Survival Prediction using Artificial Neural Networks on Censored Data

Elvire Roblin

▶ **To cite this version:**

Elvire Roblin. Survival Prediction using Artificial Neural Networks on Censored Data. Cancer. Université Paris-Saclay, 2023. English. NNT : 2023UPASL028 . tel-04194460

HAL Id: tel-04194460
https://theses.hal.science/tel-04194460

Submitted on 3 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université
PARIS-SACLAY

Survival Prediction using Artificial Neural Networks on Censored Data

Développements des réseaux de neurones pour données censurées: prédiction de survie associée à une mesure d'incertitude et recommandation de traitement

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 582: cancérologie : biologie - médecine - santé (CBMS)
Spécialité de doctorat: Sciences du cancer
Graduate School : Life Science and Health
Référent : Faculté de médecine

Thèse préparée dans les unités de recherche INSERM UMR 1018/CESP Centre de recherche en épidémiologie et en santé des populations, sous la direction de Stefan MICHIELS, PhD-HDR, et Mathématiques et Informatique pour la Complexité et les Systèmes (Université Paris-Saclay, CentraleSupélec) sous la co-direction de Paul-Henry COURNEDE, PhD-HDR et Professeur

Thèse soutenue à Paris-Saclay, le 21 mars 2023, par

Elvire ROBLIN

Membres du jury avec voix délibérative

Hugues Talbot Professeur, CentraleSupelec, Université Paris Saclay	Président
Anne-Laure Boulesteix Professeure des universités, Medizinische Fakultät – Ludwig Maximilians Universität München	Rapporteure & Examinatrice
Pascal Roy Professeur des universités - praticien hospitalier, Université Claude Bernard Lyon 1	Rapporteur & Examinateur
Axel Benner Directeur de recherche, Deutsches Krebsforschungszentrum	Examinateur
Angelina Roche Maître de conférences, CEREMADE, Université Paris Dauphine	Examinatrice

THESE DE DOCTORAT
NNT : 2023UPASL028

Remerciements

Qui aurait cru qu'un stage de journalisme de troisième à l'Agence France Presse puisse mener à une thèse en statistiques à l'IGR ? Avec les sœurs Kandel, tout est possible! Merci à Valentine et Marguerite, sans qui je ne serais pas là aujourd'hui.

Stefan, merci infiniment de m'avoir permis de réaliser cette thèse. Merci pour tes enseignements précieux, ta disponibilité et ton écoute. Grâce à toi, j'ai pu découvrir le monde de l'hôpital et de la recherche en biostatistiques. Maintenant, je peux le dire, dans la tête des chercheurs, "ça fume !". Paul-Henry, merci de m'avoir ouvert les portes du MICS et de m'avoir apporté un regard complémentaire à celui de Stefan. Merci pour ton excellence et ta rigueur scientifiques. Notre trio de travail a dû apprendre à fonctionner entre deux laboratoires et dans un contexte de pandémie mondiale : l'aboutissement de ce travail montre que nous avons réussi à surmonter ces difficultés.

Merci beaucoup au SBE (oui oui, le nom historique), à Alicia pour son soutien inconditionnel et quotidien (j'espère que tu vas t'amuser dans ta nouvelle vie professionnelle !), à Matthieu un assistant comme on en fait plus, à Nusaibah la commentatrice sportive et musicale capable de passer de Djoko à Beyoncé, à Sarah Flora la commandante et surtout enquêtrice du navire 72, à "Taznim" la maman un petit peu sévère vérifiant avec bienveillance que ma thèse avance, à Isabelle ma répétitrice Pufff anti-pacer (et c'est très important de savoir ralentir un petit peu des fois), à Yves et sa guitare toujours prêt à pousser la chansonnette...

Merci au MICS, mon histoire avec ce laboratoire a commencé dans les meilleures conditions possibles, par un séminaire à Corfou... Merci à Mathilde et Mahmoud, ou plus précisément la Team Kwak, pour leur soutien quotidien et leur aide, à Sylvain le brasseur : hâte de continuer à découvrir toutes les bonnes adresses de FRG ensemble. Dans la team Last doctorants, je compte aussi Stefania, namasté ! Après avoir commencé nos thèses le même jour, nous voyons le bout du tunnel, hâte de fêter ça entre docteurs...

Mes débuts à l'IGR sont aussi marqués par mon travail avec l'équipe du DITEP : Héloïse, merci pour ces journées de télétravail où nous avons dégusté toutes les meilleures pâtisseries de Paris, désolée Sandrine pour toutes les fois où j'ai reporté nos réunions, merci à Séverine, la spécialiste du "gating" pour m'avoir fait découvrir le vocabulaire et le travail des biologistes, ChifChif et nos pauses café-potins à 15h en planifiant un voyage hypothétique à Zanzibar, Delphine et Fx je vous attends pour une prochaine soirée surprises.

Merci à Ariane de m'avoir toujours soutenue. Je quitte enfin la table de la salle à manger avec mes cahiers, et promis, c'est fini les études! Arthur et Lucile, ye hopeula la thèse c'est fini, comme on dit là-bas, dans la ville de mes pas origines. Mathilde et Rébecca, depuis notre rencontre en CP, j'ai pas mal progressé en calcul, j'espère que je vous aurai toujours à mes côtés pour m'aider et me soutenir. Très heureuse d'avoir vécu cette thèse avec mes différents collocs : Hugo, un consultant en arrêt de travail relecteur hors pair de toutes mes présentations, Zaza la petite maman et avec qui je pouvais faire des séances d'abdo avant même 7h du matin, Dianou qui m'a appris à faire la sieste

comme il se doit... Merci à Dominique et Isabelle, les coronas brothers, pour leur accueil pendant le premier confinement, à tonton Philippe et tata Christine pour leur soutien sportif, à Emilie et Fred qui m'ont ouvert leur coin de paradis à Champsecret.

Merci aux copains de l'ENSAE, en particulier Kolia et Victor pour leurs aides statistiques ponctuelles mais très utiles, à Bressounou pour le soutien moral et sportif, à SamSam pour son offre de relecture. Non, Anne-Camille et Coline, je ne finirai pas ma thèse en 2027! Merci à Benjamax qui m'a montré la voie de la thèse après m'avoir entraîné sur la côte ouest des Etats-Unis. Yannis, à quand ta thèse?

Je finirais par un hommage à Christiane, venue malgré elle me soutenir à l'IGR, et qui me répétait pour m'encourager: "bientôt la quille, bordel!".

Scientific Production

Articles

- Roblin, E., Cournède, P.-H., Michiels, S.. Confidence intervals of survival predictions with neural networks trained on molecular data. Article Submitted
- E. Roblin et al. "On the Use of Neural Networks with Censored Time-to-Event Data". In: *Mathematical and Computational Oncology*. Ed. by G. Bebis et al. Cham: Springer International Publishing, 2020, pp. 56–67. ISBN: 978-3-030-64511-3. DOI: 10.1007/978-3-030-64511-3_6

Oral Presentations

- **ISCB 43:** Roblin, E.; Cournede P.H.; Michiels S. Uncertainty measures of survival predictions with neural networks applied to molecular data. Presented at the 43rd Annual Conference of the International Society for Clinical Biostatistics; August 2022; Newcastle, UK.
- **EPICLIN 16:** Roblin E., Cournède P.H., Michiels S., Prédiction de survie avec réseaux de neurones et mesures d'incertitude, Revue d'Épidémiologie et de Santé Publique, Volume 70, Supplement 2, 2022, Pages S84-S85, ISSN 0398-7620, doi:10.1016/j.respe.2022.03.116.
- **ISCB 42:** Roblin, E.; Cournede P.H.; Michiels S. Survival Predictions and Uncertainty Measures with Censored Data. Presented at the 42nd Annual Conference of the International Society for Clinical Biostatistics; July 2021; Lyon, France.
- **EPICLIN 15:** Roblin, E.; Cournede P.H.; Michiels S. Développements de réseaux de neurones pour données censurées et prédiction de survie. Presented at Epiclin ; June 2021 ; Marseille, France.
- **ISMCO:** Roblin, E.; Cournede P.H.; Michiels S. On the use of neural networks with censored tome-to-event data. Presented at the 2nd International Symposium on Mathematical and Computational Oncology; October 2020; San Diego, CA

Poster Presentation

- **ISCB 41:** Roblin E.; Cournede P.H.; Michiels S. On the use of neural networks for survival models with censored data. Poster presented at the 41st Annual Conference of the International Society for Clinical Biostatistics; August 2020; Krakow, Polen.

