Politecnico di Torino

POLITECNICO MILANO 1863

Doctoral Dissertation
Doctoral Program in Artificial Intelligence (37^{th} cycle)

Federated Survival Analysis

Ensemble and Neural Methods for Distributed Time-to-Event Data

Alberto Archetti

* * * * *

Supervisor
Prof. Matteo Matteucci

Politecnico di Milano
July 2025

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Alberto Archetti
Milan, July 2025

Summary

Survival analysis is a foundational branch of statistics devoted to modeling time-to-event processes, such as patient mortality or disease progression. These models are of paramount importance in medicine, where forecasting patient risk has broad implications for resource allocation, treatment customization, and clinical decision-making. Yet, the adoption of advanced machine learning and deep learning in healthcare-based survival analysis is limited by two major bottlenecks: scarcity of high-quality data and stringent privacy regulations that restrict data sharing.

Federated learning, a paradigm allowing distributed model training without centralizing sensitive datasets, offers a way to overcome both the data scarcity and privacy issues. By enabling multiple institutions to collaborate on model development without disclosing local data information, federated learning brings together large, diverse patient cohorts to boost predictive power and generalization.

In this thesis, we propose and evaluate new methods to advance survival analysis within federated settings. First, we introduce FPBoost, a fully parametric gradient boosting algorithm tailored for survival problems. FPBoost overcomes the restrictions of traditional methods by directly optimizing the full survival likelihood and modeling risk flexibly through sums of parametric distributions. We further explore polynomial-based interpolation as a postprocessing step to enhance the calibration of discrete-output neural models.

Building upon ensemble methods, we develop FedSurF, a federated extension of random survival forests, demonstrating that, through a single communication round, we can aggregate locally trained forests to produce accurate global models even in heterogeneous or data-limited scenarios. Additionally, we investigate the use of generative models to synthesize high-fidelity, privacy-preserving data, enabling large-scale validation and improved robustness of survival predictors.

Collectively, these studies reveal that careful adaptation of modern machine learning methods can yield superior discrimination and calibration of survival models, resulting in better risk assessment. With these studies, our goal is to promote a data-aware and responsible resource planning, improving patient treatments in safety-critical healthcare applications.

Acknowledgements

I would like to thank my supervisor, Prof. Matteo Matteucci, for his generous guidance, support, and encouragement during my Ph.D. journey. I am also sincerely thankful to Professors Danilo Ardagna, Giacomo Boracchi, and Massimiliano Pierobon for the valuable discussions and enjoyable collaboration over these three years.

My appreciation extends to the members of the Ph.D. examination committee for their time, effort, and thoughtful evaluation of my work. In particular, I thank Prof. Luigi De Russis, president of the committee; Professors George Chen and Kevin Xu, thesis reviewers; and Professors Pietro Ducange and Sanjay Purushotham, committee members.

I am especially grateful to my colleagues Eugenio, Diego, and Francesco for their friendship, support, and for making this experience truly memorable.

A heartfelt thank you to my family, who encouraged me to pursue research and helped me discover a career that perfectly matches my curiosity and passion for artificial intelligence.

Finally, this work was partially funded by the European Commission under the H2020 grant N. 101016577 AI-SPRINT and supported by the FAIR project (Future Artificial Intelligence Research), funded through the NextGenerationEU program (PNRR-PE-AI, M4C2, investment 1.3).

Contents

List of Tables

List of Figures

Part I

Single-Client Survival Analysis

Chapter 1

Introduction

Predicting the timing of key events has long been a cornerstone of modern statistics, with applications spanning from public health to industrial engineering. In healthcare, for instance, accurate estimates of when a patient may experience disease onset or relapse can guide treatment decisions, optimize resource allocation, and ultimately improve patient outcomes. In industrial settings, timely forecasts of machine failures enable proactive maintenance schedules, thereby minimizing downtime and reducing operational costs. Whether it involves projecting the risk of heart failure or evaluating the lifespan of mechanical components, effective estimation of event times remains crucial for advancing efficiency, safety, and utility in real-world contexts.

Motivated by the pivotal role of event-time estimation in numerous applications, this work focuses on survival analysis, a branch of statistics dedicated to modeling the time until a specified event occurs in a population [65, 111]. While survival analysis traditionally examines clinical outcomes like death, disease progression, or relapse, it works equally well for positive events such as patients awakening from comas, students graduating, or individuals getting married [21]. The methodology even extends beyond actual time measurements to abstract ordering systems, as demonstrated in inventory control applications that treat incomplete demand data as survival information [51]. Through estimating survival probabilities and rates, analysts can monitor how risks and opportunities evolve over time, enabling informed decisions about interventions and resource allocation. This capability proves invaluable for evidence-based planning, particularly in environments with constrained resources where strategic decision-making can have far-reaching consequences.

In this context, artificial intelligence has revolutionized large-scale data analysis over the past decade. Machine and deep learning algorithms have shown remarkable success in fields ranging from computer vision to natural language processing, content generation, and healthcare analytics [21, 115]. In the context of survival analysis, the flexibility of ML models allows them to capture complex nonlinear relationships and high-dimensional interactions in clinical data that traditional

statistical techniques may overlook. Deep neural networks, in particular, have demonstrated high accuracy in predicting patient outcomes from multimodal cancer electronic health records [109, 24, 56, 120], exemplifying their potential to transform healthcare through earlier and more precise prognoses.

Despite these advances, the full adoption of ML-driven survival analysis in healthcare remains hindered by significant challenges. Among them, one of the most compelling is the issue of *data privacy*: patient data are tightly regulated by frameworks such as HIPAA in the United States and GDPR in Europe, limiting the ways that information can be shared or migrated [79]. Alongside the privacy problem is the *fragmentation of data* across multiple institutions; relevant patient information is often dispersed among different hospitals, each of which may have a relatively small and heterogeneous dataset [102]. As a result, a single institution may struggle to gather sufficient data to develop robust and generalizable survival models, particularly for rare conditions or long-term outcomes.

Federated learning (FL) has emerged as a powerful paradigm for tackling both privacy and data scarcity in machine learning [87, 57]. By enabling multiple institutions to jointly train a global model without consolidating their individual datasets, FL ensures that sensitive medical records remain local. Instead, a central server coordinates the exchange of model parameters or gradients, protecting patient data from direct exposure. This collaborative learning approach has consequently been identified as a vital enabler for multi-center digital health initiatives [102].

At a high level, FL proceeds through iterative communication rounds designed to maintain both privacy and model performance. A central server typically broadcasts an initial global model—often a random or pre-trained model—to each participating site. Each site then performs local training on its own patient records using techniques such as stochastic gradient descent. Rather than transmitting any patient-level data, the site sends only its updated model parameters or gradients to the central server. These updates are aggregated (e.g., via the FedAvg algorithm [87]) to produce an improved global model, which is subsequently shared again with the sites. This *broadcast–local training–aggregation* cycle is repeated until convergence. Privacy-enhancing tools like secure multiparty computation and differential privacy can be integrated to further protect against information leakage [75, 57].

By facilitating large-scale collaboration, FL offers an opportunity to build more diverse and representative survival models. Models trained across multiple institutions capture a broader range of clinical variables, treatment protocols, and patient demographics, resulting in better performance and generalizability than models trained on isolated datasets. For instance, a federated model for predicting cancer recurrence could integrate data from oncology clinics worldwide, identifying patterns specific to certain regions or care approaches. Initial research in this area has extended classical and advanced survival methods to FL environments [80, 5, 30, 119], affirming the feasibility of federated approaches to time-to-event analysis. These studies highlight the substantial promise of federated survival analysis and

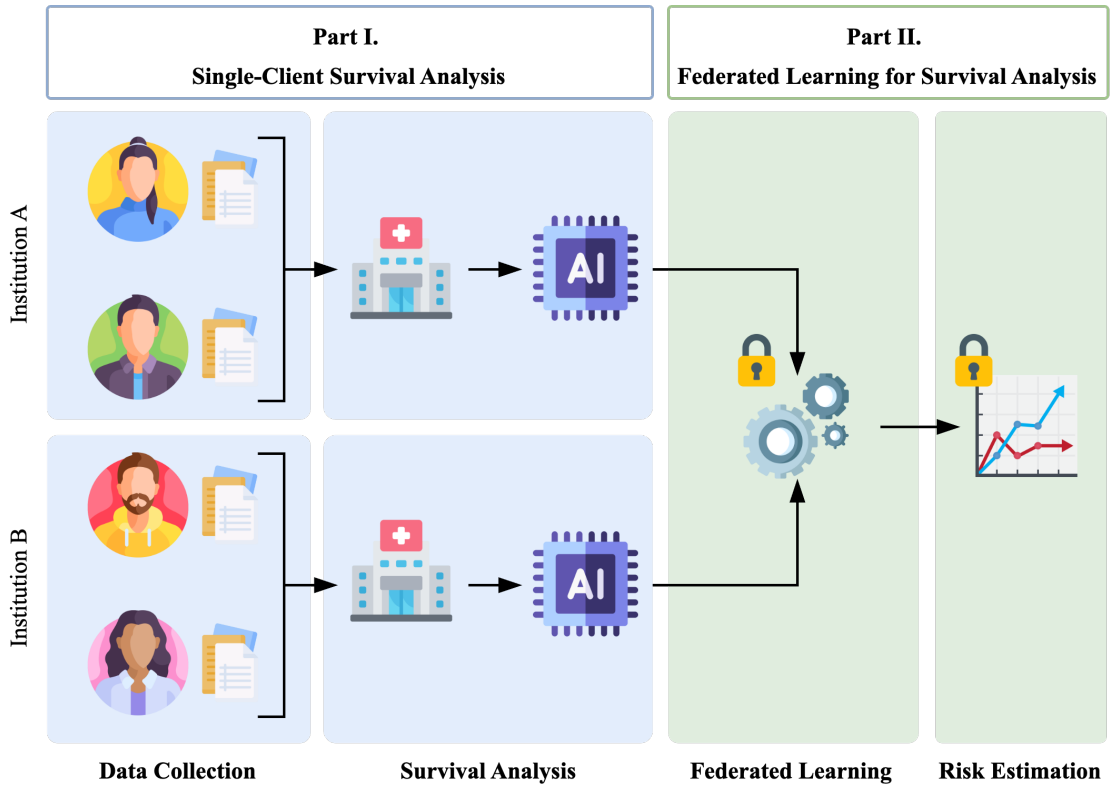

Figure 1.1: A general overview of the topics involved in this study.

underscore the need for novel algorithms capable of harnessing modern ML techniques to their fullest potential.

It is within this rapidly evolving landscape that our work is situated. Over the past three years of doctoral research, we have investigated and experimented with state-of-the-art methods drawn from both survival analysis and federated learning, aiming to push scientific progress in these domains. Grounded in an artificial intelligence background, our primary focus has been to enhance the raw algorithmic performance of various models while rigorously testing their capabilities on established benchmark datasets. Specifically, our efforts have concentrated on (i) collecting, studying, and potentially generating data from heterogeneous sources in a distributed fashion, (ii) designing and training novel survival analysis algorithms at the local institution level, (iii) incorporating these models into a federated learning framework for collaborative training across institutions, and (iv) refining the resulting risk estimates to strengthen their clinical value. An overview of these research activities is provided in Figure 1.1.

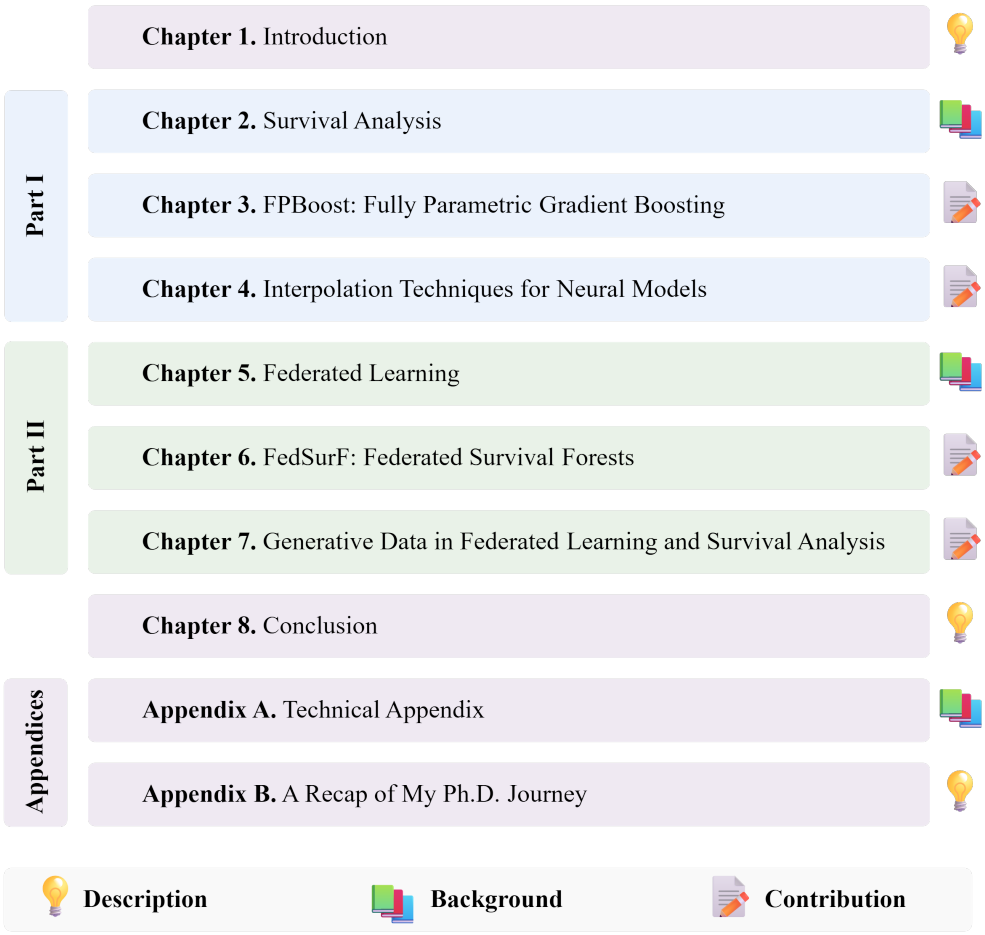

Figure 1.2: Manuscript outline, highlighting the role of each chapter.

1.1 Contributions

This section summarizes the main contributions of our doctoral studies. The thesis is divided into two parts, *Single-Client Survival Analysis* and *Federated Learning for Survival Analysis*, each consisting of three chapters. In both parts, the first chapter presents the background and current state of the art, while the subsequent two chapters detail the original research contributions. Although the studies are not presented in strict chronological order, this organization improves clarity and coherence by moving progressively from single-institution scenarios to federated, multi-institution applications. An overview of the manuscript structure is given in Figure 1.2.

Part I. Single-Client Survival Analysis. This section explores the study, comparison, and refinement of survival models for single-dataset settings. These

foundational works aim to develop robust methods capable of accurately estimating individual risk rates over time, laying the groundwork for further extension to federated environments. We begin in Chapter 2 with a comprehensive introduction to survival analysis, reviewing key concepts such as survival and hazard functions, as well as essential methods for time-to-event data, including non-parametric, semi-parametric, and advanced machine learning approaches.

Chapter 3 introduces our first contribution, FPBoost, a fully parametric gradient boosting algorithm designed to optimize the full likelihood of survival data. FPBoost combines ensembles of parametric survival distributions as base learners, delivering both flexibility and interpretability in modeling hazard functions. We show that FPBoost demonstrates robust predictive discrimination and calibration, comparable to or exceeding state-of-the-art neural and tree-based methods. A preprint of this study has been submitted for peer review [11].

Chapter 4 investigates post-processing strategies to enhance the calibration of neural network-based survival models that produce discrete survival probabilities. These interpolation techniques improve agreement between predicted and actual event probabilities without compromising discrimination. This research has resulted in both a workshop publication and a journal extension [10, 9].

Part II. Federated Learning for Survival Analysis. Shifting the focus to distributed training, Chapter 5 provides the background on federated learning (FL) in clinical contexts. We discuss the core principles of FL, the taxonomy of different federation scenarios, popular training algorithms, privacy-preserving methodologies, and key applications of FL in healthcare, with an emphasis on survival analysis.

Chapter 6 presents our federated extension of Random Survival Forests, called FedSurF. This algorithm allows multiple institutions to train survival decision trees locally and share them in a single communication round to construct a global ensemble. We demonstrate that FedSurF achieves competitive performance on both simulated federations and real-world multi-center datasets, particularly in heterogeneous environments. The research behind FedSurF has appeared in a workshop, a conference, and a journal paper [12, 8, 7].

Chapter 7 concludes our technical contributions by exploring synthetic data generation for privacy-preserving survival analysis. We introduce SGDE, an approach enabling institutions to share learned distributions via generative models rather than exposing raw patient data. SGDE was published in the context of the AI-SPRINT European project [79]. We also describe a specialized generative pipeline that selects only high-quality survival samples; the results are still unpublished, and a preview is provided here.

Finally, Chapter 8 concludes the thesis, summarizing the primary findings and how they advance the field of federated survival analysis. It also discusses directions for future work and broader implications for healthcare practice. Two appendices follow. Appendix A details the datasets used in the research and compiles the

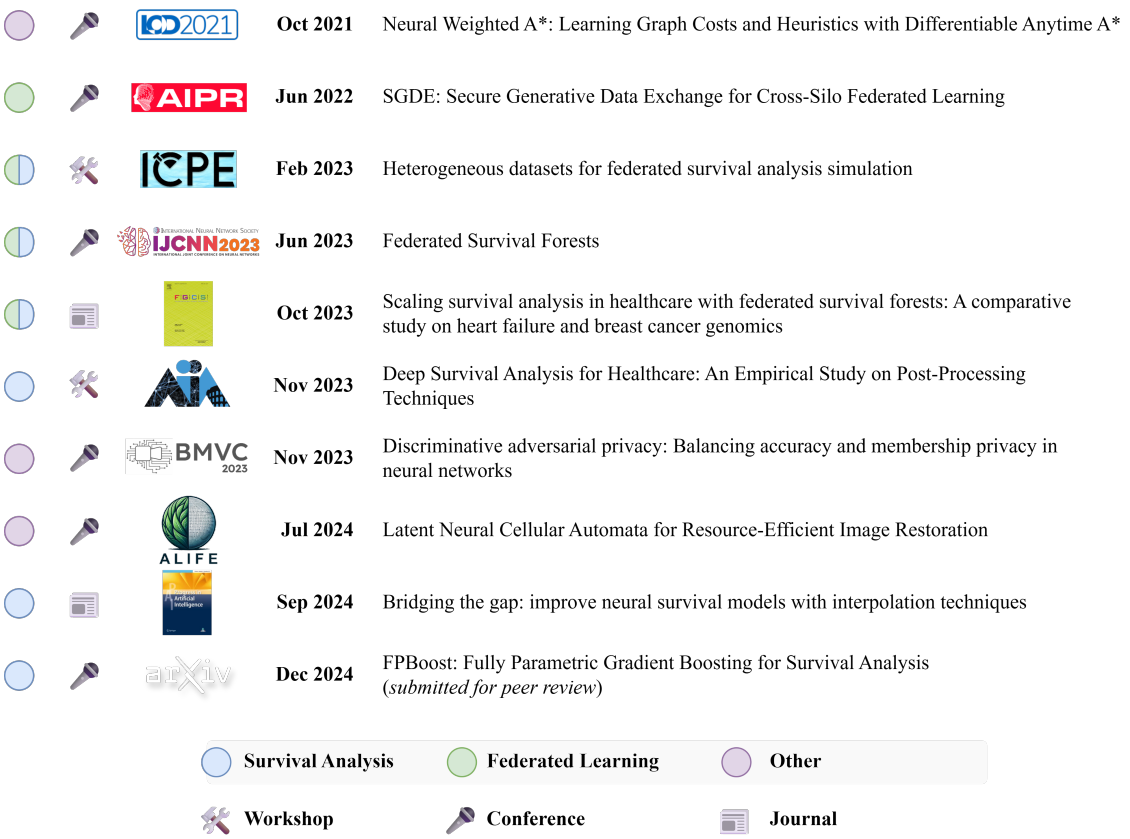

Figure 1.3: List of publications authored during the doctoral program in chronological order, categorized by main topic and publication venue.

relevant code repositories, whereas Appendix B offers an overview of additional research activities (including conference talks and teaching) undertaken during the doctoral program. Lastly, Figure 1.3 provides a chronological list of Ph.D. publications.

Chapter 2

Survival Analysis

Survival analysis is a branch of statistics designed to predict the occurrence of specific events within a given population [65, 111]. Originally developed for biomedical applications, it has since found widespread use in numerous disciplines, including engineering, economics, and social sciences. In healthcare, survival analysis is crucial for estimating patient survival probabilities, assessing disease progression, and evaluating treatment effectiveness. By analyzing data collected in clinical trials it provides a suite of statistical techniques to compare the efficacy of new drugs and treatments.

Beyond medicine, survival analysis plays an important role in industrial maintenance and reliability engineering, where it helps predict the failure times of mechanical components and optimize preventive maintenance schedules. In this context, manufacturers use survival models to estimate product lifespans and warranty policies, minimizing operational downtime and reducing costs. Similarly, in economics and finance, survival analysis informs credit risk assessment by predicting loan default probabilities and customer churn rates in subscription-based services. Labor market studies also leverage survival models to analyze employment duration and job turnover patterns.

A key advantage of survival analysis is its ability to handle censored data—observations where the event of interest has not yet occurred by the end of the study period. Unlike traditional regression models, which require complete data, survival analysis provides robust estimates even when some events remain unobserved. This characteristic makes it particularly well-suited for longitudinal studies, where data collection spans extended timeframes [73].

Survival analysis can also be extended to account for competing risks, where multiple potential events can occur. For example, in oncology, a patient may experience either cancer recurrence or death from another cause, requiring statistical models to disentangle these competing outcomes. Similarly, in the insurance industry, policyholders may either file a claim, cancel their policy, or reach the policy's maturity, necessitating ad-hoc survival techniques to model all these competing

possibilities simultaneously.

2.1 The Survival Function

The primary goal of survival analysis is to study the distribution of a non–negative random variable T that records the time until a specified event takes place. The *cumulative density function* (CDF)

$$F(t) = P(T \leq t) = \int_0^t f(u)\, du,$$

gives the probability that the event has occurred by time t. Its derivative,

$$f(t) = \frac{dF(t)}{dt} = \lim_{\Delta t \to 0} \frac{P\Big(t \leq T < t + \Delta t\Big)}{\Delta t},$$

is the *probability density function* (PDF), which quantifies the instantaneous rate at which events accumulate.

Because many applications focus on subjects that have *not* yet experienced the event—or, in other words, *survived* up to a certain time—a complementary description is provided by the *survival function*

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

This function represents the probability that the event has not occurred by time t. As t increases from 0 to ∞, $S(t)$ decreases monotonically from 1 to 0, assuming that every individual will eventually experience the event.

Another key quantity in survival analysis is the *hazard function* $h(t)$. It measures the instantaneous rate at which events occur at time t, given that the event has not occurred before t. Formally,

$$h(t) \;=\; \lim_{\Delta t \to 0} \frac{P\Big(t \leq T < t + \Delta t \;\Big|\; T \geq t\Big)}{\Delta t} \;=\; \frac{f(t)}{S(t)}.$$

Intuitively, $h(t)$ captures how the risk of event occurrence changes over time. Since $h(t)$ is not a probability, it ranges in the interval $[0, \infty)$ for all values of t. Integrating the hazard function over an interval $[0, t]$ yields the *cumulative hazard function*,

$$H(t) \;=\; \int_0^t h(u)\, du.$$

This function quantifies the total accumulation of risk up to time t.

The survival function $S(t)$ and the cumulative hazard function $H(t)$ are connected through the following key relation:

$$S(t) \;=\; \exp\Big(-H(t)\Big).$$

This relation can be derived as follows. Starting with the definition of $H(t)$:

$$H(t) \;=\; \int_0^t h(u)\,du \;=\; \int_0^t \frac{f(u)}{S(u)}\,du.$$

Notice that

$$\frac{d}{du}\Big(-\log S(u)\Big) \;=\; -\frac{1}{S(u)}\cdot\frac{d}{du}S(u) \;=\; \frac{f(u)}{S(u)}$$

because $\frac{d}{du}S(u) = \frac{d}{du}\Big(1 - F(u)\Big) = -f(u)$. Thus,

$$\frac{d}{du}\Big(-\log S(u)\Big) \;=\; h(u).$$

Integrating both sides from 0 to t gives

$$-\log S(t) + \log S(0) \;=\; \int_0^t h(u)\,du \;=\; H(t).$$

Since $S(0) = 1$, we have $\log S(0) = 0$. Hence,

$$-\log S(t) \;=\; H(t) \quad\Longrightarrow\quad S(t) \;=\; \exp\Big(-H(t)\Big).$$

This relationship is foundational in survival analysis. It provides a direct way to transition between the cumulative hazard function and the survival function. Consequently, one can design statistical or machine learning methods to model $h(t)$ or $H(t)$ and immediately obtain $S(t)$, or vice versa.

2.2 Survival Data and Censoring

In survival analysis, not all individuals experience the event of interest within the study period. When the event remains unobserved for certain subjects, their data are said to be *censored*. Censoring is a fundamental aspect of survival data and must be properly accounted for two main reasons. First, even if certain subjects did not experience the event—and thus, they are not paired with an exact label—the simple fact that they survived up to the censoring time constitutes additional information that can be exploited for learning purposes. Second, properly considering censoring probability over time is essential to ensure unbiased estimation of survival probabilities and hazard rates [65].

The most common form of censoring is *right-censoring*, which occurs when an individual has not yet experienced the event by the last recorded observation time. This situation frequently arises in clinical trials where patients are followed for a predefined period, but some remain *event-free* (i.e., censored) at the study's conclusion. In a survival dataset, each observation is represented as a triplet

$$(\mathbf{x}_i, \delta_i, t_i),$$

where $\mathbf{x}_i$ denotes a vector of features—often referred to as covariates in statistics literature—associated with the individual, t_i is the observed time (which could be either the time of event occurrence or the censoring time), and δ_i is the event indicator, taking the value of 1 if the event is observed and 0 if the observation is censored.

Beyond right-censoring, other types of censoring arise in practice. *Left-censoring* occurs when the event has already taken place before the individual enters the study, but the exact timing remains unknown. This situation is commonly encountered in retrospective studies. For example, in an occupational health study investigating the effects of asbestos exposure, a worker may already have developed a lung disease before enrollment, yet the precise onset time of the condition is unavailable. Another form, known as *interval-censoring*, arises when the event is known to have occurred within a specific time interval but the exact moment remains undetermined. This frequently happens in medical studies where patients undergo periodic health screenings. If a tumor is detected during a scheduled examination but was absent in the previous one, the exact onset of the tumor remains unknown, falling somewhere between the two visits.

A related issue in survival analysis is *truncation*, which differs from censoring in that certain individuals are systematically excluded from the study based on the timing of the event. A common form of truncation is *left-truncation*, where individuals can only enter the study if they have already survived past a certain time. This leads to a biased dataset that overrepresents longer survival times. One example is found in mortality studies using insurance data, where individuals are only included if they have survived long enough to purchase a life insurance policy. Similarly, in hospital-based studies, patients must survive long enough to be admitted for treatment, which can exclude individuals with rapidly fatal conditions.

2.3 Maximum Likelihood Estimation

In survival analysis, the event time T is typically assumed to follow a probability distribution parameterized by a set of unknown parameters Θ. The objective of maximum likelihood estimation (MLE) is to determine the values of Θ that maximize the likelihood of the observed data, ensuring that the assumed model best explains the given survival times and censoring indicators.

Given a survival dataset consisting of N triplets $\{(\mathbf{x}_i, \delta_i, t_i)\}_{i=1}^{N}$, each observation comprises a feature vector $\mathbf{x}_i$, a censoring indicator δ_i, and an observed time t_i. In many applications, the parameters Θ are not directly estimated from survival times but are instead modeled as a function of the input features $\mathbf{x}_i$. This is commonly achieved using regression-based approaches or more complex machine learning models, such as neural networks or decision trees, which learn a mapping

function ϕ from $\mathbf{x}_i$ to the parameter vector θ_i, as

$$\theta_i = \phi(\mathbf{x}_i).$$

In this framework, the observed time t_i for each subject corresponds to the minimum between the event time t_i^e and the censoring time t_i^c:

$$t_i = \min\{t_i^e, t_i^c\}.$$

The censoring indicator δ_i takes the value 1 if the event is observed ($t_i^e \leq t_i^c$) and 0 otherwise. We adopt the commonly used non-informative censoring assumption in the conditional form $(t_i^e \perp\!\!\!\perp t_i^c) \mid \mathbf{x}_i$ for every $i = 1, \ldots, n$ [45]. This assumption states that that event time and censoring time are independent, given the individual's features. This formulation is weaker than assuming independence between all event times and all censoring times in the marginal distribution, yet it is sufficient for consistent estimation of survival quantities and widely used in all models considered in this work. Under this assumption, the likelihood function [65, 70] is given by:

$$L(\Theta) = \prod_{i=1}^{N} f(t_i \mid \mathbf{x}_i)^{\delta_i} \, S(t_i \mid \mathbf{x}_i)^{1-\delta_i}.$$

The likelihood function is often reformulated in terms of the hazard function $h(t \mid \mathbf{x})$, defined as

$$h(t \mid \mathbf{x}) = \frac{f(t \mid \mathbf{x})}{S(t \mid \mathbf{x})}.$$

Substituting this into the likelihood function, we obtain:

$$L(\Theta) = \prod_{i=1}^{N} h(t_i \mid \mathbf{x}_i)^{\delta_i} \, S(t_i \mid \mathbf{x}_i).$$

To facilitate numerical optimization, it is standard practice to take the natural logarithm of the likelihood function, yielding the log-likelihood:

$$\ell(\Theta) = \sum_{i=1}^{N} \left[\delta_i \log h(t_i \mid \mathbf{x}_i) + \log S(t_i \mid \mathbf{x}_i)\right].$$

Maximizing this function is typically performed using iterative optimization techniques such as Newton-Raphson or gradient-based methods, as closed-form solutions are rarely available for complex parametric models.

Many methods discussed in the following chapters rely on learning a mapping from feature representations to distribution parameters in order to maximize the likelihood function. This approach ensures that data-driven survival estimates are grounded in statistical principles.

2.4 Discrete Survival Analysis

In many applications, event times are observed at discrete time points, or continuous survival data are discretized into fixed intervals for computational or methodological convenience. Discrete survival analysis [70] provides a framework to model event probabilities within such time intervals, allowing the application of standard classification or regression techniques, including neural networks, to estimate survival functions.

Let the time axis be divided into intervals $\{[0, \tau_1), [\tau_1, \tau_2), \ldots\}$ with event times occurring at discrete points τ_j. The PMF, denoted by $f(\tau_j)$, represents the probability that an event occurs at exactly τ_j, and is defined as

$$f(\tau_j) = P(T = \tau_j).$$

The CDF is given by:

$$F(\tau_j) = P(T \leq \tau_j) = \sum_{k=1}^{j} f(\tau_k),$$

while the survival function is

$$S(\tau_j) = P(T > \tau_j) = 1 - F(\tau_j) = \sum_{k=j+1}^{\infty} f(\tau_k).$$

The discrete hazard function, which quantifies the probability of experiencing the event at τ_j given survival until τ_{j-1}, is defined as

$$h(\tau_j) = P(T = \tau_j \mid T \geq \tau_j) = \frac{f(\tau_j)}{S(\tau_{j-1})}.$$

This leads to a recursive relationship between the survival function and the hazard function:

$$S(\tau_j) = (1 - h(\tau_j))S(\tau_{j-1}).$$

Expanding iteratively, we obtain

$$S(\tau_j) = \prod_{k=1}^{j} (1 - h(\tau_k)).$$

Finally, the cumulative hazard function, which represents the accumulation of hazard over time, is given by

$$H(\tau_j) = \sum_{k=1}^{j} h(\tau_k).$$

Following the same assumptions from time-continuous survival analysis (non-informative censoring and i.i.d. data) we can define the likelihood function for the

discrete case. Let $\{(\mathbf{x}_i, \delta_i, t_i)\}_{i=1}^{N}$ be n independent observations and $\kappa(t)$ a function returning the index j of the discrete time point τ_j related to t. Thus, the discrete survival likelihood can be expressed as

$$\begin{aligned} L(\Theta) &= \prod_{i=1}^{N} f(t_i \mid \theta_i)^{\delta}_{i} \, S(t_i \mid \theta_i)^{1-\delta_i} \\ &= \prod_{i=1}^{N} \Big(h(t_i \mid \theta_i) \, S(\tau_{\kappa(t_i)-1} \mid \theta_i) \Big)^{\delta_i} \Big((1 - h(t_i \mid \theta_i)) \, S(\tau_{\kappa(t_i)-1} \mid \theta_i) \Big)^{1-\delta_i} \\ &= \prod_{i=1}^{N} h(t_i \mid \theta_i)^{\delta_i} (1 - h(t_i \mid \theta_i))^{1-\delta_i} \prod_{j=1}^{\kappa(t_i)-1} (1 - h(\tau_j \mid \theta_i)). \end{aligned}$$

Taking the logarithm of this expression, we get

$$\ell(\Theta) = \sum_{i=1}^{N} \sum_{j=1}^{\kappa(t_i)} y_{ij} \log(h(\tau_j \mid \theta_i)) + (1 - y_{ij}) \log(1 - h(\tau_j \mid \theta_i)),$$

where $y_{ij} = 1\,(\kappa(t_i) = j, \delta_i = 1)$ is an indicator function with ones corresponding to the time intervals in which events occurred. Notably, this formulation is equivalent to a binary cross-entropy loss function for Bernoulli data, where positive samples correspond to event occurrences in the original dataset.

