Stockholms universitet

Beyond Traditional Boundaries: Harnessing the Power of Deep Learning for Enhanced Survival Analysis and Interpretability

Adam Goran

Masteruppsats i matematisk statistik
Master Thesis in Mathematical Statistics

Masteruppsats 2023:8
Matematisk statistik
Juni 2023

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Mathematical Statistics
Stockholm University
Master Thesis **2023:8**
http://www.math.su.se

Beyond Traditional Boundaries: Harnessing the Power of Deep Learning for Enhanced Survival Analysis and Interpretability

Adam Goran*

June 2023

Abstract

Survival analysis is a key statistical method for studying time-to-event data, which plays a critical role in various fields, including medicine, finance, and social sciences. A traditional and still popular model in this context is the Cox proportional hazards model. Despite its widespread use, it relies on assumptions that may not hold in real-world scenarios, leading to potential inaccuracies in predictions. This master thesis investigates the application of deep learning techniques, specifically a special type of neural network model, to overcome the limitations of the traditional Cox model and provide a more flexible approach to survival analysis.

The `DeepHit` model is a deep learning-based approach that adapts to complex relationships in the data without relying on restrictive assumptions. We compare the performance of the Cox model and the `DeepHit` model through simulation experiments and a real data experiment, demonstrating the strengths and weaknesses of each approach. The results indicate that the `DeepHit` model can outperform the Cox model in certain cases, particularly when the assumptions of the Cox model are violated.

Moreover, the thesis addresses the interpretability issue commonly associated with deep learning models by introducing SHAP (SHapley Additive exPlanation) values to approximate and explain the predictions made by the `DeepHit` model. This addition enhances the practicality of the `DeepHit` model for real-world applications, especially in medical applications, where understanding a model can be crucial.

In conclusion, this thesis advocates for the adoption of more sophisticated methods, such as deep learning techniques, in survival analysis to avoid making restrictive assumptions about the data at hand. The findings contribute valuable insights for practitioners and researchers in the field, encouraging the exploration and refinement of advanced methods for improved performance across various applications. Recent advancements in deep learning demonstrates its seemingly unlimited potential in numerous fields.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: adamgoran1999@gmail.com. Supervisor: Chun-Biu Li & Marie Linder.

Acknowledgements

This thesis constitutes a master's thesis of 30 ECTS in Mathematical Statistics at the Department of Mathematics at Stockholm University. I am sincerely grateful for the steadfast support and guidance from my supervisor at the department, Chun-Biu Li, whose valuable advice and constructive feedback have been immensely helpful. I would also like to extend my deep appreciation to the Karolinska Institute, which furnished me with insights and resources, as well as my supervisor there, Marie Linder, whose expertise and unwavering commitment have played a pivotal role in shaping the direction and scope of this thesis.

Contents

1 Introduction

Background and Motivation. Survival analysis is a branch of statistical analysis that focuses on the study of time-to-event data. It has widespread applications across various fields, including medicine, finance, engineering, and social sciences (Miller, 2011). In medical research, for instance, survival analysis is employed to evaluate the time to death or the duration until a patient experiences a specific event, such as a disease recurrence. This information is essential for understanding the effectiveness of treatments, identifying risk factors, and making informed decisions about patient care.

The Cox proportional hazards model (Cox, 1972), introduced by Sir David Cox in 1972, has been the cornerstone of survival analysis for decades. The popularity of the Cox model can be attributed to its simplicity and ease of interpretation. However, the model relies on a set of assumptions that may not always hold true in real-world scenarios. These assumptions include proportional hazards and a linear relationship between the covariates and the log hazard function. When these assumptions are violated, the Cox model may produce inaccurate predictions.

The advent of deep learning has revolutionised the field of machine learning and has shown promising results in various domains. In recent years, researchers have begun to explore the potential of deep learning techniques in survival analysis, aiming to address the limitations of traditional models like the Cox proportional hazards model (Lee, 2018). These advanced methods can adapt to complex relationships in the data without making restrictive assumptions, offering a more flexible approach to survival analysis. By comparing the traditional Cox proportional hazards model to the more advanced `DeepHit` model, we plan to demonstrate that embracing sophisticated techniques like deep learning can unlock the full potential of survival analysis and lead to improved performance across various applications.

An additional motivating factor for this thesis has been the availability and richness of the Swedish national health data registers, which serve as the primary data source for many researchers at the Karolinska Institute. These comprehensive registers, maintained and updated by the Swedish government, contain extensive information on individuals' medical history, treatment, and outcomes. With a unique personal identity number assigned to each Swedish resident, the registers enable efficient and accurate linkage of data across various sources, including patient registries, prescription registries, and cause of death registries (Ludvigsson *et al.*, 2009).

The Swedish national health data registers offer a wealth of information, enabling some of the large-scale and long-term studies conducted at the Karolinska Institute. However, the complex and high-dimensional nature of the data poses significant challenges for traditional survival analysis methods, such as the Cox proportional hazards model, which may not

adequately capture the intricate relationships among variables and is generally incapable of using all the available information. Consequently, the availability and unique characteristics of these registers provided a strong motivation for the development and application of more flexible and sophisticated methods, such as the `DeepHit` model, to better analyse and interpret the data.

Objectives and Scope. This master thesis aims to investigate the potential benefits of utilising deep learning techniques in survival analysis, with a focus on the comparison between the traditional Cox proportional hazards model and the more advanced `DeepHit` model. The main objectives of this thesis are:

- To provide a comprehensive overview of the Cox proportional hazards model, discussing its assumptions, limitations, and implications for survival analysis.
- To introduce the `DeepHit` model, a deep learning-based approach to survival analysis, and demonstrate its flexibility in estimating the survival function. Since understanding this model requires fundamental knowledge about feedforward neural networks, we provide the necessary theory for doing so.
- To compare the performance of the Cox model and the `DeepHit` model using simulation experiments and a real data experiment, evaluating their strengths and weaknesses.
- To address the issue of interpretability in the `DeepHit` model by introducing SHAP (SHapley Additive exPlanation) values, which can be used to approximate and explain the predictions made by this model.

By achieving these objectives, this thesis aims to shed light on the potential of deep learning techniques in survival analysis and advocate for the adoption of more sophisticated methods that avoid making restrictive assumptions about the data at hand. We hope to offer valuable insights and guidance for practitioners and researchers in this field.

Main Results and Implications. The main results of this thesis include:

- A comprehensive comparison of the Cox proportional hazards model and the `DeepHit` model, illustrating the strengths and weaknesses of each approach.
- Demonstration of the `DeepHit` model's superior performance in certain cases, particularly when the assumptions of the Cox model are not met, through simulation

experiments and a real data experiment. The test results suggest that the `DeepHit` model is able to provide more accurate and flexible estimations of survival functions.

- Introduction of SHAP values to address the interpretability issue associated with the `DeepHit` model, making the model more practical for real-world applications. We take a distinct approach to calculating these values, compared to what has been suggested in the references.

These results advocate for the adoption of more sophisticated methods, such as deep learning, to avoid making restrictive assumptions about the data at hand. Furthermore, they give directions for areas of further research and inspire the exploration of more deep learning in the field.

Thesis Structure. The thesis is organised as follows:

- Section 2 provides most of the relevant theory. First, we give a brief introduction to survival analysis, including the necessary concepts and characteristics for this thesis. We then introduce the Cox proportional hazards model and analyse it in detail. Further on in the section, we provide the theory of feedforward neural networks, in particular how they are defined and trained, after which a detailed description of `DeepHit` is given. Lastly, we introduce SHAP values as an interpretation tool for black box models.
- Section 3 presents the results and discussion thereof. We introduce evaluation metrics for comparison of the models, as well as relevant information regarding optimisation. The simulation procedures are described and test results are presented. Furthermore, the results and interpretation for the real dataset are outlined.
- Section 4 concludes the thesis by summarising the key findings, discussing their implications for the field of survival analysis, and suggesting future research directions.

2 Theory

2.1 Survival Analysis: Concepts, Challenges, and Techniques

Survival analysis comes with the challenge of working with clinical data. Several properties characterise clinical data, and they must be addressed carefully to ensure valid inferences and conclusions. The challenges of interpreting results produced by studies on clinical data include the following:

- Heterogeneity: Clinical data comes from a diverse population of patients, with varying demographics, medical histories, and conditions. This heterogeneity can make it challenging to draw conclusions from the data and generalise findings to other patient populations.
- Incompleteness: Missing values and incomplete records makes the data incomplete. Handling missing data can be done by excluding those patients or replacing the missing values with e.g. the mean or mode of the population, or using proxies. This showcases one of the challenges in analysing the data.
- Bias: Various biases can occur in clinical data, such as selection bias and measurement bias. For example, patients who are more likely to seek medical care may be overrepresented in the data, and this can bias estimates of disease prevalence or treatment outcomes.
- Privacy and security concerns: Data that is subject to strict privacy and security regulations to protect patient confidentiality cannot be analysed by anyone. As an implication, data used in this project cannot be shared, limiting the reproducibility and further improvement of suggested models. Even for researchers with full access to registers and data bases, there could be a lack of valuable measurements.
- Longitudinal nature: Clinical data is often collected over time, allowing for the analysis of temporal relationships between exposures and outcomes. However, it can be difficult to detect and account for variables that are dependent on time. This could for example mean that results are valid for certain time intervals but not for others.

Despite these challenges, it is essential to work with clinical data for medical research and decision-making. Another important matter of concern in clinical settings is confounding (Vittinghoff *et al.*, 2004, chapter 4). Confounding occurs when the relationship between an exposure and an outcome is distorted by the presence of a third variable, known as a confounder. In statistical terms, exposure refers to the presence or degree of a specific variable that potentially influences the probability of an outcome or event of interest. To address this issue, one has to control for the confounding variable, for example, by creating a stratified model that examines the association between other variables at different values of the confounder. Figure 1 provides a basic example.

In Figure 1, suppose we want to study the relationship between X and Y. If Z is not included in this study, we could encounter an instance of the famous Simpson's paradox. This is a situation where separate groups of observations exhibit a certain correlation, but when the groups are combined, the opposite correlation emerges. In other words, the value of Z affects both X and Y, complicating their association. Now consider the case when the arrow between Z and X is reversed. This scenario creates a mediator situation in which X affects both Y and Z, and including Z in the study influences the relationship between X

and Y. The results of such studies can be severely misleading, and controlling for Z should be avoided in this case.

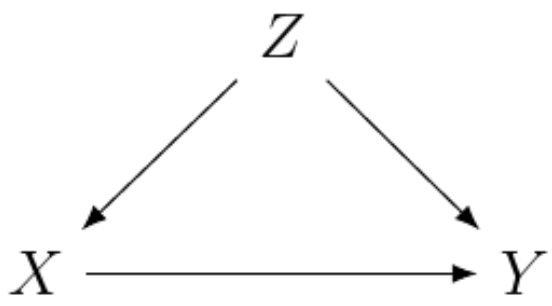

Figure 1: Graph of variables in a study and their relationships. The presence or level of X affects the outcome Y, whilst the presence or level of Z affects both X and Y. This is a situation in which confounding can occur.

Directed acyclic graphs (Digitale *et al.*, 2021) provide a rigorous yet intuitive tool for detecting and planning for confounding before conducting a study. This approach establishes causal effects by creating a graph similar to the one in Figure 1, a task that requires prior knowledge of the research field. It is important to note that the use of this tool merely provides an illustration of the relevant assumptions. We introduce the topic of confounding here to emphasise its importance as a potential pitfall in survival analysis. In many cases, deep learning models trained on large datasets can inherently account for complex relationships between variables, alleviating the need for explicit assumptions about confounders. This advantage can lead to more accurate and robust results in survival analysis when using deep learning models compared to traditional methods.

Returning to the unique aspects of survival analysis, another notable challenge in our study, which sets it apart from conventional supervised learning tasks, is the substantial presence of censored data. Censored data refers to incomplete information about an observation. Especially relevant to our study is right-censored data. This situation arises when the outcome of interest has not been observed for some individuals by the end of the study period. Patients have varying lengths of follow-up time, depending on when they received the diagnosis and when the study ends, which must be accounted for to avoid introducing serious bias in the study. Another critical consideration in the analysis of right-censored data is the assumption of non-informative censoring. This assumption states that the probability of censoring is independent of the survival time, given the covariates. If this assumption is violated, inferences may be incorrect. In summary, we must use models that are compatible with censored data.

2.1.1 The Kaplan-Meier curve

The survival function $S(t) = P(T > t)$ is the probability that a certain patient or subject survives past some time point t, or does not experience an outcome before t. Kaplan and

Meier (1958) made a major contribution to the field of survival analysis with the introduction of a method for estimating survival curves in the presence of censoring. In the absence of censoring, all observations in the dataset have time points $t_i, i = 1, \ldots, n$ at which the event was experienced. The most natural estimation of the survival curve in this case would be

$$\hat{S}(t) = \frac{1}{n} \sum_{i=1}^{n} I(t_i < t), \tag{1}$$

where $I(t_i < t)$ is the indicator function of the time point t_i being smaller than t. This ratio represents the events experienced up to time t and can be referred to as the empirical survival curve. Note that this is not a cumulative distribution function, but $1 - S(t) = F(t)$ is. To construct the Kaplan-Meier estimator, let $m < n$ be the number of observations not censored and order the event times $t_{(1)} < t_{(2)} < \cdots < t_{(m)}$. Let d_i be the number of events occurring at time $t_{(i)}$ and $d_i = 0$ for all time points before $t_{(1)}$. Furthermore, let $|R_i|$ be the number of observations at risk, i.e. the number of observations not censored or with the event already experienced, just before time $t_{(i)}$. Then, the Kaplan-Meier estimator (Kaplan & Meier, 1958) of the survival function is defined as

$$\hat{S}(t) = \prod_{i:\, t_{(i)} < t} \left(1 - \frac{d_i}{|R_i|}\right). \tag{2}$$

This estimator is intuitive in that surviving in any time interval $[t_{(i-1)}, t_{(i)})$ is equal to 1 minus the probability of dying in the same time period, which is naturally estimated by $d_i/|R_i|$. The probability of surviving past time t becomes the product of surviving all preceding time intervals. The result is a stepwise function, which becomes smoother as we have more observations with outcomes. Calculating the expected survival time is straightforward:

$$E(T) = \sum_{i=1}^{m} t_{(i)} S(t_{(i)}), \tag{3}$$

which represents the area under the curve. In summary, the Kaplan-Meier curve is fundamental and can be used for various purposes, especially when we need a basis for comparison or a starting point for analysis.

2.1.2 Cox proportional hazards model

Instead of directly modelling the survival function, the Cox proportional hazards model, also known as Cox regression, models the hazard function $\lambda(t)$. The survival function $S(t)$, relates to the hazard function via

$$\lambda(t) = -\frac{S'(t)}{S(t)} = -\frac{d}{dt}\log S(t) \qquad \Longrightarrow \qquad S(t) = \exp\left[-\int_0^t \lambda(t')dt'\right]. \tag{4}$$

In words, Vittinghoff *et al.* (2004, chapter 6) describes the hazard function as the short-term event rate of experiencing the outcome event for those who have not yet done it. The Cox proportional hazards model takes covariates x and a time point t as input and assumes the form

$$\lambda(t; x) = \lambda_0(t)\exp(\beta_1 x_1 + \ldots + \beta_p x_p) = \lambda_0(t)\exp(\beta x), \tag{5}$$

which explains why it is referred to as a regression model. Here, $\lambda(t; x)$ is interpreted as the hazard at time t for an individual with covariate values x and $\lambda_0(t)$ is the baseline hazard function. Observe that $\lambda_0(t)$ is the hazard for an individual at time t that has all covariate values equal to 0. Furthermore, the baseline hazard can be interpreted as an intercept in correspondence to other regression models by letting $\lambda_0(t) = \exp(\beta_0(t))$, although it changes with time. The model falls under the category of a generalised linear model with logarithmic link function (Sundberg, 2019, chapter 9). Also, note that each covariate has a multiplicative effect on the hazard.

Cox regression is a popular model in medical research because, in addition to being able to handle right-censored data through the baseline hazard $\lambda_0(t)$, it can easily estimate the impact (or association) of a particular covariate through the hazard ratio. Suppose, for example, that there are two treatment groups in the data, coded into a covariate as a binary 0-1 variable. Then we can hypothetically take the ratio of the hazard for an individual belonging to treatment group 1 with an individual belonging to group 0 while keeping all other covariates fixed. Let x_p denote the binary covariate not kept fixed and let $x^{(1)}$ denote the covariates of the individual in the first group and let $x^{(0)}$ be the covariates of the second individual. Then we obtain

$$\frac{\lambda(t; x^{(1)})}{\lambda(t; x^{(0)})} = \frac{\lambda_0(t)\exp(\beta x^{(1)}))}{\lambda_0(t)\exp(\beta x^{(0)}))} = \exp(\beta_1(x_1^{(1)} - x_1^{(0)}) + \ldots + \beta_p(x_p^{(1)} - x_p^{(0)})) = \exp(\beta_p). \tag{6}$$

That is, without any assumption or estimation of the baseline hazard, we can directly measure the impact of any covariate through the hazard ratio, which in this example amounts to comparing two different treatment groups in terms of how the risk of an event differs between them. However, the inference is valid only if the model assumptions are met and the fit to the data is sufficiently good. We discuss the model assumptions later on.