Content

Figures

Tables

Abbreviations

AE	Autoencoder
AFT	Accelerated Failure Time
Adam	Adaptative Moment Estimation
ATE	Average Treatment Effect
Boot	Bootstrap
BS	Brier Score
C-statistic	Concordance Index
CVAE	Conditional Variational Autoencoder
CI	Confidence Interval
CR	Coverage Rate
CV	Cross-Validation
CHF	Cumulative Hazard Function
CoxPH	Cox Proportional-Hazards Model
DeepEns	Deep Ensemble
ER	Estrogen Receptor
FNN	Feedforward Neural Network
FBMask	Fixed Bernoulli Mask
HR	Hazard Ratio
Her2	Human epidermal growth factor 2
tanh	Hyperbolic Tangent
iid	independent and identically distributed
IBS	Integrated Brier Score
IPCW	Inverse Probability of Censoring Weighting
IF	Interaction Forest
KM	Kaplan Meier
KL	Kullback-Leibler
LASSO	Least Absolute Shrinkage and Selection Operator
LCE	Lung Cancer Explorer
METABRIC	Molecular Taxonomy of Breast Cancer International Consortium
MCDrop	Monte-Carlo Dropout
OOB	Out-Of-Bag
PLANN	Partial Logistic Artificial Neural Network
PMF	Probability Mass Function
PR	Progesterone Receptor
RCT	Randomized Controlled Trial
RSF	Random Survival Forest
ReLU	Rectified Linear Unit
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
VAE	Variational Autoencoder

1 - Introduction

1.1 . Context

Traditionally, practitioners have adopted a one-size-fits-all approach to assess treatments to cure large groups of patients with a similar disease and analogous symptoms under the assumption that treatment effects were broadly similar in different individuals. A paradigm change occurred in 1953 with the discovery of the double helix structure of deoxyribonucleic acid (DNA) by James Watson and Francis Crick. It marked a milestone in the genomic revolution in medicine. In 1977, Sanger et al. (1977) first introduced a method enabling DNA sequencing. It made possible the Human Genome Project, an international program whose goal was the complete mapping and understanding of all the $20,500$ genes of human beings (Venter et al., 2001); (Lander et al., 2001). This work showed that humans are 99.9% identical in their genetic makeup. The differences in the remaining 0.1% hold clues about both the causes of diseases and why people can respond differently to treatments. DNA microarrays (Schena et al., 1995) appeared in the 1990s, and constituted the first large-scale technology giving access to the expression profile of genes. In the 2000s, a new technology based on next-generation sequencing, RNA-Seq, emerged (Margulies et al., 2005). RNA-seq is now frequently used in oncology to characterize the transcriptome of tumor cells, with the objective of a more specific and personalized understanding of cancer.

RNAseq data provides over $20,000$ gene expressions (even more at the transcript scale). For medical use, it is important to identify among these variables those of interest with regard to some specific clinical questions. They are called biomarkers. According to the Biomarkers Definitions Working Group (Atkinson et al., 2001), a biomarker is "a characteristic that is objectively measured and evaluated as an indicator of normal biological processes, pathogenic processes, or pharmacologic responses to a therapeutic intervention." Biomarkers are usually divided into two groups: prognostic and predictive biomarkers. A prognostic biomarker is associated with the clinical outcome independently of treatment and predicts the likely course of a disease in a defined clinical population. It can discriminate between patients according to their risk level. In oncology, one example is the estrogen receptor (ER) or progesterone receptor (PR) and human epidermal growth factor receptor 2 (Her2) for breast cancer. Several biomarkers can also be grouped into a prognostic gene expression signature, for example, the MammaPrint microarray-based signature of 70 genes developed by the Netherlands Cancer Institute (Veer et al., 2002). It is a prognostic marker for the occurrence of distant metastases in women with early breast cancer: patients with a poor-prognosis MammaPrint signature have a much higher risk of distant metastases within 5 years compared with patients with a good prognosis signature. A predictive biomarker (also called treatment-effect modifier) forecasts how individual patients will respond to specific treatments. Predictive biomarkers can predict the difference in clinical outcome between two treatment arms, if any. This interaction with the treatment effect allows the selection of subgroups of patients likely to benefit from treatment. The relative effect of the treatment varies with the value of the biomarker. An example of a predictive biomarker is the hormone-receptor status that predicts the response to endocrine therapies in breast cancer. There also exist signatures combining several predictive biomarkers, such as signatures of 14 biomarkers predictive of the effect of trastuzumab in breast cancer (Pogue-Geile et al., 2013).

Precision medicine is built upon these predictive biomarkers. It is defined by the European Society for Medical Oncology (Yates et al., 2018) as a "healthcare approach with the primary aim of identifying which interventions are likely to be of most benefit to which patients based upon the features of the individual and their disease." It focuses on creating a detailed picture of the patient and finding a tailored treatment for this patient. To demonstrate the predictive effect of a biomarker in the context of precision medicine, it is necessary to have data with a large number of patients (often more than 500 patients) in a randomized clinical trial. Clinical trials have revolutionized medicine by providing reliable evidence of the efficacy and safety of new treatments. They are conducted to assess the efficacy of new treatment regimens and are the foundation of clinical research in oncology. Following the definition of Kelly et al. (2010), clinical trials are scientific research that evaluates the efficacy and safety of a new therapeutic approach in a defined group of patients. They can change medical practice, improve patients' quality of life, and improve overall survival or recurrence-free survival, etc. Many cancer clinical trials involve following patients over a long period of time, from a few months to several years. The primary endpoint in these studies may be the occurrence of death, the development of an adverse reaction, the recurrence of the disease, the side effects of treatment, or the development of a new disease. A surrogate endpoint may replace the clinical endpoint with a faster evaluation of the effect of the experimental treatment. During the follow-up period, the patient's life data are collected and evaluated. The variable of interest in these data is most commonly of time-to-event nature, i.e., the time between the patient's inclusion in the study and a subsequent event. Recently, new approaches based on organoids have been developed in precision medicine (Boileve et al., 2020). These self-organized three-dimensional tissue cultures are derived from primary cells, and can replicate the complexity of an organ. It gives scientists the opportunity to mimic patient-specific disease development and see how drugs interact with these mini-organs.

Survival analysis was specifically developed to analyze time-to-event data, usually using patient clinical data as inputs. It consists in analyzing the time until an event occurs (Singh et al., 2011). Common events studied are death, disease, relapse, and recovery. It incorporates censorship, in which data about the event of interest is unknown because of the withdrawal of the patient from the study. The survival function gives the probability that a patient survives longer than some specified time point. It can be depicted with a Kaplan-Meier (KM) curve. Several statistical methods are available to analyze time-to-event data with respect to several factors simultaneously. The most frequently used approach is the Cox Proportional-Hazards (CoxPH) model (Cox, 1972), a regression method for survival data, which simultaneously evaluates the effect of several variables on survival. It is a semi-parametric method, as the baseline hazard function does not need to be specified. The model is based on two main assumptions: the hazard curves for the patients groups defined by variable levels are proportional; the relationship between the log hazard and each variable is linear.

With the availability of RNA-seq data and the interest of finding genes or gene signatures as interesting biomarkers, these molecular data can be used along clinical data in survival models. High-dimensional data refers to a dataset in which the number of features is larger than the number of observations. As mentioned above, RNA-seq data provides over $20,000$ gene expressions. Meanwhile, the sample size in clinical studies is generally limited (typically a few hundred). In the framework of high dimensional data, the hypotheses of the CoxPH model are restrictive, for

instance, when nonlinear and complex interactions exist in the data. Hence, usual methods can no longer be applied, and new approaches are necessary. In this context, statistical learning methods have become a popular tool for medical researchers. These techniques can discover and identify patterns and relationships between biomarkers for complex datasets and effectively predict future outcomes. Here, we will focus on artificial neural networks that can model complex patterns. A feedforward neural network (FNN), also known as a multi-layer perceptron, can be seen as a "series of logistic regression models stacked on top of each other" (Murphy, 2012). This is a biologically inspired classification algorithm. The first artificial neural network, called perceptron, was developed by Rosenblatt (1958), and the first gradient-based FNN was proposed by Rumelhart et al. (1986). It is one of the most popular algorithms with successful applications in many fields, from physics to finance to health. The power of the FNN comes from its architecture, defined by the number of its layers, the number of nodes per layer, the nonlinear activation functions, and adaptive weights. The popularity of neural networks has kept growing since the end of the 1980s, and the data considered is more and more complex, from images to time series. As they are flexible non-linear models, they are particularly relevant when a high number of candidate variables and complex interactions are to be evaluated. Neural networks have been introduced in survival analysis. The extension of the CoxPH regression with neural networks was first proposed by Faraggi et al. (1995), who replaced the linear predictor of the CoxPH model with a one hidden layer FNN. Nowadays, several variants of FNNs in survival analysis exist, depending on the choice of the structure of the neural network and the degree of complexity. Other statistical learning methods have been adapted to survival, including Random Survival Forests (RSF) (Ishwaran et al., 2007).