2.5 Survival Models

In this section, we describe the survival models included as baselines in the experimental studies covered by this work. Models categorized as non-parametric, semi-parametric, and fully parametric [111]. Each category provides different trade-offs in terms of interpretability, flexibility, and complexity. Figure 2.1 shows the taxonomy related to the models involved in this study.

2.5.1 Non-Parametric Models

Non-parametric methods make minimal assumptions on the underlying distribution of survival times, making them particularly useful for exploratory data analysis, survival function comparisons across groups, or as a preliminary step before selecting more complex modeling approaches. As shown in Figure 2.2, these methods primarily rely on aggregate survival information without explicitly modeling covariate effects, providing a simple and intuitive way to analyze survival data.

Kaplan–Meier Estimator

The Kaplan–Meier (KM) estimator [61] is the most widely used non-parametric method for estimating the survival function. It is defined as

$$S(t) = \prod_{t_i \leq t} \left(1 - \frac{d_i}{r_i}\right),$$

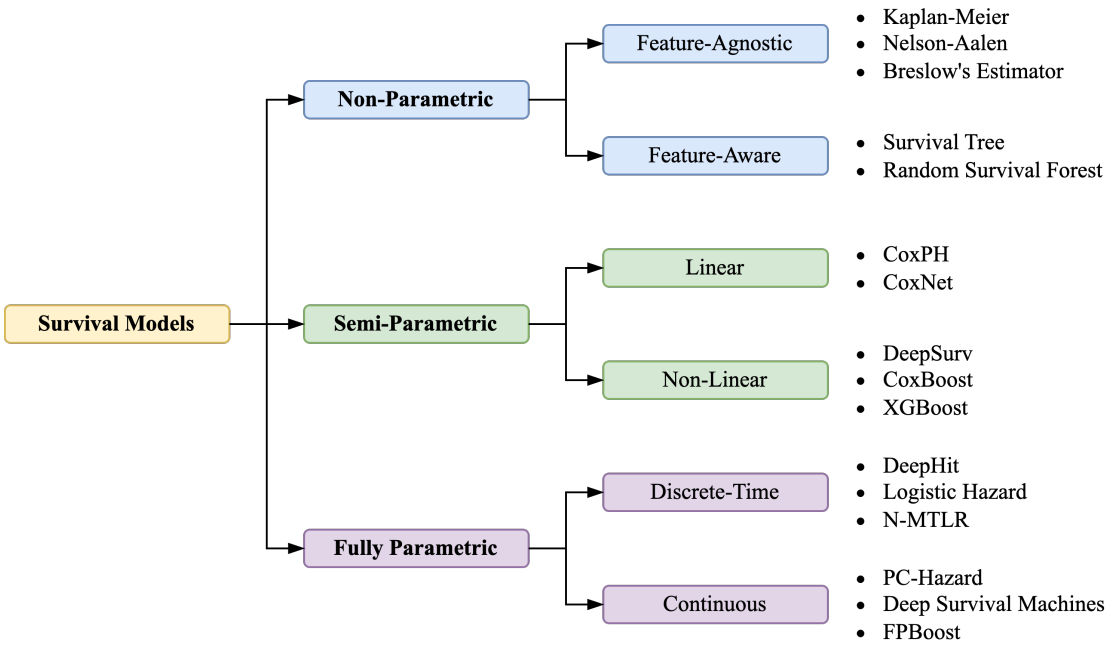

Figure 2.1: Taxonomy of survival models considered in this study.

Figure 2.2: General description of a non-parametric survival model. The survival function is the result of aggregate statistics on the input dataset, often without considering sample features.

where t_i represents the unique event times in the dataset, d_i is the number of events (e.g., deaths or failures) occurring at time t_i, and r_i is the number of individuals at risk just before t_i, i.e., those who have neither experienced the event nor been censored.

The KM estimator produces a step-wise estimate of the survival function, decreasing at each observed event time while remaining constant between events. By utilizing only event times and censoring information, it offers a non-parametric approach to visualizing survival probabilities over time. KM curves are commonly used in medical and clinical research to compare the survival probabilities of patients under different treatments, such as comparing the effectiveness of a new drug versus a placebo.

A standard method for comparing survival distributions across groups is the

log-rank test [16], a non-parametric hypothesis test that assesses whether two or more survival functions differ significantly over time. The null hypothesis of the log-rank test states that there is no difference in survival between the groups being compared.

The test statistic is computed by comparing the observed and expected number of events in each group at each event time. Specifically, at each event time t_i, the expected number of events in a given group is calculated under the assumption that the probability of experiencing the event is the same across all groups. The log-rank test statistic is then given by

$$\sum_{t_i} \frac{(O_{1i} - E_{1i})^2}{V_{1i}},$$

where O_{1i} is the observed number of events in group 1 at time t_i, E_{1i} is the expected number of events under the null hypothesis, and V_{1i} is the variance of the expected number of events. The summation runs over all observed event times. The test statistic follows an asymptotic chi-square distribution with degrees of freedom equal to the number of groups minus one. The log-rank test is widely used in clinical trials and epidemiological studies to compare the survival outcomes of different patient groups. A significant log-rank test result suggests that the survival curves differ, indicating a potential effect of the grouping variable (e.g., treatment, risk factor, etc.).

Nelson–Aalen Estimator

The Nelson–Aalen (NA) estimator [90] is a non-parametric estimator of the cumulative hazard function $H(t)$. It is given by

$$H(t) = \sum_{t_i \leq t} \frac{d_i}{r_i},$$

where t_i, d_i, and r_i follow the same definitions as in the Kaplan–Meier estimator. Both the KM and NA estimators are consistent estimators of the survival function. The KM estimator produces a step-wise estimate directly, whereas the NA estimator computes the cumulative hazard that can be transformed into a survival function. In practice, the KM estimator is more commonly reported for interpretability, but the NA estimator is often preferred when the cumulative hazard form is more convenient for subsequent modeling steps or for inference on the hazard function.

Survival Trees

Survival trees [49] extend the idea of decision trees to censored time-to-event data. They recursively partition the feature space into more homogeneous subsets

(i.e., terminal nodes or leaves), based on splitting rules that maximize heterogeneity between different nodes. A common approach for splitting nodes in survival trees is to use log-rank test. At each node, candidate splits on the features are evaluated by testing whether the resulting subgroups show significantly different survival curves. The best split is the one that maximizes the difference in survival between child nodes (e.g., largest log-rank statistic).

In survival trees, each leaf has its own survival curve, estimated using a non-parametric method like the KM or NA estimator for the samples falling into that leaf. Predictions for new data points are made by first assigning them to a leaf according to the same splitting rules and then reporting the survival function estimated for that leaf.

Random Survival Forest

The Random Survival Forest (RSF) [54] is one of the most powerful survival algorithms from the machine learning literature. Similar to Random Forests in regression and classification settings, RSF is an ensemble of survival trees built on bootstrap samples of the data. The final prediction of an RSF is the average survival rate across all trees in the forest.

Each survival tree in an RSF is typically constructed using the Classification And Regression Tree (CART) methodology [17] but adapted for survival outcomes. For each tree, a bootstrap sample is drawn from the original dataset, and at each node, a random subset of features is considered for splitting. The chosen split maximizes the log-rank statistics. To predict the survival function for a new observation, the survival curves from all trees in the ensemble are averaged, using an unweighted mean of the leaf-specific NA estimates.

Differently from regression and classification, where gradient boosting outperforms bagging in most practical applications, in survival analysis RSFs consistently show better or, at least, comparable results to boosting models.

2.5.2 Semi-Parametric Models

Semi-parametric models provide a middle ground between the flexibility of non-parametric methods and the interpretability of fully parametric models. These models do not impose a strict parametric form on the baseline hazard function or the overall survival time distribution, but they assume a parametric relationship between features and the hazard function. This allows for the estimation of feature effects while maintaining a data-driven representation of the underlying hazard structure. Figure 2.3 displays a general description of semi-parametric models.

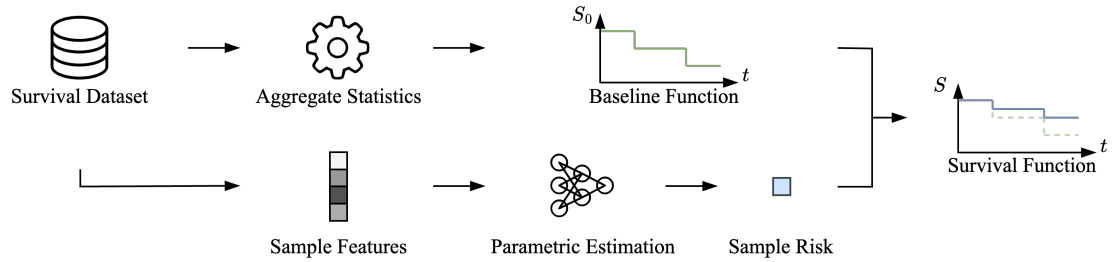

Figure 2.3: General description of a semi-parametric survival model. The survival function is the combination of a non-parametric aggregate estimate multiplied by a feature-dependent risk factor.

Cox Proportional Hazards Model

The Cox Proportional Hazards (CoxPH) model [27], is the most widely used semi-parametric approach in survival analysis due to its simplicity and efficiency. The model assumes that the hazard function for an individual with covariates $\mathbf{x}$ is given by

$$h(t \mid \mathbf{x}) = h_0(t)\, \exp(\beta^\top \mathbf{x}),$$

where $h_0(t)$ is the unspecified baseline hazard function, and $\exp(\beta^\top \mathbf{x})$ represents the relative risk associated with the covariates. Unlike fully parametric models, the Cox model does not require explicit modeling of $h_0(t)$, making it particularly useful when the shape of the baseline hazard is unknown or difficult to estimate.

A key feature of the Cox model is the proportional hazards assumption, which states that the hazard ratio between any two individuals remains constant over time. That is, if two individuals have feature vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, their hazard ratio is given by

$$\frac{h(t \mid \mathbf{x}_1)}{h(t \mid \mathbf{x}_2)} = \exp\left(\beta^\top (\mathbf{x}_1 - \mathbf{x}_2)\right),$$

which is independent of time t. This property facilitates interpretation of the β coefficients, where each component β_j represents the log-hazard ratio associated with a one-unit increase in the corresponding feature x_j.

The Cox model parameters are estimated using the partial likelihood approach, which focuses on the relative ordering of event times rather than their exact values. Given n observed failure times t_i, the partial likelihood function is expressed as

$$L_{\text{partial}}(\beta) = \prod_{i=1}^{N} \frac{\exp(\beta^\top \mathbf{x}_i)}{\sum_{j \in R(t_i)} \exp(\beta^\top \mathbf{x}_j)},$$

where $R(t_i)$ denotes the risk set, i.e., the set of individuals still at risk just before time t_i. This formulation optimizes the relative hazard contributions, assigning higher relative risk estimates to individuals experiencing the event earlier than others

still at risk. Maximum likelihood estimation of β is typically performed via iterative methods such as Newton-Raphson or gradient descent. Extensions of the Cox model include regularization terms in the loss formulation, such as CoxNet [106].

Once the regression coefficients β are estimated, the baseline hazard function $h_0(t)$ can be recovered non-parametrically. A common approach is the Breslow estimator [18] for the cumulative baseline hazard:

$$H_0(t) = \sum_{t_i \leq t} \frac{d_i}{\sum_{j \in R(t_i)} \exp(\beta^\top \mathbf{x}_j)},$$

where d_i is the number of events occurring at time t_i. The baseline hazard $h_0(t)$ is then obtained by differentiating $H_0(t)$ with respect to t using finite differences.

Despite its popularity, the Cox model relies on the proportional hazards assumption, which introduces an inductive bias that can be beneficial in small and simple datasets, but it may constrain model generalization in real-world settings. Consequently, if this assumption is violated in practice, alternative non-linear approaches, may be required to ensure more robust and reliable survival estimates.

DeepSurv

DeepSurv [63] generalizes the Cox model by replacing the linear predictor $\beta^\top \mathbf{x}$ with a neural network $\phi(\mathbf{x}; \theta)$. The hazard is thus

$$h(t \mid \mathbf{x}) = h_0(t) \, \exp\Big(\phi(\mathbf{x}; \theta)\Big).$$

The network parameters θ are estimated by minimizing the negative partial likelihood loss adapted to neural networks:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \delta_i \left(\phi(\mathbf{x}_i; \theta) - \log \sum_{j \in R(t_i)} \exp\Big(\phi(\mathbf{x}_j; \theta)\Big) \right),$$

where δ_i is the event indicator for individual i. This approach allows for flexible, non-linear effects of covariates while maintaining the proportional hazards assumption. As for the RSF model, DeepSurv consistently shows extremely competitive results in practice, even when compared to more complex neural network architectures. Thus, it strikes an optimal balance between the non-linear flexibility of feature engineering and the survival bias introduces by the proportional hazard assumption.

Boosting Approaches

Boosting-based approaches offer a powerful alternative to traditional survival models by iteratively refining predictions through the aggregation of multiple weak learners [41, 38]. Instead of modeling the relationship between subject features

x and survival outcomes using a single-output neural network, boosting methods adaptively enhance model performance by focusing on misclassified instances or high-loss observations at each iteration.

One of the most common boosting techniques for survival analysis is CoxBoost [101], which extends the Cox proportional hazards model by incorporating boosting into its estimation process. CoxBoost iteratively updates the model by adding weak learners—typically linear base learners for each feature—aiming to optimize the partial log-likelihood function. This approach is particularly advantageous for high-dimensional datasets, such as genomic data in medical research, where the number of features far exceeds the number of observations.

Beyond CoxBoost, more advanced gradient boosting techniques have been adapted for survival analysis by incorporating tree-based learners. One of the most prominent is XGBoost [25], which provides a scalable and highly efficient alternative to CoxBoost. Other boosting frameworks extended for survival analysis are LightGBM [64], which is known for its efficiency in handling large-scale datasets with sparse features and CatBoost [96], which is optimized for categorical data. Despite their advantages, boosting models require careful hyperparameter tuning and can be computationally intensive, especially for large datasets. Furthermore, translating the complex, non-linear decision boundaries learned by such ensembles into explanations that are as transparent as those of the Cox proportional-hazards model remains non-trivial. Model-agnostic post-hoc explanation frameworks, most notably SHAP (SHapley Additive exPlanations) [82], extend beyond tree-based learners and can be applied to neural networks, support-vector machines, and gradient-boosted survival forests by estimating the marginal contribution of each feature to the predicted risk for every individual. Nevertheless, a growing body of work highlights fundamental limitations: explanations can disagree across reasonable parametrizations [69], are sensitive to correlated or out-of-distribution inputs [43], and may say little about causal drivers of a decision unless combined with counterfactual reasoning [39].

2.5.3 Fully Parametric Models

Fully parametric models assume a specific parametric form for the baseline distribution of survival times or the hazard function. Traditional survival analysis approaches often model survival functions using well-known parametric distributions, such as the exponential, Weibull, log-normal, and log-logistic distributions [65]. Similar to semi-parametric estimators, these models provide sample-level survival estimates but with the added advantage of a fully specified distribution, allowing for extrapolation of complex patterns beyond aggregate statistics.

While classical parametric models offer a structured and interpretable framework, more advanced techniques leverage neural networks and non-linear parameter estimation to model survival functions in the presence of complex feature interactions. These advanced approaches, which form the focus of this study, aim to enhance

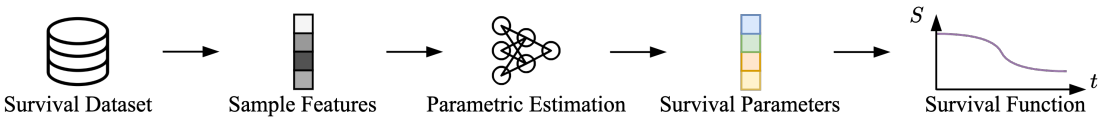

Figure 2.4: General description of a fully parametric survival model. The survival function is entirely described by a finite set of discrete or continuous values.

predictive performance by capturing non-linear dependencies between features and survival outcomes. Figure 2.4 shows the general steps behind fully parametric survival estimation.

Logistic Hazard

The Logistic Hazard model [70] is based on a discrete-time survival framework, where the objective is to estimate the conditional probability of an event occurring within each predefined time interval. Specifically, let the survival time T be discretized into intervals $[\tau_k, \tau_{k+1})$. The model then estimates the discrete hazard function as:

$$h(\tau_k \mid \mathbf{x}) = P(T \in [\tau_k, \tau_{k+1}) \mid \mathbf{x}),$$

which represents the probability that the event occurs within the given interval, given the covariates $\mathbf{x}$.

To model this probability, one common approach is to use a parametric function such as a linear regression model or a neural network, followed by a sigmoid activation function to ensure that the output remains within the range [0,1]. The parameters of the model are estimated by maximizing the Bernoulli likelihood across all intervals, treating the occurrence of the event in each interval as a binary classification problem.

DeepHit

DeepHit [72] is a deep learning-based survival analysis model designed to estimate the distribution of survival times. A key feature of DeepHit is its ability to handle *competing risks*—scenarios where multiple event types are possible for an individual, such as different causes of death in medical applications. Competing risks introduce significant challenges in survival analysis. In fact, all the models analyzed so far assume that each individual is subject to only one type of event, treating other event types as censoring. However, in reality, different event types may be correlated, and ignoring this dependence can lead to biased estimates. For example, in oncology, a patient may die from cardiovascular disease before cancer progression, complicating the estimation of cancer-specific survival probabilities.

DeepHit addresses this issue by jointly modeling survival distributions across multiple competing risks with a multi-task neural network architecture. This

architecture is composed of a shared feature extraction network and a set of event-specific subnetworks. The shared network processes input features $\mathbf{x}$ through multiple fully connected layers to learn a latent representation. This shared representation captures general patterns that are relevant across all competing risks. Then, each competing risk k has a dedicated subnetwork that takes as input both the shared representation and the original features. This structure allows each subnetwork to learn risk-specific representations while leveraging common information. Finally, a softmax output layer produces a probability distribution over discrete survival times for each competing risk. The output consists of probabilities $y_{k,s}$ representing the likelihood that an individual with covariates $\mathbf{x}$ will experience event k at time s. This way, for each event type $k \in \{1, \ldots, K\}$ and discrete survival time s, the network outputs:

$$y_{k,s} = P(T = s, K = k \mid \mathbf{x}),$$

where T is the survival time and K is the event type.

DeepHit is trained using a composite loss function designed to handle censored data and encourage event-time ranking. The total loss function is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_1 + \mathcal{L}_2,$$

where

- $\mathcal{L}_1$ is the log-likelihood loss, which maximizes the likelihood of the observed event-time pairs while accounting for censoring. For uncensored instances, it encourages correct classification of the observed survival time and event type. For censored instances, it ensures that the cumulative incidence function accounts for all potential event types:

$$\mathcal{L}_1 = -\sum_{i=1}^{N} \left[1(k_i \neq \emptyset) \log y_{k_i,s_i} + 1(k_i = \emptyset) \log \left(1 - \sum_{k=1}^{K} F_k(s_i \mid \mathbf{x}_i) \right) \right],$$

  where $F_k(s \mid \mathbf{x})$ is the estimated cumulative incidence function for event k.

- $\mathcal{L}_2$ is a ranking loss, which ensures that patients experiencing an event at earlier times are assigned higher risk scores than those who survive longer. Inspired by the concordance index, this loss compares pairs of patients who experienced the same event at different times and penalizes incorrect rankings:

$$\mathcal{L}_2 = \sum_{k=1}^{K} \alpha_k \sum_{i \neq j} A_{k,i,j} \eta(F_k(s_i \mid \mathbf{x}_i), F_k(s_i \mid \mathbf{x}_j)),$$

  where $A_{k,i,j}$ is an indicator function for valid patient pairs and $\eta(x, y)$ is a convex ranking loss function, e.g., an exponential function. For simplicity, α_k is often set to the same value α for all K event types.

Neural Multi-Task Logistic Regression

The Neural Multi-Task Logistic Regression (N-MTLR) [40] is an extension of the MTLR model [118], by substituting linear heads with neural networks. Similarly to LogisticHazard and DeepHit, this model is based on the discrete survival framework. Specifically, it starts by partitioning time into B intervals $[\tau_0, \tau_1), \ldots, [\tau_{J-1}, \infty)$. A neural mapping $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_B(\mathbf{x}))$ defines

$$f\Big(\tau_j \mid \mathbf{x}\Big) = \frac{\exp\Big(\sum_{b=j}^{J-1} \phi_b(\mathbf{x})\Big)}{\sum_{b=1}^{J} \exp\Big(\sum_{b'=b}^{J-1} \phi_{b'}(\mathbf{x})\Big)}, \quad S(\tau_{j-1} \mid \mathbf{x}) = \sum_{b=j}^{J} f\Big(\tau_b \mid \mathbf{x}\Big).$$

The model is trained minimizing the negative loglikelihood loss as

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \Big[\delta_i \, \ln f\Big(\tau_i \mid \mathbf{x}_i\Big) + \Big(1 - \delta_i\Big) \, \ln S\Big(\tau_{i-1} \mid \mathbf{x}_i\Big)\Big].$$

Piecewise Constant Hazard Model

The Piecewise Constant Hazard (PC-Hazard) model [70] is a continuous-time survival model that parametrizes the hazard function as a step function by partitioning the time axis into intervals and assuming constant hazard within each interval. This approach is particularly useful in scenarios where survival times exhibit non-monotonic hazard trends that are difficult to capture with standard parametric models.

Consider a time partition $0 = \tau_0 < \tau_1 < \cdots < \tau_m = \tau$, where each time interval $(\tau_{k-1}, \tau_k]$ is associated with a constant hazard. Let $\kappa(t)$ denote the interval index such that $t \in (\tau_{\kappa(t)-1}, \tau_{\kappa(t)}]$. Thus, the hazard function is defined as

$$h(t) = \eta_{\kappa(t)},$$

where $\eta_{\kappa(t)}$ represents a set of non-negative constants representing the hazard rates within each interval. The cumulative hazard function can be expressed as:

$$H(t) = \sum_{j=1}^{\kappa(t)-1} \eta_j \Delta\tau_j + \eta_{\kappa(t)}(t - \tau_{\kappa(t)-1}),$$

where $\Delta\tau_j = \tau_j - \tau_{j-1}$ is the width of each interval.

The likelihood contribution for an individual i with event time t_i and event indicator δ_i is given by:

$$L_i = h(t_i)^{\delta_i} \exp[-H(t_i)] = \eta_{\kappa(t_i)}^{\delta_i} \exp\Big[-\eta_{\kappa(t_i)}(t - \tau_{\kappa(t)-1})\Big] \prod_{j=1}^{\kappa(t_i)-1} \exp[-\eta_j \Delta\tau_j].$$

The model introduces a reparameterization of the hazard values as

$$\tilde{\eta}_j = \eta_j \Delta\tau_j,$$

which allows the likelihood to be rewritten in terms of $\tilde{\eta}_j$ rather than directly estimating the interval boundaries τ_j, simplifying optimization. PC-Hazard utilizes a neural network with a softplus activation function to learn these values dynamically. The network is trained by minimizing the mean negative log-likelihood across all individuals as

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left(\delta_i \log \tilde{\eta}_{\kappa(t_i)}(\mathbf{x}_i) - \tilde{\eta}_{\kappa(t_i)}(\mathbf{x}_i)\rho(t_i) - \sum_{j=1}^{\kappa(t_i)-1}\tilde{\eta}_j(\mathbf{x}_i)\right),$$

where

$$\rho(t) = \frac{t - \tau_{\kappa(t)-1}}{\Delta\tau_{\kappa(t)}}$$

represents the proportion of time elapsed within the current interval. This construction shares many similarities with the Logistic Hazard model, particularly in its use of piecewise hazards. However, while Logistic Hazard discretizes time into predefined intervals and models event probabilities as a classification problem, PC-Hazard uses observed event times and censoring times to define the time partition dynamically. This distinction allows PC-Hazard to more naturally extend to settings where a fine-grained continuous risk estimation is relevant.

Deep Survival Machines

Deep Survival Machines (DSM) [89] is a fully parametric survival algorithm that models time-to-event distributions as a mixture of parametric distributions, allowing it to handle censored data and competing risks. DSM is built upon a deep neural network architecture that estimates the conditional survival distribution as a weighted mixture of K parametric distributions. The main components of the model are:

- A shared feature extraction network, as in DeepHit, which maps the input covariates $\mathbf{x}$ to a latent representation $\phi(\mathbf{x})$ using a fully connected neural network. This shared network captures the underlying structure of the input data.
- Mixture component networks, where each mixture component $k \in \{1, \ldots, K\}$ represents a parametric survival distribution (e.g., Weibull or Log-Normal). The parameters of each distribution (scale and shape) are estimated using event-specific sub-networks.

- Mixture weight network, which assigns probabilities w_k to each component, representing the probability that an individual follows a specific parametric survival distribution.

Thus, given input features $\mathbf{x}$, DSM estimates the survival distribution as

$$P(T > t \mid \mathbf{x}) = \sum_{k=1}^{K} w_k(\mathbf{x})\, S_k(t \mid \mathbf{x}),$$

where $S_k(t \mid \mathbf{x})$ is the survival function of the k-th mixture component. For each mixture component, DSM models each parameter θ as:

$$\theta = \theta^0 + \mathrm{act}(\zeta^\top \phi(\mathbf{x})),$$

where $\phi(\mathbf{x})$ is the feature representation learned by the neural network, ζ is a learnable parameter, and $\mathrm{act}(\cdot)$ is an activation function (e.g., SELU or Tanh) ensuring appropriate parameter ranges. DSM is trained using a likelihood-based loss function called ELBO loss with a regularization term to mitigate long tail bias. For competing risks, DSM extends its formulation by associating separate mixture models with each competing event type. Let K_j be the number of mixture components for event type j. Then, the common intermediate representation $\phi(\mathbf{x})$ interacts with distinct parameters θ_k for each competing event, while treating the occurrence of one event before another as independent censoring.

2.6 Survival Metrics

Evaluating the performance of survival models requires metrics that properly account for right-censoring and measure both *discrimination* and *calibration*. Discrimination refers to how well a model can distinguish between individuals who experience an event earlier versus later. Calibration assesses how closely the predicted survival probabilities align with actual outcome frequencies. This section reviews common survival metrics, their definitions, and their practical usage for model validation and prediction.

2.6.1 Concordance Index

The concordance index (C-Index) [108] extends the concept of the area under the ROC curve to survival analysis. It measures the discriminatory power of a model by quantifying the pairwise agreement between predicted risk and observed event times. Specifically, the model prediction is said to be *concordant* for a pair (i, j) if the individual with the shorter survival time has a higher predicted risk. Let t_i and t_j be the observed times for individuals i and j, δ_i and δ_j their event indicators, and r_i, r_j their predicted risks. The risk can be defined as the mean or

median survival time for general survival models, or the multiplicative risk factor for semi-parametric models. The C-Index focuses only on *comparable* pairs, meaning (i, j) is considered comparable if $t_i < t_j$ and $\delta_i = 1$. The C-Index is defined as

$$\text{C-Index} \;=\; \frac{\sum_{i,j} 1\Big(t_i < t_j \,,\, \delta_i = 1 \,,\, r_i > r_j\Big)}{\sum_{i,j} 1\Big(t_i < t_j \,,\, \delta_i = 1\Big)},$$

where $1(\cdot)$ is an indicator function that is 1 if the condition is met and 0 otherwise. A C-Index of 1.0 indicates perfect ranking of survival times according to predicted risk, whereas 0.5 corresponds to random prediction. Because it relies on pairwise comparisons, the C-Index emphasizes *discrimination*—namely, how effectively the model ranks individuals.

2.6.2 Integrated Brier Score

The Brier score [44] is a mean squared error measure for predicted survival probabilities. It captures both discrimination and calibration by comparing predicted survival $S(t \mid \mathbf{x}_i)$ against actual outcomes at time t. For an individual i, let $1(t_i \le t, \delta_i = 1)$ indicate whether the event occurred by t. Then the Brier score at time t over N individuals is

$$\text{BS}(t) \;=\; \frac{1}{N} \sum_{i=1}^{N} w_i(t) \Big(1(t_i \le t, \delta_i = 1) \;-\; S(t \mid \mathbf{x}_i)\Big)^2,$$

where $w_i(t)$ is a weight to adjust for censoring, typically derived from inverse probability of censoring weights (IPCW) [103, 108]. In this case,

$$w_i(t) \;=\; \frac{1(t_i \le t)}{G(t_i)}, \qquad G(t) \;=\; \prod_{t_i \le t} \Big(1 - \frac{d_i^c}{r_i^c}\Big),$$

with $G(t)$ the Kaplan–Meier estimate of the censoring distribution, d_i^c the number of censored events at time t_i, and r_i^c the number at risk just before t_i. Lower Brier scores indicate that predicted survival probabilities are closer to the observed outcomes, implying better calibration and discrimination jointly. The *Integrated Brier Score* (IBS) then averages the Brier score over a time horizon τ:

$$\text{IBS} \;=\; \frac{1}{\tau} \int_0^{\tau} \text{BS}(t)\, \mathrm{d}t,$$

providing a single statistic that summarizes overall predictive accuracy across time.

2.6.3 Cumulative AUC

The time-dependent area under the curve (AUC) [52] evaluates discrimination at specific time points. For a given time t, it measures the probability that an

individual who fails before t has a higher predicted risk than an individual who remains event-free beyond t. Formally,

$$\text{AUC}(t) \;=\; P\Big(S(t \mid \mathbf{x}_i) < S(t \mid \mathbf{x}_j) \;\Big|\; t_i \le t,\; t_j > t\Big).$$

This definition dynamically updates the risk sets over time, allowing the discrimination ability of the model to be assessed at each time point. A higher time-dependent AUC indicates stronger separation between those who fail before t and those who survive past t.

A summary measure for dynamic AUC is the *cumulative AUC*, which aggregates time-dependent AUC values over a pre-specified time horizon τ to provide an overall assessment of discrimination across time. As in the IBS case, it is computed as

$$\text{Cumulative AUC} = \frac{1}{\tau}\int_0^\tau \text{AUC}(t)\,\mathrm{d}t.$$

and it can be weighted with by the inverse censoring probability.

2.6.4 Metrics Discussion

The C-Index focuses primarily on ranking individuals by risk and thus addresses discrimination without directly evaluating the accuracy of survival probability estimates. In contrast, the Brier score (and IBS) simultaneously assesses discrimination and calibration by penalizing models that predict survival probabilities inconsistent with observed outcomes. The time-dependent AUC provides a more detailed view of discrimination at different time points, making it especially valuable when interest lies in dynamic risk prediction. In practice, these metrics are often used in combination: the C-Index captures whether the ordering of risk is appropriate, while the Brier score and integrated variants quantify how well predicted probabilities match actual rates of survival, and the dynamic AUC reveals time-specific discrimination. Only together they can provide a comprehensive assessment of model performance [111].

Chapter 3

FPBoost: Fully Parametric Gradient Boosting

The first portion of this thesis focuses on single-client survival analysis, without incorporating federated learning. In particular, our goal is to improve survival models based on machine and deep learning techniques, maximizing the downstream concordance and calibration on benchmark datasets. Specifically, in this chapter, we present FPBoost, a novel survival method developed by leveraging the most interesting empirical insights from our investigations over the past three years of work [11].

Two main trends emerge from the survival modeling literature. First, tree-based bagging methods—especially Random Survival Forests (RSFs)—consistently outperform many existing approaches in our experiments, often surpassing boosting algorithms. This finding stands in contrast to standard classification and regression problems, where boosting methods (e.g., XGBoost, CatBoost, LightGBM) are typically the only rivals of deep neural networks on large tabular datasets. We attribute this discrepancy to the fact that boosting in the survival context is typically implemented within semi-parametric frameworks, employing simplified losses such as the partial likelihood. This restriction may hinder the practical effectiveness of boosting to a greater extent than simpler, assumption-free bagging methods such as RSFs.

Second, none of the existing gradient-based approaches—whether neural or tree-based—directly optimize the full survival likelihood. Even Deep Survival Machines relies on an ELBO loss derived from the variational inference framework. Motivated by these observations, we started to develop a survival method that applies gradient boosting to the complete survival likelihood. Our objective was to synthesize the advantages of a statistically principled maximum likelihood method with the expressive capabilities of gradient-boosted decision trees. This chapter presents the resulting algorithm, referred to as Fully Parametric Gradient Boosting, or, in short, FPBoost.