Model motivation and connection to survival function. The main reason for using the exponential function in the Cox proportional hazards model is to ensure that the hazard rate has the correct domain. Since the survival function has the domain $0 \leq S(t) \leq 1$, and its derivative has the domain (due to the survival function being non-increasing) $-\infty < S'(t) \leq 0$, we see that the hazard function must have the domain:

$$0 \leq \lambda(t) < \infty. \tag{7}$$

Hence, with the exponential form, the predictors can have both decreasing and increasing effects on the overall hazard rate, without making it negative, which is consistent with the domain constraints. There are many functions that achieve this, but the exponential function is chosen because it provides a straightforward interpretation of the estimated coefficients. A positive coefficient means increased hazard, and a negative coefficient means decreased hazard as the value of the corresponding predictor increases.

When the hazard rate is modelled as a proportional hazards model, the survival function takes the form

$$S(t;x) = \exp\left[-\int_0^t \lambda_0(t')\exp(\beta x)dt'\right] = \exp\left[-\exp(\beta x)\Lambda(t)\right], \tag{8}$$

where $\Lambda(t)$ is the cumulative baseline hazard rate from 0 to t. Hence, we can see that it does not factorise the same way the hazard rate does. Although a straightforward interpretation of the survival probability ratio between two individuals with non-identical covariates does not exist, it is still easy to see in which direction the survival probability goes as covariates shift. Taking the ratio of the survival probability for two different observations we obtain

$$\frac{S(t;x^{(1)})}{S(t;x^{(2)})} = \exp\left[-\Lambda(t)(\exp(\beta x^{(1)}) - \exp(\beta x^{(2)}))\right]. \tag{9}$$

This may not produce any meaningful insights the same way the hazard ratio did, but a closer look reveals that the survival probability ratio is not constant in time. Instead, it is

amplified by time. This is because the cumulative hazard $\Lambda(t)$ is non-decreasing in t. The logarithm of the preceding equation yields

$$\log S(t; x^{(1)}) - \log S(t; x^{(2)}) = \Lambda(t)(\exp(\beta x^{(2)}) - \exp(\beta x^{(1)})). \tag{10}$$

It is now clear that the survival probability ratio between two individuals with nonidentical covariates is not constant in time. Instead, the two survival curves experience increasing gaps between them the longer the time goes. Also in this form, the expression factorises into one part depending only on time and a second part depending only on the covariates. Extracting the linear regression form from the survival function requires the following transformation:

$$\log\left(\frac{-\log S(t; x)}{\Lambda(t)}\right) = \beta_1 x_1 + \ldots + \beta_p x_p, \tag{11}$$

which is rather complicated and difficult to interpret. Nonetheless, the Cox proportional hazards model provides a valuable way to model survival data through the hazard function.

Model assumptions and limitations. The primary assumption of the Cox proportional hazards model is that hazards for different observations or groups are proportional, meaning the hazard ratio between two observations is constant over time. This assumption arises from the functional form of the baseline hazard. While this is a strong assumption, it can be relaxed by creating a stratified model, described in Vittinghoff *et al.* (2004, chapter 6), which allows for different baseline hazards for different levels of a particular covariate or groups. However, the proportionality assumption still holds within each stratum and can be difficult to fully circumvent. Additionally, using strata for one covariate removes the same covariate and its associated beta-parameter from the model, which may not be ideal if this covariate is of primary interest. An alternative approach is to allow for time-varying covariates.

The interpretation of a Cox regression model when the proportionality assumption is violated can be controversial. However, it has been shown that meaningful conclusions can still be drawn from such models, which explains why the model remains popular among practitioners.

Survival models, in general, assume independence between censoring and survival time, as well as between the censoring mechanism and the covariates. In other words, the probability of an observation being censored is independent of both the survival time and the covariates. This assumption can be questioned in cases where certain individuals are more likely to drop out from a study, for example.

Although Cox regression offers some flexibility when estimating the baseline hazard, it is a linear model in its covariate parameters and does not account for interactions between covariates unless explicitly including such variables. This limitation is a key reason for exploring machine learning models, which can offer greater flexibility and the ability to capture more complex relationships in the data.

Coefficient estimation. To derive maximum likelihood estimates in survival analysis models in general, different contributions are made to the likelihood function depending on whether the observation is censored, asserted in Cox (1972). Before dealing with the intricacies of the likelihood theory for Cox regression, we present the following realisation, which will become relevant in developing machine learning models as well. Let $F(t) = 1 - S(t) = P(T \leq t)$, then the probability density function becomes $F'(t) = f(t)$. On the other hand,

$$S'(t) = \frac{d}{dt} \exp\left[-\int_0^t \lambda(t')dt'\right] = -\lambda(t) \exp\left[-\int_0^t \lambda(t')dt'\right] = -S(t)\lambda(t), \tag{12}$$

which implies $f(t) = S(t)\lambda(t)$. Now if subject i is uncensored with observation time t_i, it contributes to the likelihood through $f(t_i) = S(t_i)\lambda(t_i)$. Otherwise if the subject is right-censored after being observed for a time t_i, we only know $P(T_i > t_i) = S(t_i)$, which is a contribution containing less information. Furthermore, denote by δ_i an indicator taking value 1 if subject i is uncensored and 0 otherwise. Also, denote by θ all model parameters of the survival function we aim to estimate with maximum likelihood. The general likelihood function becomes

$$L(\theta; t) = \prod_{i=1}^{n} \lambda(t_i; \theta)^{\delta_i} S(t_i; \theta). \tag{13}$$

Probably the most simple example is when assuming data from an exponential distribution, since then $\lambda(t_i; \theta) = \theta$ and $S(t_i; \theta) = e^{-\theta t_i}$. By taking the log-likelihood function we get after some basic calculations the maximum likelihood estimate

$$\hat{\theta} = \frac{\sum_{i=1}^{n} \delta_i}{\sum_{i=1}^{n} t_i}, \tag{14}$$

which is simply the total number of observed outcomes divided by the total observation time across all subjects. In the special case of Cox regression, we can view the estimation

of the baseline hazard and effect parameters as separate processes. We do not need (13) for these tasks, although it will prove to be highly relevant later on. To estimate the effect parameters β of a Cox regression model, we first consider the probability

$$P(\text{subject } i \text{ experiences the outcome at } t_i | \text{an outcome is observed at } t_i). \tag{15}$$

This probability conditions on the time instances at which outcomes are observed, and we aim to estimate it. By Bayes's rule and further conditioning on the subjects at risk at t_i, the probability above equals

$$\frac{P(\text{subject } i \text{ experiences the outcome at } t_i)}{P(\text{an outcome is observed at } t_i)} = \tag{16}$$

$$= \frac{P(\text{subject } i \text{ experiences the outcome at } t_i | \text{subject } i \text{ is at risk at } t_i)}{P(\text{an outcome is observed at } t_i | \text{the subject is at risk at } t_i)}. \tag{17}$$

Note that the numerator in the last equation above is the hazard rate at time t_i for subject i and the denominator is the sum of corresponding hazards for all subjects at risk at t_i. Hence, we obtain

$$\frac{\lambda(t_i; x^{(i)})}{\sum_{j \in R_i} \lambda(t_i; x^{(j)})} = \frac{\lambda_0(t_i) \exp(\beta x^{(i)})}{\sum_{j \in R_i} \lambda_0(t_i) \exp(\beta x^{(j)})} = \frac{\exp(\beta x^{(i)})}{\sum_{j \in R_i} \exp(\beta x^{(j)})}, \tag{18}$$

where R_i is the subset of subjects at risk just before time t_i. Further details are referred to Cox (1972). Since the preceding only holds for uncensored observations, the likelihood function used to derive maximum likelihood estimates is obtained by taking the product of (18) over the index i, ranging from 1 to the number of observed events m. The derivative of the log-likelihood with respect to the parameter β_k then becomes

$$\frac{\partial}{\partial \beta_k} l(\beta) = \sum_{i=1}^{m} x_k^{(i)} - \sum_{i=1}^{m} \frac{\sum_{j \in R_i} x_k^{(j)} \exp(\beta x^{(j)})}{\sum_{j \in R_i} \exp(\beta x^{(j)})}. \tag{19}$$

Solving $\frac{\partial}{\partial \beta_k} l(\beta) = 0$ for $k = 1, \ldots, p$ has to be performed numerically, which is usually carried out through the Newton-Raphson method.

An issue appears in solving the maximum likelihood estimation in (19) if there are ties in the data set, i.e. if some time points $t_i, i = 1, \ldots, m$ are equal. Essentially, it can cause the

likelihood function to become discontinuous when the values of the covariates differ. This can be explained by how the probability in (18) was defined. The denominator represents the risk set R_i just before time t_i, which includes all individuals who have not yet experienced the event at time t_i. Now, if we have ties in our data, meaning multiple events occur at the same time t_i, it is unclear which individuals should be included in the risk set R_i for that time point. To resolve the problem, one can add a small random number to the tied times.

The unit of the measured time needs to be defined before conducting a study and it is important to be consistent with this unit. Units for t can for example be minutes, days or years.

Standard approaches like Wald's test, the score test, and the likelihood ratio test can be used to perform hypothesis tests on the parameters, determining if they significantly differ from 0 and have a significant effect on the outcome.

Testing the validity of the proportional hazards assumption. The proportional hazards assumption is a critical requirement for the Cox model, and its validity can be assessed using both informal and formal methods. Informal approaches involve visual inspection of residual plots against time, similar to assessing linear regression model residuals.

On a more formal note, the hypothesis of proportional hazards can be tested on both individual covariate and global levels. Here, the null hypothesis is the existence of proportional hazards, while the alternative hypothesis suggests non-proportional hazards concerning time. One widely used technique for such testing involves Schoenfeld residuals (see e.g. Grambsch & Therneau, 1994).

The Schoenfeld residual for the kth covariate of the subject with the ith event time is denoted as $r_k^{(i)}$ and is calculated as

$$r_k^{(i)} = x_k^{(i)} - \sum_{j \in R_i} x_k^{(j)} \cdot \frac{\exp(\hat{\beta}_k x_k^{(j)})}{\sum_{l \in R_i} \exp(\hat{\beta}_k x_k^{(l)})}. \tag{20}$$

Here, R_i is defined as in (18). Note that these are true residuals as their expected value is zero under the model.

To test the proportional hazards assumption, we create a multivariate linear regression model with residuals r_k as explanatory variables. If the response variable is time, the residuals should not exhibit a significant effect. Covariates producing significant p-values violate the proportionality assumption. A chi-squared test can further assess the global level as-

sumption.

For formal testing, let $\tau_1, \tau_2, \ldots, \tau_d$ denote the d observed death times across the dataset. The test statistic of Grambsch and Therneau (1994) is given by

$$\frac{\left(\sum_{i=1}^{d}(\tau_i - \bar{\tau})r_k^{*(i)}\right)^2}{d \cdot \mathrm{var}(\hat{\beta}_k)\sum_{i=1}^{d}(\tau_i - \bar{\tau})^2}, \tag{21}$$

where $r_k^{*(i)}$ are the scaled Schoenfeld residuals for variable k, and $\bar{\tau}$ is the mean of observed death times. This test statistic follows a χ^2 distribution under the null hypothesis with 1 degree of freedom.

A global test of proportionality can also be performed using the test statistic

$$\frac{(\tau - \bar{\tau})^T S\,\mathrm{var}(\hat{\beta})S^T(\tau - \bar{\tau})}{d\sum_{i=1}^{d}(\tau_i - \bar{\tau})^2}, \tag{22}$$

where S is the matrix of unscaled Schoenfeld residuals, and $\mathrm{var}(\hat{\beta})$ is the covariance matrix of the estimated coefficients. This statistic, too, follows a χ^2 distribution, albeit with p degrees of freedom, where p is the number of covariates in the model. The variance appears in the numerator due to the use of unscaled residuals. For an in-depth understanding and further details, Grambsch and Therneau (1994) is a recommended reference.

Estimation of the baseline hazard function Although the primary focus of many clinical studies is not the estimation of the baseline hazard function, it is essential for computing survival function estimates. Cox and Oakes (1984, Ch. 7.8) proposed the standard approach for estimating the hazard at a given time point. This approach, widely implemented in statistical software, is expressed as

$$\hat{\lambda}_0(t^{(i)}) = \frac{d_i}{\sum_{j \in R_i}\exp(\hat{\beta}x^{(j)})}, \tag{23}$$

where d_i signifies the number of events at time $t^{(i)}$, and R_i denotes the indices of subjects at risk just before $t^{(i)}$. This formula arises from the partial likelihood estimation method used in the Cox model. The idea is that the hazard of the event for a subject who experiences the event at time $t^{(i)}$ is proportional to the sum of the hazards for all subjects who are

at risk at that time. The proportionality constant is d_i, the number of events at time $t^{(i)}$, and the sum of the hazards for all subjects at risk is given by $\sum_{j \in R_i} \exp(\hat{\beta} x^{(j)})$. Therefore, the estimate of the baseline hazard function at time $t^{(i)}$ is given by the ratio of these two quantities.

The estimated hazard rate becomes zero at all time points lacking events. Nevertheless, the survival function formula only requires the cumulative hazard, estimated as

$$\hat{\Lambda}_0(t) = \sum_{j:\, t^{(j)} \leq t} \hat{\lambda}_0(t^{(j)}). \tag{24}$$

One of the key features of this estimator is its non-parametric nature. It necessitates the coefficients $\hat{\beta}$, which are derived from a Cox regression model. Often, this estimator is known as the Breslow estimator. Some also refer to it as the Nelson-Aalen estimator, although this technically pertains to a slightly different method for estimating the cumulative hazard. The original Nelson-Aalen estimator would replace the sum of exponentiated regression values in equation (23) with the number of subjects at risk, $|R_i|$. The introduced modifications in the proposed estimator can be interpreted as incorporating the attributes of the subjects at risk, analogous to assigning different weights to various subjects.

2.2 Feedforward neural networks

This section provides a basic description of how feedforward neural networks are built and explores details, gradually increasing the particularity. Feedforward neural networks consist of nodes, or neurons, and connections between them. The neurons are grouped into different layers and connections only exist from one layer to another, as the name suggests, in one direction. Figure 2 illustrates a shallow and basic feedforward network, along with an illustration of the feedforward structure.

To build a feedforward neural network more specifically, following the description given by Bishop (2006, chapter 5), suppose that there are p input variables $x = (x_1, x_2, \ldots, x_p)^T$. These form the first layer. The second layer, also referred to as the first hidden layer $z^{(1)}$, is obtained after combining and transforming the inputs from the previous layer, which can be written compactly as $z^{(1)} = h^{(1)}(W^{(1)}x + b^{(1)})$. Here, $h^{(1)}(\cdot)$ is an element-wise applied function called an activation function, $W^{(1)}$ is a matrix of parameters (or weights) of dimension $d^{(1)} \times p$ and $b^{(1)}$ is a vector of $d^{(1)}$ constant parameters referred to as biases. In Figure 2, the biases are omitted in the network architecture for simplicity, but can be thought of as inherent in the transformation functions. The magnitude of $d^{(1)}$ determines the number of components of $z^{(1)}$ and hence the number of neurons in the first hidden layer.

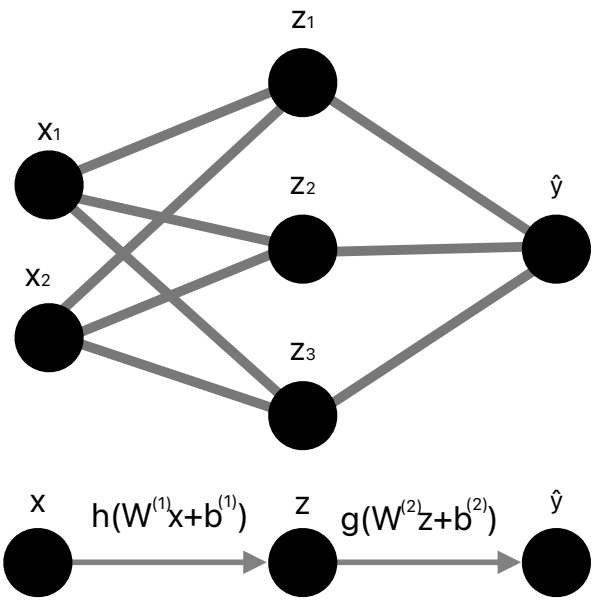

Figure 2: Example of a feedforward neural network. The top drawing shows the architecture of how inputs are connected to the output through a middle layer of higher dimension. Nodes are represented as black dots and connections as grey lines. The bottom drawing outlines the information flow for the network. Information flows from the input layer, through transformations, to the hidden layer, which passes through another transformation to produce the output.

