Alma Mater Studiorum · Università di Bologna

Dipartimento di Informatica — Scienza e Ingegneria
Corso di Laurea Magistrale in Informatica

Efficiency of foundation models on detecting patient outcomes from clinical data

Relatore:
Prof. Alina Sîrbu

Presentata da:
Luca Cotugno

Correlatori:
Prof. Martin Crane
Prof. Marija Bezbradica

Sessione Luglio 2026
Anno Accademico 2025/2026

A tutte le persone con dentro un continuo mare in tempesta.
Non serve a niente rintanarvi in voi stessi
perché siamo solo conchiglie, sparse sulla sabbia ...

Abstract

Survival analysis plays a central role in clinical research, where predicting the time until an event such as death is essential to support decision-making. While classical statistical and machine learning models are well established for this task, the recent emergence of tabular foundation models - large models pretrained on many datasets and applied through in-context learning - raises the question of whether they can also be useful for survival analysis on clinical data.

This thesis investigates the use of tabular foundation models as feature extractors for clinical survival analysis. A pipeline is developed in which two pretrained models, TabPFN and TabICL, produce embeddings from the input data, and a set of survival heads - a Cox model, a Random Survival Forest, and neural DeepSurv models - is trained on top of these embeddings. The same heads applied to the raw features serve as baselines, so that the contribution of the embeddings can be isolated. The approach is evaluated on two real clinical datasets using the Antolini time-dependent concordance index, and is studied with respect to model comparison, training set size, robustness to missing values, and interpretability through SHAP.

The results show that the embedding-based approach is a viable alternative to classical models but does not generally outperform them, with TabPFN consistently stronger than TabICL. Its main advantage emerges in specific settings, most notably in the low-data regime, where the TabPFN embeddings clearly outperform the baselines. The SHAP analysis further reveals that the choice of backbone, rather than the survival head, primarily determines which clinical features drive the predictions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Survival analysis [1–4] is a branch of statistics focused on modeling the time until the occurrence of a specific event, such as death in a biological organism or failure in a mechanical system. It plays a crucial role in several domains, particularly in healthcare, where understanding patient outcomes and risk trajectories can support clinical decision-making and personalized medicine.

Traditional survival analysis methods, such as the Cox Proportional Hazards Model [5], have been widely adopted due to their interpretability and statistical robustness. However, these approaches often rely on strong assumptions, including proportional hazards and linear relationships between covariates and risk. As medical datasets become increasingly large, heterogeneous, and high-dimensional, conventional statistical models may struggle to capture complex nonlinear patterns present in modern clinical data.

In recent years, machine learning and deep learning techniques have emerged as promising alternatives for survival prediction tasks. Models based on neural networks [6] or ensemble learning [7], have demonstrated the ability to model complex interactions among variables and improve predictive performance in several clinical applications. At the same time, the rapid development of foundation models has significantly transformed the field of artificial intelligence. Foundation models are large-scale models trained on broad

and diverse datasets using self-supervised or transfer learning approaches, enabling adaptation to multiple downstream tasks. Initially introduced and popularized in natural language processing through transformer-based architectures, foundation models have subsequently expanded to domains such as computer vision, multimodal learning, and healthcare.

The application of foundation models to clinical and biomedical data represents a promising research direction. By leveraging large-scale pretraining, these models may provide more robust feature representations, improved generalization capabilities, and better performance in low-label scenarios. Despite their growing success in several healthcare-related tasks, their effectiveness in survival analysis remains relatively underexplored.

This thesis investigates the effectiveness of foundation models in survival analysis tasks using clinical data. The study evaluates whether pretrained models can improve predictive performance compared to traditional machine learning and deep learning approaches. Different experimental settings and two datasets are considered to assess both predictive accuracy and model robustness. In addition, SHAP analysis was conducted to provide a more in-depth interpretation of the proposed systems. The complete source code and all experimental results are publicly available in the GitHub repository at: `https://github.com/lucacotu/Foundation_model_clinical_data`

The main contributions of this thesis can be summarized as follows:

- A complete analysis pipeline that combines tabular foundation models with survival analysis, in which two pretrained tabular foundation models (TabPFN and TabICL) are used as feature extractors whose embeddings feed a set of survival heads.

- A systematic comparison methodology for evaluating this approach, based on cross-validation repeated over multiple seeds and on the Antolini time-dependent concordance index, which allows the embedding-based configurations to be compared against baseline models on identical data and under consistent conditions.

- A set of survival models trained on two real clinical datasets, providing concrete evidence on the behavior of foundation-model-based survival prediction on this kind of data.

- An analysis of the robustness of the approach with respect to two factors: the size of the training set and the presence of missing values, the latter compared against a range of conventional imputation strategies.

- An interpretability analysis based on SHAP values, which attributes the predicted risk to the original clinical variables and enables a comparison of feature importance across the different models.

This thesis is the result of a three-month research period carried out abroad at the ADAPT Centre of Dublin City University (DCU), in Dublin, within a collaboration program between the two universities. This experience provided the opportunity to carry out the present work within an international research environment.

During this period, a number of competences were acquired. On the technical side, these include the use of tabular foundation models, the application of deep learning to survival analysis, the development of reproducible experimental pipelines on a high-performance computing cluster, and the use of interpretability techniques such as SHAP. On the personal side, the experience fostered the ability to work within an international research group, to independently manage a research project, and to write academic papers.

Part of the methodology developed in this thesis has been accepted for presentation at AIIH (AI in Healthcare) 2026, held in London. In addition, an extended version of this work is currently being prepared for submission to a journal.

The remainder of this thesis is organized as follows. Chapter 2 provides a review of the literature on survival analysis and tabular foundation models. Chapter 3 presents the theoretical background of the considered algorithms, the evaluation metrics employed in the experiments, and the SHAP framework adopted for model interpretability analysis. Chapter 4 describes the

two datasets considered in this work. Chapter 5 details the experimental setup and the conducted experiments. Chapter 6 discusses the experimental results and the SHAP-based analysis used to evaluate feature contributions and model interpretability. Chapter 7 concludes the thesis by discussing the main findings and suggesting directions for future work.

Chapter 2

Literature Review

2.1 Survival Analysis

Survival analysis is concerned with predicting the time until the occurrence of an event of interest, a setting that differs from standard regression because of censoring: for some subjects the event is not observed within the study period, and only a lower bound on their time-to-event is available. Any method designed for this task must therefore account for both the observed event times and the censored observations.

For several decades, the field has been dominated by classical statistical methods. The Kaplan-Meier estimator [8] provides a non-parametric estimate of the survival function, while the Cox proportional hazards model [5] relates the covariates to the hazard through a regression formulation, and has become the most widely used model in the field thanks to its interpretability and statistical robustness. These methods, however, rely on strong assumptions - in particular the proportionality of hazards and a linear relationship between the covariates and the log-hazard - which may not hold for the complex, high-dimensional data encountered in modern applications.

To relax these assumptions, machine learning methods have been adapted to time-to-event data. Random Survival Forests [7] extend the random forest framework to censored data by modifying the splitting criterion, making

it possible to capture non-linear effects and interactions among covariates without specifying them in advance.

More recently, deep learning has been applied to survival analysis, giving rise to a broad family family of neural methods [9]. DeepSurv [6] replaces the linear predictor of the Cox model with a neural network, retaining the proportional hazards formulation, while allowing non-linear risk functions to be learned. DeepHit [10] departs from the proportional hazards assumption altogether, directly modeling the distribution of event times in a discrete-time setting and naturally accommodating competing risks. Cox-Time [11] further extends the Cox framework to non-proportional hazards and introduces a mini-batch approximation of the partial likelihood that allows such models to scale to large datasets. These approaches have been shown to improve predictive performance in several clinical applications, at the cost of greater model complexity and, typically, needing a larger amount of training data.

2.2 Foundation Models

In parallel with these developments, the broader field of artificial intelligence has been reshaped by the emergence of foundation models [12]. A foundation model is a large-scale model pretrained on broad and diverse data through self-supervised learning, which can then be adapted to a wide range of downstream tasks rather than being trained from scratch for each one. This paradigm was enabled by the transformer architecture and its self-attention mechanism [13], which allowed models to be scaled effectively to large amounts of data.

Foundation models first emerged and matured in natural language processing. Pretrained language models such as BERT [14] showed that a single model, pretrained once, could be adapted to many language tasks and achieve strong performance, while GPT-3 [15] demonstrated that, at a sufficient scale, such models can address new tasks from only a few examples provided as context, without any update of their parameters. This last property, known as

in-context learning, is particularly relevant to the present work.

Despite their success across these and other domains, foundation models were initially developed for unstructured data such as text. Tabular data, which is ubiquitous in clinical and scientific applications, has only more recently become the object of comparable efforts, as discussed in the following section.

2.3 Tabular Foundation Models

Among the domains in which foundation models have only recently gained traction, tabular data occupies a peculiar position. Unlike text and images, tabular data had long resisted the advances of deep learning: for years, ensembles of gradient-boosted decision trees remained the state of the art, consistently outperforming neural approaches on typical tabular problems [16]. This long-standing dominance is precisely what makes the recent emergence of tabular foundation models notable.

These models challenge the established paradigm by bringing in-context learning to tabular data: rather than being trained on the target dataset, the model receives the training data as context and produces predictions for the query samples in a single forward pass, without any update of its parameters. TabPFN [17] was among the first models to follow this approach: it is a Prior-Fitted Network pretrained on a large collection of synthetic datasets, which learns to approximate Bayesian inference and delivers strong performance on small tabular problems without task-specific training. Its successor, TabPFN v2 [18], substantially improved both predictive performance and scalability, and is the version employed in this work.

A subsequent model has addressed different limitation of this paradigm. TabICL [19] introduces a two-stage, column-then-row architecture designed to scale in-context learning to much larger datasets, while retaining performance comparable to TabPFN v2. Both models share the same underlying principle of in-context learning and have demonstrated strong results

on tabular classification and regression tasks. Their application to survival analysis, and more broadly to clinical data, has nonetheless remained largely unexplored, as discussed in the following section.

2.4 Foundation Models for Survival Analysis

The intersection between tabular foundation models and survival analysis is a very recent and rapidly developing research direction, with most contributions appearing only in the last year. Since these models are natively designed for classification and regression rather than for censored time-to-event data, the existing works differ mainly in how they bridge this gap, and several distinct strategies have emerged.

A first line of work reformulates survival analysis as a classification problem so that existing tabular foundation models can be applied directly. Kim et al. [20], for instance, discretize the event times and cast both static and dynamic survival analysis as a sequence of binary classification problems, treating censored observations as examples with missing labels, which allows the foundation models to perform survival analysis through in-context learning without explicit training. A second line of work instead designs foundation models specifically for survival data: the Survival In-Context model [21], for example, is pretrained on synthetic data generated from a survival-specific prior, so that time-to-event prediction with censoring is supported natively.

This thesis is positioned within this emerging research direction and builds on our own previous work [22], in which tabular foundation models were used as feature extractors and combined with a survival-aware head to model right-censored outcomes on clinical data. The present thesis adopts the same embedding-extraction strategy but differs from that work in several respects. First, in addition to using the embeddings, that work also employed the foundation models in a zero-shot fashion, producing the survival predictions directly without a trained head; the present work, in contrast, relies

exclusively on the extracted embeddings. Second, in that work the embeddings were combined with a single multi-task logistic regression head, while classical and neural survival models were used only as baselines on the raw data; here, instead, a broader range of such classical and neural survival heads is built directly on top of the embeddings. Finally, the present work is applied to different clinical datasets and extends the analysis beyond predictive performance, studying the robustness of the approach with respect to the training set size and the presence of missing values, as well as the interpretability of the models through SHAP. To the best of our knowledge, this combination has not yet been systematically investigated, which is the gap this thesis aims to address.

Chapter 3

Background

3.1 Survival Analysis Fundamentals

3.1.1 Time-to-Event Data and Censoring

Survival analysis [23] deals with modeling the time to the occurrence of an event of interest, referred to as an event time. What distinguishes it from a standard regression problem is that, for some subjects, the event is not observed during the study period: this phenomenon is known as censoring. The most common form, and the one considered in this work, is right-censoring, which occurs when a subject leaves the study, is lost to follow-up, or has not yet experienced the event by the end of the observation window. In all these cases, the exact event time is unknown and only a lower bound on it is available - namely, the time at which the subject was last observed.

For each subject i, the data are therefore represented by a pair (t_i, δ_i). Here t_i denotes the observed time, which corresponds to the event time if the event was observed and to the censoring time otherwise, while $\delta_i \in \{0, 1\}$ is the event indicator: $\delta_i = 1$ if the event was observed at t_i and $\delta_i = 0$ if the observation was censored. Each subject is also associated with a vector of covariates x_i, that is, the features used to predict the outcome. The goal of survival analysis is to model the distribution of the event time as a function

of these covariates, while correctly accounting for the information carried by censored observations.

Discarding censored subjects, or treating their observed time as if it were the true event time, would waste information and introduce bias, since censored observations still indicate that the subject survived at least up to the censoring time. Survival analysis methods are designed precisely to incorporate this partial information into the estimation.

3.1.2 Survival and Hazard Functions

The distribution of the event time T, viewed as a non-negative random variable, can be described through several equivalent functions. The central one is the survival function $S(t)$, defined as the probability that the event has not yet occurred by time t:

$$S(t) = P(T > t) = 1 - F(t), \tag{3.1}$$

where $F(t)$ is the cumulative distribution function of T. By definition, $S(t)$ is a monotonically non-increasing function with $S(0) = 1$, since every subject is event-free at the origin, and $S(t) \to 0$ as $t \to \infty$. In the survival setting, $S(t)$ is often the quantity of primary interest, as it directly expresses the probability of survival beyond a given time.

An equivalent characterization is given by the hazard function $h(t)$, which describes the instantaneous rate at which the event occurs at time t, given that the subject has survived up to that point:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t}. \tag{3.2}$$

While the survival function describes the probability of surviving up to a certain time, the hazard function describes the risk of experiencing the event at that time, conditional on having survived so far. It is therefore a natural way to express how the instantaneous risk evolves over time, and it plays a

central role in many survival models, including those considered in this work.

Closely related is the cumulative hazard function $H(t)$, obtained by integrating the hazard over time:

$$H(t) = \int_0^t h(u)\, du. \tag{3.3}$$

These functions are not independent: knowing any one of them determines the others. In particular, the survival function and the cumulative hazard are linked by the relation

$$S(t) = \exp\big(-H(t)\big), \tag{3.4}$$

which shows that the survival function can be recovered from the hazard, and vice versa. This equivalence is what allows different survival models to be formulated in terms of whichever function is most convenient - for example, the hazard for the models discussed in the following sections - while still describing the same underlying distribution.

3.1.3 The Kaplan-Meier Estimator

The functions introduced so far describe the event-time distribution in theory, but in practice they must be estimated from observed data. The most widely used non-parametric estimator of the survival function is the Kaplan-Meier estimator [8], which estimates $S(t)$ directly from the observed times and event indicators, without assuming any parametric form for the distribution.

The estimator is built from the distinct times at which events are observed. Let $t_1 < t_2 < \cdots < t_m$ be these ordered event times, and, for each of them, let d_j be the number of events occurring at t_j and n_j the number of subjects still at risk just before t_j, that is, those who have neither experienced the event nor been censored up to that point. The Kaplan-Meier estimate of

the survival function is then given by the product

$$\hat{S}(t) = \prod_{j:\, t_j \leq t} \left(1 - \frac{d_j}{n_j}\right). \tag{3.5}$$

Intuitively, at each event time the factor $1 - d_j/n_j$ represents the proportion of at-risk subjects who survive that specific instant, and the overall survival probability up to time t is obtained by multiplying these conditional survival proportions across all event times up to t. The result is a step function that starts at 1 and decreases at each observed event time, remaining constant between consecutive events.

Censoring is handled naturally within this formulation: a censored subject contributes to the number at risk n_j up to its censoring time, and is then removed from the risk set without producing a drop in the estimate. In this way, the partial information carried by censored observations is retained rather than discarded. The Kaplan-Meier estimator provides a population-level description of survival that does not depend on covariates; modeling how survival varies as a function of the covariates x_i requires the regression-based approaches discussed in the following section.

3.2 Survival Models

3.2.1 Cox Proportional Hazards Model

The Cox Proportional Hazards model [5] is the most widely used regression model in survival analysis. Rather than modeling the survival function directly, it models the hazard function as a function of the covariates, and does so in a semi-parametric way: the effect of the covariates is specified explicitly, while the baseline evolution of the hazard over time is left unspecified. For a subject with covariate vector x, the hazard is written as

$$h(t \mid x) = h_0(t)\, \exp\big(\beta^\top x\big), \tag{3.6}$$

where $h_0(t)$ is the baseline hazard, common to all subjects and left unspecified, and β is the vector of regression coefficients to be estimated. The term $\exp(\beta^\top x)$ scales the baseline hazard according to the covariates, so that each subject's hazard is a fixed multiple of the same baseline function.

This formulation embodies the proportional hazards assumption: the ratio of the hazards of any two subjects,

$$\frac{h(t \mid x_i)}{h(t \mid x_j)} = \exp\big(\beta^\top (x_i - x_j)\big), \tag{3.7}$$

does not depend on time, since the baseline hazard $h_0(t)$ cancels out. In other words, the model assumes that the relative risk between two subjects remains constant over the whole follow-up period. A further assumption is that the log-hazard is a linear function of the covariates, through the term $\beta^\top x$.

A distinctive feature of the model is that the coefficients β can be estimated without specifying the baseline hazard $h_0(t)$. This is achieved by maximizing the partial likelihood, which exploits the ordering of the observed event times and the corresponding risk sets, without requiring the baseline hazard to be specified. At each observed event time, the partial likelihood compares the covariates of the subject who experienced the event with those of all subjects still at risk at the moment:

$$L(\beta) = \prod_{i\,:\,\delta_i = 1} \frac{\exp\big(\beta^\top x_i\big)}{\sum_{j \in R(t_i)} \exp\big(\beta^\top x_j\big)}, \tag{3.8}$$

where the product is taken over the subjects for which the event was observed ($\delta_i = 1$), and $R(t_i)$ denotes the risk set at time t_i, that is, the set of subjects still under observation just before t_i. Maximizing $L(\beta)$ yields the coefficient estimates, from which the relative risk associated with each covariate can be interpreted. The interpretability of the coefficients, together with the fact that the baseline hazard need not be specified, is one of the main reasons for the enduring popularity of the model; its limitations lie

precisely in its assumptions of proportional hazards and log-linearity, which motivate the more flexible approaches described next.

3.2.2 Random Survival Forests

Random Survival Forests [7] extend the random forest framework to right-censored time-to-event data. Unlike the Cox model, they make no assumption of proportional hazards or of a log-linear relationship between the covariates and the risk, and can capture non-linear effects and interactions among the covariates without specifying them in advance. This flexibility comes at the cost of losing the direct interpretability offered by the regression coefficients of the Cox model.

As in standard random forests, the model is an ensemble of decision trees, each grown on a bootstrap sample of the training data and using a random subset of the covariates at each split, so that the trees are diverse and their combination reduces variance. The key difference lies in the splitting criterion, which is adapted to censored data: instead of minimizing a regression or classification loss, each candidate split is evaluated by how well it separates subjects with different survival behavior, typically using a log-rank statistic that compares the survival distributions of the two resulting groups. The split that yields the largest separation is selected, so that each tree progressively partitions the subjects into groups that are increasingly homogeneous in terms of their time-to-event outcome.

Within each terminal node of a tree, the subjects are treated as a homogeneous group, and a non-parametric estimate of their outcome is computed - for instance, a Kaplan-Meier estimate of the survival function, or the corresponding cumulative hazard estimate. A prediction for a new subject is obtained by dropping it down each tree until it reaches a terminal node, and reading off the estimate stored there. The prediction of the forest is then the

average of the estimates produced by the individual trees:

$$\hat{S}(t \mid x) = \frac{1}{B} \sum_{b=1}^{B} \hat{S}_b(t \mid x), \tag{3.9}$$

where B is the number of trees and $\hat{S}_b(t \mid x)$ is the survival function estimated by the b-th tree for a subject with covariates x. By averaging over many diverse trees, the forest produces a smoother and more robust estimate of the survival function than any single tree, while retaining the ability to model complex, non-linear dependencies on the covariates.

3.2.3 DeepSurv

DeepSurv [6] is a neural network extension of the Cox proportional hazards model. It retains the proportional hazards structure but replaces the linear predictor $\beta^\top x$ with a non-linear function of the covariates learned by a neural network. The hazard is written as

$$h(t \mid x) = h_0(t) \exp\big(f_\theta(x)\big), \tag{3.10}$$

where $f_\theta(x)$ is the output of a neural network with parameters θ, which produces a scalar risk score for each subject. This allows the model to capture non-linear effects and interactions among the covariates, which the standard Cox model cannot represent, while keeping the same multiplicative relationship between the risk score and the baseline hazard.

The network is trained by minimizing a loss derived from the Cox partial likelihood introduced in Section 3.2.1, in which the linear term $\beta^\top x$ is replaced by the network output $f_\theta(x)$. The corresponding objective is the negative log partial likelihood,

$$\ell(\theta) = -\sum_{i\,:\,\delta_i=1} \left(f_\theta(x_i) - \log \sum_{j \in R(t_i)} \exp\big(f_\theta(x_j)\big) \right), \tag{3.11}$$

where the sum is taken over the subjects for which the event was observed and $R(t_i)$ is the risk set at time t_i. Minimizing this loss with respect to θ, by gradient descent, yields a network that assigns higher risk scores to subjects who experience the event earlier. As in the Cox model, the network produces only a relative risk score; to obtain a full survival function, the baseline hazard $h_0(t)$ is estimated separately from the training data, after which the survival function of any subject can be reconstructed.

It is worth noting the relationship between DeepSurv and the classical Cox model. When the network f_θ reduces to a single linear layer with no hidden layers, the risk score becomes a linear function of the covariates, and the model shares exactly the same proportional-hazards, log-linear form as the Cox model of Section 3.2.1. In this case, the two models differ only in how their parameters are estimated: the classical Cox model maximizes the partial likelihood using dedicated optimization procedures, whereas DeepSurv optimizes the corresponding negative log partial likelihood by gradient-based optimization. In practice, neural-network implementations often perform this optimization using stochastic or mini-batch gradient descent, although the specific optimization strategy depends on the implementation. The two therefore describe the same functional form through different estimation procedures, and coincide in the limit of a purely linear network.

3.3 Tabular Foundation Models

3.3.1 In-Context Learning for Tabular Data

Foundation models for tabular data depart from the conventional machine learning workflow, in which a model is trained on a specific dataset before being used for prediction. Instead, they rely on in-context learning [24]: a single model is pretrained once, on a large collection of datasets, and is then applied to new tasks without task-specific training. At inference time, the model receives a set of labeled examples as context and produces predictions for the query samples in a single forward pass, conditioning its output on the

provided context rather than adjusting its weights.

Concretely, given a labeled training set $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$ and a query sample x, the model produces a prediction $\hat{y} = f_\theta(x; \mathcal{D}_{\text{train}})$, where the parameters θ are those learned during pretraining and are kept fixed. The training set therefore plays the role of a context that steers the prediction, in a way that is analogous to how large language models solve new tasks from a few examples provided in their prompt. This is what allows such models to be used as general-purpose predictors, or, as in this work, as feature extractors whose internal representations can be reused by other models.

The models considered in this work - TabPFN and TabICL - both follow this in-context learning paradigm, but differ in the way they are pretrained and in the architectural choices they adopt to encode the context. The following sections describe each of them, focusing on the aspects that distinguish one from another.

3.3.2 TabPFN

TabPFN [17] is a Prior-Data Fitted Network (PFN), a transformer-based model that casts tabular prediction as amortized Bayesian inference. Rather than being fitted to a specific dataset, it approximates the posterior predictive distribution $p(y \mid x, \mathcal{D}_{\text{train}})$ directly: the outcome for a query sample x is predicted by conditioning on the training set $\mathcal{D}_{\text{train}}$ provided as context, without any parameter update. The learning procedure that a classical model would carry out on each dataset is thus internalized once, at pretraining time, into the fixed weights of the network.

The distinctive aspect of TabPFN lies in how it is pretrained. Instead of using real-world data, the model is trained on a very large collection of synthetic datasets sampled from a synthetic data-generating prior based on structural causal models. This prior is designed to generate data that reproduce characteristics commonly found in real tabular data, such as non-linear feature interactions, noise, and mixed feature types. Each synthetic dataset is split into a context part and a query part, and the transformer is trained to

predict the query labels given the context, so that over millions of such tasks it learns a general-purpose approximation of a Bayesian inference procedure. The prior, favoring simple and causally structured relationships, acts as an inductive bias embedded in the pretrained model.

Architecturally, TabPFN processes a table with a two-way attention mechanism that alternates between two directions. In one direction, attention is applied across features within each sample; in the other direction, attention is applied across samples for each feature. This design makes the model invariant to the ordering of both rows and columns, which is a natural property to require of tabular data, and allows the same pretrained model to be applied to tables with different numbers of features and samples. The version used in this work is TabPFN v2, which improves the predictive performance and the scalability of the original model to larger tabular problems.