FPBoost is a decision tree-based ensemble model designed to provide accurate and fine-grained survival estimates without resorting to restrictive assumptions such as the proportional hazards requirement in semi-parametric models or time discretization in neural networks. The core idea of FPBoost is to represent the predicted hazard as a weighted sum of standard parametric hazard functions, where the parameters are learned via gradient boosting. By leveraging closed-form hazard functions, FPBoost benefits from a tractable likelihood, facilitating effective optimization with gradient-based methods. Moreover, constructing the hazard as a sum of simpler distributions preserves the desirable bagging-like behavior of RSFs while leveraging on the generalization power inherent to boosting.

3.1 Model Architecture

FPBoost constructs a global hazard function for each individual through a weighted sum of parametric functions, referred to as heads, where each head represents a known survival distribution. By combining multiple heads, FPBoost is capable of approximating general hazard patterns within bounded precision, provided the model includes a sufficient number of heads. Without loss of generality, in this work we consider Weibull and LogLogistic parametric families [65]. Specifically, the Weibull distribution is formulated as

$$h(t \mid \eta, k) = \eta k t^{k-1}, \quad H(t \mid \eta, k) = \eta t^k$$

and the LogLogistic distribution is given by

$$h(t \mid \eta, k) = \eta k t^{k-1}/(1 + \eta t^k), \quad H(t \mid \eta, k) = \log(1 + \eta t^k).$$

For a subject with covariates $\mathbf{x}$, let J denote the total number of heads. Each head $j \in \{1, \ldots, J\}$ outputs a set of parameters $\theta_j(\mathbf{x})$, which includes shape $k_j(\mathbf{x})$ and scale $\eta_j(\mathbf{x})$ for Weibull or LogLogistic distributions. A trainable weight $w_j(\mathbf{x})$ is also assigned to each head. Collecting the parameters and weights into $\Theta(\mathbf{x})$, the overall hazard is given by

$$h\Big(t \mid \Theta(\mathbf{x})\Big) \;=\; \sum_{j=1}^{J} w_j(\mathbf{x})\, h_j\Big(t \;\mid\; \eta_j(\mathbf{x})\,,\, k_j(\mathbf{x})\Big),$$

where $h_j(\cdot)$ denotes the hazard function for the j-th parametric distribution.

Any parametric survival distribution can, in principle, serve as a head within FPBoost. However, we focus on the Weibull and LogLogistic families because they can capture a wide range of real-world failure patterns. Specifically, the Weibull distribution typically models risks that increase over time (e.g., aging or wear), whereas the LogLogistic distribution can capture early failures. Summing multiple

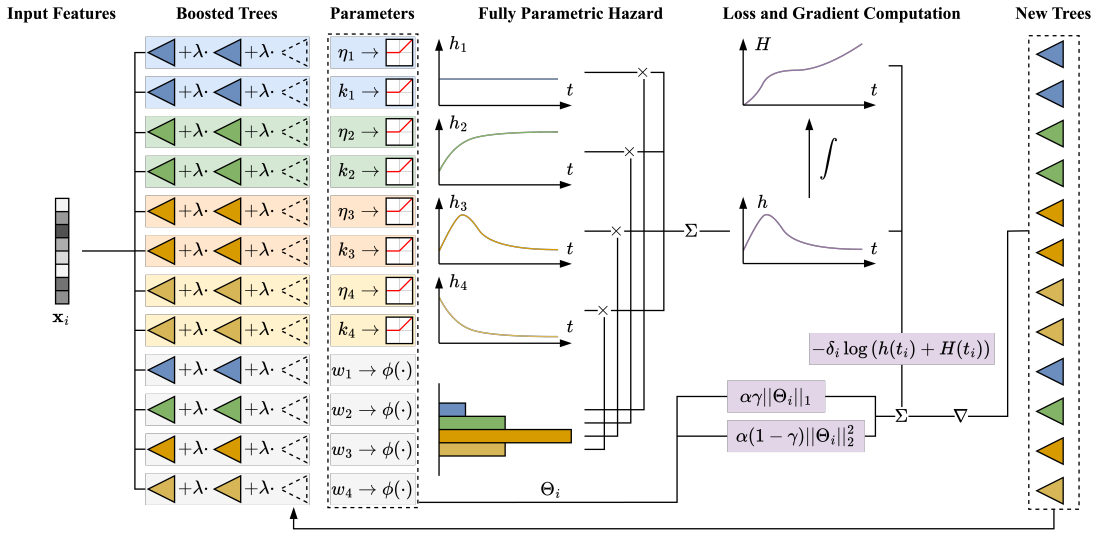

Figure 3.1: FPBoost example architecture with four heads. Starting from the input features $\mathbf{x}$, 12 sets of gradient-boosted trees estimate a scale η_j, a shape k_j, and a weight w_j parameter for each of the four heads. Heads 1 (blue) and 2 (green) follow a Weibull distribution. Instead, heads 3 (orange) and 4 (yellow) follow a LogLogistic distribution. A ReLU activation function ensures that distribution parameters are nonnegative, while weights (gray) are processed by a generic activation function $\phi(\cdot)$. Heads are combined to form a single hazard function and its corresponding cumulative hazard function. New trees are built by fitting the gradient of the negative log-likelihood and ElasticNet loss function (purple).

Weibull and LogLogistic heads allows FPBoost to reproduce complex bathtub or multimodal hazard curves observed in many practical survival datasets.

Figure 3.1 illustrates an example architecture with four heads ($J = 4$), where the first two heads use the Weibull distribution and the other two the LogLogistic distribution. In this setup, FPBoost requires learning $3J$ parameters: scale, shape, and weight for each head. These parameters are estimated via a set of gradient-boosted decision trees, grown iteratively in a standard boosting framework by fitting the loss-based residuals at each round. Details regarding the training procedure are given in Section 3.3. To ensure that the scale and shape parameters fit within the proper definition ranges, we apply a rectified linear unit (ReLU) activation function. This way, these parameters are nonnegative. Concerning the weights, instead, we can apply any activation function we want, provided that the resulting hazard is clipped to be nonnegative too.

3.2 Universal Approximation

A fundamental question in machine learning is whether a given model can learn arbitrarily complex target functions. In the context of survival analysis, this translates to asking whether a model can approximate any hazard function arbitrarily well, assuming access to sufficiently many samples and model capacity. Such analyses of *universal approximation* focus on theoretical expressiveness rather than practical convergence guarantees: although a model may, in principle, represent a broad class of hazard functions, the optimization process might fail to converge to the best parameters. Historically, similar universal approximation theorems trace back to classical results on neural networks, which show that they can approximate a wide variety of functions given suitable architectures.

In the study of FPBoost, we demonstrate that sums of Weibull hazards with appropriately chosen shape and scale parameters can approximate any risk profile. Specifically, these sums are dense in the space of continuous, nonnegative functions on any finite time interval, which corresponds to the set of all possible hazard functions. Formally:

Theorem 3.2.1. *Let $\mathcal{H}$ be the set of continuous, nonnegative real-valued functions $h\colon [0,T] \to \mathbb{R}_{\geq 0}$ for which the cumulative hazard diverges on $[0,\infty)$. For any $h^\star \in \mathcal{H}$ and $\varepsilon > 0$, there exists a finite sum of Weibull hazards*

$$h(t) \;=\; \sum_{j=1}^{J} w_j \, h_W\big(t \mid \eta_j, k_j\big)$$

such that

$$\sup_{t\in[0,T]} \Big| \, h^\star(t) \;-\; h(t) \Big| \;<\; \varepsilon.$$

This result follows from noting that a Weibull hazard with an integer-valued shape parameter can be expressed as a monomial $b\,t^n$. Summations of monomials form polynomials; and by the Weierstrass Approximation Theorem [55], polynomials are dense in the space of continuous functions on a compact interval. Consequently, suitable linear combinations of Weibull components approximate any continuous, nonnegative hazard function $h^\star$ on $[0,T]$ to arbitrary precision. Following this intuition, we provide a formal proof of Theorem 3.2.1.

Proof. Fix $h^\star \in \mathcal{H}$ and $\varepsilon > 0$, and define $\delta := \varepsilon/2$. Let $g(t)$ be a strictly positive lift $g(t) = h^\star(t) + \delta$ of the target hazard for $t \in [0,T]$. By the Weierstrass Approximation Theorem, there exists a polynomial $p(t) = \sum_{n=0}^{N} b_n t^n$ such that

$$\sup_{t\in[0,T]} |p(t) - g(t)| < \delta. \tag{1}$$

Since $g \geq \delta$ and (1) bounds $|p - g|$ by the same δ, we have $p(t) \geq 0$ for all $t \in [0, T]$, ensuring that the eventual hazard estimate never becomes negative. Moreover,

$$\sup_{t\in[0,T]} |p(t) - h^\star(t)| \leq \sup_t |p(t) - g(t)| + \delta < \delta + \delta = \varepsilon, \tag{2}$$

so p approximates $h^\star$ within tolerance ε.

Next, for each exponent n choose the Weibull-head parameters $k_n := n + 1$ (shape) and $\eta_n := 1$ (scale), and define the weight $w_n := b_n/k_n$. Observing that $w_n\, h_W(t \,|\, 1, k_n) = w_n\, k_n\, t^{k_n-1} = b_n t^n$, we rewrite the polynomial as

$$p(t) = \sum_{n=0}^{N} b_n t^n = \sum_{n=0}^{N} w_n\, h_W(t \,|\, 1, k_n).$$

Finally, set $h(t) := \sum_{n=0}^{N} w_n\, h_W(t \,|\, 1, k_n)$. Equation (2) guarantees that h uniformly approximates $h^\star$ within ε on $[0, T]$, while the earlier non-negativity argument ensures $h(t) \geq 0$ on the same interval. Because the representation uses only finitely many Weibull heads, h belongs to the model class $\mathcal{H}$, completing the proof. □

3.2.1 Bounding the Number of Heads

Although Theorem 3.2.1 ensures that FPBoost can model any continuous hazard arbitrarily well, it does not specify how many Weibull heads (J) are required to reach a given approximation accuracy. Fortunately, a classical result from function-approximation theory called Jackson inequality [67] provides a useful bound under mild smoothness assumptions. In particular, suppose the target hazard $h^\star(t)$ is Lipschitz continuous with parameter L on $[0, T]$. Then, the number of Weibull components needed to achieve an approximation error ε is of the order

$$O\big(\tfrac{LT}{\varepsilon}\big).$$

Consequently, for many well-behaved hazard functions, relatively small values of J may suffice in practice. We provide a formal statement for this result.

Theorem 3.2.2. *Let $h^\star\colon [0, T] \to \mathbb{R}_{\geq 0}$ be Lipschitz-continuous with constant L, i.e. $|h^\star(t) - h^\star(s)| \leq L|t - s|$ for all $s, t \in [0, T]$. For any $\varepsilon > 0$ choose*

$$N \;:=\; \left\lceil \frac{6\, L\, T}{\varepsilon} \right\rceil \quad \text{and set} \quad J := N + 1.$$

Then there exist shape parameters $k_j = j$ and weights $w_j \in \mathbb{R}$ such that the hazard

$$h(t) \;=\; \sum_{j=1}^{J} w_j\, h_W\Big(t \;\Big|\; \eta = 1,\; k = j\Big)$$

satisfies the uniform bound $\sup_{t\in[0,T]} \big|h(t) - h^\star(t)\big| \leq \varepsilon$.

Proof. Rescale the domain by $\tau = t/T$ and set $f(\tau) := h^\star(T\tau)$ on [0,1]. The rescaled function is LT-Lipschitz, so its modulus of continuity obeys $\omega(f, \delta) \le LT\delta$. The Jackson–Favard inequality [67] for algebraic polynomials p_N of degree at most N gives

$$\|f - p_N\|_\infty \le \frac{6\,\omega\big(f, 1/N\big)}{N} \;\le\; \frac{6\,L\,T}{N}\,.$$

Choosing $N \ge 6LT/\varepsilon$ ensures $\|f - p_N\|_\infty \le \varepsilon$. Writing $p_N(\tau) = \sum_{n=0}^{N} b_n \tau^n$ and reverting to $t = T\tau$ yields $p_N(t/T) = \sum_{n=0}^{N} b_n T^{-n} t^n$. Each monomial t^n equals $\frac{1}{n+1} h_W(t \mid \eta = 1, k = n+1)$, so setting $w_{n+1} = b_n T^{-n}(n+1)$ for $n = 0, \dots, N$ expresses $p_N(t/T)$ as a sum of $J = N + 1$ Weibull hazards. Finally, undoing the rescaling shows that the resulting mixture h approximates $h^\star$ within ε on $[0, T]$. □

3.2.2 The Role of LogLogistic Heads

While FPBoost's universal approximation result relies exclusively on Weibull components, incorporating LogLogistic heads offers practical advantages. First, from an interpretability standpoint, Weibull hazards often capture aging and wear processes (monotonically increasing risk), whereas LogLogistic hazards can model early failures (infant mortality). By combining these distributions, FPBoost produces a more interpretable decomposition of risk over time. Second, from an optimization perspective, the LogLogistic heads enable the model to capture non-monotonic patterns more efficiently. Rather than requiring a large number of Weibull components to represent complex hazard shapes, a relatively small set of LogLogistic heads can directly capture early-failure behaviors, reducing model complexity, training time, and the risk of overfitting.

3.2.3 Universality Across Survival Models

In this section, we showed that FPBoost is a gradient-boosting method for survival analysis whose hazard formulation allows to represent arbitrary hazard functions with an explicit constructive proof. Our argument parallels the classical universal-approximation theorem for neural networks: by expanding the hazard in a finite set of polynomials, an additive boosting ensemble can approximate any positive, integrable hazard function on a compact time domain to arbitrary precision.

That said, universal approximation results are not unique to FPBoost. Long-standing non-parametric statistics shows that kernel and nearest-neighbor versions of the conditional Kaplan–Meier or Nelson–Aalen estimators are uniformly consistent over very large families of survival or hazard functions [45, 29, 110, 20]. Additionally, in the machine-learning literature, discrete-time networks such as DeepHit [72] and Logistic Hazard [70] can also approximate any function with arbitrary precision, provided enough discretization points.

3.3 Training Algorithm

FPBoost learns its parameters by maximizing the full survival likelihood for right-censored, single-event data. The following describes the loss formulation and the gradient-boosting procedure used to train the model.

Survival Likelihood. Consider a dataset of n samples, where each sample i has:

- a feature vector $\mathbf{x}_i \in \mathbb{R}^d$,
- an observed time $t_i > 0$,
- an event indicator $\delta_i \in \{0,1\}$ (1 if the event is observed at t_i, 0 if the sample is censored).

Let $\Theta_i = \{\eta_j(\mathbf{x}_i), k_j(\mathbf{x}_i), w_j(\mathbf{x}_i)\}_{j=1}^{J}$ collect the parameters associated with sample i. FPBoost represents the hazard function for subject i as

$$h\Big(t_i \mid \Theta_i\Big) \;=\; \sum_{j=1}^{J} w_j(\mathbf{x}_i)\, h_j\Big(t_i \;\mid\; \eta_j(\mathbf{x}_i),\, k_j(\mathbf{x}_i)\Big).$$

The corresponding cumulative hazard is denoted $H\Big(t_i \mid \Theta_i\Big)$. At the iteration 0, assume that each distribution parameter and weight is initialized randomly. For right-censored data, the negative log-likelihood (NLL) is

$$\mathcal{L}_{\text{lik}}(\Theta) \;=\; -\frac{1}{N}\sum_{i=1}^{N}\Big[\delta_i\, \ln h\Big(t_i \mid \Theta_i\Big) \;-\; H\Big(t_i \mid \Theta_i\Big)\Big],$$

where Θ denotes the full set of parameters for all samples.

Gradient Boosting Framework. To optimize Θ, FPBoost uses a gradient-boosting framework. For each type of parameter, we introduce an additive model

$$F_\varphi(\mathbf{x}) \;=\; \sum_{m=1}^{M} \lambda\, \tau_\varphi^{(m)}(\mathbf{x}),$$

where $\varphi \in \{\eta_j,\, k_j,\, w_j\}$ and $\tau_\varphi^{(m)}(\cdot)$ is a regression tree fitted at iteration m. $\lambda > 0$ is the learning rate, weighting the contribution of each subsequent tree. We map $F_\varphi(\mathbf{x})$ to the desired parameter space (e.g., positive real numbers) via an activation function:

$$\varphi(\mathbf{x}) \;=\; \phi\Big(F_\varphi(\mathbf{x})\Big).$$

We set ϕ as a ReLU function for η_j and k_j to ensure nonnegativity. Instead, any other function (e.g., ReLU, softmax, identity, hyperbolic tangect, etc.) can be applied to w_j. Allowing for negative weights, e.g., identity or hyperbolic tangent activations, enables FPBoost universal approximation. However, to ensure its validity, the hazard function must be clipped above 0.

ElasticNet Regularization. To mitigate overfitting, FPBoost augments the negative log-likelihood with an ElasticNet penalty:

$$\mathcal{L}_{\text{reg}}(\Theta) \;=\; \alpha \left(\gamma \, \|\Theta\|_1 \;+\; (1-\gamma) \, \|\Theta\|_2^2 \right),$$

where $\alpha \geq 0$ is the regularization weight and $\gamma \in [0,1]$ controls the relative contributions of $L1$ and $L2$ norms. The total loss to be minimized becomes

$$\mathcal{L}(\Theta) \;=\; \mathcal{L}_{\text{lik}}(\Theta) \;+\; \mathcal{L}_{\text{reg}}(\Theta).$$

Iterative Optimization Algorithm 1 outlines the main steps in the FPBoost training loop. At each boosting iteration, first, we estimate all the parameters Θ required to compute the hazard function. From these, we evaluate the loss $\mathcal{L}(\Theta)$. Then, the negative gradient of the total loss, called pseudo-residual, is computed for each to each parameter $F_\varphi(\mathbf{x})$. At this point, we fit a regression tree to each of these residuals, and update $F_\varphi(\mathbf{x})$ accordingly by adding the newly trained tree to the existing list of trees. By including the new tree, we encourage the model to refine its parameter predictions in order to lower the total loss, as iterations progress. Training continues for M rounds or until an early-stopping criterion based on validation concordance is met.

3.4 Experiments

In this section, we discuss the empirical validation of FPBoost from our previous work [11].

3.4.1 Data and Environment

We evaluate FPBoost on several publicly available survival datasets, detailed in Section A.1. Specifically, FPBoost and all baseline models are tested on the AIDS, Breast Cancer, FLCHAIN, GBSG2, METABRIC, SUPPORT2, Veterans, and WHAS datasets. Each dataset is partitioned into a training set (80% of the samples) and a test set (20%). The test split remains fixed across all experiments, ensuring a fair comparison among models. For datasets coming from the DeepSurv repository[1] we preserve the predefined test splits. For the remaining datasets, we apply a stratified split on the event variable, using a fixed random seed for reproducibility. Each experiment is run 30 times. Reported metrics include the mean and 95% confidence intervals over these runs.

[1] https://github.com/jaredleekatzman/DeepSurv

Algorithm 1 FPBoost Training Algorithm

Require: Training data $\{(\mathbf{x}_i, \delta_i, t_i)\}_{i=1}^{N}$; number of heads J;
parametric forms h_j (e.g., Weibull, LogLogistic); learning rate λ;
maximum boosting rounds M; regularization parameters α, γ.

Ensure: A trained FPBoost model $\{\eta_j(\cdot),\, k_j(\cdot),\, w_j(\cdot)\}_{j=1}^{J}$.

Initialize $F_{\eta_j}, F_{k_j}, F_{w_j}$ for $j = 1, \ldots, J$ to random values.
for $m = 1$ to M **do**
  for $j = 1$ to J **do**
    Compute current parameters for all samples i:

$$\begin{aligned} \eta_j(\mathbf{x}_i) &\leftarrow \mathrm{ReLU}\Big(F_{\eta_j}(\mathbf{x}_i)\Big), \\ k_j(\mathbf{x}_i) &\leftarrow \mathrm{ReLU}\Big(F_{k_j}(\mathbf{x}_i)\Big), \\ w_j(\mathbf{x}_i) &\leftarrow \phi\Big(F_{w_j}(\mathbf{x}_i)\Big). \end{aligned}$$

  end for
  Compute total loss:

$$\begin{aligned} \mathcal{L}_{\text{lik}}(\Theta) &= -\frac{1}{N}\sum_{i=1}^{N}\Big[\delta_i \,\ln h\Big(t_i \mid \Theta_i\Big) \;-\; H\Big(t_i \mid \Theta_i\Big)\Big], \\ \mathcal{L}_{\text{reg}}(\Theta) &= \alpha\left(\gamma\,\|\Theta\|_1 \;+\; (1-\gamma)\,\|\Theta\|_2^2\right), \\ \mathcal{L}(\Theta) &= \mathcal{L}_{\text{lik}}(\Theta) \;+\; \mathcal{L}_{\text{reg}}(\Theta). \end{aligned}$$

  for each parameter estimator F_φ, $\varphi \in \{\eta_j, k_j, w_j\}_{j=1}^{J}$ **do**
    Compute the pseudo-residual

$$r_\varphi(\mathbf{x}_i) \;=\; -\frac{\partial \mathcal{L}(\Theta)}{\partial F_\varphi(\mathbf{x}_i)}.$$

    Fit a regression tree $\tau_\varphi^{(m)}$ to $r_\varphi(\mathbf{x}_i)$.
    Update $F_\varphi(\mathbf{x})$ as

$$F_\varphi(\mathbf{x}) \;\leftarrow\; F_\varphi(\mathbf{x}) \;+\; \lambda\,\tau_\varphi^{(m)}(\mathbf{x}).$$

  end for
  (*Optional*) Check early-stopping based on a validation set.
end for
return The final set of parameter estimators $\{F_{\eta_j}, F_{k_j}, F_{w_j}\}$.

Within the training set from each experiment instance, an internal validation set is obtained by performing an additional 80–20 split. Model hyperparameters are

tuned by maximizing the validation C-Index on the validation data.

Data preprocessing follows a minimal approach. In particular, numerical features are standardized to zero mean and unit variance. For categorical features, instead, we perform one-hot encoding. Missing values are imputed with the mean for numerical columns and the mode for categorical columns. By design, this procedure yields a uniform, NaN-free representation across all datasets that can be processes by both tree-based and neural network-based models.

3.4.2 Baselines

To benchmark FPBoost's performance, we compare it against several existing survival analysis methods:

- Random Survival Forests (RSF) from `scikit-survival`, using default parameters.
- CoxPH and DeepSurv from the `pycox` library. DeepSurv employs a three-layer neural network with a number of neurons proportional to (3,5,3) $\times$ (# of features), following the guidelines in the original paper.
- CoxBoost from `scikit-survival`, adopting the default parameter settings.
- XGBoost (using the `cox` loss) from the `xgboost` library.
- DeepHit from `pycox` and Deep Survival Machines from `auton-survival` as representatives of fully parametric approaches. For both these models, the backbone neural architectures use the same number of layers and neurons as DeepSurv.

3.4.3 Hyperparameter Tuning for FPBoost

The tuning process for FPBoost involves searching over multiple architectural and training parameters, selecting the combination that maximizes the validation C-Index. The hyperparameters explored are

- Number of Weibull heads ($\{0,\dots,32\}$).
- Number of LogLogistic heads ($\{0,\dots,32\}$).
- Maximum number of trees per parameter ($\{1,\dots,512\}$).
- Maximum tree depth ($\{1,\dots,6\}$).
- Activation for head weights (one of ReLU, sigmoid, softmax, tanh, and identity).

Table 3.1: Best hyperparameter configuration for the FPBoost model across all datasets.

Parameter	**AIDS**	**Breast Cancer**	**FLCHAIN**	**GBSG2**
Estimators	16	32	32	16
Weibull	32	1	64	4
LogLogistic	4	0	1	8
Max Depth	1	1	3	1
λ	1.0	1.0	1.0	1.0
α	0.5	0.01	0.1	0.01
γ	0.0	0.25	0.0	0.0
$\phi(\cdot)$	ReLU	ReLU	ReLU	ReLU
Initialization	Random	Random	Random	Random
Patience	–	–	–	–
Parameter	**METABRIC**	**SUPPORT2**	**Veterans**	**WHAS**
Estimators	32	16	64	128
Weibull	1	4	1	16
LogLogistic	1	8	4	1
Max Depth	1	3	1	6
λ	1.0	1.0	1.0	1.0
α	0.0	0.1	0.0	0.0
γ	–	0.0	–	–
$\phi(\cdot)$	ReLU	ReLU	ReLU	ReLU
Initialization	Random	Random	Random	Kaplan–Meier
Patience	–	–	–	16

- Learning rate $\lambda \in [0.01,1]$.
- ElasticNet hyperparameters: $\alpha \in [0,1]$ and $\gamma \in [0,1]$.
- Patience: the number of consecutive iterations allowed without an increase in the validation C-Index before early stopping.

During the tuning process, we also tried two different initialization strategies for parameter estimators. The first is random, following these distributions: $\eta \sim \mathcal{N}(0.5,1)$, $k \sim \mathcal{N}(0,2)$, and $w \sim \mathcal{N}(0,1)$.

The second is based on Kaplan–Meier. Specifically, the procedure fits a Weibull and LogLogistic distribution to the KM estimator to obtain $\bar{\eta}, \bar{k}$ and initializes parameters as

$$\eta \sim \mathcal{N}\big(\bar{\eta},\, \bar{\eta}/10\big), \quad k \sim \mathcal{N}\big(\bar{k},\, \bar{k}/10\big),$$

while weights are initialized with the same value $1/J$.

The selected hyperparameters for each dataset are reported in Table 3.1. Our open-source implementation of FPBoost is available on GitHub[2], along with scripts to replicate all experiments. FPBoost follows the `scikit-survival` interface for seamless integration into existing pipelines, allowing standard functionalities such as cross-validation and performance metrics from the `scikit` ecosystem.

3.5 Results

This section presents the empirical performance of FPBoost in comparison with classical and modern survival models: RSF, CoxPH, CoxBoost, DeepSurv, Deep Survival Machines (DSM), and DeepHit. We report results for the three standard metrics in survival analysis: C-Index, Integrated Brier Score (IBS), and Cumulative AUC. Results in the tables in this section are scaled up by a factor of 100 for readability.

3.5.1 C-Index

Table 3.2 gives a full picture of the concordance indices of each model on the selected benchmark datasets. Alongside these results, Table 3.3 provides an aggregated view of the same results, averaged across baseline types. Specifically, the latter table collects the average performance difference between FPBoost and the considered baselines.

FPBoost achieves top or near-top performance across nearly all experiments. Notably, it is first-ranked on most datasets, with only two exceptions (AIDS and Veterans) where RSF and DeepSurv surpass it. On average, FPBoost improves the C-Index by 4.6 points over any other baseline. Since a random guess yields a C-Index of 50, this gain corresponds to approximately 9% relative improvement.

Comparing FPBoost to semi-parametric approaches (CoxPH, CoxBoost, DeepSurv) reveals an average improvement of 4.1, suggesting that strict proportional-hazard assumptions may limit model expressiveness for more complex datasets. Furthermore, when comparing FPBoost to neural network models (DeepSurv, DSM, DeepHit), FPBoost exhibits a larger margin (5.5 points on average), implying that

[2] https://github.com/archettialberto/fpboost

Table 3.2: Test C-Index (↑) and 95% confidence interval for each model and dataset, averaged over 30 splits (same test, different train and validation sets). To enhance readability, all values are scaled by a factor of 100. Best results are highlighted in **bold**, while the second best are underlined.

Model	AIDS	Breast Cancer	FLCHAIN	GBSG2
Cox	77.8 ± 1.9	63.4 ± 1.9	$\underline{93.7 \pm 0.0}$	69.3 ± 0.3
CoxBoost	76.1 ± 0.9	60.0 ± 2.6	93.7 ± 0.0	68.9 ± 0.6
XGBoost	53.2 ± 1.5	57.1 ± 2.7	88.9 ± 0.1	63.4 ± 0.9
RSF	$\mathbf{80.1 \pm 0.8}$	58.5 ± 1.6	93.7 ± 0.0	68.6 ± 0.4
DeepSurv	70.7 ± 3.0	64.9 ± 1.7	93.6 ± 0.0	$\underline{69.5 \pm 0.4}$
DeepHit	$\underline{78.4 \pm 0.9}$	64.6 ± 2.4	93.5 ± 0.0	65.5 ± 1.0
DSM	76.8 ± 1.0	$\underline{66.2 \pm 0.8}$	50.0 ± 0.0	49.9 ± 0.3
FPBoost	78.1 ± 0.7	$\mathbf{66.6 \pm 3.2}$	$\mathbf{93.8 \pm 0.0}$	$\mathbf{69.7 \pm 0.4}$
Model	**METABRIC**	**SUPPORT2**	**Veterans**	**WHAS**
Cox	63.2 ± 0.1	82.7 ± 0.0	75.6 ± 1.0	81.7 ± 0.1
CoxBoost	63.2 ± 0.2	83.4 ± 0.0	72.1 ± 1.9	85.1 ± 0.1
XGBoost	61.4 ± 0.5	56.4 ± 0.7	70.9 ± 1.5	83.4 ± 0.3
RSF	61.6 ± 0.2	$\underline{84.2 \pm 0.1}$	$\underline{75.8 \pm 1.0}$	$\underline{85.8 \pm 0.1}$
DeepSurv	$\underline{63.4 \pm 0.2}$	82.6 ± 0.1	$\mathbf{76.7 \pm 1.0}$	83.7 ± 0.1
DeepHit	61.7 ± 0.3	82.2 ± 0.1	72.0 ± 1.4	82.6 ± 0.2
DSM	61.3 ± 0.1	83.6 ± 0.3	65.4 ± 0.2	69.7 ± 0.6
FPBoost	$\mathbf{64.0 \pm 0.1}$	$\mathbf{84.3 \pm 0.4}$	74.8 ± 1.3	$\mathbf{89.0 \pm 0.3}$

Table 3.3: FPBoost's mean improvement on C-Index (↑) across all datasets.

C-Index	Linear	Tree	Neural	All
Non-Parametric	–	+1.5	–	+1.5
Semi-Parametric	+1.6	+6.4	+1.9	+4.1
Fully Parametric	–	–	+7.3	+7.3
All	+1.6	+4.8	+5.5	+4.6

neural architectures might be more prone to overfitting without careful regularization or architectural tuning. In contrast, FPBoost's tree-based nature seems to provide a more robust inductive bias. Notably, RSF's performance is closer to that of FPBoost than neural baselines, reflecting the potential advantage of tree ensembles for survival tasks.

3.5.2 Integrated Brier Score

Following the same results collection as in Section 3.5.1, Table 3.4 reports the detailed calibration performance through the IBS metric, while Table 3.5 summarizes the same findings averaging over datasets and baselines. The overall trends mirror those observed for the C-Index: FPBoost consistently ranks first or second on every dataset except AIDS. Averaged over all datasets (excluding XGBoost due to its outlier behavior), FPBoost achieves a 2.8-point reduction in IBS relative to the baseline approaches. Since random guessing has an IBS of 25, this improvement is about 11% in relative terms.

When focusing on model families, FPBoost displays a modest advantage of 0.7 points over RSF, while its advantage over semi-parametric models is 1.7 points. The gap grows to 4.5 points compared to neural network-based methods, aligning with the observations derived from the C-Index. Hence, the strength of FPBoost in ranking consistency (C-Index) are also reflected in its ability to produce well-calibrated survival probabilities (IBS).

3.5.3 Cumulative AUC

Trends in Cumulative AUC largely mirror those observed in C-Index and IBS. The detailed results are presented in Table 3.6, with a summary in Table 3.7. Once again, FPBoost ranks first across multiple datasets, though the advantage is less pronounced compared to the previous metrics. Since Cumulative AUC is a ranking-based measure, we expect trends similar to C-Index. In fact, FPBoost underperforms relative to other models in the AIDS and Veterans datasets, as well as in Breast Cancer, while securing second place on SUPPORT2. The impact of model type on performance remains significant, with the largest differences emerging when FPBoost is compared to neural network-based models. In contrast, its performance relative to tree-based models shows a much smaller gap, reinforcing the idea that without extensive fine-tuning, tree-based approaches tend to be more stable and easier to train on small-to-medium scale tabular datasets.

Table 3.4: Test IBS (↓) and 95% confidence interval for each model and dataset, averaged over 30 splits (same test, different train and validation sets). To enhance readability, all values are scaled by a factor of 100. Best results are highlighted in **bold**, while the second best are underlined.