The subsequent layers follow the same structure, i.e. for k hidden layers we have

$$\begin{aligned} z^{(2)} &= h^{(2)}(W^{(2)}z^{(1)} + b^{(2)}), \\ z^{(3)} &= h^{(3)}(W^{(3)}z^{(2)} + b^{(3)}), \\ &\vdots \\ z^{(k)} &= h^{(k)}(W^{(k)}z^{(k-1)} + b^{(k)}), \\ \hat{y} &= g(W^{(k+1)}z^{(k)} + b^{(k+1)}). \end{aligned} \tag{25}$$

The activation functions $h^{(1)}, h^{(2)}, \ldots, h^{(k)}$ are usually but not necessarily the same, but $g(\cdot)$ should be appropriately chosen for the task at hand. The last layer is called the output layer and in case $\hat{y}$ is scalar it can be written as

$$\hat{y} = g(b^{(k+1)} + \sum_{i}^{d^{(k)}} W_i^{(k+1)} z^{(k)}). \tag{26}$$

In general, as for the choice of $g(\cdot)$, the specific task at hand should govern the number of output units, i.e. the length of $\hat{y}$. We can read out from the sequence of equations above that if the activation functions are all the identity function, then the model becomes linear in both its inputs and parameters. Hence, the activation functions are crucial in making the model able to capture nonlinear patterns. Common activation functions $h(\cdot)$ include the hyperbolic tangent, the logistic sigmoid function and the rectified linear unit (ReLU), defined respectively as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \sigma(x) = \frac{1}{1 + e^{-x}}, \quad \mathrm{ReLU}(x) = \max(0, x). \tag{27}$$

In the supervised learning setting, the output unit function $g(\cdot)$ could for example be the identity function if we are dealing with a regression problem. The fact that the logistic sigmoid function ranges between 0 and 1 makes it a suitable choice for $g(\cdot)$ in binary classification problems, since if the output unit is univariate the model can predict the respective probabilities of belonging to the two classes.

2.2.1 How feedforward neural networks learn

Similarly to other machine learning methods, feedforward neural networks use a gradient-based learning approach (Goodfellow *et al.*, 2016, chapter 6). This means that the parameters of the network are learned using the gradient of some pre-specified cost (or loss) function that measures the distance between the true and predicted values. For convenience, let θ denote a collection of all the parameters in the neural network model, i.e. all the weights $W^{(j)}$ and biases $b^{(j)}$ for $j = 1, 2, \ldots, k, k+1$ in a network with k hidden layers. The cost function takes the general form

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}(x_i; \theta), y_i), \tag{28}$$

where we made the model's dependency on x and θ explicit. Before elaborating on this, we describe how the cost function arises naturally when neural networks adopt maximum likelihood theory. Since our model is parametric, it can be used to define a distribution $p(y|x; \theta)$

over our response variable y. The distribution then determines the cost function, which is the negative log-likelihood of observing our training data under the defined distribution. In other words, assuming there are n training observations, we wish to minimise

$$J(\theta) = -\sum_{i=1}^{n} \log p(y_i|x_i;\theta) \tag{29}$$

with respect to θ. The index i here refers to the data observation index and not the entry of the vector if y or x are vectors. As an illustration, consider the regression problem where we assume that the likelihood takes the form of a normal distribution. In this case the model tries to estimate the mean of the normal distribution and the cost function $J(\theta)$ becomes proportional to the well-known mean squared error cost

$$J(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}(x_i;\theta))^2. \tag{30}$$

If we instead consider the binary classification setting, the underlying distribution is assumed to be the Bernoulli distribution and the network attempts to estimate the single probability parameter in the model, yielding the cross-entropy cost

$$J(\theta) = -\sum_{i=1}^{n} (y_i \log \hat{y}(x_i;\theta) + (1 - y_i) \log(1 - \hat{y}(x_i;\theta))) \tag{31}$$

to be minimised. After designing a network architecture and specifying the cost function, the next issue we encounter is how to calculate gradients, given the complicated structure of neural networks. Gradient descent is an iterative approach for updating the parameters of a model and is needed to train neural networks because solving the optimisation problem

$$\min J(\theta) \quad \text{w.r.t. } \theta \tag{32}$$

is typically not analytically possible when the cost function is nonlinear in θ. As a consequence, we cannot always expect to find a global minima, but the hope is to find a sufficiently good local minima, which we seek to realise through several modifications to be discussed later on. Anyhow, in order to descend the cost into a minima of some sort requires us to calculate gradients and update

$$\theta \leftarrow \theta - \epsilon \frac{\partial J(\theta)}{\partial \theta}, \tag{33}$$

where ϵ is a constant we discuss in more detail in the following pages.

2.2.2 Back-propagation

The computations are done through a method called back-propagation. It merely uses the chain rule of calculus to obtain the desired gradients. This is analytically straightforward and will now be illustrated in a simple one-hidden layer neural network with one input unit and hidden unit, with ReLU as activation function for the hidden layer and the sigmoid output unit function. Following the notation from before and adopting the cross-entropy cost we obtain using eq. (31)

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial W^{(2)}} = \frac{\hat{y} - y}{\hat{y} - \hat{y}^2} \frac{e^{-(W^{(2)} z^{(1)} + b^{(2)})}}{(1 + e^{-(W^{(2)} z^{(1)} + b^{(2)})})^2} z^{(1)}. \tag{34}$$

Similarly the we have

$$\frac{\partial J}{\partial b^{(2)}} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b^{(2)}} = \frac{\hat{y} - y}{\hat{y} - \hat{y}^2} \frac{e^{-(W^{(2)} z^{(1)} + b^{(2)})}}{(1 + e^{-(W^{(2)} z^{(1)} + b^{(2)})})^2}. \tag{35}$$

To derive the gradients for the parameters of the first layer we have to use the chain rule repeatedly, yielding

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial W^{(1)}}. \tag{36}$$

The gradient w.r.t. $b^{(1)}$ is evaluated similarly. It is not hard, but somewhat tedious to generalise this example to the case when $W^{(1)}$ is not scalar but a vector or matrix. To calculate $\frac{\partial \hat{y}}{\partial z^{(1)}}$ above we need to adopt a convention for differentiating the ReLU function, since it is not continuous at 0. If $f(x) = \text{ReLU}(x) = \max(0, x)$ the convention used in practice is $f'(x) = 0$ if $x \leq 0$ and $f'(x) = 1$ if $x > 0$. In the simple illustration for calculating the gradients we took the input x and output $\hat{y}$ to be general, but as $J(\theta)$ is typically a sum over several data points, we obtain a sum of gradients when considering more than one data point.

2.2.3 Stochastic gradient descent

The way we defined the cost function $J(\theta)$ has an equivalent representation up to a scaling factor in

$$J(\theta) = -E_{x,y\sim\hat{p}} \log p(y|x;\theta), \tag{37}$$

i.e. as an expectation over the empirical data distribution $\hat{p}$. This implies that calculating the gradients of the cost function for all observations in the training set is equivalent (again up to a scaling factor) to calculating the expected value

$$\frac{\partial J(\theta)}{\partial \theta} = -E_{x,y\sim\hat{p}} \frac{\partial \log p(y|x;\theta)}{\partial \theta}. \tag{38}$$

With these representations, it is clear what we are trying to do: compute expectations. To do this exactly, we need to evaluate the cost for all observations, meaning that we need to pass all observations through the network and back-propagate to find the desired gradients using all observations. This can become computationally expensive, especially with a large number of observations. To address this issue, we use the popular optimisation procedure of stochastic gradient descent. This involves sampling a number of observations and running an iteration of the training algorithm using these observations only.

Basic probability theory tells us that no matter how many samples we use, the sample mean estimates of the expected cost and expected gradient will be unbiased, but increasing the sample size increases the accuracy of the estimate with less than linear return. Furthermore, there are often observations in the data set that are very similar to each other, making it redundant to use all of them for learning. The samples should be drawn without replacement and the size of sample, also called mini batch, is left for the user to decide. Batch size is one of many hyperparameters for neural networks and could for example be 8, 16, 32 or 64, depending on the circumstances specific to the problem at hand. When we are done with updating the parameters using one sample we collect a new mini batch of the same size and repeat. After every observation has made an impact, we are done with one epoch. The number of epochs regulates how long we train the network and is yet another hyperparameter, unless techniques to let the training progress decide for itself are used, such as early stopping (Goodfellow *et al.* 2016, chapter 7).

Before concluding this subsection and presenting the final optimisation procedure we use for training, there is one more aspect to address. When updating the parameters of the model, we multiply the computed gradient estimate with a decaying factor ϵ_k referred to as the learning rate, where k indicates the iteration step. This parameter is an essential tool

for ensuring the success of the optimisation process for several reasons. It slows down the learning process and in fact, slow learning is a common theme in well performing machine learning methods and might be one of the reasons for their success. One reason could be to ensure convergence of stochastic gradient descent, which unlike regular gradient descent is not guaranteed due to the noise introduced by the random sampling. The sufficient condition on ϵ_k for convergence (Robbins & Monro, 1951) is that ϵ_k should be decreasing and satisfy

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty. \tag{39}$$

These conditions ensure a delicate balance between making updates large enough to allow the algorithm to reach the solution, while also being small enough to ensure convergence. The sequence of updates ϵ_k should be decreasing over time, to prevent large jumps around the solution, and to refine the estimate gradually. The sum of the ϵ_k should be infinite, guaranteeing that the algorithm can make enough total change to reach the solution from any initial estimate. Lastly, the sum of the squares of the ϵ_k should be finite, controlling the variability of the sequence of updates and preventing excessively large jumps, thus facilitating convergence to the solution.

2.2.4 Adam: adaptive moment estimation

In choosing the learning rate, we will adopt the modern method in which the learning rate adapts itself according to the training progress. Specifically, we will use the **Adam** optimisation algorithm (Kingma & Ba, 2014) that makes use of adaptive moments. Adaptive moments are a type of stochastic gradient descent (SGD) optimisation method that computes individual adaptive learning rates for each parameter. In other words, it adapts the learning rate for each parameter based on the historical gradient information. **Adam** uses both the first and second moments of the gradient to update the parameters. The first moment is the exponential moving average of the gradient, while the second moment is the exponential moving average of the squared gradient. **Adam** then updates the parameters by combining these two moments and scaling the learning rate accordingly. This makes **Adam** more efficient than other SGD optimisation methods since it adapts the learning rate for each parameter, thereby avoiding the need to manually tune the learning rate. Algorithm 1 summarises the **Adam** optimisation algorithm.

Algorithm 1 can be stopped either by limiting the number of epochs or by checking if convergence has been achieved, meaning that $\epsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$ approaches 0. Kingma and Ba (2014) provide a convergence analysis for this algorithm.

Algorithm 1 Adam optimisation algorithm

1. **Input parameters:**
   - Step size ϵ (default suggested: 0.001)
   - $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates (defaults suggested: 0.9 and 0.999 respectively)
   - δ : Small constant to ensure numerical stability (default suggested: 10^{-8})
   - m : Mini batch size to be used
   - θ : initial network parameters
2. **Initialise**:
   - $s = 0$: 1st moment vector
   - $r = 0$: 2nd moment vector
   - $t = 0$: time step
3. **While** Stopping criterion not met
   (a) Shuffle training data into batches of size m, leaving the last batch to possibly be smaller. This gives B mini batches, say
   (b) **For** i in $1, 2, \ldots, B$
   - Use data in mini batch i
   - Compute gradients $g = \frac{1}{m} \sum_{j=1}^{m} \frac{\partial}{\partial \theta} L(\hat{y}(x_j; \theta), y_j)$
   - Update time step $t = t + 1$
   - Update biased first moment estimate $s = \beta_1 s + (1 - \beta_1) g$
   - Update biased second raw moment estimate $r = \beta_2 r + (1 - \beta_2) g \odot g$
   - Compute bias-corrected first moment estimate $\hat{s} = \frac{s}{1-\beta_1^t}$
   - Compute bias-corrected second raw moment estimate $\hat{r} = \frac{r}{1-\beta_2^t}$
   - Update parameters $\theta = \theta - \epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$ (operations are applied element-wise)

The algorithm computes moving averages, or moments, of the gradient s and the squared gradient r using exponential decay rates controlled by the hyperparameters β_1 and β_2. These averages represent the mean and uncentered variance of the gradient, respectively. The first moment helps in identifying the general direction in which the parameters should be updated. The second moment helps in assessing the variability or noise in the gradients. By considering both these moments, Adam can adaptively adjust the learning rate for each parameter, allowing for more efficient and robust optimisation. However, since these averages are initialised as zeros, there is a bias towards zero, especially at the beginning and with low decay rates (close to 1). To correct this bias, bias-corrected estimates $\hat{s}$ and $\hat{r}$ are computed.

To see why bias correction is necessary, consider the recursive formula for the first moment estimate $s^{(t)}$ given by

$$s^{(t)} = (1 - \beta_1) \sum_{i=1}^{t} \beta_1^{t-i} g^{(i)}, \tag{40}$$

where $g^{(t)}$ is the gradient computed at time step t in the algorithm. The expected value of $s^{(t)}$ can be written as

$$E(s^{(t)}) = E\left((1 - \beta_1) \sum_{i=1}^{t} \beta_1^{t-i} g^{(i)}\right) = E(g^{(t)})(1-\beta_1) \sum_{i=1}^{t} \beta_1^{t-i} + c = E(g^{(t)})(1-\beta_1^t) + c. \tag{41}$$

where c is an error term that comes from approximating all gradients by the current gradient $g^{(t)}$. If the true first moment $E(g^{(t)})$ is stationary, then $c = 0$, but if not, c can be kept small by appropriately choosing the decay rate β_1. This is because small weights are assigned to gradients in the far past for the exponential moving average. Dividing by the factor $(1 - \beta_1^t)$ corrects for this bias. The case for the second moment estimate r is similar.

The step size taken in updating the parameters at time step t, assuming $\delta = 0$, is

$$\Delta_t = \epsilon \frac{\hat{s}}{\sqrt{\hat{r}}}. \tag{42}$$

The step size has two upper bounds

$$|\Delta_t| \leq \epsilon \frac{1-\beta_1}{\sqrt{1-\beta_2}} \quad \text{if } (1-\beta_1) > \sqrt{1-\beta_2}, \tag{43}$$

$$|\Delta_t| \leq \epsilon \quad \text{if } (1-\beta_1) \leq \sqrt{1-\beta_2}. \tag{44}$$

For the suggested defaults of β_1 and β_2, we get $0.1 > \sqrt{0.001} \approx 0.03$, so the first upper bound holds, even though $|\Delta_t|$ is still smaller than the second bound in most cases. Another property is the invariance to scaling of the gradients. For example, multiply g by m, which makes the average gradient of the loss in Algorithm 1 simply a sum. Then we have

$$\Delta_t = \epsilon \frac{m\hat{s}}{\sqrt{m^2\hat{r}}} = \epsilon \frac{\hat{s}}{\sqrt{\hat{r}}}, \tag{45}$$

which is easily seen from the alternative representation (40).

2.2.5 Parameter initialisation

One of the input parameters in Algorithm 1 is θ, the initial parameters of the network. Although not a hyperparameter in the strictest sense, it can significantly impact the performance of the model in practice. The bias parameters are typically initialised to 0 (Goodfellow *et al.*, 2014, chapter 8), while the weights are initialised by sampling from a normal or uniform distribution with a mean of 0. Various sophisticated methods exist for choosing the variance of these distributions; for instance, the popular Xavier initialisation employs the following normal distribution:

$$W^{(l)} \sim N(\mu = 0, \sigma^2 = 1/d^{(l-1)}), \tag{46}$$

where $d^{(l-1)}$ is the number of neurons in layer $l-1$ ($l = 0$ is the input layer). To demonstrate the importance of weight initialisation, consider a neural network with a two-dimensional input and a hidden layer with two nodes. Let the initial biases be 0, and all weights be set to some constant c. During the first pass of x_1 and x_2 through the network, we obtain $z_1^{(1)} = z_1^{(1)}(cx_1+cx_2)$ and $z_2^{(1)} = z_2^{(1)}(cx_1+cx_2)$, meaning both hidden units will have the same value. Consequently, the hidden units will influence the cost in the same way and have identical gradients. Both neurons will learn the same information, as they receive identical updates throughout the training process. Failing to use a proper weight initialisation technique can jeopardise the training for other reasons not addressed here.

Relating to the weight initialisation procedure is the vanishing gradients problem. This issue is particularly prevalent in deep neural networks, where the gradients of the loss function with respect to the weights become progressively smaller as we move backward through the layers during backpropagation. As a result, the weights in earlier layers receive smaller updates, which can significantly slow down the learning process and potentially lead to suboptimal model performance. Residual connections, which for instance adds connections from the input to the deeper layers, skipping the layers in between, can resolve vanishing gradients problem.

Most issues and successes of neural networks can be traced back to the cost function. Since the cost function is nonlinear, it can have several local minima, meaning that the optimisation problem is non-convex. Finding an optimal solution for a non-convex problem is known to be NP-hard, which implies that there is no known algorithm that can find the global minimum in polynomial time. In practice, this means for instance that:

1. The optimisation process is more likely to converge to local minima or plateau points, rather than the global minimum, potentially leading to suboptimal model performance.
2. The choice of initial parameter values and optimisation algorithms can have a significant impact on the quality of the final solution, as they may determine which local minimum the optimisation process converges to.
3. It can be challenging to determine whether a solution is truly optimal, as there may be other local minima with similar performance characteristics that have not been explored by the optimisation process.

Despite these challenges, deep learning models have been shown to achieve remarkable performance in various applications, often outperforming traditional methods. This success can be attributed to factors such as the use of advanced optimisation algorithms (e.g., Adam), better weight initialisation techniques, and the ability of deep networks to learn complex hierarchical representations, which can help mitigate the issues associated with non-convex optimisation problems.

2.3 Neural network for survival analysis: `DeepHit`

Some other suggested models. Several neural network implementations have been proposed to handle survival data. Advances have been made in recent times due to the flexibility and popularity of neural networks. Among the notable contributions is the implementation `DeepSurv` introduced by Katzman *et al.* (2018). Recalling the Cox proportional

hazards model $\lambda(t;x) = \lambda_0(t)\exp(\beta x)$, the aim of `DeepSurv` is to abandon the linearity assumption and produce a hazard function in the form of:

$$\lambda(t;x) = \lambda_0(t)\exp(g(x)), \tag{47}$$

where $g(x)$ is a nonlinear neural network capable of modelling complex patterns and interactions. To train such a network, one uses the usual Cox likelihood as the loss function, and the network outputs a scalar $g(x)$. Although this model has intriguing applications, such as personalised treatment recommendations, it has the evident drawback of still assuming proportional hazards. The model might not perform well if this assumption is not fulfilled by the data.