3.3.3 TabICL

TabICL [19] is a tabular foundation model that follows the same in-context learning paradigm as TabPFN, but is specifically designed to scale to large datasets. The two-way attention mechanism of TabPFN operates directly on the original table representation, and its computational cost becomes prohibitive as the number of samples increases. TabICL addresses this limitation through a two-stage architecture that first compresses each row into a fixed-dimensional embedding, and only then performs in-context learning on these compact row representations, thereby reducing the cost of the most expensive stage.

The first stage builds the row embeddings through a column-then-row mechanism. In the column-wise step, each feature value is embedded using column-wise attention that captures the distributional structure of the feature across samples, producing distribution-aware representations of the individual cells. In the row-wise step, a second transformer lets the features of the same row interact, and aggregates them into a single fixed-size vector that represents the entire row. This collapses the column dimension: regard-

less of the original number of features, each sample is described by a row embedding of the same dimensionality, which is what makes the subsequent stage efficient.

The second stage performs the actual in-context learning. The row embeddings of the context, together with their labels, and those of the query samples are processed by a final transformer that predicts the query labels in a single forward pass. Compared with TabPFN, this separation between a row-embedding stage and an in-context learning stage is the key architectural difference: by paying the cost of the column-wise computation only once, when building the row embeddings, TabICL retains in-context learning while scaling to larger tables.

3.4 Evaluation and Interpretation

3.4.1 The Concordance Index

The predictive performance of survival models is commonly assessed through the concordance index (C-index), a measure of how well a model ranks subjects according to their risk. The underlying idea is that, for a pair of subjects, the one who experiences the event earlier should be assigned a higher risk by the model. A pair is said to be concordant when the model's predicted ranking agrees with the observed ordering of the event times. The C-index is then defined as the proportion of concordant pairs among all comparable pairs, so that a value of 1 corresponds to a perfect ranking, while a value of 0.5 corresponds to a ranking no better than random.

Censoring makes this comparison more delicate, since not all pairs can be ordered: a pair is comparable only when the ordering of the event times can be determined despite censoring. For instance, if the subject with the shorter observed time is censored, it is not possible to know whether the true event would have occurred before or after that of the other subject, and the pair is therefore excluded from the computation. The classical formulation of the C-index, introduced by Harrell [25], relies on a single scalar score per

subject to establish this ranking, and is the most widely used version of the metric in classical survival analysis.

A limitation of Harrell's formulation is that it summarizes each subject with a single, time-independent risk score, which does not fully reflect models whose predicted risk varies over time. To address this, Antolini [26] proposed a time-dependent concordance index, which compares subjects using their predicted survival probabilities at the specific time at which the earlier event occurs, rather than using a time-independent risk score. In this way, the ranking is evaluated in terms of the full predicted survival function $\hat{S}(t \mid x)$, which makes the metric consistent with the models providing time-dependent survival predictions. For this reason, the time-dependent C-index of Antolini is the metric adopted in this work, as it provides a more faithful assessment of the survival predictions than the time-independent formulation.

3.4.2 SHAP Values

While the concordance index quantifies how well a model predicts, it says nothing about why a given prediction is made. To interpret the models and understand the contribution of each feature to their predictions, this work relies on SHAP (SHapley Additive exPlanations) [27], a framework for explaining the output of machine learning models.

SHAP is grounded in the concept of Shapley values, originally introduced in cooperative game theory to distribute the payoff of a game fairly among its players. The idea is to quantify the contribution of each player by considering how much it adds to every possible coalition of the other players. Transposed to machine learning, the players become the input features and the payoff becomes the model's prediction: the Shapley value of a feature measures its average contribution to the prediction, computed over all possible subsets of the remaining features. This construction gives SHAP two desirable properties. It is additive, in the sense that the contributions of the individual features sum up to the difference between the model's prediction and a baseline expected value, and it provides a principled attribution of the

prediction that satisfies the Shapley axioms.

For a given prediction, each feature receives a SHAP value that expresses both the magnitude and the direction of its contribution: a positive value indicates that the feature pushes the prediction above the baseline, while a negative value indicates that it pushes it below. Because a SHAP value is computed for each feature and each sample, these local explanations can be aggregated across the dataset - for example, by averaging their absolute values - to obtain a global measure of the overall importance of each feature. In this work, SHAP is used to interpret the predictions of the survival models by attributing the model output to the original clinical features, so as to identify which variables most influence the predictions and to compare the behavior of the different models.

Chapter 4

Datasets

This chapter describes the two datasets employed in this work. Both are private and were kindly provided by the University of Pisa and by the IFC institute at CNR. As they are not publicly available, they cannot be redistributed, and access was granted exclusively for the purposes of this research.

The two datasets describe distinct patient populations and differ substantially in size and clinical focus, which allows the proposed approach to be evaluated on two independent cohorts and thus provides a more robust assessment of its generalizability.

4.1 IHD

The dataset is an extension of that used in Pingitore et al. [28] and contains data from Ischemic Heart Disease (IHD) patients hospitalized at the CNR Clinical Physiology Institute in Pisa, Italy and it is composed of 8,144 samples described by 85 features. It originates from a cardiovascular cohort with a particular focus on thyroid function and its relationship with cardiac conditions. The variables span several clinical domains: demographic information (age, sex), thyroid function tests (TSH, fT3, fT4) together with the corresponding thyroid status categories (euthyroidism, subclinical hypo- and

hyperthyroidism, low-T3, hypo- and hyperthyroidism), the lipid profile (total cholesterol, HDL, LDL, triglycerides), glycaemia, anthropometric measurements (weight, height, BMI), and haemodynamic parameters (systolic and diastolic blood pressure, heart rate). A large group of variables encodes the cardiovascular history and diagnoses of each patient (e.g. angina, previous CABG and PCI, previous and acute myocardial infarction, coronary artery disease, number of affected vessels, and several forms of cardiomyopathy), as well as the main cardiovascular risk factors (smoking, diabetes, hypertension, dyslipidaemia, atrial fibrillation) and the pharmacological treatment in use. The dataset also includes echocardiographic measurements (such as the wall motion score index, septal and posterior wall dimensions, and ejection fraction) and a longitudinal component: seven date fields (sampling date, follow-up date, date and cause of death, and the dates of selected events such as CABG, non-fatal AMI, stroke and PCI) record the temporal evolution of each case, together with the corresponding clinical outcomes (overall and cardiovascular mortality, fatal myocardial infarction or sudden death, non-fatal events and revascularisation). Most variables are numerical (76 attributes), of which 52 are binary indicators of the presence or absence of a given condition; 7 fields are dates and 2 are textual. A complete description of the variables is provided in Table A.1 in Appendix A.

4.2 URRAH

The second dataset is considerably larger, comprising 27,078 patients described by 85 variables, and originates from the Uric acid Right for heArt Health (URRAH) project [29], a retrospective, observational, multicenter cohort study centered on arterial hypertension and its cardiovascular complications. Rather than being collected ad hoc, the data were assembled retrospectively by aggregating clinical information already gathered during routine care across several pre-existing cohorts from Italian hypertension centres distributed over most of the country, which were merged into a single

general database. A patient was included provided that at least one serum uric acid measurement was available at the indexing date and that the study variables were sufficiently complete. Each record is associated with a follow-up of at least twenty years (extending up to 31 July 2017), and all data were anonymized before processing. Alongside demographic data (age, sex), the dataset records family history (of hypertension and of cardiovascular disease), lifestyle factors (current and former smoking, alcohol consumption and estimated daily ethanol intake, physical activity), and relevant comorbidities (diabetes, chronic kidney disease, gout). Clinical measurements include anthropometry (weight, height, BMI, waist circumference), blood pressure and heart rate, and an extensive laboratory panel (creatinine, urea, uric acid, several albuminuria indices, glucose, total cholesterol, HDL, triglycerides, haemoglobin and haematocrit). Markers of target-organ damage are also present (left ventricular hypertrophy, carotid intima-media thickness and plaque), together with a detailed record of antihypertensive and lipid-lowering medication (ACE inhibitors, ARBs, calcium-channel blockers, beta-blockers, several classes of diuretics, statins, fibrates and ezetimibe). Finally, the dataset reports the cardiovascular outcomes observed during follow-up - non-fatal and fatal myocardial infarction, cerebrovascular events, heart failure, coronary revascularization and cardiovascular death - each accompanied by the corresponding follow-up time. The variables are almost entirely numerical (82 continuous attributes and two integer fields), with a single textual field, and 44 of them are binary. A complete description of the variables is provided in Table A.2 in Appendix A.

4.3 Preprocessing

The preprocessing applied to both datasets was deliberately kept light, being limited to the operations strictly necessary to obtain clean, consistent data suitable for the subsequent analysis. No heavy transformation of the original clinical values was performed, so as to keep the data as close as

possible to their original form. An overall summary of the two datasets, including some additional details, is provided in the following section.

Three groups of variables are distinguished in both datasets. The first is the survival target, which consists of an event indicator and a follow-up time: `Total mortality` and `Follow Up Data` for the IHD dataset, and `STATO_AL_FU` and `FU` for the URRAH dataset. The second group comprises a set of variables that are directly related to the occurrence of death - such as the date, cause, and type of fatal event - which would not be available at prediction time and would leak information about the outcome if used as predictors; these variables were removed from the set of covariates. [1] The third group consists of all the remaining features, which are treated as independent covariates.

Two general choices apply to both datasets. First, no feature scaling, normalization or encoding was carried out at this stage. Since the aim of the project is to compare several models, and some of these are sensitive to such transformations while others are not, applying them could have favored or penalized specific models and biased the comparison; for a first, consistent comparison, this type of preprocessing was therefore intentionally avoided. Second, regarding missing values, listwise deletion was applied only to the survival target, since a valid event indicator and follow-up time are essential for each sample. Apart from a few feature-level adjustments described below, missing values in the indipendent covariates were intentionally left untouched: as will be discussed later, the presence of missing values is itself an object of analysis in this work, and no exhaustive, feature-by-feature imputation was therefore performed.

For the IHD dataset, a small number of individual records containing inconsistent values were first corrected, and the records with a missing value in the event variable were removed. The available dates were then converted into time-to-event information, as required by the survival analysis:

[1] For the IHD dataset: `Data of death`, `Fatal MI or Sudden death`, `UnKnown`, `Accident`, `Suicide`, `CVD Death`, `Data prelievo`, `Cause of death`, `Collected by`. For the URRAH dataset: `F_IMA`, `F_CBV`, `F_HF`, `MORTE_CV`, `FU_F_IMA`, `FU_F_CBV`, `FU_F_HF`.

the follow-up time was computed, in days, as the difference between the follow-up date and the sampling date. The same transformation was applied to the dates of the non-fatal events (`CABG`, `Non Fatal AMI (Follow-Up)`, `Ictus` and `PCI`), each of which was encoded as a pair consisting of the time to the event, in days, and a binary indicator of its occurrence; this makes it possible to consider targets other than mortality in future work. Finally, the variables deemed irrelevant to the analysis were discarded. For the URRAH dataset, the records with a missing value in the target followup variable were removed. The information on menopause was corrected by imputing, for the women recorded as being in menopause but without a reported time since its onset, the corresponding mean value. A few variables were dropped because they were uninformative or affected by an excessive proportion of missing values (`LVH`, `VES`, `SAPEVA` and `SESSO0`). The several variables encoding urinary albumin in different ways were consolidated into a single categorical variable (`ALB_CAT`), and the redundant ones were removed.

4.4 Dataset overview

This section summarizes the two datasets in their final form, that is, after the preprocessing described in the previous section, and compares their main characteristics. All the figures reported here refer to the datasets as actually used in the experiments, and therefore differ from the initial figures given in Section 4.1 and Section 4.2.

After removing the records with a missing target value and discarding the variables deemed irrelevant, redundant or related to death, the IHD dataset consists of 8,065 samples described by 78 features, while the URRAH dataset comprises 21,198 samples described by 63 features. Thus, although the two datasets start from a comparable number of variables, they differ substantially in the number of samples, with URRAH containing roughly three times as many records as IHD; they also retain a distinct clinical focus, the former centered on thyroid function and cardiac conditions and the latter on

arterial hypertension and its cardiovascular complications. As for the completeness of the data, the overall proportion of missing values, computed as the ratio between the number of missing cells and the total number of cells, is 1.08% for the IHD dataset and 39.59% for URRAH. These values are not uniformly distributed across the features: a number of attributes are fully observed, whereas others exhibit considerably higher proportions of missing entries. The target variables, by contrast, contain no missing values, as a direct consequence of the listwise deletion applied to them during preprocessing. These proportions of missing values were intentionally preserved. With the sole exception of the menopause variable in the URRAH dataset, no feature-by-feature imputation was performed: the missingness is itself an object of analysis in this work. This is especially relevant in a clinical dataset, where the absence of a value is often not random but carries information in itself - for instance, a test may not have been prescribed because it was deemed clinically unnecessary, so that the missing entry already reflects an underlying clinical judgment. Replacing the missing entries would therefore have altered the very pattern that the analysis aims to investigate, besides introducing arbitrary assumptions about the unobserved values. For this reason, the missing values in the remaining features were deliberately left untouched. Table 4.1 provides a synthetic comparison of the two datasets along their main dimensions.

	IHD	**URRAH**
Samples	8,065	21,198
Features	78	63
Binary features	51	37
Missing values	1.08%	39.59%
Clinical focus	Thyroid / cardiac	Hypertension / CV

Table 4.1: Comparative overview of the two datasets after preprocessing.

Chapter 5

Methodology: Experimental Setup

5.1 Overview

This chapter describes the experimental setup used to evaluate the proposed framework, which leverages tabular foundation models as feature extractors for survival analysis. The experiments presented here pursue a twofold goal: first to assess whether tabular foundation models can be effectively employed in the context of survival analysis, and second to evaluate whether the representations they produce lead to improved survival prediction compared to the baseline models applied directly to the raw data.

To this end, all experiments share a common pipeline, in which a tabular foundation model acts as a backbone that extracts embeddings from the input data, which are then passed to a survival head that produces the risk estimates. This shared framework - including the foundation models, the survival heads, the baseline models, the handling of missing values and the evaluation protocol based on cross-validation, multiple seeds, and the Antolini concordance index - is described in Section 5.2 and summarized in Section 5.3.

Building on this framework, three experiments are carried out. The first

and main experiment (Section 5.4) compares the different combinations of foundation models and survival heads against the baseline models, and additionally analyzes feature importance through SHAP values. The second experiment (Section 5.5) investigates how the size of the training set affects predictive performance. The third experiment (Section 5.6) evaluates the robustness of the framework to missing values by progressively injecting missing entries into the training data. Finally, Section 5.7 reports the computational environment and the details required to ensure the reproducibility of the results.

5.2 Experimental Framework

5.2.1 Pipeline Overview

The proposed framework is built around a common pipeline that is shared by all the experiments described in this chapter. The core idea is to decouple representation learning from survival modeling: a tabular foundation model is used as a backbone to transform the input data into a set of embeddings, which are then consumed by a survival head responsible for producing the risk estimates. Figure 5.1 illustrates the overall architecture.

The pipeline can be summarized in three stages. In the first stage, the input data is fed to one of the tabular foundation models (TabPFN, TabICL), which acts as a frozen feature extractor: the model is not trained or fine-tuned on the data, but is used solely to map each sample into an embedding representation. The embeddings are extracted once per cross-validation fold and the same representation is then reused by all the survival heads, ensuring that they are compared on identical inputs. The dimensionality of the embeddings depends on the backbone: TabPFN produces embeddings of size 192, whereas TabICL produces embeddings of size 512. The details of how the embeddings are obtained from each backbone are described in Section 5.2.3.

In the second stage, the survival head is trained on the embeddings of

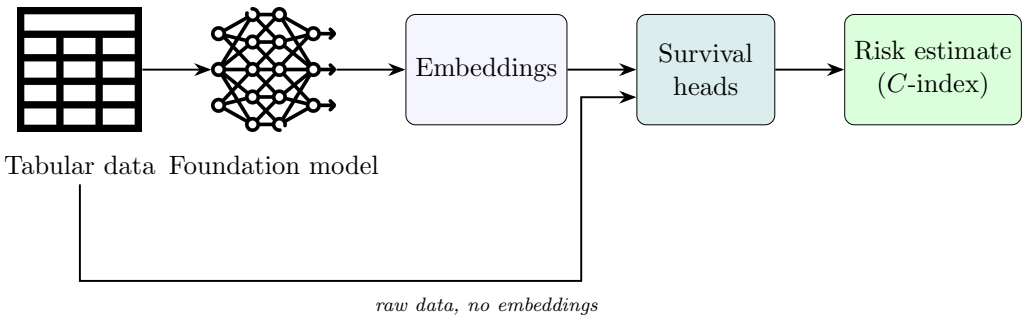

Figure 5.1: Overview of the proposed framework. The tabular data is processed by a foundation model that produces embeddings, which are fed to the survival heads to obtain the risk estimate. The same survival heads are also trained directly on the raw tabular data, bypassing the foundation model, so that the two input representations can be compared within a single component.

the training samples, while the embeddings of the validation set - which is held out of the cross-validation and kept fixed across all folds - are used, where applicable, both to validate the model during training, for instance for early stopping, and to guide the hyperparameter tuning described in Section 5.2.7. Several survival heads are considered, ranging from classical models to neural approaches, and are described in Section 5.2.5. In the third stage, the trained survival head is applied to the embeddings of the test samples of the fold, producing for each of them an estimated risk, from which the final survival predictions are derived. The way the data is partitioned into validation, training, and test sets is detailed in Section 5.2.6.

Alongside this foundation-model-based pipeline, the framework also includes a set of baseline models that operate directly on the input data, without relying on any foundation model. In this case the embedding-extraction stage is skipped, and the survival model is trained on the raw features. These baselines, described in Section 5.2.5, serve as a reference point: by comparing them with the corresponding foundation-model-based configurations, it is possible to isolate the contribution of the embeddings produced by the tabular foundation models. Since the baseline survival models cannot natively handle missing values, this comparison is only carried out in the data variant

where missing entries are replaced with valid entries (Section 5.2.2)

Regardless of the specific configuration, every model is evaluated following the same protocol, based on cross-validation repeated across multiple random seeds and on the Antolini concordance index as the evaluation metric as detailed in Section 5.2.6

5.2.2 Missing Data Variants (NaN vs -1)

Before describing the individual components of the framework, it is necessary to introduce a dimension that cuts across the entire experimental setup: the way in which missing values are treated upstream of the models. As anticipated in the pipeline overview, the framework can operate either directly on the foundation-model embeddings or on the raw features; these two paths differ in whether they can accept incomplete inputs, and this motivates the definition of two data variants. Both variants are used throughout the experiments: unless otherwise specified, each experiment is carried out under both of them.

In the first variant, the missing values are retained as such and passed to the foundation models without any imputation. This variant exploits the ability of the tabular foundation models to accept incomplete inputs directly, and it is therefore only applicable to the configurations that rely on the extracted embeddings.

In the second variant, the missing values are replaced by the constant -1 before the data enters the pipeline. Since all features are numerical after preprocessing, the substitution is applied uniformly to the numerical features, and no categorical features are involved. The value -1 was chosen because it does not occur naturally in any of the features of the datasets, so that it can act as a recognizable indicator of a missing entry rather than being confused with a valid measurement. This variant makes the data usable also by models that cannot natively handle missing values - in particular the baseline models introduced later in this chapter - and it is consequently the only variant in which those models can be evaluated, alongside the configurations based on

the embeddings.

The two variants thus serve complementary purposes. The first isolates the behavior of the foundation models when they are allowed to process missing values through their own internal mechanisms, while the second provides a common ground on which the embedding-based configurations and the models operating on the raw features can be compared on identical inputs. The way each foundation model handles the retained missing values, and the way the survival heads and baselines make use of the two variants, are detailed in the following sections.

5.2.3 Embedding Extraction from the Foundation Models

Since TabPFN and TabICL are designed for tabular classification and regression rather than for survival modeling, they are not used here to produce predictions, but as frozen feature extractors. For each model, the embeddings correspond to the internal representation that immediately precedes the model's classification head - that is, the penultimate-layer representation that the model would otherwise use to compute its output. The extraction mechanism differs slightly across the two backbones, depending on the interfaces they expose:

- For TabPFN, the embeddings are obtained through the official embedding interface provided by the library, which returns the representation produced by the transformer encoder stack before the decoder head. The resulting dimensionality is 192, corresponding to the hidden size of the pretrained TabPFN v2 model.

- For TabICL, the embeddings are extracted from a local copy of the model, by retrieving the output of its in-context learning transformer, taken before the final layer normalization and the classification head. The resulting dimensionality is 512, which arises from the model's archi-

tecture, where the per-row representation produced by multiple learned aggregation tokens is concatenated before the in-context learning stage.

Both backbones are in-context learning models: rather than being trained on the data, they receive a set of labeled examples as context and produce their internal representation for the query samples conditioned on that context. In the proposed framework, the context is always the training portion of the current fold, while the samples whose embeddings are being computed - those of the training, validation, and test sets - are processed as unlabeled queries. The way the context is supplied depends on the model: TabPFN installs the training set as context and processes context and query samples within a single forward pass; TabICL concatenates context and query samples into a single input where only the context samples carry their labels.

A key aspect of this stage concerns the label used as context. The only target provided to the foundation models is the binary event indicator, denoting whether the event of interest occurred for a given sample, while the survival time is never passed to the backbones. As a consequence, the foundation models act as a supervised feature extractors with respect to the occurrence of the event, producing representations that capture patterns associated with whether the event takes place, rather than with its timing. The temporal information and the censoring are introduced only in the subsequent stage, where the embeddings are consumed by the survival heads, which are responsible for modeling time and censored observations.

Because the context is in all cases restricted to the training portion of the fold, the embeddings of the validation and test samples are computed without exposing the models to any information outside the training data, thus preventing information leakage. The extracted embeddings are passed directly to the survival heads, without any further normalization or transformation.

5.2.4 Foundation Models and Missing Value Handling

A central motivation for adopting tabular foundation models as backbones in the proposed framework is their ability to process incomplete data

without explicit imputation. Classical survival models such as Cox regression and Random Survival Forest cannot operate on inputs containing missing entries, which is why the baseline models (Section 5.2.5) can only be applied to the data variant in which missing values are corrected (Section 5.2.2). Both foundation models, in contrast, accept inputs with missing values directly. The mechanism through which they do so, however, differs from one model to another, and this distinction is relevant both for the comparison between the two data variants and for the robustness experiment described in Section 5.6.

- TabPFN handles missing values natively, within the model itself. During its pre-training on synthetic datasets, missing values are deliberately introduced, so that the model learns to deal with them as part of its in-context learning mechanism. The data - including any missing entries - is therefore passed directly to the model, which conditions its representations on the presence of missingness without any external preprocessing.

- TabICL does not process missing values within the model, but through a preprocessing step applied before the data enters the network. Numerical features are mean-imputed, with the mean computed on the training data while for the categorical features the missing entries are encoded as a dedicated category. As a consequence, the model never observes missing values: they are resolved upstream.

It follows that, of the two backbones, only TabPFN exhibits a genuinely learned handling of missing values, in which the absence of a value is itself an informative signal that the model can exploit. TabICL, on the other hand, effectively relies on mean imputation applied as an external preprocessing step, so that the information carried by the missingness pattern is not preserved. This difference is worth keeping in mind when interpreting the results: in the variant that retains missing values, the two backbones do not handle them in an equivalent way, and their behavior under increasing

fractions of missing data may reflect the different strategies described here rather than a single shared mechanism.

5.2.5 Survival Heads

The embeddings produced by the foundation models are used as input features by a survival head, the component responsible for modeling the time-to-event outcome together with the censoring. Three families of survival heads are considered - a classical Cox model, Random Survival Forest, and DeepSurv - covering classical statistical, ensemble and neural approaches. For the Random Survival Forest and DeepSurv families, two configurations are evaluated: one in which the hyperparameters are fixed a priori, and one in which they are selected through the tuning procedure described in Section 5.2.7. Regardless of the family, every head ultimately produces, for each subject, an estimated survival function over a set of time points, which constitutes the input to the evaluation metric defined in Section 5.2.6. The specific hyperparameter values are reported in the configuration table in Section 5.2.7

Cox proportional hazards. The first head is the classical Cox proportional hazards model, in which the log-hazard is assumed to be a linear function of the input features. The model is estimated by maximizing the partial likelihood, with an L2 penalty on the coefficients, and directly yields a survival function for each subject. A single configuration is used for this head, without hyperparameter tuning.

Random Survival Forest. The second head is a Random Survival Forest, an ensemble of survival trees in which each tree estimates a survival function for a given subject and the final estimate is obtained by averaging across the trees. Unlike the Cox model, it can capture non-linear effects and interactions among the features without specifying them explicitly.

DeepSurv. The third head is DeepSurv, a Cox proportional hazards model in which the linear predictor is replaced by a neural network: the network outputs a scalar risk score for each subject, and its parameters are

optimized via gradient descent using a mini-batch approximation of the Cox partial likelihood, in which the risk set is restricted to the samples of the current batch. A baseline hazard is then estimated from the training data through the Breslow estimator, in order to convert the risk scores into survival functions. Two architectures are used for this head: a multi-layer network with hidden layers, and a network consisting of a single linear layer, which has no hidden layers and therefore no architectural hyperparameters to set.

