, i.e. let $w_{m;j^*}$ be the optimal weak learner from iteration m and let the fitted GBM (Line 9 (7)) be $\hat{g} := \{w_{m;j^*}\}_{m=1}^M$.[2] With this formulation, making predictions from the GBM can be demonstrated simply in ((**alg-surv-gbm-pred?**)).

Algorithm 5 Predicting from a Gradient Boosting Machine.

Input Fitted GBM, $\hat{g} := \{w_{m;j^*}\}_{m=1}^M$, trained with step-size ν. Testing data $X^* \sim \mathcal{X}$.
Output Prediction, $\hat{Y} \sim \mathcal{Y}$.

1: Initialize $\hat{Y} = 0$
2: **for** $m = 1, ..., M$ **do**
3: $\quad \hat{Y} \leftarrow \hat{Y} + \nu w_{m;j^*}(X^*)$
4: **end forreturn** $\hat{Y}$

The biggest advantages of boosting are firstly relatively few hyper-parameters, which all

[2] This formulation is computationally and mathematically identical to the formulation in (@alg-surv-gbm) and is practically more convenient for implementation, indeed this is the implementation in **mboost** [@pkgmboost]. Despite this, the formulation in (@alg-surv-gbm) is common in the literature, which often conflates model training and predicting.

have a meaningful and intuitive interpretation, and secondly its modular nature means that, like random forests, relatively few parts need to be updated to derive a novel model. First the model components will be discussed and then the hyper-parameters. Once this has been established, deriving survival variants can be simply presented.

14.1.1.1 Losses and Learners

Losses

Building a GBM requires selection of the loss to minimise, L, selection of weak learners, w_j, and a method to compare the weak learners to the loss gradient. The only constraint in selecting a loss, L, is that it must be differentiable with respect to $g(X)$ (Hastie, Tibshirani, and Friedman 2001). Of course a sensible loss should be chosen (a classification loss should not be used for regression) and different choices of losses will optimise different tasks. L_2-losses have been demonstrated to be effective for regression boosting, especially with high-dimensional data (Bühlmann and Yu 2003); this is referred to as L_2-boosting.

Weak Learners

(4) is specifically a *componentwise* GBM (Bühlmann and Yu 2003), which means that each of the p weak learners is fit on a single covariate from the data. This method simplifies selecting the possible choices for the weak learners to selecting the class of weak learner (below). Additionally, componentwise GBMs provide a natural and interpretable feature selection method as selecting the optimal learner ((7), line 6) corresponds to selecting the feature that minimises the chosen loss in iteration m.

Only three weak, or 'base', learner classes are commonly used in componentwise GBMs (Hothorn et al. 2020; Z. Wang and Wang 2010). These are linear least squares (J. H. Friedman 2001), smoothing splines (Bühlmann and Yu 2003), and decision stumps (Bühlmann and Yu 2003; J. H. Friedman 2001). Let L be a loss with negative gradient for observation i in the mth iteration, r_{im}, and let $\mathcal{D}_{train}$ be the usual training data. For linear least squares, an individual weak learner is fit by (J. H. Friedman 2001; Z. Wang and Wang 2010),

$$w_j(\mathcal{D}_{train}) = X_{;j}\frac{\sum_{i=1}^{n} X_{ij}r_{im}}{\sum_{i=1}^{n}(X_{ij})^2}$$

For smoothing splines, usually cubic splines are implemented, these fit weak learners as the minimisers of the equation (Bühlmann and Yu 2003),

$$w_j := \underset{g\in\mathcal{G}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(r_{im} - g(X_{ij}))^2 + \lambda\int(g''(u))^2du$$

where g'' is the second derivative of g, $\mathcal{G}$ is the set of functions, $\setminus\ \mathcal{G} := \{g : g$ is twice continuously differentiable and $\int(g''(x))^2dx < \infty\}$, and λ is a hyper-parameter usually chosen so that the number of degrees of freedom, df, is small, with df ≈ 4 suggested (Bühlmann and Yu 2003; Schmid and Hothorn 2008a; Z. Wang and Wang 2010).

Finally for decision stumps ((**?@fig-surv-stump**)), a decision tree, w_j, is grown ((**alg-dt-fit?**)) on $(X_{;j}, r_m)$ to depth one (equivalently to two terminal nodes) for each of the $j = 1, ..., p$ covariates (J. H. Friedman 2001).

14.1.1.2 Hyper-Parameters

The hyper-parameters in (7) are the 'step-size', ν, the sampling fraction, ϕ, and the number of iterations, M.

Number of iterations, M

The number of iterations is often claimed to be the most important hyper-parameter in GBMs and it has been demonstrated that as the number of iterations increases, so too does the model performance (with respect to a given loss on test data) up to a certain point of overfitting (Buhlmann 2006; Hastie, Tibshirani, and Friedman 2001; Schmid and Hothorn 2008a). This is an intuitive result as the foundation of boosting rests on the idea that weak learners can slowly be combined to form a single powerful model. This is especially true in componentwise GBMs as time is required to learn which features are important. Finding the optimal value of M is critical as a value too small will result in poor predictions, whilst a value too large will result in model overfitting. Two primary methods have been suggested for finding the optimal value of M. The first is to find the $M \in \mathbb{N}_{>0}$ that minimises a given measure based on the AIC (Akaike 1974), the second is the 'usual' empirical selection by nested cross-validation. In practice the latter method is usually employed.

Step-size, ν

The step-size parameter ((7), line 7), ν, is a shrinkage parameter that controls the contribution of each weak learner at each iteration. Several studies have demonstrated that GBMs perform better when shrinkage is applied and a value of $\nu = 0.1$ is often suggested (Buhlmann and Hothorn 2007; Hastie, Tibshirani, and Friedman 2001; J. H. Friedman 2001; D. K. K. Lee, Chen, and Ishwaran 2019; Schmid and Hothorn 2008a). The optimal values of ν and M depend on each other, such that smaller values of ν require larger values of M, and vice versa. This is intuitive as smaller ν results in a slower learning algorithm and therefore more iterations are required to fit the model. Accurately selecting the M parameter is generally considered to be of more importance, and therefore a value of ν is often chosen heuristically (e.g. the common value of 0.1) and then M is tuned by cross-validation and/or early-stopping.

Sampling Fraction, ϕ

Motivated by the success of bagging in random forests, stochastic gradient boosting (J. Friedman 1999) randomly samples the data in each iteration. It appears that subsampling performs best when also combined with shrinkage (Hastie, Tibshirani, and Friedman 2001) and as with the other hyper-parameters, selection of ϕ is usually performed by nested cross-validation.

14.1.2 Gradient Boosting Machines for Survival Analysis

In a componentwise GBM framework, adapting boosting to survival analysis requires only selecting a sensible choice of loss function L. Therefore fitting and predicting algorithms for componentwise survival GBMs are not discussed as these are fully described in algorithms (7) and ((**alg-surv-gbm-pred?**)) respectively. However, some GBMs in this section are not componentwise and therefore require some more detailed consideration. Interestingly, unlike other machine learning algorithms that historically ignored survival analysis, early GBM papers considered boosting in a survival context (Ridgeway 1999); though there appears to be a decade gap before further considerations were made in the survival setting. After that period, several developments by Binder, Schmid, and Hothorn, adapted component-

wise GBMs to a framework suitable for survival analysis. Their developments are covered exhaustively in the R packages **gbm** (Greenwell et al. 2019) and **mboost** (Hothorn et al. 2020). This survey continues with the predict type taxonomy.

14.1.2.1 Cox Survival Models

All survival GBMs make ranking predictions and none are able to directly predict survival distributions. However, the GBMs discussed in this section all have natural compositions to distributions as they are modelled in the semi-parametric proportional hazards framework (Chapter 15). The models discussed in the next section can also be composed to distributions though the choice of composition is less clear and therefore they are listed as pure 'ranking' models.

GBM-COX {#mod-gdcox} {#mod-gbmcox}\ The 'GBM-COX' aims to predict the distribution of data following the PH assumption by estimating the coefficients of a Cox model in a boosting framework (Ridgeway 1999). The model attempts to predict $\hat{g}(X^*) = \hat{\eta} := X^*\hat{\beta}$, by minimising a suitable loss function. As the model assumes a PH specification, the natural loss to optimise is the Cox partial likelihood (Cox 1972, 1975), more specifically to minimise the negative partial log-likelihood, $-l$, where

$$l(\beta) = \sum_{i=1}^{n} \Delta_i \Big[\eta_i \; - \; \log\Big(\sum_{j \in \mathcal{R}_{t_i}}^{n} \exp(\eta_i)\Big)\Big] \tag{14.1}$$

where $\mathcal{R}_{t_i}$ is the set of patients at risk at time t_i and $\eta_i = X_i\beta$. The gradient of $-l(\beta)$ at iteration m is

$$r_{im} := \Delta_i - \sum_{j=1}^{n} \Delta_j \frac{\mathbb{I}(T_i \geq T_j)\exp(g_{m-1}(X_i))}{\sum_{k \in \mathcal{R}_{t_j}} \exp(g_{m-1}(X_k))} \tag{14.2}$$

where $g_{m-1}(X_i) = X_i\beta_{m-1}$.

(5) now follows with the loss $L := -l(\beta)$.[3]

The GBM-COX is implemented in **mboost** (Hothorn et al. 2020) and has been demonstrated to perform well even when the data violates the PH assumption (Johnson and Long 2011). Despite being a black-box, GBMs are well-understood and individual weak learners are highly interpretable, thus making GBMs highly transparent. Several well-established software packages implement GBM-COX and those that do not tend to be very flexible with respect to custom implementations.

CoxBoost {#mod-coxboost}\ The CoxBoost algorithm boosts the Cox PH by optimising the penalized partial-log likelihood; additionally the algorithm allows for mandatory (or 'forced') covariates (Harald Binder and Schumacher 2008). In medical domains the inclusion of mandatory covariates may be essential, either for model interpretability, or due to prior expert knowledge. This is not a feature usually supported by boosting. CoxBoost deviates from (7) by instead using an offset-based approach for generalized linear models (Tutz and Binder 2007). This model has a non-componentwise and componentwise framework but only the latter is implemented by the authors (Harold Binder 2013) and discussed

[3] Early implementations and publications of the GBM algorithm [@Friedman1999; @Friedman2001] included an additional step to the algorithm in which a step size is estimated by line search. More recent research has determined that this additional step is unneccesary [@Buhlmann2007] and the line search method does not appear to be used in practice.

here. Let $\mathcal{J}_{mand}$ be the indices of the mandatory covariates to be included in all iterations, $m = 1, ..., M$, then for an iteration m the indices to consider for fitting are the set

$$I_m = \{\{1\} \cup \mathcal{J}_{mand}, ..., \{p\} \cup \mathcal{J}_{mand}\}/\{\{j\} \cup \mathcal{J}_{mand} : j \in \mathcal{J}_{mand}\}$$

i.e. in each iteration the algorithm fits a weak learner on the mandatory covariates and one additional (non-mandatory) covariate (hence still being componentwise).

In addition, a penalty matrix $\mathbf{P} \in \mathbb{R}^{p \times p}$ is considered such that $P_{ii} > 0$ implies that covariate i is penalized and $P_{ii} = 0$ means no penalization. In practice this is usually a diagonal matrix (Harald Binder and Schumacher 2008) and by setting $P_{ii} = 0, i \in I_{mand}$ and $P_{ii} > 0, i \notin I_{mand}$, only optional (non-mandatory) covariates are penalized. The penalty matrix can be allowed to vary with each iteration, which allows for a highly flexible approach, however in implementation a simpler approach is to either select a single penalty to be applied in each iteration step or to have a single penalty matrix (Harold Binder 2013).

At the mth iteration and the kth set of indices to consider ($k = 1, ..., p$), the loss to optimize is the penalized partial-log likelihood given by

$$l_{pen}(\gamma_{mk}) = \sum_{i=1}^{n} \Delta_i \Big[\eta_{i,m-1} + X_{i,\mathcal{J}_{mk}} \gamma_{mk}^T\Big] -$$
$$\Delta_i \log\Big(\sum_{j=1}^{n} \mathbb{I}(T_j \leq T_i) \exp(\eta_{i,m-1} + X_{i,\mathcal{J}_{mk}} \gamma_{mk}^T\Big) - \lambda \gamma_{mk} \mathbf{P}_{mk} \gamma_{mk}^T$$

where $\eta_{i,m} = X_i \beta_m$, γ_{mk} are the coefficients corresponding to the covariates in $\mathcal{J}_{mk}$ which is the possible set of candidates for a subset of total candidates $k = 1, ..., p$, $\mathbf{P}_{mk}$ is the penalty matrix, and λ is a penalty hyper-parameter to be tuned or selected.[4]

In each iteration, all potential candidate sets (the union of mandatory covariates and one other covariate) are updated by

$$\hat{\gamma}_{mk} = \mathbf{I}_{pen}^{-1}(0) U(0)$$

where $U(\gamma) = \partial l / \partial \gamma(\gamma)$ and $\mathbf{I}_{pen}^{-1} = \partial^2 l / \partial \gamma \partial \gamma^T (\gamma + \lambda \mathbf{P}_{mk})$ are the first and second derivatives of the unpenalized partial-log-likelihood. The optimal set is then found as

$$k^* := \underset{k}{\operatorname{argmax}}\, l_{pen}(\gamma_{mk})$$

and the estimated coefficients are updated with

$$\hat{\beta}_m = \hat{\beta}_{m-1} + \gamma_{mk^*}, \quad k^* \in \mathcal{J}_{mk}$$

The step size, ν, is then one, but this could potentially be altered.

The algorithm deviates from (7) as l_{pen} is directly optimised and not its gradient, additionally model coefficients are iteratively updated instead of a more general model form. The algorithm is implemented in **CoxBoost** (Harold Binder 2013). Experiments suggest that including the 'correct' mandatory covariates may increase predictive performance (Harald Binder and Schumacher 2008). CoxBoost is less accessible than other boosting methods as it requires a unique boosting algorithm, as such only one off-shelf implementation appears

[4] On notation, note that $\mathbf{P}_{ij}$ refers to the penalty matrix in the ith iteration for the jth set of indices, whereas P_{ij} is the (i, j)th element in the matrix $\mathbf{P}$.

to exist and even this implementation has been removed from CRAN as of 2020-11-11. CoxBoost is also less transparent as the underlying algorithm is more complex, though is well-explained by the authors (Harald Binder and Schumacher 2008). There is good indication that CoxBoost is performant (R. E. B. Sonabend 2021). In a non-medical domain, where performance may be the most important metric, then perhaps CoxBoost can be recommended as a powerful model. However, when sensitive predictions are required, CoxBoost may not be recommended. Further papers studying the model and more off-shelf implementations could change this in the future.