Model	AIDS	Breast Cancer	FLCHAIN	GBSG2
Cox	**5.8 ± 0.0**	22.2 ± 1.2	**4.6 ± 0.0**	17.7 ± 0.1
CoxBoost	6.2 ± 0.1	19.8 ± 0.9	4.7 ± 0.0	17.3 ± 0.2
XGBoost	8.5 ± 0.2	22.8 ± 0.9	12.1 ± 0.1	26.9 ± 0.7
RSF	5.8 ± 0.0	18.0 ± 0.3	4.7 ± 0.0	17.7 ± 0.2
DeepSurv	6.2 ± 0.1	26.4 ± 1.1	4.7 ± 0.0	17.5 ± 0.1
DeepHit	5.8 ± 0.0	23.9 ± 1.3	6.3 ± 0.1	21.4 ± 0.1
DSM	6.2 ± 0.0	17.8 ± 0.0	13.9 ± 0.0	21.6 ± 0.0
FPBoost	6.0 ± 0.0	**17.1 ± 0.9**	4.7 ± 0.0	**17.1 ± 0.2**
Model	**METABRIC**	**SUPPORT2**	**Veterans**	**WHAS**
Cox	19.9 ± 0.0	13.2 ± 0.0	13.4 ± 0.2	14.0 ± 0.0
CoxBoost	20.9 ± 0.1	12.6 ± 0.0	14.7 ± 0.7	11.9 ± 0.1
XGBoost	24.3 ± 0.4	60.6 ± 0.6	47.4 ± 1.7	18.7 ± 0.4
RSF	21.0 ± 0.1	**12.3 ± 0.0**	**12.7 ± 0.2**	8.5 ± 0.1
DeepSurv	20.4 ± 0.1	14.6 ± 0.1	14.4 ± 0.2	12.2 ± 0.1
DeepHit	22.7 ± 0.1	14.7 ± 0.1	29.4 ± 0.9	17.2 ± 0.1
DSM	23.7 ± 0.0	19.4 ± 0.3	23.0 ± 0.1	20.5 ± 0.0
FPBoost	**19.8 ± 0.0**	12.5 ± 0.1	12.8 ± 0.3	**8.4 ± 0.3**

Table 3.5: FPBoost's IBS mean improvement (↓) across all datasets.

IBS	Linear	Tree	Neural	All
Non-Parametric	–	−0.3	–	−0.3
Semi-Parametric	−1.5	−1.2	−2.3	−1.7
Fully Parametric	–	–	−5.6	−5.6
All	−1.5	−0.7	−4.5	−2.8

Table 3.6: Test AUC (↑) and 95% confidence interval for each model and dataset, averaged over 30 splits (same test, different train and validation sets). To enhance readability, all values are scaled by a factor of 100. Best results are highlighted in **bold**, while the second best are underlined.

Model	AIDS	Breast Cancer	FLCHAIN	GBSG2
Cox	**78.9 ± 2.0**	63.0 ± 2.1	95.4 ± 0.0	77.8 ± 0.3
CoxBoost	76.4 ± 1.1	60.3 ± 2.6	95.5 ± 0.0	76.8 ± 0.7
XGBoost	55.8 ± 2.1	58.1 ± 3.0	91.4 ± 0.1	67.2 ± 1.3
RSF	72.8 ± 1.6	61.2 ± 2.2	95.7 ± 0.0	76.6 ± 0.5
DeepSurv	70.9 ± 3.2	**63.8 ± 2.2**	95.5 ± 0.0	77.7 ± 0.4
DeepHit	78.4 ± 1.1	58.9 ± 2.6	95.5 ± 0.1	65.9 ± 1.8
DSM	75.4 ± 1.3	62.8 ± 0.8	50.0 ± 0.0	48.6 ± 0.5
FPBoost	77.1 ± 0.9	62.9 ± 3.8	**95.8 ± 0.0**	**77.9 ± 0.4**
Model	**METABRIC**	**SUPPORT2**	**Veterans**	**WHAS**
Cox	69.0 ± 0.1	91.0 ± 0.0	86.3 ± 1.1	84.8 ± 0.1
CoxBoost	65.3 ± 0.6	91.8 ± 0.0	81.8 ± 2.0	88.2 ± 0.1
XGBoost	64.9 ± 0.8	56.6 ± 0.8	80.8 ± 1.7	87.3 ± 0.4
RSF	67.9 ± 0.3	**91.9 ± 0.0**	82.9 ± 0.8	92.1 ± 0.1
DeepSurv	68.3 ± 0.3	90.6 ± 0.1	**87.6 ± 1.0**	86.7 ± 0.1
DeepHit	67.4 ± 0.4	38.0 ± 0.3	48.0 ± 4.6	72.1 ± 0.4
DSM	66.8 ± 0.1	90.8 ± 0.1	77.4 ± 0.1	71.5 ± 0.4
FPBoost	**71.4 ± 0.3**	91.8 ± 0.0	86.1 ± 1.0	**92.8 ± 0.3**

Table 3.7: FPBoost's Cumulative AUC mean improvement (↑) across all datasets.

Cumulative AUC	Linear	Tree	Neural	All
Non-Parametric	–	+1.8	–	+1.8
Semi-Parametric	+1.2	+7.1	+1.8	+4.3
Fully Parametric	–	–	+15.3	+15.3
All	+1.2	+5.3	+10.8	+7.1

3.6 Discussion

FPBoost emerges as a promising addition to the landscape of survival models, offering a tree-based architecture that directly optimizes the survival likelihood through a weighted sum of parametric hazard functions. Its empirical results demonstrate competitive performance, revealing gains in both concordance and calibration when compared to classical and state-of-the-art alternatives, including neural-network-based approaches. These improvements appear particularly pronounced against models that rely on partial or discrete-time assumptions, thereby highlighting the advantages of avoiding the inclusion of restrictive assumptions in modeling survival events from complex data.

Starting from this work, there remain several interesting research directions for further exploration. We could refine the theoretical grounding of FPBoost by exploring better approximation bounds and convergence properties. Additionally, the ensemble-like nature of FPBoost promotes its extension to competing risks frameworks, in which multiple heads could address different events of interest. Equally compelling is the application of FPBoost on larger and more complex datasets, including multimodal medical datasets and federated learning scenarios also warrants attention.

Ultimately, the ability to provide accurate and reliable survival estimates holds considerable promise for medical applications, where personalized risk assessments can guide more informed decisions about patient care and resource allocation.

Chapter 4

Interpolation Techniques for Neural Models

Alongside FPBoost, which is built on decision trees, we pursued a complementary line of research aimed at enhancing discrete-time neural survival models, such as Logistic Hazard and DeepHit. These methods cast survival prediction as a multi-output classification problem by estimating a separate probability to each point on a user-defined time grid. Although, in principle, the grid may be as fine as the set of unique observed event times—mirroring the discretization used by the Kaplan–Meier estimator or the Breslow baseline in the Cox model—our experiments show that pushing to such high resolution enlarges the parameter space and slows optimization, ultimately harming both discrimination and calibration. By contrast, models trained on a coarser grid are far easier to optimize and still yield accurate survival estimates. This empirical trade-off motivates the central question examined in the remainder of this chapter: to what extent can a coarse-grid model recover fine-grained survival estimations by simply applying an interpolation algorithm between its outputs?

Investigating this question, we observed that interpolation, which converts discrete estimates into a continuous survival function, is rarely exploited in the literature on neural survival models. A previous study [70] compares continuous and linearly-interpolated discrete survival models—lacking however an extensive comparison on additional interpolation techniques. Another work [107] systematically explores the impact of interpolation on the discrete outputs of several survival models but limits the investigation to linear ones. We therefore extended these comparisons with different interpolation techniques and neural-based models, measuring how interpolation can enhance performance at a minimal computational cost.

Interpolation can take various forms—from simple linear functions to more complex monotonic splines. Although each method has its own properties and computational demands, all tested techniques have only a negligible overhead compared to the training of a survival model. In essence, interpolation introduces a

principled inductive bias at a small cost, by assuming that the probability between two discrete estimates varies smoothly. This trend should encourage models to provide predictions closer to real-world scenarios, where survival probabilities on average do not follow step-like abrupt changes.

4.1 Interpolation Methods

Neural network–based survival models typically yield *discrete* survival probability estimates defined at a limited set of time points. This discretization produces a coarse, stepwise survival function that can be difficult to interpret and may lead to inaccuracies when evaluating time-dependent metrics. In practice, the number of output time bins is usually much smaller than the number of unique event times in the data, resulting in a loss of resolution in the predicted survival curve.

To define our interpolation framework, let $\{\tau_1, \tau_2, \ldots, \tau_B\}$ denote the increasing sequence of time points that define the boundaries (bins) of the discrete intervals used by the model (referred to as *anchor times*). The neural model provides survival probabilities $\{s_1, s_2, \ldots, s_B\}$ corresponding to each τ_i, with the convention

$$1 = s_0 \geq s_1 \geq s_2 \geq \cdots \geq s_B \geq s_{B+1} = 0,$$

where $s_0 = 1$ at time $\tau_0 = 0$ (indicating 100% survival at time 0) and $s_{B+1} = 0$ at a sufficiently large time τ_{B+1} (reflecting eventual failure). These anchor points, including (0,1) and (τ_{B+1},0), are connected by an interpolating function to yield a continuous survival curve $S(t)$.

In the following, we describe the interpolation methods included in this study, visually summarized in Figure 4.1.

4.1.1 Step-Forward Interpolation

The simplest and most widely used approach to provide a time-continuous probability function $S(t)$ from a set of discrete time points is *step-wise interpolation*, which produces a piecewise-constant survival function by extending each anchor value forward until the next anchor time. Strictly speaking, this is not a proper interpolation method, as the resulting function is discontinuous. We include it as a baseline ("no interpolation" method) to compare against the interpolation techniques that follow.

In the *Step-Forward (Step-FWD)* variant, for any time t in the interval $[\tau_i,\ \tau_{i+1})$, the survival function is defined as the value of the survival at the beginning of that interval:

$$S(t) = s_i, \quad \text{for } t \in [\tau_i,\ \tau_{i+1}) \ .$$

In other words, the survival probability remains at the level s_i immediately after time τ_i and is held constant until the next discrete time point is reached. This yields

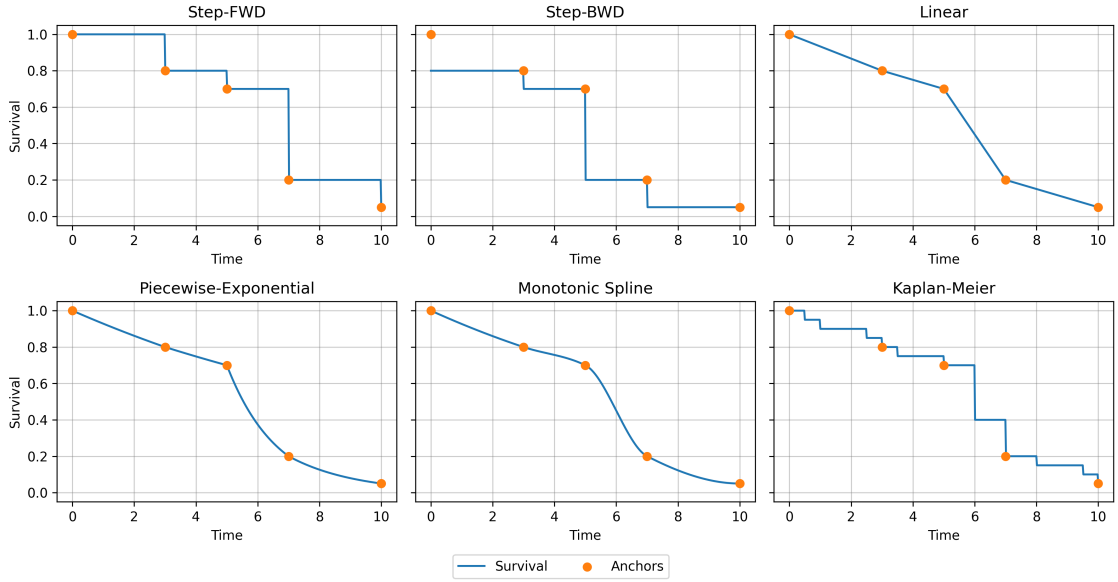

Figure 4.1: Visualization of the interpolation methods studied in this chapter. Each method shares the same five anchors points with times ranging from 0 to 10.

a right-continuous step function that drops instantaneously at each anchor time τ_{i+1} to the next level s_{i+1}. This approach assumes that no change in survival occurs between known time points (i.e., it propagates the last known survival probability forward in time). Step-Forward is the default in many neural survival studies, but it can lead to an overly optimistic survival probability just before an observed drop.

4.1.2 Step-Backward Interpolation

An alternative stepwise strategy is the *Step-Backward (Step-BWD)* interpolation. Intuitively, for $t \in [\tau_i, \tau_{i+1})$ the survival function is taken to be the next anchor value:

$$S(t) = s_{i+1}, \quad \text{for } t \in [\tau_i,\ \tau_{i+1}).$$

Here, the survival probability steps down immediately after τ_i to the level of the following interval. Equivalently, one can view Step-BWD as carrying the survival estimate backward from the next known time point. Clinically, this could correspond to a more cautious scenario anticipating future events soon after τ_i. Step-Backward interpolation tends to yield a more conservative survival estimate in each interval compared to Step-Forward.

4.1.3 Linear Interpolation

A straightforward way to introduce continuity is *linear interpolation* between anchor points. Instead of a jump, the survival probability is assumed to change uniformly from one anchor value to the next. For t between τ_i and τ_{i+1}, linear interpolation defines $S(t)$ as:

$$S(t) = s_i \; + \; \frac{t - \tau_i}{\tau_{i+1} - \tau_i} \Big(s_{i+1} - s_i \Big) \; , \quad \text{for } t \in [\tau_i, \; \tau_{i+1}),$$

which is the line connecting the points (τ_i, s_i) and (τ_{i+1}, s_{i+1}). By construction, $S(t)$ changes at a constant rate within the interval and $S(t)$ is continuous at τ_i and τ_{i+1}. Differently from stepwise methods, this yields a continuous survival curve, though its slope will change abruptly at each anchor point, meaning that the derivative is not continuous. Linear interpolation is often a reasonable default when one expects the survival probability to decrease roughly uniformly between known time points, and it tends to improve integrated metrics by filling in a more realistic gradual decrease rather than assuming survival stays flat and then drops suddenly.

4.1.4 Piecewise-Exponential Interpolation

The *Piecewise-Exponential (PWE) interpolation* is inspired by the piecewise-constant hazard model known as *PC-Hazard*, which extends piecewise-constant hazard models to the continuous survival domain.

Instead of assuming a linear change in survival, this method assumes the *hazard rate* is constant within each interval. Recall that the hazard $h(t)$ is related to the survival function by $h(t) = -\frac{d}{dt} \ln S(t)$. If $h(t)$ is constant on $[\tau_i, \; \tau_{i+1})$, then the survival function in that interval will follow an exponential decay. Thus, the PWE interpolation defines $S(t)$ such as

$$S(t) = s_i \, \exp\left(\, - \lambda_i \, \frac{t - \tau_i}{\tau_{i+1} - \tau_i} \right) \; , \quad \text{for } t \in [\tau_i, \; \tau_{i+1}),$$

where the rate λ_i is chosen to match the next anchor point, i.e. it is set by the condition $S(\tau_{i+1}) = s_{i+1}$. Solving $s_{i+1} = s_i \exp(\lambda_i)$ over the interval length, we get:

$$\lambda_i = \ln \frac{s_{i+1}}{s_i},$$

which is non-positive since $s_{i+1} \leq s_i$. This method yields a continuous and differentiable survival function on each interval (though the derivative still changes abruptly at the anchor points from interval to interval). The piecewise-exponential interpolation has a probabilistic interpretation: it is equivalent to assuming a constant conditional event rate between τ_i and τ_{i+1}. It originates from the PC-Hazard model, which treats survival analysis as a Poisson regression problem by breaking

time into intervals with constant hazard. By using PWE interpolation, we leverage this idea as a smoothing technique, by assuming that the underlying process within each time interval can be modeled with an exponential distribution.

4.1.5 Monotonic Cubic Spline Interpolation

While linear and exponential interpolations ensure continuity of $S(t)$, their first derivatives are not continuous at anchor points. *Monotonic cubic spline interpolation* addresses this by fitting a smooth cubic polynomial between each pair of anchors such that both $S(t)$ and its first derivative $S'(t)$ are continuous everywhere, and the curve is constrained to be non-increasing. We employ a Hermite cubic spline with monotonicity constraints, specifically using the Fritsch–Carlson method to determine the spline parameters.

For each adjacent anchor pair (τ_i, s_i) and (τ_{i+1}, s_{i+1}), we first compute the secant slope:

$$\delta_i = \frac{s_{i+1} - s_i}{\tau_{i+1} - \tau_i} \ ,$$

which is non-positive since $s_{i+1} \leq s_i$. Then, an initial guess for the derivative at each anchor (the spline tangent m_i) is set as the average of adjacent secants:

$$m_i = \frac{1}{2} \left(\delta_{i-1} + \delta_i \right) \ ,$$

with end conditions $m_1 = \delta_1$ and $m_B = \delta_B$ for the first and last internal anchor. If any $s_i = s_{i+1}$ (a flat segment in the survival function), we set $m_i = 0$. To preserve monotonicity, the Fritsch–Carlson method limits the m_i whenever they would cause overshoot: if the ratios $\alpha_i = m_i/\delta_i$ or $\beta_i = m_{i+1}/\delta_i$ exceed 3, then m_i and/or m_{i+1} are reduced such that $m_i = 3\,\delta_i$. This effectively means if a spline segment would swing upward making the survival function non-monotonic, it is adjusted to maintain this property. After determining suitable m_i at every anchor, the survival function on $[\tau_i, \tau_{i+1})$ is given by the Hermite cubic polynomial:

$$\begin{aligned} S(t) = &\left(2\mu^3 - 3\mu^2 + 1\right) s_i \ + \ \left(\mu^3 - 2\mu^2 + \mu\right) m_i \\ &+ \ \left(-2\mu^3 + 3\mu^2\right) s_{i+1} \ + \ \left(\mu^3 - \mu^2\right) m_{i+1} \ , \end{aligned}$$

where $\mu = \frac{t - \tau_i}{\tau_{i+1} - \tau_i}$ is the normalized position in the interval. By construction, $S(\tau_i) = s_i$ and $S(\tau_{i+1}) = s_{i+1}$. The tangents at the boundaries are $S'(\tau_i) = m_i$ and $S'(\tau_{i+1}) = m_{i+1}$, ensuring $S'(t)$ is continuous. This spline interpolation produces a smooth survival function that eliminates the abrupt changes in slope seen in linear or PWE interpolation. This method can capture changes in the survival function more flexibly than linear or exponential interpolation, offering visually realistic survival curves that could be closer to the true continuous survival function, especially when the underlying hazard rate is not constant between time anchors.

4.1.6 Kaplan–Meier Interpolation

All the interpolation methods discussed so far rely only on the model's output anchor points and assume a certain shape between them (flat, linear, exponential, or cubic polynomial). *Kaplan–Meier interpolation* instead incorporates empirical censoring information from the dataset to guide the interpolation. The idea is to leverage the KM estimator as a prior for the shape between anchors. The KM estimator, herein referred to as $S_{\text{KM}}(t)$ and detailed in Section 2.5.1, represents the observed survival probability up to time t accounting for censoring. To interpolate between (τ_i, s_i) and (τ_{i+1}, s_{i+1}), we squish the segment of the KM curve that falls in that interval to fit between the two anchor points.

Specifically, suppose we have two consecutive anchors τ_i, τ_{i+1}. We look at the KM estimate over $[\tau_i, \tau_{i+1})$. Let $S_{\text{KM}}(\tau_i)$ and $S_{\text{KM}}(\tau_{i+1})$ be the KM survival values at the endpoints of the interval. We then define $S(t)$ for $t \in [\tau_i, \tau_{i+1})$ as

$$S(t) = s_i \; - \; \Delta_i \Big(S_{\text{KM}}(t) \; - \; S_{\text{KM}}(\tau_{i+1}) \Big),$$

where Δ_i is a scaling factor chosen such that $S(t)$ will equal s_{i+1} at $t = \tau_{i+1}$ defined as

$$\Delta_i = \frac{s_{i+1} - s_i}{S_{\text{KM}}(\tau_{i+1}) - S_{\text{KM}}(\tau_i)}.$$

Conversely, if $S_{\text{KM}}(\tau_{i+1}) = S_{\text{KM}}(\tau_i)$, meaning that the KM estimate is constant within the interval, we set the interpolated survival function to be the average of the two anchor points as

$$S(t) = \frac{1}{2}\left(s_i + s_{i+1}\right).$$

Differently from other methods, the Kaplan–Meier interpolation introduces a data-driven shape between the anchors: essentially, it preserves the pattern of survival observed in the population, but scales it to fit the model's predicted survival rates s_i and s_{i+1}. The potential benefit is improved accuracy when the model's discrete predictions are uncertain as the KM provides guidance on how survival typically declines in that interval in contexts with a high censoring percentage or few data samples.

4.2 Experiments

To compare and validate the effectiveness of the proposed interpolation techniques, we conducted a comprehensive set of experiments on multiple datasets, using various neural survival models and baseline methods for comparison. This section describes the data and experimental setup.

4.2.1 Scaling the Number of Anchors

The first set of experiments from [10] is about model performance resulted after scaling the number of anchors. The most interesting find is analyzing the trend occurring within multiple interpolation techniques when changing the number of anchor points for different models. These results highlight the relevance of correctly choosing the number of anchors to maximize the downstream performance of discrete neural network-based survival models.

Data and Environment

This set of experiments covers the WHAS, GBSG2, METABRIC, and TCGA-BRCA datasets, all described in Section A.1. Each dataset is split into training, validation, and test, accounting for 60%, 20%, and 20% of the total number of samples, respectively. Split operations are performed with stratification on the event variable to ensure the same representation of censored instances on each split. Preprocessing is kept minimal, with standardization of numerical features and one-hot encoding of categorical features.

Models involved in the experiments are DeepSurv, DeepHit, Logistic Hazard, and N-MTLR. Here, only DeepSurv is based on the proportional hazard assumption. This means that it does not admit a tunable number of anchors, as the number of anchors is determined by the Breslow estimator of the baseline function and it is equal to the number of unique time points in the dataset. We opted to apply interpolation procedures to the step-wise function of DeepSurv—computed on the baseline Breslow's estimator and the scalar factor predicted by the neural network—as an additional investigation direction on a high, fixed number of anchors.

Concerning models different from DeepSurv, we adopted a uniform splitting approach. Each neural network comprises two fully connected layers with a hidden size of 32. Each layer is followed by a ReLU activation function and a dropout regularization layer with 0.1 probability. In the experiments, The IBS and the Cumulative AUC were integrated over the 25^{th} and 75^{th} percentiles of the test times, to limit tail noise at the endpoints of the time spectrum. Finally, results are averaged over 30 independent executions.

Results

Figure 4.2 shows the results obtained for a varying number of anchors, from 5 to 1000. To assess how different interpolation techniques perform as the number of anchors increases, we plot the average IBS across all baseline models—namely, DeepHit, Logistic Hazard, and N-MTLR—for each dataset. We selected IBS as the primary metric because it effectively captures the time-dependent aspects of both calibration and partial discrimination. Two key observations emerge.

First, performance consistently degrades when the number of anchors exceeds 100. In fact, the optimal anchor values, depending on the interpolation technique, are typically found between 5 and 10. On top of that, the performance of Step-FWD and Step-BWD (the "no interpolation" techniques) is similar, highlighting that the addition of continuity is essential to improve calibration, as opposed to a fully optimistic or pessimistic approach.

Second, interpolation is beneficial compared to using no interpolation (i.e., Step-FWD and Step-BWD). At lower anchor counts (5–10), interpolation enhances performance, enabling models with fewer anchors to achieve results comparable to those with a larger number of anchors. Conversely, as expected, when the number of anchors exceeds 100, the advantage of interpolation becomes negligible, and no technique consistently outperforms the others.

A Note on Proportional-Hazard Models

Results for CoxPH and DeepSurv, which can be found in [10], have not been reported here. The reason is that in proportional-hazards frameworks, any time-grid interpolation simply rescales the common baseline hazard; the relative effects remain unchanged for all individuals, so every sample receives an identical adjustment. Consequently, the per-sample impact of interpolation is marginal compared with non-proportional models, where each individual's discrete hazard is parameterized separately. For this reason we restrict our interpolation study to the latter family.

Additionally, in all experiments from [10], we used the standard implementations of CoxPH and DeepSurv, whose baseline hazard is estimated at every unique observed event time via the Breslow method and therefore introduces one anchor per failure time. However, many works show that the anchor set can be coarsened or refined also for non-parametric estimators, including Kaplan–Meier, Breslow, and consequently, proportional-hazard models. This choice may improve survival concordance or calibration [22, 23, 21]. A comparison with non-standard preprocessing strategies is addressed in Section 4.2.2.

4.2.2 Assessing Interpolation Techniques

The second set of experiments [9] comes from an extension of the previous study [10] with some novel additions provided in this manuscript. These extension aims at improving the characterization of the differences between interpolation techniques in settings with lower number of anchors, where interpolation is relevant and exhibits a strong impact on model performance, even with data at scale.

Data and Environment

Alongside the datasets considered previously, WHAS500, GBSG2, METABRIC, and TCGA-BRCA, this study includes AIDS, SUPPORT2, and MIMIC to the

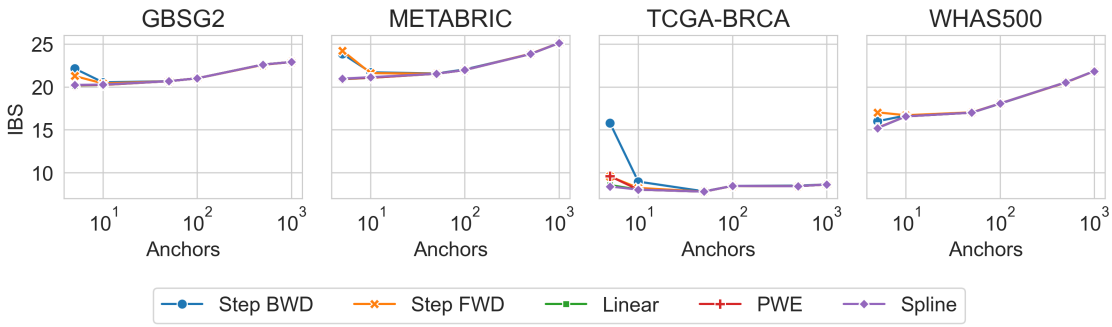

Figure 4.2: IBS (↓) values measured on different number of anchors, averaged over all the discrete baselines considered—DeepHit, Logistic Hazard, and N-MTLR. IBS values are scaled by 100 and the x-axis is in log-scale for readability. Results are averaged over 50 runs.

analysis. These datasets feature a larger number of samples, allowing to assess observations at scale. Splitting and preprocessing follows the same outlined in Section 4.2.1, with stratified train-validation-test splitting and minimal preprocessing.

The considered survival models are Cox PH, DeepSurv, MTLR, N-MTLR, Logistic Hazard, and DeepHit. This time, the number of anchors ranged from 2 to 10, assessing interpolation in the range more relevant to model performance, in accordance with the findings outlined in Section 4.2.1. Concerning proportional-hazard models—namely, Cox PH and DeepHit—we applied time discretization solely to the training set, computing the baseline hazard on discretized times. This approach follows previous studies on interpolation techniques and non-parametric estimation [107, 22, 23, 21].

Lastly, the results cover a single stepwise interpolation method, Step-FWD, as its performance has been shown to be comparable with Step-BWD. Thus, we refer to Step-FWD as Step for simplicity.

Results

Figure 4.3 collects the IBS results analyzing how the performance of survival models varies with the number of anchor points and interpolation methods. Primarily, Step interpolation consistently underperforms at lower anchor counts across all datasets, highlighting how the most common technique in the literature can fail to accurately model the underlying survival distribution. In contrast, all other interpolation methods—Linear, PWE, Spline, and Kaplan–Meier—improve the model calibration, as measured by IBS. For instance, linear interpolation, despite its simplicity, often achieves results comparable to more sophisticated techniques.

An additional observation is related to KM interpolation, which leverages censoring information. KM interpolation demonstrates a distinct advantage in IBS,

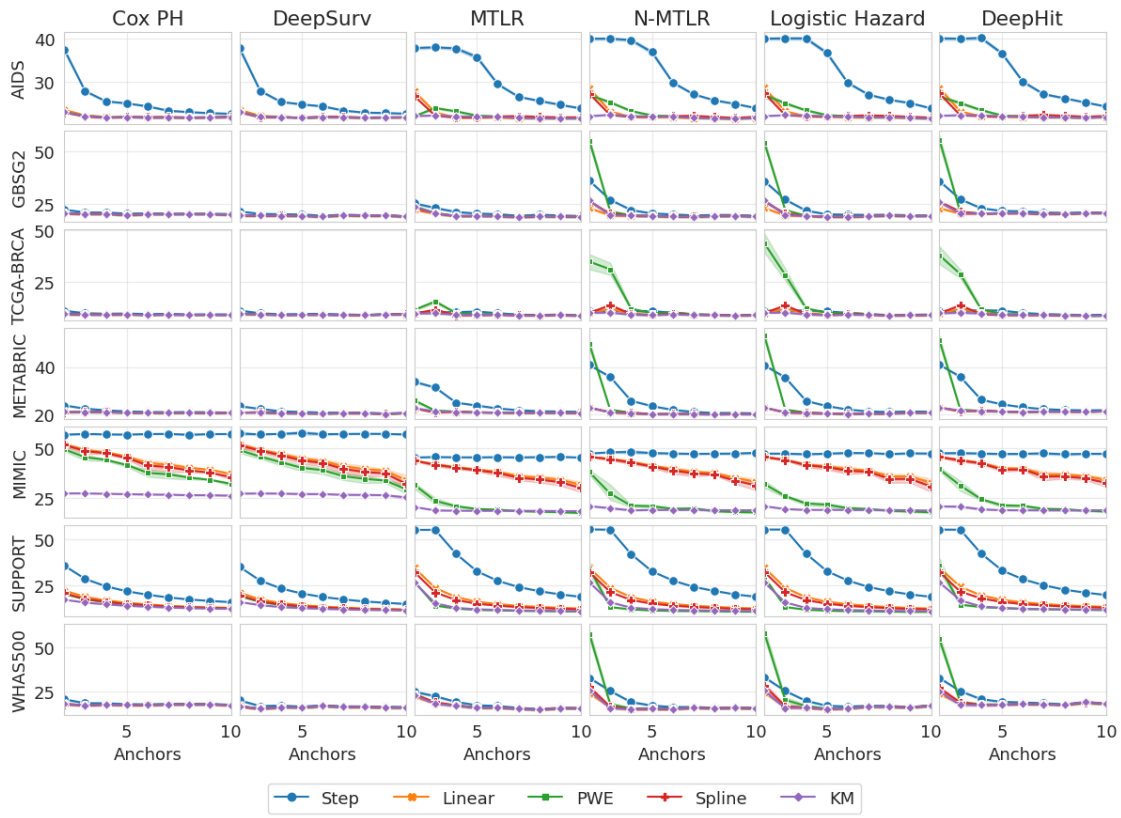

Figure 4.3: IBS (↓) values measured on different number of anchors, reported for all the discrete baselines considered—CoxPH, DeepSurv, MTLR, N-MTLR, Logistic Hazard, and DeepHit. In the legend, Step is a shorthand for Step-FWD. IBS values are scaled by 100 for readability. Results are averaged over 100 runs.

particularly for larger datasets—AIDS, MIMIC, and SUPPORT. This advantage arises due to the sensitive nature of IBS to censoring, partly related to the use of IPCW to weight its computation. Here, KM interpolation effectively leverages censoring information to enhance calibration, bringing an advantage with respect to more straightforward approaches. Thus, when survival data are abundant or when censoring bias is relevant, KM-based interpolation can yield better performance than linear interpolation.

4.3 Discussion

Summarizing the findings from Sections 4.2.1 and 4.2.2, we can derive a set of guidelines for employing interpolation as a lightweight postprocessing step to enhance the performance of neural network-based survival models. These guidelines extend the findings from [107] limited to linear survival models to the broader scope

of neural network-based estimators.

First, we observed that interpolation consistently improves model performance, particularly when the number of anchors is low. Generally, models achieve optimal discrimination and calibration with a lower number of anchors—typically in the range of 5 to 10. Within this range, utilizing a simple interpolation technique, such as linear interpolation, consistently improves calibration across all evaluated scenarios. This improvement can be observed both on proportional and non-proportional hazard models.

Second, the specific choice of interpolation technique does not significantly impact overall model performance. Consequently, simpler methods like linear interpolation, which can be computed locally without requiring additional global estimations, are recommended. However, in cases where calibration is important, the KM interpolation method demonstrated a slightly better performance, particularly in large datasets with fewer anchors. Although KM interpolation necessitates a global estimation step, it remains extremely fast to compute and offers a consistent advantage over alternative interpolation techniques.