Another method, introduced by Hu *et al.* (2021), employs the Transformer to estimate the hazard function on a grid of time points. This approach is somewhat more complicated than a regular feedforward neural network, so it will not be described in more detail here. It may prove to be a high-performing implementation in the future, but the concept of the Transformer is quite recent and not extensively exploited, especially in survival analysis. It shares many similarities with the model we are going to consider, one of which is that it is a discrete-time model.

Discrete-time neural network models for survival analysis If we let the random survival time T^* be a discrete variable instead of continuous, a new framework has to be established. We do this following Kvamme and Borgan (2019). The main purpose for discretisation is that function estimation becomes easier, at least for the kind of neural networks considered here. Also, the data provided in most scenarios measures time on a discrete scale, e.g. days or years. Measuring T^* on a discrete scale means that $T \in \{\tau_1, \tau_2, \ldots\}$, with $\tau_1 < \tau_2 \ldots$ and define for convenience $\tau_0 = 0 < \tau_1$. The probability mass function and survival function are similarly defined as

$$\begin{aligned} f(\tau_j) &= P(T^* = \tau_j), \quad P(T^* = \tau_0) = 0, \\ S(\tau_j) &= P(T^* > \tau_j) = \sum_{k>j} f(\tau_k), \quad j = 1, 2, \ldots \end{aligned} \tag{48}$$

respectively. The hazard function can be obtained directly from the survival function via

$$\lambda(\tau_j) = P(T^* = \tau_j | T^* > \tau_{j-1}) = \frac{f(\tau_j)}{S(\tau_{j-1})} = \frac{S(\tau_{j-1}) - S(\tau_j)}{S(\tau_{j-1})}, \tag{49}$$

where we used Bayes' rule. Note that this implies, unlike in the continuous setting, that $0 < \lambda(\tau_j) < 1$, since $S(\tau_j) < S(\tau_{j-1})$. Furthermore, to allow for right-censoring, let $C^* \in \{\tau_1, \tau_2, \ldots, \tau_m\}$ be a random variable denoting the censoring time. Similarly as for the survival time, this variable has a PMF and survival function corresponding to

$$f_{C^*}(\tau_j) = P(C^* = \tau_j), \quad P(C^* = \tau_0) = 0, \tag{50}$$

$$S_{C^*}(\tau_j) = P(C^* > \tau_j) = \sum_{k>j} f_{C^*}(\tau_k), \quad j = 1, \ldots, m, \tag{51}$$

respectively. Note that the discrete time scale is the same for C^* as for T^*, with the exception of a maximum time point τ_m on its domain. This maximum time corresponds for example to the length of a study. In other situations, τ_m could be unknown. The issue now is that in reality, we do neither observe T^* nor C^* directly, but instead

$$T = \min\{T^*, C^*\}, \tag{52}$$

$$D = I(T^* \leq C^*), \tag{53}$$

where D is the event indicator. Since $C^* \leq \tau_m$, we cannot observe $T^* > \tau_m$, meaning that we are restricted to model event times distributed on $\{\tau_1, \tau_2, \ldots, \tau_m\}$. If the maximum censoring time is unknown, we take τ_m practically to be the largest survival time observed. Assuming independence between censoring times and survival times T^*, as per usual for survival analysis, we retrieve

$$\begin{aligned}
P(T = t, D = \delta) &= P(T^* = t, C^* \geq t)^{\delta} P(T^* > t, C^* = t)^{1-\delta} && (54)\\
&= [P(T^* = t)P(C^* \geq t)]^{\delta}[P(T^* > t)P(C^* = t)]^{1-\delta} && (55)\\
&= [f(t)(S_{C^*}(t) + f_{C^*}(t))]^{\delta}[S(t)f_{C^*}(t)]^{1-\delta} && (56)\\
&= \left[f(t)^{\delta} S(t)^{1-\delta}\right]\left[f_{C^*}(t)^{1-\delta}(S_{C^*}(t) + f_{C^*}(t))^{\delta}\right]. && (57)
\end{aligned}$$

It follows that the two distributions separate, so given that $f(t)$ and $f_{C^*}(t)$ are completely different models with disjoint sets of parameters, we can estimate these functions separately by maximum likelihood. However, the censoring distribution is seldom of interest. Thus, if we only want to estimate the event time distribution, each observation in the dataset contributes to the likelihood with a factor

$$\mathcal{L}_i = f(t^{(i)}; x^{(i)})^{\delta^{(i)}} S(t^{(i)}; x^{(i)})^{1-\delta^{(i)}}. \tag{58}$$

Note that this is the same as in (13), which we derived in the continuous setting. The reason we took time to go through this derivation is because the `DeepHit` model partly uses the loss function

$$L(\theta, x, t, \delta) = -\sum_{i=1}^{n} \log \mathcal{L}_i, \tag{59}$$

where θ is the neural network parameters as per usual.

DeepHit. Now we present `DeepHit`, introduced by Lee *et al.* (2018). For our experiments, the model will be a standard fully connected feedforward neural network with a specific loss function and output type. However, the original implementation allows for expanding the model to handle competing risk events as well. Competing risk events are events that can prevent or modify the occurrence of the primary event of interest. For example, we might be interested in studying the incidence of lung cancer in a population of smokers, where death from other causes such as cardiovascular disease or respiratory failure would be a competing risk, as it prevents the occurrence of lung cancer. `DeepHit` addresses this by employing a shared network connecting to case-specific networks, with a residual connection from the inputs to the case-specific networks, which in turn connects to a common output layer. We do not need to delve deeper into the description as our data will not include competing risks.

The implementation of `DeepHit` we will be using follows the general description of Lee *et al.* (2018), but incorporates suggested modifications from Kvamme *et al.* (2019). The output of `DeepHit` is a vector,

$$y = (f(\tau_1), \ldots, f(\tau_m), S(\tau_m)), \tag{60}$$

where the set of time points $\tau_1, \ldots, \tau_m$ is determined by the user. The cardinality of this set corresponds to the number of output nodes minus 1. Recall that $f(\tau_j) = P(T^* = \tau_j)$ and $S(\tau_m) = P(T^* > \tau_m)$, so y is a model estimate of the PMF of the survival time T^*. The reason for including $(S(\tau_m)$ in the output is so that we do not have to assume that everyone dies before or at time τ_m. We require $\sum_{i=1}^{m+1} y_i = 1$ to have a proper probability distribution, which is ensured by the use of softmax as the output function,

$$\text{softmax: } g_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{m+1} e^{x_j}}, \quad i = 1, \ldots, m+1. \tag{61}$$

To obtain an estimation of the survival curve from the model output, we simply have to sum up individual probabilities according to (48).

Moving on, let $\kappa(t) \in \{0, \ldots, m\}$ denote the index of the discrete time t, i.e., $t = \tau_{\kappa(t)}$. In the case $\tau_j < t < \tau_{j+1}$ for some j, we let $\kappa(t) = \kappa(\tau_{j+1}) = j + 1$. We can now express the loss function used by the model as

$$L_1 = -\sum_{i=1}^{n} \left[\delta^{(i)} \log y^{(i)}_{\kappa(t^{(i)})} + (1 - \delta^{(i)}) \log \left(\sum_{k=\kappa(t^{(i)})+1}^{m+1} y^{(i)}_k, \right) \right] \tag{62}$$

where the first term captures the uncensored observations, while the second term captures the censored ones. Overall, this loss aims to estimate the time distribution of the event probabilities. We can express L_1 using the survival function as:

$$L_1 = -\sum_{i=1}^{n} \left[\delta^{(i)} \log \left(S(\tau_{\kappa(t^{(i)})-1}; x^{(i)}) - S(t^{(i)}; x^{(i)}) \right) + (1 - \delta^{(i)}) \log \left(S(t^{(i)}; x^{(i)}). \right) \right]. \tag{63}$$

The second part of the loss function addresses the orderings of the observations, i.e., subjects with a longer observed time until the event should survive longer than a subject with a shorter observed time. To handle this, the authors define a second loss as

$$L_2 = \sum_{i,j} \delta^{(i)} I \left(t^{(i)} < t^{(j)} \right) \exp \left(\frac{\sum_{k > \kappa(t^{(i)})} (y^{(i)}_k - y^{(j)}_k)}{\gamma} \right), \tag{64}$$

where γ is a hyperparameter. Analysing this loss function, we can see that it includes two indicators. The first indicator, $\delta^{(i)}$, is sensible to include because we do not want to penalise survival estimates if we do not know the actual survival time. The second indicator, $I\left(t^{(i)} < t^{(j)}\right)$, is essential for the exponential factor to make sense. However, if $t^{(i)} = t^{(j)}$, we would like the survival curves to be similar, and the loss could be improved to capture this. The last factor is exponentiated to ensure positiveness, and the parameter γ allows us to decide how heavily to penalise differences. If a subject with shorter survival time indeed

has lower survival probabilities, larger discrepancies are encouraged, without considering how much the survival times differ. Note that we can express L_2 using the survival function more comprehensibly as

$$L_2 = \sum_{i,j} \delta^{(i)} I\left(t^{(i)} < t^{(j)}\right) \exp\left(\frac{S(t^{(i)}; x^{(i)}) - S(t^{(i)}; x^{(j)})}{\gamma}\right). \tag{65}$$

Finally, the total loss is determined by

$$L_{\text{total}} = \alpha L_1 + (1-\alpha) L_2, \tag{66}$$

where α is another hyperparameter balancing how much we value concordance versus general goodness of fit. The effect of this hyperparameter can be further analysed if we write

$$(1-\alpha) L_2 = \sum_{i,j} \delta^{(i)} I\left(t^{(i)} < t^{(j)}\right) \exp\left(\frac{S(t^{(i)}; x^{(i)}) - S(t^{(i)}; x^{(j)})}{\gamma} + \log(1-\alpha)\right). \tag{67}$$

Now, we can see that the two hyperparameters, γ and α, have similar regularising effects, making it somewhat redundant to have them both. However, we note that γ scales the loss exponentially, while α has a constant multiplicative scaling effect. Setting these hyperparameters can be difficult and computationally costly. Unfortunately, making them learnable like the other parameters of a neural network is challenging, since increasing γ always decreases the loss, and if L_1 is smaller than L_2, then α will be set to 1, causing L_2 to be disregarded.

If we want to generate a continuous curve from the output of `DeepHit`, we can interpolate using smoothing splines or any other smoothing technique applied to the step function. The accuracy of the smoothing would improve by adding more time intervals. From there, we can take the logarithm and differentiate to obtain the corresponding hazard rate. If we want to work with a discrete hazard function, we can use

$$\lambda(t) = \frac{S(t-1) - S(t)}{S(t-1)}. \tag{68}$$

Possible weaknesses of `DeepHit`. Hu *et al.* (2021) argue that their Transformer model can achieve better results since their method uses ordinal regression, and ordinal regression

is known to perform better than softmax on ordinal data. Ordinal regression recognises the inherent ordered relationships within the data, allowing for more accurate and contextually relevant model predictions. On the other hand, softmax is generally more appropriate for nominal categorical data. The reasoning is that categorical data treated by softmax lack any inherent order, making it less suited for data where the relative order of the categories carries significant meaning.

An important remark is that `DeepHit` makes no assumptions about the functional form of the hazard function, according to the authors. However, the independence between censoring and survival time is hard to come by and is implicitly adopted in the training process. Overall, having no further assumptions can be advantageous as it offers a fully flexible model. Nevertheless, flexibility comes with its costs, such as lack of interpretability and training difficulties.

Another possible weakness of the model is due to the ranking part of the loss function. In order to seek concordance, i.e., correct ordering of subjects, the actual survival probabilities are deprioritised. In fact, inspecting the formula for L_2 reveals that smaller survival probabilities are encouraged since, with everything else unchanged, it decreases the total L_2 penalty.

2.4 Interpretable approximations of complex models

In many situations, knowing why a model produces a particular forecast can be just as important as knowing if the prediction is accurate. There is a conflict between accuracy and interpretability because the maximum accuracy for huge modern datasets is sometimes achieved by intricate models that even experts have trouble interpreting, like ensemble or deep learning models. To help users interpret the predictions of complicated models in response, a number of ways have lately been put forth, although it is frequently unclear how these methods relate to one another and in what situations one method is better. In the following, we present a unified method to interpret complex models, by perceiving the explanation of a model's prediction as a model itself. Of course, the best explanation of a model is the model itself, provided it is simple enough to understand. When we have a model that is too complex to understand directly, it is better to define an interpretable approximation of the original model.

2.4.1 SHAP values

The main idea behind interpretable approximations of complex models is to find a simpler, more interpretable model that can approximate the predictions of the complex model rea-

sonably well. These simpler models can help users understand how input features are related to the model's predictions, making it easier to interpret the complex model's behaviour.

The unified interpretation method from which SHAP values are derived was defined by Lundberg and Lee (2017). This method provides a local explanation of prediction by defining a model's prediction for any observation $f(x^{(i)})$ as

$$f(x^{(i)}) = \phi_0 + \sum_{j=1}^{p} \phi_j(f, x_j^{(i)}), \tag{69}$$

where $\phi_j(f, x_j^{(i)}) \in \mathbb{R}$ is the SHAP value for observation i and explanatory variable j, based on the prediction model f, and ϕ_0 is a constant. The preceding formula was originally formulated as an approximation of $f(x^{(i)})$ based on using a latent input space $x' \in \mathbb{R}^q$ instead of $x \in \mathbb{R}^p$ where $q < p$, but for our purpose, we do not need to take this approach. It is an additive model that can be used to compare the coefficients $\phi_j(\cdot), j = 1, \ldots, p$ in a straightforward manner. As defined, this is a local explanation method, meaning that it provides explanation of a model's prediction for a particular observation. To use it as a global explanation method, i.e. to describe what explanatory variables are important for the overall prediction, we may use

$$\bar{\phi}_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j(f, x_j^{(i)})|. \tag{70}$$

Using simply the mean and not mean absolute SHAP values is not a sensible approach as an explanatory variable can impact the prediction in both directions.

We have now seen that a prediction by a model can be decomposed as a sum of a certain base value and the individual contributions from the explanatory variables. Next up we address how SHAP values can be calculated.

SHAP values are a modification of Shapley values, a concept that originates from cooperative game theory. In this context, each explanatory variable is considered a player in a game, with the objective of measuring the player's contribution to the prediction while accounting for other players' influence. The prediction serves as the payoff in this game. Shapley values were first introduced by Ferguson (1959) to evaluate the relative importance of explanatory variables in multivariate linear regression models, particularly in the presence of multicollinearity, where effect parameters can be unreliable.

To compute these values, predictions are compared when a variable is present in the model

versus when it is absent. Let F be the set of explanatory variables with cardinality $|F| = p$. Consider a subset of variables $S \subseteq F \setminus \{j\}$ that excludes explanatory variable j. We train a model $f_{S\cup\{j\}}$ with the variable present and another model f_S with the variable absent, denoting the input variables corresponding to set S as $x_{\{S\}}$. The proposed estimation is

$$\phi_j(f, x_j^{(i)}) = \sum_{S\subseteq F\setminus\{j\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}(f_{S\cup\{j\}}(x^{(i)}_{\{S\}\cup\{j\}}) - f_S(x^{(i)}_{\{S\}})). \tag{71}$$

This equation takes the sum over all possible subsets and calculates a weighted average of the difference in prediction output. The factor

$$\frac{|S|!(|F|-|S|-1)!}{|F|!} \tag{72}$$

is the reciprocal of $|F|$ choose $|S|$, the possible number of subsets, divided by $(|F| - |S|)$ since the subset size must be smaller than $|F|$. This ensures proper weighting of differences for various subset sizes. To avoid confusion in notation, the function f input to $\phi_j(f, x_j^{(i)})$ can be thought of as the general prediction method used, such as linear regression in this case. By letting $\phi_0 = f_\emptyset(\emptyset)$ correspond to the intercept of the model, we retrieve formula (69) for the full model.

Although the previously described method establishes an intriguing connection between game theory and mathematical statistics and can be useful even if the underlying model is not linear, it is often infeasible to perform the calculations in a modern setting where the input space has high dimensionality. If our model is nonlinear, formula (69) is not exact, but becomes a local linear approximation of the true function. To make the computations more efficient in general, we need to make further approximations.

SHAP values, as introduced by Lundberg and Lee (2017), are somewhat abstract, and several methods have been proposed to estimate them. Instead of calculating $f_S(x^{(i)}_{\{S\}})$ as in (71), they advocate for using the conditional expectation

$$E[f(x^{(i)})|x_{\{S\}}] \approx E_{S^c|S}[f(x^{(i)})]. \tag{73}$$

Here, S^c is the complement of S. The purpose of using the first conditional expectation is that we do not need to retrain a model for every subset of variables, but can instead use the original full model. This is an approximation under the assumption that the model's predictions are approximately linear in the subset of features S. The second approximation

to $E_{S^c|S}[f(x^{(i)})]$ is exact if the underlying model is additive and is made to ensure efficient computations. The approximations are done similarly for $f_{S\cup\{j\}}(x^{(i)}_{\{S\}\cup\{j\}})$.