14.1.2.2 Ranking Survival Models

The ranking survival models in this section are all unified as they make predictions of the linear predictor, $\hat{g}(X^*) = X^*\hat{\beta}$.[5]

GBM-AFT {#mod-gbmaft}\ Schmid and Hothorn (2008) (Schmid and Hothorn 2008b) published a GBM for accelerated failure time models in response to PH-boosted models that may not be suitable for non-PH data. Their model fits into the GBM framework by assuming a fully-parametric AFT and simultaneously estimating the linear predictor, $\hat{g}(X_i) = \hat{\eta}$, and the scale parameter, $\hat{\sigma}$, controlling the amount of noise in the distribution. The (fully-parametric) AFT is defined by

$$\log Y = \eta + \sigma W$$

where W is a random variable independent of the covariates that follows a given distribution and controls the noise in the model. By assuming a distribution on W, a distribution is assumed for the full parametric model. The full likelihood, $\mathcal{L}$, is given by

$$\mathcal{L}(\mathcal{D}_{train}|\mu, \sigma, W) = \prod_{i=1}^{n} \Big[\frac{1}{\sigma} f_W\Big(\frac{\log(T_i) - \mu}{\sigma}\Big)\Big]^{\Delta_i} \Big[S_W\Big(\frac{\log(T_i) - \mu}{\sigma}\Big)\Big]^{(1-\Delta_i)} \tag{14.3}$$

where f_W, S_W is the pdf and survival function of W for a given distribution. By setting $\mu := g(X_i)$, σ is then rescaled according to known results depending on the distribution (Klein and Moeschberger 2003). The gradient of the negative log-likelihood, $-l$, is minimised in the mth iteration where

$$l(\mathcal{D}_{train}|\hat{g}, \hat{\sigma}, W) = \sum_{i=1}^{n} \Delta_i \Big[-\log\sigma + \log f_W\Big(\frac{\log(T_i) - \hat{g}_{m-1}(X_i)}{\hat{\sigma}_{m-1}}\Big)\Big] + (1 - \Delta_i)\Big[\log S_W\Big(\frac{\log(T_i) - \hat{g}_{m-1}(X_i)}{\hat{\sigma}_{m-1}}\Big)\Big]$$

where $\hat{g}_{m-1}, \hat{\sigma}_{m-1}$ are the location-scale parameters estimated in the previous iteration. Note this key difference to other GBM methods in which two estimates are made in each iteration step. In order to allow for this, (7) is run as normal but in addition, after updating $\hat{g}_m$, one then updates $\hat{\sigma}_m$ as

$$\hat{\sigma}_m := \underset{\sigma}{\operatorname{argmin}} -l(\mathcal{D}_{train}|g_m, \sigma, W)$$

σ_0 is initialized at the start of the algorithm with $\sigma_0 = 1$ suggested (Schmid and Hothorn 2008b).

[5]This is commonly referred to as a 'linear predictor' as it directly relates to the boosted linear model (e.g. Cox PH), however it is more accurately a 'prognostic index' as the final prediction is not the true linear predictor.

This algorithm provides a ranking prediction without enforcing an often-unrealistic PH assumption on the data. This model is implemented in **mboost** and **xgboost**. Experiments indicate that this may outperform the Cox PH (Schmid and Hothorn 2008b). Moreover the model has the same transparency and accessibility as the GBM-COX.

GBM-GEH {#mod-gbmgeh}\ The concordance index is likely the most popular measure of discrimination, this in part due to the fact that it makes little-to-no assumptions about the data (Chapter 6). A less common measure is the Gehan loss, motivated by the semi-parametric AFT. Johnson and Long proposed the GBM with Gehan loss, here termed GBM-GEH, to optimise separation within an AFT framework (Johnson and Long 2011).

The semi-parametric AFT is defined by the linear model,

$$\log Y = \eta + \epsilon$$

for some error term, ϵ.

The D-dimensional Gehan loss to minimise is given by,

$$G_D(\mathcal{D}_{train}, \hat{g}) = -\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\Delta_i(\hat{e}_i - \hat{e}_j)\mathbb{I}(\hat{e}_i \leq \hat{e}_j)$$

where $\hat{e}_i = \log T_i - \hat{g}(X_i)$. The negative gradient of the loss is,

$$r_{im} := \frac{\sum_{j=1}^{n}\Delta_j\mathbb{I}(\hat{e}_{m-1,i} \geq \hat{e}_{m-1,j}) - \Delta_i\mathbb{I}(\hat{e}_{m-1,i} \leq \hat{e}_{m-1,j})}{n}$$

where $\hat{e}_{m-1,i} = \log T_i - \hat{g}_{m-1}(X_i)$.

(6) then follows naturally substituting the loss and gradient above. The algorithm is implemented in **mboost**. Simulation studies on the performance of the model are inconclusive (Johnson and Long 2011) however the results in (R. E. B. Sonabend 2021) indicate strong predictive performance.

GBM-BUJAR {#mod-gbmbujar}\ GBM-BUJAR is another boosted semi-parametric AFT. However the algorithm introduced by Wang and Wang (2010) (Z. Wang and Wang 2010) uses Buckley-James imputation and minimisation. This algorithm is almost identical to a regression GBM (i.e. using squared loss or similar for L), except with one additional step to iteratively impute censored survival times. Assuming a semi-parametric AFT model, the GBM-BUJAR algorithm iteratively updates imputed outcomes with the Buckley-James estimator (Buckley and James 1979),

$$T^*_{m,i} := \hat{g}_{m-1}(X_i) + e_{m-1,i}\Delta_i + (1-\Delta_i)\Big[\hat{S}_{KM}(e_{m-1,i})^{-1}\sum_{e_{m-1,j}>e_{m-1,i}} e_{m-1,j}\Delta_j\hat{p}_{KM}(e_{m-1,j})\Big]$$

where $\hat{g}_{m-1}(X_i) = \hat{\eta}_{m-1}$, and $\hat{S}_{KM}, \hat{p}_{KM}$ are Kaplan-Meier estimates of the survival and probability mass functions respectively fit on some training data, and $e_{m-1,i} := \log(T_i) - g_{m-1}(X_i)$. Once $T^*_{m,i}$ has been updated, (7) continues from with least squares as with any regression model.

GBM-BUJAR is implemented in **bujar** (Z. Wang 2019) though without a separated fit/predict interface, its accessibility is therefore limited. There is no evidence of wide usage of this algorithm nor simulation studies demonstrating its predictive ability. Finally,

there are many known problems with semi-parametric AFT models and the Buckey-James procedure (Wei 1992), hence GBM-BUJAR is also not transparent.

GBM-UNO {#mod-gbmuno}\ Instead of optimising models based on a given model form, Chen *et al.* (Y. Chen et al. 2013) studied direct optimisation of discrimination by Harrell's C whereas Mayr and Schmid (Mayr and Schmid 2014) focused instead on Uno's C. Only an implementation of the Uno's C method could be found, this is therefore discussed here and termed 'GBM-UNO'.

The GBM-UNO attempts to predict $\hat{g}(X^*) := \hat{\eta}$ by optimising Uno's C (Section 6.1),

$$C_U(\hat{g}, \mathcal{D}_{train}) = \frac{\sum_{i \neq j} \Delta_i \{\hat{G}_{KM}(T_i)\}^{-2} \mathbb{I}(T_i < T_j) \mathbb{I}(\hat{g}(X_i) > \hat{g}(X_j))}{\sum_{i \neq j} \Delta_i \{\hat{G}_{KM}(T_i)\}^{-2} \mathbb{I}(T_i < T_j)}$$

The GBM algorithm requires that the chosen loss, here C_U, be differentiable with respect to $\hat{g}(X)$, which is not the case here due to the indicator term, $\mathbb{I}(\hat{g}(X_i) > \hat{g}(X_j))$. Therefore a smoothed version is instead considered where the indicator is approximated by the sigmoid function (Ma and Huang 2006),

$$K(u|\sigma) = (1 + \exp(-u/\sigma))^{-1}$$

where σ is a hyper-parameter controlling the smoothness of the approximation. The measure to optimise is then,

$$C_{USmooth}(\mathcal{D}_{train}|\sigma) = \sum_{i \neq j} \frac{k_{ij}}{1 + \exp\left[(\hat{g}(X_j) - \hat{g}(X_i))/\sigma\right]} \tag{14.4}$$

with

$$k_{ij} = \frac{\Delta_i (\hat{G}_{KM}(T_i))^{-2} \mathbb{I}(T_i < T_j)}{\sum_{i \neq j}^{n} \Delta_i (\hat{G}_{KM}(T_i))^{-2} \mathbb{I}(T_i < T_j)}$$

The negative gradient at iteration m for observation i can then be found,

$$r_{im} := -\sum_{j=1}^{n} k_{ij} \frac{-\exp(\frac{\hat{g}_{m-1}(X_j) - \hat{g}_{m-1}(X_i)}{\sigma})}{\sigma(1 + \exp(\frac{\hat{g}_{m-1}(X_j) - \hat{g}_{m-1}(X_i)}{\sigma}))} \tag{14.5}$$

(7) can then be followed exactly by substituting this loss and gradient; this is implemented in **mboost**. One disadvantage of GBM-UNO is that C-index boosting is more insensitive to overfitting than other methods (Mayr, Hofner, and Schmid 2016), therefore stability selection (Meinshausen and Bühlmann 2010) can be considered for variable selection; this is possible with **mboost**. Despite directly optimising discrimination, simulation studies do not indicate that this model has better separation than other boosted or lasso models (Mayr and Schmid 2014). GBM-UNO has the same accessibility, transparency, and performance (R. E. B. Sonabend 2021) as previous boosting models.

14.1.3 Conclusions

Componentwise gradient boosting machines are a highly flexible and powerful machine learning tool. They have proven particularly useful in survival analysis as minimal adjustments are required to make use of off-shelf software. The flexibility of the algorithm allows all the models above to be implemented in very few R (and other programming languages) packages.

Boosting is a method that often relies on intensive computing power and therefore dedicated packages, such as **xgboost** (T. Chen et al. 2020), exist to push CPU/GPUs to their limits in order to optimise predictive performance. This can be viewed as a strong advantage though one should be careful not to focus too much on predictive performance to the detriment of accessibility and transparency.

Boosting, especially with tree learners, is viewed as a black-box model that is increasingly difficult to interpret as the number of iterations increase. However, there are several methods for increasing interpretability, such as variable importance and SHAPs (Lundberg and Lee 2017). There is also evidence that boosting models can outperform the Cox PH (Schmid and Hothorn 2008b) (not something all ML models can claim).

Part IV

Reduction Techniques

15

Reductions

TODO (150-200 WORDS)

> ! Page in progress!
>
> **This page is a work in progress and will change significantly over time.**

In this chapter, composition and reduction are formally introduced, defined and demonstrated within survival analysis. Neither of these are novel concepts in general or in survival, with several applications already seen earlier when reviewing models (particularly in neural networks), however a lack of formalisation has led to much repeated work and at times questionable applications (**?@sec-surv-ml-models-nn**). The primary purpose of this chapter is to formalise composition and reduction for survival and to unify references and strategies for future use. These strategies are introduced in the context of minimal 'workflows' and graphical 'pipelines' in order to maximise their generalisability. The pipelines discussed in this chapter are implemented in `mlr3proba`.

A *workflow* is a generic term given to a series of sequential operations. For example a standard ML workflow is fit/predict/evaluate, which means a model is fit, predictions are made, and these are evaluated. In this book, a *pipeline* is the name given to a concrete workflow. Section 15.1 demonstrates how pipelines are represented in this book.

Composition (Section 15.2) is a general process in which an object is built (or composed) from other objects and parameters. Reduction (Section 15.3) is a closely related concept that utilises composition in order to transform one problem into another. Concrete strategies for composition and reduction are detailed in sections Section 15.4 and Section 15.5.

Notation and Terminology

The notation introduced in Chapter 4 is recapped for use in this chapter: the generative survival template for the survival setting is given by (X, T, Δ, Y, C) *t.v.i.* $\mathcal{X} \times \mathcal{T} \times \{0,1\} \times \mathcal{T} \times \mathcal{T}$ where $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$, where C, Y are unobservable, $T := \min\{Y, C\}$, and $\Delta = \mathbb{I}(Y = T)$. Random survival data is given by $(X_i, T_i, \Delta_i, Y_i, C_i) \overset{i.i.d.}{\sim} (X, T, \Delta, Y, C)$. Usually data will instead be presented as a training dataset, $\mathcal{D}_{train} = \{(X_1, T_1, \Delta_1), ..., (X_n, T_n, \Delta_n)\}$ where $(X_i, T_i, \Delta_i) \overset{i.i.d.}{\sim} (X, T, \Delta)$, and some test data $\mathcal{D}_{test} = (X^*, T^*, \Delta^*) \sim (X, T, \Delta)$.

For regression models the generative template is given by (X, Y) t.v.i. $\mathcal{X} \subseteq \mathbb{R}^p$ and $Y \subseteq \mathbb{R}$. As with the survival setting, a regression training set is given by $\{(X_1, Y_1), ..., (X_n, Y_n)\}$ where $(X_i, Y_i) \overset{i.i.d.}{\sim} (X, Y)$ and some test data $(X^*, Y^*) \sim (X, Y)$.

15.1 Representing Pipelines

Before introducing concrete composition and reduction algorithms, this section briefly demonstrates how these pipelines will be represented in this book.

Pipelines are represented by graphs designed in the following way: all are drawn with operations progressing sequentially from left to right; graphs are comprised of nodes (or 'vertices') and arrows (or 'directed edges'); a rounded rectangular node represents a process such as a function or model fitting/predicting; a (regular) rectangular node represents objects such as data or hyper-parameters. Output from rounded nodes are sometimes explicitly drawn but when omitted the output from the node is the input to the next.

These features are demonstrated in **?@fig-car-example**. Say $y = 2$ and $a = 2$, then: data is provided ($y = 2$) and passed to the shift function ($f(x) = x+2$), the output of this function ($y = 4$) is passed directly to the next ($h(x|a) = x^a$), this function requires a parameter which is also input ($a = 2$), finally the resulting output is returned ($y^* = 16$). Programmatically, $a = 2$ would be a hyper-parameter that is stored and passed to the required function when the function is called.