Machine learning models, specifically neural networks, are often referred to as black boxes, models that are hard to interpret directly. There is a need for interpretability when applying these models in clinical settings, which has raised a huge literature in the last years (Molnar et al., 2020); (Charachon et al., 2022). Another important issue when using a predictive model is to measure its trustworthiness, that can be evaluated by quantifying the level of uncertainty. Indeed, these deep models can only be applied to make a medical diagnosis at the patient level or to develop decision tools in various healthcare settings with an associated uncertainty measure. Ideally, a survival model should achieve good predictive performances and provide a well-calibrated measure of uncertainty associated with survival predictions. If survival prediction models are to be deployed in clinical settings, such uncertainty measures will make the model interpretable and trustworthy. Krzywinski et al. (2013) recall the "importance of being uncertain" and of quantifying this uncertainty. The predictive uncertainty stems from two sources that are aleatoric and epistemic uncertainties. Aleatoric uncertainty comes from noisy data, for example, due to measurement imprecision. Epistemic uncertainty, also called model uncertainty, is the uncertainty associated with the estimated model parameters, the model structure, the specification of the model, or even the choice of the model. Many different uncertainty quantification techniques exist, but they need to be adapted to deep learning models.

Personalized treatment recommendations are predominantly informed by results from randomized controlled trials (RCT), using the average relative treatment effect to decide on how to treat patients. Determining the most beneficial treatment for an individual amounts to identifying the effect of the treatment on the outcome variable (e.g. survival time). The potential outcomes framework (Rubin, 1974) defines the individual treatment effect (ITE) as the difference between potential

outcomes of distinct treatment alternatives. Classical statistical models like CoxPH can be used to identify the ITE but need an a priori specification of treatment interaction terms. Statistical or machine learning models have become increasingly popular to estimate treatment effects at the individual level. To do so, clinical prediction models forecast patient's outcomes under different treatment conditions. The difficulty is that standard models are not directly designed for comparing outcomes under different treatment regimens.

1.2 . Outline

In this context, we explore different ways of applying FNNs to time-to-event data in high-dimensional settings. The main objectives of this thesis are the followings:

1. Compare different FNNs adapted to time-to-event data and propose various ways of handling censoring;

2. Develop measures of uncertainty associated with the predictions constructed with an ensemble of FNNs;

3. Develop and evaluate treatment recommendation methods using FNNs.

The first part of this work consists in investigating the prediction ability of neural network models with time-to-event data, using specific ways to handle censored observations such as specific loss functions, or pseudo-observations, and studying their operating characteristics. In a continuous time framework, two models are implemented: CoxCC (Kvamme et al., 2019) uses a special loss based on a case-control approximation; CoxTime is an extension of CoxCC that includes the time variable. Two models are defined in a discrete-time framework: Biganzoli et al. (1998) include censoring in their neural network model using a piecewise exponential function, and use a cross-entropy loss function; DeepHit (Lee et al., 2018) is a model that estimates the probability mass function and combines a log-likelihood with a ranking loss. We also propose and implement FNNs that rely on pseudo-observations, a specific way to handle incomplete data. Two new ways of defining pseudo-observations are introduced and compared to one existing definition proposed in a discrete-time framework (Zhao et al., 2019). We explore ways of reducing the dimensionality of the data using neural network structures, namely autoencoders and variational autoencoders, to see if there is an improvement in the performances of the FNNs. The FNN models are benchmarked with RSF and a penalized CoxPH with the Least Absolute Shrinkage and Selection Operator penalty or LASSO (Tibshirani, 1996). These two models are chosen as they are credible competitors to FNNs in a high-dimensional setting.

In the second part, we build confidence intervals at the patient level to quantify the degree of certainty in the model's survival predictions using an ensemble of FNNs. Here, we focus on specific aspects of epistemic uncertainty and build an ensemble of neural networks to associate an uncertainty measure with expected survival predictions, adapting the quantification of uncertainty associated with survival predictions using confidence intervals. We compare existing ensemble methods by introducing randomness and applying them to time-to-event data. With Bootstrap (Efron, 1979), a training subset is randomly drawn with replacement to obtain M FNNs. To form a Deep Ensemble (Lakshminarayanan et al., 2016) of FNNs, the parameters of the M FNNs are randomly

initialized during training. Gal et al. (2016) propose to apply M dropout masks randomly during the test phase: this is Monte-Carlo Dropout (MCDrop) (Gal et al., 2016). The idea of the fixed Bernoulli mask (FBMask) (Mancini et al., 2020) is to previously generate M dropout masks applied at the time of model training and testing. Each ensemble method is built on different FNN survival predictors (either CoxCC, CoxTime, or DeepHit). With the ensemble methods, we associate confidence intervals (CI) with survival predictions. We measure the quality of these CIs using the coverage rate (CR).

In the third part, we focus on the possibility of formulating individualized treatment recommendations based on patient characteristics in the context of a RCT. We formulate a recommendation treatment rule using FNNs and inspired by the work of Katzman et al. (2016). It is built on the difference in the expected probability at a fixed time point of an event in treatment minus that in control. In this part, the FNNs are benchmarked with LASSO and Interaction Forests (IF) (Hornung et al., 2022), a specific type of RSF that models interaction effects using bivariate splits.

Two types of data are used: simulated data and real patient cohorts. Synthetic data are simulated to explore the operating characteristics of the different models. They are simulated based either on a CoxPH model or an Accelerated Failure Time (AFT) model. With the AFT model, some simulations include a random function generator to simulate nonlinear data, including high-order interactions. 2 high-dimensional patient cohorts are used for illustration in Chapters 4 and 5. The METABRIC (Molecular Taxonomy of Breast Cancer International Consortium) breast cancer data include $1,960$ patients, 6 clinicopathological variables, and the expression of $1,000$ genes. The Lung Cancer Explorer (LCE) consists of $4,120$ patients, 3 clinical variables, and $1,000$ genes. For Chapter 5, we use data from a RCT including $1,574$ patients and promoted by the National Cancer Institute / National Institute of Health, which evaluates the effect of adjuvant trastuzumab in breast cancer. The expression of 462 genes is available in addition to clinicopathological variables such as ER and lymph node status, or tumor size.

2 - Survival Analysis

2.1 . Time-to-event data

Survival or time-to-event analysis corresponds to the time to occurrence of a particular event for one or several groups of individuals. This event is associated with a change of status, e.g., the death of an individual from a specific cause. Survival data analysis is used in the context of longitudinal studies; in this work, it is applied to clinical trials in a high-dimensional context.

Multivariable survival analysis leads to the identification of variables that influence the survival time. These are either prognostic variables, i.e., criteria that predict the patient's future regardless of the treatment applied, or predictive biomarkers, which predict the effectiveness of the proposed treatment. The CoxPH model (Cox, 1972) is the reference model applied in survival analysis. It relies on the partial likelihood for parameter estimation.

Let the random variable $T \in \mathbb{R}^+$ represent the survival time, that is the time between the starting point and the occurrence of a given event, e.g., the time between a patient's treatment assignment and death. Often, T is not observed for all individuals: time-to-event data is censored. For example, when the event studied is death, the occurrence of this event is not observed for subjects still alive at the end of the study. Survival data is said to be right-censored if the individual leaves the study before the end point or if the event did not happen before the end of the follow-up. In this case, the only information we have is that the event time is longer than the follow-up time. In the case of right-censoring, we define $(C_i)_{i=1,\dots,n}$ the independent and identically distributed censoring times of n individuals. We can set $\{(\tilde{T}_i, X_i, D_i)\}_{i=1,\dots,n}$ where $\tilde{T}_i = \min(T_i, C_i)$ is the time until death or censoring, $X_i = (X_{i,1}, \dots, X_{i,p})^T \in \mathbb{R}^p$ denotes a p-dimensional vector of variables, and $D_i = \mathbb{1}\{\tilde{T}_i = T_i\}$ is the censoring indicator. We suppose that the survival time T_i is independent of the censoring time conditionally on the variables vector X_i for $i = 1, \dots, n$.