Limitations of SHAP values. While SHAP values provide a powerful way to interpret complex models, they have some limitations, including except what is already mentioned that they are model-agnostic but not entirely universally applicable. Some models, like recurrent neural networks and certain Bayesian models, may not be easily explainable using SHAP values. Furthermore, SHAP values primarily provide local explanations, meaning they explain individual predictions. While they can be aggregated to create global explanations, this aggregation might not always provide the best overall understanding of the model's behaviour. Also, SHAP values can be sensitive to the input representation of the data, meaning that different feature engineering choices could lead to different interpretations. This can make it challenging to determine the best representation for interpretability purposes. The last issue is especially relevant for this thesis.

SHAP values for neural networks. We propose our own approach for calculating SHAP values based on formula (71). From the definition of neural networks, we know that setting an input variable to 0 eliminates all its connections to the second layer. Thus, a model with one input variable always set to 0 is essentially the same as a model without that variable. Moreover, since a model with or without certain explanatory variables shares many characteristics, we do not need to train all models in (71) from scratch. Instead, we train the full model first and then fine-tune each smaller model by running the training algorithm for a few more epochs, enough to make a significant update. The approach we take to calculate the SHAP value for explanatory variable j can be summarised as follows.

1. Train the full neural network model f.
2. For each subset of explanatory variables $S \subseteq F \setminus \{j\}$ of decreasing size:
   - Train the model $f_{S\cup\{j\}}$ by continuing to train f for 5 epochs with all variables except $S \cup \{j\}$ set to 0.
   - Train the model f_S similarly for another 5 epochs.
   - Calculate their difference.
3. Retrieve the SHAP value $\hat{\phi}_j$ after weighing and taking the sum as in (71).

We choose decreasing subset sizes in the second point above for training efficiency. Since we train the full model first, we want to make small changes to the model we are retraining for each subset. The choice of 5 epochs is based on empirical results on the chosen dataset,

as we found that most training progress was made within 5 epochs. We use a hat on ϕ_j in step 3 to emphasise the approximation of the actual Shapley values, which only hold for additive models. Note that a significant amount of time is saved by performing the prediction difference calculation in step 2 simultaneously for all observations. In the end, we obtain one SHAP value for each observation and explanatory variable.

In our case, f will not be a scalar, but a probability mass function. There are several options for addressing the issue of f being multidimensional. We will take f to be the median survival time, which is more suitable than the mean for survival analysis. The median survival time is determined by the PMF and simplifies the original output of the model. This introduces a previously untouched aspect of SHAP values. As they are defined, their primary use is for regression problems. This limitation restricts their applicability to a single problem domain, but it also opens up possibilities for exploring generalisations.

As discussed earlier, our approach is not feasible if there are too many input variables. In such cases, an alternative would be to sample uniformly from the power set of variables. This is similar to employing dropout on the input layer, a popular regularisation technique.

To verify the consistency of the proposed approach for calculating SHAP values, one can set up a basic simulation case where the effects of covariates are additive, so that the contribution of each variable is precisely known. This would help in assessing the accuracy and reliability of the calculated SHAP values in comparison to the true contributions.

In conclusion, the method we have proposed for calculating SHAP values is an extension of the original approach, making it applicable to more complex models like neural networks and survival analysis. However, it's important to keep in mind the limitations and assumptions made in the process. Specifically, the method's feasibility depends on the number of input variables, and the approximations may be difficult to check if they are reasonable. Finally, when interpreting the results, it's essential to remember that SHAP values provide an approximation of each variable's contribution. The contribution of an explanatory variable is in this context defined as how much the actual prediction will differ from the mean prediction is at the current level of the explanatory variable, in the presence of the other variables.

3 Results and discussion

3.1 Evaluation metrics

Evaluation metrics serve as crucial yardsticks for gauging the performance of models, especially when comparing those with distinct cost or likelihood functions. These metrics furnish an objective lens, providing quantifiable insights into the model's predictive prowess. An arsenal of dependable evaluation metrics is therefore instrumental not only in assessing the model's current performance, but also in illuminating potential avenues for further refinement and enhancement.

3.1.1 Time dependent C-index

The concordance index (C-index), a widely-used performance measure for survival models, was initially introduced by Harrell *et al.* (1982). It is calculated as the proportion of concordant pairs of observations, where a concordant pair represents a pair in which the event for an individual with a higher predicted risk occurs before the event for an individual with a lower predicted risk. Extensions to the C-index were made by Antolini *et al.* (2005), and are now implemented in most programming packages for survival analysis. Specifically, let $t^{(i)}$ and $t^{(j)}$ be the survival times for individuals i and j, and let $1-\hat{S}(t^{(i)}|x^{(i)}) = \hat{F}(t^{(i)}|x^{(i)})$ and $1-\hat{S}(t^{(i)}|x^{(i)}) = \hat{F}(t^{(i)}|x^{(j)})$ be their corresponding predicted risks at time point $t^{(i)}$. The C-index is defined as, when using the survival function instead of the CDF,

$$\hat{C}^{\text{td}} = \frac{\sum_{i<j} I(t^{(i)} < t^{(j)}, \delta_i = 1) \cdot I(\hat{S}(t^{(i)}|x^{(i)}) < \hat{S}(t^{(i)}|x^{(j)}))}{\sum_{i<j} I(t^{(i)} < t^{(j)}, \delta_i = 1)}, \tag{74}$$

where δ_i is the event indicator for individual i. Moreover, the indicator function $I(t^{(i)} < t^{(j)}, \delta_i = 1)$ has two arguments and takes the value 1 if both of them are true. The formula essentially divides the number of concordant pairs by the number of concordant and discordant pairs (put differently, comparable pairs). If one individual has a shorter survival time than another individual and a higher estimated probability of experiencing the event up until the time point of the event, we have a concordant pair. It essentially estimates the probability that two individuals are concordant given that they are comparable, which corresponds to averaging the probability

$$C^{\text{td}} = P(\hat{S}(t^{(i)}|x^{(i)}) < \hat{S}(t^{(i)}|x^{(j)}) \mid t^{(i)} < t^{(j)}, \delta_i = 1), \tag{75}$$

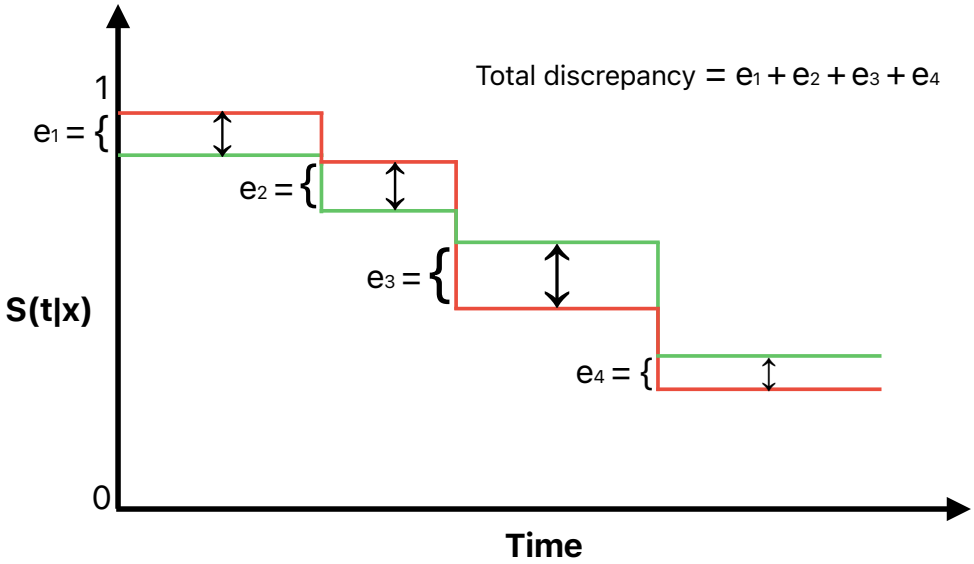

Figure 3: Manhattan distance as a measure of evaluation. Two survival functions are displayed in different colours. We sum up the individual errors at each time point $e_1, \ldots, e_4$ to get the total discrepancy, which is used for evaluation.

over i, j. First, going back to (74), note that the time point $t^{(i)}$ is used for calculating the risk for both individuals, which is necessary as the cumulative risk always increases and thus we cannot use different time points when comparing. Secondly, since censored observations do not reveal the actual survival time, we cannot compare their risks. Hence, pairs of censored observations are completely disregarded. Still, if individual i is uncensored and $t^{(i)} < t^{(j)}$, the individuals are comparable regardless of whether individual j is censored. The measure ranges from 0 to 1, with 1 corresponding to perfect ranking of individuals.

3.1.2 Manhattan distance

When we know the true survival function, the absolute distance between the estimated and the true curve is a natural measure. This is also referred to as Manhattan distance. An illustration is provided in Figure 3, where two different survival functions are drawn and we measure the total absolute difference between them. We can then take the average discrepancy across all validation observations as a performance measure.

Mathematically, we can express the formula for Manhattan distance in the context of survival analysis between a survival function S and an estimate $\hat{S}$ as

$$M_{\text{dist}}(S, \hat{S}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} |S(\tau_j | x^{(i)}) - \hat{S}(\tau_j | x^{(i)})|. \tag{76}$$

More precisely, the inner sum of this equation is the Manhattan distance indicated in Figure 3, and M_{dist} is the average over n observations. We can see that M_{dist} is bounded by 0 below and by m, the number of time ticks used, above. We could divide M_{dist} by m to get a measure always in the range 0 to 1, but the measure is solely used to compare different models, so we avoid doing this. Note also that we have no guidelines except intuition to what values are good or bad in a universal sense. Overall, this measure is a good complement to the C-index as it measures the closeness of fit, but does not directly account for ranking of individuals.

As mentioned in the beginning, this measure is only applicable when we know the true survival function. It completely ignores the actual survival time and censoring status, as they are stochastic.

3.1.3 Integrated Brier score

If we do not have access to the true survival function, which is the case when we are working with real data, the Brier score provides a good evaluation measure alternative that focuses on closeness of fit rather than ranking of individuals. It has been generalised by Graf *et al.* (1999) from a binary classification setting to the survival analysis setting with censored observations. We define it as

$$\text{BS}(t) = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\hat{S}(t|x^{(i)})^2 \cdot I(t^{(i)} \leq t, \delta_i = 1)}{\hat{G}(t^{(i)})} + \frac{(1 - \hat{S}(t|x^{(i)}))^2 \cdot I(t^{(i)} > t)}{\hat{G}(t)} \right], \tag{77}$$

where $\hat{G}(t)$ is the Kaplan-Meier estimate for $P(C^* > t)$, i.e. the censoring survival function. The role of $\hat{G}(\cdot)$ is to weigh scores, such that longer survival times are given higher weights, since the survival function by definition is smaller for larger t. The weight given corresponds to the inverse censoring distribution. If it happens to be in a situation similar to the third simulation experiment described later on, where all observations are censored if the time reaches a specific time point, then $\hat{G}(t) = 1$ for all t smaller than this time point. We use the Kaplan-Meier estimate of C^* instead of T^* or $T = \min(C^*, T^*)$ in the formula. This is to account for censoring information, i.e. the probability of being censored at a specific time point, yielding more accurate survival probabilities.

To gain better understanding, we take a simple example. Assume that individuals either experience an event at time 1, 2 or 3, and censoring occurs at time 1 with probability 1/2.

For a single individual experiencing the event at time 2 without being censored, we have $\mathrm{BS}(1) = (1 - \hat{S}(1|x^{(i)}))^2 \cdot 2$. Now, the lower the score, the better, so we want $\hat{S}(1|x^{(i)})$ to be large. Similarly, we have $\mathrm{BS}(2) = \hat{S}(2|x^{(i)})^2 \cdot 2$ and thus we want $\hat{S}(2|x^{(i)})^2$ to be small, which aligns with the intuition. Since a subject cannot survive past time 3, the survival function should be 0 for $t \geq 3$.

Further examination of (77) reveals that censored observations with $t^{(i)} < t$ does not contribute to the score. This makes sense as we do not know whether the actual survival time is larger or smaller than t. The squares in the formula arise from its earlier definition of being a mean squared error estimate.

To obtain a single value over an entire time interval, we integrate the Brier score to obtain

$$\mathrm{IBS} = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} \mathrm{BS}(s) ds. \tag{78}$$

This is the integrated Brier score we use for evaluation. The integral is approximated numerically in applications and our implementation will set T_1 to be the smallest observed time in the validation set and T_2 will be the largest. We use 100 time points in between for the integration, supported by Kvamme *et al.* (2019).

3.2 Optimisation procedure

To perform a fair evaluation of our models, we need to split each dataset into a training and validation set. We will use a validation set consisting of 30% of the data points for the simulation experiments. For the colon cancer dataset, we will instead use 20% due to its limited size. However, we will not have a separate test set for the ultimate comparison of models. It is somewhat controversial to use a single validation set for both selecting a model's hyperparameters and evaluating its performance, but in our setting, it is justified. Our study focuses on model selection rather than providing a final, unbiased performance estimate, which is typically the purpose of a test set. Moreover, the purpose of validation data here is not only to select hyperparameters but also to ensure that we are not overfitting the training data.

3.2.1 Choosing time points

Due to the nature of `DeepHit` and discrete survival analysis in general, we need to define the time points for which we seek to estimate the survival probabilities. Even in situations where

the survival data is inherently discrete, it is often desirable to create a more coarse grid of time points instead of using the full discrete time scale provided by data. One example is to reduce the number of parameters in a neural network. The obvious way to achieve such a grid for a defined interval $[0, \tau]$ is to set $0 = \tau_0 < \tau_1 < \cdots < \tau_m = \tau$, with $\tau_i = i \cdot \tau / m$. This creates a grid with equidistant time intervals. One obvious drawback is that time intervals will contain different numbers of observations, possibly making the estimation unstable. Therefore, we adopt a more sophisticated approach.

We define the time grid based on quantiles of the survival function $S(t)$. For our simulated datasets, we have access to $S(t)$, but in general, this is not the case. Hence, we will estimate it by the Kaplan-Meier curve $\hat{S}(t)$. Each time interval will then correspond to a decrease of $\frac{1-\hat{S}(\tau)}{m}$ in the estimated survival function. Note here that τ represents the largest event time and it is not for certain that every subject has experienced the event by this time. To retrieve the time points after the quantiles have been settled, one solves

$$\hat{S}(\tau_i) = \hat{S}(\tau_{i-1}) - \frac{1 - \hat{S}(\tau)}{m}, \tag{79}$$

with $\hat{S}(\tau_0) = 1$. We do this separately for all datasets, and a visualisation is shown in Figure 4, which depicts a discrimination grid example for our first simulated dataset. In the plot, the vertical dashed lines represent the time points where the survival function is to be estimated by modelling. Hence, these will be the time points where we evaluate the models as well. We can see that the gap between the time points grows larger over time, as events occur less frequently with time.

Choosing the number of time points m is ultimately an assessment task for the user. While having a larger m provides a smoother estimate, with limited training data, the estimation becomes more difficult. In other words, the more time points we have, the worse the accuracy becomes for the estimation at each time point. We also have to factor in the characteristics of `DeepHit`. For every increase of m by 1, we increase the total number of parameters by 1 plus the number of nodes in the last hidden layer. Furthermore, most neural networks used for prediction output one or a few values from a larger amount of input variables. Here, we are possibly looking at a situation where the input dimension is smaller than that of the output. Still, we need a sufficient number of time points to have a meaningful representation. With this in mind, we will choose m between 15 and 30, depending on the dataset. An alternative to increasing m to obtain smoothness is to apply interpolation methods, such as those discussed in Kvamme and Borgan (2019).

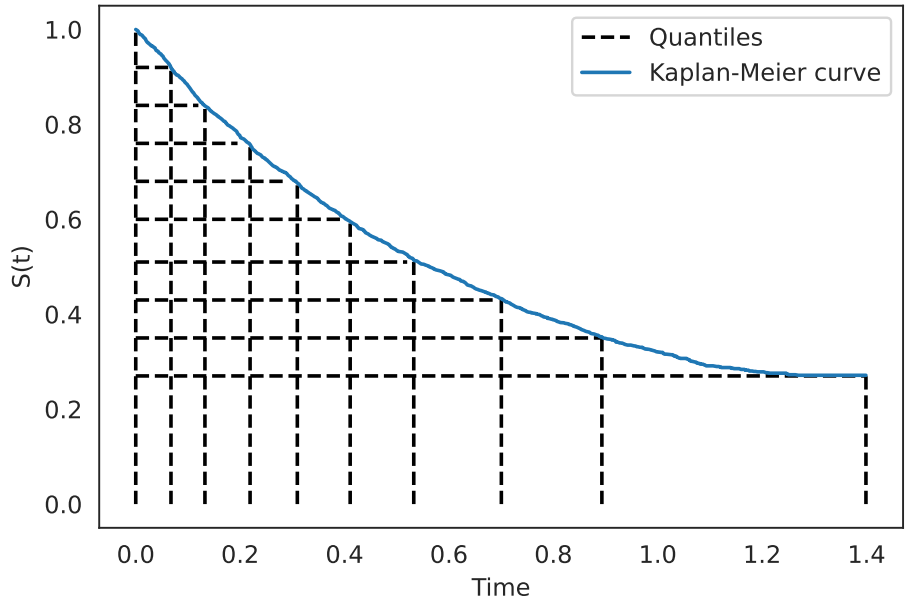

Figure 4: Example of the quantile approach to determine time points of evaluation. The vertical dashed lines show the discretisation grid and each interval corresponds to equally sized decreases of the survival curve, indicated by the horizontal dashed lines. The curve does not go down to 0 because there are still people at risk after the last observed event.