It is worth noting that the linear DeepSurv variant shares the same proportional-hazards functional form as the classical Cox head, since in both cases the log-hazard is linear in the features. The difference lies in the estimation procedure: the classical Cox model maximizes the exact partial likelihood over the full risk set, whereas the linear DeepSurv variant optimizes a batched approximation of the same partial likelihood through gradient descent. The two heads therefore offer a comparison between two estimation strategies applied to an identical functional form.

For the Random Survival Forest and DeepSurv multi-layer heads, two configurations are evaluated: one in which the hyperparameters are fixed a priori, and one in which they are selected through the tuning procedure described in Section 5.2.7. The Cox head and the linear DeepSurv head, having no architectural hyperparameters to tune, are used in a single configuration.

Baseline models. The same survival heads also serve as baselines for assessing the contribution of the tabular foundation models. In this case the raw covariates are passed directly to the heads, without the embedding-extraction stage, so that comparing a baseline with its embedding-based counterpart isolates the effect of replacing the raw features with the representations produced by the foundation models. Since the classical survival models cannot process missing values, the baselines are evaluated only on the data variant in which the missing entries are replaced by -1.

The baselines mirror the fixed-configuration heads: the Cox model, a Random Survival Forest with fixed hyperparameters, a multi-layer DeepSurv model with a fixed architecture, and a single-layer DeepSurv model. Each

baseline uses exactly the same configuration as its embedding-based counterpart, so that the only difference between the two arms is the presence or absence of the foundation-model embeddings. No tuned variant is included among the baselines: tuning every baseline separately would have added a considerable computational cost, and it was therefore intentionally left out. This choice may have penalized the baselines to some extent, since tuning could have improved their performance; the resulting comparison can thus be read as a conservative one with respect to the baselines, and this should be kept in mind when interpreting the results.

5.2.6 Evaluation Protocol

All configurations of the framework - both the embedding-based heads and the baselines - are evaluated following the same protocol, so that their results are directly comparable. This section describes how the data is partitioned, how the evaluation is repeated to account for randomness, how the results are aggregated, and which metric is used to assess predictive performance.

Data partitioning. The data is first divided into a training set and a validation set, using 80/20 split. The validation set is held out from the cross-validation and kept fixed across all folds; it is used for early stopping and for hyperparameter tuning, where applicable. The cross-validation is then performed on the training set using five folds: at each fold, four parts are used to train the survival head and the remaining part is used as the test set on which performance is measured. As a result, the training and test partitions change from fold to fold, while the validation set remains the same throughout.

Because survival data is involved, the initial split is stratified along two dimensions simultaneously, in order to avoid imbalances that could distort either the training or the validation. A combined stratum is defined by crossing the binary event indicator with the quartile of the follow-up time: the follow-up time is discretized into quartiles, and each stratum corresponds

to a specific combination of event status and follow-up quartile. Stratifying on this combined variable ensures that both the proportion of events and the distribution of follow-up times are preserved across the split. This is particularly relevant for the Cox-based models, whose estimates rely on the comparison of subjects at risk at each time point and which would therefore be sensitive to a mismatch in the temporal distribution between the partitions. The stratum is used only to guide the split and is not provided as a feature to any model. The cross-validation folds, in contrast, are not stratified: stratification is applied only to the initial train-validation split.

It is worth noting that a fully nested cross-validation, with an inner cross-validation loop for hyperparameter selection inside each outer fold, was deliberately not adopted. Instead, the fixed held-out validation set described above is used for tuning and early stopping, which avoids information leakage from the test folds while keeping the number of models to be traines substantially lower. This choice was made for computational reasons: a fully nested procedure would have multiplied the number of trained models by the size of the inner loop, which was not feasibile given the number of configurations, seeds, and experiments considered in this work.

Repetition and aggregation. To account for variability introduced by the stochastic components of the pipeline, the entire procedure is repeated across five fixed random seeds (42,77,123,456, and 789). For a given seed, the Antolini concordance index is computed on the test partition of each fold and averages across the five folds, together with its standard deviation across folds, are reported. These per-seed quantities are then averaged across the five seeds, so that the final performance of each configuration is reported as the mean concordance index, accompanied by the corresponding standard deviation. The comparison between the different configurations is based on these aggregated values rather than on formal statistical significance tests.

Evaluation metric. Predictive performance is measured through the time-dependent concordance index proposed by Antolini. The concordance index quantifies the discriminative ability of a model, that is, its capacity

to correctly order subjects according to their risk, taking the value 0.5 for a model that is no better than random and 1 for a model with perfect discrimination. Unlike Harrell's concordance index, which relies on a single scalar risk score per subject and is computed independently of time, the Antolini index makes use of the entire predicted survival function: subjects are compared at the specific times at which events occur, using the survival probability estimated at those times. This makes it more appropriate for models that produce full survival curves, as is the case for all the heads considered here, and allows it to account for situations in which the relative ordering of risks changes over time. Censoring is handled through a Kaplan-Meier estimate of the censoring distribution, which reduces the bias that censored observations would otherwise introduce into the index.

5.2.7 Hyperparameter Tuning

Among the survival heads, two admit a tuned variant: the DeepSurv model with a multi-layer architecture and the Random Survival Forest. The remaining heads have no hyperparameters to optimize - the linear DeepSurv variant has no architectural hyperparameters, and the Cox model uses a single fixed penalty term - and are therefore always run in their fixed configuration. Hyperparameter tuning is applied only in the main experiment (Section 5.4); the other experiments use the fixed configurations only.

Search procedure. The search is carried out with Optuna, using a Tree-structured Parzen Estimator (TPE) [30] as the sampler. For each tuned head, 100 trials are evaluated, and the search is directed at maximizing the predictive performance of the resulting model. The sampler is seeded to ensure that the search is reproducible.

Selection criterion and leakage prevention. Each candidate configuration is assessed using the Antolini concordance index defined in Section 5.2.6, computed on the fixed validation set that is held out from the cross-validation. The configuration that maximizes this index on the validation set is then retained. Because the selection relies exclusively on the validation

set, the test partitions of the cross-validation folds are never involved in the tuning process, which prevents information from the test data from influencing the choice of hyperparameters. The tuning is performed independently for each fold and separately under each of the two data variants, so that the selected hyperparameters are specific to the training data of each fold and to the way missing values are handled.

Search space. The hyperparameters considered for each tuned head, together with their respective ranges, are reported in Table 5.2. For the DeepSurv head, the search covers the architecture of the network, the dropout rate, the learning rate and the use of batch normalization; for the Random Survival Forest it covers the number of trees, the depth and splitting criteria of the trees, and the number of features considered at each split.

Training details. During the search, each DeepSurv trial is trained for a fixed number of epochs without early stopping, and its configuration is scored on the validation set. Once the best configuration has been selected, the final model is retrained with a large number of maximum epochs and with early stopping based on the validation set, so that the final training length is adapted to the chosen configuration. For the Random Survival Forest each trial and the final model correspond to a single fit with the trial's hyperparameters. The specific values of these training settings are reported in Table 5.2.

5.3 Experimental Design Summary

The three experiments presented in the remainder of this chapter share the common framework described in Section 5.2 - the same pipeline, foundation models, survival heads, evaluation protocol, and missing-data variants - and differ only in which models and configurations they include and in how the data is manipulated. Before describing each experiment in detail, this section provides an overall summary of the experimental design, so that the scope of each experiment can be appreciated at a glance. Table 5.4

Table 5.1: Fixed hyperparameter configurations of the survival heads. The baseline models use the same configuration as their embedding-based counterparts.

Survival head	Hyperparameter	Value
Cox PH	penalizer (L2)	0.1
Random Survival Forest (fixed)	`n_estimators`	300
	`max_depth`	10
	`min_samples_split`	15
	`min_samples_leaf`	10
	`max_features`	log2
DeepSurv (MLP, fixed)	`num_nodes`	[32, 32]
	`batch_norm`	True
	`dropout`	0.1
DeepSurv (linear)	architecture	single linear layer
DeepSurv training settings (all non-tuned variants)	learning rate	0.01
	epochs	100
	batch size	256
	early stopping	patience 20, `min_delta` 10^{-4}

reports, for each experiment, the survival heads involved, whether the baseline models are included, whether hyperparameter tuning is applied, and the missing-data variants under which the experiment is carried out. The two foundation models (TabPFN and TabICL) are used as backbones in all three experiments.

A few aspects of the design are worth highlighting, as they distinguish the three experiments. Hyperparameter tuning is applied only in the first experiment; the second and third experiment rely exclusively on the fixed configurations of the survival heads. In the first two experiments, the baseline models are the survival heads applied directly to the raw features, and they are present only in the variant where missing values are replaced by -1, since they cannot process missing entries. The third experiment follows a different scheme: it starts from the variant in which missing values are replaced by -1 and progressively injects synthetic missing values, into all data partitions, at

Table 5.2: Hyperparameter search space explored during tuning. The search is performed with Optuna (TPE sampler, 100 trials per head), maximizing the Antolini concordance index on the held-out validation set.

Survival head	Hyperparameter	Search space
DeepSurv (MLP, tuned)	`num_nodes`	{[32], [64], [128], [64, 32], [128, 64]}
	`dropout`	[0.0, 0.3]
	learning rate (log)	$[10^{-4}, 10^{-2}]$
	`batch_norm`	{True, False}
Random Survival Forest (tuned)	`n_estimators` (step 50)	[50, 300]
	`max_depth`	[3, 10]
	`min_samples_split`	[5, 20]
	`min_samples_leaf`	[3, 15]
	`max_features`	{sqrt, log2}

Table 5.3: Training settings used during the tuning of the DeepSurv head.

Phase	Setting	Value
Search (per trial)	epochs	100
	batch size	256
	early stopping	none
Final training	max epochs	200
	batch size	128
	early stopping	patience 20, `min_delta` 10^{-4}
	validation	validation set

increasing rates; there, the embedding-based approach is compared against five conventional imputation methods applied to the raw features, which play the role of the baselines. These differences are detailed in the corresponding section.

Table 5.4: Summary of the experimental design. For each experiment, the table reports the survival heads involved, whether baseline models and hyperparameter tuning are included, and the missing-data variants under which the experiment is carried out. Both foundation models (TabPFN and TabICL) are used as backbones in every experiment. "Fixed heads" denotes Cox, Random Survival Forest, DeepSurv (MLP) and DeepSurv (linear) in their fixed configuration.

Experiment	Survival heads	Baselines	Tuning	Missing-data variants
Exp. 1 (Model comparison)	All six heads	Yes (in -1 only)	Yes	NaN and -1
Exp. 2 (Training set size)	Fixed heads	Yes (in -1 only)	No	NaN and -1
Exp. 3 (Missing-value robustness)	Fixed heads	Imputation methods	No	-1 base, NaN injection (all splits)

5.4 Experiment 1: Model Comparison

5.4.1 Objective

The first and main experiment compares the different configurations of the framework in order to assess the central question of this work: whether using tabular foundation models as feature extractors is beneficial for survival analysis. The comparison is articulated along two axes. The first is the comparison between the embedding-based configurations and the baseline models that operate directly on the raw features, which serves to determine whether the representations produced by the foundation models lead to better survival prediction than the raw data. The second is the comparison among the different combinations of backbone and survival head, which makes it possible to observe how the two foundation models behave relative to one another and whether the benefit of the embeddings depends on the survival head they are paired with.

5.4.2 Models and Configurations

This experiment uses the full set of models described in the common framework. All six survival heads are evaluated - Cox, Random Survival Forest in its fixed and tuned variants, and DeepSurv in its fixed multi-layer, tuned multi-layer, and linear variants - each applied to the embeddings produced by the two backbones, and accompanied by the corresponding baseline models. It is the only experiment in which hyperparameter tuning is applied and it is carried out under both missing-data variants, with the baseline models evaluated only in the variant where missing values are replaced by -1. The evaluation follows the protocol described in Section 5.2.6, and the configurations and search spaces are those reported in the corresponding configuration tables. A summary of the models involved is given in Table 5.4.

5.4.3 Feature Importance via SHAP

In addition to the predictive comparison, this experiment includes an analysis of feature importance based on SHAP (SHapley Additive exPlanations) values, in order to assess which input variables contribute most to the risk predicted by each model. The quantity being explained is the risk score produced by each survival head: the log-partial hazard for the DeepSurv heads, the predicted risk score - corresponding to the average cumulative hazard - for the Random Survival Forest, and the partial hazard for the Cox model. The SHAP values therefore quantify the contribution of each variable to the variation of this risk score.

A central aspect of this analysis is that the SHAP values are always computed with respect to the original features of the dataset, for every model, so that the resulting importances are expressed in terms of interpretable clinical variables rather than the abstract dimensions of the embeddings. For the embedding-based configurations, this is achieved by treating the entire pipeline as a single black-box function: the function receives the original features as input, internally performs the embedding-extraction step, and

returns the risk score. SHAP thus measures how the risk score changes when an original feature is perturbed, while the dependence on the embeddings is absorbed into the black box and never exposed in the attributions. The baseline models follow the same logic, without the embedding step.

A single, model-agnostic explainer (Kernel SHAP) is used for all models, so that the importances are obtained in a consistent way regardless of the underlying survival head. For each fold, a set of up to 100 training samples is used as the background distribution, and the explanations are computed on up to 100 samples drawn from the test partition of that fold. The mean absolute SHAP value of each feature across the explained samples is retained as its importance for that fold; these per-fold importances are averaged across folds, and the resulting values are then averaged across the five seeds, consistently with the predictive evaluation. As with the predictive comparison, the feature-importance analysis is carried out under both missing-data variants.

5.5 Experiment 2: Effect of Training Set Size

5.5.1 Objective

The second experiment investigates how the predictive performance of the framework depends on the amount of data available for training. The motivation is twofold. On the one hand, it characterizes the data efficiency of the embedding-based configurations, that is, how much training data they require to reach a given level of performance. On the other hand, it is particularly relevant for the foundation models, since reducing the training set also reduces the context that the backbones rely on to produce their embeddings: this experiment therefore probes how the quality of the embeddings, and ultimately of the survival predictions, degrades as the in-context information becomes scarcer. The behavior of the embedding-based configurations and of the baselines is compared as the training set size varies, to observe whether the foundation models retain an advantage in the low-data regime.

5.5.2 Models and Configurations

The experiment is run by progressively varying the size of the training set, using a predefined set of target sizes expressed as absolute numbers of training samples within each fold: 100, 500, 1000, 2500, 5000, 10000, and 15000. Each target size acts as an upper bound: when it exceeds the number of training samples available in a fold, the effective size is capped at the size of the fold's training partition, and target sizes that would result in a duplicate effective size are discarded. For each fold, the training subsets are obtained by drawing a single random permutation of the fold's training samples and taking its first elements, so that the subsets are nested - each smaller subset is contained in the larger ones. The sampling is a simple random sampling, not stratified on the event indicator, and it depends on the seed, so that different seeds yield different subsets.

A key point is that everything other than the training subset is kept fixed as the size varies. The test partition of each fold and the held-out validation set are never subsampled: the models are always evaluated on the same data, and only the quantity of data used for training changes. Moreover, for the embedding-based configurations, the reduced training subset also serves as the context from which the foundation models extract the embeddings of the training, validation, and test samples; consequently, varying the training size changes not only the data seen by the survival head, but also the in-context information available to the backbone, and therefore the embeddings themselves.

Consistently with the previous experiment, the comparison is carried out under both missing-data variants, with the baseline models evaluated only in the variant where missing values are replaced by -1. The survival heads are restricted to their fixed-configuration variants - Cox, the fixed Random Survival Forest, the fixed multi-layer DeepSurv, and the linear DeepSurv - together with the corresponding baselines; the tuned variants are not included. This choice is primarily methodological: if the hyperparameters were re-selected at each training size, the differences in performance could no longer

be attributed to the amount of data alone, since they would be confounded with the effect of the chosen hyperparameters. Keeping the head configurations fixed isolates the effect of the training set size, which is the object of this experiment. Excluding the tuning also keeps the computational cost of the experiment manageable.

5.6 Experiment 3: Robustness to Missing Values

5.6.1 Objective

The third experiment evaluates how robust the framework is to the presence of missing values, and more specifically, whether using the foundation models to deal with missingness is advantageous compared to traditional imputation strategies. Synthetic missing values are introduced at increasing rates, and the embedding-based approach is compared, on the same downstream models and under the same conditions, against a set of conventional imputation methods applied to the original features. The experiment thus measures not only how performance degrades as the data becomes more incomplete, but also how the foundation-model representations compare with explicit imputation as the amount of missingness grows. Because the missing values are injected into all data partitions, the experiment jointly assesses the robustness of training on incomplete data and that of predicting on incomplete inputs.

5.6.2 Missing Value Injection Procedure

This experiment is based on the variant in which the original missing values are replaced by -1, and it introduces synthetic missing values on top of it. The procedure starts from the dataset in which any pre-existing missing entry has already been encoded as -1, across the whole matrix, before any split is performed. The cross-validation split is then carried out on this

dataset, and only afterwards are the synthetic missing values injected into each fold. In this way the original missingness is neutrally encoded as -1 and kept separate from the synthetic missing values, which are the actual object of the experiment.

The synthetic missing values are injected at five increasing levels, corresponding to 10%, 30%, 50%, 70% and 90% of the data. The percentage refers to the total number of cells of the feature matrix - the entries obtained by crossing rows and features - rather than to individual rows or columns: the corresponding number of cells is selected uniformly at random across the entire matrix, with no balancing constraint per row or per feature, and all features are eligible. For a given partition, the injection is nested across levels: the set of cells made missing at a lower percentage is contained in the set made missing at a higher one, since all levels draw the leading cells of a single random ordering. The injection is applied independently to the training, test, and validation partitions, each with its own ordering, and it depends on the seed, so that the positions of the synthetic missing values differ across the five seeds.

5.6.3 Compared Approaches

At each missingness level, two families of approaches for dealing with the injected missing values are compared on the same downstream survival models. The first is the embedding-based approach: the data, containing the genuine synthetic missing values, is passed to the foundation models, which handle the missingness through their native mechanisms and produce the embeddings. The second is explicit imputation of the original features, considered through five methods: mean imputation, median imputation, constant imputation with the value -1, k-nearest-neighbors imputation, and an iterative model-based imputation. Each imputer is fitted on the training data and applied to the validation and test data. The constant imputation with -1 is a particular case, as it assigns to the synthetic missing values the same encoding already used for the original missing entries, thereby treating

both kinds of missingness uniformly.

The same four fixed-configuration survival models are then trained and evaluated on the output of each approach: the Cox model, the fixed Random Survival Forest, the linear DeepSurv, and the multi-layer DeepSurv. For the embedding-based approach these models operate on the embeddings, whereas for the five imputation methods they operate on the imputed original features, which therefore act as the baselines against which the embedding-based approach is assessed. This design makes it possible to compare, at every missingness level and for every downstream model, the representations produced by the foundation models against conventional imputation of the raw data.

For the two DeepSurv models, the number of training epochs is not tuned separately for each approach. Instead, it is calibrated once per fold and missingness level, using mean imputation as a pilot case: a DeepSurv model is trained on the mean-imputed data with early stopping, and the resulting number of epochs is then reused for all the other approaches that require it, at the same fold and missingness level. Mean imputation was chosen as the reference case on which to base this calibration. This keeps the training procedure uniform across approaches and avoids a separate epoch search for each of them, at the cost of applying an epoch count calibrated on a single approach to all the others.

5.7 Experimental Environment and Reproducibility

This section reports the computational environment in which the experiments were run and the measures adopted to ensure their reproducibility.

Hardware. The experiments were carried out on a high-performance computing (HPC) cluster, using NVIDIA GPUs - either an RTX 2080 or an L40, depending on availability. The GPUs are used for the foundation models, which produce the embeddings, and for training the neural survival

Table 5.5: Software environment: main libraries and their versions. TabICL is used as local copy of its original code; for it, the pretrained checkpoint used is reported instead of a package version.

Component	Version / checkpoint
Python	3.12.12
TabPFN	6.4.1
TabICL (checkpoint)	`tabicl-classifier-v1.1-0506.ckpt`
PyTorch	2.10.0
pycox	0.3.0
torchtuples	0.2.2
scikit-survival	0.27.0
lifelines	0.30.0
scikit-learn	1.8.0
Optuna	4.8.0
SHAP	0.51.0
NumPy	2.4.3
pandas	3.0.1
SciPy	1.16.3

heads.

Software environment. The framework is implemented in Python 3.12. The TabPFN backbone is used through its official library, whereas TabICL is included as a local copy of its original code, adapted to extract the embeddings as described in Section 5.2.3. The survival heads rely on established survival-analysis libraries: lifelines for the Cox model, scikit-survival for the Random Survival Forest, and pycox (built on PyTorch) for the DeepSurv models. The imputation methods used in the third experiment are those provided by scikit-learn, hyperparameter tuning is performed with Optuna, and the feature-importance analysis uses the SHAP library. The exact versions of all libraries, together with the pretrained checkpoint used for TabICL, are reported in Table 5.5.

Randomness control. To make the experiments reproducible, all the sources of randomness are seeded through a single routine that sets the seeds of the Python, NumPy, and PyTorch random number generators, on both

CPU and GPU, and configures the cuDNN backend in deterministic mode. The whole evaluation is repeated over five fixed seeds (42,77,123,456,789). Within each run, the seed used for each model is offset by the fold index and by a model-specific offset, so that different folds start from different initializations and the models within a fold are made independent of one another, rather than sharing a random state that depends on the order in which they are executed. The Random Survival Forest is an exception: its internal randomness is controlled by a fixed random state shared across folds, so that the variability between folds reflects only the difference in the training data and not the initialization of the forest.

Persistence and reuse. To avoid recomputation and to guarantee that the same artifacts are reused across configurations, several intermediate results are persisted to disk and reloaded when available. These include the cross-validation fold partitions, the orderings used to inject the synthetic missing values in the third experiment, the fitted imputers, the model checkpoints, and the computed SHAP values. This ensures, for instance, that all configurations share exactly the same folds and the same missing-value masks, so that they are compared on identical data.

Code and data availability. The code that implements the framework and the experiments is publicly available in the repository referenced in the introduction. The two datasets, on the other hand, are private and were made available only for the purposes of this project, as detailed in Chapter 4; they cannot therefore be shared publicly. This places a limitation on the full reproducibility of the results by third parties, although the complete experimental pipeline is available and can be applied to other survival datasets.

Chapter 6

Results

6.1 Overview

This chapter presents the experimental results obtained with the framework described in the previous chapter. The results are organized around the three experiments introduced there: the comparison between the embedding-based configurations and the baseline models, together with the associated feature-importance analysis (Section 5.4); the study of how predictive performance depends on the size of the training set (Section 5.5); and the assessment of the robustness of the framework to increasing amounts of missing data (Section 5.6). Throughout the chapter, predictive performance is measured by the Antolini time-dependent concordance index defined in Chapter 5, and, unless otherwise specified, results are reported for both datasets, IHD and URRAH, and under both missing-data variants. A general overview of the findings across the three experiments, together with the conclusions that can be drawn from them, is provided in the concluding chapter.

6.2 Experiment 1: Model Comparison

This section reports the results of the main experiment, which compares the different configurations of the framework in terms of the Antolini con-

cordance index. The comparison addresses the central question of this work: whether using the tabular foundation models as feature extractors yields better survival prediction than applying the survival heads directly to the raw features. Two axes of comparison are considered: the comparison between the embedding-based configurations and the corresponding baselines, and the comparison between the two backbones, TabPFN and TabICL, across the different survival heads. Results are reported separately for the two datasets and, for each of them, under both the NaN and the -1 missing-data variants. All values are reported as the mean concordance index over the five seeds, together with the corresponding standard deviation.

The numerical results are collected in Table 6.1 for the IHD dataset and Table 6.2 for the URRAH dataset, which report the concordance index of every survival head, for both backbones and both missing-data variants, together with the baseline models. To complement the tables, Figures 6.1 and 6.2present the same results as bar charts, one figure per dataset. In each figure, the top panel corresponds to the -1 variant – including the embeddings of both backbones and the baselines – and the bottom panel to the NaN variant, with the embeddings only.

6.2.1 Results on the IHD Dataset

Table 6.1 reports the concordance index obtained on the IHD dataset. All configurations achieve a concordance index between roughly 0.74 and 0.80, indicating a good overall discriminative ability. The results are, however, markedly different between the two backbones: the configurations based on TabPFN reach values around 0.79, whereas those based on TabICL are consistently lower, around 0.74 - 0.75, a gap that holds across all survival heads and both missing-data variants.

Comparing the embedding-based configurations with the baselines that operate on the raw features, the baselines attain the highest values overall: the best result on this dataset is obtained by the Random Survival Forest baseline (0.796 ± 0.010), closely followed by the Cox baseline (0.794 ± 0.009).