We want to study the survival function, that is, the probability of not experiencing the event before time t:

$$S(t) = P(T > t) = 1 - F(t). \tag{2.1}$$

It is based on $F(t)$, the probability of experiencing the event between the initial time of the study $t_0 = 0$ and t. $F(t)$ can be written using the cumulative distribution of the even time:

$$F(t) = P(T \leq t) = \int_0^t f(u)du. \tag{2.2}$$

It is an increasing function with $F(0) = 0$ and $\lim_{t \to +\infty} F(t) = 1$, and it is computed as the integral of the probability density function $f(t)$:

$$f(t) = \lim_{\Delta t \to 0^+} \frac{P(t \leq T < t + \Delta t)}{\Delta t}, \tag{2.3}$$

with the assumption that this limit exists for all t. It enables the definition of the risk function h that characterizes the distribution of the survival time T. It is written as:

$$h(t) = \frac{f(t)}{S(t)} = \lim_{\Delta t \to 0^+} \frac{P(t \leq T < t + \Delta t | T \geq T)}{\Delta t}. \tag{2.4}$$

$h(t)\Delta t$ corresponds to the probability of an individual experiencing an event between t and $t + \Delta t$ if the subject is still at risk at time t. Then the cumulative hazard is written:

$$H(t) = \int_0^t h(u)du. \tag{2.5}$$

The survival function can be retrieved through the cumulative hazard by :

$$S(t) = exp(-H(t)). \tag{2.6}$$

2.2 . Log-likelihood for censored data

In a survival model with right censoring, let's suppose that $(C_i)_{i=1,...,n}$ are fixed. It is the case, for instance, when the only subjects censored are those excluded alive at a fixed time point at the end of the study. In this case, the individual contribution to the likelihood is either $f(\tilde{T}_i)$ if the patient i experienced the event $(D_i = 1)$ or $S(\tilde{T}_i)$ if the patient's survival time is right censored $(D_i = 0)$. Censoring is said to be non-informative.

We can define the likelihood of a sample of size n as a function of the parameters of both the chosen model and the data of the sample studied. It represents the probability of observing a particular realization of this sample for these parameters, and it is equal to the product of the probabilities of each of the observations. It follows that the likelihood is written:

$$\begin{aligned}
\mathcal{L} &= \prod_{i=1}^{n} f(\tilde{T}_i)^{D_i} S(\tilde{T}_i)^{(1-D_i)} \\
&= \prod_{i=1}^{n} (S(\tilde{T}_i)h(\tilde{T}_i))^{D_i} S(\tilde{T}_i)^{1-D_i} \\
&= \prod_{i=1}^{n} h(\tilde{T}_i)^{D_i} S(\tilde{T}_i) \\
&= \prod_{i} h(\tilde{T}_i)^{D_i} exp(-H(\tilde{T}_i)).
\end{aligned}$$

Let us now consider the case where $(C_i)_{i=1,...,n}$ are random variables. In this case, the likelihood also depends on the distribution of C_i. If we suppose that $(C_i)_{i=1,...,n}$ and $(T_i)_{i=1,...,n}$ are independent for all i, the likelihood is written:

$$\mathcal{L} = \prod_{i=1}^{n} h(\tilde{T}_i)^{D_i} S(\tilde{T}_i) \prod_{i=1}^{n} h_C(\tilde{T}_i)^{1-D_i} S_C(\tilde{T}_i), \tag{2.7}$$

where h_C and S_C are the risk function and the survival function of C_i respectively.

2.3 . The Kaplan-Meier estimator, a non-parametric estimator for survival analysis

In survival analysis, non-parametric methods are smooth as they do not rely on specific assumptions concerning the data distribution. Yet they require independent and identically distributed (iid) observations. A non-parametric and empirical estimate of the survival function, the KM estimator, can be used to retrieve the survival curve representing $S(t)$. It does not rely on any assumption on

the distribution of survival times.

The KM estimator (Kaplan et al., 1958), also known as the product-limit estimator, corresponds to the non-parametric maximum likelihood estimator of the survival function $S(.)$ at the specified survival time t_j based on a sample of n individuals. It is based on the idea that the probability of not having experienced the event $P(T > t_j)$ is similar to the probability of not yet having experienced the event at t_{j-1} times the probability of not experiencing the event at t_j knowing that the event did not happen until t_{j-1}. It can be written as follow:

$$\begin{aligned} S(t_j) &= P(T > t_j) \\ &= P(T > t_j | T > t_{j-1}) \dots P(T > t_2 | T > t_1) P(T > t_1) \\ &= P_j P_{j-1} \dots P_1, \end{aligned}$$

where $P_j = P(T > t_j | T > t_{j-1})$. $t_1, \dots, t_k$ represent the distinct times at which failures are observed. The number of individuals at risk at time t_j can be written r_j, m_j is the number of individuals for whom the event occurred at time t_j, and P_j is estimated by:

$$\hat{P}_j = \frac{(r_j - m_j)}{r_j}.$$

It follows that the KM estimator of the survival probability until time t is:

$$\hat{S}(t) = \prod_{j:t_j \leq t} \frac{r_j - m_j}{r_j} = \prod_{j:t_j \leq t} \left(1 - \frac{m_j}{r_j}\right). \tag{2.8}$$

We have:

$$\hat{S}(t_{j+1}) = \hat{S}(t_j) \times \left(1 - \frac{m_{j+1}}{r_{j+1}}\right). \tag{2.9}$$

By definition, $\hat{S}(0) = 1$. $\hat{S}(t)$ is a decreasing step function, constant between two consecutive event times, continuous on the right, with a step at each observed event time. It is not defined after the last observed time if this time is censored.

2.4 . The Cox Proportional-Hazards model, a semi-parametric estimator

The CoxPH model (Cox, 1972) is the most commonly used regression method in survival analysis. In a regression model, the objective is to estimate regression coefficients to assess the strength of association between the predictors X and the outcome. Here, the CoxPH model enables us to investigate simultaneously the link between one or more risk factors and the patient's survival time as it directly models the relationship between the p variables and the survival time T. For instance, we can focus on prognostic biomarkers associated with the risk of relapse in early breast cancer patients or the effects of treatment.

For a given patient i, represented by a triplet $(X_i, \tilde{T}_i, D_i)$, the hazard function $h(t|X_i)$ examines how variables influence the rate of a particular event happening at a specific time t. It is written as:

$$h(t|X_i) = h_0(t) exp(\beta^T X_i) \text{ for } i = 1, \dots, n, \tag{2.10}$$

where the baseline hazard function, $h_0(t)$, can be an arbitrary non-negative function of time. $\beta^T =$

$(\beta_1, \ldots, \beta_p)$ is the coefficient vector associated with variables. The CoxPH model is semi-parametric since it is parameterized by a vector of real parameters, while $h_0(t)$ remains unspecified.

The CoxPH model relies on two assumptions. First, it supposes a linear relationship between the log hazard and the variables X_i, with the baseline hazard being an intercept term that varies with time. Indeed, if we take the logarithm of the risk function, we obtain a linear function of X:

$$\ln\left(\frac{h(t|X_i)}{h_0(t)}\right) = \beta^T X_i.$$

Additionally, the ratio of the instantaneous risk for any two patients is independent of time t: this corresponds to the proportional hazard assumption. For two patients i and j that differ in their variables denoted by X_i and X_j, the ratio is given by :

$$\frac{h(t|X_i)}{h(t|X_j)} = \frac{h_0(t)exp(\beta^T X_i)}{h_0(t)exp(\beta^T X_j)} = exp\left[\beta^T(X_i - X_j)\right].$$

This hypothesis implies that the hazard curves for two patients should be proportional and cannot cross.

The challenge is to estimate the vector β. By estimating the coefficients β, we can identify specific prognostic biomarkers. The problem is that we cannot use the likelihood as $h_0(.)$ remains unspecified.

Cox suggested to estimate β by maximizing $\mathcal{L}(\beta)$. This function does not depend either on h_0 or on the actual death times (only their order). It is possible to find a unique solution β which maximizes the function $\mathcal{L}(\beta)$ if and only if the number of parameters to be estimated p is less than the number of events among the observations n. Otherwise, the model is not identifiable, i.e., it is impossible to find a unique solution for β.

The coefficient vector $\hat{\beta}$ is estimated by maximizing the partial likelihood, or equivalently, minimizing the negative log-partial likelihood $\mathcal{NLL}(\beta)$ for improving efficiency. It is written as:

$$\mathcal{NLL}(\beta) = \sum_j D_j[\beta^T X_j - \log[\sum_{j \in R_j} exp(\beta^T X_j)]]. \tag{2.11}$$

The maximum partial likelihood estimator can be obtained with the numerical Newton-Raphson method to iteratively find an estimator $\hat{\beta}$ which minimizes $\mathcal{NLL}(\beta)$.