3.2.2 Neural network training

Training and optimising a neural network is not a trivial task by any means. Most other machine learning methods are more straightforward to train, and cross-validation can be used to set hyperparameters. However, when it comes to neural networks, there is an abundance of hyperparameters, settings, and other features, making it intractable to use cross-validation for finding the best performing model. Furthermore, in applications where neural networks are used today, training takes a significant amount of time, making it inappropriate to use cross-validation. With that said, we will rely on existing implementations and relevant suggestions to build the neural network models. We train our neural networks using the `pycox` package in `Python`.

Avoiding overfitting is perhaps more important than using optimal hyperparameter values. Numerous methods exist to address this issue as well. The `Adam` optimiser described earlier can somewhat alleviate the problem by shrinking the learning rate, but ultimately we will rely on manual inspection of the training and validation error curves to spot overfitting. A

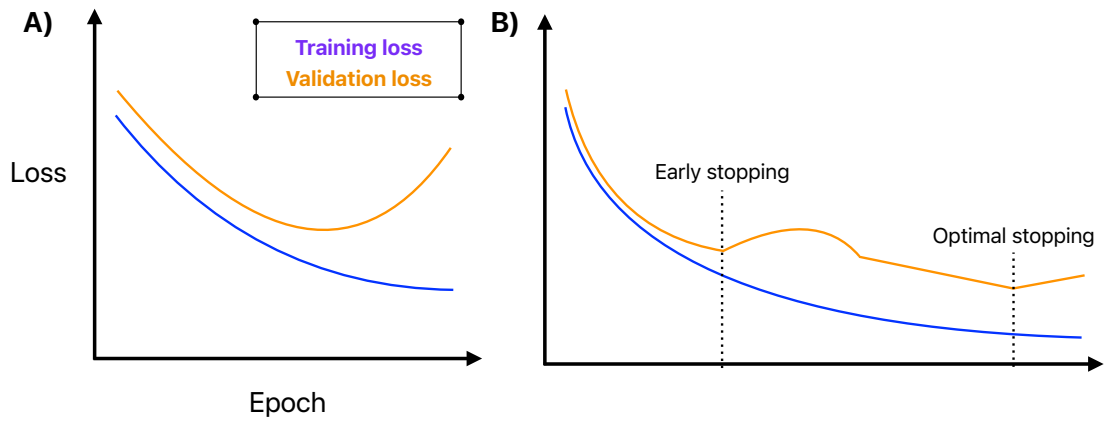

Figure 5: Two theoretical examples of how training and validation loss evolves the longer a model is trained. Algorithms seek to decrease training loss without taking into consideration how well the model can generalise. A) shows a situation where the model starts to overfit as the number of epochs increase, seen by the discrepancy of the two curves. B) shows a situation where the validation loss starts increasing but then decreases again, and early stopping may sub-optimally cause the algorithm to stop training at this point.

toy example for illustration is given in Figure 5, where the issue with a popular overfitting prevention technique is highlighted. This technique, called early stopping, is in its most basic form just an instruction to the algorithm to stop training when the validation error starts to increase.

Two problems with training neural networks are showcased in Figure 5. A naive training strategy is to set the number of epochs at some level and use the fully trained model, without validating the losses. In that case, we could end up in a situation similar to A), where the model's ability to generalise would benefit from less training. To avoid such a situation, early stopping can be used. However, it is not impossible for the validation loss to start increasing and then decrease again to reach even lower levels of error, as shown in B). In this situation, it is better to train the model for many epochs until we are more certain the validation loss will not decrease further. Afterwards, we can use the weights and parameter values obtained after the optimal number of epochs, indicated in the figure.

Real-life examples of training and validation loss curves are usually not smooth, due to the stochastic gradient descent. In general, the larger batch size we have, the less volatile the training will be. An example from our experiments is shown in Figure 6, where a model was trained on the third simulated dataset. We can see that after around 40 epochs, there is no further significant increase or decrease in the validation loss. We can also see that the training loss keeps decreasing, although at a very slow pace, which eventually might lead

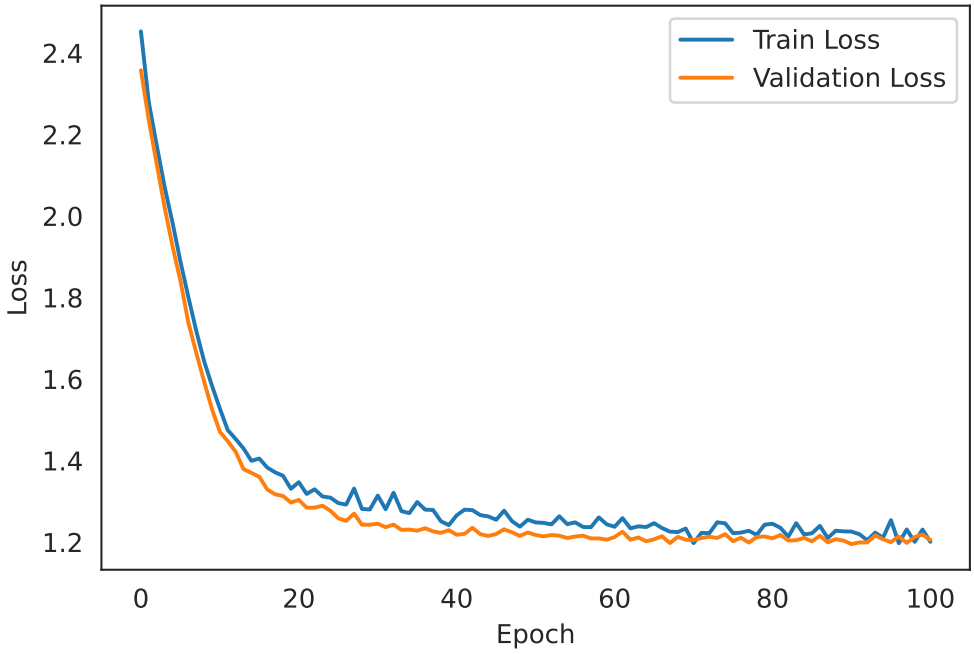

Figure 6: Training and validation loss when training a model on the third simulated dataset. Both losses decrease and stabilise quickly.

to overfitting. Overall, we would probably benefit from reducing the number of epochs for this particular experiment.

There are several procedures that can be used to check whether the training algorithm has converged. The easiest way is arguably to monitor the sizes of the updates produced by the `Adam` algorithm and see if they converge to 0. As the main point of this paper is not to optimise neural networks, we will not go into more detail. The takeaway from the preceding paragraphs is that we train our neural networks for many epochs and look at the iteration where validation error is smallest, meaning that we rely on manual assessment of convergence or reach of optimum. When we compare performances of different models to determine, for example, what architecture to use, we will take the smallest validation loss obtained during training as a reference point.

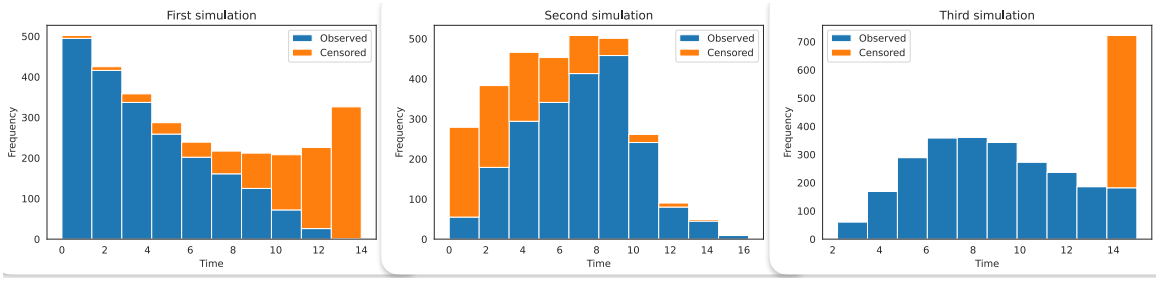

Figure 7: Survival and censoring times for the three simulations, from the first to the third. Both the actual and censored times exhibit varying distributions.

3.3 Comparison on simulated data

3.3.1 Simulation methods

In this study, we have chosen three different simulation approaches to thoroughly evaluate the performance of our models under various scenarios. These simulation methods have been designed to represent a wide range of survival time distributions and censoring mechanisms, which are common in real-world survival analysis problems. By employing exponential, Weibull, and lognormal distributions, we aim to cover a broad range of hazard functions, from constant to monotonically increasing and more complex shapes. Furthermore, these simulations allow us to test the model's robustness and adaptability under different violation levels of the proportionality and linearity assumptions.

The selected censoring mechanisms reflect practical situations commonly encountered in survival analysis, such as study time limitations, competing risks, or event-independent censoring. Through these simulations, we can assess the model's ability to handle diverse censoring patterns and evaluate its performance under different censoring rates. In essence, the chosen simulation approaches provide a comprehensive test basis for our models, ensuring that the results and conclusions derived from these experiments are insightful and applicable to a wide range of real-world survival analysis problems. The limitation is that we assume well-known distributions when simulating data.

Before describing how the data points for the simulations are generated, we show the survival times for the three experiments in Figure 7. All simulations include 3000 data points each. The experiments are conducted to give distributions of diverse shapes. Even though samples are drawn from the exponential, Weibull, and lognormal distributions respectively, the censoring mechanism alters the actual distributions. Only for the third simulation, shown in the rightmost plot, the blue bars represent the true distribution, with its tail being cut

off and stacked into the orange bar.

First simulation. In the first experiment, we simulate survival times from the exponential distribution with density $f(t|x) = \lambda(t|x)\exp(-t\lambda(t|x))$. Here, $\lambda(t|x)$ is both the rate parameter and the hazard rate, which is in fact independent of time. Hence, we simply define $\lambda(t|x) = c * \exp(\beta x)$, and the inverse of this quantity is the mean survival time for a particular subject. We let $c = 1$, or in other words $\lambda_0(t) = 1$. Thus, we assure that the linearity and proportionality assumptions are fulfilled. We simulate the covariates x with no correlation and set the effect parameters β at both negative and positive values. The covariate vector x is 15-dimensional and consists of 10 binary variables with varying success probabilities, 2 normally distributed variables with different variances, and 3 uniform, one of which is designed to emulate an "age" variable, commonly included in real datasets.

To evaluate survival probabilities for a given subject, we can simply take 1 minus the cumulative distribution function

$$P(t \geq t^{(i)}) = 1 - F(t^{(i)}|x^{(i)}) = \exp(-t^{(i)}\lambda(t^{(i)}|x^{(i)})) = \exp(-t^{(i)}\exp(\beta x^{(i)})). \tag{80}$$

To synthetically make some observations censored, we simulate for each survival time a uniformly distributed variable and let the observed time be the minimum of the two. Hence, if the uniformly distributed random variable is smaller than the simulated survival time, we make the corresponding observation censored and set the observation time to the uniformly distributed variable. This was adjusted to yield around 30% censored observations. The used censoring approach can be motivated by real-world examples where the time for the end of the study is set and subjects enter the study at random time points, similarly to the real dataset we will study later on.

Second simulation. For the second experiment, we simulate survival times from the Weibull distribution, which is much more flexible than the exponential distribution. The density function is

$$f(t|x) = \frac{k}{\lambda_x}\left(\frac{t}{\lambda_x}\right)^{k-1} e^{-(t/\lambda_x)^k}, \tag{81}$$

and letting $k = 1$ gives the exponential distribution. Here, λ_x is a scale parameter we will make dependent on the covariate values. The hazard function for the Weibull distribution is

$$\lambda(t|x) = \frac{k}{\lambda_x}\left(\frac{t}{\lambda_x}\right)^{k-1}. \tag{82}$$

For this experiment, we want to keep the proportional hazards assumption, but abandon the log-linearity assumption. Hence, we wish to specify $\lambda(t|x) = \lambda_0(t)\exp(g(x))$. We fix $k = 3$, meaning that the hazard rate increases quadratically in time, and using the above the baseline hazard becomes $\lambda_0(t) = 3t^2$. This means that $\lambda_x = \exp((-1/3)g(x))$ and using a 5-dimensional input we define the nonlinear function

$$g(x) = x_1x_2 + x_3x_4 + x_1^2 + x_2^3 + \sin(x_3) + \cos(x_4) + \log(x_5), \tag{83}$$

where all covariates are uncorrelated and simulated from uniform distributions on different scales. It is possible to deduce that the mean survival time is decreasing in the covariates x_1, x_2 and x_5.

The theoretical survival function becomes

$$1 - \int_0^t f(t|x)dt = \exp(-(t/\lambda_x)^k) = \exp\left(-\frac{t}{\exp(-(1/3)g(x))}\right)^3. \tag{84}$$

To introduce censoring in this experiment, we sample for each survival time a realisation from the exponential distribution, and take the minimum of the two. This means that if the sample from the exponential distribution is smaller than the corresponding simulated survival time, we right-censor the observation and let the sample from the exponential distribution be the observed time. Large survival times are thus likely to be censored. The mean of the exponential distribution was adjusted to yield around 30% censored observations. To motivate the carried out censoring method, we can think of a situation where death is not of main interest but rather recurrence of disease or similar. In this case, the time of death might be exponentially distributed and interrupt the follow-up study.

Third simulation. One way to simulate survival times from a distribution with an arbitrary non-negative hazard function $\lambda(t|x)$ is to use Monte Carlo simulation. The hazard rate method of simulation can be used if we ensure that $\lambda(t|x) \leq \lambda$ is bounded by some constant λ for all $t \geq 0$ and

$$\int_0^\infty \lambda(t|x)dt = \infty. \tag{85}$$

The procedure is as follows.

1. Simulate a Poisson process with rate λ.
2. With probability $\frac{\lambda(s|x)}{\lambda}$, accept an event that occurs at time s.
3. The time of the first accepted event is a realisation from the desired distribution.

With enough realisations, we can get a good estimate of the probability density function for the survival time.

Another more direct approach could be used in the discrete-time framework. Specify an arbitrary non-negative hazard function and ensure it is bounded above by 1, which can be accomplished by applying the logistic transformation. Recall that this is a requirement when working with discrete survival times. From previous sections, we know that the probability mass function can be obtained directly from the hazard function, making simulation straight-forward.

We adopt neither of the above approaches and choose to instead work with a well-known probability distribution yet again. This is done for convenience, as having an analytical formula for the density function is beneficial and assuming a discrete hazard function is restricting. This time, consider the lognormal distribution

$$f(t|x) = \frac{1}{t\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln t - \mu_x)^2}{2\sigma^2}\right). \tag{86}$$

We let the covariates x be embedded in the parameter μ_x of the distribution. Specifically, our experiment uses 4 covariate variables and we set

$$\mu_x = 2^{x_1} + \frac{1}{\sqrt{1+x_2}} + \log(1+x_3) + x_4, \tag{87}$$

where x_1, x_2, x_3 are simulated from different uniform distributions and x_4 is a binary variable. This gives us a hazard function $\lambda(t|x) = \frac{f(t|x)}{1-F(t|x)}$ that is nonlinear in the covariates and t, and does not have proportional hazards with respect to the covariates. At the same time, it offers us straight-forward simulation where we know exactly how the mean relates to the covariates. Note however that μ_x is not the mean of this distribution and σ^2 is not the variance. We set $\sigma^2 = 0.1$ to reduce randomness compared to the previous simulations. For evaluation later, the survival probability for a realisation is of course $1 - F(t^{(i)}|x^{(i)})$.

To make some observations censored, for each survival time we use a cut-off such that all survival times exceeding this cut-off time are censored at the cut-off time. The maximum observed time will in this case be 15, which was chosen to obtain a censoring fraction of around 20%. This censoring approach is supposed to mimic a study where every subject is followed up for a fixed amount of time, unless death occurs earlier.

3.3.2 Neural network architecture and hyperparameters

The selected hyperparameters in Table 1 were chosen based on a combination of empirical evidence, manual experimentation, and common practices in the field. This approach was taken to provide a balanced and effective starting point for the model, allowing it to adapt and perform well across diverse datasets. It is essential to note that the performance of the model could potentially be further improved by fine-tuning these hyperparameters for each specific dataset. However, given the limited computational resources and the focus of this study on providing a unified framework, the chosen settings offer a practical and efficient solution for training the neural networks.

For the three hyperparameters in the table set by manually trying different values, experimenting was done on the third simulated dataset, since in theory it has the most complicated structure. With limited computational power, we can only vary one parameter at a time and keep the others fixed. Our motivations in Table 1 are not thorough, but does at least give some insight of the thought process behind determining the settings.

An important side note has to be made regarding the motivation of α in Table 1. We cannot rely on validation loss to assess which parameter value works the best, since αL_1 can be much bigger than $(1-\alpha)L_2$ even if α is small. Hence, performance here is measured in terms of Manhattan distance score. Furthermore, our experiments showed that the test results were very sensitive to the choice of both α and γ, highlighting a potential weakness of the model.