Table 6.1: Concordance index (mean ± standard deviation over the five seeds) on the IHD dataset for every survival head, under the two backbones (TabPFN and TabICL) and the two missing-data variants (NaN and −1). The baseline models operate on the raw features and are therefore evaluated only in the −1 variant. Best mean value in each column is in bold.

	TabPFN		TabICL	
Survival head	NaN	−1	NaN	−1
Cox	0.793 ± 0.010	0.788 ± 0.012	0.745 ± 0.010	0.745 ± 0.009
RSF (fixed)	0.786 ± 0.012	0.787 ± 0.012	0.746 ± 0.013	0.739 ± 0.012
RSF (tuned)	0.788 ± 0.011	0.790 ± 0.011	0.747 ± 0.011	0.738 ± 0.011
DeepSurv (MLP, fixed)	0.785 ± 0.012	0.784 ± 0.013	0.744 ± 0.011	0.747 ± 0.009
DeepSurv (MLP, tuned)	**0.794 ± 0.011**	0.791 ± 0.010	**0.749 ± 0.011**	0.746 ± 0.010
DeepSurv (linear)	**0.794 ± 0.010**	0.793 ± 0.009	0.738 ± 0.014	0.738 ± 0.013
Baselines (raw features)				
Cox	–	0.794 ± 0.009	–	0.794 ± 0.009
RSF (fixed)	–	**0.796 ± 0.010**	–	**0.796 ± 0.010**
DeepSurv (MLP, fixed)	–	0.791 ± 0.010	–	0.791 ± 0.010
DeepSurv (linear)	–	0.780 ± 0.013	–	0.780 ± 0.013

The TabPFN-based configurations are essentially on par with these baselines. In the NaN variant, the tuned multi-layer DeepSurv and the linear DeepSurv on the TabPFN embeddings both reach 0.794 (respectively ± 0.011 and ± 0.010), and the differences from the baselines are small relative to the standard deviations. The TabICL-based configurations, in contrast, remain below the baselines by a clear margin. On this dataset, therefore, the embeddings do not improve over the raw features, although the TabPFN embeddings essentially match them.

The two missing-data variants yield almost identical results: for every head and both backbones, the NaN and −1 columns differ only at the third decimal and well within the standard deviation, indicating that on IHD the way missing values are handled has essentially no effect on performance. This is consistent with the low overall proportion of missing values in this dataset (around 1%). The same results are shown graphically in Figure 6.1, where the height of each bar corresponds to the mean concordance index of the respective head.

Table 6.2: Concordance index (mean ± standard deviation over the five seeds) on the URRAH dataset for every survival head, under the two backbones (TabPFN and TabICL) and the two missing-data variants (NaN and −1). The baseline models operate on the raw features and are therefore evaluated only in the −1 variant. Best mean value in each column is in bold.

	TabPFN		TabICL	
Survival head	NaN	−1	NaN	−1
Cox	0.841 ± 0.006	0.845 ± 0.006	0.739 ± 0.009	0.769 ± 0.009
RSF (fixed)	0.798 ± 0.018	0.825 ± 0.012	0.722 ± 0.011	0.750 ± 0.011
RSF (tuned)	0.811 ± 0.012	0.829 ± 0.010	0.747 ± 0.010	0.755 ± 0.011
DeepSurv (MLP, fixed)	0.746 ± 0.106	0.786 ± 0.058	0.761 ± 0.019	0.775 ± 0.016
DeepSurv (MLP, tuned)	**0.842 ± 0.010**	0.848 ± 0.006	**0.801 ± 0.009**	0.803 ± 0.008
DeepSurv (linear)	0.841 ± 0.006	0.844 ± 0.007	0.775 ± 0.017	0.794 ± 0.015
Baselines (raw features)				
Cox	–	0.826 ± 0.007	–	0.826 ± 0.007
RSF (fixed)	–	**0.850 ± 0.007**	–	**0.850 ± 0.007**
DeepSurv (MLP, fixed)	–	0.846 ± 0.007	–	0.846 ± 0.007
DeepSurv (linear)	–	0.833 ± 0.008	–	0.833 ± 0.008

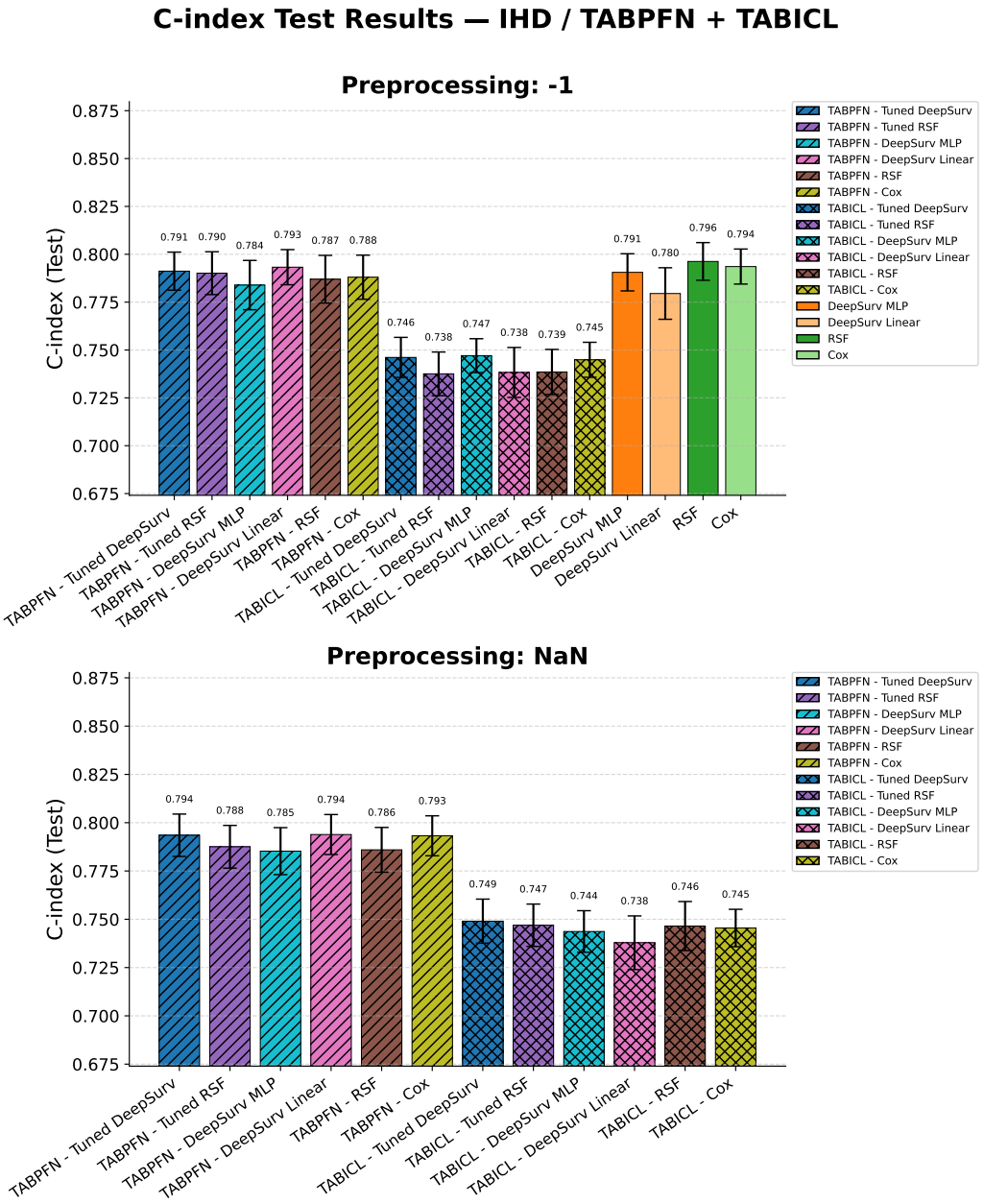

Figure 6.1: Concordance index of the survival heads on the IHD dataset. The top panel shows the -1 variant, including the TabPFN and TabICL embeddings and the baselines; the bottom panel shows the NaN variant, with the TabPFN and TabICL embeddings only. Each bar corresponds to a survival head.

6.2.2 Results on the URRAH Dataset

Table 6.2 reports the corresponding results on the URRAH dataset, which show a wider range of behaviors. As on IHD, the TabPFN-based configurations clearly outperform the TabICL-based ones, but here the best embedding-based results are higher and more competitive with the baselines. The strongest embedding-based configuration is the tuned multi-layer DeepSurv on the TabPFN embeddings, which achieves the highest value among the embedding-based models in both variants (0.842 ± 0.010 in the NaN variant and 0.848 ± 0.006 in the -1 variant); the Cox head and the linear DeepSurv on the same embeddings also reach about 0.84. Among the baselines, the highest value is again attained by the Random Survival Forest (0.850 ± 0.007), followed by the multi-layer DeepSurv baseline (0.846 ± 0.007).

As on IHD, the best embedding-based configurations are essentially on par with the best baseline, with differences well within the standard deviations. Unlike on IHD, however, several TabPFN configurations here exceed some of the baselines: the tuned multi-layer DeepSurv, the Cox head and the linear DeepSurv on the TabPFN embeddings all surpass the Cox baseline (0.826 ± 0.007) and the linear DeepSurv baseline (0.833 ± 0.008), even if none of them reaches the Random Survival Forest baseline. The TabICL-based configurations remain the lowest, although the gap with respect to the baselines is smaller than the one observed on IHD.

Two further aspects are worth noting. First, unlike on IHD, the missing-data variant has a visible effect on URRAH: for most heads the -1 variant yields a slightly higher index than the NaN variant, and this difference is more pronounced for the TabICL backbone - for example, the Cox head on the TabICL embeddings goes from 0.739 ± 0.009 in the NaN variant to 0.769 ± 0.009 in the -1 variant. This larger sensitivity is consistent with the much higher proportion of missing values in URRAH (close to 40%). Second, one configuration stands out for its instability: the fixed multi-layer DeepSurv on the TabPFN embeddings in the NaN variant has a very large standard deviation (0.746 ± 0.106), far above that of any other configuration,

indicating that this particular setting is highly sensitive to the seed. The corresponding bar chart is shown in Figure 6.2.

Comparing the two datasets, some patterns are consistent - TabPFN always outperforms TabICL, and the best baseline are never clearly surpassed - while others differ: on URRAH several embedding-based configurations exceed some of the baselines, which does not happen on IHD, and the effect of the missing-data variant, negligible on IHD, becomes appreciable on URRAH. These cross-dataset observations are taken up in the summary at the end of the section.

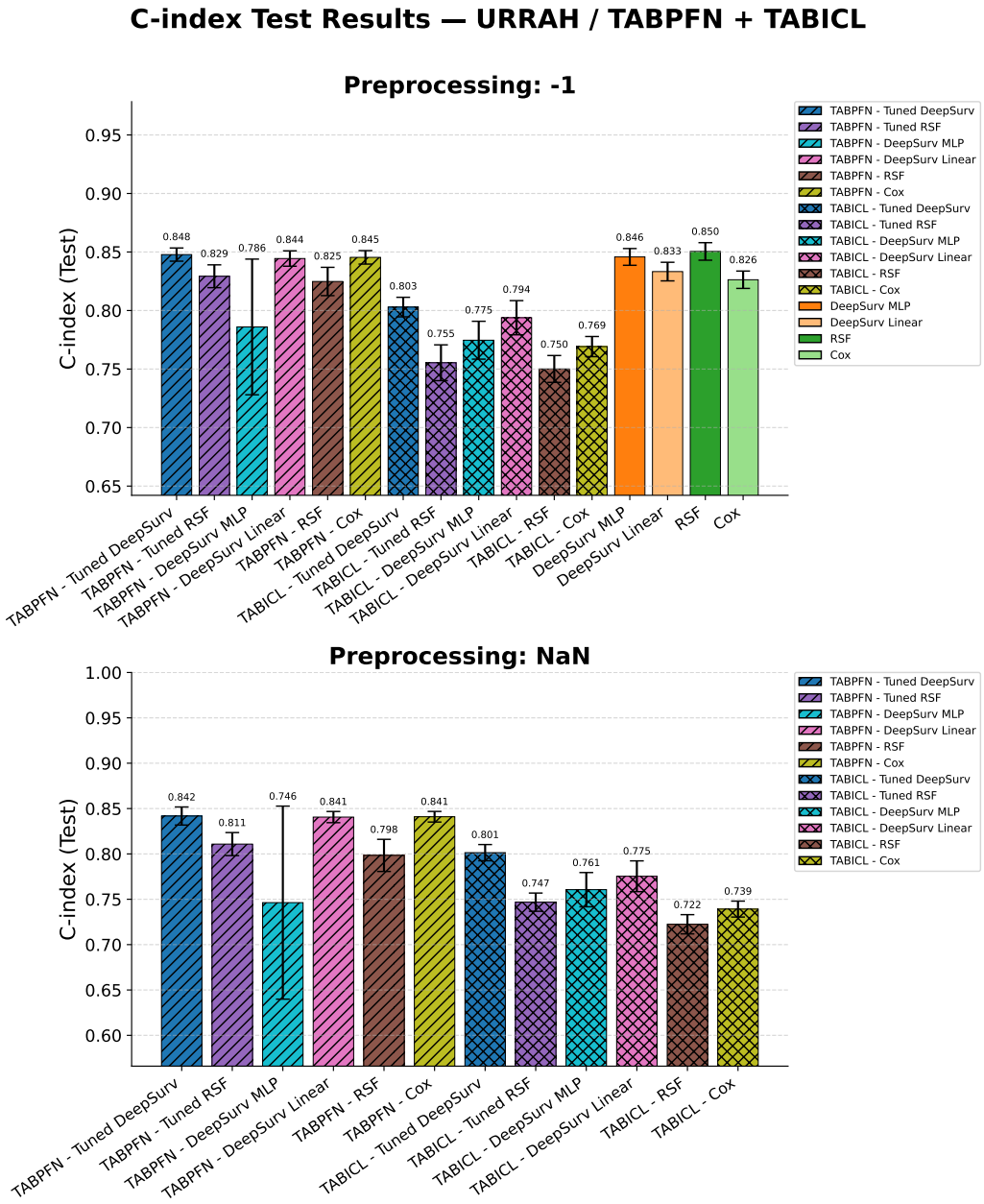

Figure 6.2: Concordance index of the survival heads on the URRAH dataset. The top panel shows the -1 variant, including the TabPFN and TabICL embeddings and the baselines; the bottom panel shows the NaN variant, with the TabPFN and TabICL embeddings only. Each bar corresponds to a survival head.

6.2.3 Feature Importance via SHAP

Beyond predictive performance, this experiment includes a feature-importance analysis based on SHAP values, aimed at identifying which clinical variables contribute most to the risk predicted by each configuration. As described in Chapter 5, the SHAP values are computed with respect to the original clinical features for every model, including the embedding-based ones, so that the resulting importances are directly interpretable. The analysis is carried out for both backbones, both datasets, and both missing-data variants.

Before describing the results, one aspect should be kept in mind: the raw magnitude of the SHAP values is not directly comparable across survival heads, since each head produces a risk score on a different scale (the partial hazard of the Cox model, the average cumulative hazard for the Random Survival Forest, and the log-partial hazard for the DeepSurv heads). For this reason, the analysis focuses on the relative ranking of the features within each model. In the heatmaps discussed below, the values are normalized per model, which removes the scale differences and makes the importances visually comparable across heads.

Consistency across heads, difference across backbones. The most evident pattern is that the feature importances are largely determined by the backbone rather than by the survival head. For a given backbone, the different heads (Cox, Random Survival Forest, and DeepSurv in its variants) agree closely on which features are most important, producing very similar rankings. The two backbones, however, lead to markedly different importances. This holds on both datasets and under both missing-data variants.

With the TabPFN embeddings, the predictions are dominated by age (`Age` on IHD, `ETA` on URRAH), which ranks first by a large margin across all heads, followed by a small set of clinical variables such as creatinine (`Creatinina`), ejection fraction (`fe`) and lipid-lowering therapy (`Ipolipemizzanti`) on IHD, and azotemia (`AZOTEMIA`), sex (`SESSO`) and a few medications on URRAH. Notably, the baseline models, which operate on the raw features, exhibit a very similar ranking, with age again in first position. This suggests that the

TabPFN embeddings preserve the importance structure of the raw data.

With the TabICL embeddings, the picture is different: age loses its dominant role - on URRAH it drops to the very bottom of the ranking for some heads - and the most important features become a different set of clinical variables, such as diuretics (`Diuretics`), angiography (`Angiography`) and the number of affected vessels (`Vessels`) on IHD, or allopurinol (`ALLOPURINOLO`) and several therapies and markers on URRAH. The importance is therefore redistributed over a different group of features compared with both TabPFN and the baselines.

Importance heatmaps. Figures 6.3 and 6.4 show the feature-importance heatmaps for the -1 variant, in which each row corresponds to a feature and each column to a model – the survival heads on the TabPFN and TabICL embeddings, together with the baselines – with the color encoding the per-model normalized mean absolute SHAP value from low (yellow) to high (red). The -1 variant is shown here because it also includes the baseline models, allowing the embedding-based configurations and the baselines to be compared directly. The heatmaps make the consistency across heads immediately visible: for a given backbone, the rows that stand out are largely the same across all columns, indicating that the different heads rely on a common set of features. The contrast between the two backbones is equally apparent, as the block of dominant features differs between the TabPFN and TabICL heatmaps, in line with the shift described above. The corresponding top-feature bar plots, which report the twenty most important features for each configuration together with their standard deviation across seeds, are provided in Appendix B for both datasets and both missing-data variants, together with the heatmaps for the NaN variant.

A comparison between the two missing-data variants shows that the stability of the feature rankings depends on the dataset. on IHD, the rankings under the -1 and NaN variants are almost identical, with nearly the same features appearing in the same order for every model; on URRAH, in contrast, the rankings change appreaciably between the two variants, with a

smaller overlap among the top features. This is consistent with the different amount of missing data in the two datasets: where missing values are rare (IHD), the way they are handled has little effect on the importances, whereas where they are frequent (URRAH), it noticeably alters which features the models rely on. Even on URRAH, however, the leading feature identified by the TabPFN embeddings (age) remains at the top under both variants, while for the TabICL embeddings the change is more pronounced.

Summary. Overall, the SHAP analysis shows that the feature importances are driven mainly by the choice of backbone: the different survival heads built on the same embeddings agree closely, whereas the two backbones lead to distinct sets of influential features. The TabPFN embeddings, like the raw-feature baselines, concentrate the importance on age, while the TabICL embeddings distribute it over a different group of clinical variables. The features that emerge are, in both cases, clinically plausible. Why the two backbones give rise to such different importance structures is not investigated here and represents a natural direction for future work, together with a more detailed clinical interpretation of the identified variables.

6.3 Experiment 2: Effect of the Training Set Size

This section studies how the predictive performance of the different configurations depends on the amount of training data. The models are trained on training subsets of increasing size, starting from a small number of samples and progressively adding more, and the resulting concordance index is tracked as a function of the training set size. The goal is to assess whether the tabular foundation models, through their pretraining, offer an advantage when only a small number of samples is available, and how this advantage evolves as more data is added. As in the previous experiment, the analysis covers both datasets, both backbones, and both missing-data variants, with the baselines included in the -1 variant; the tuned configurations are

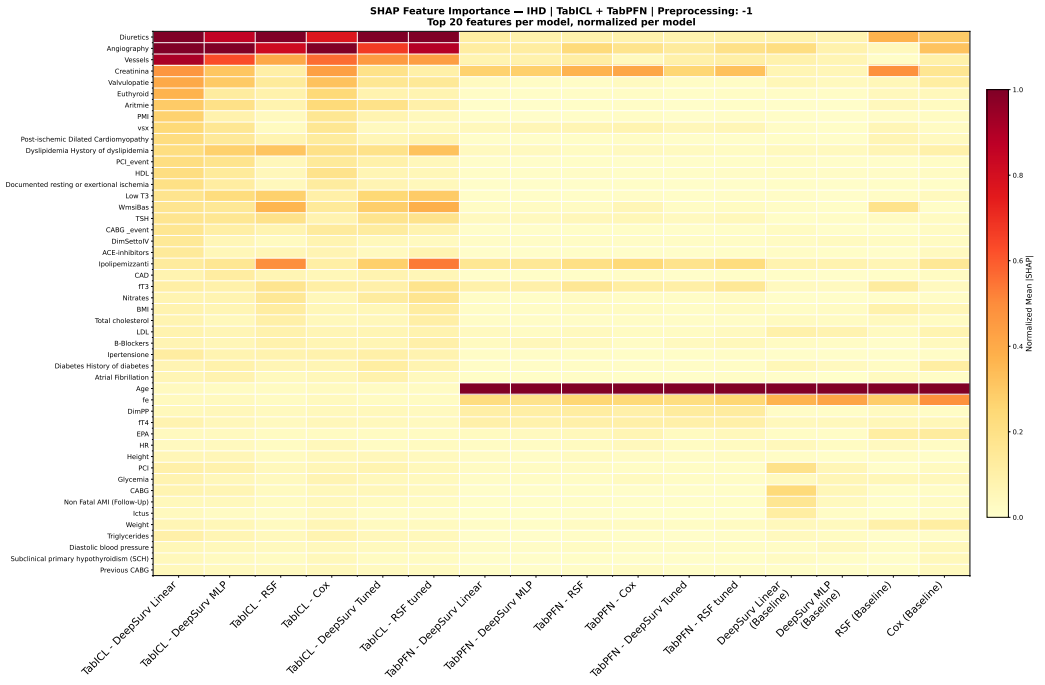

Figure 6.3: Feature-importance heatmap (per-model normalized mean absolute SHAP value) on the IHD dataset (-1 variant). Each row corresponds to a feature and each column to a model, including the survival heads on the TabPFN and TabICL embeddings and the baselines. Colors range from low (yellow) to high (red).

excluded, for the reasons discussed in Chapter 5.

The results are presented as performance curves in Figures 6.5–6.8, one figure per dataset and backbone, each showing the -1 variant on the left and the NaN variant on the right. In every plot, the horizontal axis reports the training set size and the vertical axis the concordance index, with one curve per survival head and, in the -1 variant, the baselines. The complete numerical values are reported in Appendix C. It should be noted that the two datasets span different ranges of training set size: the IHD dataset, being smaller, only reaches about 5000 samples, whereas the URRAH dataset extends up to 13000.

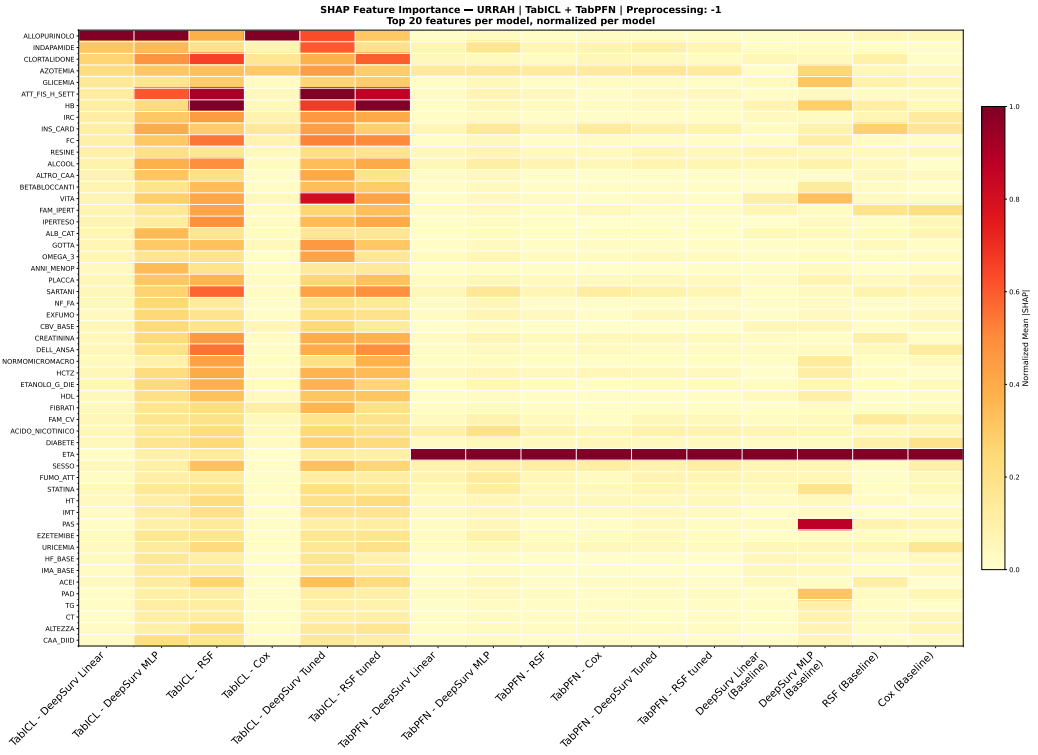

Figure 6.4: Feature-importance heatmap (per-model normalized mean absolute SHAP value) on the URRAH dataset (−1 variant). Each row corresponds to a feature and each column to a model, including the survival heads on the TabPFN and TabICL embeddings and the baselines. Colors range from low (yellow) to high (red).