The survival function can be computed as follows:

$$S(t|X_i) = exp(-H_0(t)exp(\beta^T X_i)) = S_0(t)^{exp(\beta^T X_i)},$$

where $H_0(t)$ is the cumulative baseline hazard function, and $S_0(t) = exp(-H_0(t))$ represents the baseline survival function.

If survival times follow a particular distribution, we can implement a parametric method, as it is easy to interpret, and as we only need to estimate a few parameters that completely characterize the distribution, unlike with the CoxPH model.

2.5 . The Accelerated Failure Time model, an alternative to CoxPH

The AFT model represents an alternative to the CoxPH model. It provides a direct interpretation of the relationship between patient variables and failure times, as it assumes that the relationship of the logarithm of survival times T and the variables is linear. Since T is positive, $\log T$ is defined, and the AFT model can be written in the following form:

$$\log T = \beta^T X + \sigma\epsilon. \tag{2.12}$$

β corresponds to the vector of regression coefficients, $\sigma(\sigma > 0)$ denotes an unknown scale parameter, and ϵ is an error term. If the error term ϵ follows a specific distribution, the model $\log T$ follows the same known probability distribution. The exponential, Weibull, log-logistic, Gompertz, and log-normal probability distributions are the most commonly used distributions. The AFT model can verify the proportional hazards and multiplicative effects hypotheses with the exponential or Weibull distributions. By choosing a log-logistic AFT model, the model does not verify the proportional hazards hypothesis. If the Gompertz distribution is used, then the proportional hazards hypothesis is verified, but the hypothesis of the multiplicative effect is not.

Let $S_0(t)$ be the basis survival function for $X = 0$, corresponding to the tail of the distribution of $\exp(\epsilon)$. The survival function in an AFT model is written as:

$$S(t|X_i) = S_0(t \exp(\beta^T X_i)). \tag{2.13}$$

It then follows the expression of the risk function:

$$h(t|X_i) = \exp(\beta^T X_i) h_0(t \exp(\beta^T X_i)). \tag{2.14}$$

As shown with the equation 2.14, variables have a multiplicative effect on t rather than on the risk function, as is the case for the CoxPH model. If $\exp(\beta^T X_i)$ is greater than 1, variables have an accelerating effect on the event's occurrence, i.e., they decrease the survival time of the individual i. If $\exp(\beta^T X_i)$ is lower than 1, variables have a decelerating effect on the risk, increasing the survival time of the individual i.

The parameters of an AFT model can be estimated using the maximum likelihood when the distribution of the basis risk function h_0 is known, using a parametric function for h_0. If no parametric hypothesis is made on h_0, then the AFT model is a semi-parametric model. However, there is no equivalent of the CoxPH partial likelihood for the AFT model, allowing the elimination of the parameter h_0. The estimation of the parameters β and h_0 is thus complex.

2.6 . Counting process approach

Other non-parametric estimators were also developed and are used to compute the cumulative hazard, known as the Nelson-Aalen estimator or Breslow estimator.

2.6.1 . The Nelson-Aalen estimator

In order to predict how long an individual will survive a given disease, we need to estimate the full risk function. For example, determining the individual survival function of a breast cancer patient could help us identify when to start the screening for this patient.

For $t_j \leq t \leq t_{j+1}$, the instantaneous risk h is estimated as:

$$\hat{h}(t_j) = \frac{m_j}{r_j}. \tag{2.15}$$

Then, as $H(t) = \int_0^t h(u)du$, we have:

$$\hat{H}(t) = \sum_{\{j:t_j \leq t\}} \frac{m_j}{r_j}. \tag{2.16}$$

$\hat{H}(t)$ corresponds to the Nelson-Aalen estimator. First introduced by Aalen (1978), the Nelson-Aalen generalizes the empirical cumulative intensity proposed by Nelson (1969) to the case of censored survival times.

2.6.2 . The Breslow estimator

The Breslow estimator (Breslow, 1972) of the cumulative hazard is an extension of the Nelson-Aalen estimator. It is obtained using the KM estimator and based on the relation: $H(t) = -\log[S(t)]$. It is written as:

$$\hat{H}(t) = -\log\left[\hat{S}(t)\right] = -\sum_{j:t_j \leq t} \log\left(1 - \frac{m_j}{r_j}\right). \tag{2.17}$$

An estimator of the basis cumulative risk function can be obtained as:

$$\hat{H}_0(t) = \sum_{j:t_j \geq t} \frac{m_j}{\sum\limits_{i \in \mathcal{R}_j} \exp(\hat{\beta}^T X_i)}. \tag{2.18}$$

with $\mathcal{R}_j = \{i | t_i \geq t_j\}$ the risk set, i.e., the index set of subjects at risk at time t_j. The KM estimator describes survival according to a single risk factor but does not investigate the impact of multiple factors, while the CoxPH model does. The CoxPH model is a semi-parametric method based on multiple assumptions.

2.7 . Discrete time framework

We can extend the previous functions to a discrete time scale. Within this framework, the continuous survival time is divided into L time intervals $A_l =]t_{l-1}, t_l], l = 1, \ldots, L$, with midpoint α_l. The probability mass function (PMF) for the individual i with baseline variables X_i is defined as:

$$f_{il} = P(T_i \in A_l | X_i) = S(t_{l-1}|X_i) - S(t_l|X_i). \tag{2.19}$$

It follows that the survival function is written as:

$$S_i(t|X_i) = P(T_i > t|X_i) = \sum_{k>l} f_{il}. \tag{2.20}$$

With discrete times, the hazard in a specific interval corresponds to the probability of an individual experiencing the event during that interval given that the individual survives up to the start of that

interval. Thus, we can define the hazard rate as:

$$\begin{aligned} h_{il}(X_i) &= P(T_i \in A_l | T_i > t_{l-1}, X_i) \\ &= P(t_{l-1} < T_i \leq t_l | T > t_{l-1}, X_i) \\ &= \frac{f_{il}}{S(t_{l-1}|X_i)} \\ &= \frac{S(t_{l-1}|X_i) - S(t_l|X_i)}{S(t_{l-1}|X_i)}. \end{aligned} \tag{2.21}$$

The PMF can be rewritten as:

$$f_{il} = h_{il}(X_i) S(t_{l-1}|X_i), \tag{2.22}$$

and the survival function can be expressed as:

$$\begin{aligned} S_i(t|X_i) &= [1 - h_{il}(X_i)] S(t_{l-1}|X_i) \\ &= \prod_{k=1}^{l} [1 - h_{ik}(X_i)]. \end{aligned} \tag{2.23}$$

The PMF and survival function associated with the censoring time C are defined as:

$$f_{il} = P(C_i \in A_l | X_i), \tag{2.24}$$

$$S_i = P(C_i > t | X_i). \tag{2.25}$$

Assuming that T and C are independent, we can derive the likelihood function for right-censored survival data in a discrete-time framework. Cox considered this an ad hoc modification of his continuous-time model and, therefore, proposed an estimation procedure analogous to the partial likelihood. He introduced a discrete-time hazard parametrization by considering the sigmoid of the linear predictor, noted as $\phi_j(X_i) = \alpha_j + \beta^T X_i$. It follows that:

$$h_{il}(X_i) = \frac{1}{1 + \exp(-\phi_j(X_i))}. \tag{2.26}$$

Brown (1975) later showed that we can estimate the parameters of $h_{il}(X_i)$ in the same manner as a regular logistic regression. To construct the likelihood, the individual i contributes to the product of the conditional survival probabilities for the time intervals in which he is observed but does not experience the event. If the event is observed for individual i in the interval $A_{l_i} =]t_{l_i-1}, t_l]$, the individual additionally contributes to the product of conditional survival probabilities. By defining the event label $d_{il} = \mathbb{I}(T_i \in A_{l_i}) = \mathbb{I}(t_{l-1} < T_i \leq t_l)$, which is given by $(d_{i1}, \dots d_{il_i}) = (0, \dots, 0)$ for censored individuals and $(d_{i1}, \dots d_{il_i}) = (0, \dots, 1)$ for individuals with an event, the likelihood is given by:

$$\mathcal{L} = \prod_i \prod_{l=1}^{l_i} h_{il}(X_i)^{d_{il}} \left[1 - h_{il}(X_i)\right]^{1-d_{il}}, \tag{2.27}$$

which we recognize as the Bernoulli likelihood. It can be referred to as a discrete logistic hazard and

partial logistic regression. With this formulation, the negative log-likelihood can be rewritten as:

$$\mathcal{NLL} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{l=1}^{l_i}(d_{il}\log[h_{il}(X_i)] + (1-d_{il})\log[1-h_{il}(X_i)]). \tag{2.28}$$

This is the negative log-likelihood for Bernoulli data or binary cross-entropy.