3.3.3 Performance measure scores for the simulations

In Table 2, we show the results combined from all simulations. We can see that `DeepHit` achieves lower scores for all experiments in terms of Manhattan distance, in fact much lower for the second and third simulation. This was expected as only data from the first simulation fits the assumptions of Cox regression. Furthermore, since the same amount of time steps were used in all experiments, we can compare the scores between experiments as well. Hence, we can see that the estimated survival curves in the third experiment were estimated the best for the neural network model. Both models struggles with estimation for the second

Hyperparameter	Value	Motivation
Activation function	`ReLU`	Fast training and used by Lee et al. (2018). This is used across all hidden layers.
Weight initialisation	`Xavier`	Introduced earlier in this paper and recommended by DeepLearning.AI (Katanforoosh & Kunin, 2019)
Optimiser	`Adam`	Often used as the default optimiser in modern machine learning. β_1, β_2, δ set to default. For step size (ϵ) see below.
`DeepHit` specific hyperparameters		
γ (amplification parameter in L_2)	1	The parameter is somewhat redundant in the presence of α below. We fix it at 1 to reduce the number of factors and vary α instead.
α (tradeoff between L_1 and L_2)	0.7	Tested $0.2, 0.5$ and 0.8. Performance was much worse for the smaller α and slightly worse for $\alpha = 0.8$.
Remaining hyperparameters set by manual testing		
Architecture	2-3-5	(Meaning three hidden layer network with hidden nodes 2, 3 and 5 times the input dimension, respectively.) We tested 2-3 and 3-5 as well, with worse results. We avoid deeper and wider networks to simplify the training process.
Mini batch size	32	A size of 8 yielded unstable and worse results, and a size of 128 stabilises the loss, but at a higher minimum.
Step size (ϵ)	0.001	A value of 0.0001 resulted in too slow learning, and a value of 0.01 resulted in too fast learning with worse outcome.

Table 1: Hyperparameters and their settings for the neural networks trained on the three simulated datasets.

Manhattan distance between survival functions			
Model	Simulation 1	Simulation 2	Simulation 3
CoxPH	0.803	3.443	0.825
DeepHit	**0.726**	**1.725**	**0.344**

Concordance index			
Model	Simulation 1	Simulation 2	Simulation 3
CoxPH	0.585	0.890	**0.922**
DeepHit	**0.594**	**0.956**	0.821

Table 2: Results reported as Manhattan distance and concordance index for all simulations.

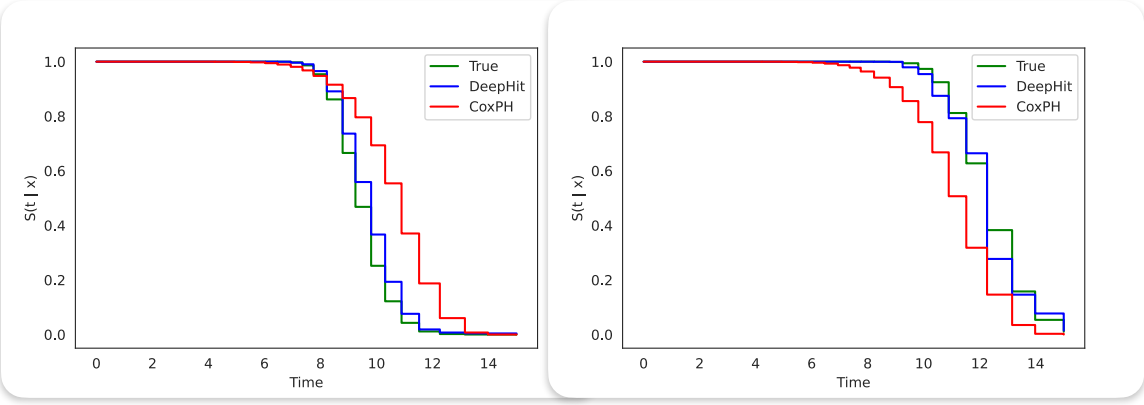

Figure 8: Estimated and true survival curves for two random individuals. The `DeepHit` model is able to fit the true curve rather closely, while the Cox model is struggling somewhat. The sample is taken from the third simulation.

simulated dataset. In particular, the generalisation seems to fail for some observations, since by looking at estimated survival curves we find that it is severely misestimated for a few observations.

For illustration, Figure 8 shows the estimated and true survival curves from the third simulation for two random individuals. We can see that the Cox model fails to adopt different shapes of the survival function for different individuals. Although this is a theoretical example on simulated data, it exhibits the flexibility of `DeepHit` and the limitations of Cox regression.

Going back to Table 2, we can see that `DeepHit` achieves higher concordance scores for the first and second simulation. Perhaps a bit surprisingly, the Cox model scores a better concordance index for the third simulation. Even more surprisingly is the high score in comparison to the other experiments, despite the nonlinearity. This is probably because the third experiment uses the smallest set of covariates. The highest concordance index is achieved by `DeepHit` for the second experiment, indicating that observations were the easiest to differentiate from each other here, even though the model struggled with fitting the survival curve. Note that there is a tradeoff between optimising likelihood and optimising concordance when training the `DeepHit` model. The experiments were very sensitive to changes in the tradeoff parameter α, so we would have ended up with different results had we chosen other optimising priorities.

A possible weakness of a model can be that it systematically over- or underestimates the true survival function. A check for simulation 3 shows that neither method does this. The

check is done by removing the absolute value in the formula for computing the Manhattan distance and see whether the average deviates a lot from 0. For the second experiment, the check shows that DeepHit overestimates somewhat (0.06 on average), and CoxPH overestimates the survival substantially with an average of 0.14, i.e. the accumulated survival probabilities are overestimated by 0.14 units. In cases where the models overestimate survival probabilities, calibration methods could be applied to improve the accuracy of the predictions.

Diagnosing the poor results for the first simulation, CoxPH probably struggles with the larger amount of covariates. Both models systematically overestimate the survival probability, especially CoxPH. It may be due to the structure of data, that it has much randomness in the distribution of survival times and a considerable probability of very large survival times, which is because of simulation from the exponential distribution.

In summary, the results from the simulations highlight the advantages of the `DeepHit` model over the traditional CoxPH model in terms of flexibility and ability to capture nonlinear relationships. However, the concordance index results reveal that the Cox model can still perform surprisingly well in certain situations, such as the third simulation, where it achieved a higher concordance score than `DeepHit`. This indicates that the Cox model can still differentiate between observations relatively well, even when faced with nonlinear relationships.

3.4 Comparison on real data and `DeepHit` interpretation

3.4.1 Data description

The dataset encompasses a study on colon cancer, where patients underwent various treatments with the aim of enhancing their chances of prolonged survival. The data can be found in the `R` package `survival`, and we adhere to the same preparation procedure as outlined by Sundrani and Lu (2021), which involves removing redundant variables and recoding categorical variables. Additionally, the data addresses cancer recurrence, which can be perceived as a competing risk; however, we opt to focus solely on the time to death, irrespective of recurrence. The dataset contains 929 observations.

There are 10 explanatory variables, which consist of the following types.

- 5 binary variables: sex, obstruct (obstruction of colon by tumour), perfor (perforation of colon), adhere (adherence to nearby organs) and surg (short or long time from surgery to registration). Sex is balanced, whereas the other variables are unbalanced toward the baseline level.

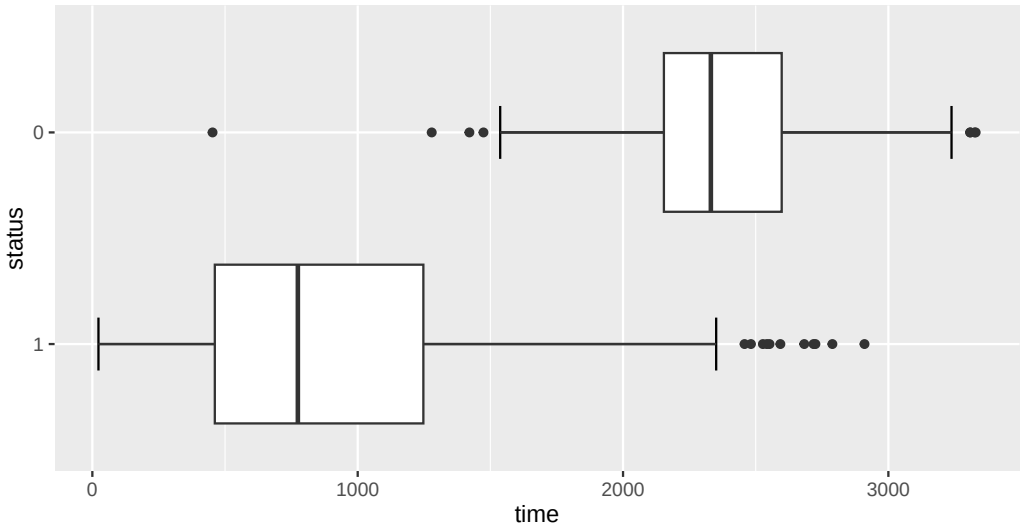

Figure 9: Boxplot of times to event for the colon cancer dataset, where event is either death censoring. Status 1 means censoring, i.e. no death within the study time, and status 0 means no censoring, i.e. observed death. Most deaths occur between 1500 and 3000 days after diagnosis, while the distribution for censored times is less concentrated.

- 2 continuous variables: age and nodes (number of lymph nodes with detectable cancer). The mean (standard deviation) for age is 59.8 (11.9) and for nodes 3.7 (3.6).
- 2 ordinal categorical variables: differ (differentiation of tumour, with levels well, moderate and poor) and extent (extent of local spread, with levels submucosa, muscle, serosa and contiguous). The variable differ is unbalanced with most cases being at the moderate level and extent is unbalanced with most cases being at the serosa level.
- 1 nominal categorical variable: rx (treatment). The levels are observation (control group) and two different types of chemotherapy. Each group has almost the same number of observations.

The two variables, nodes and differ, have 18 and 23 missing values respectively. Rather than discarding these rows from the dataset, we substitute the missing values with the mode value for the differ variable and the median value for the nodes variable.

The dataset includes two additional variables. A binary variable indicates death or, alternatively, censoring, which is relatively balanced with 452 deaths and 477 censored observations. The last variable is a continuous survival time variable measured in days. Figure 9 displays the distribution of observed times for uncensored and censored cases separately. In the upper boxplot, we can see that the survival times are fairly centred around 2300 days, with only

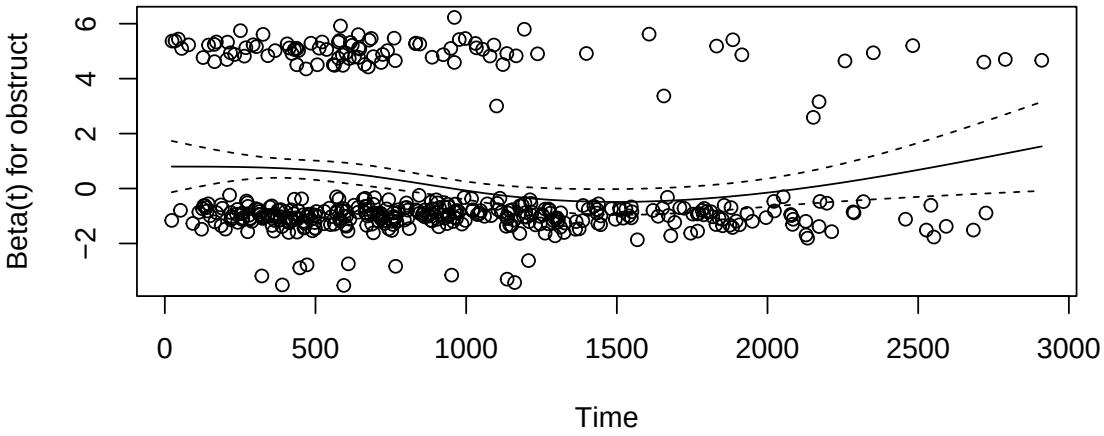

Figure 10: Test of time independence for the binary obstruct variable. The plot displays the Schoenfeld residuals for this variable, along with its estimated time-dependent coefficient and standard error. The line does not appear horizontal, which is consistent with the test outcome.

4 patients not surviving past 1500 days. In contrast, the censoring times, as shown in the lower boxplot, are more dispersed. Overall, this provides an understanding of how survival times are distributed. Most censoring times are shorter than the median survival time, as anticipated. If the opposite were true, it would be concerning since survival times would be more challenging to estimate, suggesting that the study duration might be insufficient.

Preparation We randomly divide the data into a training set, consisting of 80% of the observations, and a validation set, comprising the remaining 20%. Binary and continuous variables require no modification. However, nominal categorical variables need to be encoded as one-hot variables for the Cox regression model to be meaningful. As for ordinal variables, they can be treated as continuous variables by assigning different categories to increasing numerical values. For the `DeepHit` model, we do not recode any of the explanatory variables.

Test of proportionality assumption Initially, we fit a Cox proportional hazards model to the data. The first step is to test the proportional hazards assumption. Using the method outlined in the theory section, we determine that the null hypothesis of proportional hazards

Model	Concordance index	Integrated Brier score
CoxPH	0.494	0.083
DeepHit	**0.542**	**0.061**

Table 3: Test results on the colon cancer dataset. Concordance index and integrated Brier score measures are reported.

is globally rejected at all reasonable significance levels, with a p-value of approximately 0.0003. Moreover, we find that the **obstruct** variable violates the assumption of time-independence if we adopt the 0.05 significance level. Figure 10 provides an example of how Schoenfeld residuals may vary with time. Although we could attempt to circumvent these violations, the aim of this report is not to optimise the Cox regression model; instead, we use it for baseline comparison.

3.4.2 Performance scores

As depicted in Table 3, the `DeepHit` model outperforms the CoxPH model in terms of the selected performance measures. However, it is crucial to distinguish between the performance measure scores and the actual objectives of the models. Both models aim to minimise their respective loss functions, not the concordance index or Brier score. Since these loss functions differ, they cannot be used for direct comparison. Consequently, any chosen comparison measure will introduce some level of bias. Although the DeepHit model surpasses the Cox model in terms of the concordance index and Brier score, this does not necessarily imply that the DeepHit model is inherently superior across all dimensions. The disparities in performance could be due to the different objectives of the two models, as expressed in their loss functions.

Delving deeper into the results, it is essential to consider the context in which these models were tested. The `DeepHit` model's superior performance in this specific dataset may be due to its ability to handle complex interactions between variables and adapt to potential violations of the proportional hazards assumption, which was found to be an issue for the CoxPH model. The flexibility of neural networks, which form the basis of the `DeepHit` model, allows for better handling of nonlinear relationships and time-varying effects among covariates. This advantage could be particularly relevant in the colon cancer dataset, where the presence of both continuous and categorical variables with varying degrees of imbalance can lead to complicated interdependencies.

However, it is crucial to remember that this is a single small-data experiment in which the same fraction of data is allocated for validation and testing. The results should only be interpreted as an indication that `DeepHit` functions effectively, which has been demon-

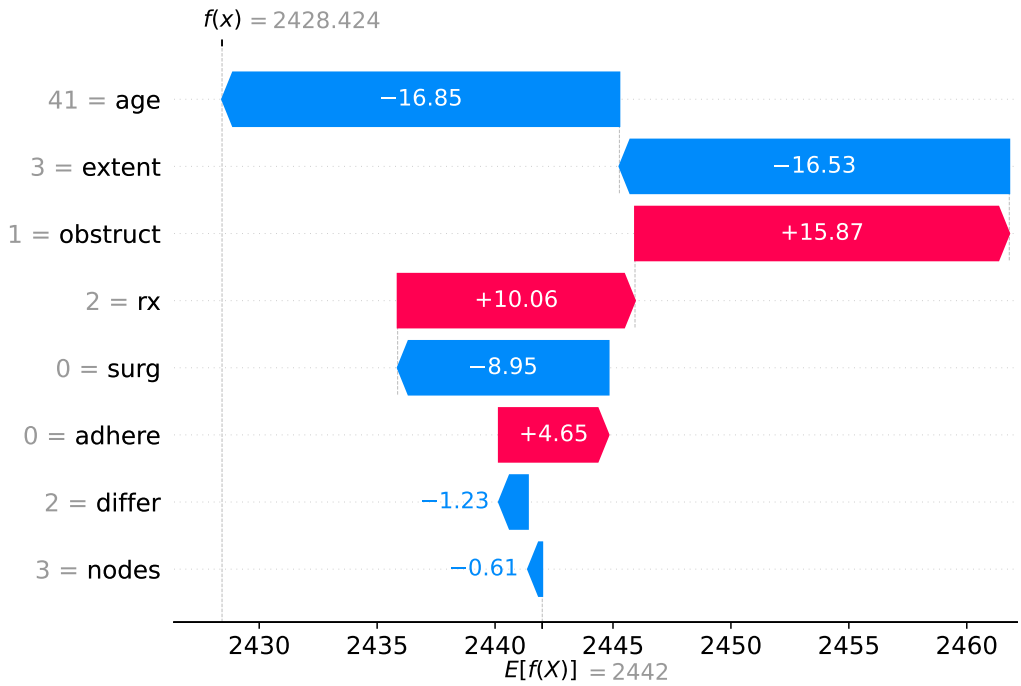

Figure 11: Illustration of how factors contribute to a single prediction made by `DeepHit`, in terms of SHAP values. The final predicted number by the model, seen at the top, is the estimated median survival time in days, depicted on the x-axis. At the bottom below the x-axis, we see the base value that would have been predicted if all covariates were set to their average value. On the y-axis, the effect of the covariates is shown along with their respective values. From the bottom and up, we see how the covariate values have an increasing impact on the output.

strated in other studies as well. The findings also suggest that `DeepHit` may outperform Cox regression in ranking individuals based on risk levels and accurately estimating survival probabilities.