This pipeline is represented as a pseudo-algorithm in (**alg-car-ex?**), though of course is overly complicated and in practice one would just code $(y + 2)^\wedge a$.

Algorithm 6 Example pipeline.
Input Data, $y \in \mathbb{R}$. Parameter, $a \in \mathbb{R}$.
Output Transformed data, $x \in \mathbb{R}$.

$x \leftarrow y$
$x \leftarrow x + 2$
$x \leftarrow x^\wedge a$ **return** x

15.2 Introduction to Composition

This section introduces composition, defines a taxonomy for describing compositors (Section 15.2.1), and provides some motivating examples of composition in survival analysis (Section 15.2.2).

In the simplest definition, a model (be it mathematical, computational, machine learning, etc.) is called a *composite model* if it is built of two or more constituent parts. This can be simplest defined in terms of objects. Just as objects in the real-world can be combined in some way, so can mathematical objects. The exact 'combining' process (or 'compositor') depends on the specific composition, so too do the inputs and outputs. By example, a wooden table can be thought of as a composite object (Figure 15.1). The inputs are wood and nails, the combining process is hammering (assuming the wood is pre-chopped), and the output is a surface for eating. In mathematics, this process is mirrored. Take the example of a shifted linear regression model. This is defined by a linear regression model, $f(x) = \beta_0 + x\beta_1$, a shifting parameter, α, and a compositor $g(x|\alpha) = f(x)+\alpha$. Mathematically this example is overly trivial as this could be directly modelled with $f(x) = \alpha+\beta_0+x\beta_1$, but algorithmically there is a difference. The composite model g, is defined by first fitting the linear regression

model, f, and then applying a shift, α; as opposed to fitting a directly shifted model.

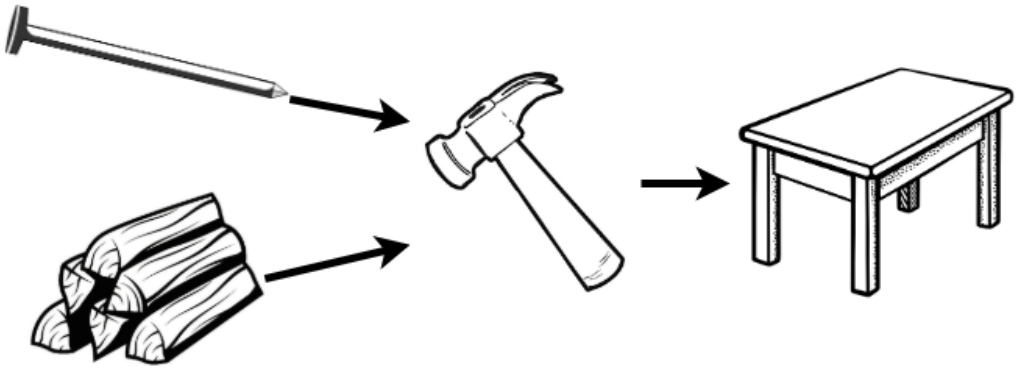


Figure 15.1: Visualising composition in the real-world. A table is a composite object built from nails and wood, which are combined with a hammer 'compositor'. Figure not to scale.

Why Composition?

Tables tend to be better surfaces for eating your dinner than bundles of wood. Or in modelling terms, it is well-known that ensemble methods (e.g. random forests) will generally outperform their components (e.g. decision trees). All ensemble methods are composite models and this demonstrates one of the key use-cases of composition: improved predictive performance. The second key use-case is reduction, which is fully discussed in Section 15.3. Section 15.2.2 motivates composition in survival analysis by demonstrating how it is already prevalent but requires formalisation to make compositions more transparent and accessible.

Composite Model vs. Sub-models

A bundle of wood and nails is not a table and $1,000$ decision trees are not a random forest, both require a compositor. The compositor in a composite model combines the components into a single model. Considering a composite model as a single model enables the hyperparameters of the compositor and the component model(s) to be efficiently tuned whilst being evaluated as a single model. This further allows the composite to be compared to other models, including its own components, which is required to justify complexity instead of parsimony in model building (**?@sec-eval-why-why**).

15.2.1 Taxonomy of Compositors

Just as there are an infinite number of ways to make a table, composition can come in infinite forms. However there are relatively few categories that these can be grouped into. Two primary taxonomies are identified here. The first is the 'composition type' and relates to the number of objects composed:

[i)] i. Single-Object Composition (SOC) – This form of composition either makes use of parameters or a transformation to alter a single object. The shifted linear regression model above is one example of this, another is given in Section 15.4.3. i. Multi-Object Composition

(MOC) – In contrast, this form of composition combines multiple objects into a single one. Both examples in Section 15.2.2 are multi-object compositions.

The second grouping is the 'composition level' and determines at what 'level' the composition takes place:

[i)] i. Prediction Composition – This applies at the level of predictions; the component models could be forgotten at this point. Predictions may be combined from multiple models (MOC) or transformed from a single model (SOC). Both examples in Section 15.2.2 are prediction compositions. i. Task Composition – This occurs when one task (e.g. regression) is transformed to one or more others (e.g. classification), therefore always SOC. This is seen mainly in the context of reduction (Section 15.3). i. Model Composition – This is commonly seen in the context of wrappers (Section 15.5.1.4), in which one model is contained within another. i. Data Composition – This is transformation of training/testing data types, which occurs at the first stage of every pipeline.

15.2.2 Motivation for Composition

Two examples are provided below to demonstrate common uses of composition in survival analysis and to motivate the compositions introduced in Section 15.4.

Example 1: Cox Proportional Hazards

Common implementations of well-known models can themselves be viewed as composite models, the Cox PH is the most prominent example in survival analysis. Recall the model defined by

$$h(\tau|X_i) = h_0(\tau)\exp(\beta X_i)$$

where h_0 is the baseline hazard and β are the model coefficients.

This can be seen as a composite model as Cox defines the model in two stages (Cox 1972): first fitting the β-coefficients using the partial likelihood and then by suggesting an estimate for the baseline distribution. This first stage produces a linear predictor return type (Section 4.3) and the second stage returns a survival distribution prediction. Therefore the Cox model for linear predictions is a single (non-composite) model, however when used to make distribution predictions then it is a composite. Cox implicitly describes the model as a composite by writing ''alternative simpler procedures would be worth having" (Cox 1972), which implies a decision in fitting (a key feature of composition). This composition is formalised in Section 15.4.1 as a general pipeline (C1). The Cox model utilises the (C1) pipeline with a PH form and Kaplan-Meier baseline.

Example 2: Random Survival Forests

Fully discussed in Section 13.1, random survival forests are composed from many individual decision trees via a prediction composition algorithm ((**alg-rsf-pred?**)). In general, random forests perform better than their component decision trees, which tends to be true of all ensemble methods. Aggregation of predictions in survival analysis requires slightly more care than other fields due to the multiple prediction types, however this is still possible and is formalised in Section 15.4.4.

15.3 Introduction to Reduction

This section introduces reduction, motivates its use in survival analysis (Section 15.3.1), details an abstract reduction pipeline and defines the difference between a complete/incomplete reduction (Section 15.3.2), and outlines some common mistakes that have been observed in the literature when applying reduction (Section 15.3.3).

Reduction is a concept found across disciplines with varying definitions. This report uses the Langford definition: reduction is ''a complex problem decomposed into simpler subproblems so that a solution to the subproblems gives a solution to the complex problem" (Langford et al. 2016). Generalisation (or induction) is a common real-world use of reduction, for example sampling a subset of a population in order to estimate population-level results. The true answer (population-level values) may not always be found in this way but very good approximations can be made with simpler sub-problems (sub-sampling).

Reductions are workflows that utilise composition. By including hyper-parameters, even complex reduction strategies can remain relatively flexible. To illustrate reduction by example, recall the table-building example (Section 15.2) in which the task of interest is to acquire a table. The most direct but complex solution is to fell a tree and directly saw it into a table (Figure 15.2, top), clearly this is not a sensible process. Instead the problem can be reduced into simpler sub-problems: saw the tree into bundles of wood, acquire nails, and then use the 'hammer compositor' (Figure 15.1) to create a table (Figure 15.2, bottom).

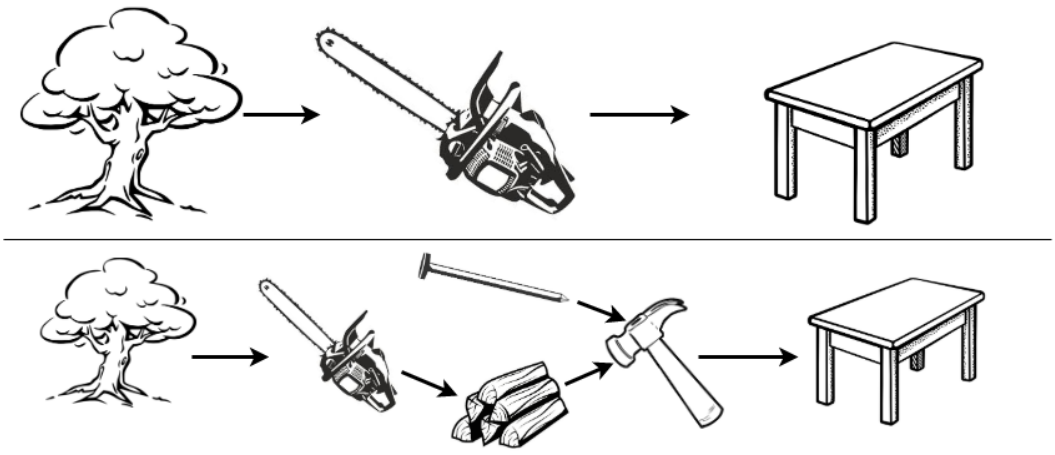


Figure 15.2: Visualising reduction in the real-world. The complex process (top) of directly sawing a tree into a table is inefficient and unnecessarily complex. The reduction (bottom) that involves first creating bundles of wood is simpler, more efficient, and yields the same result, though technically requiring more steps.

In a modelling example, predicting a survival distribution with the Cox model can be viewed as a reduction in which two sub-problems are solved and composed:

i. predict continuous ranking;
ii. estimate baseline hazard; and

iii. compose with (C1) (Section 15.4.1).

This is visualised as a reduction strategy in **?@fig-car-cargraph**. The entire process from defining the original problem, to combining the simpler sub-solutions (in green), is the reduction (in red).

15.3.1 Reduction Motivation

Formalisation of reduction positively impacts upon accessibility, transparency, and predictive performance. Improvements to predictive performance have already been demonstrated when comparing random forests to decision trees. In addition, a reduction with multiple stages and many hyper-parameters allows for fine tuning for improved transparency and model performance (usual overfitting caveat applies, as does the trade-off described in **?@sec-car-pipelines-trade**).

The survey of ANNs (**?@sec-surv-ml-models-nn**) demonstrated how reduction is currently utilised without transparency. Many of these ANNs are implicitly reductions to probabilistic classification (Section 15.5.1.6) however none include details about how the reduction is performed. Furthermore in implementation, none provide interface points to the reduction hyper-parameters. Formalisation encourages consistent terminology, methodology and transparent implementation, which can only improve model performance by exposing further hyper-parameters.

Accessibility is improved by formalising specific reduction workflows that previously demanded expert knowledge in deriving, building, and running these pipelines. All regression reductions in this chapter, are implemented in \ `mlr3proba` (R. Sonabend et al. 2021) and can be utilised with any possible survival model.

Finally there is an economic and efficiency advantage to reduction. A reduction model is relatively 'cheap' to explore as they utilise pre-established models and components to solve a new problem. Therefore if a certain degree of predictive ability can be demonstrated from reduction models, it may not be worth the expense of pursuing more novel ideas and hence reduction can help direct future research.

15.3.2 Task, Loss, and Data Reduction

Reduction can be categorised into task, loss, and data reduction, often these must be used in conjunction with each other. The direction of the reductions may be one- or two-way; this is visualised in **?@fig-car-reduxdiag**. This diagram should not be viewed as a strict fit/predict/evaluation workflow but instead as a guidance for which tasks, T, data, D, models, M, and losses, L, are required for each other. The subscript O refers to the original object 'level' before reduction, whereas the subscript R is in reference to the reduced object.

The individual task, model, and data compositions in the diagram are listed below, the reduction from survival to classification (Section 15.5.1) is utilised as a running example to help exposition.

- $T_O \rightarrow T_R$: By definition of a machine learning reduction, task reduction will always be one way. A more complex task, T_O, is reduced to a simpler one, T_R, for solving. T_R could also be multiple simpler tasks. For example, solving a survival task, T_O, by classification, T_R (Section 15.5.1).
- $T_R \rightarrow M_R$: All machine learning tasks have models that are designed to solve them. For example logistic regression, M_R, for classification tasks, T_R.

- $M_R \to M_O$: The simpler models, M_R, are used for the express purpose to solve the original task, T_O, via solving the simpler ones. To solve T_O, a compositor must be applied, which may transform one (SOC) or multiple models (MOC) at a model- or prediction-level, thus creating M_O. For example predicting survival probabilities with logistic regression, M_R, at times $1, ..., \tau^*$ for some $\tau^* \in \mathbb{N}_{>0}$ (Section 15.5.1.4).
- $M_O \to T_O$: The original task should be solvable by the composite model. For example predicting a discrete survival distribution by concatenating probabilistic predictions at the times $1, ..., \tau^*$ (Section 15.5.1.6).
- $D_O \to D_R$: Just as the tasks and models are reduced, the data required to fit these must likewise be reduced. Similarly to task reduction, data reduction can usually only take place in one direction, to see why this is the case take an example of data reduction by summaries. If presented with 10 data-points $\{1, 1, 1, 5, 7, 3, 5, 4, 3, 3\}$ then these could be reduced to a single point by calculating the sample mean, 3.3. Clearly given only the number 3.3 there is no strategy to recover the original data. There are very few (if any) data reduction strategies that allow recovery of the original data. Continuing the running example, survival data, D_O, can be binned (Section 15.5.1.1) to classification data, D_R.

There is no arrow between D_O and M_O as the composite model is never fit directly, only via composition from $M_R \to M_O$. However, the original data, D_O, is required when evaluating the composite model against the respective loss, L_O.[1] Reduction should be directly comparable to non-reduction models, hence this diagram does not include loss reduction and instead insists that all models are compared against the same loss L_O.