Part II

Federated Learning for Survival Analysis

Chapter 5

Federated Learning

The phrase "data is the new oil," attributed to British mathematician Clive Humby back in 2006 and popularized by the Economist in 2017 [37], has proven remarkably prescient. The rapid expansion of AI applications has been fueled by the increasing availability of vast datasets used to train machine learning models. The proliferation of large-scale data sources, coupled with advancements in specialized computing hardware, from GPUs to TPUs, has enabled the scaling of AI models to tackle complex challenges in vision and language processing. Advances such as diffusion and large language models have sparked discussions about progress toward all-purpose AI systems, though equating these advancements to general intelligence may still be premature.

However, much has changed since 2006, particularly regarding security, privacy, and the ethical use of user data. The fragmentation of data remains a critical challenge, leading to the emergence of so-called *data islands*. In domains such as healthcare, finance, and mobile applications, data are generated and stored in a distributed manner but cannot be freely shared due to privacy concerns, regulatory constraints, and logistical barriers. These restrictions make centralized data collection impractical, hindering the development of data-driven models that require access to diverse and representative datasets [5, 102].

Federated Learning (FL) [75, 57] has emerged as a promising solution to this challenge, enabling the collaborative training of machine learning models without requiring raw data to be exchanged. Unlike traditional machine learning paradigms that rely on centralized data aggregation, FL allows multiple decentralized clients—ranging from hospitals and financial institutions to smartphones and IoT devices—to contribute to a shared model while keeping their data local. Instead of transmitting sensitive data, FL collects only model updates and orchestrates the learning process through a central server. This approach enhances privacy by ensuring that data remain undisclosed to external entities, reduces communication overhead by sharing only model weights rather than entire datasets, and assists organizations in complying with data protection regulations such as GDPR [2].

Since its inception in 2016, FL has demonstrated success across various healthcare applications and moved well beyond proof-of-concept. Recent reviews of smart-health systems and IoT wearables highlight how FL simultaneously satisfies strict privacy regulations and delivers state-of-the-art model accuracy across heterogeneous institutions [28]. Additionally, it enables collaborative predictive modeling across hospitals while preserving patient confidentiality in clinically complex scenarios, such as cancer genomics [5, 81, 94], stroke detection [35], and COVID-19 survival [113]. In fact, one the most recent bibliometric analyses of healthcare papers related to federated learning confirms a sharp surge of real-world deployments—spanning disease diagnosis, risk assessment, medical-image analysis and drug discovery [83]. Collectively, these studies converge on two decisive advantages for practical FL applications: (i) regulatory compliance via on-premise training, and (ii) better generalization than any single-site or synthetic-data alternative, thanks to the natural diversity of decentralized cohorts.

5.1 The Federated Scenario

Federated learning aims to optimize a global objective function using data spread across multiple nodes, while preserving local data privacy. Rather than sharing the raw data, each node—often referred to as a client—executes local computations and communicates only the resulting updates to a central server. In the simplest formulation, we consider a federation consisting of K clients, where each client k holds a local dataset $\mathcal{D}_k$ with N_k samples. Defining $N = \sum_{k=1}^{K} N_k$ as the total number of samples across all clients, federated learning seeks to find model parameters θ that minimize the global loss function $\mathcal{L}(\theta)$. The most common approach is to optimize the weighted sum of local losses as

$$\min_{w} \mathcal{L}(\theta) = \sum_{k=1}^{K} w_k \mathcal{L}_k(\theta), \tag{5.1}$$

where w_k is the relative contribution of client k (often taken as $w_k = \frac{N_k}{N}$) and $\mathcal{L}_k(\theta)$ is the local loss function computed on $\mathcal{D}_k$.

Real-world federated learning scenarios include *cross-device* settings, where a large number of low-power personal devices (e.g., smartphones or IoT sensors) collaborate to train models such as next-word prediction or recommendation systems, while keeping users' data on their own devices. Thus, in cross-device settings, K is high, but $\mathcal{L}_k$ must be computationally inexpensive.

On the other side of the spectrum, *cross-silo* federated learning involves a small number of organizations (e.g., hospitals or financial institutions) jointly training models on hardware with high computational power. Therefore, in cross-silo scenarios, K is expected to be small—on the order of tens or at most hundreds

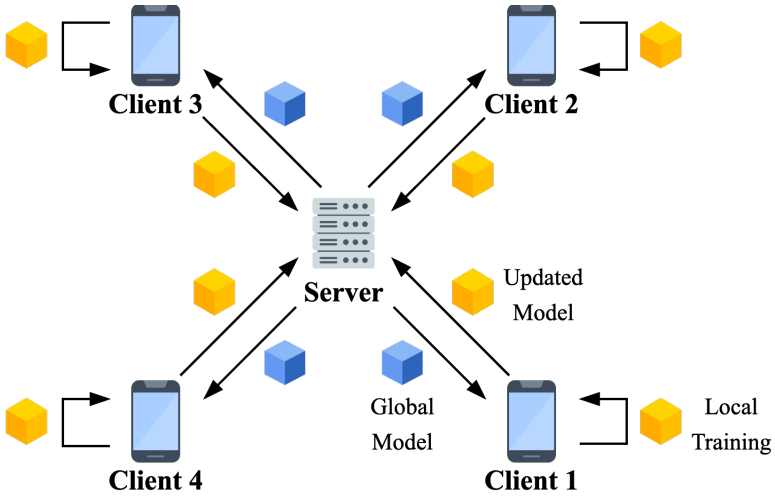

Figure 5.1: Basic scheme of a client–server interaction within a round of federated averaging. First, the server broadcasts a global model. Second, each—possibly sampled—client trains the received model on local data for few epochs. Finally, local models are sent back to the server and aggregated.

of clients—and each $\mathcal{L}_k$ could potentially be more computationally expensive to work with.

5.2 Federated Averaging

Federated Averaging (FedAvg) [87] is one of the most widely adopted optimization algorithms in federated learning. It extends the conventional mini-batch Stochastic Gradient Descent (SGD) by allowing each client to perform multiple local updates before sending model parameters back to a central server. FedAvg addresses the minimization problem in Equation (5.1) iteratively over a total of M rounds. The main steps of the FedAvg algorithm are depicted in Figure 5.1.

At the beginning, a global model with parameters θ^0 is initialized on the server. Then, at iteration m, the server broadcasts the current model parameters θ^m to a subset of $K^m \leq K$ clients, denoted by $\mathcal{S}^m$. Depending on the application and type of federation, this subset may be a small fraction of all clients to limit communication overhead (typical of cross-device setups) or, in contrast, it may include all clients in every round (common in cross-silo environments).

Each selected client $k \in \mathcal{S}^m$ trains the received model using local SGD over a small number of epochs:

$$\theta_k^{m+1} = \theta^m - \eta \nabla \mathcal{L}_k(\theta^m),$$

where η is the local learning rate and $\mathcal{L}_k$ is the loss function computed on dataset $\mathcal{D}_k$ from client k. The updated parameters θ_k^{m+1} are then sent back to the server

Algorithm 2 FedAvg Algorithm

Require: Federation of K clients with local datasets $\{\mathcal{D}_k\}_{k=1}^{K}$;
total number of global rounds M; learning rate η;
selection strategy for client subsets $\mathcal{S}^m \subseteq \{1, \ldots, K\}$.
Ensure: Final model parameters θ^M.

Initialize global model parameters θ^0.
for $m = 0$ to $M - 1$ **do**
  Server selects a subset of clients $\mathcal{S}^m \subseteq \{1, \ldots, K\}$.
  Server broadcasts θ^m to each client $k \in \mathcal{S}^m$.
  for each client $k \in \mathcal{S}^m$ **in parallel do**
    Client updates local model:

$$\theta_k^{m+1} \leftarrow \theta^m - \eta \nabla \mathcal{L}_k(\theta^m).$$

    Client sends updated parameters θ_k^{m+1} back to the server.
  end for
  Server aggregates local updates:

$$\theta^{m+1} \leftarrow \sum_{k \in \mathcal{S}^m} w_k \, \theta_k^{m+1} \quad \text{with} \quad w_k = \frac{N_k}{\sum_{j \in \mathcal{S}^m} N_j}.$$

end for
return θ^M.

for aggregation. The server aggregates all received updates via a weighted average:

$$\theta^{m+1} = \sum_{k \in \mathcal{S}^m} w_k \, \theta_k^{m+1},$$

where w_k is typically $\frac{N_k}{N}$, i.e., the ratio of client k's dataset size to the total number of samples in the federation. These steps (broadcast, local training, and aggregation) repeat for M rounds or until convergence.

Although simple, FedAvg has empirically demonstrated excellent performance in many federated scenarios, offering results comparable to training on the entire dataset in one location—especially when data across clients share similar statistical properties. Algorithm 2 highlights the steps of the FedAvg procedure.

5.3 Taxonomy of Federated Learning Techniques

Federated learning techniques can be classified according to multiple criteria, including how data are distributed across clients, the communication architecture

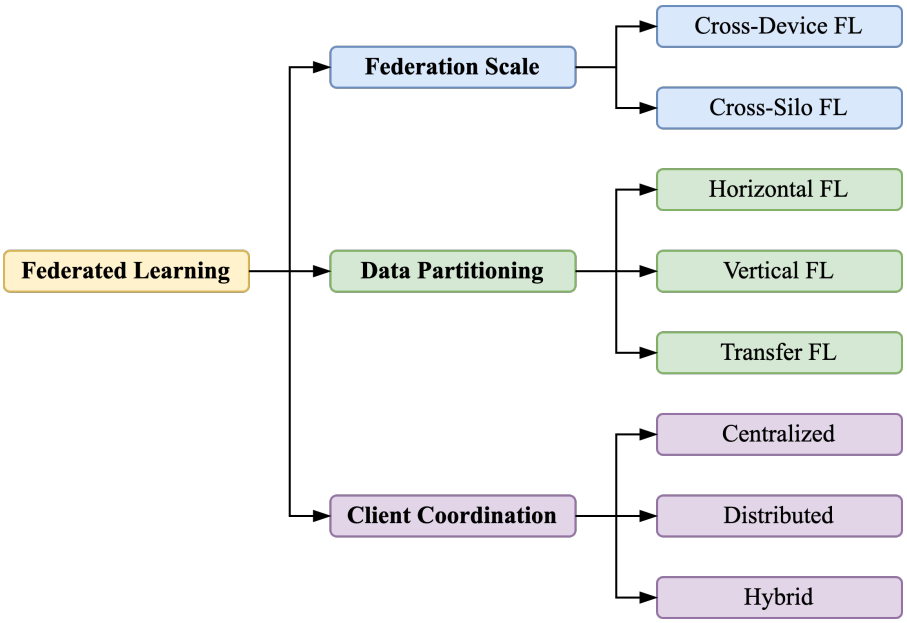

Figure 5.2: Overview of macro-areas related to federation types within the FL literature.

used to coordinate training, and the extent to which knowledge transfer is exploited. This section provides a concise taxonomy of the most widely discussed federated learning paradigms and their salient properties, summarized in Figure 5.2.

5.3.1 Federation Scale: Cross-Silo vs. Cross-Device

A frequently used distinction in federated learning arises from the nature and number of clients involved. As previously stated, *cross-silo* FL typically occurs among a relatively small set of reliable and resource-rich institutions or data centers (e.g., hospitals, banks, universities) that collaborate to train a model without sharing raw data. This approach leverages predictable and generally robust communication channels as well as substantial computational capabilities on each client. Consequently, many protocols and algorithms in cross-silo settings can assume that each client is almost always available and stateful (i.e., participating in training at multiple stages). Example tasks include collaborative research across several hospitals for improved diagnostic modeling or fraud detection in multiple financial institutions sharing sensitive records they cannot pool in a central repository.

On the other hand, *cross-device* FL addresses scenarios with potentially thousands or millions of personal devices, each storing heterogeneous data under limited computational resources and intermittent availability. This setting arises naturally in large-scale consumer applications, such as next-word prediction for smartphone

keyboards. Unlike the cross-silo environment, cross-device clients are typically stateless (participating only once, due to limited battery or connectivity), and training protocols must accommodate frequent device unavailability and high communication cost.

5.3.2 Data Partitioning: Horizontal, Vertical, and Transfer Learning

Another angle of classification focuses on the structure of the data partition itself. *Horizontal* (or example-partitioned) FL arises when all clients share the same feature space but hold disjoint subsets of examples. This is common for device-level FL in consumer applications, where each phone or IoT sensor logs similar feature types but on different samples over time.

In *vertical* (or feature-partitioned) FL, the same set of users may be split across multiple entities holding different, non-overlapping features. For instance, one company may have comprehensive profile information about each user but lack additional transactional or clinical data that another organization holds.

Finally, *transfer* FL encompasses situations in which clients may only partially overlap in both user and feature-space. A prominent example is adapting a broadly trained global model to a specialized local task, such as personalizing a language model for a single user's unique typing habits.

5.3.3 Client Coordination: Centralized and Distributed Approaches

A final dimension of federated learning taxonomy addresses how model updates are aggregated and how the system is orchestrated. *Centralized* FL coordinates all training and model aggregation through a single server or service provider. As in standard FedAvg, each client locally computes a gradient update on its private data, and the central server applies a secure aggregation protocol to combine these updates. Centralized systems simplify orchestration but require trust in one aggregation node.

Distributed or *fully decentralized* FL removes the need for a central server by arranging clients in a peer-to-peer topology. Here, subsets of neighbors exchange model parameters or gradients directly, gradually propagating updates throughout the network until some notion of consensus is reached. Such systems can be attractive when a fully trusted central party is unavailable, although they may require additional mechanisms, such as blockchain, to handle unreliable connections, malicious nodes, and efficient topology management.

Hybrid FL combines elements of both centralized and decentralized coordination, often through hierarchical structures. For instance, edge nodes or regional servers can aggregate local updates from nearby clients, then periodically synchronize with

a central cloud service.

5.4 Challenges in Federated Learning

Recent studies have highlighted the limitations of FedAvg in practical scenarios. In particular, the algorithm struggles with extreme data heterogeneity, as local model updates may diverge significantly, leading to slower convergence or poor generalization. Furthermore, partial client participation and system instability can introduce additional optimization challenges. Researchers have proposed various extensions to FedAvg, including adaptive aggregation techniques and personalized FL methods to address these issues. This section discusses three primary areas of concern in federated learning: data heterogeneity, system heterogeneity, and security.

5.4.1 Data Heterogeneity

Data heterogeneity in FL refers to how each client k holds a local data distribution $\mathcal{D}_k$ that can differ substantially from others, either in terms of label distribution, feature space, or data quantity. More formally, let θ denote the global model parameters. Then, each client's local objective can be written as

$$\mathcal{L}_k(\theta) \;=\; \mathbb{E}_{\mathcal{D}_k \sim \xi_k}\Big[\mathcal{L}(\theta \mid \mathcal{D}_k)\Big],$$

where ξ_k identifies a data generation process potentially widely different for each client. As a result, a single global model update as in standard FedAvg may be pulled in incompatible directions by different local losses. Such behavior creates several distinct challenges:

- *Gradient divergence*: Clients may produce updates that are inconsistent, slowing convergence or causing training instability.
- *Personalization needs*: When data distributions differ greatly (e.g., different usage patterns across regions), a single global model may be suboptimal compared to locally fine-tuned or personalized variants.
- *Partial participation*: Only a fraction of clients participates at each round, adding an additional layer of sampling variance.

FedProx

One of the early methods proposed to handle data heterogeneity is FedProx [76]. FedProx introduces a proximal term in the local objective of each client to limit the

magnitude of local updates and the divergence with respect to the global model. Specifically, instead of minimizing $\mathcal{L}_k(\theta)$ alone, client k minimizes

$$\mathcal{L}_k(\theta) \; + \; \tfrac{\mu}{2} \, \|\theta - \bar{\theta}\|^2,$$

where $\bar{\theta}$ is the current global model broadcast by the server, and $\mu > 0$ is a hyper-parameter controlling the amount of regularization. This approach encourages each client to remain close to the global parameters, constraining large local drifts. Empirical studies have demonstrated that FedProx can improve stability and convergence, especially when the number of local updates is large.

SCAFFOLD

Another approach to mitigating data heterogeneity is SCAFFOLD [62], which introduces control variates to correct for client drift. Specifically, each client k maintains and updates two parameter vectors: θ_k (its local copy of the global model) and a control variate c_k that tracks local gradient information. The server also holds a global control variate c_s. By exchanging these variates, SCAFFOLD reduces the bias introduced by client drift in the presence of non-i.i.d. data.

FedOpt

While FedAvg aggregates client updates via a simple (possibly weighted) average, FedOpt [100] uses more sophisticated server-based optimizers to better adjust the global model across rounds. For instance, FedOpt can apply momentum, Adam-like updates, or other modifications at the server after collecting client updates. By treating the global model update itself as an optimization variable, these strategies have shown improved robustness to heterogeneous data distributions and faster convergence.

5.4.2 System Heterogeneity

Federated learning often occurs in environments where each client device has different computational capabilities, communication bandwidths, and availability patterns. These differences are collectively referred to as system heterogeneity.

Specifically, devices can differ in terms of *computational resources*: some devices may have powerful accelerators tailored for deep learning, while others may not have accelerators at all, memory in the order of Megabytes, struggle with slow networks, limited memory, or quickly draining batteries. Ensuring that all participants can train locally without straining resources is nontrivial.

In cross-device FL, only a fraction of eligible clients is typically online at any one time. This partial *availability* can bias the training process if certain device profiles frequently fail to appear, and it introduces intermittent node dropout.

Another issue is *straggler clients* and fault tolerance. Long-tail distributions of device speeds or sudden dropouts can hinder synchronous protocols. Asynchronous or semi-asynchronous methods are studied to mitigate straggling effects, but they often complicate privacy and security guarantees.

Scheduling strategies, adaptive local workloads (e.g., adaptive mini-batch sizes), and straggler-resilient methods have all been proposed to handle these aspects of system heterogeneity. In practice, ignoring slow clients can accelerate training, but may degrade model quality if certain demographics or behaviors are systematically underrepresented. For cross-silo FL (where clients are large data centers or institutional servers), clients can often be considered fully reliable; however, heterogeneity in compute and data sharing policies may still arise and demand additional solutions.

5.4.3 Security

Federated learning continuously faces security threats that can compromise training integrity, system availability, and data confidentiality, revealing sensitive information. In the following, we focus here on two significant attack types: data poisoning and membership inference.

Data Poisoning Attacks

Data poisoning occur when a malicious client modifies or inserts carefully crafted samples into one or more local datasets to drive the training process away from the optimum. Since federated learning relies on local training data from potentially untrusted participants, even a small fraction of compromised clients can degrade or corrupt the global model, without a straightforward way for the server to detect poisoning attacks. Common poisoning strategies include

- *Global degradation*: The adversary aims to lower accuracy on the entire global task, for example by injecting random or mislabeled data.
- *Backdoor attacks*: The adversary introduces trigger patterns in local data to embed a hidden classification rule. The global model behaves normally on clean inputs but misclassifies inputs containing the trigger.

Research on robust aggregation rules (e.g., coordinate-wise median, trimmed mean, or spectral methods) has shown promise in resisting some data poisoning attacks. However, strong adversaries may circumvent robust estimators, so systematic evaluations remain critical. Additionally, synergy between privacy mechanisms (e.g., differential privacy) and robust optimization is an active research area, since noise or clipping can inadvertently reduce or improve resilience to poisoning, depending on the scenario.

Membership Inference Attacks

Membership inference attacks (MIAs) aim to determine whether a specific data record was used to train a target model [105, 50]. These attacks pose a significant privacy risk: knowing that a certain sample (e.g., medical record) contributed to training can reveal sensitive information about the individual to whom the sample belongs. Since large machine learning models often capture fine-grained details of their training data, MIAs are particularly relevant in scenarios where models are publicly accessible, such as machine-learning-as-a-service platforms.

A common and effective strategy to launch a MIA is the *shadow-model* approach [105]. First, an attacker trains multiple models, called shadow models, to mimic the behavior of the model under attack. Each shadow model is trained on a known dataset split into two parts: one that acts as its shadow training set and another that acts as its shadow test set. By observing how the predictions of each shadow model differ for members (training samples) versus non-members (test samples), the attacker collects a labeled dataset of prediction vectors. Subsequently, a separate binary classifier—often called *discriminator*—is trained on these prediction vectors to decide whether a given instance was in or out of the training set. Once trained, this discriminator can be applied to predictions from the real target model to infer membership. In practice, membership inference can serve as starting point for more advanced attacks, highlighting the importance of designing countermeasures that prevent unauthorized disclosure of training data.

Techniques like differential privacy at the user-level, secure aggregation, and limiting direct access to model updates (e.g., restricting the frequency of model checkpoints) can all mitigate membership inference risks. Nonetheless, the tension between utility and privacy persists: strong defenses typically introduce more noise or constraints on training, potentially reducing model accuracy.

Other Security Threats

Beyond data poisoning and membership inference, FL systems may face many additional threats. Among those, the main attacks are

- *Model inversion attacks*: Adversaries aim to recover representative features or class prototypes from a trained model.
- *Sybil attacks*: Malicious entities create multiple fake clients, amplifying their ability to poison data or manipulate the global model by controlling a larger proportion of the participating clients.

Defenses include cryptographic algorithms (secure aggregation or homomorphic encryption), trust-based mechanisms (reputation scores), and robust learning algorithms that disregard outlier updates. However, each countermeasure entails trade-offs in computational efficiency, communication overhead, and model accuracy.

5.5 Federated Survival Analysis

The emergence of federated learning has generated remarkable interest in the medical community. The capability to train models on distributed data sources subject to strict data-sharing constraints is a key opportunity for data-driven healthcare. Indeed, the principal obstacles to building robust and reliable clinical models at scale are often related to data availability and quality, which, in many real-world scenarios, remain limited.

Survival analysis can particularly benefit from the increased data access provided by FL. In survival analysis, many samples are censored, thereby only capturing a fraction of the individual lifespans. Moreover, these studies commonly investigate long-term survival outcomes, requiring extensive observation periods and repeated measurements spread over time. Such time-consuming assessments also risk patient dropout or the occurrence of competing events that complicate the study objectives. Consequently, having larger datasets is essential for training reliable survival models. In practice, federated learning has been applied successfully to clinical survival studies across a variety of conditions, including oncology [5, 42, 26, 46], heart failure [35], and COVID-19 [113].

5.5.1 Federated Cox Models

As introduced in Section 2.5.2, the Cox model [27] is one of the most influential frameworks in survival analysis; indeed, its seminal paper is the 24$^{\text{th}}$ most cited research article in history, according to a 2014 study from Nature [91]. For these reasons, the earliest work on federated survival analysis aimed to extend the Cox model into the federated domain [80, 5, 35, 14, 30, 46, 59, 85, 113, 53, 119].

Although the Cox model is efficient, conceptually straightforward, and semi-parametric (i.e., it does not require specifying the baseline hazard), its application in federated scenarios poses several challenges. First, the partial likelihood loss requires careful handling, as each term in the likelihood involves knowledge of the entire risk sets across federated nodes. Second, the linear dependence between features and individual risk estimates can limit the expressiveness of the model in complex, heterogeneous clinical data settings. Consequently, various extensions have emerged to address these limitations. For instance, some approaches employ the discrete survival framework, formulating Cox-based methods in a classification setting [5, 119]. Other works propose surrogate losses that aggregate patient information locally before sharing it across the federation [35]. Additionally, several contributions focus on vertical federated learning scenarios, where data are split by features rather than samples, leading to secure collaborative training protocols [30, 59, 53].

5.5.2 Other Federated Survival Models

Beyond the Cox model, additional survival methods have been extended to federated settings. A notable example from the non-parametric family is FAMHE [42], which introduces a privacy-preserving protocol to compute Kaplan–Meier estimators on federated datasets. Postprocessing mechanisms for differentially private models have also been investigated [97], where the magnitude of federated parameter updates is tuned at each iteration to mitigate the effect of noise introduced by federated updates and differential privacy. Other research directions include applying weakly supervised attention layers to enhance survival predictions [81] and leveraging pseudo-values [98] as surrogate labels to reformulate survival tasks as regression problems.

5.5.3 Privacy Protection

Privacy preservation is paramount in any federated learning application. This is especially true for survival analysis, where medical data are strictly regulated and patient confidentiality is critical. Consequently, several privacy-preserving techniques have been adopted or tailored for federated survival models. A common technique to ensure privacy preservation is differential privacy [36]. By introducing well-calibrated noise during model training, differential privacy mathematically guarantees that adding or removing a single individual from the dataset does not substantially alter the outcome of the training procedure, thereby limiting the risk of data leakage [81, 97].

Secure multi-party computation (SMPC) is another common approach to protect privacy. SMPC designs protocols based on the exchange of privacy-safe intermediate information to perform distributed computations. SMPC has been applied to distributed partial log-likelihood optimization problems based on the Newton–Raphson method [59]. Alternative methods compute additional partial statistics during training to ensure no private information is exposed [84, 53].

Lastly, FAMHE [42] exploits homomorphic encryption (HE), a cryptographic method allowing certain operations (e.g., additions, multiplications) to be performed directly on encrypted data. Once decrypted, the resulting output is equivalent to the same operations performed on unencrypted data. While HE offers a strong level of protection, its computational overhead has limited wide adoption.

Chapter 6

FedSurF: Federated Survival Forests

While chapters from Part I have focused exclusively on survival analysis, this chapter extends our investigation to the realm of federated learning. The potential of federated learning for data-driven medical applications has been extensively commented on throughout this manuscript and many research directions can still be further explored. Indeed, the research path presented in this chapter—centered on federated ensemble methods—was a pivotal factor in directing our subsequent work towards survival analysis.

Our interest was particularly sparked by the challenges inherent in federated learning, especially the complexities arising from heterogeneous datasets. Training algorithms on decentralized data distributed across clients with differing underlying patterns presents a formidable yet intriguing problem. Addressing this challenge required a deep exploration of distributed optimization techniques and advanced statistical methodologies, approached through the lens of survival analysis.

The studies discussed in this chapter collectively contribute to a comprehensive set of techniques for tree-based federated survival analysis. First, we developed an approach for constructing realistic federated survival settings with a controllable level of heterogeneity [12]. Building on this foundation, we proposed a federated survival algorithm based on random survival forests [8] and subsequently extended this method by incorporating an improved bagging strategy [7]. Together, these contributions form the research arc that narrates the development of Federated Survival Forests.

6.1 The FedSurF Algorithm

Federated Survival Forests (FedSurF) is a federated ensemble-based algorithm designed to extend Random Survival Forests (RSF) to decentralized settings. Its

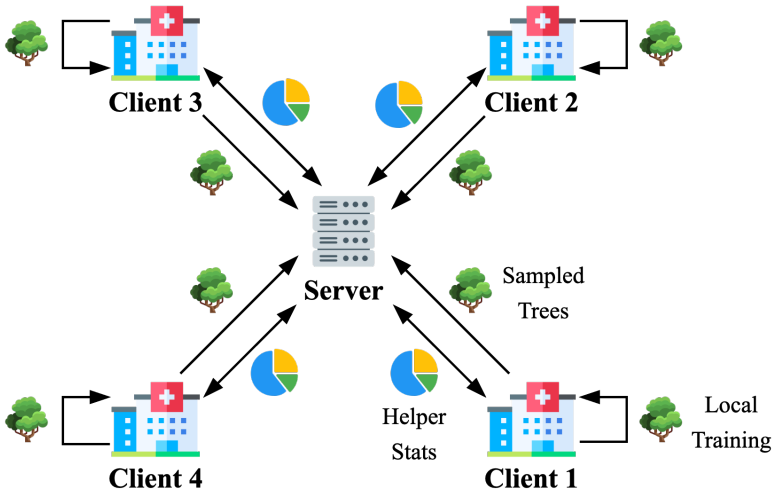

Figure 6.1: Basic scheme of the client–server interaction of FedSurF. First, clients communicate to the server the number of samples available locally. Then, the server assigns a target number of trees to each client. Clients train locally a survival forest and select in probability the best trees to send back. Finally, the server aggregates the received trees into a global survival forest.

primary objective is to train local forests on client data and combine only the best trees into a global ensemble. A central server orchestrates this process by sampling trees from each client, based on local performance metrics. This approach ensures that the final federated model emphasizes higher-performing trees while limiting communication overhead to a single round. FedSurF is composed of three steps: local training, tree assignment, and tree sampling, highlighted in Figure 6.1.

Local Training. Consider a federation of K clients, each owning a survival dataset $\mathcal{D}_k$ for $k \in \{1, \ldots, K\}$. Each client k trains a RSF $\mathcal{E}_k = \{\tau_j\}_{j=1}^{J_k}$ on its local dataset $\mathcal{D}_k$. Here, J_k denotes the number of survival trees in the local forest. Each tree τ_j is built following the CART approach [17], where node splits are determined by maximizing the survival difference via the logrank test. Each leaf node stores the Nelson–Aalen estimator, and the forest prediction is taken as the average of the predictions from its leaves. During this phase, each client can independently tune hyperparameters (e.g., the maximum number of trees or tree depth) to accommodate its own computational constraints.

Tree Assignment. The server then determines the number of trees to request from each client to form the global model. Let J be the target total number of trees in the final ensemble, satisfying $J \leq \sum_{k=1}^{K} J_k$. The server initializes counters

$C_k \leftarrow 0$ for each client k. A client is selected for tree inclusion with probability

$$P(k) \;=\; \frac{1(C_k < J_k) \cdot |\mathcal{D}_k|}{\sum_{k':\, C_{k'} < J_{k'}} |\mathcal{D}_{k'}|},$$

where $1(C_k < J_k)$ is an indicator function that excludes clients whose counters have already reached their local maximum J_k. After selecting a client k, the server increments C_k by 1. This procedure is repeated until $\sum_{k=1}^{K} C_k = J$. By weighting the probabilities according to $|\mathcal{D}_k|$, FedSurF imitates the weighting strategy from FedAvg, giving more influence to clients owning larger datasets.

Tree Sampling. Finally, each client k samples C_k trees from its local forest $\mathcal{E}_k$ without replacement. The sampling probability is proportional to a performance metric $\mu(\tau_j)$ measured on a local hold-out set, for example, the concordance index or the integrated Brier score. Specifically, a tree τ_j is chosen with probability

$$P(j) \;=\; \frac{\mu(\tau_j)}{\sum_{j'=1}^{J_k} \mu(\tau_{j'})}.$$

Once sampled, the selected trees are sent to the server, which aggregates them into a global ensemble $\mathcal{E}$. Importantly, only one communication round is required, significantly reducing the bandwidth cost of FedSurF with respect to gradient-based methods. The complete procedure is summarized in Algorithm 3.

6.2 Computational Complexity

Analyzing computational complexity in a federated environment can be challenging due to real-world factors (e.g., communication costs, parallelization strategies) that go beyond traditional worst-case analysis. Nonetheless, a rough estimate helps to identify where practical improvements might have the greatest impact.

FedSurF requires training J_k survival trees on each client k. Training one survival tree via the CART procedure on $|\mathcal{D}_k|$ samples and F features involves evaluating node splits. For a node with N samples, computing the logrank test for each feature has $O(N)$ cost, repeated for $\sqrt{F}$ randomly sampled features, and sorting takes $O(N \log(N))$. In the worst case, this might be repeated for N nodes, yielding an upper-bound complexity of $O(\sqrt{F}\, N^2 \log(N))$. However, in practice, the implementations of survival trees promote reasonably balanced structures or have bounded depth λ. Hence, a more realistic total complexity is

$$O\Big(\lambda \sqrt{F} \max_{k \in \{1,\ldots,K\}} \big\{ |\mathcal{D}_k|^2 \log(|\mathcal{D}_k|) \big\}\Big).$$

where the max operator accounts for the slowest participant, since all clients train in parallel.

Algorithm 3 FedSurF Training Algorithm

Require: Federation of K clients with local datasets $\{\mathcal{D}_k\}_{k=1}^{K}$;
Local RSF configuration for each client (e.g., number of trees J_k);
Global ensemble size $J \leq \sum_{k=1}^{K} J_k$;
Survival metric $\mu(\cdot)$ (e.g., C-index, IBS), for tree sampling.
Ensure: A federated survival forest $\mathcal{E}$ consisting of J trees.