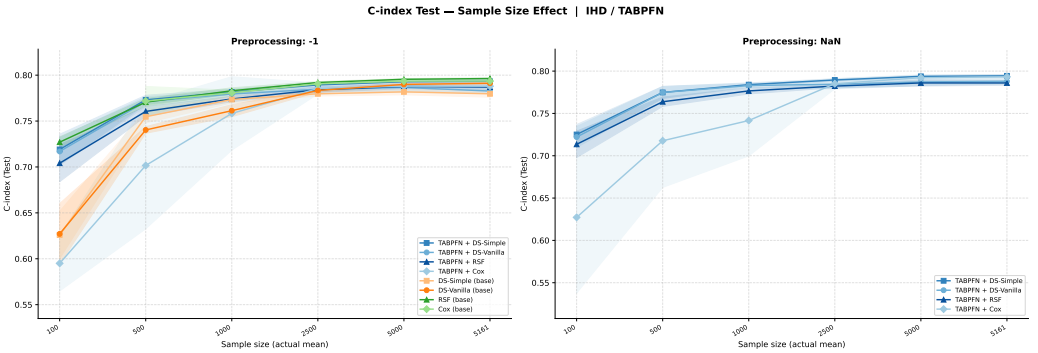

Figure 6.5: Concordance index as a function of the training set size on the IHD dataset with the TabPFN backbone, for the −1 variant (left) and the NaN variant (right). Each curve corresponds to a survival head; the shaded areas denote the standard deviation across the five seeds.

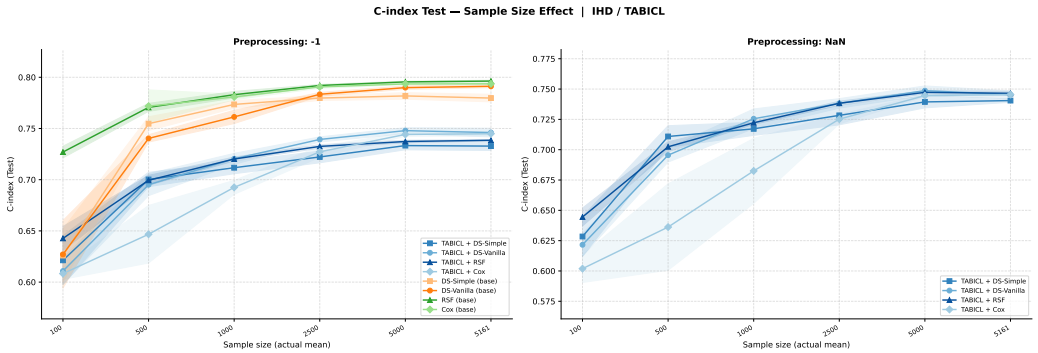

Figure 6.6: Concordance index as a function of the training set size on the IHD dataset with the TabICL backbone, for the −1 variant (left) and the NaN variant (right). Each curve corresponds to a survival head; the shaded areas denote the standard deviation across the five seeds.

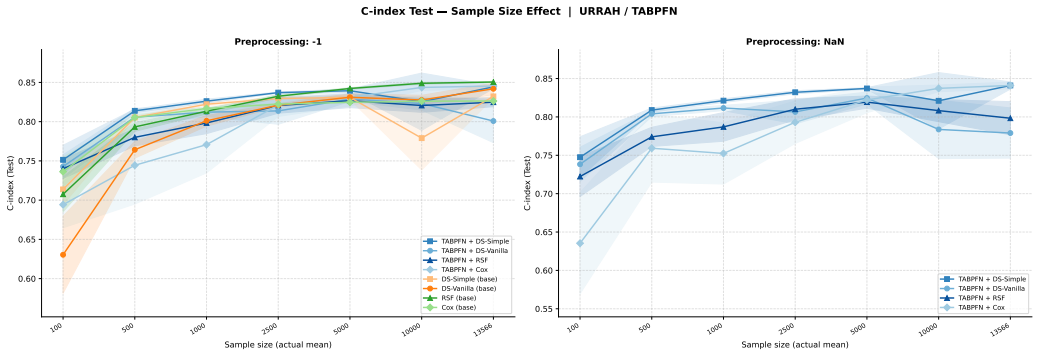

Figure 6.7: Concordance index as a function of the training set size on the URRAH dataset with the TabPFN backbone, for the −1 variant (left) and the NaN variant (right). Each curve corresponds to a survival head; the shaded areas denote the standard deviation across the five seeds.

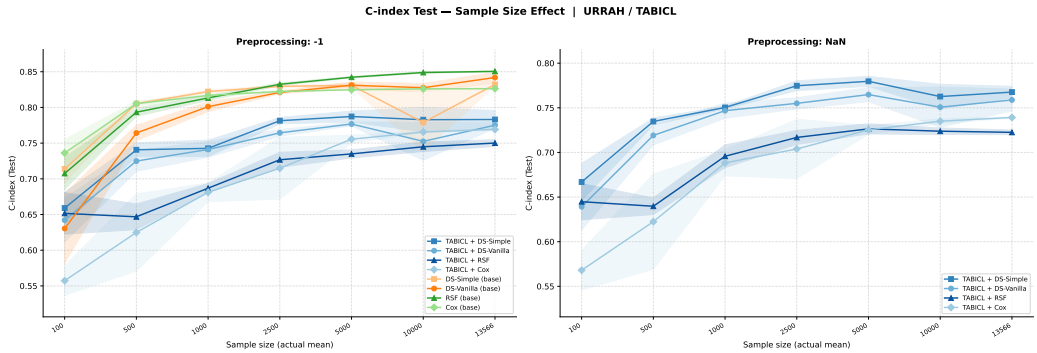

Figure 6.8: Concordance index as a function of the training set size on the URRAH dataset with the TabICL backbone, for the -1 variant (left) and the NaN variant (right). Each curve corresponds to a survival head; the shaded areas denote the standard deviation across the five seeds.

6.3.1 TabPFN

The clearest pattern emerges with the TabPFN backbone. On both datasets, the embedding-based configurations already reach a high concordance index with very few training samples: with only 100 samples, the Cox and DeepSurv heads on the TabPFN embeddings attain a concordance index around 0.76 on IHD and around 0.80 on URRAH. The baselines, in contrast, start from clearly lower values at the same size - around 0.70 - 0.71 on IHD - and improve as the training set grows. As the number of samples increase, the baselines progressively close the gap, and at the largest sizes the embedding-based configurations and the baselines become comparable, consistently with the results of the first experiment at full training size.

This behavior is the one expected from a tabular foundation model: the knowledge acquired during pretraining provides a substantial advantage in the low-data regime, while its benefit diminishes as more task-specific data becomes available. The advantage is most pronounced at the smallest sizes and gradually fades along the curve. It can also be observed that, at the smallest sizes, the curves - particularly those of the DeepSurv heads - are affected by a large standard deviation, indicating that training on very few samples is not only less accurate on average but also less stable.

6.3.2 TabICL

The TabICL backbone behaves differently. The low-data advantage observed for TabPFN is here much weaker, and in several cases absent: the TabICL-based configurations start from lower values at small sizes and remain close to, or below the baselines, throughout most of the range. The overall level of the curves is also lower than for TabPFN, in line with the results of the first experiment. As with TabPFN, the curves rise and tend to converge as the training set grows.

This difference between the two backbones is consistent with their design: TabPFN is specifically intended for prediction on small tabular datasets,

which matches the strong low-data behavior observed here, whereas TabICL is designed to scale to large datasets and does not exhibit the same advantage when only a few samples are available. A more detailed investigation of the reasons behind this difference is left as a direction for future work.

6.3.3 Summary

Overall, this experiment shows that the benefit of using the tabular foundation models as feature extractors depends stronlgy on the amount of available data and on the backbone. With TabPFN, the embeddings provide a clear advantage in the low-data regime, where they outperform the baselines by a wide margin, an advantage that gradually vanishes as the training set grows until the two become comparable. With TabICL, this advantage is largely absent. In all cases, the different configurations tend to converge as more data is added, and the results at the smallest sizes are characterized by a higher variability across seeds. These observations describe the behavior found in the results; a deeper analysis of its causes is left for future work, as discussed in the concluding chapter.

6.4 Experiment 3: Robustness to Missing Values

This section studies how the different approaches behave when the amount of missing data increases. Starting from the original data, missing values are artificially injected at increasing rates - 10%, 30%, 50%, 70% and 90% of the cells - and the concordance index is tracked as a function of the injected missing rate. The goal is to compare two ways of dealing with missing values: passing the incomplete data to the tabular foundation models, which produce the embeddings, against imputing the missing entries with conventional techniques before applying the survival heads. Five imputation methods are considered as a comparison: constant imputation (replacing the missing entries

with -1), mean, median, k-nearest-neighbors, and an iterative (Bayesian) imputer. The analysis is carried out on both datasets, for both backbones and for the four fixed survival heads (Cox, Random Survival Forest, and the multi-layer and linear DeepSurv).

The results are presented as performance curves. For each dataset and backbone, Figures 6.9–6.12 compare the imputation methods, with one panel per survival head; the curves for the embedding-based approach are compared across backbones and datasets in Figure 6.13. In all plots the horizontal axis reports the injected missing rate and the vertical axis the concordance index on the test folds. The complete numerical values are reported in Appendix D.

6.4.1 Comparison of the Imputation Methods

Considering first the imputation methods, all of them lose predictive power as the missing rate increases, but they do so at different speeds. The constant imputation, which replaces the missing entries with -1, is almost always the weakest: on IHD with the Cox head, for instance, it drops from 0.731 at a missing rate of 10% to 0.525 at 90%, consistently below the other methods. The more informed imputation methods degrade more gracefully. The iterative (Bayesian) imputer is generally the most robust, often achieving the highest concordance among the imputation methods at high missing rates, closely followed by the mean and median imputers, while k-nearest-neighbors tends to fall in between the constant and the mean-based methods. This ordering is consistent across both datasets and both backbones: the more information an imputation method uses to reconstruct the missing entries, the better it preserves the discriminative ability of the survival head as missingness grows.

6.4.2 Embeddings versus Imputation

The embedding-based approach behaves differently depending on the backbone. With TabPFN, the embeddings are competitive at low missing

rates and, for some heads, the best-performing option: on URRAH with the Cox head, the TabPFN embeddings reach 0.838 at a 10% missing rate, above all the imputation methods, and on IHD they remain close to the best imputers. As the missing rate increases, however, the TabPFN embeddings tend to degrade more rapidly than the informed imputers, so that at the highest missing rates the iterative and mean imputers often match or surpass them. With TabICL, the embeddings are generally weaker than the imputation methods across the whole range: for most heads and missing rates the TabICL embeddings lie below the mean and iterative imputers, in line with the lower performance of TabICL already observed in the previous experiments. Notably, the TabICL embeddings do not coincide with the mean imputation, despite the internal mean imputation that this backbone applies to missing values: they generally perform below it.

6.4.3 Failure of the Multi-Layer DeepSurv on the Embeddings

One combination stands out for its anomalous behavior: the multi-layer DeepSurv head applied to the embeddings. Unlike every other configuration, it yields concordance values close to 0.5 - that is, close to random ranking - across all missing rates and on both datasets, with a large variability across seeds (for example, 0.547 ± 0.089 on IHD and 0.598 ± 0.116 on URRAH at a 10% missing rate with TabPFN). The same head applied to the imputed data does not show this behavior, reaching the values expected for a DeepSurv model. This failure is therefore specific to the pairing of the multi-layer DeepSurv head with the embeddings in this experiment; its cause is not investigated here and is left as a direction for future work.

6.4.4 Summary

Overall, this experiment shows that robustness to missing values depends strongly on how the missing entries are handled. Among the imputation

methods, the more informed ones - the iterative and mean imputers - degrade more gracefully than the constant substitution, which is consistently the weakest. The embedding-based approach is competitive only with the TabPFN backbone and mainly at low missing rates, where it can match or exceed the best imputers, but it loses this advantage as missingness increases; with TabICL, the embeddings remain below the imputation methods throughout. Finally, the multi-layer DeepSurv head fails when applied to the embeddings, yielding near-random performance. As in the previous experiments, these observations describe the behavior found in the results, while a deeper analysis of its causes is left for future work, as discussed in the concluding chapter.

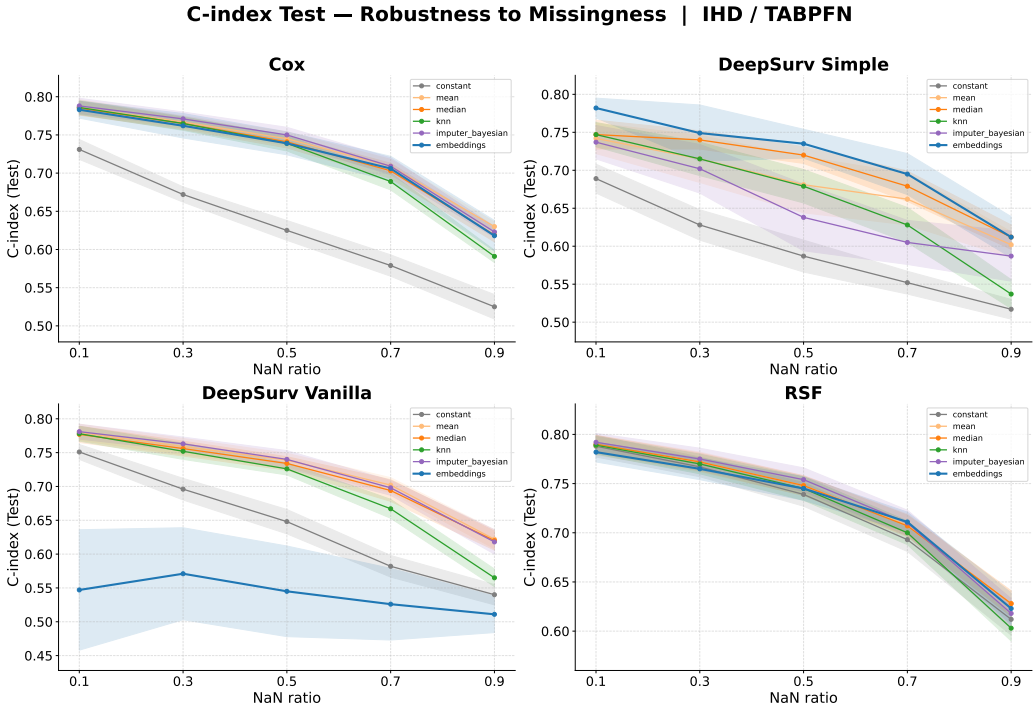

Figure 6.9: Concordance index (test) as a function of the injected missing rate on the IHD dataset with the TabPFN backbone, comparing the imputation methods, with one panel per survival head. The shaded areas denote the standard deviation across the five seeds.

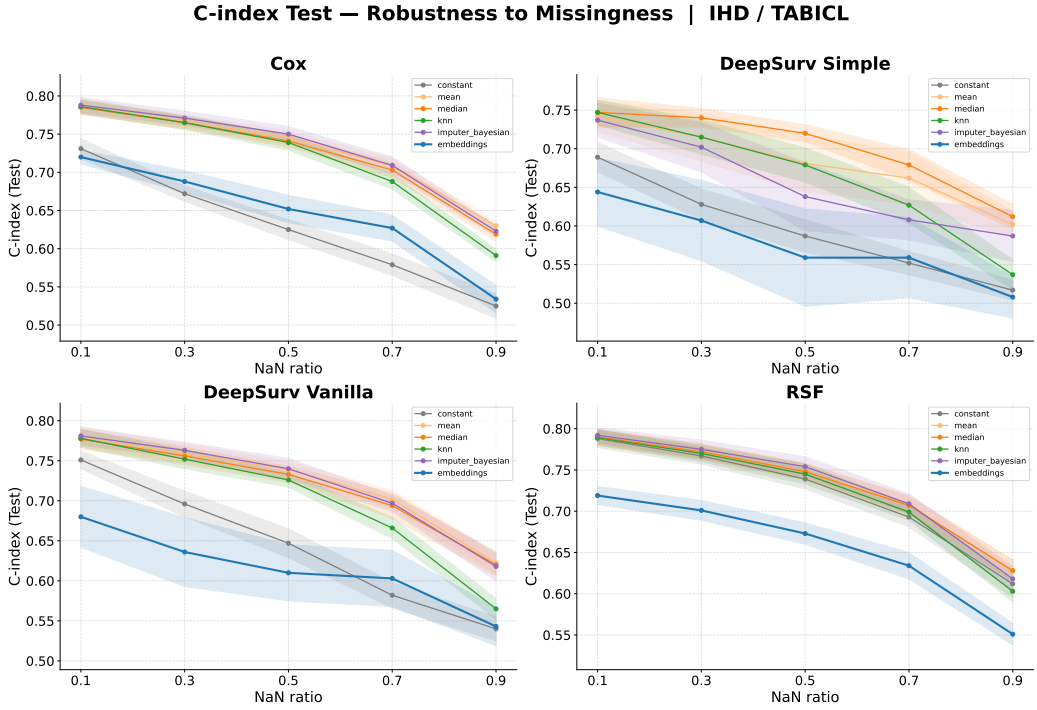

Figure 6.10: Concordance index (test) as a function of the injected missing rate on the IHD dataset with the TabICL backbone, comparing the imputation methods, with one panel per survival head. The shaded areas denote the standard deviation across the five seeds.

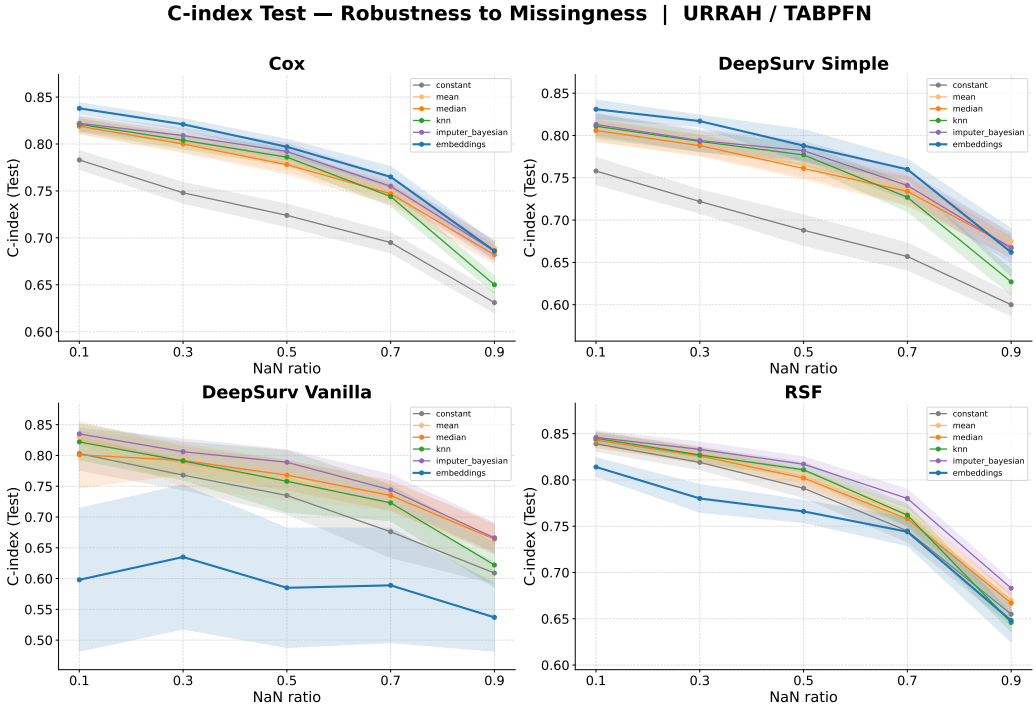

Figure 6.11: Concordance index (test) as a function of the injected missing rate on the URRAH dataset with the TabPFN backbone, comparing the imputation methods, with one panel per survival head. The shaded areas denote the standard deviation across the five seeds.

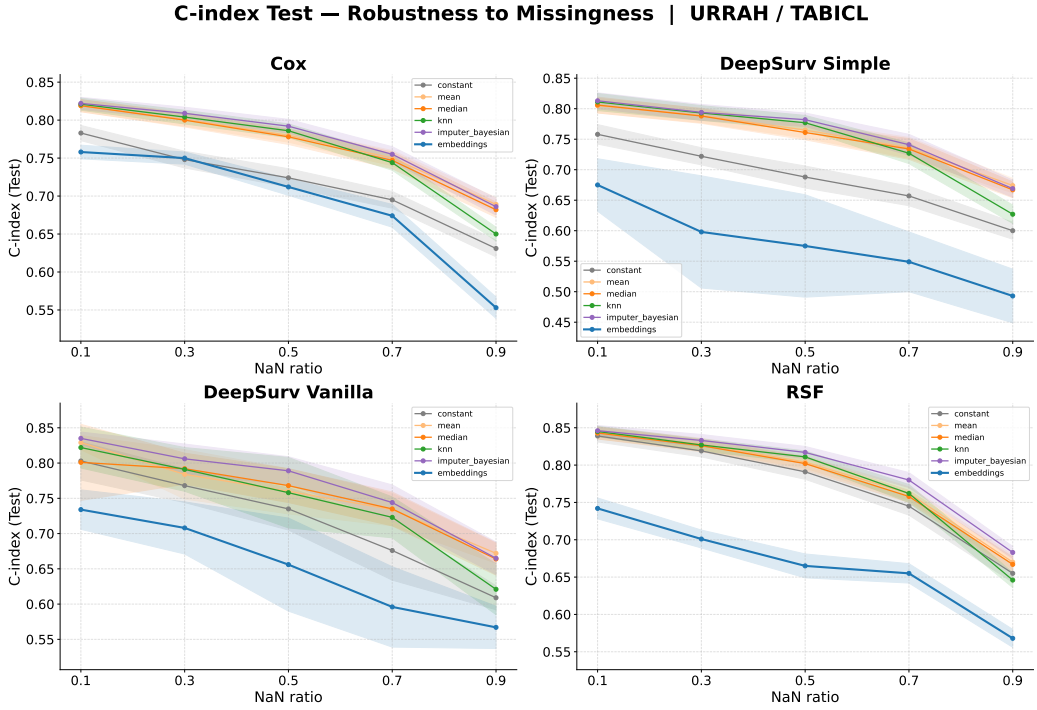

Figure 6.12: Concordance index (test) as a function of the injected missing rate on the URRAH dataset with the TabICL backbone, comparing the imputation methods, with one panel per survival head. The shaded areas denote the standard deviation across the five seeds.

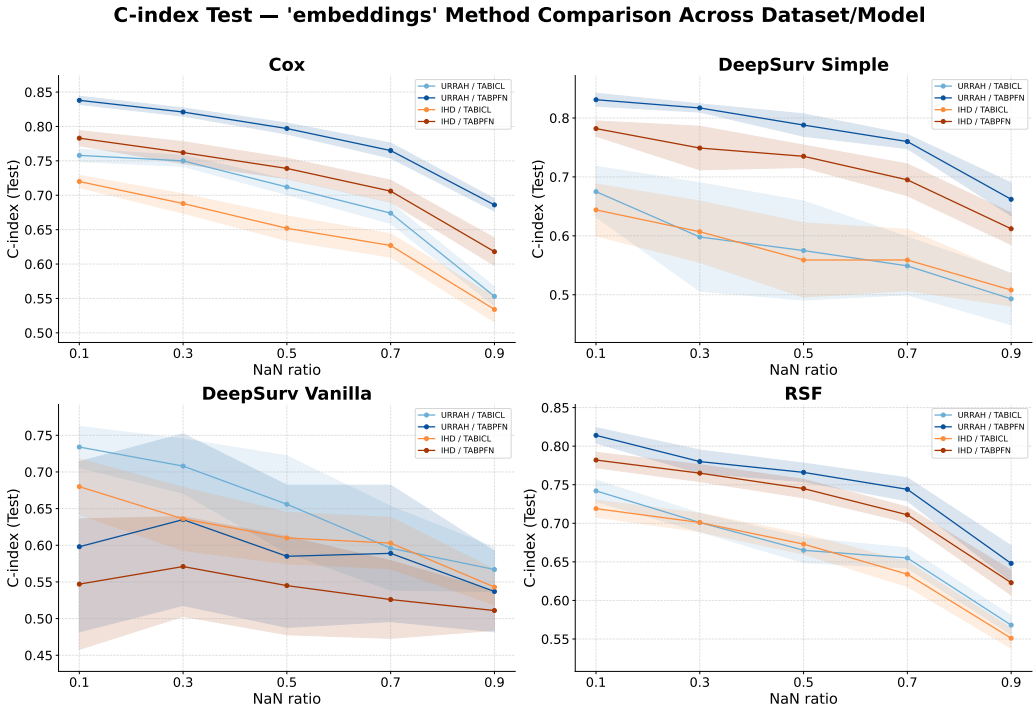

Figure 6.13: Concordance index (test) of the embedding-based approach as a function of the injected missing rate, with one panel per survival head. Each panel compares the four combinations of backbone and dataset (TabPFN and TabICL on IHD and URRAH). The shaded areas denote the standard deviation across the five seeds.