A reduction is said to be *complete* if there is a full pipeline from $T_O \to M_O$ and the original task is solved, otherwise it is *incomplete*. The simplest complete reduction is comprised of the pipeline $T_O \to T_R \to M_R \to M_O$. Usually this is not sufficient on its own as the reduced models are fit on the reduced data, $D_R \to M_R$.

A complete reduction can be specified by detailing:

i. the original task and the sub-task(s) to be solved, $T_O \to T_R$;
ii. the original dataset and the transformation to the reduced one, $D_O \to D_R$ (if required); and
iii. the composition from the simpler model to the complex one, $M_R \to M_O$.

15.3.3 Common Mistakes in Implementation of Reduction

In surveying models and measures, several common mistakes in the implementation of reduction and composition were found to be particularly prevalent and problematic throughout the literature. It is assumed that these are indeed mistakes (not deliberate) and result from a lack of prior formalisation. These mistakes were even identified 20 years ago (Schwarzer, Vach, and Schumacher 2000) but are provided in more detail in order to highlight their current prevalence and why they cannot be ignored.

RM1. Incomplete reduction. This occurs when a reduction workflow is presented as if it solves the original task but fails to do so and only the reduction strategy is solved. A common example is claiming to solve the survival task by using binary classification, e.g. erroneously claiming that a model predicts survival probabilities (which implies distribution) when it actually predicts a five year probability of death ((**box-task-classif?**)). This is a

[1] A complete diagram would indicate that D_O is split into training data, which is subsequently reduced, and test data, which is passed to L_O. All reductions in this section can be applied to any data splitting process.

mistake as it misleads readers into believing that the model solves a survival task ((**box-task-surv?**)) when it does not. This is usually a semantic not mathematical error and results from misuse of terminology. It is important to be clear about model predict types (Section 4.3) and general terms such as 'survival predictions' should be avoided unless they refer to one of the three prediction tasks. RM2. Inappropriate comparisons. This is a direct consequence of (RM1) and the two are often seen together. (RM2) occurs when an incomplete reduction is directly compared to a survival model (or complete reduction model) using a measure appropriate for the reduction. This may lead to a reduction model appearing erroneously superior. For example, comparing a logistic regression to a random survival forest (RSF) (Section 13.1) for predicting survival probabilities at a single time using the accuracy measure is an unfair comparison as the RSF is optimised for distribution predictions. This would be non-problematic if a suitable composition is clearly utilised. For example a regression SSVM predicting survival time cannot be directly compared to a Cox PH. However the SSVM can be compared to a CPH composed with the probabilistic to deterministic compositor (C3), then conclusions can be drawn about comparison to the composite survival time Cox model (and not simply a Cox PH). RM3. Na"ive censoring deletion. This common mistake occurs when trying to reduce survival to regression or classification by simply deleting all censored observations, even if censoring is informative. This is a mistake as it creates bias in the dataset, which can be substantial if the proportion of censoring is high and informative. More robust deletion methods are described in **?@sec-redux-regr**. RM4. Oversampling uncensored observations. This is often seen when trying to reduce survival to regression or classification, and often alongside (RM3). Oversampling is the process of replicating observations to artificially inflate the sample size of the data. Whilst this process does not create any new information, it can help a model detect important features in the data. However, by only oversampling uncensored observations, this creates a source of bias in the data and ignores the potentially informative information provided by the proportion of censoring.

15.4 Composition Strategies for Survival Analysis

Though composition is common practice in survival analysis, with the Cox model being a prominent example, a lack of formalisation means a lack of consensus in simple operations. For example, it is often asked in survival analysis how a model predicting a survival distribution can be used to return a survival time prediction. A common strategy is to define the survival time prediction as the median of the predicted survival curve however there is no clear reason why this should be more sensible than returning the distribution mean, mode, or some random quantile. Formalisation allow these choices to be analytically compared both theoretically and practically as hyper-parameters in a workflow. Four prediction compositions are discussed in this section ((**tab-car-taxredcar?**)), three are utilised to convert prediction types between one another, the fourth is for aggregating multiple predictions. One data composition is discussed for converting survival to regression data. Each is first graphically represented and then the components are discussed in detail. As with losses in the previous chapter, compositions are discussed at an individual observation level but extend trivially to multiple observations.

Table 15.1: Compositions formalised in Section 15.4.

ID[1]	Composition	Type[2]	Level[3]
C1)	Linear predictor to distribution	MOC	Prediction
C2)	Survival time to distribution	MOC	Prediction
C3)	Distribution to survival time	SOC	Prediction
C4)	Survival model averaging	MOC	Prediction
C5)	Survival to regression	SOC	Data

1. ID for reference throughout this book. 2. Composition type. Multi-object composition (MOC) or single-object composition (SOC). 3. Composition level.

15.4.1 C1) Linear Predictor → Distribution

This is a prediction-level MOC that composes a survival distribution from a predicted linear predictor and estimated baseline survival distribution. The composition (**?@fig-car-comp-distr**) requires:

- $\hat{\eta}$: Predicted linear predictor. $\hat{\eta}$ can be tuned by including this composition multiple times in a benchmark experiment with different models predicting $\hat{\eta}$. In theory any continuous ranking could be utilised instead of a linear predictor though results may be less sensible (**?@sec-car-pipelines-trade**).
- $\hat{S}_0$: Estimated baseline survival function. This is usually estimated by the Kaplan-Meier estimator fit on training data, $\hat{S}_{KM}$. However any model that can predict a survival distribution can estimate the baseline distribution (caveat: see **?@sec-car-pipelines-trade**) by taking a uniform mixture of the predicted individual distributions: say $\xi_1, ..., \xi_m$ are m predicted distributions, then $\hat{S}_0(\tau) = \frac{1}{m}\sum_{i=1}^{m} \xi_i.S(\tau)$. The mixture is required as the baseline must be the same for all observations. Alternatively, parametric distributions can be assumed for the baseline, e.g. $\xi = \text{Exp}(2)$ and $\xi.S(t) = \exp(-2t)$. As with $\hat{\eta}$, this parameter is also tunable.
- M: Chosen model form, which theoretically can be any non-increasing right-continuous function but is usually one of:
- Proportional Hazards (PH): $S_{PH}(\tau|\eta, S_0) = S_0(\tau)^{\exp(\eta)}$
- Accelerated Failure Time (AFT): $S_{AFT}(\tau|\eta, S_0) = S_0(\frac{\tau}{\exp(\eta)})$
- Proportional Odds (PO): $S_{PO}(\tau|\eta, S_0) = \frac{S_0(\tau)}{\exp(-\eta)+(1-\exp(-\eta))S_0(\tau)}$

Models that predict linear predictors will make assumptions about the model form and therefore dictate sensible choices of M, for example the Cox model assumes a PH form. This does not mean other choices of M cannot be specified but that interpretation may be more difficult (**?@sec-car-pipelines-trade**). The model form can be treated as a hyper-parameter to tune. * C: Compositor returning the composed distribution, $\zeta := C(M, \hat{\eta}, \hat{S}_0)$ where ζ has survival function $\zeta.S(\tau) = M(\tau|\hat{\eta}, \hat{S}_0)$.

Pseudo-code for training ((**alg-car-comp-distr-fit?**)) and predicting ((**alg-car-comp-distr-pred?**)) this composition as a model 'wrapper' with sensible parameter choices (**?@sec-car-pipelines-trade**) is provided in appendix (**app-car?**).

15.4.2 C2) Survival Time → Distribution

This is a prediction-level MOC that composes a distribution from a predicted survival time and assumed location-scale distribution. The composition (**?@fig-car-comp-response**) requires:

- $\hat{T}$: A predicted survival time. As with the previous composition, this is tunable. In theory any continuous ranking could replace $\hat{T}$, though the resulting distribution may not be sensible (**?@sec-car-pipelines-trade**).
- ξ: A specified location-scale distribution, $\xi(\mu, \sigma)$, e.g. Normal distribution.
- $\hat{\sigma}$: Estimated scale parameter for the distribution. This can be treated as a hyper-parameter or predicted by another model.
- C: Compositor returning the composed distribution $\zeta := C(\xi, \hat{T}, \hat{\sigma}) = \xi(\hat{T}, \hat{\sigma})$.

Pseudo-code for training ((**alg-car-comp-response-fit?**)) and predicting ((**alg-car-comp-response-pred?**)) this composition as a model 'wrapper' with sensible parameter choices (**?@sec-car-pipelines-trade**) is provided in appendix (**app-car?**).

15.4.3 C3) Distribution → Survival Time Composition

This is a prediction-level SOC that composes a survival time from a predicted distribution. Any paper that evaluates a distribution on concordance is implicitly using this composition in some manner. Not acknowledging the composition leads to unfair model comparison (Section 15.3.3). The composition (**?@fig-car-comp-crank**) requires:

- ζ: A predicted survival distribution, which again is 'tunable'.
- ϕ: A distribution summary method. Common examples include the mean, median and mode. Other alternatives include distribution quantiles, $\zeta.F^{-1}(\alpha), \backslash\alpha \in [0, 1]$; α could be tuned as a hyper-parameter.
- C: Compositor returning composed survival time predictions, $\hat{T} := C(\phi, \zeta) = \phi(\zeta)$.

Pseudo-code for training ((**alg-car-comp-crank-fit?**)) and predicting ((**alg-car-comp-crank-pred?**)) this composition as a model 'wrapper' with sensible parameter choices (**?@sec-car-pipelines-trade**) is provided in appendix (**app-car?**).

15.4.4 C4) Survival Model Averaging

Ensembling is likely the most common composition in machine learning. In survival it is complicated slightly as multiple prediction types means one of two possible compositions is utilised to average predictions. The (**?@fig-car-comp-avg**) composition requires:

- $\rho = \rho_1, ..., \rho_B$: B predictions (not necessarily from the same model) of the same type: ranking, survival time or distribution; again 'tunable'.
- $w = w_1, ..., w_B$: Weights that sum to one.
- C: Compositor returning combined predictions, $\hat{\rho} := C(\rho, w)$ where $C(\rho, w) = \frac{1}{B}\sum_{i=1}^{B} w_i\rho_i$, if ρ are ranking of survival time predictions; or $C(\rho, w) = \zeta$ where ζ is the distribution defined by the survival function $\zeta.S(\tau) = \frac{1}{B}\sum_{i=1}^{B} w_i\rho_i.S(\tau)$, if ρ are distribution predictions.

Pseudo-code for training ((**alg-car-comp-avg-fit?**)) and predicting ((**alg-car-comp-avg-pred?**)) this composition as a model 'wrapper' with sensible parameter choices (**?@sec-car-pipelines-trade**) is provided in appendix (**app-car?**).

15.5 Novel Survival Reductions

This section collects the various strategies and settings discussed previously into complete reduction workflows. (**tab-car-reduxes?**) lists the reductions discussed in this section with IDs for future reference. All strategies are described by visualising a graphical pipeline and then listing the composition steps required in fitting and predicting.

This section only includes novel reduction strategies and does not provide a survey of pre-existing strategies. This limitation is primarily due to time (and page) constraints as every method has very distinct workflows that require complex exposition. Well-established strategies are briefly mentioned below and future research is planned to survey and compare all strategies with respect to empirical performance (i.e. in benchmark experiments).

Two prominent reductions are 'landmarking' (Van Houwelingen 2007) and piecewise exponential models (M. Friedman 1982). Both are reductions for time-varying covariates and hence outside the scope of this book. Relevant to this book scope is a large class of strategies that utilise 'discrete time survival analysis' (Tutz and Schmid 2016); these strategies include reductions (R7) and (R8). Methodology for discrete time survival analysis has been seen in the literature for the past three decades (Liestol, Andersen, and Andersen 1994). The primary reduction strategy for discrete time survival analysis is implemented in the R package `discSurv` (Welchowski and Schmid 2019); this is very similar to (R7) except that it enforces stricter constraints in the composition procedures and forces a 'discrete-hazard' instead of 'discrete-survival' representation (Section 15.5.1.2).

15.5.1 R7-R8) Survival → Probabilistic Classification

Two separate reductions are presented in **?@fig-car-R7R8** however as both are reductions to probabilistic classification and are only different in the very last step, both are presented in this section. Steps and compositions of the reduction (**?@fig-car-R7R8**):

Fit F1) A survival dataset, $\mathcal{D}_{train}$, is binned, B, with a continuous to discrete data composition (Section 15.5.1.1). F2) A multi-label classification model, with adaptations for censoring, $g_L(D_B|\theta)$, is fit on the transformed dataset, D_B. Optionally, g_L could be further reduced to binary, g_B, or multi-class classification, g_c, (Section 15.5.1.4). **Predict** P1) Testing survival data, $\mathcal{D}_{test}$, is passed to the trained classification model, $\hat{g}$, to predict pseudo-survival probabilities $\tilde{S}$ (or optionally hazards (Section 15.5.1.2)). P2a) Predictions can be composed, T_1, into a survival distribution prediction, $\zeta = \zeta_1, ..., \zeta_m$ (Section 15.5.1.6); or, P2b) Predictions can be composed, T_2, to survival time predictions, $\hat{T} = \hat{T}_1, ..., \hat{T}_m$ (Section 15.5.1.7).

Further details for binning, multi-label classification, and transformation of pseudo-survival probabilities are now provided.

15.5.1.1 Composition: Binning Survival Times

An essential part of the reduction is the transformation from a survival dataset to a classification dataset, which requires two separate compositions. The first (discussed here) is to discretise the survival times ($B(\mathcal{D}_{train}|w)$ in **?@fig-car-R7R8**) and the second is to merge the survival time and censoring indicator into a single outcome (Section 15.5.1.2).

Discretising survival times is achieved by the common 'binning' composition, in which a con-

tinuous outcome is discretised into 'bins' according to specified thresholds. These thresholds are usually determined by specifying the width of the bins as a hyper-parameter w.[2] This is a common transformation and therefore further discussion is not provided here. An example is given below with the original survival data on the left and the binned data on the right ($w = 1$).