Initialize counters $C_k \leftarrow 0$ for all $k \in \{1, \ldots, K\}$.
for $k = 1$ to K **in parallel do**
  Train a local RSF $\mathcal{E}_k = \{\tau_j\}_{j=1}^{J_k}$ on dataset $\mathcal{D}_k$.
end for
while $\sum_{k=1}^{K} C_k < J$ **do**
  Select a client k with probability:

$$P(k) \;=\; \frac{1(C_k < J_k) \cdot |\mathcal{D}_k|}{\sum\limits_{k' : C_{k'} < J_{k'}} |\mathcal{D}_{k'}|}.$$

  $C_k \leftarrow C_k + 1$.
end while
for $k = 1$ to K **in parallel do**
  For each tree $\tau_j \in \mathcal{E}_k$, compute its performance $\mu(\tau_j)$ on a local hold-out set.
  Sample C_k trees without replacement from $\mathcal{E}_k$ with probability:

$$P(j) \;=\; \frac{\mu(\tau_j)}{\sum_{j'=1}^{J_k} \mu(\tau_{j'})}.$$

  Send these C_k selected trees to the server.
end for
Collect all sampled trees into the global ensemble $\mathcal{E}$.
return $\mathcal{E}$

6.3 Handling Data Heterogeneity

One of the main challenges in developing FedSurF was training models in environments that accurately reflect real-world conditions, where each client may have a distinct data distribution. This issue is especially relevant in medical survival analysis, as healthcare data often reside in separate institutions with stringent privacy regulations but high-performance computational infrastructures. Although federated learning is highly appealing for these scenarios, each institution typically represents a specific cohort of patients with shared characteristics (e.g., ethnicity, disease status, or geographical location), thereby generating a high degree of data

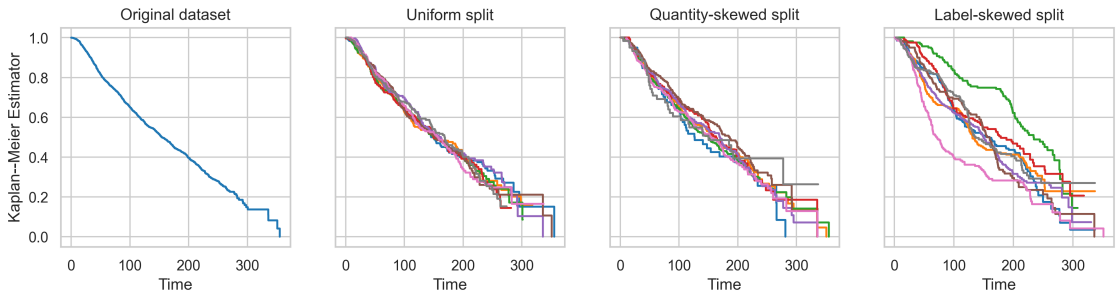

Figure 6.2: Kaplan–Meier curves computed on the METABRIC dataset split in three different ways. From the left, the first plot shows the KM estimator on the entire dataset. Second, we show an example of uniform splitting between 10 clients. Third, we assign samples resulting in different data quantities for each client. Finally, we explicitly introduce heterogeneity by assigning populations with different survival patterns to each client. For both quantity and label-skewed splits, we set $\alpha = 1.8$.

heterogeneity. In such cases, differing client model updates may impede convergence.

To address this, we developed a procedure to artificially impose heterogeneity across client datasets, motivated by the lack of preexisting federated survival analysis benchmarks beyond Fed-TCGA-BRCA [94]. The level of heterogeneity is controlled by a hyperparameter, and it can be quantified using a dedicated heterogeneity metric.

6.3.1 Building Custom Federated Datasets

Let $\mathcal{D}$ denote a dataset of N survival analysis samples, defined as

$$\mathcal{D} = \{(\mathbf{x}_i, \delta_i, t_i)\}_{i=1}^{N}.$$

We aim to distribute these N samples across K clients, leading to K disjoint subsets $\mathcal{D}_k$ satisfying

$$\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}_k.$$

We introduce heterogeneity in two main ways: (1) by constructing clients with differing numbers of samples and (2) by shaping clients to exhibit diverse survival patterns. The results of these sample assignments are highlighted in Figure 6.2.

Quantity-Skewed Splitting

The first method, called *Quantity-Skewed Splitting*, controls the number of samples allocated to each client, producing some clients that hold very few samples.

This helps evaluate federated models' robustness to data imbalance. We first sample K probabilities from a Dirichlet distribution:

$$\mathbf{p} \;=\; \mathrm{Dir}(\alpha\,\mathbf{1}_K),$$

where $\mathbf{p}$ is a K-dimensional vector of probabilities, $0 \le \mathbf{p}[k] \le 1$ with $\sum_{k=1}^{K} \mathbf{p}[k] = 1$, Dir denotes the Dirichlet distribution, $\mathbf{1}_K$ is a K-dimensional vector of ones, and $\alpha > 0$ is a hyperparameter controlling heterogeneity. A lower α induces more uneven sample distributions, whereas $\alpha \to \infty$ yields subsets of nearly equal cardinalities. Given $\mathbf{p}$, each sample is assigned to client k with probability $\mathbf{p}[k]$:

$$P\Big((\mathbf{x}_i, \delta_i, t_i) \in \mathcal{D}_k\Big) \;=\; \mathbf{p}[k].$$

Label-Skewed Splitting

In the second approach, called *Label-Skewed Splitting*, we assign samples according to their event or censoring times t_i. This results in different survival patterns across clients (e.g., higher or lower survival probabilities), allowing us to evaluate how federated models handle label-distribution shifts.

We start by dividing the dataset timeline into B time bins via anchor times $\tau_0, \ldots, \tau_B$, computed either uniformly or via time quantiles as detailed in [70]. Next, for each bin $b = 1, \ldots, B$, we draw a probability vector $\mathbf{p}_b$ from a Dirichlet distribution,

$$\mathbf{p}_b \;=\; \mathrm{Dir}(\alpha\,\mathbf{1}_K),$$

where $\alpha > 0$ is a global heterogeneity hyperparameter. Each sample is then assigned to client k with probability $\mathbf{p}_b[k]$, depending on the time bin to which that sample belongs:

$$P\Big((\mathbf{x}_i, \delta_i, t_i) \in \mathcal{D}_k\Big) \;=\; \mathbf{p}_b[k].$$

As before, lower values of α increase heterogeneity among clients, whereas $\alpha \to \infty$ produces uniform label distributions across all clients.

6.3.2 Assessing Federation Heterogeneity

To determine whether two client datasets differ significantly, we use the logrank test (Section 2.5.1). We choose to reject the null hypothesis that the two distributions are statistically indistinguishable when the p-value is below 0.05. To summarize overall federation heterogeneity, we compute the average pairwise heterogeneity score:

$$\mathcal{H}(\{\mathcal{D}_k\}_{k=1}^{K}) \;=\; \frac{1}{|\mathcal{P}|} \sum_{k_1,k_2 \in \mathcal{P}} 1\Big(\mathrm{logrank}(\mathcal{D}_{k_1}, \mathcal{D}_{k_2}) \le 0.05\Big),$$

where $\mathcal{P} = \{(k_1, k_2) \mid k_1 < k_2,\, k_1, k_2 = 1, \ldots, K\}$ is the set of distinct unordered pairs of clients, and $1(\cdot)$ is the indicator function. Consequently, $\mathcal{H}(\cdot)$ measures the

Table 6.1: Mean ($\pm$ standard deviation) heterogeneity score $\mathcal{H}(\cdot)$ for quantity-skewed splits with $K = 10$ across varying α.

Dataset	$\alpha = 10^3$	$\alpha = 10^2$	$\alpha = 10$	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.1$
GBSG2	4.1 ± 4.8	3.9 ± 5.0	3.0 ± 4.9	3.0 ± 4.3	3.0 ± 4.5	1.9 ± 3.3
METABRIC	3.6 ± 5.3	4.7 ± 6.0	4.4 ± 5.9	3.5 ± 5.6	3.6 ± 4.7	1.7 ± 4.0
AIDS	4.1 ± 5.6	4.5 ± 5.5	3.8 ± 5.4	4.5 ± 6.4	4.5 ± 6.3	2.3 ± 3.6
FLCHAIN	3.7 ± 4.8	3.4 ± 5.0	3.6 ± 4.5	5.5 ± 6.7	4.2 ± 6.2	2.3 ± 4.0
SUPPORT	4.1 ± 5.8	3.4 ± 4.6	4.0 ± 4.8	3.9 ± 5.7	4.2 ± 6.5	1.0 ± 2.3

percentage of client pairs that exhibit a significant difference based on the logrank test.

Summarizing the results obtained in our previous study on heterogeneity [12], we examine the GBSG2, METABRIC, AIDS, FLCHAIN, and SUPPORT2 datasets. Here, we form a federation of $K = 10$ clients and vary the level of heterogeneity α. Tables 6.1 and 6.2 report the results under quantity-skewed and label-skewed splitting, respectively, averaged over 100 seeded runs and scaled by 100 for easier interpretation.

For quantity-skewed splits, the average heterogeneity score remains below 5% across all settings, indicating that this method primarily tests model robustness to differences in dataset sizes rather than differences in label distributions. Conversely, label-skewed splitting shows heterogeneity scores that vary substantially with α. For all tested datasets, $\mathcal{H}$ is nearly zero when $\alpha = 10^3$, but the score increases as α decreases. In datasets with smaller sample sizes (GBSG2, METABRIC, and AIDS), α must be below 10 to achieve noticeable heterogeneity, while for larger datasets (FLCHAIN and SUPPORT), significant diversity emerges already at $\alpha = 10^2$. As a result, we will focus on label-skewed splits for evaluating the FedSurF algorithm under heterogeneous label distributions.

6.4 Experiments

This section outlines the results obtained in our studies on FedSurF [8, 7], building upon the heterogeneous federated survival datasets from Section 6.3.

6.4.1 Data and Environment

Experiments on FedSurF comprise two types of datasets, common benchmarks and real-world datasets. The set of common benchmarks comprise WHAS500,

Table 6.2: Mean ($\pm$ standard deviation) heterogeneity score (in percentage) $\mathcal{H}(\cdot)$ for label-skewed splits with $K = 10$ across varying α.

Dataset	$\alpha = 10^3$	$\alpha = 10^2$	$\alpha = 10$	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.1$
GBSG2	0.4 ± 1.5	0.6 ± 1.5	2.8 ± 4.3	32.2 ± 11.7	43.7 ± 11.5	63.2 ± 12.9
METABRIC	0.1 ± 0.4	0.2 ± 1.0	10.6 ± 8.4	54.6 ± 13.6	66.5 ± 10.1	76.7 ± 8.7
AIDS	0.3 ± 1.0	1.4 ± 2.7	24.7 ± 12.6	68.1 ± 9.0	74.0 ± 9.1	77.7 ± 8.4
FLCHAIN	0.4 ± 1.2	4.2 ± 5.5	42.8 ± 13.0	78.2 ± 8.8	84.9 ± 5.6	89.3 ± 5.8
SUPPORT	0.1 ± 0.4	14.7 ± 9.7	63.2 ± 10.5	87.0 ± 4.9	88.5 ± 4.7	89.7 ± 6.1

GBSG2, METABRIC, and FLCHAIN. Each of the benchmark datasets has been adapted to survival analysis using a label-skewed splitting algorithm. Real-world datasets, instead, comprise LombardyHF [86] and Fed-TCGA-BRCA [94]. The latter datasets come already equipped with federated client splits, eliminating the need for ad-hoc splitting algorithms.

Experiments see FedSurF compared to several survival baselines: CoxPH, DeepSurv, DeepHit, N-MTLR, and PC-Hazard. For each baseline, we followed the implementation of the `pycox` library. Each neural network comprises two hidden neurons with 32 nodes each, with a 0.1 dropout rate to prevent overfitting. Models are trained using the Adam optimizer with a learning rate of 0.01. While the output of CoxPH and DeepSurv is scalar, the outputs of DeepHit, N-MTLR, and the hazard function of PC-Hazard are discretized between 10 uniformly split time intervals.

Concerning the RSF algorithm in FedSurF, we performed a random search for the best hyperparameter configuration with cross-validation. Among the most important parameters to tune, we identified the number of estimators and the maximum tree depth. Table 6.3 identifies the best RSF parameters for each dataset. The rest of the RSF parameters are the default values from the `scikit-survival` RSF implementation (minimum number of samples for splitting equal to 6 and minimum number of samples per leaf equal to 3). Notably, all clients share the same tree hyperparameters, but, in principle, clients could fine-tune RSF parameters on local hold-out sets.

6.4.2 Federated Scenarios

We evaluate model performance across three scenarios: *local*, *federated*, and *global*.

In the *local* scenario, each client trains a model independently using only its local data. The final performance is computed as the average performance across clients, assessed on their respective hold-out sets. This scenario serves as an empirical lower

Table 6.3: FedSurF hyperparameters for each dataset.

Dataset	Number of Trees	Maximum Tree Depth
WHAS500	400	1
GBSG2	700	1
METABRIC	500	∞
NWTCO	600	1
FLCHAIN	200	∞
LombardyHF	1000	∞
Fed-TCGA-BRCA	1000	∞

bound, providing a baseline for comparison with federated learning approaches. The goal of federated learning is to improve upon this baseline by leveraging data from all clients while preserving privacy.

The second scenario is the *federated* setting, where models are trained using the standard FedAvg algorithm for 150 global rounds, with two local epochs per round. Given the artificially heterogeneous nature of our dataset, we also experimented with FedProx for baseline training. However, pairwise t-tests indicated that only a small fraction ($\sim$5%) of experiments showed a statistically significant difference between FedProx and FedAvg. Consequently, we report results only for FedAvg in our study. Both algorithms were implemented using `flower` [15]. In our experiments, we assume that proportional hazard models, such as CoxPH and DeepSurv, share a common global baseline hazard. For RSF, we employ FedSurF as its tree-based federated counterpart.

Finally, the *global* scenario represents an idealized setting in which all data from the federation is pooled together and treated as if belonging to a single client. This scenario provides an empirical upper bound on model performance, allowing us to quantify the impact of data decentralization. By comparing federated learning results against both the local and global settings, we assess the trade-off between data privacy and predictive performance.

6.5 Results

In this section, we present the results of our FedSurF experiments on the WHAS500, GBSG2, METABRIC, and FLCHAIN datasets, evaluated under the *local*, *federated*, and *global* scenarios. We consider two federations, each composed of 10 clients ($K = 10$), but characterized by different levels of heterogeneity. To

construct the first federation, we apply the label-skewed splitting algorithm from Section 6.3.1 with $\alpha \to \infty$, producing a uniform federation in which each client's data distribution is roughly identical. This setup serves as an idealized benchmark for FedSurF (see Table 6.4).

Conversely, Table 6.5 reports results for a second federation configured with $\alpha = 5$, yielding a significantly more heterogeneous data distribution across the 10 clients. This allows us to assess the FedSurF performance in scenarios where client datasets exhibit statistical differences, and to compare it against neural network-based models under more realistic conditions.

In both tables, four FedSurF variants are presented. The first, labeled simply **FedSurF**, selects trees from each client proportionally to the size of that client's local dataset, assigning equal probability to trees within the same client. The remaining three variants, **+ C-Index**, **+ IBS**, and **+ AUC**, incorporate metric-based tree selection, where the probability of selecting each tree is weighted by its local hold-out performance on the corresponding metric.

6.5.1 Uniform Federations

From Table 6.4, we observe that federated learning consistently surpasses local training. In some rare cases, the federated models even outperform those trained on the combined global dataset. While this fact can be attributed to random variations and stochastic optimization, the key point is that federated models tend to be much closer in performance to the global benchmark than to the local one, corroborating the relevance of federated learning for privacy-preserved distributed data.

Regarding individual models, DeepSurv achieves excellent performance, attaining the highest C-Index and IBS on GBSG2 and METABRIC. Nonetheless, FedSurF remains competitive, especially in WHAS500, where each FedSurF variant attains higher C-Index than any other model. In line with its nonparametric nature, FedSurF excels in discrimination metrics (C-Index and Cumulative AUC) but is comparatively weaker in calibration (IBS).

Overall, no single approach definitively outperforms all others. However, FedSurF requires only $\mathcal{O}(1)$ communication, as all client trees are exchanged in a single federated round. In contrast, the other models rely on multiple rounds of FedAvg to converge.

6.5.2 Heterogeneous Federations

In the more heterogeneous federation with $\alpha = 5$ (Table 6.5), FedSurF outperforms the other models on WHAS500, GBSG2, and FLCHAIN based on both C-Index and Cumulative AUC. This finding is particularly relevant since heterogeneous federations better reflect the statistical variation commonly encountered in real-world datasets drawn from different medical institutions. FedSurF's strong

Table 6.4: Combined metrics for survival models evaluated on simulated uniform federations ($K = 10,\ \alpha \to \infty$). All metrics are reported as mean values over 20 runs and scaled by 100 for readability. The best results are highlighted in **bold** and second best are underlined.

	WHAS500			GBSG2			METABRIC			FLCHAIN		
C-Index (↑)	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**
CoxPH	65.0	74.5	77.8	56.9	61.2	63.7	57.8	62.0	64.6	90.7	91.7	94.2
DeepSurv	67.8	77.4	73.3	58.3	**65.9**	65.9	60.3	**64.5**	64.8	93.3	**94.2**	94.3
DeepHit	66.8	76.8	75.2	56.2	63.5	64.3	57.4	62.4	62.0	92.7	93.7	94.1
N-MTLR	66.8	76.2	74.4	58.0	65.5	63.9	59.6	63.8	64.4	93.5	**94.2**	94.1
PC-Hazard	65.1	74.7	75.5	54.9	60.6	63.6	50.4	58.9	62.9	88.1	93.8	94.2
FedSurF	73.0	79.2	78.6	61.8	65.4	64.2	60.7	63.8	64.0	93.6	93.8	93.9
+ C-Index	–	79.5	–	–	65.3	–	–	63.9	–	–	93.9	–
+ IBS	–	79.3	–	–	65.6	–	–	63.9	–	–	93.9	–
+ AUC	–	**79.8**	–	–	65.4	–	–	64.0	–	–	93.8	–
IBS (↓)	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**
CoxPH	18.5	**16.5**	14.8	19.1	18.0	16.8	17.3	16.4	15.8	8.4	6.7	4.1
DeepSurv	19.4	**16.5**	21.5	19.1	**16.6**	16.8	17.2	**15.8**	16.2	5.7	4.2	4.2
DeepHit	19.6	17.0	16.5	20.1	18.2	17.9	17.7	17.1	16.3	6.4	4.5	4.3
N-MTLR	19.4	17.5	20.4	19.4	16.8	17.9	17.2	15.9	16.1	4.9	**4.1**	4.4
PC-Hazard	22.1	18.5	17.3	21.6	19.0	17.2	22.3	22.8	16.5	7.6	4.6	4.2
FedSurF	18.1	17.4	17.5	18.7	18.0	18.2	17.3	16.2	16.2	4.7	4.5	4.1
+ C-Index	–	17.0	–	–	17.8	–	–	16.1	–	–	4.4	–
+ IBS	–	17.0	–	–	17.9	–	–	16.1	–	–	4.3	–
+ AUC	–	17.0	–	–	17.8	–	–	16.1	–	–	4.4	–
AUC (↑)	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**
CoxPH	66.4	75.0	79.3	61.8	69.5	74.4	60.1	65.8	69.0	92.3	93.2	95.6
DeepSurv	68.0	77.4	73.0	63.0	74.8	73.7	63.2	69.0	68.9	94.7	95.6	95.9
DeepHit	65.7	76.0	74.5	58.9	71.5	70.4	59.8	69.0	69.8	93.9	95.5	95.4
N-MTLR	67.4	75.9	73.4	62.3	73.4	72.1	63.3	**70.7**	72.3	94.7	**95.8**	95.7
PC-Hazard	62.3	72.1	74.5	57.7	67.7	73.0	48.8	60.5	69.5	88.9	94.9	95.3
FedSurF	73.6	79.7	80.0	68.4	74.9	73.7	64.8	69.9	71.0	94.9	95.6	96.1
+ C-Index	–	79.9	–	–	74.9	–	–	70.1	–	–	95.6	–
+ IBS	–	79.8	–	–	**75.1**	–	–	70.3	–	–	95.7	–
+ AUC	–	**80.3**	–	–	75.0	–	–	70.5	–	–	95.6	–

survival performance and minimal communication overhead make it a compelling alternative to other state-of-the-art models, especially when discrimination metrics

Table 6.5: Combined metrics for survival models evaluated on simulated heterogeneous federations ($K = 10,\ \alpha = 5$). All metrics are reported as mean values over 20 runs and scaled by 100 for readability. The best results are highlighted in **bold** and the second best are underlined.

	WHAS500			GBSG2			METABRIC			FLCHAIN		
C-Index (↑)	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**
CoxPH	65.0	74.3	77.8	56.7	61.8	63.9	57.3	61.6	64.5	88.7	89.7	94.2
DeepSurv	68.8	77.3	73.4	58.0	65.1	65.7	60.6	**64.2**	64.8	93.1	89.6	94.3
DeepHit	67.4	77.1	75.2	57.4	63.0	64.2	57.2	61.3	62.0	92.5	93.6	94.1
N-MTLR	68.5	76.6	74.6	58.2	64.8	63.7	60.0	63.8	64.5	93.5	**94.2**	94.1
PC-Hazard	66.0	75.1	75.6	55.1	61.1	63.6	50.3	58.4	62.8	86.7	93.8	94.2
FedSurF	73.0	78.4	78.5	61.9	65.2	64.2	60.7	63.8	64.0	93.6	93.8	93.8
+ C-Index	–	**79.3**	–	–	65.1	–	–	63.8	–	–	93.8	–
+ IBS	–	79.0	–	–	**65.4**	–	–	63.8	–	–	93.8	–
+ AUC	–	79.1	–	–	**65.4**	–	–	63.7	–	–	93.8	–
IBS (↓)	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**
CoxPH	18.6	**16.4**	14.8	19.1	17.8	16.8	17.4	16.4	15.8	8.5	7.7	4.1
DeepSurv	19.1	16.5	21.5	19.1	**16.9**	16.8	17.1	**16.0**	16.2	5.7	6.1	4.2
DeepHit	19.6	16.7	16.5	20.1	18.3	17.9	17.8	17.4	16.3	6.5	4.5	4.3
N-MTLR	19.4	17.4	20.2	19.4	17.1	18.0	17.3	**16.0**	16.1	4.9	**4.2**	4.4
PC-Hazard	22.1	18.7	17.2	21.5	18.7	17.2	22.4	18.8	16.5	8.1	4.5	4.2
FedSurF	18.2	17.5	17.5	18.7	18.0	18.2	17.4	16.2	16.2	4.8	4.6	4.1
+ C-Index	–	17.0	–	–	17.8	–	–	16.2	–	–	4.4	–
+ IBS	–	17.1	–	–	17.9	–	–	16.2	–	–	4.3	–
+ AUC	–	17.1	–	–	17.8	–	–	16.2	–	–	4.4	–
AUC (↑)	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**	**Loc.**	**Fed.**	**Glob.**
CoxPH	65.7	75.8	79.4	61.2	70.3	74.6	59.5	65.4	68.9	90.3	91.5	95.6
DeepSurv	69.1	77.5	73.2	62.9	73.9	73.4	63.8	68.9	68.9	94.3	91.3	95.9
DeepHit	66.6	76.0	74.5	60.7	71.3	70.6	60.3	66.4	69.7	93.7	95.3	95.4
N-MTLR	68.4	76.4	73.8	62.6	72.7	72.0	64.1	**71.0**	72.3	94.6	**95.6**	95.8
PC-Hazard	63.1	72.4	74.7	57.9	68.8	72.9	48.5	61.1	69.3	87.2	94.8	95.3
FedSurF	73.5	78.8	79.9	69.0	74.9	73.6	64.9	70.1	70.9	94.8	95.5	96.1
+ C-Index	–	**79.8**	–	–	74.8	–	–	70.2	–	–	**95.6**	–
+ IBS	–	79.4	–	–	75.1	–	–	70.2	–	–	**95.6**	–
+ AUC	–	**79.8**	–	–	**75.2**	–	–	70.3	–	–	95.5	–

are considered. Conversely, in settings where calibration is crucial, methods such as DeepSurv may be preferred, as evidenced by its IBS scores on GBSG2 and

METABRIC.

6.5.3 Real-World Federations

Table 6.6 presents the performance of FedSurF and the baseline models under *local*, *federated*, and *global* training scenarios on two real-world datasets, Lombardy Heart-Failure and Fed-TCGA-BRCA, as analyzed in [7].

Interestingly, these datasets exhibit opposing patterns: in Lombardy Heart-Failure, FedSurF achieves the best discrimination performance while DeepSurv shows the best calibration; in Fed-TCGA-BRCA, the roles are reversed. This underscores the importance of assessing survival models with both discrimination and calibration metrics, as they do not necessarily agree on which model is best.

Overall, these results are consistent with those from artificially partitioned federations, where federated training surpasses local training and closely approaches the performance of training on the entire global dataset. As before, FedSurF is generally competitive across metrics, even if it does not always achieve the top rank.

A noteworthy result, observed across uniform, heterogeneous, and real-world federations, is that the sampling strategy for tree selection in FedSurF does not drastically affect performance. Although at least one among the three strategies outperforms plain **FedSurF**, the differences across strategies remain small. Consequently, the most practical choice is to employ the C-Index, as its local computation avoids the overhead required by IBS or cumulative AUC (both of which require inverse probability of censoring estimates via a global Kaplan–Meier estimator). Hence, **FedSurF + C-Index** benefits from metric-aware sampling without needing additional communication rounds, making C-Index the most convenient and efficient option.

6.6 Discussion

Results show that FedSurF is competitive with modern federated neural survival methods trained with FedAvg. Thus, exchanging trees rather than iterative gradient updates can be a viable strategy in federated survival analysis. In the experiments on both synthetically crafted and real-world federations, FedSurF provides an edge with respect to local training baselines and approaches the performance of training on all data centrally. Its performance increases in more heterogeneous federations, where clients possess significantly different data distributions, which pertain to more realistic and applied settings.

One of the key benefits of FedSurF is its one-round communication scheme, which avoids the iterative nature of common federated procedures that often require tens or hundreds of global rounds. This design is especially appealing in settings where bandwidth is limited or where multiple sites prefer minimal interaction for policy

or logistical reasons. A second advantage is the versatility of the method. Each institution can build as many trees as its computational resources allow, tune model complexity independently, and selectively transmit only its best-performing trees. In most experimental settings, concordance-based tree sampling offers additional performance gains, due to a finer selection of high-quality trees for the global model.

Table 6.6: Combined metrics for survival models evaluated on real-world federations. All metrics are reported as mean values over 20 runs and scaled by 100 for readability. The best results are highlighted in **bold** and the second best are underlined.

	Lombardy Heart Failure			Fed-TCGA-BRCA		
C-Index (↑)	**Local**	**Federated**	**Global**	**Local**	**Federated**	**Global**
CoxPH	58.4 ± 1.4	69.2 ± 2.4	71.4 ± 0.4	60.3 ± 5.5	75.4 ± 4.6	77.0 ± 0.9
DeepSurv	59.9 ± 1.9	71.7 ± 0.9	70.6 ± 0.8	61.8 ± 4.4	**78.9 ± 2.1**	72.3 ± 2.5
DeepHit	57.6 ± 1.4	69.1 ± 2.0	71.4 ± 0.9	59.3 ± 4.2	76.4 ± 5.7	79.9 ± 2.1
N-MTLR	57.7 ± 1.9	70.9 ± 1.2	69.6 ± 0.8	60.6 ± 4.5	77.7 ± 3.4	75.0 ± 3.2
PC-Hazard	50.8 ± 1.2	69.2 ± 1.3	71.4 ± 0.4	56.9 ± 2.8	68.5 ± 7.4	76.3 ± 2.4
FedSurF	61.7 ± 0.9	73.6 ± 0.9	72.7 ± 0.1	66.7 ± 3.1	76.3 ± 2.3	72.3 ± 0.7
+ C-Index	–	73.5 ± 0.7	–	–	77.2 ± 2.1	–
+ IBS	–	**73.7 ± 0.8**	–	–	76.9 ± 1.8	–
+ AUC	–	73.6 ± 0.5	–	–	77.1 ± 1.9	–
IBS (↓)	**Local**	**Federated**	**Global**	**Local**	**Federated**	**Global**
CoxPH	13.6 ± 0.1	13.0 ± 0.3	12.4 ± 0.1	28.2 ± 1.9	24.8 ± 2.1	24.7 ± 0.4
DeepSurv	14.0 ± 0.3	**12.3 ± 0.3**	13.0 ± 0.2	28.8 ± 1.2	25.7 ± 1.6	32.3 ± 4.3
DeepHit	17.5 ± 0.3	14.3 ± 3.6	12.5 ± 0.1	31.2 ± 0.4	28.2 ± 2.4	27.3 ± 2.2
N-MTLR	15.3 ± 0.3	12.5 ± 0.3	13.2 ± 0.3	29.5 ± 2.1	26.6 ± 4.1	36.2 ± 5.6
PC-Hazard	20.0 ± 0.5	12.8 ± 0.3	12.6 ± 0.1	46.7 ± 1.2	43.7 ± 14.9	26.3 ± 0.8
FedSurF	14.2 ± 0.1	13.3 ± 0.1	12.1 ± 0.0	26.5 ± 0.7	23.2 ± 0.4	24.2 ± 0.3
+ C-Index	–	13.1 ± 0.1	–	–	**22.9 ± 0.3**	–
+ IBS	–	13.2 ± 0.0	–	–	23.1 ± 0.3	–
+ AUC	–	13.1 ± 0.0	–	–	23.0 ± 0.3	–
AUC (↑)	**Local**	**Federated**	**Global**	**Local**	**Federated**	**Global**
CoxPH	58.6 ± 1.4	70.5 ± 2.7	73.6 ± 0.3	62.7 ± 6.1	79.7 ± 3.6	77.6 ± 0.5
DeepSurv	60.5 ± 2.0	73.5 ± 1.1	74.2 ± 1.1	63.4 ± 4.9	**80.7 ± 2.0**	71.0 ± 4.3
DeepHit	57.6 ± 1.2	70.4 ± 2.4	72.4 ± 1.1	59.0 ± 5.0	75.0 ± 5.5	77.2 ± 2.1
N-MTLR	57.4 ± 2.2	71.9 ± 1.1	69.9 ± 1.3	61.3 ± 5.8	77.2 ± 5.0	72.7 ± 4.0
PC-Hazard	51.1 ± 1.2	70.2 ± 1.9	72.9 ± 0.7	55.3 ± 2.3	67.0 ± 8.1	76.0 ± 2.9
FedSurF	60.3 ± 0.9	**74.9 ± 1.3**	73.4 ± 0.2	64.5 ± 3.0	72.9 ± 2.1	72.1 ± 0.5
+ C-Index	–	74.8 ± 1.0	–	–	73.8 ± 1.6	–
+ IBS	–	74.8 ± 1.2	–	–	73.9 ± 1.7	–
+ AUC	–	**74.9 ± 1.0**	–	–	73.7 ± 1.4	–

Chapter 7

Generative Data in Federated Learning and Survival Analysis

Building upon the topics of federated learning and survival analysis, this chapter addresses the final part of this thesis: the use of generative data techniques in both fields. Generative methods offer significant potential benefits for federated learning and survival analysis. In federated learning, several vulnerabilities exist that threaten the three primary security principles—confidentiality, integrity, and availability. Malicious participants can execute diverse attacks, including data poisoning, backdoor, membership inference, and denial of service, among others. To specifically address these security concerns, we developed the Secure Generative Data Exchange (SGDE) protocol [79], presented as part of the initial contributions of this doctoral research. SGDE is tailored for cross-silo federations, which typically consist of a small number of clients characterized by high computational capabilities, full availability, and stringent privacy constraints. Rather than exchanging model parameters or gradients, SGDE relies on the exchange of entire generative models among federation members. In practice, clients generate and share privacy-preserving generative models, enabling others within the federation to download these models and subsequently construct synthetic datasets offline. This approach significantly mitigates potential attack surfaces, as offline datasets simplify the detection of data poisoning attacks, reduce the risk of model inversion and membership inference, and require only a single, asynchronous exchange round, minimizing communication constraints. Our empirical analysis demonstrates that models trained on synthetically generated data achieve performance comparable to models trained directly on real data.

The second contribution, completed towards the end of this research journey, investigates the application of filtering techniques for improving the quality of synthetic samples specifically in survival analysis. In this study, we leveraged multiple generative models to produce an extensive set of synthetic survival data. Subsequently, we employed a loss-aware filtering model, trained on real data, to select

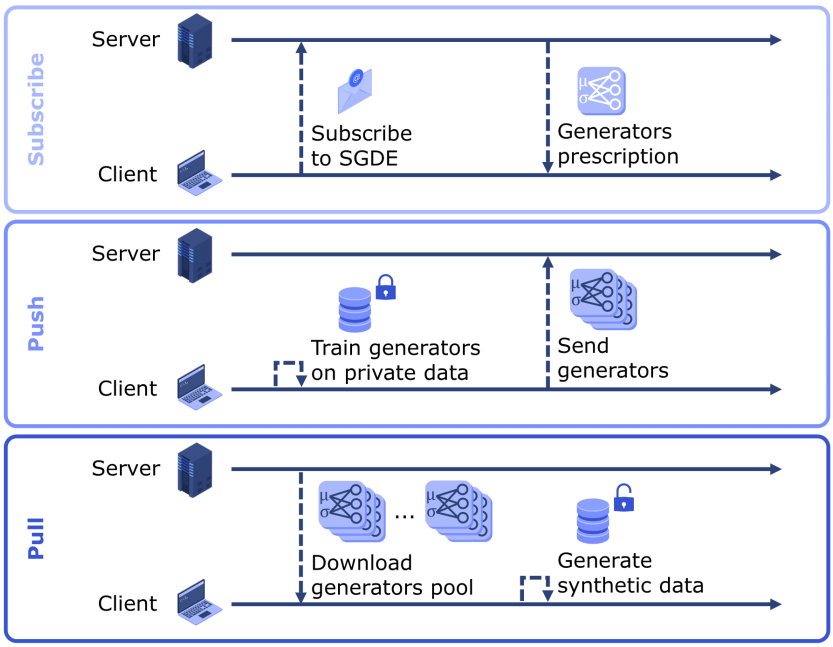

Figure 7.1: Overview of the SGDE protocol, divided in *subscribe*, *push*, and *pull* phases. At the end of the protocol execution, a privacy-preserving synthetic dataset is available to the client for offline usage.

the most informative generated samples. The performance of survival models trained on filtered synthetic datasets was compared to models trained on unfiltered synthetic data and real datasets, with results indicating general performance enhancements, often surpassing even those achieved by models trained solely on real data. This work demonstrates that generative methods can significantly alleviate issues related to data scarcity and privacy concerns in medical contexts by providing privacy-compliant synthetic data, ultimately boosting the efficacy of medical survival analysis models.