2.8 . Evaluation Criteria

The models were compared using two metrics: the Concordance index (C-statistic) for discrimination, and the Integrated Brier Score (IBS) that measures both the discrimination and calibration of the model.

2.8.1 . The Brier Score and the Integrated Brier Score

The Brier Score (BS) is used to evaluate the accuracy of the predicted survival function at a given time t. It is based on the root mean square error and focuses on the difference between the observed survival status and the predicted probability of survival. It lies between 0 (best possible value) and 1. The BS for uncensored data is written as :

$$\begin{aligned}\hat{BS}(t) =& \frac{1}{n}\sum_{i=1}^{n}\left[\mathbb{I}\{T_i > t\} - \hat{S}(t|X_i)\right]^2 \\ =& \frac{1}{n}\sum_{i=1}^{n}\left[\hat{S}(t|X_i)^2\mathbb{I}\{T_i \leq t\} + (1-\hat{S}(t|X_i))^2\mathbb{I}\{T_i > t\}\right]\end{aligned} \tag{2.29}$$

With censored data, only a subset of the event times is observed. Graf et al. (1999) introduced a weighting of the BS based on the inverse probability of censoring (IPCW). It can be rewritten as:

$$\begin{aligned}\hat{BS}(t)_{IPCW} =& \frac{1}{n}\sum_{i=1}^{n}\left[\hat{S}(t|X_i)^2\frac{\mathbb{I}\{\tilde{T}_i \leq t, D_i = 1\}}{\hat{G}(\tilde{T}_i)}\right. \\ &\left. + (1-\hat{S}(t|X_i))^2\frac{\mathbb{I}\{\tilde{T}_i > t\}}{\hat{G}(\tilde{T}_i)}\right]\end{aligned} \tag{2.30}$$

The IBS is then written as:

$$I\hat{B}S(t)_{IPCW} = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2} BS(s)ds \tag{2.31}$$

2.8.2 . The Concordance Index

The C-statistic measures the discrimination ability of a model, that is, its ability to distinguish high-risk and low-risk patients. Specifically, it estimates the probability of agreement, i.e., the probability that two patients randomly selected are ordered similarly in terms of survival prediction and in their observed survival data. We use here a concordance measure that accounts for the censored data using the inverse probability of censoring weighting (Uno et al., 2011). The concordance index for time t is then:

$$\hat{C}(t) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} D_i \hat{G}(\tilde{T}_i)^{-2} \mathbb{I}\{\tilde{T}_i < \tilde{T}_j, \tilde{T}_i < t\} \mathbb{I}\{\hat{S}(t|X_i) < \hat{S}(t|X_j)\}}{\sum_{i=1}^{n}\sum_{j=1}^{n} D_i \hat{G}(\tilde{T}_i)^{-2} \mathbb{I}\{\tilde{T}_i < \tilde{T}_j, \tilde{T}_i < t\}} \tag{2.32}$$

The value of the C-statistic lies between 0.5 and 1, with 0.5 equivalent to a random prediction and 1 corresponding to a perfect ability to rank.

3 - Survival Analysis for High-Dimensional Data

3.1 . Context

In recent years, machine learning models have been increasingly used in various domains, particularly in medical research. Healthcare applications of machine learning models are diverse, from estimating the failure time distribution to the prognostic evaluation of different variables.

Machine learning models are often adapted to learn complex and nonlinear dependencies between numerous input variables and an output to predict. However, the specificities of survival data make the application of machine learning models challenging in survival analysis. Machine learning algorithms are adequate when there are many instances (e.g. n patients) in a reasonable dimensional feature space (e.g. p biomarkers). In the context of survival analysis for precision oncology, with high throughput sequencing, large amounts of input features data are collected for each individual, and the number of input variables often exceeds the number of individuals ($p \gg n$), which raises an overfitting problem. Another challenge for machine learning models application concerns censoring. Censoring differs from missing data: it carries partial information and should be integrated into the prediction algorithm to obtain the most optimal result.

While neural networks are well-suited for complex datasets, they unfortunately cannot be directly applied to survival analysis, as they were not originally designed to handle censored data. FNNs are a specific type of neural networks developed by Rosenblatt (1958) that were not applied to survival analysis until the work of Faraggi et al. (1995). Since then, multiple applications of neural network models have been subsequently developed to handle censored data. Some models directly change the output and overcome censoring using pseudo-observations, while other methods are based on specific loss adapted to censored data, such as CoxCC, CoxTime (Kvamme et al., 2019), and DeepHit (Lee et al., 2018).

In the following section, FNNs currently adapted to survival analysis are presented. We also introduce 2 statistical learning methods, RSF and penalized CoxPH, adapted to survival analysis in high dimensional settings and used as benchmark models.

3.2 . Feedforward neural networks

Let $X = (X_1, \ldots, X_p)$ be the vector of input variables. A FNN operates a transformation of the input variables:

$$Y = \phi(X, W, b), \tag{3.1}$$

where ϕ is the transformation, W and b are the weights and biases of the neural network to be estimated, and Y is the explained variable.

3.2.1 . Architecture of a FNN

A FNN consists of layers: the input layer, one or several hidden layers, and the output layer. Each layer is composed of several units, also called nodes or neurons. The input layer includes all the P explanatory variables, such as the clinical characteristics or gene expression data. Each hidden layer comprises a certain number of neurons we need to determine. The output layer Y is continuous. Apart from the output nodes, every neuron in a layer is connected with all the units in

the next layer: the network is fully connected, and there is no feedback between layers.

The FNN's architecture is defined by the number of its layers, the number of nodes per layer, and the activation function. These components are called hyperparameters of the models. Figure 3.1 shows an example of a FNN with $P = 3$ input variables, $K = 4$ neurons per hidden layer, and $H = 1$ hidden layer.

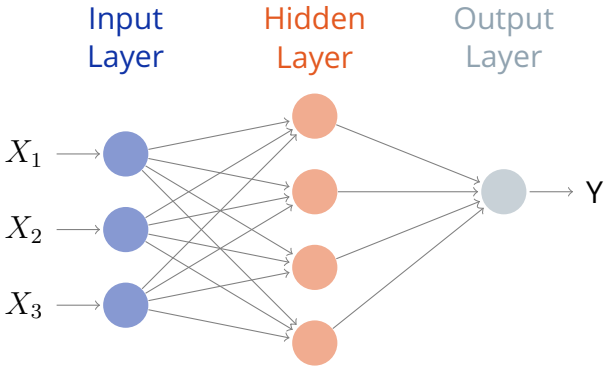


Figure 3.1: Example of a FNN's architecture

The output of a neural network with P input variables, 1 hidden layer with K neurons, is written:

$$\hat{Y}_i = \hat{\phi}(X_i; W, b) = f_2\left(b + \sum_{k=1}^{K} W_k f_1\left(b_k + \sum_{p=1}^{P} W_{pk} X_{ip}\right)\right), \tag{3.2}$$

where f_1 and f_2 are the activation functions of the input layer and the hidden layer, respectively, W_{pk} the weights from the input to the hidden layer, W_k the weights from the hidden to the output layer and b the biases or offsets. As we see in Equation 3.2, the neurons of the layers are combined using a weighted sum.

The activation function f is a transformation applied to the combined neurons of the hidden layer before being transmitted to the next layer. It needs to be differentiable and non-linear. A classical choice for the activation function is the sigmoid function. It is defined by:

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}. \tag{3.3}$$

The function's input is transformed into a value between 0.0 and 1.0. To avoid the addition of biases, the centralized form of the sigmoid, the hyperbolic tangent or $\tanh$, can also be applied:

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}. \tag{3.4}$$

It outputs values between -1.0 and 1.0.

We can face two problems by applying a sigmoid or hyperbolic tangent. These functions can saturate: large values snap to 1.0, and small values snap to -1 or 0 for tanh and sigmoid, respectively. Further, the functions are only really sensitive to changes around the mid-point of their input, such as 0.5 for sigmoid and 0.0 for tanh. With deep neural networks composed of numerous hidden layers, these functions also raise the vanishing gradient problem: the amount of error back propagated and used to update weights decreases dramatically with each additional layer through which it is propagated.

The Rectified Linear Unit (ReLU) activation function is a piecewise linear function that overcomes

the limits of tanh and sigmoid. It returns the value provided as input directly or the value 0.0 if the input is 0.0 or less. It can be expressed as follows:

$$f(x) = max(0, x). \tag{3.5}$$

In this case, no transformation is applied to the input neurons; that is, f is the identity function.