X	Time (Cont.)	Died
1	1.56	0
2	2	1
3	3.3	1
4	3.6	0
5	4	0

X	Time (Disc.)	Died
1	[1, 2)	0
2	[2, 3)	1
3	[3, 4)	1
4	[3, 4)	0
5	[4, 5)	0

15.5.1.2 Composition: Survival to Classification Outcome

The binned dataset still has the unique survival data format of utilising two outcomes for training (time and status) but only making a prediction for one outcome (distribution). In order for this to be compatible with classification, the two outcome variables are composed into a single variable.[3] This is achieved by casting the survival times into a 'wide' format and creating a new outcome indicator.[4] Two outcome transformations are possible, the first represents a discrete survival function and the second represents a discrete hazard function.[5]

Discrete Survival Function Composition

In this composition, the data in the transformed dataset represents the discrete survival function. The new indicator is defined as follows,

$$Y_{i;\tau} := \begin{cases} 1, & T_i > \tau \\ 0, & T_i \leq \tau \cap \Delta_i = 1 \\ -1, & T_i \leq \tau \cap \Delta_i = 0 \end{cases}$$

At a given discrete time τ, an observation, i, is either alive ($Y_{i;\tau} = 1$), dead ($Y_{i;\tau} = 0$), or censored ($Y_{i;\tau} = -1$). Therefore $\hat{P}(Y_{i;\tau} = 1) = \hat{S}_i(\tau)$, motivating this particular choice of representation.

[2] Binning is described here with equal widths but generalises to unequal widths trivially.

[3] This is the first key divergence from other discrete-time classification strategies, which use the censoring indicator as the outcome and the time outcome as a feature.

[4] This is the second key divergence from other discrete-time classification strategies, which keep the data in a 'long' format.

[5] This is the final key divergence from other discrete-time classification strategies, which enforce the discrete hazard representation.

This composition is demonstrated below with the binned data (left) and the composed classification data (right).

X	Time (Disc.)	Died
1	[1, 2)	0
2	[2, 3)	1
3	[3, 4)	1
4	[3, 4)	0
5	[4, 5)	0

X	[1,2)	[2,3)	[3,4)	[4,5)
1	-1	-1	-1	-1
2	1	0	0	0
3	1	1	0	0
4	1	1	-1	-1
5	1	1	-1	-1

Discrete Hazard Function Composition

In this composition, the data in the transformed dataset represents the discrete hazard function. The new indicator is defined as follows,

$$Y^*_{i;\tau} := \begin{cases} 1, & T_i = \tau \cap \Delta_i = 1 \\ -1, & T_i = \tau \cap \Delta_i = 0 \\ 0, & \text{otherwise} \end{cases}$$

At a given discrete time τ, an observation, i, either experiences the event ($Y^*_{i;\tau} = 1$), experiences censoring ($Y_{i;\tau} = -1$), or neither ($Y_{i;\tau} = 0$). Utilising sequential multi-label classification problem transformation methods (Section 15.5.1.4) results in $\widehat{P}(Y^*_{i;\tau} = 1) = \widehat{h}_i(\tau)$. If methods are utilised that do not 'look back' at predictions then $\widehat{P}(Y^*_{i;\tau} = 1) = \widehat{p}_i(\tau)$ (Section 15.5.1.4).[6]

This composition is demonstrated below with the binned data (left) and the composed classification data (right).

X	Time (Disc.)	Died
1	[1, 2)	0
2	[2, 3)	1
3	[3, 4)	1
4	[3, 4)	0
5	[4, 5)	0

[6]This important distinction is not required in other discrete-time reduction strategies that automatically condition the prediction by including time as a feature.

X	[1,2)	[2,3)	[3,4)	[4,5)
1	-1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	-1	0
5	0	0	0	-1

Multi-Label Classification Data

In both compositions, survival data t.v.i. $\mathbb{R}^p \times \mathbb{R}_{\geq 0} \times \{0, 1\}$ is transformed to multi-label classification data t.v.i. $\mathbb{R}^p \times \{-1, 0, 1\}^K$ for K binned time-intervals. The multi-label classification task is defined in Section 15.5.1.4 with possible algorithms.

The discrete survival representation has a slightly more natural interpretation and is 'easier' for classifiers to use for training as there are more positive events (i.e. more observations alive) to train on, whereas the discrete hazard representation will have relatively few events in each time-point. However the hazard representation leads to more natural predictions (Section 15.5.1.6).

A particular bias that may easily result from the composition of survival to classification data is now discussed.

15.5.1.3 Reduction to Classification Bias

The reduction to classification bias is commonly known (Zhou et al. 2005) but is reiterated briefly here as it must be accounted for in any automated reduction to classification workflow. This bias occurs when making classification predictions about survival at a given time and incorrectly censoring patients who have not been observed long enough, instead of removing them.

By example, say the prediction of interest is five-year survival probabilities after a particular diagnosis, clearly a patient who has only been diagnosed for three years cannot inform this prediction. The bias is introduced if this patient is censored at five-years instead of being removed from the dataset. The result of this bias is to artificially inflate the probability of survival at each time-point as an unknown outcome is treated as censored and therefore alive.

This bias is simply dealt with by removing patients who have not been alive 'long enough'.[7] Paradoxically, even if a patient is observed to die before the time-point of interest, they should still be removed if they have not been in the dataset 'long enough' as failing to do so will result in a bias in the opposite direction, thus over-inflating the proportion of dead observations.

Accounting for this bias is particularly important in the multi-label reduction as the number of observable patients will decrease over time due to censoring.

15.5.1.4 Multi-Label Classification Algorithms

As the work in this section is completely out of the book scope, the full text is in appendix (**app-mlc?**). The most important contributions from this section are:

[7] Accounting for this bias is only possible if the study start and end dates are known, as well as the date the patient entered the study.

- Reviewing problem transformation methods (Tsoumakas and Katakis 2007) for multi-label classification;
- Identifying that only binary relevance, nested stacking, and classifier chains are appropriate in this reduction; and
- Generalising these methods into a single wrapper for any binary classifier, the 'LWrapper'.

15.5.1.5 Censoring in Classification

Classification algorithms cannot natively handle the censoring that is included in the survival reduction, but this can be incorporated using one of two approaches.

Multi-Class Classification

All multi-label datasets can also handle multi-class data, hence the simplest way in which to handle censoring is to make multi-class predictions in each label for the outcome Y_τ $t.v.i.\{-1, 0, 1\}$. Many off-shelf classification learners can make multi-class predictions natively and simple reductions exist for those that cannot. As a disadvantage to this method, classifiers would then predict if an individual is dead or alive or censored (each mutually exclusive), and not simply alive or dead. Though this could be perceived as an advantage when censoring is informative as this will accurately reflect a real-world competing-risks set-up.

Subsetting/Hurdle Models

For this approach, the multi-class task is reduced to two binary class tasks: first predict if a subject is censored or not (dead or alive) and only if the prediction for censoring is below some threshold, $\alpha \in [0, 1]$, then predict if the subject is alive or not (dead or censored). If the probability of censoring is high in the first task then the probability of being alive is automatically set to zero in the final prediction, otherwise the prediction from the second task is used. Any classifier can utilise this approach and it has a meaningful interpretation, additionally α is a tunable hyper-parameter. The main disadvantage is increases to storage and run-time requirements as double the number of models may be fit.

Once the datasets have been composed to classification datasets and censoring is suitably incorporated by either approach, then any probabilistic classification model can be fit on the data. Predictions from these models can either be composed to a distribution prediction (R7) or a survival time prediction (R8).

15.5.1.6 R7) Probabilistic Survival → Probabilistic Classification

This final part of the (R7) reduction is described separately for discrete hazard and survival representations of the data (Section 15.5.1.2).

Discrete Hazard Representation

In this representation recall that predictions of the positive class, $P(Y_\tau = 1)$, are estimating the quantity $h(\tau)$. These predictions provide a natural and efficient transformation from predicted hazards to survival probabilities. Let $\hat{h}_i$ be a predicted hazard function for some observation i, then the survival function for that observation can be found with a Kaplan-Meier type estimator,

$$\tilde{S}_i(\tau^*) = \prod_\tau 1 - \hat{h}_i(\tau)$$

Now predictions are for a pseudo-survival function, which is 'pseudo' as it is not right-continuous. Resolving this is discussed below.

Discrete Survival Representation

In this representation, $P(Y_\tau = 1)$ is estimating $S(\tau)$, which means that predictions from a classification model result in discrete point predictions and not a right-continuous function. More importantly, there is no guarantee that a non-increasing function will be predicted, i.e. there is no guarantee that $P(Y_j = 1) < P(Y_i = 1)$, for time-points $j > i$.

Unfortunately there is no optimal way of dealing with predictions of this sort and 'mistakes' of this kind have been observed in some software implementation. One point to note is that in practice these are quite rare as the probability of survival will always decrease over time. Therefore the 'usual' approach is quite 'hacky' and involves imputing increasing predictions with the previous prediction, formally,

$$\tilde{S}(i+1) := \min\{P(Y_{i+1} = 1), P(Y_i = 1)\}, \forall i = \mathbb{R}_{\geq 0}$$

assuming $\tilde{S}(0) = 1$. Future research should seek more robust alternatives.

Right-Continuous Survival Function

From either representation, a \ non-increasing but non-continuous pseudo-survival function, $\tilde{S}$, is now predicted. Creating a right-continuous function ('$T_1(\tilde{S})$' in **?@fig-car-R7**) from these point predictions (Figure 15.3 (a)) is relatively simple and well-known with accessible off-shelf software. At the very least, one can assume a constant hazard rate between predictions and cast them into a step function (Figure 15.3 (b)). This is a fairly common assumption and is usually valid as bin-width decreases. Alternatively, the point predictions can be smoothed into a continuous function with off-shelf software, for example with polynomial local regression smoothing (Figure 15.3 (c)) or generalised linear smoothing (Figure 15.3 (d)). Whichever method is chosen, the survival function is now non-increasing right-continuous and the (R7) reduction is complete.

15.5.1.7 R8) Deterministic Survival → Probabilistic Classification

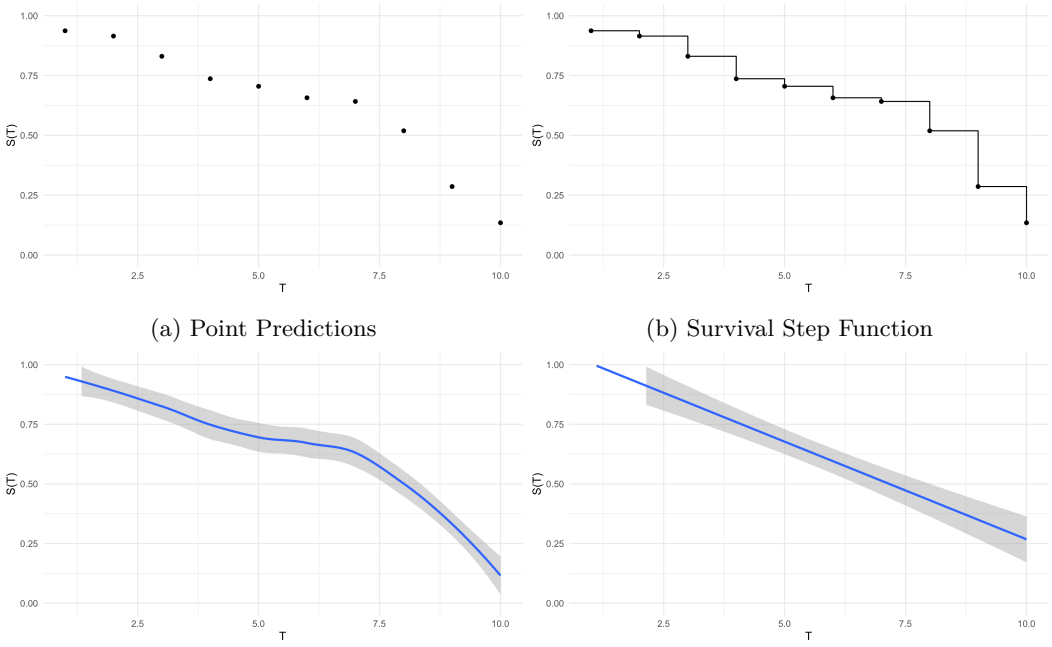
Predicting a deterministic survival time from the multi-label classification predictions is relatively straightforward and can be viewed as a discrete analogue to (C3) (Section 15.4.3). For the discrete hazard representation, one can simply take the predicted time-point for an individual to be time at which the predicted hazard probability is highest however this could easily be problematic as there may be multiple time-points at which the predicted hazard equals 1. Instead it is cleaner to first cast the hazard to a pseudo-survival probability (Section 15.5.1.6) and then treat both representations the same.

Let $\tilde{S}_i$ be the predicted multi-label survival probabilities for an observation i such that $\tilde{S}_i(\tau)$ corresponds with $\hat{P}(Y_{i;\tau} = 1)$ for label $\tau \in \mathcal{K}$ where $Y_{i;\tau}$ is defined in Section 15.5.1.2 and $\mathcal{K} = \{1, ..., K\}$ is the set of labels for which to make predictions. Then the survival time transformation is defined by

$$T_2(\tilde{S}_i) = \inf\{\tau \in \mathcal{K} : \tilde{S}_i(\tau) \leq \beta\}$$

for some $\beta \in [0, 1]$.

This is interpreted as defining the predicted survival time as the first time-point in which the predicted probability of being alive drops below a certain threshold β. Usually $\beta = 0.5$, though this can be treated as a hyper-parameter for tuning. This composition can be utilised even if predictions are not non-increasing, as only the first time the predicted survival probability drops below the threshold is considered. With this composition the (R8) reduction is now complete.



(a) Point Predictions

(b) Survival Step Function

(c) Local polynomial regression smoothing

(d) Generalised linear smoothing

Figure 15.3: Survival function as a: point prediction (a), step function assuming constant risk (b), local polynomial regression smoothing (c), and generalised linear smoothing (d). (c) and (d) computed with `ggplot2` (Wickham 2016).

15.6 Conclusions

This chapter introduced composition and reduction to survival analysis and formalised specific strategies. Formalising these concepts allows for better quality of research and most importantly improved transparency. Clear interface points for hyper-parameters and compositions allow for reproducibility that was previously obfuscated by unclear workflows and imprecise documentation for pipelines.

Additionally, composition and reduction improves accessibility. Reduction workflows vastly increase the number of machine learning models that can be utilised in survival analysis, thus opening the field to those whose experience is limited to regression or classification. Formalisation of workflows allows for precise implementation of model-agnostic pipelines as computational objects, as opposed to functions that are built directly into an algorithm without external interface points.

Finally, predictive performance is also increased by these methods, which is most prominently the case for the survival model averaging compositor (C4) (as demonstrated by RSFs).

All compositions in this chapter, as well as (R1)-(R6), have been implemented in `mlr3proba` with the `mlr3pipelines` (M. Binder et al. 2019) interface. The reductions to classification will be implemented in a near-future update. Additionally the `discSurv` package (Welchowski and Schmid 2019) will be interfaced as a `mlr3proba` pipeline to incorporate further discrete-time strategies.