7.1 SGDE: Secure Generative Data Exchange

This section presents the Secure Generative Data Exchange (SGDE), a protocol designed to facilitate secure and privacy-preserving data exchange in federated learning scenarios. SGDE leverages generative models to enable clients within a federation to share synthetic data generators, providing a secure alternative to conventional federated learning methods. The deployment of SGDE has been included as a contribution for the European project AI-SPRINT [78, 13]. Specifically, the Dockerized server and client-side modules have been released and tested on a scalable infrastructure within the AI-SPRINT framework.

7.1.1 The SGDE Protocol

As illustrated in Figure 7.1, the protocol consists of three distinct phases: *Subscribe*, *Push*, and *Pull*. In the *Subscribe* phase, clients initiate participation in the federation by registering to a central server. Upon subscription, the central server specifies necessary criteria to clients, including required class labels, generator configurations, and minimum differential privacy parameters (ε, δ) to ensure secure generator training [36].

During the *Push* phase, each client k independently trains a separate generative model $\mathcal{G}_k^y : \mathcal{Z} \to \mathcal{X}$ for every class $y \in \mathcal{Y}$ in their dataset, where $\mathcal{X}$ is the feature space, $\mathcal{Y}$ the output (class) space, and $\mathcal{Z}$ the noise space. These generative models are trained to synthesize privacy-preserving data samples that encapsulate the statistical characteristics of the local data without exposing sensitive details. After completing the training, clients report the achieved differential privacy levels alongside their trained models to the central server. All the other aspects of the training process remain confidential and are not disclosed with the central server.

In the *Pull* phase, the central server aggregates and distributes the generative models submitted by all federation participants. Clients are then able to selectively download and locally utilize these models to generate synthetic datasets. Each generative model supports the creation of an arbitrary number of synthetic samples, allowing clients to customize datasets based on specific needs, such as addressing class imbalance or augmenting their local data resources.

7.1.2 Advantages and Threat Assessment

SGDE provides several advantages compared to traditional federated learning frameworks. Firstly, it offers more flexibility, as clients retain complete control over synthetic dataset creation. They can independently select suitable generators and determine dataset sizes according to their unique requirements. The task-agnostic nature of generated data further enhances adaptability, supporting various machine learning applications without requiring central coordination.

Secondly, SGDE significantly enhances privacy and security. Since private datasets remain local, only the generative models, which are differentially private by design, are shared with federation members. This approach effectively mitigates risks associated with inference attacks and data leakage, thereby preserving confidentiality. Additionally, SGDE promotes fairness by enabling clients to control the distribution and number of synthetic samples produced per class.

Finally, SGDE avoids iterative data exchanges typical in traditional federated learning approaches and relies instead on a single asynchronous exchange of generative models, significantly reducing communication overhead and increasing system robustness.

7.1.3 Experiments

This section outlines the empirical experiments we carried out to validate the performance of SGDE with respect to standard federated learning algorithms.

Data and Environment

The experiments include five tabular datasets from the UCI Machine Learning repository [34]—Titanic, Breast Cancer Wisconsin Diagnostic, Mushroom, Adult, and Wine Quality—and two image datasets, MNIST [71] and Fashion MNIST [116]. For datasets not already equipped with a test set, we applied a 90%–10% stratified random split. Each federation considered is composed of 20 clients with an equal amount of samples drawn uniformly from the original dataset without replacement.

Experiments comprise three settings similar to the ones outlined in Section 6.4.2, *local*, *federated*, and *synthetic*. These settings aim at comparing the performance of machine learning models on single-client data, federated learning, and SGDE-based data.

Concerning the models trained with local or federated learning, we considered a fully connected neural architecture comprising two hidden layers of 64 and 32 neurons and LeakyReLU activation for tabular datasets or a three-layer convolutional architecture with 128, 256, and 512 filters, max pooling, Swish [99] activation, and kernel size 3 followed by a linear layer. Concerning the SGDE generative models, we used β-VAEs [47], with a decoder architecture mimicking the reversed structure of the decoder. To preserve privacy, β-VAEs are trained with a differentially private version of the Adam optimizer [1].

Results

Table 7.1 compares the classification performance of three training settings: local training, federated learning, and SGDE-based synthetic data generation. The table reports both accuracy and F1 scores.

The results show that federated learning and synthetic data yield comparable performance overall. The only exception is the Adult dataset, where local learning slightly outperforms both methods in terms of accuracy. Nonetheless, examining the F1 score reveals that the synthetic data approach achieves higher performance than both local and federated learning in nearly all cases, with the exception of Titanic and MNIST. These findings suggest that SGDE-generated data not only provides notable advantages in terms of privacy and federated computation, as outlined in Section 7.1.2, but also guarantees, on average, a performance improvement over standard federated learning.

Table 7.1: Accuracy and F1 score (in percentage) for local, federated, and synthetic settings across different datasets measured on a hold-out set. The best results are highlighted in **bold**.

	Accuracy			F1 Score		
Dataset	**Local**	**Fed.**	**Synth.**	**Local**	**Fed.**	**Synth.**
Titanic	71.83	**74.81**	74.01	29.70	**71.61**	56.00
Breast Cancer	89.42	91.86	**93.02**	92.25	90.82	**94.78**
Mushrooms	92.56	91.27	**93.49**	91.92	91.20	**93.14**
Adult	**80.87**	76.47	79.00	50.14	47.70	**60.21**
Wine Quality	92.57	90.23	**97.79**	82.42	85.10	**95.70**
MNIST	97.76	**99.08**	98.49	97.71	**99.08**	98.49
Fashion MNIST	85.97	87.94	**88.13**	85.81	87.99	**88.04**

7.2 Synthetic Data Filtering for Survival Analysis

Synthetic data hold significant promise for the future of survival analysis in medical applications. Their benefits are twofold. First, synthetic generators can produce large datasets, enabling survival models to be tested on a scale beyond typical benchmark datasets, which rarely exceed 1–5 thousand samples. This larger dataset size can reveal algorithmic behaviors that only emerge when both non-linearity and sample numerosity are sufficiently high. Second, synthetic data inherently address privacy concerns, making their distribution less restricted than that of real data, even in anonymized form. Together, these advantages make generative pipelines for survival data especially appealing for developing large-scale models able to provide meaningful insights across extensive data repositories.

In this work, we propose a synthetic data filtering pipeline that involves training a generative model and subsequently ranking its generated samples. This ranking step enables users to assemble a synthetic dataset comprising only the highest-quality samples, as measured by a likelihood-based metric. Our results show that survival models trained on these filtered synthetic datasets exhibit enhanced performance compared to those trained on unfiltered real data.

7.2.1 The Filtering Pipeline

Our goal is to build a synthetic dataset

$$\mathcal{D}_S = \{(\mathbf{x}_i, \delta_i, t_i)\}_{i=1}^{N_S}$$

of size N_S, whose samples reflect the population patterns of a real dataset $\mathcal{D}_R$ of size N_R, assuming that any sample $(\mathbf{x}, \delta, t)$ in either $\mathcal{D}_R$ or $\mathcal{D}_S$ comes from the same underlying (and unobserved) data-generation process.

In the first step of our pipeline, we train a generator model $\mathcal{G}$ on $\mathcal{D}_R$, which then produces synthetic samples. Examples of such generators include ARF [114], Bayesian Networks [6], SurvivalCTGAN [117, 92], and TabDDPM [68]. These machine learning models can generate an arbitrary number of synthetic survival samples after being trained on real-world tabular survival data, yielding an initial synthetic dataset $\mathcal{D}_S^0$.

Next, we incorporate a survival model $\mathcal{S}$ as a quality estimator to improve the informative content of $\mathcal{D}_S$. Specifically, we train $\mathcal{S}$ on $\mathcal{D}_R$ and compute its log-likelihood as a quality metric for each synthetic sample. Because not all survival models provide a direct way to compute the likelihood, in the cases where an explicit loss formulation is not available, we approximate it using the survival function $S(t)$ of $\mathcal{S}$ and apply finite differences as

$$\begin{aligned} h(t_i) &= \tfrac{1}{\Delta t}\Big(H(t_i + \Delta t) - H(t_i)\Big) \\ &= \tfrac{1}{\Delta t}\Big(-\log S(t_i + \Delta t) \;+\; \log S(t_i)\Big), \end{aligned}$$

where Δt is a small positive constant. Then, we define the loss of each sample as

$$\mathcal{L}_i \;=\; \delta_i \log\Big(h(t_i)\Big) \;+\; \log S(t_i).$$

From here, we construct the final synthetic dataset by retaining only those samples with loss below a specified threshold $\alpha > 0$:

$$\mathcal{D}_{\mathcal{S}} \;=\; \Big\{\, (\mathbf{x}_i, \delta_i, t_i) \in \mathcal{D}_S^0 \;:\; \mathcal{L}_i < \alpha \Big\}.$$

To obtain exactly N_S samples, one can repeatedly generate samples from $\mathcal{G}$ until reaching the desired size of filtered data. Alternatively, if a fixed dataset size is not required, one may opt to generate a large pool of samples and then choose α so that only samples within the desired loss percentile are retained.

7.2.2 Experiments

In this section, we outline the empirical validation of the proposed filtering pipeline on downstream survival tasks.

Data and Environment

The experiments comprise the following survival datasets: AIDS, FLCHAIN, Framingham, GBSG2, METABRIC, NWTCO, and WHAS500. Each experiment is run in 10-fold cross-validation, retaining at each iteration both a validation

Table 7.2: Average C-Index for each cardinality (*Card.*) considering the best filtering configuration for each dataset. The reported values represent the mean difference between survival models trained on synthetic and real data. The best result is in **bold**, and the second best is underlined.

Card.	ARF	Bayesian Network	TabDDPM	SurvivalCTGAN
1×	-0.11 ± 0.49	-0.07 ± 0.54	-0.48 ± 0.89	-1.48 ± 1.24
2×	0.43 ± 0.24	0.17 ± 0.74	-0.29 ± 0.75	-1.01 ± 1.56
5×	0.80 ± 0.44	0.30 ± 0.39	0.32 ± 0.36	-0.96 ± 1.64
10×	1.03 ± 0.59	$\mathbf{0.43 \pm 0.60}$	0.72 ± 0.39	-0.53 ± 1.61
20×	$\mathbf{1.12 \pm 0.71}$	0.42 ± 0.68	$\mathbf{0.86 \pm 0.49}$	$\mathbf{-0.28 \pm 1.68}$

and a test set, accounting for 20% of the total samples each. Model and pipeline hyperaprameters are tuned on the validation split, while the reported results are averaged over the test set.

As filtering models, we included CoxPH, DeepSurv, FPBoost, and Logistic Hazard. DeepSurv and Logistic Hazard are built as three-layer neural networks with a number of neurons equal to (3,5,3) × (# of features), following the guidelines from the original DeepSurv paper [63]. The same models are used to assess the final downstream survival performance of the synthetic dataset $\mathcal{D}_S$.

Experiments comprise synthetic datasets of different cardinalities. Specifically, we included cardinalities equal to 2×, 5×, 10×, and 20× the size of the original real dataset. Additionally, we tested several thresholding strategies, covering the 25%, 50%, 75%, and 90% of the best generated samples.

Results

Table 7.2 displays the average performance difference in terms of C-Index between the best model and percentile filtering configuration and the real data across different cardinalities. The results indicate that synthetic data surpasses real data performance for all models, except SurvivalCTGAN, starting at a 5× cardinality. Performance improvements generally increase with cardinality, highlighting how important is dataset size in highly parametrized survival models.

Alongside results on cardinality, Table 7.3 provides an overall evaluation of the pipeline, showing the best results achieved with our method for each filtering model, threshold, and cardinality across all datasets. These results are compared with the performance on real data. No single filtering configuration was best for all datasets; each dataset required tuning. However, in every case we tested, the models trained on filtered synthetic data performed as well as, or better than, those trained

Table 7.3: C-Index comparison between Real and synthetic data considering the best filtering model and threshold. The best result is in **bold**, and the second best is underlined.

	CoxPH		**DeepSurv**		**FPBoost**		**Logistic Hazard**	
Dataset	**Real**	**Synth.**	**Real**	**Synth.**	**Real**	**Synth.**	**Real**	**Synth.**
AIDS	73.95	75.37	71.71	75.57	70.84	74.01	72.28	**76.35**
FLCHAIN	93.46	93.53	93.56	93.75	89.86	92.60	93.46	**93.76**
Framingham	71.40	**71.45**	70.50	71.26	69.14	70.08	70.23	71.18
GBSG2	68.13	68.48	66.67	69.67	69.29	**70.07**	66.48	69.89
METABRIC	63.51	64.02	63.45	**65.25**	62.46	64.75	64.51	65.18
NWTCO	70.92	71.18	71.32	**72.54**	65.51	66.73	65.48	71.97
WHAS500	77.33	78.18	73.62	78.44	77.04	**78.51**	72.01	77.81

on the real data. Together, these results showcase the effectiveness of generative datasets, which are capable of encoding and improving patterns from original data and helping downstream models in generalization.

Chapter 8

Conclusion

The studies presented in this thesis address a central challenge in medical data analysis: estimating time-to-event outcomes while adhering to privacy requirements. By unifying concepts from survival analysis and federated learning, we introduced several algorithmic advancements aimed at improving both discrimination and calibration in survival models, as well as ensuring efficient distributed training across multiple institutions.

A first thread of contributions focused on enhancing predictive accuracy in centralized survival tasks. We proposed FPBoost, a gradient boosting algorithm for continuous-time survival analysis that maximizes the full likelihood by summing multiple parametric hazards. Unlike semi-parametric or partially discrete approaches, FPBoost leverages a richer parametric representation to capture non-linear failure patterns. In parallel, we explored interpolation techniques for discrete neural survival models, demonstrating that simple post-processing steps (e.g., linear or spline interpolation) significantly refine coarse-grained predictions and consistently improve calibration metrics over different benchmarks.

A second line of research focused on making survival modeling feasible in decentralized settings characterized by strict data-access regulations. We introduced a federated bagging framework, FedSurF, which adapts random survival forests for distributed training. In contrast to federated averaging, requiring multiple communication rounds, FedSurF reduces communication to a single exchange of locally trained trees, drastically reducing bandwidth demands while offering competitive predictive performance. Remarkably, this single-round strategy showed the best results against conventional gradient-based methods, particularly in high-heterogeneity clinical datasets.

Lastly, we advanced generative techniques for producing synthetic survival data under federated constraints by building high-fidelity and privacy-protected synthetic samples. Beyond preserving confidentiality, these generative approaches hold promise for augmenting limited real-world datasets and for mitigating skewed distributions, thus enabling more robust and fair survival estimates.

Overall, our findings illustrate how modern survival methods can be paired with secure, large-scale training. By jointly considering discrimination, calibration, and privacy, we aim to improve algorithms for user-centered healthcare, wherein risk estimates can be collaboratively learned from diverse patient populations. In high-impact scenarios such as intensive care or oncology, this synergy between advanced survival models and federated infrastructures can guide resource-allocation decisions more effectively, ultimately contributing to better clinical treatments.

8.1 Future Work

Our contributions open several promising lines of research thatweI intend to explore during my post-doctoral work. Below, we outline the short- to mid-term directions.

8.1.1 Federated FPBoost

From an algorithmic standpoint, the first priority is to adapt FPBoost to federated learning, thereby unifying the two core contributions of this thesis. FPBoost was deliberately implemented as an ensemble so that bags of trees can be merged probabilistically into a global model, enabling an asynchronous single-round aggregation inspired by FedSurF. In addition, FPBoost supports boosting with warm-start training, enabling four different federated learning modalities:

1. **Uniform Bagging**: Each client provides a subset of heads—parameterized by η, k, and w—in proportion to its data volume, mirroring the policy in the original FedSurF algorithm [8].

2. **Metric Bagging**: Heads are sampled with probability proportional to their C-Index on a local hold-out set, emulating the metric-aware selection of [7].

3. **Cyclic Boosting on Trees**: A fixed set of heads circulates among clients; each client augments the circulated heads by adding new trees trained on local data, following conventional federated boosting.

4. **Cyclic Boosting on Heads**: Instead of enriching existing heads with trees, every client appends a predetermined number of new heads, allowing the global model to adapt to client-specific patterns while keeping the previous heads frozen.

Among these modes, bagging variants are communication-efficient, converging after a single server–client exchange. Boosting variants, by contrast, require several sequential passes but may achieve higher predictive performance in practice, as the

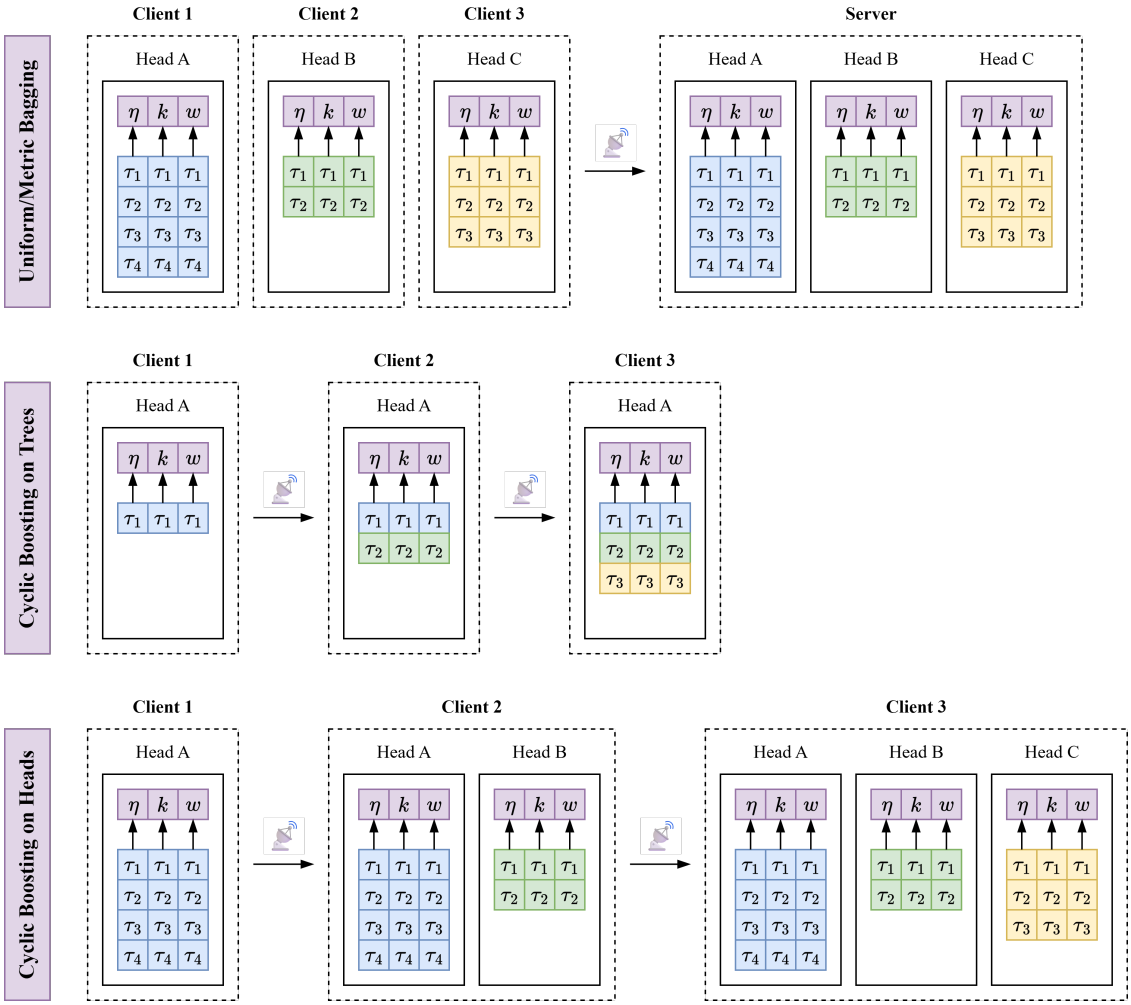

Figure 8.1: Overview of the four strategies for federating FPBoost. The top row shows the two bagging modes, where the server aggregates some or all local trees into a global ensemble. The second and third rows illustrate cyclic boosting on trees and on heads, respectively. In the former, clients sequentially add trees to existing heads, whereas in the latter they append new heads while keeping previous ones fixed.

global model is progressively refined over multiple rounds. Figure 8.1 summarizes the four modalities and the role of each client during training.

Table 8.1 shows preliminary results on the GBSG dataset, comparing the four federated FPBoost variants with established baselines and FedSurF on a federation of five clients. FedSurF remains the top performer in both the IID setting (uniform data split) and the non-IID, label-skewed setting ($\alpha = 1.0$, as in [12]). All FPBoost variants form the second tier of methods, slightly surpassing the remaining baselines.

Table 8.1: Test C-Index (mean ± 95% CI) across 5 seeds for the GBSG dataset with 5 clients. In the **IID** setting, data are split uniformly, while in the **Non-IID** setting data are label-skewed with $\alpha = 1.0$. All results are scaled by a factor of 100 for readability and the best result is highlighted in **bold**.

Model	IID	Non-IID
Cox PH	65.05 ± 1.44	64.16 ± 1.53
DeepSurv	63.16 ± 1.58	62.3 ± 1.9
Logistic Hazard	57.59 ± 1.15	57.19 ± 1.37
DeepHit	64.59 ± 1.38	63.56 ± 1.55
Survival Boosting	65.06 ± 1.39	63.51 ± 1.27
XGBoost	62.2 ± 1.48	61.14 ± 1.36
FedSurF	69.36 ± 0.94	69.32 ± 0.92
FedSurF + C-Index	$\mathbf{69.50 \pm 0.97}$	$\mathbf{69.39 \pm 1.01}$
FPBoost (Uniform Bagging)	64.86 ± 1.33	65.26 ± 0.86
FPBoost (Metric Bagging)	65.61 ± 1.31	65.66 ± 1.43
FPBoost (Boosting Trees)	66.48 ± 0.91	65.96 ± 0.97
FPBoost (Boosting Heads)	65.94 ± 1.14	64.88 ± 1.44

Among them, Metric Bagging and Cyclic Boosting on Trees deliver the best concordance indices—an expected outcome for Metric Bagging, which mirrors FedSurF's metric-aware selection, and a plausible one for Tree Boosting, whose refinement of existing heads could be less noisy than injecting client-specific heads. These findings are preliminary and will require further experimentation before definitive conclusions can be drawn.

8.1.2 Generative and Multimodal Survival Data

Beyond FPBoost and federated learning, the expressive power of neural networks promotes the integration of multimodal information—clinical variables, genomic profiles, and medical images—into survival models. Doing so will demand sophisticated embedding and alignment techniques capable of handling heterogeneous, high-dimensional, censored, and partially missing inputs. Coupling such multimodal fusion with contrastive losses may reveal latent patient subgroups and enable weakly supervised risk estimation.

Finally, deeper investigation of generative pipelines could alleviate data scarcity and distribution shifts, particularly in multi-ethnic and highly imbalanced cohorts.

However, generating realistic yet privacy-preserving survival data is challenging: the distribution of censored observations must be preserved, as well as all the key statistical properties of original data. Taken together, these directions highlight a future in which flexible, privacy-aware, and high-performing survival analysis algorithms will drastically improve the next generations of medical decision-support systems.

8.2 Engineering Barriers of Federated Survival Analysis

Despite the growing research interest in survival analysis and federated learning, the path from a proof-of-concept notebook to a production-grade hospital service is filled with practical issues that go well beyond algorithmic performance. Recent surveys of federated healthcare projects consistently emphasize that communication bottlenecks, data heterogeneity, security threats, immature tooling and regulatory ambiguity each threaten large-scale deployment and, when combined, create a last-mile gap that many prototypes never cross [74].

A first obstacle is the sheer volume of network traffic entailed by iterative gradient-based protocols such as FedAvg [75]. With neural models, each global epoch triggers a bidirectional exchange of weight tensors whose size grows linearly with model dimension, leading to hundreds of megabytes per round for modern deep nets. On top of that, straggler nodes further prolong synchronous training because aggregation must wait for the slowest client update, amplifying latency in multi-institution cohorts. In this work, we particularly focused on these aspects by developing single-round asynchronous federated algorithms—FedSurF and SGDE—converging in $\mathcal{O}(1)$ steps. These protocols ignore model complexity and do not require synchronous updates, turning communication from a blocking factor into a negligible constant.

Equally impactful is the heterogeneity of electronic health-records [75]. Diagnostic values, data granularity and censoring are all contributing factors for heterogeneity across sites. Thus, features that are essential in one silo may be entirely absent in another, producing the non-IID distributions known to slow or destabilize federated optimization [76]. Our work specifically focused on this aspect by testing federated algorithms, such as FedSurF, both in homogeneous and heterogeneous scenarios, empirically showcasing a stronger performance of our solutions in heterogeneous federations.

Lastly, repeated gradient exchange exposes models to membership-inference and model-inversion attacks that can reveal sensitive patient information [77], while denial-of-service attacks threaten availability. By limiting the number of communication rounds, our algorithms mitigate these issues, reducing the potential attack surface over the network that can be exploited by malicious attackers. In

particular, with both FedSurF and SGDE, gradients are not exchanged at all, making gradient-based federated attacks completely ineffective.

Taken together, these observations show that substantial engineering and governance issues persist whenever complex, multi-node algorithms meet clinical practice. However, our work shows that concrete solutions are possible, from lower-bandwidth protocols, to local-first optimization and heterogeneity-aware aggregation. By embedding such ideas into existing and new survival models, this thesis offers an actionable starting blueprint for hospitals that seek to move from local survival modeling to privacy-preserving multi-node applications.

8.3 Ethical Implications of Survival Analysis

While our research has primarily focused on algorithmic improvements aimed at maximizing scores defined on mathematical objects, applying these advancements to healthcare, risk assessment, and data privacy introduces important ethical considerations. AI-driven survival models, such as those presented in this thesis, have the potential to enhance patient risk stratification and optimize treatment allocation. While beyond the scope of this work, their deployment in real-world healthcare settings necessitates addressing several ethical challenges.

When clinicians rely heavily on model predictions, concerns about fairness can arise—particularly if algorithmic outputs outweigh or influence professional judgment for cost or efficiency reasons. Bias issues are equally pressing, since survival models may inadvertently replicate historical data imbalances that undervalue certain populations, thereby intensifying health disparities [93]. In addition, model inaccuracies, such as false positives and negatives, can lead to misclassifications that deny care to those who need it or expose others to unnecessary interventions. Thus, over-reliance on automated predictions can amplify these risks. Many recent survival techniques, including deep learning and ensemble methods, also introduce transparency challenges, since their inner workings often remain opaque and difficult to interpret with a white-box approach. This opacity can undermine trust and impede causality-aware decision-making, prompting a tradeoff between raw performance and explainability [4]. In this context, to safeguard accountability, we advocate for human experts to be actively involved from model development to clinical deployment, thereby ensuring final treatment decisions reflect both data-driven insights and medical expertise.

Privacy is another key concern, as survival analysis demands large amounts of sensitive patient data; federated learning mitigates some risks by keeping data distributed, yet still requires advanced security measures like differential privacy and homomorphic encryption [3]. Taken together, these issues highlight that while modern survival models hold considerable promise, their ethical use depends on careful management of fairness, transparency, and privacy, with clear strategies to

prevent discrimination, maintain human control, and preserve patient trust.

Appendix A

Technical Appendix

A.1 Survival Datasets

In the following, we outline the core characteristics of each survival dataset considered in this thesis. Summary statistics for all the available datasets and Kaplan–Meier curves are provided in Table A.1 and Figure A.1, respectively.

AIDS. The AIDS [48] dataset comes from a medical trial comparing three-drug versus two-drug regimens in HIV-infected patients. The primary endpoint is the time to an AIDS-defining event or death, and the study reached a predefined statistical significance early, resulting in a high rate of censoring. This dataset is available in the `scikit-survival` library [95].

Breast Cancer. The Breast Cancer [31] dataset is based on a study designed to validate a 76-gene prognostic signature for predicting the risk of distant metastases in breast cancer. It includes gene expression profiles from 198 patients, focusing on identifying and stratifying high-risk individuals. This dataset is available in the `scikit-survival` library [95].

FLCHAIN. The FLCHAIN [32] dataset investigates the link between serum free light chains and mortality in a cohort of 7874 subjects. The primary endpoint is overall survival, with 2169 of the participants (27.5%) experiencing the event during follow-up. This dataset is available in the `scikit-survival` library [95].

Framingham. The Framingham [60] dataset is a subset of the larger Framingham Heart Study, originally initiated in 1948 to examine cardiovascular disease risk factors. The subset employed here contains 4699 patients and 9 covariates, capturing the influence of lifestyle and clinical measurements on cardiovascular outcomes. This dataset is available in the `SurvSet` library [33].

Table A.1: Summary statistics for the twelve publicly-available survival datasets used in this thesis. *Censoring* reports the percentage of right-censored observations. *Timeline* gives the minimum and maximum event (or censoring) times. The last two columns report the number of categorical and numerical features.

Name	Samples	Censoring	Timeline	Categorical	Numerical
AIDS	1151	91.66%	[1, 364]	8	3
Breast Cancer	198	74.24%	[125, 9108]	2	78
FLCHAIN	7874	72.45%	[0, 5215]	5	4
Framingham	4699	68.65%	[18, 11688]	2	6
GBSG2	686	56.41%	[8, 2659]	3	5
Lombardy HF	895	64.92%	[66, 2554]	1	30
METABRIC	1904	42.07%	[0, 355]	0	9
MIMIC	28161	10.20%	[0, 206]	5	24
NWTCO	4028	85.82%	[4, 6209]	5	3
SUPPORT	9105	31.89%	[3, 2029]	11	25
TCGA-BRCA	1088	86.12%	[0, 8605]	0	41
Veterans	137	6.57%	[1, 999]	3	3
WHAS	500	57.00%	[1, 2358]	8	6

GBSG2. The German Breast Cancer Study Group (GBSG2) [104] dataset is drawn from a randomized study in Germany, assessing the effect of hormone therapy on breast cancer recurrence. Key variables include age, menopausal status, tumor size, and node status, aimed at predicting relapse times. This dataset is available in the `scikit-survival` library [95].

Lombardy Heart Failure. The Lombardy Heart Failure [86] administrative dataset comes from the HFData project (RF-2009-1483329), which examined heart failure hospitalizations in Lombardy from 2000 to 2012. The initial data comprised 339,690 records with 48 features, but the subset used here was restricted to 22,418 samples by requiring at least 5 years of follow-up and removing post-2007 admissions. Pharmacological-prescription features were excluded, yielding 32 covariates. Finally, data were split by medical structure and filtered to retain 23 centers, producing 895 samples in total. Additional details are provided in [7].