3.2.2 . Learning the parameters with the optimization algorithm

The learning process relies on the backpropagation technique (Rumelhart et al., 1986). It is used to learn the weights and biases, also called the model's parameters (noted as θ). During training, input vectors are forward propagated through a randomly initialized network, the FNN produces output $\hat{Y}$. A specific loss function $L(\theta)$ is computed as a function of the difference between these outputs and the desired ones. Then, gradient descent is used to optimize the objective function $L(\theta)$. More precisely, the error is used to walk backwards through the network, using the gradients of the objective function $\nabla_\theta L(\theta)$ with respect to the parameters, and thus adjust the weights and biases in such a way that attempts to reduce the error. The gradient is adjusted by a certain amount controlled by a learning rate η, which can be fixed, adaptive, or determined by a schedule. After each backpropagation, we iterate with the dataset and continue adjusting θ until there is little to no improvement in the error, reaching a local or global minimum. At iteration k, the ordinary gradient descent can be written as follows:

$$\theta^{k+1} = \theta^k - \eta_k \nabla_{\theta^k} L(\theta^k). \tag{3.6}$$

Less expensive gradient descent methods, such as stochastic gradient descent (SGD) have been implemented. It relies on the fact that L is an average of identical and independent examples. For each iteration k, the gradient is computed on a single observation randomly picked within the dataset, not on the whole dataset. Thus, updates can be made more frequently or even on each example. For each individual i of the training set with covariates X_i and survival time Y_i, we have :

$$\theta^{k+1} = \theta^k - \eta_k \nabla_{\theta^k} L(\theta^k; X_i; Y_i). \tag{3.7}$$

Ordinary gradient descent performs redundant computations for similar individuals, whereas SGD performs one update at a time, requiring less memory and time. These frequent updates SGD enables to jump to new and potentially better local minima. However, these frequent updates with a high variance give more unstable convergence.

The limit with SGD is that the same learning rate is applied to all parameter updates. If our data is sparse and our variables have very different frequencies, we might not want to update all of them to the same extent but perform a larger update for rarely occurring variables. Another critical challenge of minimizing highly non-convex error functions common for neural networks is avoiding getting trapped in their numerous sub-optimal local minima.

Root Mean Square Propagation (RMSProp) (Tieleman et al., 2012) is an adaptive learning rate method that is based on the running average $E[(g^k)^2]$. g^k denotes the gradient at iteration k: $g^k = \nabla_\theta L(\theta^k)$. Indeed, instead of storing all previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients. The running average depends

on the previous average and the current gradient as follows:

$$E[(g^k)^2] = 0.9E[(g^{k-1})^2] + 0.1(g^k)^2. \tag{3.8}$$

RMSProp divides the learning rate by an exponentially decaying average of squared gradients:

$$\theta^{k+1} = \theta^k - \frac{\eta}{\sqrt{E[(g^k)^2] + \epsilon}} g^k. \tag{3.9}$$

Another more stable method is the Adaptative Moment Estimation (Adam) algorithm (Kingma et al., 2014). It computes a step for each parameter of the neural network. More precisely, it records the average of the gradients of previous epochs with m^k and the square of the average of the gradients of previous epochs with v^k.

$$\begin{aligned} m^k &= \beta_1 m^{k-1} + (1-\beta_1) g^k, \\ v^k &= \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2. \end{aligned}$$

m^k and v^k are the estimates of the first-order moments (the mean) and the second-order moments (the uncentred variance) of the gradients, respectively. As m^k and v^k are initialized to 0, they are biased for this value. Their estimates are then corrected for this bias:

$$\begin{aligned} \hat{m^k} &= \frac{m^k}{1-\beta_1}, \\ \hat{v^k} &= \frac{v^k}{1-\beta_2}. \end{aligned}$$

We obtain the following update rule:

$$\theta^{k+1} = \theta^k - \frac{\eta}{\sqrt{\hat{v^k}} + \epsilon} \hat{m^k}. \tag{3.10}$$

ϵ is a smoothing term that avoids division by zero. A limitation of the Adam algorithm is that sometimes it fails to converge to an optimal solution due to the use of the exponential moving average of the past squared gradients. To fix this convergence issue, AMSGrad employs the maximum of the past squared gradients, which prevents optimization from slowing down too quickly. It also removes the debiasing step. Thus, the update rule becomes:

$$\begin{aligned} m^k &= \beta_1 m^{k-1} + (1-\beta_1) g^k, \\ v^k &= \beta_2 v^{k-1} + (1-\beta_2)(g^k)^2, \\ \hat{v}^k &= max(\hat{v}^{k-1}, v^k), \\ \theta^{k+1} &= \theta^k - \frac{\eta}{\sqrt{\hat{v}^k} + \epsilon} m^k. \end{aligned} \tag{3.11}$$

3.3 . Applying FNNs to survival prediction

With the increasing application of machine learning models in various domains, numerous methods have been developed to adapt neural networks to time-to-event predictions. Multiple FNNs

have been introduced based on the CoxPH partial log-likelihood, the discrete-time framework, or the jackknife method. All these models rely on using a specific loss function to handle time-to-event data.

As early as 1995, FNNs were adapted to survival analysis in a continuous-time framework, with the linear predictor of the CoxPH being replaced by a one-hidden layer FNN (Faraggi et al., 1995). This model still relies on the proportionality assumption but can capture non-linearity in the data. However, it does not outperform CoxPH. With Cox-nnet (Ching et al., 2018) and DeepSurv (Katzman et al., 2016), the models are implemented in a deep learning framework and outperform CoxPH in terms of discrimination capacities. All these models are trained by minimizing the negative partial log-likelihood. Kvamme et al. (2019) suggested adding the time variable in the input variables so that the model overcame the proportional hazards assumption, and a specific loss function was introduced to handle both proportional and non-proportional hazards.

Neural networks have also been developed in discrete-time survival analysis, with the follow-up time being divided into fixed intervals. Here, the risk is directly estimated as the neural network's output, for each time interval. It is a more flexible framework as the model does not rely on the proportionality assumption. Biganzoli et al. (1998) introduced a neural network with a single output, where individuals are replicated for each time interval. They added the ridge penalty. Lee et al. (2018) introduced another model, DeepHit, that directly estimates the probability mass function and combines the log log-likelihood with a ranking loss.

A third solution consists in using pseudo-observations. Andersen et al. (2010) introduced pseudo-observations as an approach to perform regression based on the jackknife method in a survival context. With DNNSurv, Zhao et al. (2019) substituted the observed survival times by jackknife pseudo-observations and then used them as a response variable in a FNN. Two new methods for computing pseudo-observations are introduced here.

Two benchmark models are implemented: LASSO adds a L_1-penalty to the log-likelihood of CoxPH; RSF computes a random forest using the log-rank test as the splitting criterion. Both methods are presented in Chapter 2.

3.3.1 . Continuous time framework

Kvamme et al. (2019) introduced a method called **CoxCC** that uses a special loss based on a case-control approximation. They proposed to randomly sample a new set of controls at each iteration instead of keeping control samples fixed. The loss is written as:

$$\mathcal{L}_{\text{CoxCC}} = \frac{1}{n} \sum_{i:D_i=1} \log\Big(\sum_{j \in \tilde{\mathcal{R}}_i} \exp[\phi(X_j) - \phi(X_i)]\Big), \tag{3.12}$$

with $\tilde{\mathcal{R}}_i$ a subset of the risk set $\mathcal{R}_i$ at time t including individual i. i represents the case, and the j's are the controls sampled from the risk set. The authors can fit a neural network with their specific loss (Equation 3.12) using a mini-batch gradient descent algorithm.

They also developed a second version of their model, **CoxTime**, that is not constrained by the proportionality assumption. To do so, they added the time variable as an additional input to the model. The loss function can then be rewritten as :

$$\mathcal{L}_{\text{Cox-Time}} = \frac{1}{n} \sum_{i:D_i=1} \log\Big(\sum_{j \in \tilde{\mathcal{R}}^i} \exp[\phi(t_i, X_j) - \phi(t_i, X_i)]\Big). \tag{3.13}$$

3.3.2 . Discrete time framework

Commonly used survival models can also be extended using neural networks in a discrete-time framework, which enables overcoming the proportional hazards assumption.