The compositions (C1) and (C3) are included in the benchmark experiment in R. E. B. Sonabend (2021) so that every tested model can make probabilistic survival distribution predictions as well as deterministic survival time predictions. Future research will benchmark all the pipelines in this chapter and will cover algorithm and model selection, tuning, and comparison of performance. Strategies from other papers will also be explored.

16

Competing Risks Pipelines

TODO (150-200 WORDS)

⚠ Page coming soon!

We are working on this page and it will be available soon!

References

Aalen, Odd. 1978. "Nonparametric Inference for a Family of Counting Processes." *The Annals of Statistics* 6 (4): 701–26.

Akaike, Hirotugu. 1974. "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* 19 (6): 716–23. https://doi.org/10.1093/ietfec/e90-a.12.2762.

Akritas, Michael G. 1994. "Nearest Neighbor Estimation of a Bivariate Distribution Under Random Censoring." *Ann. Statist.* 22 (3): 1299–1327. https://doi.org/10.1214/aos/1176325630.

Andres, Axel, Aldo Montano-Loza, Russell Greiner, Max Uhlich, Ping Jin, Bret Hoehn, David Bigam, James Andrew Mark Shapiro, and Norman Mark Kneteman. 2018. "A novel learning algorithm to predict individual survival after liver transplantation for primary sclerosing cholangitis." *PLOS ONE* 13 (3): e0193523. https://doi.org/10.1371/journal.pone.0193523.

Antolini, Laura, Patrizia Boracchi, and Elia Biganzoli. 2005. "A time-dependent discrimination index for survival data." *Statistics in Medicine* 24 (24): 3927–44. https://doi.org/10.1002/sim.2427.

Avati, Anand, Tony Duan, Sharon Zhou, Kenneth Jung, Nigam H. Shah, and Andrew Ng. 2020. "Countdown Regression: Sharp and Calibrated Survival Predictions." In *Proceedings of Machine Learning Research*, 145——155. https://proceedings.mlr.press/v115/avati20a.html http://arxiv.org/abs/1806.08324.

Becker, Marc, Lennart Schneider, and Sebastian Fischer. 2024. "Hyperparameter Optimization." In *Applied Machine Learning Using mlr3 in R*, edited by Bernd Bischl, Raphael Sonabend, Lars Kotthoff, and Michel Lang. CRC Press. https://mlr3book.mlr-org.com/hyperparameter_optimization.html.

Bennett, Steve. 1983. "Analysis of survival data by the proportional odds model." *Statistics in Medicine* 2 (2): 273–77. https://doi.org/https://doi.org/10.1002/sim.4780020223.

Biganzoli, Elia, Patrizia Boracchi, Luigi Mariani, and Ettore Marubini. 1998. "Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach." *Statistics in Medicine* 17 (10): 1169–86. https://doi.org/10.1002/(SICI)1097-0258(19980530)17:10%3C1169::AID-SIM796%3E3.0.CO;2-D.

Binder, Harald, and Martin Schumacher. 2008. "Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models." *BMC Bioinformatics* 9 (1): 14. https://doi.org/10.1186/1471-2105-9-14.

Binder, Harold. 2013. "CoxBoost: Cox models by likelihood based boosting for a single survival endpoint or competing risks." CRAN.

Binder, Martin, Florian Pfisterer, Bernd Bischl, Michel Lang, and Susanne Dandl. 2019. "mlr3pipelines: Preprocessing Operators and Pipelines for 'mlr3'." CRAN. https://cran.r-project.org/package=mlr3pipelines.

Bishop, Christopher M. 2006. *Pattern recognition and machine learning*. springer.

Blanche, Paul, Jean-François Dartigues, and Hélène Jacqmin-Gadda. 2013. "Review and comparison of ROC curve estimators for a time-dependent outcome with marker-dependent censoring." *Biometrical Journal* 55 (5): 687–704. https://doi.org/10.1002/

bimj.201200045.

Blanche, Paul, Aurélien Latouche, and Vivian Viallon. 2012. "Time-dependent AUC with right-censored data: a survey study," October. https://doi.org/10.1007/978-1-4614-8981-8_11.

Bland, J Martin, and Douglas G. Altman. 2004. "The logrank test." *BMJ (Clinical Research Ed.)* 328 (7447): 1073. https://doi.org/10.1136/bmj.328.7447.1073.

Bou-Hamad, Imad, Denis Larocque, and Hatem Ben-Ameur. 2011. "A review of survival trees." *Statist. Surv.* 5: 44–71. https://doi.org/10.1214/09-SS047.

Bower, Hannah, Michael J Crowther, Mark J Rutherford, Therese M.-L. Andersson, Mark Clements, Xing-Rong Liu, Paul W Dickman, and Paul C Lambert. 2019. "Capturing simple and complex time-dependent effects using flexible parametric survival models: A simulation study." *Communications in Statistics - Simulation and Computation*, July, 1–17. https://doi.org/10.1080/03610918.2019.1634201.

Breiman, Leo. 1996. "Bagging Predictors." *Machine Learning* 24 (2): 123–40. https://doi.org/10.1023/A:1018054314350.

Breiman, Leo, and Philip Spector. 1992. "Submodel Selection and Evaluation in Regression. The X-Random Case." *International Statistical Review / Revue Internationale de Statistique* 60 (3): 291–319. https://doi.org/10.2307/1403680.

Brier, Glenn. 1950. "Verification of forecasts expressed in terms of probability." *Monthly Weather Review* 78 (1): 1–3.

Buckley, Jonathan, and Ian James. 1979. "Linear Regression with Censored Data." *Biometrika* 66 (3): 429–36. https://doi.org/10.2307/2335161.

Buhlmann, Peter. 2006. "Boosting for high-dimensional linear models." *Ann. Statist.* 34 (2): 559–83. https://doi.org/10.1214/009053606000000092.

Buhlmann, Peter, and Torsten Hothorn. 2007. "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statist. Sci.* 22 (4): 477–505. https://doi.org/10.1214/07-STS242.

Bühlmann, Peter, and Bin Yu. 2003. "Boosting With the L2 Loss." *Journal of the American Statistical Association* 98 (462): 324–39. https://doi.org/10.1198/016214503000125.

Casalicchio, Giuseppe, and Lukas Burk. 2024. "Evaluation and Benchmarking." In *Applied Machine Learning Using mlr3 in R*, edited by Bernd Bischl, Raphael Sonabend, Lars Kotthoff, and Michel Lang. CRC Press. https://mlr3book.mlr-org.com/chapters/chapter3/evaluation_and_benchmarking.html.

Chambless, Lloyd E, and Guoqing Diao. 2006. "Estimation of time-dependent area under the ROC curve for long-term risk prediction." *Statistics in Medicine* 25 (20): 3474–86. https://doi.org/10.1002/sim.2299.

Chen, Tianqi, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. 2020. "xgboost: Extreme Gradient Boosting." CRAN. https://cran.r-project.org/package=xgboost.

Chen, Yifei, Zhenyu Jia, Dan Mercola, and Xiaohui Xie. 2013. "A Gradient Boosting Algorithm for Survival Analysis via Direct Optimization of Concordance Index." Edited by Lev Klebanov. *Computational and Mathematical Methods in Medicine* 2013: 873595. https://doi.org/10.1155/2013/873595.

Ching, Travers. 2015. "Cox-Nnet." https://github.com/lanagarmire/cox-nnet.

Ching, Travers, Xun Zhu, and Lana X Garmire. 2018. "Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data." *PLOS Computational Biology* 14 (4): e1006076. https://doi.org/10.1371/journal.pcbi.1006076.

Choodari-Oskooei, Babak, Patrick Royston, and Mahesh K. B. Parmar. 2012. "A simulation study of predictive ability measures in a survival model I: Explained variation measures." *Statistics in Medicine* 31 (23): 2627–43. https://doi.org/10.1002/sim.4242.

Ciampi, Antonio, Sheilah A Hogg, Steve McKinney, and Johanne Thiffault. 1988. "REC-

PAM: a computer program for recursive partition and amalgamation for censored survival data and other situations frequently occurring in biostatistics. I. Methods and program features." *Computer Methods and Programs in Biomedicine* 26 (3): 239–56. https://doi.org/https://doi.org/10.1016/0169-2607(88)90004-1.

Ciampi, Antonio, Johanne Thiffault, Jean Pierre Nakache, and Bernard Asselain. 1986. "Stratification by stepwise regression, correspondence analysis and recursive partition: a comparison of three methods of analysis for survival data with covariates." *Computational Statistics and Data Analysis* 4 (3): 185–204. https://doi.org/10.1016/0167-9473(86)90033-2.

Collett, David. 2014. *Modelling Survival Data in Medical Research*. 3rd ed. CRC.

Collins, Gary S., Joris A. De Groot, Susan Dutton, Omar Omar, Milensu Shanyinde, Abdelouahid Tajar, Merryn Voysey, et al. 2014. "External validation of multivariable prediction models: A systematic review of methodological conduct and reporting." *BMC Medical Research Methodology* 14 (1): 1–11. https://doi.org/10.1186/1471-2288-14-40.

Colosimo, Enrico, Fla´vio Ferreira, Maristela Oliveira, and Cleide Sousa. 2002. "Empirical comparisons between Kaplan-Meier and Nelson-Aalen survival function estimators." *Journal of Statistical Computation and Simulation* 72 (4): 299–308. https://doi.org/10.1080/00949650212847.

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

———. 1975. "Partial Likelihood." *Biometrika* 62 (2): 269–76. https://doi.org/10.1080/03610910701884021.

Data Study Group Team. 2020. "Data Study Group Final Report: Great Ormond Street Hospital." https://doi.org/10.5281/zenodo.3670726.

Dawid, A P. 1984. "Present Position and Potential Developments: Some Personal Views: Statistical Theory: The Prequential Approach." *Journal of the Royal Statistical Society. Series A (General)* 147 (2): 278–92. https://doi.org/10.2307/2981683.

Dawid, A Philip. 1986. "Probability Forecasting." *Encyclopedia of Statistical Sciences* 7: 210——218.

Dawid, A Philip, and Monica Musio. 2014. "Theory and Applications of Proper Scoring Rules." *Metron* 72 (2): 169–83. https://arxiv.org/abs/arXiv:1401.0398v1.

Demler, Olga V, Nina P Paynter, and Nancy R Cook. 2015. "Tests of calibration and goodness-of-fit in the survival setting." *Statistics in Medicine* 34 (10): 1659–80. https://doi.org/10.1002/sim.6428.

Demšar, Janez. 2006. "Statistical comparisons of classifiers over multiple data sets." *Journal of Machine Learning Research* 7 (Jan): 1–30.

Dietterich, Thomas G. 1998. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms." *Neural Computation* 10 (7): 1895–1923. https://doi.org/10.1162/089976698300017197.

Faraggi, David, and Richard Simon. 1995. "A neural network model for survival data." *Statistics in Medicine* 14 (1): 73–82. https://doi.org/10.1002/sim.4780140108.

Fleming, Thomas R, Judith R O'Fallon, Peter C O'Brien, and David P Harrington. 1980. "Modified Kolmogorov-Smirnov Test Procedures with Application to Arbitrarily Right-Censored Data." *Biometrics* 36 (4): 607–25. https://doi.org/10.2307/2556114.

Foss, Natalie, and Lars Kotthoff. 2024. "Data and Basic Modeling." In *Applied Machine Learning Using mlr3 in R*, edited by Bernd Bischl, Raphael Sonabend, Lars Kotthoff, and Michel Lang. CRC Press. https://mlr3book.mlr-org.com/data_and_basic_modeling.html.

Fouodo, Cesaire J K, I Konig, C Weihs, A Ziegler, and M Wright. 2018. "Support vector machines for survival analysis with R." *The R Journal* 10 (July): 412–23.

Freund, Yoav, and Robert E Schapire. 1996. "Experiments with a new boosting algorithm."

In. Citeseer.

Friedman, Jerome. 1999. "Stochastic Gradient Boosting." *Computational Statistics & Data Analysis* 38 (March): 367–78. https://doi.org/10.1016/S0167-9473(01)00065-2.

Friedman, Jerome H. 2001. "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics* 29 (5): 1189–1232. http://www.jstor.org/stable/2699986.

Friedman, Michael. 1982. "Piecewise exponential models for survival data with covariates." *The Annals of Statistics* 10 (1): 101–13.

Gelfand, Alan E, Sujit K Ghosh, Cindy Christiansen, Stephen B Soumerai, and Thomas J McLaughlin. 2000. "Proportional hazards models: a latent competing risk approach." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 49 (3): 385–97. https://doi.org/https://doi.org/10.1111/1467-9876.00199.

Gensheimer, Michael F., and Balasubramanian Narasimhan. 2018. "A Simple Discrete-Time Survival Model for Neural Networks," 1–17. https://doi.org/arXiv:1805.00917v3.

Gensheimer, Michael F, and Balasubramanian Narasimhan. 2019. "A scalable discrete-time survival model for neural networks." *PeerJ* 7: e6257.

Georgousopoulou, Ekavi N, Christos Pitsavos, Christos Mary Yannakoulia, and Demosthenes B Panagiotakos. 2015. "Comparisons between Survival Models in Predicting Cardiovascular Disease Events : Application in the ATTICA Study (2002-2012)." *Journal of Statistics Applications & Probability* 4 (2): 203–10.

Gerds, Thomas A, and Martin Schumacher. 2006. "Consistent Estimation of the Expected Brier Score in General Survival Models with Right-Censored Event Times." *Biometrical Journal* 48 (6): 1029–40. https://doi.org/10.1002/bimj.200610301.

Gneiting, Tilmann, and Adrian E Raftery. 2007. "Strictly Proper Scoring Rules, Prediction, and Estimation." *Journal of the American Statistical Association* 102 (477): 359–78. https://doi.org/10.1198/016214506000001437.

Goli, Shahrbanoo, Hossein Mahjub, Javad Faradmal, Hoda Mashayekhi, and Ali-Reza Soltanian. 2016. "Survival Prediction and Feature Selection in Patients with Breast Cancer Using Support Vector Regression." Edited by Francesco Pappalardo. *Computational and Mathematical Methods in Medicine* 2016: 2157984. https://doi.org/10.1155/2016/2157984.