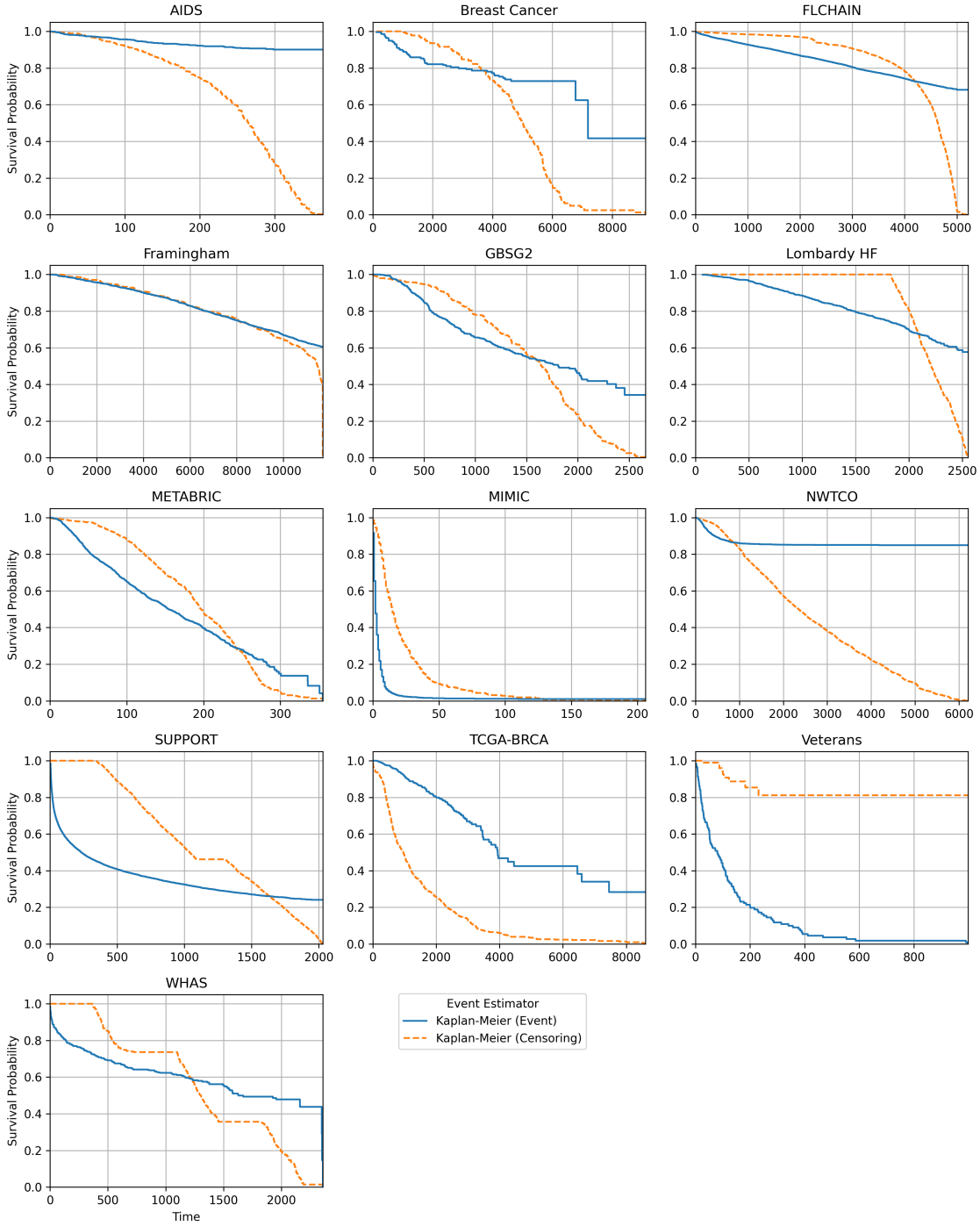

Figure A.1: Kaplan–Meier curves of the available datasets. Event estimators are reported in blue, while censoring estimators, i.e., estimators in which the event of interest is censoring, are dashed in orange.

METABRIC. The Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) [63] dataset collects clinical and molecular data to classify breast tumors and guide personalized therapy. It includes patients from Canada and the United Kingdom, with a combination of genomic and phenotypic variables. This dataset is available in the `DeepSurv` library [63].

MIMIC. The MIMIC-Extract [112] dataset converts raw MIMIC-III v1.4 electronic health records into an analysis-ready cohort of 34472 adult intensive-care unit stays, discretizing vital-sign and laboratory measurements into hourly time series. During preprocessing it harmonizes units, detects outliers, and aggregates semantically equivalent concepts to curb missingness and dampen concept drift. MIMIC-Extract can be accessed via its open-source repository on GitHub[1]. In our implementation, we further refine the data by retaining the top 5% of admission diagnoses keywords based on TF–IDF representation, filtering out rows with NaN values, and setting patient death as the event time. This results in a dataset of 28161 instances, each described by 5 numerical and 24 categorical variables.

NWTCO. The National Wilms Tumor Study (NWTCO) [19] dataset contains 4028 observations on 8 covariates related to Wilms tumor, a pediatric kidney malignancy. Time to relapse serves as the primary outcome, and variables include histology status, disease stage, and other clinical factors. This dataset is available in the `SurvSet` library [33].

SUPPORT. The Study to Understand Prognoses Preferences Outcomes and Risks of Treatment (SUPPORT or SUPPORT2) [66] investigates critically ill hospitalized patients, following each for six months. The version employed in this thesis includes 35 features, spanning both administrative and clinical measurements. This dataset is available in the `SurvSet` library [33].

TCGA-BRCA. The (Fed-)TCGA-BRCA survival dataset [94] arises from The Cancer Genome Atlas (TCGA), focusing on breast invasive carcinoma. It comprises 1088 patients with 38 binary features and is split among six geographical regions (Northeast, South, West, Midwest, Europe, and Canada). The *Fed-* prefix indicates the original federated partition of data or its aggregated use. This dataset is available in the `flamby` library [94].

Veterans. The Veterans [58] dataset originates from a trial examining two chemotherapy regimens in lung cancer patients. Its relatively small sample size has

[1] https://github.com/MLforHealth/MIMIC_Extract

made it a standard benchmark for basic survival analysis methods. This dataset is available in the `scikit-survival` library [95].

WHAS. The Worcester Heart Attack Study (WHAS)[48] follows 1638 patients hospitalized for myocardial infarction between 1997 and 2001. Variables include biometric characteristics and time-to-event information. While the original DeepSurv[63] paper used all 1638 patients, a reduced version (WHAS500) containing approximately one-third of the samples is frequently utilized in the `scikit-survival` [95] library. The complete version of this dataset is available in the `DeepSurv` library [63], while the compressed version in the `scikit-survival` library [95].

A.2 Repository Collection

In the following, we report the open source code related to the studies outlined in the thesis.

FPBoost. This code is related to the content of Chapter 3.
https://github.com/archettialberto/fpboost

Interpolation. This code is related to the content of Chapter 4.
https://github.com/archettialberto/interpolation_for_deep_survival_analysis

Survival Dataset Heterogeneity. This code is related to the content of Section 6.3.1.
https://github.com/archettialberto/federated_survival_datasets

Federated Survival Forests. This code is related to the content of Chapter 6.
https://github.com/archettialberto/federated_survival_forests

SGDE. This code is related to the content of Section 7.1.
https://github.com/archettialberto/SGDE

Appendix B

A Recap of My Ph.D. Journey

Beyond the research contributions detailed in this manuscript, my time at Politecnico di Milano has been enriched by several additional activities that have shaped both my academic and professional journey. The following sections outline these experiences. As I reflect on nearly four years of work, I recognize this as a transformative period—one that is nearing its conclusion but will remain a defining chapter of personal and professional growth.

B.1 Additional Research Studies

Emerging Molecular Communication (2024–ongoing). As part of the COMMENCE project proposal (in preparation for submission to NSF), in collaboration with the University of Nebraska-Lincoln and the University of Florence, we studied emerging molecular communication patterns within unicellular agents such as bacteria. The primary goal was to explore evolutionary pathways that yield emergent behaviors exploitable in engineering contexts. Our preliminary investigations focused on graph neural networks whose channels mimic diffusion-based communication, akin to molecular-scale media. Alongside this work, we developed an image denoising architecture based on cellular automata [88].

Federated Learning for Genome-wide Association Studies (2023–ongoing). In this research, we aim to develop a federated learning framework for training survival models on SNP data derived from real patient DNA in collaboration with the Human Technopole. The project emphasizes secure and privacy-preserving approaches to leverage genomic information across multiple institutions without centralizing sensitive data.

Anomaly Detection on Strain Gauges (2022–ongoing). This study centers on anomaly detection and survival analysis for strain gauges installed on gas pipelines

from SNAM Rete Gas. By examining their spectral profiles, we seek to characterize defective sensors and predict potential failures to ensure timely maintenance and reduced downtime.

Companies Knowledge Graph (2021–2022). In collaboration with Webratio, we contributed to the development of a knowledge graph that semantically links information about worldwide companies. Our focus was on designing a scalable architecture and schema to facilitate efficient, meaningful connections across various data sources.

B.2 Attended Conferences

AIxIA 2022 (Workshop on Machine Learning and Data Mining). I participated as a presenter to showcase *Federated Survival Analysis* at the Italian Association for Artificial Intelligence in Udine (28–29 November 2022).

International Joint Conference on Neural Networks (IJCNN 2023). At the IJCNN in Broadbeach (18–23 June 2023), I presented *Federated Survival Forests*, contributing to the session on distributed and privacy-preserving machine learning methods.

AIxIA 2023 (Workshop on Artificial Intelligence For Healthcare). I presented *Deep Survival Analysis for Healthcare: An Empirical Study on Post-Processing Techniques* at the Italian Association for Artificial Intelligence in Rome (6–9 November 2023).

British Machine Vision Conference (BMVC 2023). I attended and presented *Discriminative Adversarial Privacy: Balancing Accuracy and Membership Privacy in Neural Networks* in Aberdeen (20–24 November 2023), organized by the British Machine Vision Association.

B.3 Teaching Activities

Over the past three and a half years, I have been involved in several teaching roles at Politecnico di Milano. Below is a summary of my primary activities.

Software Engineering (Bachelor). From 2021 to 2023, I served as a lab assistant for a course led by Prof. Gianpaolo Cugola. Students were required to design and implement a Java board game under the MVC framework, working in groups over

six months. Each year, I ran 32 hours of labs, providing guidance on developing a stable, extensible codebase and supervising code reviews.

Artificial Neural Networks and Deep Learning (Master). In 2024–2025, I worked alongside Prof. Matteo Matteucci and Prof. Giacomo Boracchi on one of the larger Master's courses at Politecnico. My contributions included teaching 20 hours of coding sessions and organizing two leaderboard-based challenges for over 700 students.

Privacy-Preserving Learning (Ph.D. Level). In 2022, I delivered a 1.5-hour guest lecture on federated learning as part of the Advanced Deep Learning course from Prof. Matteo Matteucci and Prof. Giacomo Boracchi. The lecture introduced key algorithms, highlighting benefits and drawbacks of distributing training across multiple sites to maintain user privacy.

TechCamp: Python (High School Summer Program). In 2022, 2023, and 2025 I taught afternoon exercises (30 hours over two weeks in 2022 and 15 hours over a single week in the following years) for a Python course organized by Fondazione Politecnico di Milano and Professors Francesco Bruschi and Vincenzo Rana. The focus was on basic Python programming and game development with `pygame` for motivated high school students.

TechCamp: AI Bootcamp (High School Summer Program). In 2024 and 2025, I participated in a AI-focused course by Fondazione Politecnico under Prof. Matteo Matteucci and Prof. Giacomo Boracchi, teaching 15 hours of hands-on lessons per year. Students learned basic machine learning concepts and developed a webcam-based card classifier in Google Colab during the final day.

Bibliography

[1] Martin Abadi et al. "Deep Learning with Differential Privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 308–318. ISBN: 9781450341394. DOI: 10.1145/2976749.2978318. URL: https://doi.org/10.1145/2976749.2978318.

[2] J.P. Albrecht. "How the GDPR Will Change the World". In: *European Data Protection Law Review* 2.3 (2016), pp. 287–289. ISSN: 23642831, 2364284X. DOI: 10.21552/EDPL/2016/3/4.

[3] Mansoor Ali et al. "Federated learning for privacy preservation in smart healthcare systems: A comprehensive survey". In: *IEEE journal of biomedical and health informatics* 27.2 (2022), pp. 778–789.

[4] Julia Amann et al. "Explainability for artificial intelligence in healthcare: a multidisciplinary perspective". In: *BMC medical informatics and decision making* 20 (2020), pp. 1–9.

[5] Mathieu Andreux et al. "Federated survival analysis with discrete-time Cox models". In: *arXiv preprint arXiv:2006.08997* (2020).

[6] Ankur Ankan and Johannes Textor. "pgmpy: A Python Toolkit for Bayesian Networks". In: *Journal of Machine Learning Research* 25.265 (2024), pp. 1–8.

[7] Alberto Archetti, Francesca Ieva, and Matteo Matteucci. "Scaling survival analysis in healthcare with federated survival forests: A comparative study on heart failure and breast cancer genomics". In: *Future Generation Computer Systems* 149 (2023), pp. 343–358. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2023.07.036. URL: https://www.sciencedirect.com/science/article/pii/S0167739X23002935.

[8] Alberto Archetti and Matteo Matteucci. "Federated Survival Forests". In: *2023 International Joint Conference on Neural Networks (IJCNN)*. 2023, pp. 1–9. DOI: 10.1109/IJCNN54540.2023.10190999.

[9] Alberto Archetti, Francesco Stranieri, and Matteo Matteucci. "Bridging the gap: improve neural survival models with interpolation techniques". In: *Progress in Artificial Intelligence* (2024), pp. 1–16. DOI: https://doi.org/10.1007/s13748-024-00343-y.

[10] Alberto Archetti, Francesco Stranieri, and Matteo Matteucci. "Deep Survival Analysis for Healthcare: An Empirical Study on Post-Processing Techniques". In: *Proceedings of the 2nd AIxIA Workshop on Artificial Intelligence For Healthcare (HC@AIxIA 2023)*. Ed. by Francesco Calimeri, Mauro Dragoni, and Fabio Stella. Vol. 3578. CEUR Workshop Proceedings. Rome, Italy: CEUR-WS.org, Nov. 8, 2023, pp. 99–121. URL: http://ceur-ws.org/Vol-3578/.

[11] Alberto Archetti et al. "FPBoost: Fully Parametric Gradient Boosting for Survival Analysis". In: *arXiv preprint arXiv:2409.13363* (2024).

[12] Alberto Archetti et al. "Heterogeneous datasets for federated survival analysis simulation". In: *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*. 2023, pp. 173–180.

[13] Fatemeh Baghdadi et al. "Harnessing the computing continuum across personalized healthcare, maintenance and inspection, and Farming 4.0". In: *arXiv preprint arXiv:2403.14650* (2024).

[14] Soumya Banerjee et al. "dsSurvival: Privacy preserving survival models for federated individual patient meta-analysis in DataSHIELD". en. In: *BMC Research Notes* 15.1 (Dec. 2022), p. 197. ISSN: 1756-0500. DOI: 10.1186/s13104-022-06085-1. (Visited on 04/14/2023).

[15] Daniel J Beutel et al. "Flower: A Friendly Federated Learning Research Framework". In: *arXiv preprint arXiv:2007.14390* (2020).

[16] J Martin Bland and Douglas G Altman. "The logrank test". In: *Bmj* 328.7447 (2004), p. 1073.

[17] L. Breiman et al. *Classification and Regression Trees*. Taylor & Francis, 1984. ISBN: 9780412048418. DOI: https://doi.org/10.1201/9781315139470.

[18] N. Breslow. "Covariance Analysis of Censored Survival Data". In: *Biometrics* 30.1 (1974), pp. 89–99. ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/2529620 (visited on 08/14/2024).

[19] N. E. Breslow and N. Chatterjee. "Design and Analysis of Two-Phase Studies with Binary Outcome Applied to Wilms Tumour Prognosis". en. In: *Journal of the Royal Statistical Society Series C: Applied Statistics* 48.4 (Dec. 1999), pp. 457–468. ISSN: 0035-9254, 1467-9876. DOI: 10.1111/1467-9876.00165. (Visited on 04/29/2023).

[20] George Chen. "Nearest neighbor and kernel survival analysis: Nonasymptotic error bounds and strong consistency rates". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1001–1010.

[21] George H Chen et al. "An Introduction to Deep Survival Analysis Models for Predicting Time-to-Event Outcomes". In: *Foundations and Trends in Machine Learning* 17.6 (2024), pp. 921–1100.

[22] George H Chen. "Deep kernel survival analysis and subject-specific survival time prediction intervals". In: *Machine learning for healthcare conference*. PMLR. 2020, pp. 537–565.

[23] George H Chen. "Survival kernets: Scalable and interpretable deep kernel survival analysis with an accuracy guarantee". In: *Journal of Machine Learning Research* 25.40 (2024), pp. 1–78.

[24] Richard J. Chen et al. "Multimodal Co-Attention Transformer for Survival Prediction in Gigapixel Whole Slide Images". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 3995–4005. DOI: 10.1109/ICCV48922.2021.00398.

[25] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785.

[26] Alexander Chowdhury et al. "A review of medical federated learning: Applications in oncology and cancer research". In: *International MICCAI Brainlesion Workshop*. Springer. 2021, pp. 3–24.

[27] D. R. Cox. "Regression Models and Life-Tables". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 34.2 (1972), pp. 187–220. ISSN: 00359246. URL: http://www.jstor.org/stable/2985181 (visited on 01/11/2023).

[28] Matthew G Crowson et al. "A systematic review of federated learning applications for biomedical data". In: *PLOS Digital Health* 1.5 (2022).

[29] Dorota M Dabrowska. "Uniform consistency of the kernel conditional Kaplan-Meier estimate". In: *The Annals of Statistics* (1989), pp. 1157–1167.

[30] Wenrui Dai et al. "VERTICOX: Vertically Distributed Cox Proportional Hazards Model Using the Alternating Direction Method of Multipliers". In: *IEEE Transactions on Knowledge and Data Engineering* 34.2 (Feb. 2022), pp. 996–1010. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2020.2989301.

[31] Christine Desmedt et al. "Strong Time Dependence of the 76-Gene Prognostic Signature for Node-Negative Breast Cancer Patients in the TRANSBIG Multicenter Independent Validation Series". In: *Clinical Cancer Research* 13.11 (June 2007), pp. 3207–3214. ISSN: 1078-0432. DOI: 10.1158/1078-0432.CCR-06-2765. eprint: https://aacrjournals.org/clincancerres/article-pdf/13/11/3207/1969255/3207.pdf.

[32] Angela Dispenzieri et al. "Use of nonclonal serum immunoglobulin free light chains to predict overall survival in the general population". In: *Mayo Clinic Proceedings*. Vol. 87. Elsevier. 2012, pp. 517–523.

[33] Erik Drysdale. "SurvSet: An open-source time-to-event dataset repository". In: *arXiv preprint arXiv:2203.03094* (2022).

[34] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017.

[35] Rui Duan et al. "Learning from local to global: An efficient distributed algorithm for modeling time-to-event data". In: *Journal of the American Medical Informatics Association* 27.7 (July 2020), pp. 1028–1036. ISSN: 1527-974X. DOI: 10.1093/jamia/ocaa044. URL: https://doi.org/10.1093/jamia/ocaa044.

[36] Cynthia Dwork. "Differential privacy: A survey of results". In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19.

[37] The Economist. "The World's Most Valuable Resource is No Longer Oil, but Data". In: *The Economist* (2017). URL: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data.

[38] Jane Elith, John R Leathwick, and Trevor Hastie. "A working guide to boosted regression trees". In: *Journal of animal ecology* 77.4 (2008), pp. 802–813.

[39] Carlos Fernández-Loría, Foster Provost, and Xintian Han. "Explaining data-driven decisions made by AI systems: the counterfactual approach". In: *arXiv preprint arXiv:2001.07417* (2020).

[40] Stephane Fotso. "Deep neural networks for survival analysis based on a multi-task framework". In: *arXiv preprint arXiv:1801.05512* (2018).

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

[42] David Froelicher et al. "Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption". en. In: *Nature Communications* 12.1 (Oct. 2021), p. 5910. ISSN: 2041-1723. DOI: 10.1038/s41467-021-25972-y.

[43] Marzyeh Ghassemi, Luke Oakden-Rayner, and Andrew L Beam. "The false hope of current approaches to explainable artificial intelligence in health care". In: *The Lancet Digital Health* 3.11 (2021), e745–e750.

[44] Erika Graf et al. "Assessment and comparison of prognostic classification schemes for survival data". In: *Statistics in medicine* 18.17-18 (1999), pp. 2529–2545.

[45] László Györfi et al. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

[46] Christian Rønn Hansen et al. "Larynx cancer survival model developed through open-source federated learning". en. In: *Radiotherapy and Oncology* 176 (Nov. 2022), pp. 179–186. ISSN: 01678140. DOI: 10.1016/j.radonc.2022.09.023.

[47] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *2017 International Conference on Learning Representations (ICLR)*. 2017.

[48] David W. Hosmer, Stanley Lemeshow, and Susanne May. *Applied Survival Analysis: Regression Modeling of Time-to-Event Data*. Wiley Series in Probability and Statistics. Hoboken, NJ, USA: John Wiley & Sons, Inc., Feb. 2008. DOI: 10.1002/9780470258019. URL: http://doi.wiley.com/10.1002/9780470258019 (visited on 04/29/2023).

[49] Torsten Hothorn et al. "Bagging survival trees". In: *Statistics in medicine* 23.1 (2004), pp. 77–91.

[50] Hongsheng Hu et al. "Membership inference attacks on machine learning: A survey". In: *ACM Computing Surveys (CSUR)* 54.11s (2022), pp. 1–37.

[51] Woonghee Tim Huh et al. "Adaptive data-driven inventory control with censored demand based on Kaplan-Meier estimator". In: *Operations Research* 59.4 (2011), pp. 929–941.

[52] Hung Hung and Chin-Tsang Chiang. "Estimation methods for time-dependent AUC models with survival data". In: *Canadian Journal of Statistics* 38.1 (2010), pp. 8–26.

[53] Akira Imakura et al. "DC-COX: Data collaboration Cox proportional hazards model for privacy-preserving survival analysis on multiple parties". en. In: *Journal of Biomedical Informatics* 137 (Jan. 2023), p. 104264. ISSN: 15320464. DOI: 10.1016/j.jbi.2022.104264. URL: https://linkinghub.elsevier.com/retrieve/pii/S1532046422002696 (visited on 04/14/2023).

[54] Hemant Ishwaran et al. "Random survival forests". In: *The Annals of Applied Statistics* 2.3 (2008), pp. 841–860. DOI: 10.1214/08-AOAS169. URL: https://doi.org/10.1214/08-AOAS169.

[55] Harold Jeffreys and Bertha Jeffreys. *Methods of mathematical physics*. Cambridge university press, 1999.

[56] Songhan Jiang et al. "Multimodal Cross-Task Interaction for Survival Analysis in Whole Slide Pathological Images". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2024, pp. 329–339.

[57] Peter Kairouz et al. "Advances and open problems in federated learning". In: *Foundations and Trends in Machine Learning* 14.1–2 (2021), pp. 1–210.

[58] John D Kalbfleisch and Ross L Prentice. *The statistical analysis of failure time data*. John Wiley & Sons, 2011.

[59] Bart Kamphorst et al. "Accurate training of the Cox proportional hazards model on vertically-partitioned data while preserving privacy". en. In: *BMC Medical Informatics and Decision Making* 22.1 (Dec. 2022), p. 49. ISSN: 1472-6947. DOI: 10.1186/s12911-022-01771-3. URL: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-022-01771-3 (visited on 04/14/2023).

[60] William B Kannel and Daniel L McGee. "Diabetes and cardiovascular disease: the Framingham study". In: *Jama* 241.19 (1979), pp. 2035–2038.

[61] Edward L Kaplan and Paul Meier. "Nonparametric estimation from incomplete observations". In: *Journal of the American statistical association* 53.282 (1958), pp. 457–481.

[62] Sai Praneeth Karimireddy et al. "Scaffold: Stochastic controlled averaging for federated learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5132–5143.

[63] Jared L Katzman et al. "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network". In: *BMC medical research methodology* 18.1 (2018), pp. 1–12.

[64] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017).

[65] John P Klein, Melvin L Moeschberger, et al. *Survival analysis: techniques for censored and truncated data*. Vol. 1230. Springer, 2003.

[66] William Knaus et al. "The SUPPORT Prognostic Model. Objective Estimates of Survival for Seriously Ill Hospitalized Adults. Study to Understand Prognoses and Preferences for Outcomes and Risks of Treatments". In: *Annals of internal medicine* 122 (Mar. 1995), pp. 191–203.

[67] N. Korneichuk and V. Motornyi. "Jackson inequality". In: *Encyclopedia of Mathematics* (2020). URL: https://encyclopediaofmath.org/wiki/Jackson_inequality.

[68] Akim Kotelnikov et al. "Tabddpm: Modelling tabular data with diffusion models". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 17564–17579.

[69] Satyapriya Krishna et al. "The disagreement problem in explainable machine learning: A practitioner's perspective". In: *arXiv preprint arXiv:2202.01602* (2022).

[70] Håvard Kvamme and Ørnulf Borgan. "Continuous and discrete-time survival prediction with neural networks". In: *Lifetime Data Analysis* 27.4 (2021), pp. 710–736.

[71] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[72] Changhee Lee et al. "Deephit: A deep learning approach to survival analysis with competing risks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[73] Kwan-Moon Leung, Robert M Elashoff, and Abdelmonem A Afifi. "Censoring issues in survival analysis". In: *Annual review of public health* 18.1 (1997), pp. 83–104.

[74] Ming Li et al. "From challenges and pitfalls to recommendations and opportunities: Implementing federated learning in healthcare". In: *Medical Image Analysis* (2025), p. 103497.

[75] Tian Li et al. "Federated learning: Challenges, methods, and future directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60.

[76] Tian Li et al. "Federated optimization in heterogeneous networks". In: *Proceedings of Machine Learning and Systems* 2 (2020), pp. 429–450.

[77] Eugenio Lomurno et al. "Discriminative adversarial privacy: Balancing accuracy and membership privacy in neural networks". In: *arXiv preprint arXiv:2306.03054* (2023).

[78] Eugenio Lomurno et al. *SGDE: Secure Generative Data Exchange for Cross-Silo Federated Learning*. Sept. 2022. URL: https://doi.org/10.5281/zenodo.7419483.

[79] Eugenio Lomurno et al. "SGDE: Secure Generative Data Exchange for Cross-Silo Federated Learning". In: *AIPR 2022, International Conference on Artificial Intelligence and Pattern Recognition*. 2022.

[80] Chia-Lun Lu et al. "WebDISCO: a web service for distributed cox model learning without patient-level data sharing". In: *Journal of the American Medical Informatics Association* 22.6 (2015), pp. 1212–1219.

[81] Ming Y. Lu et al. "Federated learning for computational pathology on gigapixel whole slide images". en. In: *Medical Image Analysis* 76 (Feb. 2022), p. 102298. ISSN: 13618415. DOI: 10.1016/j.media.2021.102298. URL: https://linkinghub.elsevier.com/retrieve/pii/S1361841521003431 (visited on 04/14/2023).

[82] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[83] Zezhong Ma et al. "A review of federated learning technology and its research progress in healthcare applications". In: *Applied Intelligence* 55.10 (2025), pp. 1–30.

[84] Tanguy Marchand et al. *SecureFedYJ: a safe feature Gaussianization protocol for Federated Learning*. arXiv:2210.01639 [cs]. Oct. 2022. URL: http://arxiv.org/abs/2210.01639 (visited on 04/14/2023).

[85] Carlotta Masciocchi et al. "Federated Cox Proportional Hazards Model with multicentric privacy-preserving LASSO feature selection for survival analysis from the perspective of personalized medicine". In: *2022 IEEE 35th International Symposium on Computer-Based Medical Systems (CBMS)*. Shenzen, China: IEEE, July 2022, pp. 25–31. ISBN: 978-1-66546-770-4. DOI: 10.1109/CBMS55023.2022.00012. URL: https://ieeexplore.ieee.org/document/9867090/ (visited on 04/14/2023).

[86] Cristina Mazzali et al. "Methodological issues on the use of administrative data in healthcare research: the case of heart failure hospitalizations in Lombardy region, 2000 to 2012". In: *BMC Health Services Research* 16 (2016).

[87] Brendan McMahan et al. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[88] Andrea Menta, Alberto Archetti, and Matteo Matteucci. "Latent Neural Cellular Automata for Resource-Efficient Image Restoration". In: *ALIFE 2024: Proceedings of the 2024 Artificial Life Conference*. MIT Press. 2024.

[89] Chirag Nagpal, Xinyu Li, and Artur Dubrawski. "Deep Survival Machines: Fully Parametric Survival Regression and Representation Learning for Censored Data With Competing Risks". In: *IEEE Journal of Biomedical and Health Informatics* 25.8 (2021), pp. 3163–3175. DOI: 10.1109/JBHI.2021.3052441.

[90] Wayne Nelson. "Theory and applications of hazard plotting for censored failure data". In: *Technometrics* 14.4 (1972), pp. 945–966.

[91] Richard Van Noorden, Brendan Maher, and Regina Nuzzo. "The top 100 papers". In: *Nature* 514.7524 (2014), pp. 550–553. DOI: 10.1038/514550a.

[92] Alexander Norcliffe et al. “Survivalgan: Generating time-to-event data for survival analysis”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 10279–10304.

[93] Ziad Obermeyer et al. “Dissecting racial bias in an algorithm used to manage the health of populations”. In: *Science* 366.6464 (2019), pp. 447–453.

[94] Jean Ogier du Terrail et al. “FLamby: Datasets and Benchmarks for Cross-Silo Federated Learning in Realistic Healthcare Settings”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 5315–5334. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/232eee8ef411a0a316efa298d7be3c2b-Paper-Datasets_and_Benchmarks.pdf.

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

[96] Liudmila Prokhorenkova et al. “CatBoost: unbiased boosting with categorical features”. In: *Advances in neural information processing systems* 31 (2018).

[97] Shadi Rahimian et al. “Practical Challenges in Differentially-Private Federated Survival Analysis of Medical Data”. In: *Conference on Health, Inference, and Learning*. PMLR. 2022, pp. 411–425.

[98] Md Mahmudur Rahman and Sanjay Purushotham. “FedPseudo: Pseudo value-based Deep Learning Models for Federated Survival Analysis”. In: *arXiv preprint arXiv:2207.05247* (2022).

[99] Prajit Ramachandran, Barret Zoph, and Quoc V Le. “Searching for activation functions”. In: *arXiv preprint arXiv:1710.05941* (2017).

[100] Sashank Reddi et al. “Adaptive Federated Optimization”. In: *International Conference on Learning Representations* (2021). URL: https://openreview.net/forum?id=LkFG3lB13U5.

[101] Greg Ridgeway. “The state of boosting”. In: *Computing science and statistics* (1999), pp. 172–181.

[102] Nicola Rieke et al. “The future of digital health with federated learning”. In: *NPJ digital medicine* 3.1 (2020), pp. 1–7.

[103] James M Robins and Andrea Rotnitzky. “Recovery of information and adjustment for dependent censoring using surrogate markers”. In: *AIDS epidemiology*. Boston, Massachusetts, USA: Springer, 1992, pp. 297–331.

[104] M Schumacher et al. “Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. German Breast Cancer Study Group.” In: *Journal of Clinical Oncology* 12.10 (1994), pp. 2086–2093.

[105] Reza Shokri et al. "Membership inference attacks against machine learning models". In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 3–18.

[106] Noah Simon et al. "Regularization paths for Cox's proportional hazards model via coordinate descent". In: *Journal of statistical software* 39 (2011), pp. 1–13.

[107] Michael Sloma et al. "Empirical comparison of continuous and discrete-time representations for survival prediction". In: *Survival prediction-algorithms, challenges and applications*. PMLR. 2021, pp. 118–131.

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

[109] Luís A Vale-Silva and Karl Rohr. "Long-term cancer survival prediction using multimodal deep learning". In: *Scientific Reports* 11.1 (2021), p. 13505.

[110] Ingrid Van Keilegom and Noël Veraverbeke. "Hazard rate estimation in nonparametric regression with censored data". In: *Annals of the Institute of Statistical Mathematics* 53 (2001), pp. 730–745.

[111] Ping Wang, Yan Li, and Chandan K. Reddy. "Machine Learning for Survival Analysis: A Survey". In: *ACM Comput. Surv.* 51.6 (Feb. 2019). ISSN: 0360-0300. DOI: 10.1145/3214306. URL: https://doi.org/10.1145/3214306.

[112] Shirly Wang et al. "MIMIC-Extract: a data extraction, preprocessing, and representation pipeline for MIMIC-III". In: CHIL '20. Toronto, Ontario, Canada: Association for Computing Machinery, 2020, pp. 222–235. ISBN: 9781450370462. DOI: 10.1145/3368555.3384469. URL: https://doi.org/10.1145/3368555.3384469.

[113] Xuan Wang et al. "SurvMaximin: robust federated approach to transporting survival risk prediction models". In: *Journal of biomedical informatics* 134 (2022), p. 104176.

[114] David S Watson et al. "Adversarial random forests for density estimation and generative modeling". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 5357–5375.

[115] Simon Wiegrebe et al. "Deep learning for survival analysis: a review". In: *Artificial Intelligence Review* 57.3 (2024), p. 65.

[116] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *CoRR* abs/1708.07747 (2017). arXiv: 1708.07747.

[117] Lei Xu et al. "Modeling tabular data using conditional gan". In: *Advances in neural information processing systems* 32 (2019).

[118] Chun-Nam Yu et al. "Learning patient-specific cancer survival distributions as a sequence of dependent regressors". In: *Advances in neural information processing systems* 24 (2011).

[119] D Kai Zhang, Francesca Toni, and Matthew Williams. "A federated cox model with non-proportional hazards". In: *Multimodal AI in Healthcare*. Springer, 2023, pp. 171–185.

[120] Huajun Zhou, Fengtao Zhou, and Hao Chen. "Cohort-Individual Cooperative Learning for Multimodal Cancer Survival Analysis". In: *IEEE Transactions on Medical Imaging* 44.2 (2025), pp. 656–667. DOI: 10.1109/TMI.2024.3455931.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.