Chapter 7

Conclusion and Future Work

7.1 Summary of the work

This thesis investigated the use of tabular foundation models as feature extractors for survival analysis on clinical data. A complete pipeline was developed, in which two pretrained tabular foundation models, TabPFN and TabICL, are used to produce embeddings from the input data, and a set of survival heads - a classical Cox model, a Random Survival Forest, and neural DeepSurv models - is then trained on top of these embeddings to estimate the risk. The same survival heads, applied directly to the raw features, serve as baselines, so that the effect of introducing the foundation-model embeddings can be isolated. The approach was evaluated on two real clinical datasets, IHD and URRAH, using the Antolini time-dependent concordance index, and its behavior was studied along three axes: a direct comparison against the baselines, the dependence on the size of the training set, and the robustness to increasing amounts of missing data. In addition, a SHAP-based analysis was carried out to interpret the predictions in terms of the original clinical features.

7.2 Main Findings

Taken together, the experiments indicate that using the tabular foundation models as feature extractors is a viable approach for clinical survival analysis, but one that does not, in general, outperform the classical models. At full training size, the embedding-based configurations – particularly those based on TabPFN - are competitive with the baselines, but they do not surpass them by a clear margin, the best results being obtained by the baseline models on both datasets. The two backbones behave differently throughout: TabPFN is consistently stronger than TabICL, which remains the weaker of the two across all experiments.

The value of the approach emerges more clearly in specific scenarios rather than as a general improvement. The most evident case is the low-data regime: with few training samples, the TabPFN embeddings provide a substantial advantage over the baselines, which is progressively lost as more data becomes available. A similar picture arises for robustness to missing values, where the TabPFN embeddings are competitive at low missing rates - occasionally the best option - but degrade faster than the more informed imputation methods as missingness increases. The SHAP analysis, finally, showed that the feature importances are driven mainly by the backbone rather than by the survival head, with the two backbones relying on noticeably different sets of clinical features.

Overall, therefore, tabular foundation models can be regarded as a valid alternative to classical survival models rather than as a replacement for them: their use does not lead to systematically better predictions, but it can be beneficial in particular situations, most notably when only a small amount of training data is available.

7.3 Limitations

Some limitations of this work should be acknowledged. First, the comparisons are based on the mean concordance index and its standard deviation

across seeds, but no formal statistical significance testing was performed; differences that are small relative to the standard deviation should therefore be interpreted with caution. Second, the evaluation relies on two private clinical datasets, which limits the generalizability of the conclusions and prevents third parties from fully reproducing the results. Third, the study focused on describing the behavior of the different configurations rather than on explaining it: several of the patterns observed – such as the low-data advantage of TabPFN or the different feature-importance structures of the two backbones – were not investigated in terms of their underlying causes.

7.4 Future Work

The present work suggests several directions for future research. A first direction is to investigate the causes of the patterns observed here, which this thesis only described: in particular, why TabPFN provides an advantage in the low-data regime, and why the two backbones lead to such different feature importances. Establishing the reasons behind these behaviors would help clarify when and why foundation-model embeddings are beneficial.

A second direction concerns alternative ways of using the tabular foundation models. This thesis employed them exclusively as fixed feature extractors, but other modes of use are possible - for instance, training the survival head end-to-end together with the foundation model, rather than on frozen embeddings. A systematic comparison of these alternatives against the embedding-based approach, on the same datasets and under the same analyses, would give a more complete picture of how these models can best be exploited for survival analysis.

Finally, extending the evaluation to a larger and more diverse collection of datasets, including public ones, and complementing the analysis with formal statistical testing, would strengthen the conclusions and improve their generalizability.

Appendix A

Datasets

Table A.1: IHD dataset

Feature	Type	Distinct value	Missing values	% Missing values
Number	float64	8065	79	0.97
Gender (Male = 1)	float64	2	79	0.97
Age	float64	6351	79	0.97
Total cholesterol	float64	283	439	5.39
HDL	float64	107	689	8.46
LDL	float64	511	711	8.73
Triglycerides	float64	422	517	6.35
Glycemia	float64	289	493	6.05
TSH	float64	774	79	0.97
fT3	float64	431	79	0.97
fT4	float64	570	79	0.97
Euthyroid	float64	2	79	0.97
Subclinical primary hypothyroidism (SCH)	float64	2	79	0.97

Continued on next page

Table A.1 – Continued from previous page

Feature	Type	Distinct value	Missing values	% Missing values
Subclinical primary hyperthyroidism (SCT)	float64	2	79	0.97
Low T3	float64	2	79	0.97
Ipotiroidismo	float64	2	79	0.97
Ipertiroidismo	float64	2	79	0.97
Angina	float64	2	79	0.97
Previous CABG	float64	2	79	0.97
Previous PCI	float64	2	79	0.97
Previous Myocardial Infarction	float64	2	79	0.97
Acute Myocardial Infarction	float64	2	79	0.97
Angiography	float64	2	79	0.97
Vessels	float64	6	79	0.97
CAD	float64	2	79	0.97
Documented resting or exertional ischemia	float64	2	78	0.96
Post-ischemic Dilated Cardiomyopathy	float64	2	79	0.97
Primary Dilated Cardiomyopathy	float64	2	79	0.97
Normal	float64	2	79	0.97
SindromeX	float64	2	79	0.97
AMI	float64	2	79	0.97
PMI	float64	2	79	0.97

Continued on next page

Table A.1 – Continued from previous page

Feature	Type	Distinct value	Missing values	% Missing values
Aritmie	float64	2	79	0.97
MIN	float64	2	79	0.97
MIO	float64	2	79	0.97
Miocardite	float64	2	79	0.97
Pericardite	float64	2	79	0.97
Endocardite	float64	2	79	0.97
Valvulopatie	float64	2	79	0.97
MalattiaVasoAorta	float64	2	79	0.97
Ipertensione	float64	2	79	0.97
CardiopatiaCongenita	float64	1	79	0.97
EmboliaPolmonare	float64	2	79	0.97
EPA	float64	2	79	0.97
HR	float64	132	145	1.78
Weight	float64	113	493	6.05
Height	float64	64	575	7.06
BMI	float64	1954	652	8.01
Diastolic blood pressure	float64	66	240	2.95
Systolic blood pressure	float64	98	240	2.95
Smoke History of smoke	float64	2	79	0.97
Diabetes History of diabetes	float64	2	79	0.97
Hypertension History of hypertension	float64	2	79	0.97

Continued on next page

Table A.1 – Continued from previous page

Feature	Type	Distinct value	Missing values	% Missing values
Dyslipidemia Hystory of dyslipidemia	float64	2	79	0.97
Atrial Fibrillation	float64	2	79	0.97
WmsiBas	float64	124	302	3.71
DimSettoIV	float64	25	403	4.95
DimPP	float64	20	424	5.21
fe	float64	73	79	0.97
vsx	float64	65	789	9.69
B-Blockers	float64	2	79	0.97
Amiodarone	float64	2	79	0.97
Calcium channel blockers	float64	2	79	0.97
Diuretics	float64	2	79	0.97
Antiplatelet	float64	2	79	0.97
Nitrates	float64	2	79	0.97
ACE-inhibitors	float64	2	79	0.97
Ipolipemizzanti	float64	2	206	2.53
Antidiabetici	float64	2	210	2.58
Follow Up Data	datetime64[us]	2334	79	0.97
Data of death	datetime64[us]	1842	5737	70.44
Cause of death	str	23	5738	70.46
Collected by	str	8	5737	70.44
Total mortality	float64	2	79	0.97
CVD Death	float64	2	79	0.97
Fatal MI or Sudden death	float64	2	79	0.97
UnKnown	float64	2	79	0.97

Continued on next page

Table A.1 – Continued from previous page

Feature	Type	Distinct value	Missing values	% Missing values
Accident	float64	2	79	0.97
Suicide	float64	2	79	0.97
CABG	datetime64[us]	723	7318	89.86
Non Fatal AMI (Follow-Up)	datetime64[us]	303	7821	96.03
Ictus	datetime64[us]	99	8034	98.65
PCI	datetime64[us]	1123	6798	83.47
Creatinina	float64	325	675	8.29
Data prelievo	datetime64[us]	3539	79	0.97

Table A.2: URRAH dataset

Feature	Type	Distinct value	Missing values	% Missing values
SESSO	int64	2	0	0.00
SESSO0	str	2	0	0.00
ETA	float64	4708	4	0.01
FAM_IPERT	float64	2	8140	30.06
FAM_CV	float64	2	10695	39.50
DIABETE	float64	2	1176	4.34
FUMO_ATT	float64	2	2552	9.42
EXFUMO	float64	2	5023	18.55
ALCOOL	float64	2	7564	27.93
ETANOLO_G_DIE	float64	639	19848	73.30
ATT_FIS	float64	2	13473	49.76
ATT_FIS_H_SETT	float64	816	20281	74.90
SAPEVA	float64	2	7630	28.18
IRC	float64	2	1186	4.38

Continued on next page

Table A.2 – Continued from previous page

Feature	Type	Distinct value	Missing values	% Missing values
GOTTA	float64	2	13028	48.11
ALLOPURINOLO	float64	2	11029	40.73
PESO	float64	895	2550	9.42
ALTEZZA	float64	120	2584	9.54
BMI	float64	6419	2624	9.69
VITA	float64	189	14209	52.47
PAS	float64	577	589	2.18
PAD	float64	350	584	2.16
FC	float64	363	4646	17.16
CREATININA	float64	507	2373	8.76
AZOTEMIA	float64	559	12925	47.73
URICEMIA	float64	461	0	0.00
ALB_CAT	float64	2	10169	37.55
ALB_CREAT_MG_MMOL_3_4_34_MICRO	float64	1333	24680	91.14
ALB_CREAT_MG_G_30_300_MICRO	float64	1470	24370	90.00
ALBURIA_MG_24H_30_300	float64	700	25173	92.96
ALBURIA_MG_DL	float64	569	25799	95.28
NORMOMICROMACRO	float64	3	20401	75.34
GLICEMIA	float64	335	5069	18.72
CT	float64	349	1037	3.83
HDL	float64	638	4142	15.30
TG	float64	649	0	0.00

Continued on next page

Table A.2 – Continued from previous page

Feature	Type	Distinct value	Missing values	% Missing values
HB	float64	356	8517	31.45
HT	float64	477	6583	24.31
LVH	float64	2	26907	99.37
IMT	float64	221	23157	85.52
PLACCA	float64	2	16142	59.61
INS_CARD	float64	2	6845	25.28
ACEI	float64	2	6726	24.84
SARTANI	float64	2	9801	36.20
CAA_DIID	float64	2	2441	9.01
ALTRO_CAA	float64	2	13225	48.84
BETABLOCCANTI	float64	2	3516	12.98
DIURETICI	float64	2	3366	12.43
HCTZ	float64	2	17337	64.03
INDAPAMIDE	float64	2	17349	64.07
CLORTALIDONE	float64	2	14141	52.22
DELL_ANSA	float64	2	11211	41.40
STATINA	float64	2	1344	4.96
STATO_AL_FU	float64	2	5831	21.53
NF_IMA	float64	2	6315	23.32
F_IMA	float64	2	7146	26.39
NF_CBV	float64	2	6330	23.38
F_CBV	float64	2	7197	26.58
NF_HF	float64	2	7391	27.30
F_HF	float64	3	10145	37.47
RIV_COR	float64	2	14486	53.50
MORTE_CV	float64	2	5140	18.98
IPERTESO	int64	2	0	0.00
FU	float64	309	4245	15.68

Continued on next page

Table A.2 – Continued from previous page

Feature	Type	Distinct value	Missing values	% Missing values
FU_NF_IMA	float64	304	4813	17.77
FU_F_IMA	float64	259	4812	17.77
FU_NF_CBV	float64	304	4809	17.76
FU_F_CBV	float64	303	4812	17.77
FU_NF_HF	float64	259	4814	17.78
FU_F_HF	float64	308	4813	17.77
PCR	float64	498	23234	85.80
VES	float64	69	26398	97.49
OMEGA_3	float64	2	23073	85.21
FIBRATI	float64	2	22382	82.66
EZETEMIBE	float64	2	22639	83.61
ACIDO_ NICOTINICO	float64	1	23684	87.47
RESINE	float64	2	23684	87.47
MENOPAUSA	float64	2	23908	88.29
ANNI_MENOP	float64	377	25631	94.66
IMA_BASE	float64	1	25502	94.18
CBV_BASE	float64	1	25502	94.18
FA_BASE	float64	2	23466	86.66
HF_BASE	float64	1	25502	94.18
NF_FA	float64	2	24795	91.57
FU_NF_FA	float64	10	24803	91.60

Appendix B

SHAP Feature-Importance Figures

This appendix collects the complete set of SHAP feature-importance figures for Experiment 1, complementing the heatmaps shown in the main text (Section 6.2). For each dataset (IHD and URRAH), each backbone (TabPFN and TabICL) and each missing-data variant (NaN and -1), the twenty features with the highest mean absolute SHAP value are reported for every survival head, with the error bars denoting the standard deviation across the five seeds. The feature-importance heatmaps for the NaN variant are also included here, the -1 heatmaps having already been presented in the main text.

B.1 Top-Feature Bar Plots

B.1.1 IHD Dataset

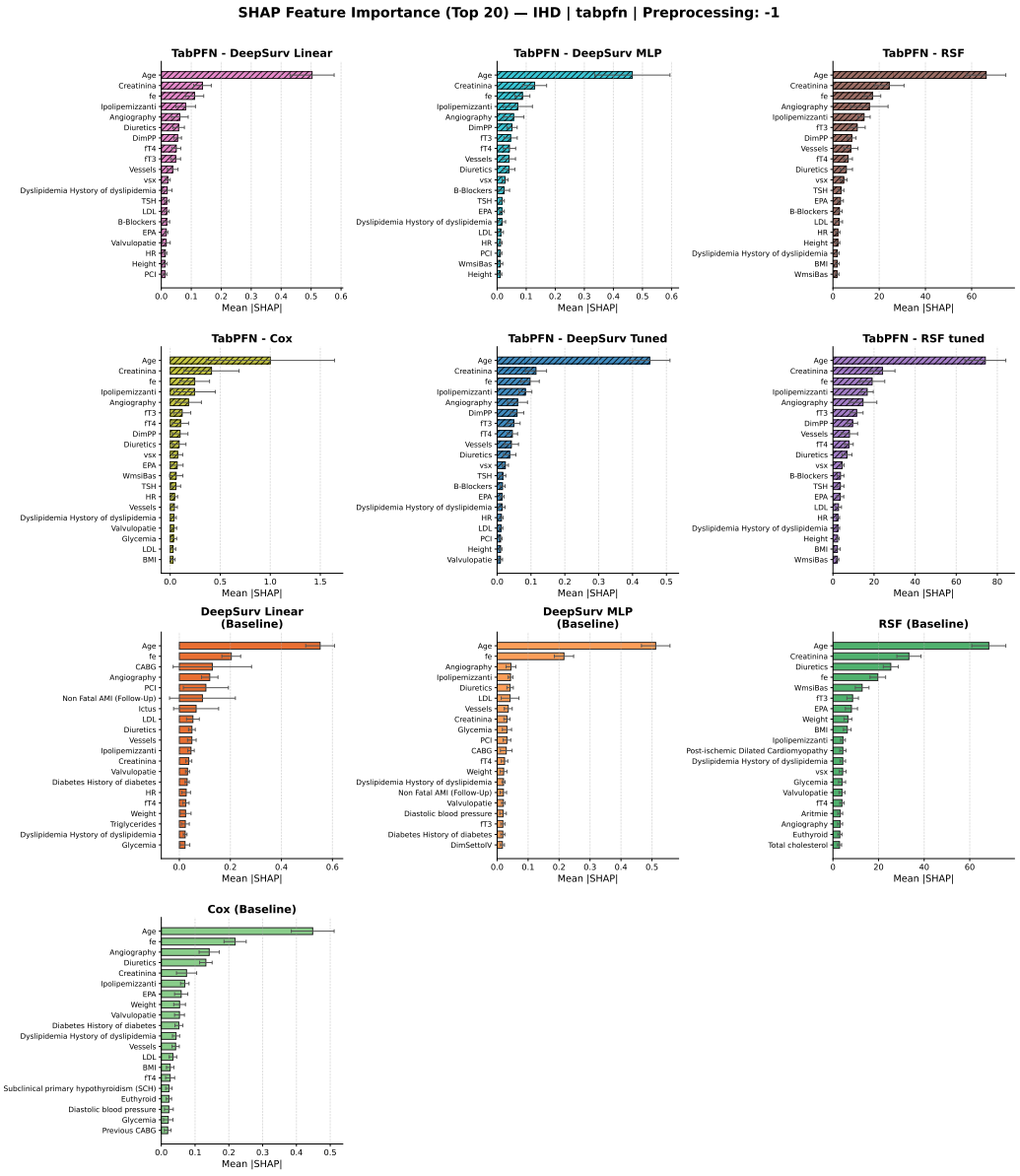

Figure B.1: Top twenty features by mean absolute SHAP value for each survival head, using the TabPFN embeddings on the IHD dataset (−1 variant).

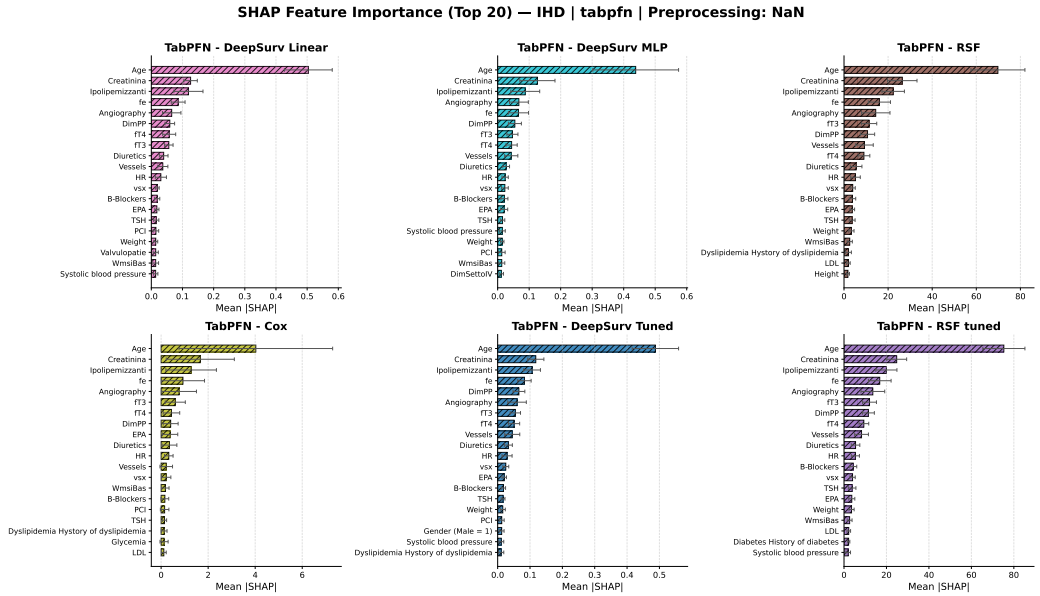

Figure B.2: Top twenty features by mean absolute SHAP value for each survival head, using the TabPFN embeddings on the IHD dataset (NaN variant).

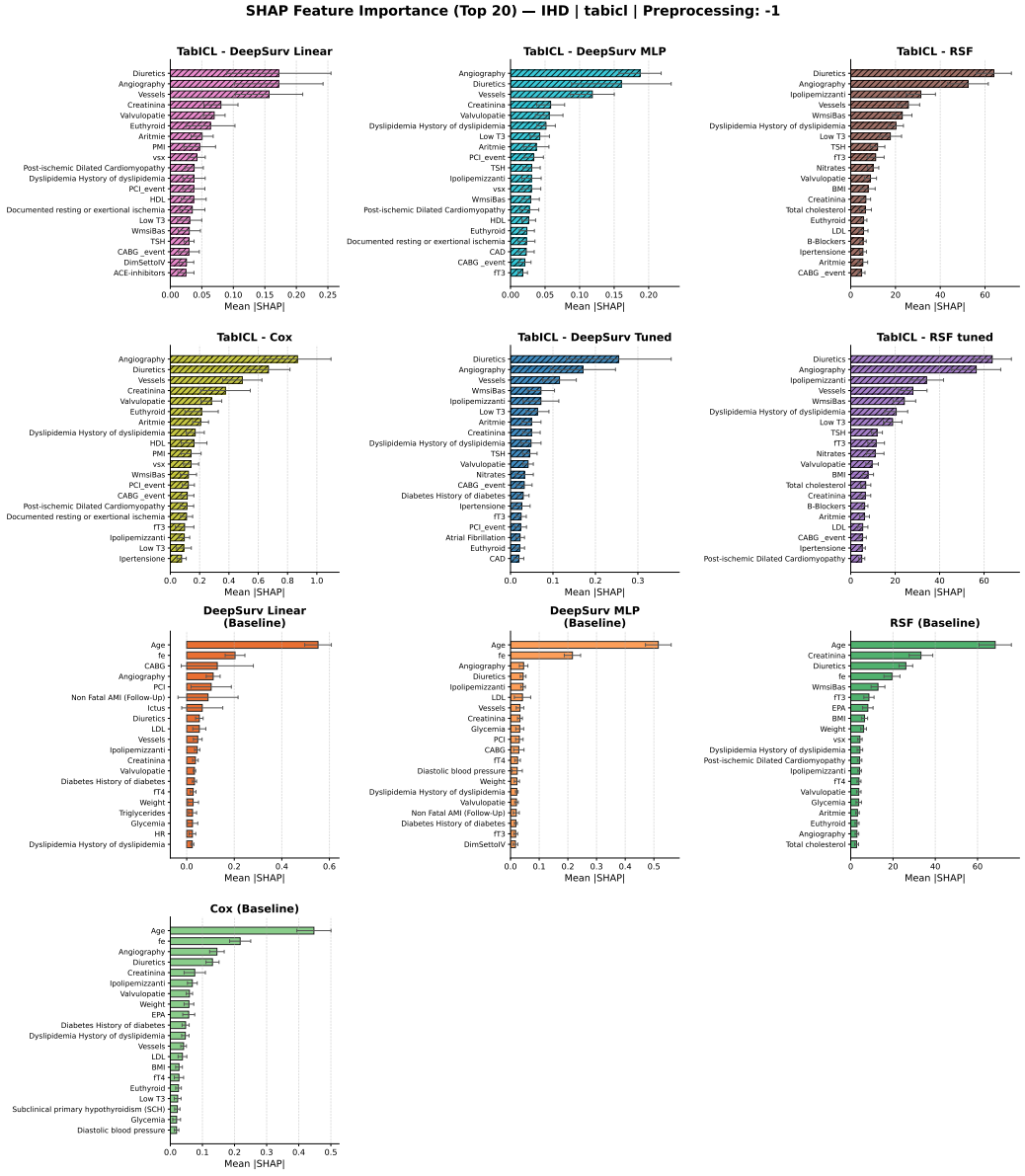

Figure B.3: Top twenty features by mean absolute SHAP value for each survival head, using the TabICL embeddings on the IHD dataset (−1 variant).

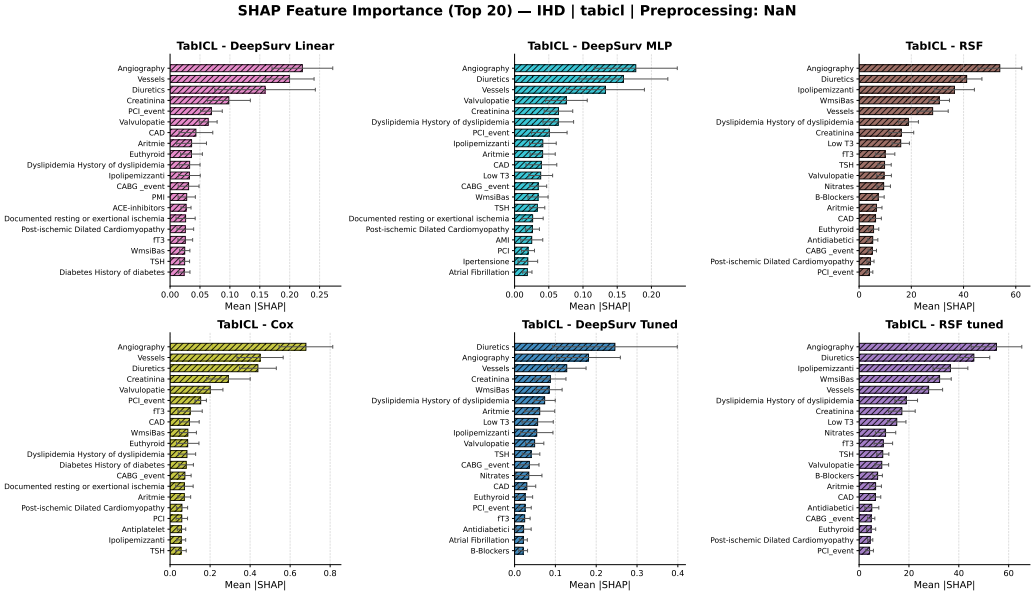

Figure B.4: Top twenty features by mean absolute SHAP value for each survival head, using the TabICL embeddings on the IHD dataset (NaN variant).