Goli, Shahrbanoo, Hossein Mahjub, Javad Faradmal, and Ali-Reza Soltanian. 2016. "Performance Evaluation of Support Vector Regression Models for Survival Analysis: A Simulation Study." *International Journal of Advanced Computer Science and Applications* 7 (June). https://doi.org/10.14569/IJACSA.2016.070650.

Gompertz, Benjamin. 1825. "On the Nature of the Function Expressive of the Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies." *Philosophical Transactions of the Royal Society of London* 115: 513–83.

Gönen, Mithat, and Glenn Heller. 2005. "Concordance Probability and Discriminatory Power in Proportional Hazards Regression." *Biometrika* 92 (4): 965–70.

Good, I J. 1952. "Rational Decisions." *Journal of the Royal Statistical Society. Series B (Methodological)* 14 (1): 107–14. http://www.jstor.org/stable/2984087.

Gordon, Louis, and Richard A Olshen. 1985. "Tree-structured survival analysis." *Cancer Treatment Reports* 69 (10): 1065–69.

Graf, Erika, Claudia Schmoor, Willi Sauerbrei, and Martin Schumacher. 1999. "Assessment and comparison of prognostic classification schemes for survival data." *Statistics in Medicine* 18 (17-18): 2529–45. https://doi.org/10.1002/(SICI)1097-0258(19990915/30)18:17/18%3C2529::AID-SIM274%3E3.0.CO;2-5.

Graf, Erika, and Martin Schumacher. 1995. "An Investigation on Measures of Explained Variation in Survival Analysis." *Journal of the Royal Statistical Society. Series D (The Statistician)* 44 (4): 497–507. https://doi.org/10.2307/2348898.

Greenwell, Brandon, Bradley Boehmke, Jay Cunningham, and. GBM Developers. 2019.

"gbm: Generalized Boosted Regression Models." CRAN. https://cran.r-project.org/package=gbm.

Gressmann, Frithjof, Franz J. Király, Bilal Mateen, and Harald Oberhauser. 2018. "Probabilistic supervised learning." https://doi.org/10.1002/iub.552.

Habibi, Danial, Mohammad Rafiei, Ali Chehrei, Zahra Shayan, and Soheil Tafaqodi. 2018. "Comparison of Survival Models for Analyzing Prognostic Factors in Gastric Cancer Patients." *Asian Pacific Journal of Cancer Prevention : APJCP* 19 (3): 749–53. https://doi.org/10.22034/APJCP.2018.19.3.749.

Haider, Humza, Bret Hoehn, Sarah Davis, and Russell Greiner. 2020. "Effective ways to build and evaluate individual survival distributions." *Journal of Machine Learning Research* 21 (85): 1–63.

Harrell, F E Jr, K L Lee, R M Califf, D B Pryor, and R A Rosati. 1984. "Regression modelling strategies for improved prognostic prediction." *Statistics in Medicine* 3 (2): 143–52. https://doi.org/10.1002/sim.4780030207.

Harrell, Frank E., Robert M. Califf, and David B. Pryor. 1982. "Evaluating the yield of medical tests." *JAMA* 247 (18): 2543–46. http://dx.doi.org/10.1001/jama.1982.03320430047030.

Harrell, Frank E., Kerry L. Lee, and Daniel B. Mark. 1996. "Multivariable Prognostic Models: Issues in Developing Models, Evaluating Assumptions and Adequacy, and Measuring and Reducing Errors." *Statistics in Medicine* 15: 361–87. https://doi.org/10.1002/0470023678.ch2b(i).

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer New York Inc.

Heagerty, Patrick J., Thomas Lumley, and Margaret S. Pepe. 2000. "Time-Dependent ROC Curves for Censored Survival Data and a Diagnostic Marker." *Biometrics* 56 (2): 337–44. https://doi.org/10.1111/j.0006-341X.2000.00337.x.

Heagerty, Patrick J, and Yingye Zheng. 2005. "Survival Model Predictive Accuracy and ROC Curves." *Biometrics* 61 (1): 92–105. https://doi.org/10.1111/j.0006-341X.2005.030814.x.

Henderson, and Velleman. 1981. "Building multiple regression models interactively." *Biometrics* 37: 391——411.

Herrmann, Moritz, Philipp Probst, Roman Hornung, Vindi Jurinovic, and Anne-Laure Boulesteix. 2021. "Large-scale benchmark study of survival prediction methods using multi-omics data." *Briefings in Bioinformatics* 22 (3). https://doi.org/10.1093/bib/bbaa167.

Hielscher, Thomas, Manuela Zucknick, Wiebke Werft, and Axel Benner. 2010. "On the Prognostic Value of Gene Expression Signatures for Censored Data BT - Advances in Data Analysis, Data Handling and Business Intelligence." In, edited by Andreas Fink, Berthold Lausen, Wilfried Seidel, and Alfred Ultsch, 663–73. Berlin, Heidelberg: Springer Berlin Heidelberg.

Hosmer, David W, and Stanley Lemeshow. 1980. "Goodness of fit tests for the multiple logistic regression model." *Communications in Statistics-Theory and Methods* 9 (10): 1043–69.

Hosmer Jr, David W, Stanley Lemeshow, and Susanne May. 2011. *Applied survival analysis: regression modeling of time-to-event data*. Vol. 618. John Wiley & Sons.

Hothorn, Torsten, Peter Buehlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. 2020. "mboost: Model-Based Boosting." CRAN. https://cran.r-project.org/package=mboost.

Hothorn, Torsten, Peter Bühlmann, Sandrine Dudoit, Annette Molinaro, and Mark J Van Der Laan. 2005. "Survival ensembles." *Biostatistics* 7 (3): 355–73. https://doi.org/10.1093/biostatistics/kxj011.

Hothorn, Torsten, Kurt Hornik, and Achim Zeileis. 2006. "Unbiased Recursive Partitioning:

A Conditional Inference Framework." *Journal of Computational and Graphical Statistics* 15 (3): 651——674.

Hothorn, Torsten, and Berthold Lausen. 2003. "On the exact distribution of maximally selected rank statistics." *Computational Statistics & Data Analysis* 43 (2): 121–37. https://doi.org/10.1016/S0167-9473(02)00225-6.

Hothorn, Torsten, Berthold Lausen, Axel Benner, and Martin Radespiel-Tröger. 2004. "Bagging survival trees." *Statistics in Medicine* 23 (1): 77–91. https://doi.org/10.1002/sim.1593.

Hothorn, Torsten, and Achim Zeileis. 2015. "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research* 16: 3905–9. http://jmlr.org/papers/v16/hothorn15a.html.

Huang, Shigao, Jie Yang, Simon Fong, and Qi Zhao. 2020. "Artificial intelligence in cancer diagnosis and prognosis: Opportunities and challenges." *Cancer Letters* 471: 61–71. https://doi.org/https://doi.org/10.1016/j.canlet.2019.12.007.

Hung, Hung, and Chin-Tsang Chiang. 2010. "Estimation methods for time-dependent AUC models with survival data." *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* 38 (1): 8–26. http://www.jstor.org/stable/27805213.

Hurvich, Clifford M, and Chih-Ling Tsai. 1989. "Regression and time series model selection in small samples." *Biometrika* 76 (2): 297–307. https://doi.org/10.1093/biomet/76.2.297.

Ishwaran, By Hemant, Udaya B Kogalur, Eugene H Blackstone, and Michael S Lauer. 2008. "Random survival forests." *The Annals of Statistics* 2 (3): 841–60. https://doi.org/10.1214/08-AOAS169.

Ishwaran, Hemant, Eugene H Blackstone, Claire E Pothier, and Michael S Lauer. 2004. "Relative Risk Forests for Exercise Heart Rate Recovery as a Predictor of Mortality." *Journal of the American Statistical Association* 99 (467): 591–600. https://doi.org/10.1198/016214504000000638.

Ishwaran, Hemant, and Udaya B Kogalur. 2018. "randomForestSRC." https://cran.r-project.org/package=randomForestSRC.

Jackson, Christopher. 2016. "flexsurv: A Platform for Parametric Survival Modeling in R." *Journal of Statistical Software* 70 (8): 1–33.

Jager, Kitty J, Paul C van Dijk, Carmine Zoccali, and Friedo W Dekker. 2008. "The analysis of survival data: the Kaplan–Meier method." *Kidney International* 74 (5): 560–65. https://doi.org/https://doi.org/10.1038/ki.2008.217.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning*. Vol. 112. New York: Springer.

Jing, Bingzhong, Tao Zhang, Zixian Wang, Ying Jin, Kuiyuan Liu, Wenze Qiu, Liangru Ke, et al. 2018. "RankDeepSurv." https://github.com/sysucc-ailab/RankDeepSurv.

———, et al. 2019. "A deep survival analysis method based on ranking." *Artificial Intelligence in Medicine* 98: 1–9. https://doi.org/https://doi.org/10.1016/j.artmed.2019.06.001.

Johnson, Brent A, and Qi Long. 2011. "Survival ensembles by the sum of pairwise differences with application to lung cancer microarray studies." *Ann. Appl. Stat.* 5 (2A): 1081–101. https://doi.org/10.1214/10-AOAS426.

Kalbfleisch, J. D., and R. L. Prentice. 1973. "Marginal likelihoods based on Cox's regression and life model." *Biometrika* 60 (2): 267–78. https://doi.org/10.1093/biomet/60.2.267.

Kalbfleisch, John D, and Ross L Prentice. 2011. *The statistical analysis of failure time data*. Vol. 360. John Wiley & Sons.

Kamarudin, Adina Najwa, Trevor Cox, and Ruwanthi Kolamunnage-Dona. 2017. "Time-dependent ROC curve analysis in medical research: Current methods and applications." *BMC Medical Research Methodology* 17 (1): 1–19. https://doi.org/10.1186/s12874-017-0332-6.

Katzman, Jared L, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. 2018. "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network." *BMC Medical Research Methodology* 18 (1): 24. https://doi.org/10.1186/s12874-018-0482-1.

Kent, John T., and John O'Quigley. 1988. "Measures of dependence for censored survival data." *Biometrika* 75 (3): 525–34. https://doi.org/10.1093/biomet/75.3.525.

Khan, Faisal M., and Valentina Bayer Zubek. 2008. "Support vector regression for censored data (SVRc): A novel tool for survival analysis." *Proceedings - IEEE International Conference on Data Mining, ICDM*, 863–68. https://doi.org/10.1109/ICDM.2008.50.

Király, Franz J., Markus Löning, Anthony Blaom, Ahmed Gueciouer, and Raphael Sonabend. 2021. "Designing Machine Learning Toolboxes: Concepts, Principles and Patterns." *arXiv*, January. http://arxiv.org/abs/2101.04938.

Kirmani, S N U A, and Ramesh C Gupta. 2001. "On the Proportional Odds Model in Survival Analysis." *Annals of the Institute of Statistical Mathematics* 53 (2): 203–16. https://doi.org/10.1023/A:1012458303498.

Klein, John P, and Melvin L Moeschberger. 2003. *Survival analysis: techniques for censored and truncated data*. 2nd ed. Springer Science & Business Media.

Kohavi, Ron. 1995. "A study of cross-validation and bootstrap for accuracy estimation and model selection." *Ijcai* 14 (2): 1137–45.

Korn, Edward L., and Richard Simon. 1990. "Measures of explained variation for survival data." *Statistics in Medicine* 9 (5): 487–503. https://doi.org/10.1002/sim.4780090503.

Korn, Edward L, and Richard Simon. 1991. "Explained Residual Variation, Explained Risk, and Goodness of Fit." *The American Statistician* 45 (3): 201–6. https://doi.org/10.2307/2684290.

Kvamme, Håvard. 2018. "Pycox." https://pypi.org/project/pycox/.

Kvamme, Håvard, Ørnulf Borgan, and Ida Scheel. 2019. "Time-to-event prediction with neural networks and Cox regression." *Journal of Machine Learning Research* 20 (129): 1–30.

Langford, John, Paul Mineiro, Alina Beygelzimer, and Hal Daume. 2016. "Learning Reductions that Really Work." *Proceedings of the IEEE* 104 (1).

Lawless, Jerald F, and Yan Yuan. 2010. "Estimation of prediction error for survival models." *Statistics in Medicine* 29 (2): 262–74. https://doi.org/10.1002/sim.3758.

LeBlanc, Michael, and John Crowley. 1992. "Relative Risk Trees for Censored Survival Data." *Biometrics* 48 (2): 411–25. https://doi.org/10.2307/2532300.

———. 1993. "Survival Trees by Goodness of Split." *Journal of the American Statistical Association* 88 (422): 457–67. https://doi.org/10.2307/2290325.

Lee, Changhee, William Zame, Jinsung Yoon, and Mihaela Van der Schaar. 2018. "DeepHit: A Deep Learning Approach to Survival Analysis With Competing Risks." *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (1). https://doi.org/10.1609/aaai.v32i1.11842.

Lee, Donald K K, Ningyuan Chen, and Hemant Ishwaran. 2019. "Boosted nonparametric hazards with time-dependent covariates." https://arxiv.org/abs/arXiv:1701.07926v6.

Li, Liang, Tom Greene, and Bo Hu. 2018. "A simple method to estimate the time-dependent receiver operating characteristic curve and the area under the curve with right censored data." *Statistical Methods in Medical Research* 27 (8): 2264–78. https://doi.org/10.1177/0962280216680239.

Liang, Hua, and Guohua Zou. 2008. "Improved AIC Selection Strategy for Survival Analysis." *Computational Statistics & Data Analysis* 52 (5): 2538–48. https://doi.org/10.1016/j.csda.2007.09.003.

Liestol, Knut, Per Kragh Andersen, and Ulrich Andersen. 1994. "Survival analysis and neural nets." *Statistics in Medicine* 13 (12): 1189–1200. https://doi.org/10.1002/sim.

4780131202.

Lundberg, Scott M, and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems* 30.

Luxhoj, James T., and Huan Jyh Shyur. 1997. "Comparison of proportional hazards models and neural networks for reliability estimation." *Journal of Intelligent Manufacturing* 8 (3): 227–34. https://doi.org/10.1023/A:1018525308809.

Ma, Shuangge, and Jian Huang. 2006. "Regularized ROC method for disease classification and biomarker selection with microarray data." *Bioinformatics (Oxford, England)* 21 (January): 4356–62. https://doi.org/10.1093/bioinformatics/bti724.

Mayr, Andreas, Benjamin Hofner, and Matthias Schmid. 2016. "Boosting the discriminatory power of sparse survival models via optimization of the concordance index and stability selection." *BMC Bioinformatics* 17 (1): 288. https://doi.org/10.1186/s12859-016-1149-8.