3.4.3 SHAP values for interpretation

This subsection presents the results of the SHAP value experiment for the colon cancer dataset. To reduce running time, two variables with the least overall impact, as identified by Sundrani and Lu (2021), were removed. The results are displayed in several plots, encompassing both local and global interpretation tools. Some ambiguity or non-intuitive aspects of these results may arise from the approximations made in calculating the SHAP values or the inherent complexity of the model itself. Pinpointing the exact reason can be challenging.

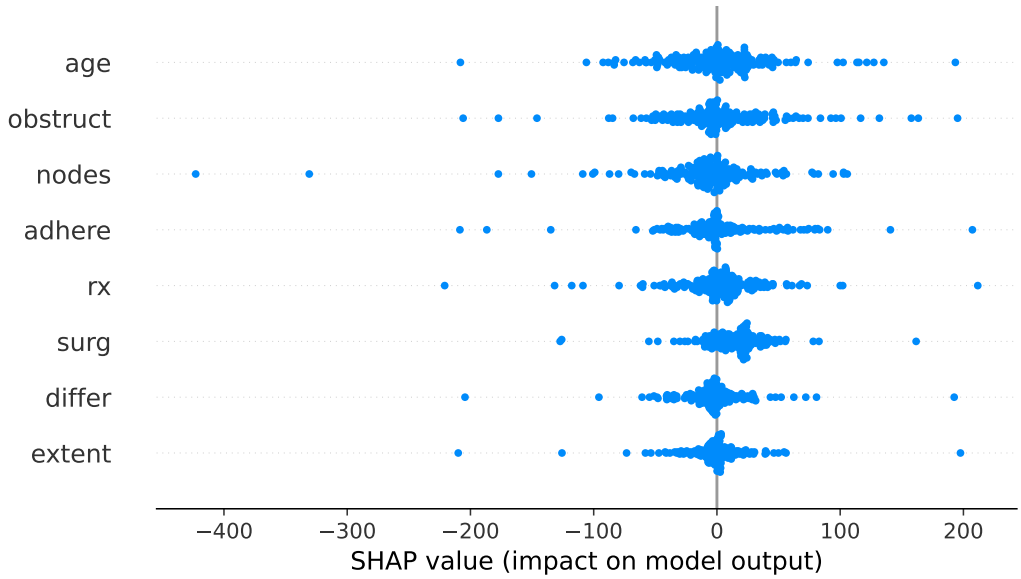

Figure 12: All calculated SHAP values illustrated in a violin plot. The thickness indicates the number of values in the same region. Each covariate has around 180 calculated SHAP values, equal to the number of observations in the validation set. Most variables have values centred around 0, but the surg variable exhibits a different shape.

We begin by demonstrating how the calculated SHAP values can be used to provide local explainability, i.e., explaining the neural network prediction for a single individual. Figure 11 illustrates how the prediction for a random individual is influenced by their covariate values. The model predicts a median survival time of around 2428 days for the individual. The overall effect of the covariates pushes the prediction to a lower value compared to the expected prediction, which is the prediction that would be made in the absence of covariates. For instance, the rx variable is at level 2, indicating that the individual belongs to the stronger chemotherapy treatment group, and it positively impacts the model prediction by increasing the median survival time by around 10 days. Meanwhile, the individual's age is 41, and it has a decreasing effect on the prediction. One might expect that for a younger individual, the age variable would instead increase the prediction.

The data reveals that the patient in Figure 11 was observed for 692 days and then censored. This information is valuable, but the model does not account for it. In the future, developing a model capable of making conditional predictions, such as predicting the survival function given that a subject has already survived for a certain number of days, would be beneficial. Furthermore, this plot particularly highlights the usefulness of employing median survival time as the prediction output. Interpreting other alternatives would be more challenging.

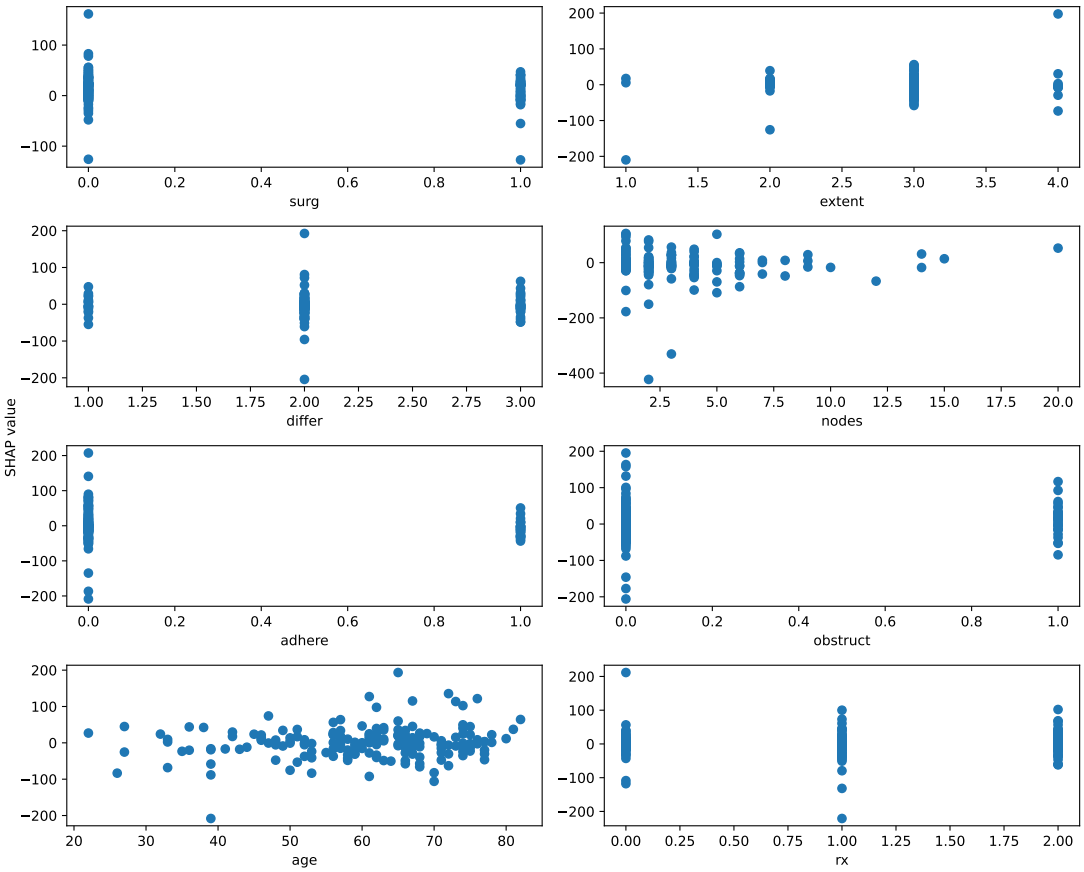

Figure 13: Scatter plot of SHAP values against their corresponding covariate value. It is hard to find any interesting patterns, but the binary variables have narrower spread for their second category.

Next, we present all the SHAP values for each explanatory variable in Figure 12 to identify potentially interesting features. Most variables appear to have a mean SHAP value close to 0, which might seem reasonable. However, it is crucial to remember that SHAP values are calculated in the presence of other explanatory variables, so this is not necessarily expected. The surg variable has its thickest part not around 0, unlike the other variables, which could be because it is binary, and one level consistently has a similar positive impact across all observations. The extent variable seems to concentrate most of its values around 0, indicating it has the least overall impact. Meanwhile, age has a larger spread, suggesting more significant contributions in general. The nodes variable has some very large negative SHAP values that appear to be outliers. The model might be extracting a pattern from the data that leads to this observation.

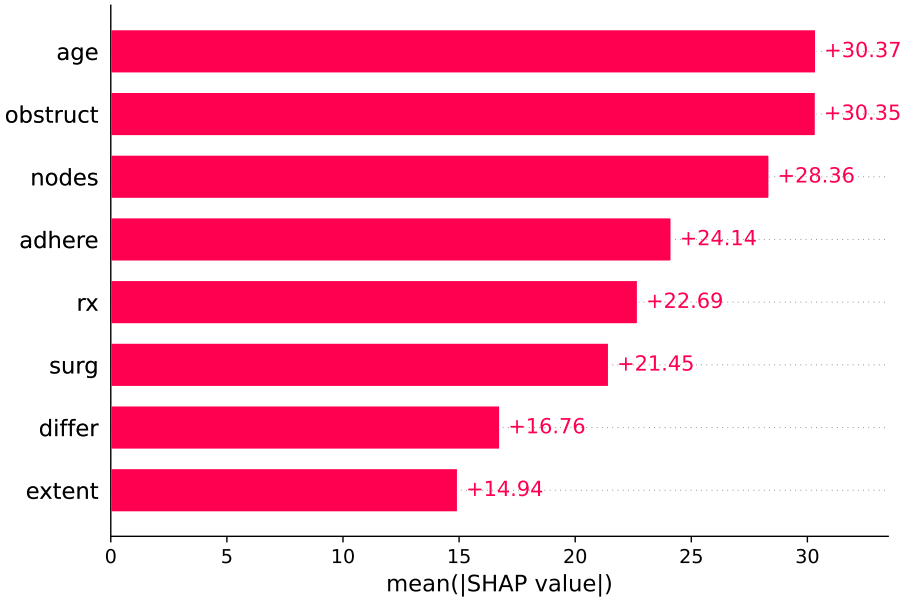

Figure 14: Mean absolute SHAP value for each covariate. This measures the overall importance of a variable to a model's predictions. According to this measure, age has the biggest impact on the output, while extent has the smallest.

As a complement to Figure 12, we plot the SHAP values against their corresponding covariate value in Figure 13. For example, we can see that for the adhere variable, all SHAP values are close to 0 when there is adherence to nearby organs concerning the patient's cancer. We can also see that the SHAP values initially decrease as the number of nodes increases, suggesting that larger values of this covariate lead to a shorter predicted median survival time. For some variables at certain levels, particularly the obstruct variable at level 0 (when there is no obstruction of the colon by the tumour), the spread is very large, suggesting that the interaction at this level with other variables is crucial in determining the impact.

Lastly, we display the mean absolute SHAP values in Figure 14. This is a global interpretation tool, aiming to measure the most impactful variables to a model's predictions overall. The idea is that the most critical variables will have large positive and negative SHAP values in general, so the average absolute value becomes large. In the figure, we can see that age and obstruct are the most important variables according to this measure, while extent and differ are the least important ones. We can also interpret that on average, the covariate value of the age variable affects the prediction by around 30 days.

4 Concluding remarks

In this thesis, we have explored the field of survival analysis, focusing on the limitations of the traditional Cox proportional hazards model and the potential advantages of adopting more advanced approaches, such as the DeepHit model. Our findings suggest that deep learning methods can offer substantial improvements in estimating the survival function, resulting in better performance across various measures. In this section, we aim to provide a summary of the main methods and results. We will also discuss the general aspects of the methods and results, including their strengths and weaknesses, as well as their relation to other studies in the field. Lastly, we will list and discuss future study directions.

Revisiting the Cox Proportional Hazards Model. We began our investigation by providing a comprehensive overview of the Cox proportional hazards model, a widely used method in survival analysis. Despite its popularity, the Cox model relies on several assumptions, including proportional hazards and a linear relationship between the covariates and the log hazard function. These assumptions, while simplifying the model, can lead to inaccurate predictions when the underlying data does not adhere to them.

The DeepHit Model: A Flexible Alternative. To address the limitations of the Cox model, we introduced the DeepHit model, a deep learning-based approach that allows for more flexibility in estimating the survival function. Through simulation experiments and a real data experiment, we demonstrated that DeepHit can outperform the Cox model according to various performance measures. This result highlights the potential of deep learning techniques in survival analysis, as they can adapt to complex relationships in the data without relying on strict assumptions. However, training deep neural networks remains a complex task and a large amount of data is usually needed to produce a model that generalises well.

Interpretability and SHAP Values. One concern with deep learning methods is their lack of interpretability, which can hinder their adoption in certain practices. To address this issue, we incorporated SHAP values into our analysis, providing a means to approximate and explain the predictions made by the DeepHit model. By illustrating the use of SHAP values in the real data experiment, we showed that the predictions of the DeepHit model can be interpreted and understood, making the model more accessible to practitioners.

Implications and Future Research. By demonstrating the superiority of the DeepHit model in certain cases, we advocate for the adoption of more sophisticated methods, such

as deep learning, that avoids making restrictive assumptions about the data for survival analysis. Furthermore, the introduction of SHAP values to improve interpretability makes these advanced models more practical for real-world applications.

There are several promising directions for future research. First, alternative deep learning architectures could be explored to further improve the performance of survival analysis models, such as recurrent neural networks (RNNs) or attention-based models. Second, the integration of other interpretability techniques, such as Local Interpretable Model-agnostic Explanations (LIME) or Counterfactual Explanations, could be investigated to enhance the understanding of the predictions made by these models. Lastly, it would be valuable to examine the performance of DeepHit and other deep learning-based survival analysis methods across a wider range of real-world datasets, assessing their generalisability and practicality in various applications.

Possibly even more promising directions could be to examine the potential for transfer learning or pre-trained models to improve the performance of deep learning-based survival analysis models, particularly in situations with limited data. Also, one could investigate the robustness of these models against potential sources of bias, such as missing or unmeasured confounders, to better understand their performance in real-world applications.

In conclusion, this thesis has shed light on the potential benefits of utilising advanced deep learning techniques in survival analysis. By comparing the Cox proportional hazards model to the more flexible DeepHit model, we have demonstrated that the latter can provide improved performance, particularly when the underlying assumptions of the Cox model are not met. Furthermore, the incorporation of SHAP values has shown that interpretability can be achieved, making deep learning methods more accessible for survival analysis practitioners. As we continue to push the boundaries of statistical modelling, embracing sophisticated techniques like deep learning will be crucial to advancing the field and unlocking the full potential of survival analysis.

5 References

Antolini, L., Boracchi, P., & Biganzoli, E. (2005). A time-dependent discrimination index for survival data. Statistics in Medicine, 24(24), 3927-3944.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Cox, D. R. (1972). Regression models and life-tables. Journal of the Royal Statistical Society: Series B (Methodological), 34(2), 187-220.

Cox, D. R., & Oakes, D. (1984). Analysis of Survival Data. Chapman & Hall/CRC.

Digitale, J., McCulloch, C., & Lange, T. (2021). Tutorial on directed acyclic graphs.

Ferguson, H. O. (1959). A method for evaluating the relative importance of factors in multivariate linear regression. Journal of the American Statistical Association, 54(286), 811-820.

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

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Grambsch, P. M., & Therneau, T. M. (1994). Proportional hazards tests and diagnostics based on weighted residuals. Biometrika, 81(3), 515-526.

Harrell, F. E., Lee, K. L., & Mark, D. B. (1982). Evaluating the yield of medical tests. Journal of the American Medical Association, 247(18), 2543-2546.

Herbert Robbins, H., & Monro, S. (1951). A stochastic approximation method. The Annals of Mathematical Statistics, 22(3), 400-407.

Hu, C., Lu, M., & Hersh, C. P. (2021). Transformer-Based Deep Survival Analysis. arXiv preprint arXiv:2106.02484.

Kaplan, E. L., & Meier, P. (1958). Nonparametric estimation from incomplete observations. Journal of the American Statistical Association, 53(282), 457-481.

Katanforoosh, K., & Kunin, A. (2019). Initializing neural networks. deeplearning.ai.

Katzman, J. L., Shaham, U., Bates, J., Cloninger, A., Jiang, T., & Kluger, Y. (2018). DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. BMC Medical Research Methodology, 18(1), 1-24.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

Kvamme, H., & Borgan, Ø. (2019). Continuous and Discrete-Time Survival Prediction with Neural Networks. arXiv preprint arXiv:1905.10403.

Kvamme, H., Borgan, Ø., & Scheel, I. (2019). Time-to-Event Prediction with Neural Networks and Cox Regression. Journal of Machine Learning Research, 20(129), 1-30.

Lee, C., Zame, W. R., Yoon, J., & van der Schaar, M. (2018). DeepHit: A Deep Learning Approach to Survival Analysis with Competing Risks. In Proceedings of the 35th International Conference on Machine Learning (ICML 2018) (pp. 3020-3029).

Ludvigsson, J. F., Otterblad-Olausson, P., Pettersson, B. U., & Ekbom, A. (2009). The Swedish personal identity number: possibilities and pitfalls in healthcare and medical research. European Journal of Epidemiology, 24(11), 659-667. doi: 10.1007/s10654-009-9350-y. Epub 2009 Jun 6. PMID: 19504049; PMCID: PMC2773709.

Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. In Advances in Neural Information Processing Systems (NIPS 2017) (pp. 4765-4774).

Miller, R. G. (2011). Survival Analysis (2nd ed.). Wiley. ISBN: 978-0-471-69252-0.

Rolf Sundberg, R. (2019). Statistical modelling by exponential families. Department of Mathematics, Stockholm University.

Sundrani, L., & Lu, M. (2021). Computing the Hazard Ratios Associated With Explanatory Variables Using Machine Learning Models of Survival Data. arXiv preprint arXiv:2102.08550.

Vittinghoff, E., Glidden, D. V., Shiboski, S. C., & McCulloch, C. E. (2004). Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models. Springer Science & Business Media.