B.1.2 URRAH Dataset

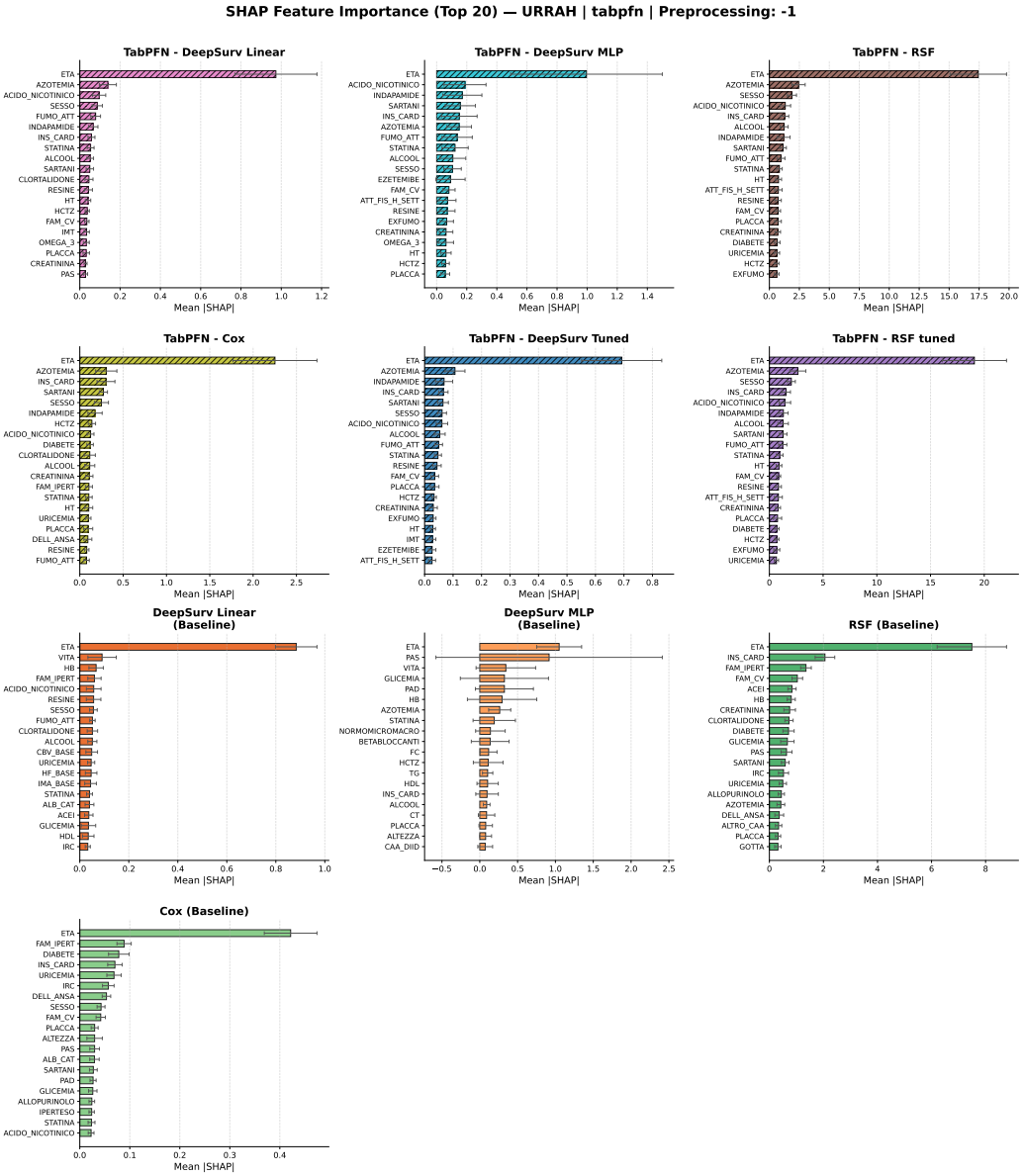

Figure B.5: Top twenty features by mean absolute SHAP value for each survival head, using the TabPFN embeddings on the URRAH dataset (−1 variant).

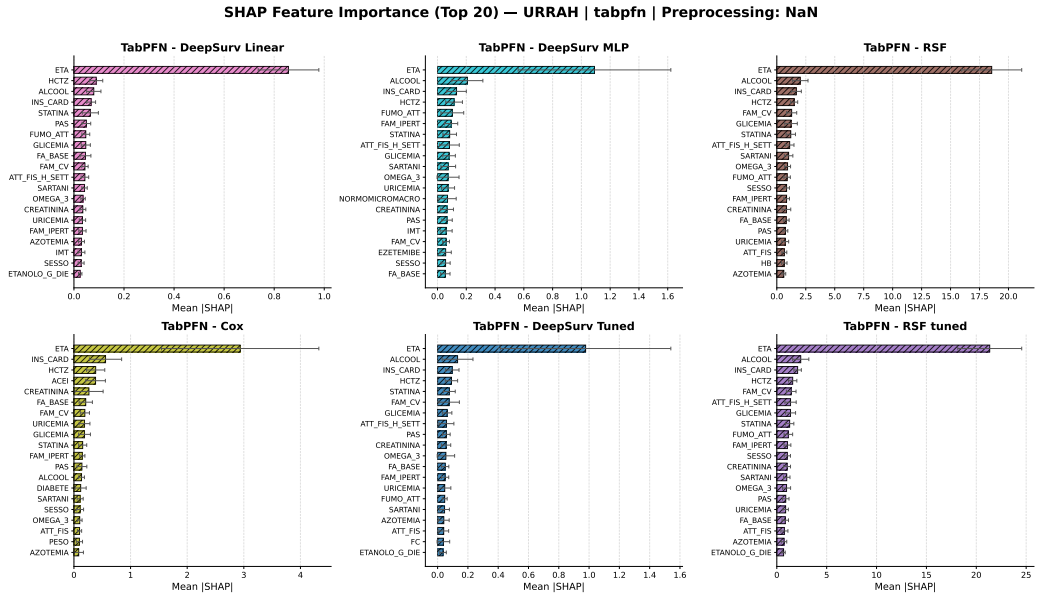

Figure B.6: Top twenty features by mean absolute SHAP value for each survival head, using the TabPFN embeddings on the URRAH dataset (NaN variant).

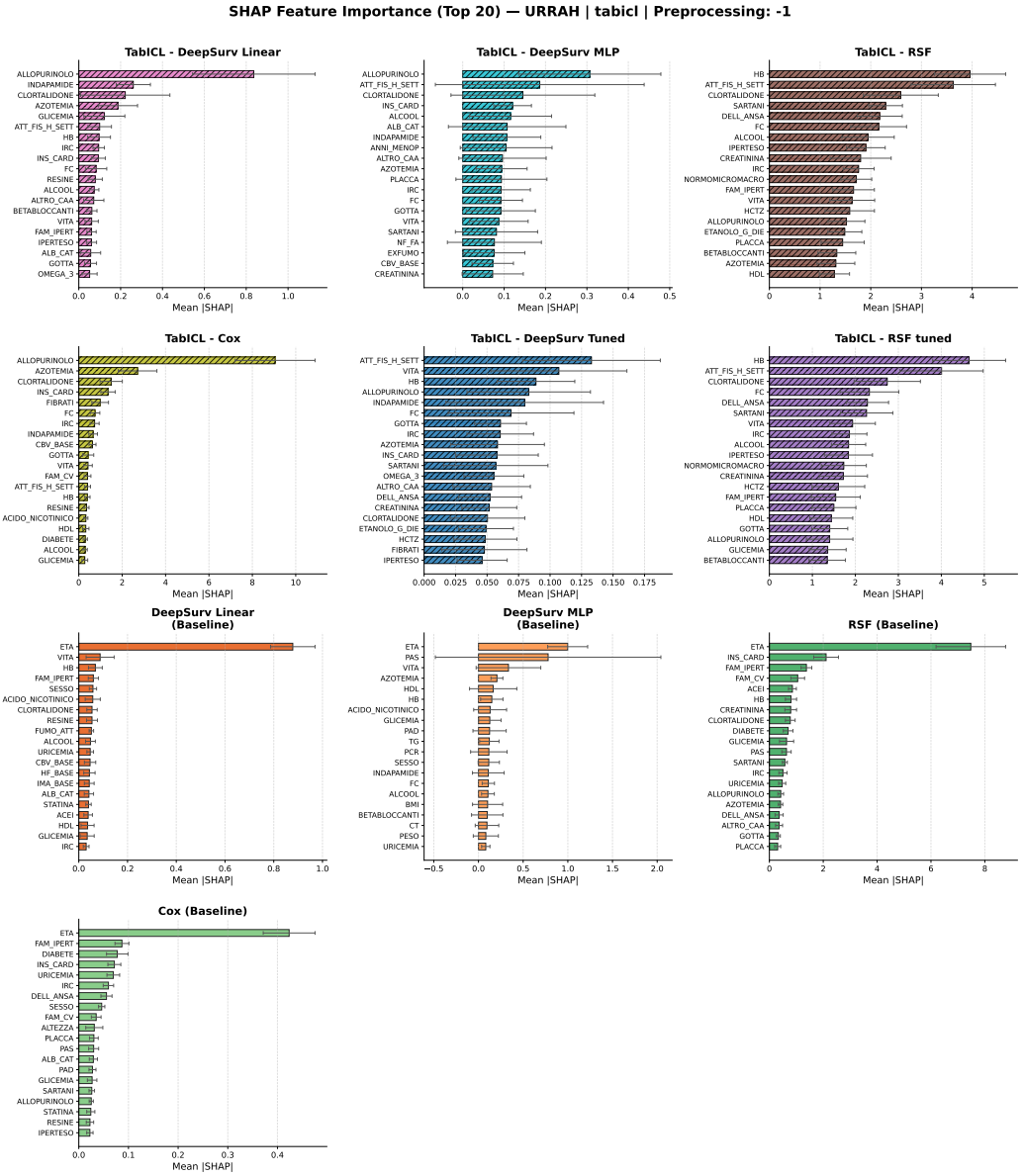

Figure B.7: Top twenty features by mean absolute SHAP value for each survival head, using the TabICL embeddings on the URRAH dataset (−1 variant).

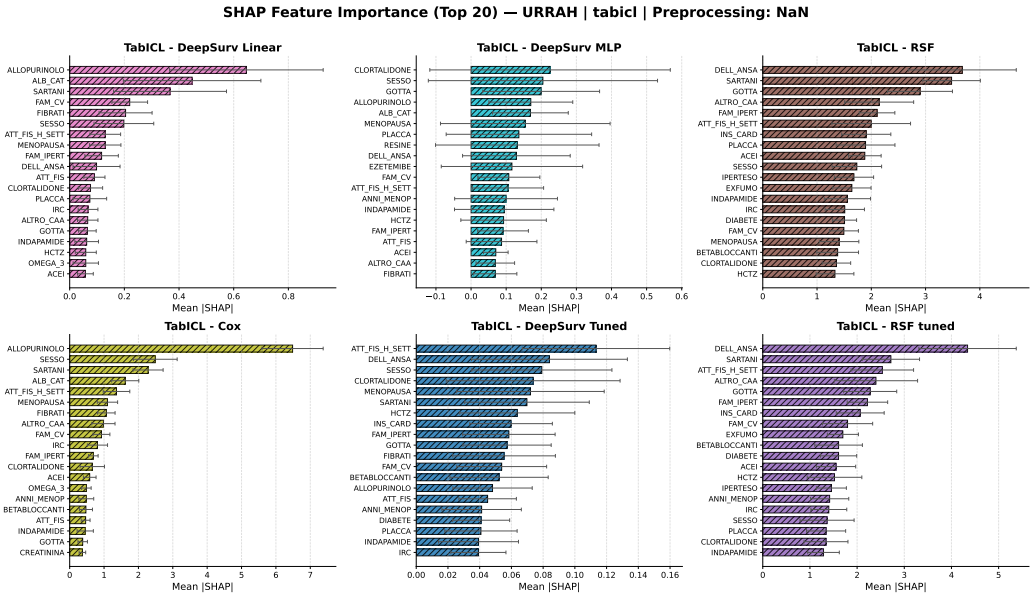

Figure B.8: Top twenty features by mean absolute SHAP value for each survival head, using the TabICL embeddings on the URRAH dataset (NaN variant).

B.2 Heatmaps (NaN Variant)

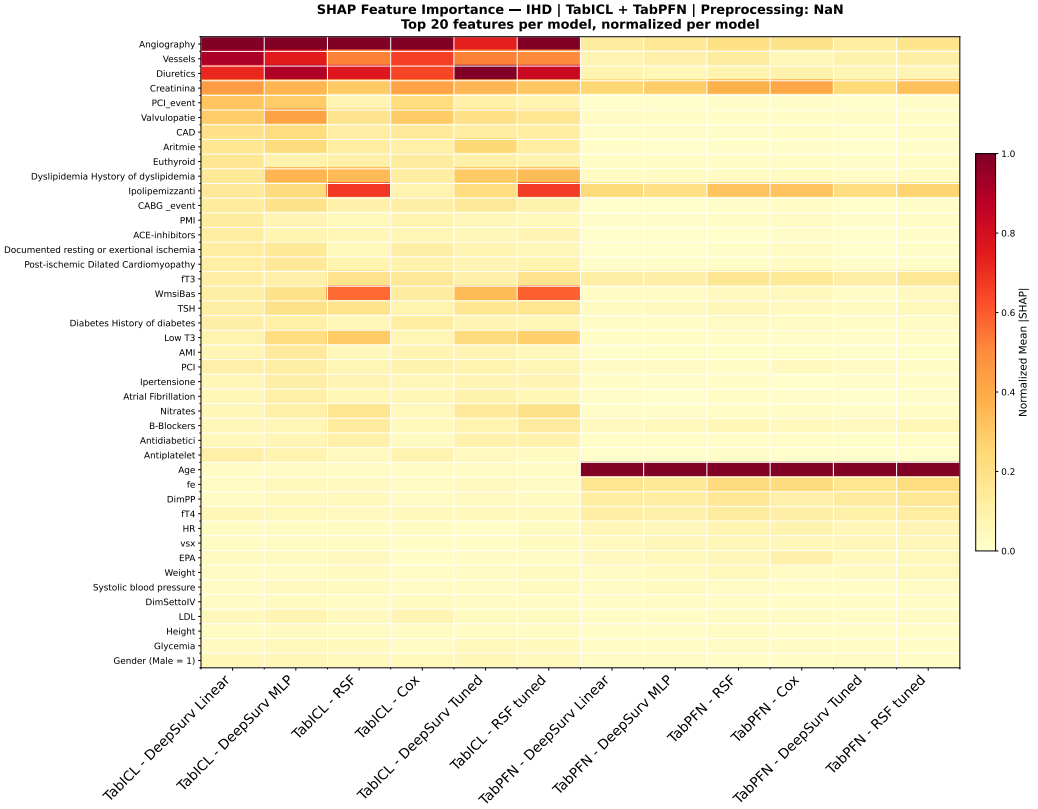

Figure B.9: Feature-importance heatmap (per-model normalized mean absolute SHAP value) on the IHD dataset (NaN variant). Each row corresponds to a feature and each column to a model, including the survival heads on the TabPFN and TabICL embeddings. Colors range from low (yellow) to high (red).

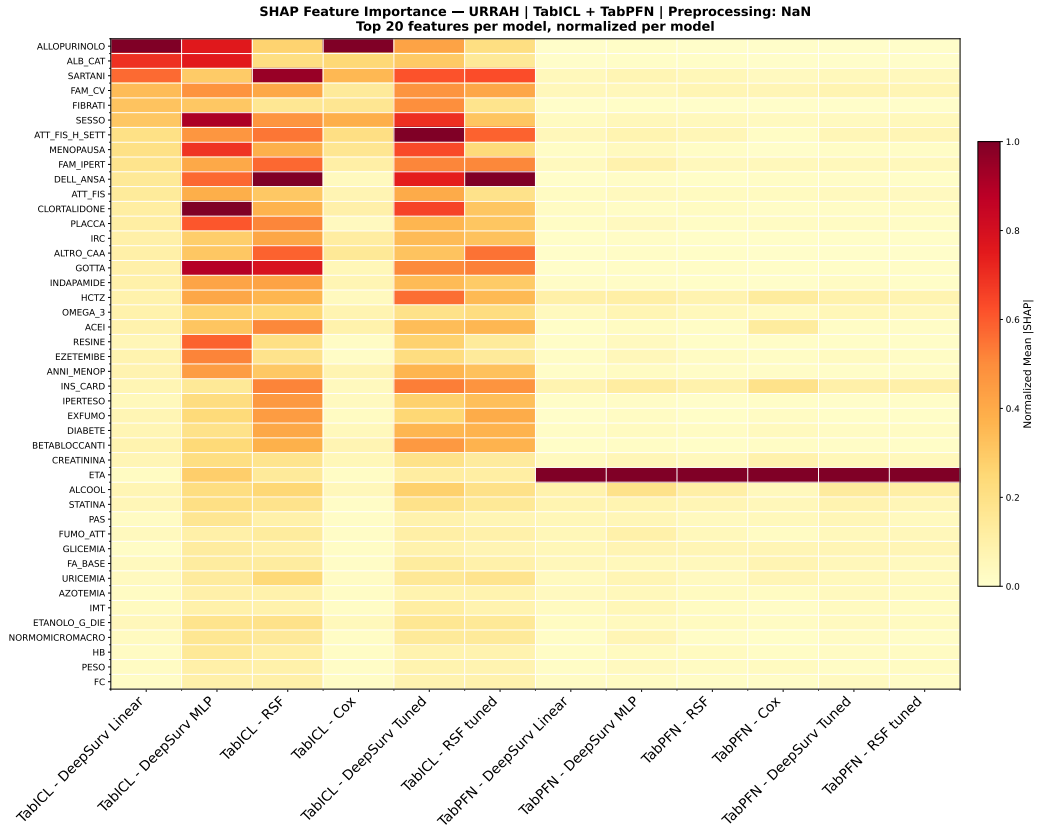

Figure B.10: Feature-importance heatmap (per-model normalized mean absolute SHAP value) on the URRAH dataset (NaN variant). Each row corresponds to a feature and each column to a model, including the survival heads on the TabPFN and TabICL embeddings. Colors range from low (yellow) to high (red).

Appendix C

Training Set Size: Complete Results

This appendix reports the complete numerical results of Experiment 2 (Section 6.3). For each dataset, backbone and missing-data variant, the concordance index (mean ± standard deviation over the five seeds) is reported for every survival head and every training set size. The baseline models are included only in the -1 variant. The figures in the main text are based on these values.

C.1 IHD Dataset

Table C.1: Concordance index on the IHD dataset with the TabPFN backbone (-1 variant), as a function of the training set size.

Survival head	100	500	1000	2500	5000	5161
Cox	0.595 ± 0.030	0.702 ± 0.069	0.758 ± 0.040	0.786 ± 0.006	0.788 ± 0.007	0.788 ± 0.008
RSF (fixed)	0.704 ± 0.020	0.761 ± 0.005	0.774 ± 0.003	0.784 ± 0.003	0.788 ± 0.002	0.787 ± 0.003
DeepSurv (MLP, fixed)	0.717 ± 0.015	0.771 ± 0.005	0.780 ± 0.003	0.785 ± 0.003	0.787 ± 0.004	0.783 ± 0.002
DeepSurv (linear)	0.719 ± 0.017	0.773 ± 0.005	0.782 ± 0.003	0.789 ± 0.002	0.793 ± 0.002	0.793 ± 0.002
Cox (baseline)	N/A	0.772 ± 0.016	0.781 ± 0.003	0.791 ± 0.001	0.794 ± 0.001	0.794 ± 0.001
RSF (baseline)	0.727 ± 0.006	0.771 ± 0.004	0.783 ± 0.003	0.792 ± 0.001	0.796 ± 0.001	0.796 ± 0.001
DeepSurv (baseline)	0.627 ± 0.033	0.740 ± 0.004	0.761 ± 0.007	0.783 ± 0.002	0.790 ± 0.002	0.791 ± 0.001
DeepSurv lin. (base.)	0.626 ± 0.026	0.755 ± 0.007	0.774 ± 0.004	0.780 ± 0.002	0.782 ± 0.003	0.780 ± 0.004

Table C.2: Concordance index on the IHD dataset with the TabPFN backbone (NaN variant), as a function of the training set size.

Survival head	100	500	1000	2500	5000	5161
Cox	0.627 ± 0.090	0.718 ± 0.055	0.742 ± 0.042	0.785 ± 0.006	0.792 ± 0.004	0.793 ± 0.003
RSF (fixed)	0.714 ± 0.016	0.764 ± 0.006	0.777 ± 0.004	0.782 ± 0.002	0.786 ± 0.004	0.786 ± 0.003
DeepSurv (MLP, fixed)	0.722 ± 0.012	0.775 ± 0.007	0.783 ± 0.002	0.785 ± 0.006	0.787 ± 0.004	0.788 ± 0.003
DeepSurv (linear)	0.725 ± 0.012	0.775 ± 0.007	0.784 ± 0.002	0.790 ± 0.001	0.794 ± 0.001	0.794 ± 0.002

Table C.3: Concordance index on the IHD dataset with the TabICL backbone (-1 variant), as a function of the training set size.

Survival head	100	500	1000	2500	5000	5161
Cox	0.608 ± 0.006	0.647 ± 0.028	0.693 ± 0.007	0.727 ± 0.005	0.744 ± 0.002	0.745 ± 0.002
RSF (fixed)	0.643 ± 0.012	0.699 ± 0.005	0.720 ± 0.002	0.733 ± 0.002	0.737 ± 0.001	0.739 ± 0.001
DeepSurv (MLP, fixed)	0.611 ± 0.013	0.695 ± 0.011	0.720 ± 0.005	0.739 ± 0.003	0.748 ± 0.003	0.746 ± 0.004
DeepSurv (linear)	0.621 ± 0.024	0.700 ± 0.007	0.712 ± 0.006	0.722 ± 0.006	0.733 ± 0.004	0.733 ± 0.004
Cox (baseline)	N/A	0.772 ± 0.016	0.781 ± 0.003	0.791 ± 0.001	0.794 ± 0.001	0.794 ± 0.001
RSF (baseline)	0.727 ± 0.006	0.771 ± 0.004	0.783 ± 0.003	0.792 ± 0.001	0.796 ± 0.001	0.796 ± 0.001
DeepSurv (baseline)	0.627 ± 0.033	0.740 ± 0.004	0.761 ± 0.007	0.783 ± 0.002	0.790 ± 0.002	0.791 ± 0.001
DeepSurv lin. (base.)	0.626 ± 0.026	0.755 ± 0.007	0.774 ± 0.004	0.780 ± 0.002	0.782 ± 0.003	0.780 ± 0.004

Table C.4: Concordance index on the IHD dataset with the TabICL backbone (NaN variant), as a function of the training set size.

Survival head	100	500	1000	2500	5000	5161
Cox	0.602 ± 0.011	0.636 ± 0.036	0.683 ± 0.027	0.725 ± 0.004	0.745 ± 0.001	0.746 ± 0.001
RSF (fixed)	0.645 ± 0.007	0.702 ± 0.004	0.722 ± 0.001	0.738 ± 0.002	0.747 ± 0.001	0.747 ± 0.002
DeepSurv (MLP, fixed)	0.622 ± 0.009	0.696 ± 0.006	0.726 ± 0.008	0.738 ± 0.004	0.749 ± 0.004	0.745 ± 0.003
DeepSurv (linear)	0.628 ± 0.016	0.711 ± 0.009	0.717 ± 0.005	0.728 ± 0.008	0.739 ± 0.005	0.741 ± 0.001

C.2 URRAH Dataset

Table C.5: Concordance index on the URRAH dataset with the TabPFN backbone (−1 variant), as a function of the training set size.

Survival head	100	500	1000	2500	5000	10000	13566
Cox	0.694 ± 0.029	0.744 ± 0.049	0.771 ± 0.036	0.821 ± 0.015	0.831 ± 0.011	0.844 ± 0.004	0.845 ± 0.004
RSF (fixed)	0.740 ± 0.013	0.780 ± 0.011	0.798 ± 0.014	0.821 ± 0.010	0.827 ± 0.009	0.821 ± 0.009	0.825 ± 0.006
DeepSurv (MLP, fixed)	0.743 ± 0.015	0.806 ± 0.011	0.812 ± 0.005	0.814 ± 0.017	0.829 ± 0.003	0.823 ± 0.008	0.801 ± 0.028
DeepSurv (linear)	0.751 ± 0.019	0.814 ± 0.002	0.826 ± 0.003	0.837 ± 0.001	0.839 ± 0.002	0.825 ± 0.037	0.844 ± 0.002
Cox (baseline)	0.736 ± 0.019	0.805 ± 0.004	0.817 ± 0.001	0.822 ± 0.001	0.825 ± 0.001	0.826 ± 0.001	0.826 ± 0.000
RSF (baseline)	0.708 ± 0.022	0.793 ± 0.006	0.813 ± 0.004	0.832 ± 0.002	0.842 ± 0.001	0.849 ± 0.001	0.851 ± 0.001
DeepSurv (baseline)	0.630 ± 0.049	0.764 ± 0.011	0.801 ± 0.007	0.821 ± 0.002	0.831 ± 0.004	0.828 ± 0.006	0.842 ± 0.007
DeepSurv lin. (base.)	0.714 ± 0.020	0.805 ± 0.002	0.822 ± 0.002	0.830 ± 0.001	0.831 ± 0.003	0.779 ± 0.041	0.832 ± 0.002

Table C.6: Concordance index on the URRAH dataset with the TabPFN backbone (NaN variant), as a function of the training set size.