Mayr, Andreas, and Matthias Schmid. 2014. "Boosting the concordance index for survival data–a unified framework to derive and evaluate biomarker combinations." *PloS One* 9 (1): e84483–83. https://doi.org/10.1371/journal.pone.0084483.

Meinshausen, Nicolai, and Peter Bühlmann. 2010. "Stability selection." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72 (4): 417–73. https://doi.org/10.1111/j.1467-9868.2010.00740.x.

Moghimi-dehkordi, Bijan, Azadeh Safaee, Mohamad Amin Pourhoseingholi, Reza Fatemi, Ziaoddin Tabeie, and Mohammad Reza Zali. 2008. "Statistical Comparison of Survival Models for Analysis of Cancer Data." *Asian Pacific Journal of Cancer Prevention* 9: 417–20.

Molnar, Christoph. 2019. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

Murphy, Allan H. 1973. "A New Vector Partition of the Probability Score." *Journal of Applied Meteorology and Climatology* 12 (4): 595–600. https://doi.org/10.1175/1520-0450(1973)012%3C0595:ANVPOT%3E2.0.CO;2.

Nadeau, Claude, and Yoshua Bengio. 2003. "Inference for the Generalization Error." *Machine Learning* 52 (3): 239–81. https://doi.org/10.1023/A:1024068626366.

Nasejje, Justine B, Henry Mwambi, Keertan Dheda, and Maia Lesosky. 2017. "A comparison of the conditional inference survival forest model to random survival forests based on a simulation study as well as on two applications with time-to-event data." *BMC Medical Research Methodology* 17 (1): 115. https://doi.org/10.1186/s12874-017-0383-8.

Nelson, Wayne. 1972. "Theory and Applications of Hazard Plotting for Censored Failure Data." *Technometrics* 14 (4): 945–66.

Ng, Ryan, Kathy Kornas, Rinku Sutradhar, Walter P. Wodchis, and Laura C. Rosella. 2018. "The current application of the Royston-Parmar model for prognostic modeling in health research: a scoping review." *Diagnostic and Prognostic Research* 2 (1): 4. https://doi.org/10.1186/s41512-018-0026-5.

Patel, Katie, Richard Kay, and Lucy Rowell. 2006. "Comparing proportional hazards and accelerated failure time models: An application in influenza." *Pharmaceutical Statistics* 5 (3): 213–24. https://doi.org/10.1002/pst.213.

Peters, Andrea, and Torsten Hothorn. 2019. "ipred: Improved Predictors." CRAN. https://cran.r-project.org/package=ipred.

Pölsterl, Sebastian. 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.

Qi, Jiezhi. 2009. "Comparison of Proportional Hazards and Accelerated Failure Time Models." PhD thesis.

R., Cox, and Snell J. 1968. "A General Definition of Residuals." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 30 (2): 248–75.

Rahman, M. Shafiqur, Gareth Ambler, Babak Choodari-Oskooei, and Rumana Z. Omar. 2017. "Review and evaluation of performance measures for survival prediction models in external validation settings." *BMC Medical Research Methodology* 17 (1): 1–15. https://doi.org/10.1186/s12874-017-0336-2.

Reid, Nancy. 1994. "A Conversation with Sir David Cox." *Statistical Science* 9 (3): 439–55. https://doi.org/10.1214/aos/1176348654.

Ridgeway, Greg. 1999. "The state of boosting." *Computing Science and Statistics* 31: 172—–181.

Rindt, David, Robert Hu, David Steinsaltz, and Dino Sejdinovic. 2022. "Survival Regression with Proper Scoring Rules and Monotonic Neural Networks," March. http://arxiv.org/abs/2103.14755.

Royston, Patrick, and Douglas G. Altman. 2013. "External validation of a Cox prognostic model: Principles and methods." *BMC Medical Research Methodology* 13 (1). https://doi.org/10.1186/1471-2288-13-33.

Royston, Patrick, and Mahesh K. B. Parmar. 2002. "Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects." *Statistics in Medicine* 21 (15): 2175–97. https://doi.org/10.1002/sim.1203.

Royston, Patrick, and Willi Sauerbrei. 2004. "A new measure of prognostic separation in survival data." *Statistics in Medicine* 23 (5): 723–48. https://doi.org/10.1002/sim.1621.

Sashegyi, Andreas, and David Ferry. 2017. "On the Interpretation of the Hazard Ratio and Communication of Survival Benefit." *The Oncologist* 22 (4): 484–86. https://doi.org/10.1634/theoncologist.2016-0198.

Schemper, Michael, and Robin Henderson. 2000. "Predictive Accuracy and Explained Variation in Cox Regression." *Biometrics* 56: 249–55. https://doi.org/10.1002/sim.1486.

Schmid, Matthias, Thomas Hielscher, Thomas Augustin, and Olaf Gefeller. 2011. "A Robust Alternative to the Schemper-Henderson Estimator of Prediction Error." *Biometrics* 67 (2): 524–35. https://doi.org/10.1111/j.1541-0420.2010.01459.x.

Schmid, Matthias, and Torsten Hothorn. 2008a. "Boosting additive models using component-wise P-splines." *Computational Statistics & Data Analysis* 53 (2): 298–311.

———. 2008b. "Flexible boosting of accelerated failure time models." *BMC Bioinformatics* 9 (February): 269. https://doi.org/10.1186/1471-2105-9-269.

Schmid, Matthias, and Sergej Potapov. 2012. "A comparison of estimators to evaluate the discriminatory power of time-to-event models." *Statistics in Medicine* 31 (23): 2588–2609. https://doi.org/10.1002/sim.5464.

Schwarzer, Guido, Werner Vach, and Martin Schumacher. 2000. "On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology." *Statistics in Medicine* 19 (4): 541–61. https://doi.org/10.1002/(SICI)1097-0258(20000229)19:4%3C541::AID-SIM355%3E3.0.CO;2-V.

Segal, Mark Robert. 1988. "Regression Trees for Censored Data." *Biometrics* 44 (1): 35—–47.

Shivaswamy, Pannagadatta K., Wei Chu, and Martin Jansche. 2007. "A support vector approach to censored targets." In *Proceedings - IEEE International Conference on Data Mining, ICDM*, 655–60. https://doi.org/10.1109/ICDM.2007.93.

Sonabend, Raphael. 2020. "survivalmodels: Models for Survival Analysis." CRAN. https://raphaels1.r-universe.dev/ui#package:survivalmodels.

———. 2022. "Scoring rules in survival analysis," December. http://arxiv.org/abs/2212.05260.

Sonabend, Raphael Edward Benjamin. 2021. "A Theoretical and Methodological Framework for Machine Learning in Survival Analysis: Enabling Transparent and Accessible Predictive Modelling on Right-Censored Time-to-Event Data." PhD, University College

London (UCL). https://discovery.ucl.ac.uk/id/eprint/10129352/.

Sonabend, Raphael, Andreas Bender, and Sebastian Vollmer. 2022. "Avoiding C-hacking when evaluating survival distribution predictions with discrimination measures." Edited by Zhiyong Lu. *Bioinformatics* 38 (17): 4178–84. https://doi.org/10.1093/bioinformatics/btac451.

Sonabend, Raphael, Franz J Király, Andreas Bender, Bernd Bischl, and Michel Lang. 2021. "mlr3proba: an R package for machine learning in survival analysis." Edited by Jonathan Wren. *Bioinformatics* 37 (17): 2789–91. https://doi.org/10.1093/bioinformatics/btab039.

Sonabend, Raphael, Florian Pfisterer, Alan Mishler, Moritz Schauer, Lukas Burk, Sumantrak Mukherjee, and Sebastian Vollmer. 2022. "Flexible Group Fairness Metrics for Survival Analysis." In *DSHealth 2022 Workshop on Applied Data Science for Healthcare at KDD2022*. http://arxiv.org/abs/2206.03256.

Song, Xiao, and Xiao-Hua Zhou. 2008. "A semiparametric approach for the covariate specific ROC curve with survival outcome." *Statistica Sinica* 18 (July): 947–65.

Spooner, Annette, Emily Chen, Arcot Sowmya, Perminder Sachdev, Nicole A Kochan, Julian Trollor, and Henry Brodaty. 2020. "A comparison of machine learning methods for survival analysis of high-dimensional clinical data for dementia prediction." *Scientific Reports* 10 (1): 20410. https://doi.org/10.1038/s41598-020-77220-w.

Spruance, Spotswood L, Julia E Reid, Michael Grace, and Matthew Samore. 2004. "Hazard ratio in clinical trials." *Antimicrobial Agents and Chemotherapy* 48 (8): 2787–92. https://doi.org/10.1128/AAC.48.8.2787-2792.2004.

Therneau, Terry M. 2015. "A Package for Survival Analysis in S." https://cran.r-project.org/package=survival.

Therneau, Terry M., and Beth Atkinson. 2019. "rpart: Recursive Partitioning and Regression Trees." CRAN.

Therneau, Terry M., and Elizabeth Atkinson. 2020. "Concordance." https://cran.r-project.org/web/packages/survival/vignettes/concordance.pdf.

Therneau, Terry M., Patricia M. Grambsch, and Thomas R. Fleming. 1990. "Martingale-based residuals for survival models." *Biometrika* 77 (1): 147–60. https://doi.org/10.1093/biomet/77.1.147.

Tsiatis, A. 1975. "A nonidentifiability aspect of the problem of competing risks." *Proceedings of the National Academy of Sciences of the United States of America* 72 (1): 20–22. https://doi.org/10.1073/pnas.72.1.20.

Tsoumakas, Grigorios, and Ioannis Katakis. 2007. "Multi-Label Classification: An Overview." *International Journal of Data Warehousing and Mining* 3 (3): 1–13. https://doi.org/10.4018/jdwm.2007070101.

Tutz, Gerhard, and Harald Binder. 2007. "Boosting Ridge Regression." *Computational Statistics & Data Analysis* 51 (February): 6044–59. https://doi.org/10.1016/j.csda.2006.11.041.

Tutz, Gerhard, and Matthias Schmid. 2016. *Modeling Discrete Time-to-Event Data*. Springer Series in Statistics. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-28158-2.

Uno, Hajime, Tianxi Cai, Michael J. Pencina, Ralph B. D'Agostino, and L J Wei. 2011. "On the C-statistics for Evaluating Overall Adequacy of Risk Prediction Procedures with Censored Survival Data." *Statistics in Medicine* 30 (10): 1105–17. https://doi.org/10.1002/sim.4154.

Uno, Hajime, Tianxi Cai, Lu Tian, and L J Wei. 2007. "Evaluating Prediction Rules for t-Year Survivors with Censored Regression Models." *Journal of the American Statistical Association* 102 (478): 527–37. http://www.jstor.org/stable/27639883.

Ushey, Kevin, J J Allaire, and Yuan Tang. 2020. "reticulate: Interface to 'Python'." CRAN.

https://cran.r-project.org/package=reticulate.

Van Belle, Vanya, Kristiaan Pelckmans, Johan A. K. Suykens, and Sabine Van Huffel. 2007. "Support Vector Machines for Survival Analysis." In *In Proceedings of the Third International Conference on Computational Intelligence in Medicine and Healthcare*. 1.

Van Belle, Vanya, Kristiaan Pelckmans, Sabine Van Huffel, and Johan A. K. Suykens. 2011. "Support vector methods for survival analysis: A comparison between ranking and regression approaches." *Artificial Intelligence in Medicine* 53 (2): 107–18. https://doi.org/10.1016/j.artmed.2011.06.006.

Van Houwelingen, Hans C. 2000. "Validation, calibration, revision and combination of prognostic survival models." *Statistics in Medicine* 19 (24): 3401–15. https://doi.org/10.1002/1097-0258(20001230)19:24%3C3401::AID-SIM554%3E3.0.CO;2-2.

———. 2007. "Dynamic prediction by landmarking in event history analysis." *Scandinavian Journal of Statistics* 34 (1): 70–85. https://doi.org/10.1111/j.1467-9469.2006.00529.x.

Volinsky, Chris T, and Adrian E Raftery. 2000. "Bayesian Information Criterion for Censored Survival Models." *International Biometric Society* 56 (1): 256–62.

Wang, Hong, and Gang Li. 2017. "A Selective Review on Random Survival Forests for High Dimensional Data." *Quantitative Bio-Science* 36 (2): 85–96. https://doi.org/10.22283/qbs.2017.36.2.85.

Wang, Ping, Yan Li, and Chandan K. Reddy. 2019. "Machine Learning for Survival Analysis." *ACM Computing Surveys* 51 (6): 1–36. https://doi.org/10.1145/3214306.

Wang, Zhu. 2019. "bujar: Buckley-James Regression for Survival Data with High-Dimensional Covariates." CRAN. https://cran.r-project.org/package=bujar.

Wang, Zhu, and C Y Wang. 2010. "Buckley-James Boosting for Survival Analysis with High-Dimensional Biomarker Data." *Statistical Applications in Genetics and Molecular Biology* 9 (1). https://doi.org/https://doi.org/10.2202/1544-6115.1550.

Wei, L J. 1992. "The Accelerated Failure Time Model: A Useful Alternative to the Cox Regression Model in Survival Analysis." *Statistics in Medicine* 11: 1871–79.

Welchowski, Thomas, and Matthias Schmid. 2019. "discSurv: Discrete Time Survival Analysis." CRAN. https://cran.r-project.org/package=discSurv.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

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

Zare, Ali, Mostafa Hosseini, Mahmood Mahmoodi, Kazem Mohammad, Hojjat Zeraati, and Kourosh Holakouie Naieni. 2015. "A Comparison between Accelerated Failure-time and Cox Proportional Hazard Models in Analyzing the Survival of Gastric Cancer Patients." *Iranian Journal of Public Health* 44 (8): 1095–1102. https://doi.org/10.1007/s00606-006-0435-8.

Zhao, Lili, and Dai Feng. 2020. "Deep Neural Networks for Survival Analysis Using Pseudo Values." *IEEE Journal of Biomedical and Health Informatics* 24 (11): 3308–14. https://doi.org/10.1109/JBHI.2020.2980204.

Zhou, Zheng, Elham Rahme, Michal Abrahamowicz, and Louise Pilote. 2005. "Survival Bias Associated with Time-to-Treatment Initiation in Drug Effectiveness Evaluation: A Comparison of Methods." *American Journal of Epidemiology* 162 (10): 1016–23. https://doi.org/10.1093/aje/kwi307.

Index