Survival head	100	500	1000	2500	5000	10000	13566
Cox	0.635 ± 0.066	0.759 ± 0.044	0.752 ± 0.040	0.793 ± 0.028	0.822 ± 0.017	0.837 ± 0.004	0.841 ± 0.004
RSF (fixed)	0.722 ± 0.026	0.774 ± 0.013	0.787 ± 0.019	0.810 ± 0.014	0.819 ± 0.008	0.808 ± 0.014	0.798 ± 0.022
DeepSurv (MLP, fixed)	0.738 ± 0.022	0.804 ± 0.004	0.812 ± 0.003	0.807 ± 0.016	0.825 ± 0.008	0.784 ± 0.038	0.779 ± 0.033
DeepSurv (linear)	0.748 ± 0.027	0.809 ± 0.003	0.821 ± 0.002	0.832 ± 0.002	0.837 ± 0.001	0.821 ± 0.037	0.841 ± 0.005

Table C.7: Concordance index on the URRAH dataset with the TabICL backbone (−1 variant), as a function of the training set size.

Survival head	100	500	1000	2500	5000	10000	13566
Cox	0.557 ± 0.021	0.625 ± 0.054	0.681 ± 0.013	0.715 ± 0.043	0.756 ± 0.006	0.766 ± 0.001	0.769 ± 0.001
RSF (fixed)	0.652 ± 0.029	0.647 ± 0.018	0.687 ± 0.007	0.727 ± 0.011	0.735 ± 0.006	0.745 ± 0.007	0.750 ± 0.005
DeepSurv (MLP, fixed)	0.642 ± 0.031	0.725 ± 0.014	0.741 ± 0.011	0.764 ± 0.005	0.777 ± 0.003	0.753 ± 0.027	0.775 ± 0.004
DeepSurv (linear)	0.659 ± 0.022	0.741 ± 0.010	0.743 ± 0.012	0.781 ± 0.005	0.787 ± 0.008	0.783 ± 0.017	0.783 ± 0.013
Cox (baseline)	0.736 ± 0.019	0.805 ± 0.004	0.817 ± 0.001	0.822 ± 0.001	0.825 ± 0.001	0.826 ± 0.001	0.826 ± 0.000
RSF (baseline)	0.708 ± 0.022	0.793 ± 0.006	0.813 ± 0.004	0.832 ± 0.002	0.842 ± 0.001	0.849 ± 0.001	0.851 ± 0.001
DeepSurv (baseline)	0.630 ± 0.049	0.764 ± 0.011	0.801 ± 0.007	0.821 ± 0.002	0.831 ± 0.004	0.828 ± 0.006	0.842 ± 0.007
DeepSurv lin. (base.)	0.714 ± 0.020	0.805 ± 0.002	0.822 ± 0.002	0.830 ± 0.001	0.831 ± 0.003	0.779 ± 0.041	0.832 ± 0.002

Table C.8: Concordance index on the URRAH dataset with the TabICL backbone (NaN variant), as a function of the training set size.

Survival head	100	500	1000	2500	5000	10000	13566
Cox	0.568 ± 0.022	0.622 ± 0.053	0.688 ± 0.015	0.704 ± 0.033	0.725 ± 0.005	0.735 ± 0.003	0.739 ± 0.001
RSF (fixed)	0.645 ± 0.020	0.640 ± 0.009	0.696 ± 0.013	0.717 ± 0.008	0.726 ± 0.006	0.724 ± 0.004	0.723 ± 0.003
DeepSurv (MLP, fixed)	0.639 ± 0.026	0.719 ± 0.011	0.747 ± 0.009	0.755 ± 0.006	0.765 ± 0.008	0.751 ± 0.023	0.759 ± 0.012
DeepSurv (linear)	0.667 ± 0.021	0.735 ± 0.004	0.750 ± 0.002	0.775 ± 0.006	0.780 ± 0.006	0.763 ± 0.014	0.768 ± 0.007

Appendix D

Robustness to Missing Values: Complete Results

This appendix reports the complete numerical results of Experiment 3 (Section 6.4). For each dataset and backbone, the concordance index on the test folds (mean ± standard deviation over the five seeds) is reported for every survival head, every injected missing rate, and every method (the embedding-based approach and the five imputation methods).

Table D.1: Concordance index (test) on the IHD dataset with the TabPFN backbone, as a function of the injected missing rate, for every survival head and every method.

Rate	Head	Constant	Embeddings	Iterative	KNN	Mean	Median
0.1	Cox	0.731 ± 0.013	0.783 ± 0.011	0.788 ± 0.010	0.786 ± 0.009	0.786 ± 0.010	0.785 ± 0.009
	RSF	0.788 ± 0.011	0.782 ± 0.010	0.792 ± 0.009	0.789 ± 0.009	0.789 ± 0.010	0.790 ± 0.009
	DeepSurv (MLP)	0.751 ± 0.011	0.547 ± 0.089	0.781 ± 0.011	0.778 ± 0.011	0.780 ± 0.012	0.777 ± 0.012
	DeepSurv (linear)	0.689 ± 0.019	0.782 ± 0.013	0.737 ± 0.022	0.747 ± 0.016	0.742 ± 0.020	0.747 ± 0.019
0.3	Cox	0.672 ± 0.010	0.762 ± 0.016	0.771 ± 0.009	0.765 ± 0.009	0.767 ± 0.008	0.765 ± 0.008
	RSF	0.767 ± 0.010	0.765 ± 0.011	0.775 ± 0.011	0.770 ± 0.010	0.772 ± 0.010	0.772 ± 0.009
	DeepSurv (MLP)	0.696 ± 0.016	0.571 ± 0.068	0.763 ± 0.010	0.752 ± 0.012	0.759 ± 0.012	0.756 ± 0.011
	DeepSurv (linear)	0.628 ± 0.020	0.749 ± 0.037	0.702 ± 0.032	0.715 ± 0.022	0.715 ± 0.031	0.740 ± 0.012
0.5	Cox	0.625 ± 0.013	0.739 ± 0.015	0.750 ± 0.010	0.739 ± 0.010	0.745 ± 0.009	0.741 ± 0.008
	RSF	0.739 ± 0.012	0.745 ± 0.012	0.754 ± 0.012	0.745 ± 0.012	0.748 ± 0.010	0.748 ± 0.010
	DeepSurv (MLP)	0.648 ± 0.018	0.545 ± 0.067	0.740 ± 0.013	0.726 ± 0.009	0.739 ± 0.010	0.734 ± 0.010
	DeepSurv (linear)	0.587 ± 0.021	0.735 ± 0.019	0.638 ± 0.044	0.679 ± 0.022	0.681 ± 0.037	0.720 ± 0.011
0.7	Cox	0.579 ± 0.014	0.706 ± 0.016	0.709 ± 0.011	0.689 ± 0.011	0.709 ± 0.009	0.703 ± 0.010
	RSF	0.693 ± 0.012	0.711 ± 0.010	0.710 ± 0.013	0.700 ± 0.012	0.708 ± 0.011	0.707 ± 0.011
	DeepSurv (MLP)	0.582 ± 0.016	0.526 ± 0.053	0.698 ± 0.011	0.667 ± 0.014	0.702 ± 0.011	0.694 ± 0.016
	DeepSurv (linear)	0.552 ± 0.015	0.695 ± 0.027	0.605 ± 0.029	0.628 ± 0.023	0.662 ± 0.034	0.679 ± 0.020
0.9	Cox	0.525 ± 0.016	0.618 ± 0.020	0.623 ± 0.007	0.591 ± 0.008	0.630 ± 0.006	0.619 ± 0.009
	RSF	0.612 ± 0.016	0.623 ± 0.016	0.618 ± 0.015	0.603 ± 0.013	0.628 ± 0.013	0.628 ± 0.013
	DeepSurv (MLP)	0.540 ± 0.015	0.511 ± 0.027	0.618 ± 0.018	0.565 ± 0.013	0.622 ± 0.014	0.620 ± 0.014
	DeepSurv (linear)	0.517 ± 0.013	0.612 ± 0.027	0.587 ± 0.033	0.537 ± 0.019	0.602 ± 0.017	0.612 ± 0.016

Table D.2: Concordance index (test) on the IHD dataset with the TabICL backbone, as a function of the injected missing rate, for every survival head and every method.

Rate	Head	Constant	Embeddings	Iterative	KNN	Mean	Median
0.1	Cox	0.731 ± 0.013	0.720 ± 0.009	0.788 ± 0.010	0.786 ± 0.009	0.786 ± 0.010	0.785 ± 0.009
	RSF	0.788 ± 0.011	0.719 ± 0.011	0.792 ± 0.009	0.789 ± 0.009	0.789 ± 0.010	0.790 ± 0.009
	DeepSurv (MLP)	0.751 ± 0.011	0.680 ± 0.038	0.781 ± 0.011	0.778 ± 0.011	0.780 ± 0.012	0.777 ± 0.012
	DeepSurv (linear)	0.689 ± 0.019	0.644 ± 0.044	0.737 ± 0.022	0.747 ± 0.016	0.742 ± 0.020	0.747 ± 0.019
0.3	Cox	0.672 ± 0.010	0.688 ± 0.014	0.771 ± 0.009	0.765 ± 0.009	0.767 ± 0.008	0.765 ± 0.008
	RSF	0.767 ± 0.010	0.701 ± 0.012	0.775 ± 0.011	0.770 ± 0.010	0.772 ± 0.010	0.772 ± 0.009
	DeepSurv (MLP)	0.696 ± 0.016	0.636 ± 0.043	0.763 ± 0.010	0.752 ± 0.012	0.759 ± 0.012	0.756 ± 0.011
	DeepSurv (linear)	0.628 ± 0.020	0.607 ± 0.052	0.702 ± 0.032	0.715 ± 0.022	0.715 ± 0.031	0.740 ± 0.012
0.5	Cox	0.625 ± 0.013	0.652 ± 0.018	0.750 ± 0.010	0.739 ± 0.010	0.745 ± 0.009	0.741 ± 0.008
	RSF	0.739 ± 0.012	0.673 ± 0.013	0.754 ± 0.012	0.745 ± 0.012	0.748 ± 0.010	0.748 ± 0.010
	DeepSurv (MLP)	0.647 ± 0.018	0.610 ± 0.035	0.740 ± 0.013	0.726 ± 0.009	0.739 ± 0.010	0.733 ± 0.010
	DeepSurv (linear)	0.587 ± 0.021	0.559 ± 0.063	0.638 ± 0.044	0.679 ± 0.022	0.681 ± 0.037	0.720 ± 0.011
0.7	Cox	0.579 ± 0.014	0.627 ± 0.017	0.709 ± 0.011	0.688 ± 0.010	0.709 ± 0.009	0.703 ± 0.010
	RSF	0.693 ± 0.012	0.634 ± 0.016	0.709 ± 0.012	0.699 ± 0.010	0.708 ± 0.011	0.707 ± 0.011
	DeepSurv (MLP)	0.582 ± 0.016	0.603 ± 0.035	0.697 ± 0.010	0.666 ± 0.013	0.702 ± 0.011	0.694 ± 0.016
	DeepSurv (linear)	0.552 ± 0.015	0.559 ± 0.052	0.608 ± 0.026	0.627 ± 0.023	0.662 ± 0.034	0.679 ± 0.020
0.9	Cox	0.525 ± 0.016	0.534 ± 0.018	0.623 ± 0.007	0.591 ± 0.008	0.630 ± 0.006	0.619 ± 0.009
	RSF	0.612 ± 0.016	0.551 ± 0.013	0.618 ± 0.015	0.603 ± 0.013	0.628 ± 0.013	0.628 ± 0.013
	DeepSurv (MLP)	0.540 ± 0.015	0.543 ± 0.024	0.618 ± 0.018	0.565 ± 0.013	0.622 ± 0.014	0.620 ± 0.014
	DeepSurv (linear)	0.517 ± 0.013	0.508 ± 0.027	0.587 ± 0.033	0.537 ± 0.019	0.602 ± 0.017	0.612 ± 0.016

Table D.3: Concordance index (test) on the URRAH dataset with the TabPFN backbone, as a function of the injected missing rate, for every survival head and every method.

Rate	Head	Constant	Embeddings	Iterative	KNN	Mean	Median
0.1	Cox	0.783 ± 0.010	0.838 ± 0.006	0.822 ± 0.008	0.821 ± 0.008	0.819 ± 0.008	0.819 ± 0.008
	RSF	0.839 ± 0.008	0.814 ± 0.010	0.846 ± 0.007	0.845 ± 0.007	0.843 ± 0.008	0.843 ± 0.008
	DeepSurv (MLP)	0.803 ± 0.027	0.598 ± 0.116	0.835 ± 0.009	0.822 ± 0.029	0.829 ± 0.010	0.801 ± 0.054
	DeepSurv (linear)	0.758 ± 0.016	0.831 ± 0.011	0.813 ± 0.013	0.811 ± 0.014	0.805 ± 0.012	0.806 ± 0.013
0.3	Cox	0.748 ± 0.011	0.821 ± 0.006	0.809 ± 0.008	0.804 ± 0.009	0.801 ± 0.009	0.800 ± 0.009
	RSF	0.819 ± 0.008	0.780 ± 0.015	0.833 ± 0.008	0.827 ± 0.008	0.828 ± 0.008	0.826 ± 0.008
	DeepSurv (MLP)	0.768 ± 0.024	0.635 ± 0.117	0.806 ± 0.021	0.791 ± 0.031	0.784 ± 0.035	0.792 ± 0.022
	DeepSurv (linear)	0.722 ± 0.014	0.817 ± 0.007	0.794 ± 0.013	0.793 ± 0.012	0.789 ± 0.012	0.788 ± 0.012
0.5	Cox	0.724 ± 0.012	0.797 ± 0.008	0.792 ± 0.009	0.786 ± 0.009	0.780 ± 0.009	0.778 ± 0.010
	RSF	0.791 ± 0.010	0.766 ± 0.012	0.817 ± 0.008	0.811 ± 0.008	0.804 ± 0.008	0.802 ± 0.009
	DeepSurv (MLP)	0.735 ± 0.030	0.585 ± 0.097	0.789 ± 0.020	0.758 ± 0.050	0.760 ± 0.032	0.768 ± 0.024
	DeepSurv (linear)	0.688 ± 0.018	0.788 ± 0.019	0.782 ± 0.011	0.777 ± 0.012	0.765 ± 0.013	0.761 ± 0.012
0.7	Cox	0.695 ± 0.011	0.765 ± 0.011	0.755 ± 0.010	0.744 ± 0.010	0.750 ± 0.010	0.747 ± 0.011
	RSF	0.745 ± 0.012	0.744 ± 0.015	0.780 ± 0.010	0.762 ± 0.012	0.762 ± 0.009	0.758 ± 0.008
	DeepSurv (MLP)	0.676 ± 0.042	0.589 ± 0.093	0.744 ± 0.025	0.723 ± 0.029	0.734 ± 0.023	0.735 ± 0.024
	DeepSurv (linear)	0.657 ± 0.016	0.760 ± 0.012	0.741 ± 0.017	0.727 ± 0.017	0.736 ± 0.017	0.734 ± 0.017
0.9	Cox	0.631 ± 0.011	0.686 ± 0.009	0.686 ± 0.011	0.650 ± 0.009	0.689 ± 0.009	0.682 ± 0.010
	RSF	0.655 ± 0.012	0.648 ± 0.023	0.683 ± 0.008	0.646 ± 0.010	0.670 ± 0.010	0.667 ± 0.010
	DeepSurv (MLP)	0.609 ± 0.017	0.537 ± 0.055	0.666 ± 0.024	0.622 ± 0.036	0.667 ± 0.022	0.664 ± 0.024
	DeepSurv (linear)	0.600 ± 0.013	0.662 ± 0.028	0.667 ± 0.017	0.627 ± 0.015	0.675 ± 0.010	0.668 ± 0.012

Table D.4: Concordance index (test) on the URRAH dataset with the TabICL backbone, as a function of the injected missing rate, for every survival head and every method.

Rate	Head	Constant	Embeddings	Iterative	KNN	Mean	Median
0.1	Cox	0.783 ± 0.010	0.758 ± 0.009	0.822 ± 0.008	0.821 ± 0.008	0.819 ± 0.008	0.819 ± 0.008
	RSF	0.839 ± 0.008	0.742 ± 0.014	0.846 ± 0.007	0.845 ± 0.007	0.843 ± 0.008	0.843 ± 0.008
	DeepSurv (MLP)	0.803 ± 0.027	0.734 ± 0.028	0.835 ± 0.009	0.822 ± 0.029	0.829 ± 0.010	0.801 ± 0.054
	DeepSurv (linear)	0.758 ± 0.016	0.675 ± 0.043	0.813 ± 0.013	0.811 ± 0.014	0.805 ± 0.012	0.806 ± 0.013
0.3	Cox	0.748 ± 0.011	0.750 ± 0.008	0.809 ± 0.008	0.804 ± 0.009	0.801 ± 0.009	0.800 ± 0.009
	RSF	0.819 ± 0.008	0.701 ± 0.012	0.833 ± 0.008	0.827 ± 0.008	0.828 ± 0.008	0.826 ± 0.008
	DeepSurv (MLP)	0.768 ± 0.024	0.708 ± 0.037	0.806 ± 0.021	0.791 ± 0.031	0.784 ± 0.035	0.792 ± 0.022
	DeepSurv (linear)	0.722 ± 0.014	0.598 ± 0.092	0.794 ± 0.013	0.793 ± 0.012	0.789 ± 0.012	0.788 ± 0.012
0.5	Cox	0.724 ± 0.012	0.712 ± 0.011	0.792 ± 0.009	0.786 ± 0.009	0.780 ± 0.009	0.778 ± 0.010
	RSF	0.791 ± 0.010	0.665 ± 0.016	0.817 ± 0.008	0.811 ± 0.008	0.804 ± 0.008	0.802 ± 0.009
	DeepSurv (MLP)	0.735 ± 0.030	0.656 ± 0.066	0.789 ± 0.020	0.758 ± 0.050	0.760 ± 0.032	0.768 ± 0.024
	DeepSurv (linear)	0.688 ± 0.018	0.575 ± 0.084	0.782 ± 0.011	0.777 ± 0.012	0.765 ± 0.013	0.761 ± 0.012
0.7	Cox	0.695 ± 0.011	0.674 ± 0.015	0.755 ± 0.010	0.744 ± 0.010	0.750 ± 0.010	0.747 ± 0.011
	RSF	0.745 ± 0.012	0.655 ± 0.013	0.780 ± 0.010	0.762 ± 0.012	0.762 ± 0.009	0.758 ± 0.008
	DeepSurv (MLP)	0.676 ± 0.042	0.596 ± 0.057	0.744 ± 0.025	0.723 ± 0.029	0.734 ± 0.023	0.735 ± 0.024
	DeepSurv (linear)	0.657 ± 0.016	0.549 ± 0.049	0.741 ± 0.017	0.727 ± 0.017	0.736 ± 0.017	0.734 ± 0.017
0.9	Cox	0.631 ± 0.011	0.553 ± 0.014	0.686 ± 0.011	0.650 ± 0.009	0.689 ± 0.009	0.682 ± 0.010
	RSF	0.655 ± 0.012	0.568 ± 0.012	0.683 ± 0.008	0.646 ± 0.010	0.670 ± 0.010	0.667 ± 0.010
	DeepSurv (MLP)	0.609 ± 0.017	0.567 ± 0.030	0.665 ± 0.023	0.621 ± 0.036	0.672 ± 0.014	0.664 ± 0.023
	DeepSurv (linear)	0.600 ± 0.014	0.493 ± 0.044	0.669 ± 0.014	0.627 ± 0.015	0.674 ± 0.012	0.667 ± 0.013

Bibliography

[1] T. G. Clark, M. J. Bradburn, S. B. Love, and D. G. Altman, "Survival analysis part i: Basic concepts and first analyses," *British journal of cancer*, vol. 89, no. 2, pp. 232–238, 2003.

[2] M. J. Bradburn, T. G. Clark, S. B. Love, and D. G. Altman, "Survival analysis part ii: Multivariate data analysis–an introduction to concepts and methods," *British journal of cancer*, vol. 89, no. 3, pp. 431–436, 2003.

[3] M. J. Bradburn, T. G. Clark, S. B. Love, and D. G. Altman, "Survival analysis part iii: Multivariate data analysis–choosing a model and assessing its adequacy and fit," *British journal of cancer*, vol. 89, no. 4, pp. 605–611, 2003.

[4] T. G. Clark, M. J. Bradburn, S. B. Love, and D. Altman, "Survival analysis part iv: Further concepts and methods in survival analysis," *British journal of cancer*, vol. 89, no. 5, pp. 781–786, 2003.

[5] D. R. Cox, "Regression models and life-tables," *Journal of the royal statistical society: Series B (methodological)*, vol. 34, no. 2, pp. 187–202, 1972.

[6] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger, "Deepsurv: Personalized treatment recommender system using a cox proportional hazards deep neural network," *BMC medical research methodology*, vol. 18, no. 1, p. 24, 2018.

[7] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, "Random survival forests," 2008.

[8] E. L. Kaplan and P. Meier, "Nonparametric estimation from incomplete observations," *Journal of the American statistical association*, vol. 53, no. 282, pp. 457–481, 1958.

[9] S. Wiegrebe, P. Kopper, R. Sonabend, B. Bischl, and A. Bender, "Deep learning for survival analysis: A review," *Artificial Intelligence Review*, vol. 57, no. 3, p. 65, 2024.

[10] C. Lee, W. Zame, J. Yoon, and M. Van Der Schaar, "Deephit: A deep learning approach to survival analysis with competing risks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[11] H. Kvamme, Ø. Borgan, and I. Scheel, "Time-to-event prediction with neural networks and cox regression," *Journal of machine learning research*, vol. 20, no. 129, pp. 1–30, 2019.

[12] R. Bommasani et al., "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[13] A. Vaswani et al., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[15] T. Brown et al., "Language models are few-shot learners (gpt-3)," *arXiv preprint arXiv:2005.14165*, vol. 75, 2020.

[16] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" *Advances in neural information processing systems*, vol. 35, pp. 507–520, 2022.

[17] N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter, "Tabpfn: A transformer that solves small tabular classification problems in a second," *arXiv preprint arXiv:2207.01848*, 2022.

[18] N. Hollmann et al., "Accurate predictions on small data with a tabular foundation model," *Nature*, vol. 637, no. 8045, pp. 319–326, 2025.

[19] J. Qu, D. HolzmÃžller, G. Varoquaux, and M. L. Morvan, "Tabicl: A tabular foundation model for in-context learning on large data," *arXiv preprint arXiv:2502.05564*, 2025.

[20] D. I. Kim, W. S. Lai, and K. W. Zhang, "Tabular foundation models can do survival analysis," *arXiv preprint arXiv:2601.22259*, 2026.

[21] D. Seletkov, P. Hager, R. Braren, D. Rueckert, and R. Rehms, "Survival in-context: Prior-fitted in-context learning tabular foundation model for survival analysis," *arXiv e-prints*, arXiv–2603, 2026.

[22] M.-K. Pham, L. Cotugno, A. Sirbu, T. T. Mai, M. Crane, and M. Bezbradica, "Tabular foundation models for clinical survival analysis via survival-aware adaptation," *arXiv preprint arXiv:2606.12006*, 2026.

[23] J. P. Klein and M. L. Moeschberger, *Survival analysis: techniques for censored and truncated data*. Springer, 2003, vol. 1230.

[24] S. Müller, N. Hollmann, S. P. Arango, J. Grabocka, and F. Hutter, "Transformers can do bayesian inference," *arXiv preprint arXiv:2112.10510*, 2021.

[25] F. E. Harrell, R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati, "Evaluating the yield of medical tests," *Jama*, vol. 247, no. 18, pp. 2543–2546, 1982.

[26] L. Antolini, P. Boracchi, and E. Biganzoli, "A time-dependent discrimination index for survival data," *Statistics in medicine*, vol. 24, no. 24, pp. 3927–3944, 2005.

[27] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[28] A. Pingitore et al., "Machine learning to identify a composite indicator to predict cardiac death in ischemic heart disease," *International journal of cardiology*, vol. 404, p. 131 981, 2024.

[29] G. Desideri, A. Virdis, E. Casiglia, and C. Borghi, "Exploration into uric and cardiovascular disease: Uric acid right for heart health (urrah) project, a study protocol for a retrospective observational study," *High Blood Pressure & Cardiovascular Prevention*, vol. 25, no. 2, pp. 197–202, 2018.

[30] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.