Biganzoli et al. (1998) presented the Partial Logistic Artificial Neural Network (**PLANN**) model. With this model, the time is divided into L time intervals $A_l =]t_{l-1}, t_l]$, $l = 1, \ldots, L$, with midpoint a_l. The input of the model is composed of the variables and the time variable a_l, while the output corresponds to the discrete instantaneous hazard, written as:

$$\begin{aligned} \hat{h}_{il} &= \hat{h}_l(X_i, a_l) \\ &= P(T_i \in A_l | T_i > t_{l-1} | X_i), \end{aligned} \tag{3.14}$$

with T_i the survival time for individual i at time a_l. With the inclusion of the time variable as input of the model, the p variables of each i are repeated for each time interval. More precisely, each patient of the training set is repeated for the number of intervals being observed, whereas, on the test set, each subject is repeated for all time intervals. As the index of the time interval is used as an explanatory variable, smooth estimates of the hazard rate can be obtained, and interactions between time and variables are considered. The FNN's architecture used by Biganzoli et al. (1998) is composed of 3 layers and a logistic activation function. The output of the neural network with K neurons in the hidden layer and P input variables is given by:

$$\hat{h}_{il} = \hat{h}(X_i, t_l) = f\left(b + \sum_{k=1}^{K} W_k f\left(a_k + \sum_{p=1}^{P} W_{pk} X_{ip}\right)\right),$$

with W_{pk},a_k and W_k, b the weights and biases of the input layer and the hidden layer, respectively, f is the logistic activation function. d_{il} is the event indicator: if the patient died in the interval A_l, then $d_{il} = 1$; if the patient is censored, then $d_{il} = 0$. l_i corresponds to the number of intervals for which the individual i is observed. Thus we have $l_i \leq L$ and $d_{i0}, \ldots, d_{i(l_i - 1)} = 0$. The cost function corresponds to the binary cross-entropy :

$$\mathcal{L}_{\text{PLANN}} = -\sum_{i=1}^{n} \sum_{l=1}^{l_i} \{d_{il} \log \hat{h}_{il}) + (1 - d_{il})[1 - \log \hat{h}_{il}]\}, \tag{3.15}$$

where $\hat{h}_{il}(x_i)$ is estimated as output value of the FNN. Biganzoli et al. (1998) add a ridge penalisation term to the loss function (3.15):

$$\mathcal{L}_{\text{PLANN}+Ridge} = \mathcal{L}_{\text{PLANN}} + \lambda \left\| W \right\|_2, \tag{3.16}$$

with λ being tuned by CV.

The authors recently published an update of their model (Kantidakis et al., 2021), in which they proposed treating the L non-overlapping intervals as L separate variables, resulting in $1 + L + p$ input nodes instead of $1 + 1 + p$ nodes. With PLANN, the survival distribution is predicted using the discrete hazard function.

An alternative model, **DeepHit** (Lee et al., 2018), directly estimates the discrete survival function. The time is divided into L time intervals. $y(x) = [y_1(x), \ldots, y_L(x)]^T$ is the output of the neural network: given a patient i with variables X_i, $y_s(X_i)$ is the probability that the patient will

experience the event at time s. Thus we have:

$$\hat{S}_i(t|X_i) = 1 - \sum_{l=1}^{l_i} y_l(X). \tag{3.17}$$

A loss function is optimized that combines the negative log-likelihood for right censored data with a ranking loss. The first term of the loss function is then written :

$$\mathcal{L}_{\text{DeepHit 1}} = -\frac{1}{n}\sum_{i=1}^{N} \underbrace{(D_i \log(y_{l_i})}_{\text{Patients with event}} + \underbrace{(1 - D_i) \log(\hat{S}(\tilde{T}|X_i))}_{\text{Censored patients}}. \tag{3.18}$$

κ_i is the index of the event time for individual i. $\mathcal{L}_{\text{DeepHit 1}}$ pushes the model to learn the underlying survival distribution.

To the term in Equation 3.18 is added another loss function which includes a constraint on the discrimination capacity of the model. It is an extension of the concordance index and acts as a ranking loss:

$$\mathcal{L}_{\text{DeepHit 2}} = \sum_{i,j} D_i \mathbb{I}\{\tilde{T}_i < \tilde{T}_j\} \exp(\frac{\hat{S}(\tilde{T}_i|X_i) - \hat{S}(\tilde{T}_i|X_j)}{\sigma}). \tag{3.19}$$

$\mathcal{L}_{\text{DeepHit 2}}$ (Equation 3.19) aims at improving the discrimination of the model and tends to force the model to focus on time intervals with a high frequency of observed events, where discrimination is more challenging. The two terms are combined using a convex combination:

$$\mathcal{L}_{\text{DeepHit}} = \alpha \mathcal{L}_{\text{DeepHit 1}} + (1 - \alpha)\mathcal{L}_{\text{DeepHit 2}}, \tag{3.20}$$

where α is a hyperparameter. The introduction of the term $\mathcal{L}_{\text{DeepHit 2}}$ can be criticised, as adding $\mathcal{L}_{\text{DeepHit 2}}$ constrains the model and helps it improve its discrimination capacities and might arm its calibration capacities. Hence, as the ultimate goal of the model is not only to solve a ranking problem like stratifying a patient risk in order to inform clinical decisions, but rather to learn survival distributions, it makes the introduction of the $\mathcal{L}_{\text{DeepHit 2}}$ term questionable.

When using continuous time survival data, these two methods could represent a drawback as the discretization of the data implies a loss of information. Furthermore, choosing the granularity of the discretization and even the discretization scheme involves a trade-off between the smoothness of the resulting hazard function estimate and considerations of computational cost. The hazard function output by the model may require smoothing and have interpretability issues.

3.3.3 . Pseudo-observations

In survival analysis, pseudo-observations provide a way to circumvent the complexity of censoring. In the work of Andersen et al. (2010), pseudo-observations computation relies on the idea of leave-one-out estimation. They approximate the missing values using the KM estimate and are computed at a finite number of time points spread on the event time scale. Andersen et al. (2010) use the pseudo-observations as the output of a generalized estimating equation model.

In survival analysis, the pseudo-observation for the individual i corresponds to this individual's contribution to the KM estimate $\hat{S}(t)$. Pseudo-observations are computed for all individuals at a given time, regardless of event time or censorship status. For each i, we have k pseudo-observations.

Let us define the pseudo-observation for individual i at time t :

$$\hat{Z}_i(t) = N\hat{S}(t) - (N-1)\hat{S}^{-i}(t). \tag{3.21}$$

Here, $\hat{S}^{-i}(t)$ denotes the leave-one-out KM estimate $\hat{S}(t)$ obtained by discarding the data for individual i. We thus have an approximation of the empirical mean of the sample deprived of individual i.

In their neural network model, DNNSurv, Zhao et al. (2019) adapted the pseudo-observations approach in a discrete-time framework. Every survival time, censored or not, is replaced by the pseudo-observation computed as:

$$\hat{Z}_{il}(t_{l+1}|\mathcal{R}_l) = \mathcal{R}_l\hat{S}(t_{l+1}|\mathcal{R}_l) - (\mathcal{R}_l - 1)\hat{S}^{-i}(t_{l+1}|\mathcal{R}_j), \tag{3.22}$$

where $\hat{S}(t_{l+1}|\mathcal{R}_j)$ is the KM estimator constructed using the remaining survival times for all patients still at risk at time t_l, and $\hat{S}^{-i}(t_{l+1}|\mathcal{R}_l)$ the KM estimator for all patients at risk but the i^{th} subject. The time variable is included in the input variables of the model. The loss function that is minimized boils down to the mean-squared error.

3.4 . Benchmark methods

With the development of data collection and detection techniques, survival analysis is often performed in a high-dimensional context, for instance, when using molecular data. The number of variables in the data exceeds the number of instances, and it becomes challenging to build the prediction model with all variables: conventional survival analysis methods might provide inaccurate results. Different solutions have been developed to overcome this issue.

We present two methods that we used as a benchmark for neural network models. One method includes a penalty function to identify the most relevant variables using sparse learning methods: this is a penalized CoxPH with LASSO (Tibshirani, 1997). This penalized CoxPH model seems more competitive than a simple CoxPH model in the context of complex and high-dimensional data. Another idea is to apply RSF (Ishwaran et al., 2008) that is made of an ensemble of survival trees. Many papers benchmarked their methods against RSF, as it is a flexible continuous-time method that is not constrained by the proportionality assumption.

3.4.1 . The CoxPH model with the LASSO penalty

In order to develop the prediction models and to determine the most relevant variables among all the variables, a penalization term can be added to the CoxPH partial-likelihood. This penalization is based on the regression coefficients and generally depends on a single parameter, either positive or null, denoted λ. One of the main objectives of penalized regressions is to force the regression coefficients to tend toward the null value. It allows the estimation of regression coefficients that are less variable but slightly biased (a compromise between variance and bias). The most used one is the LASSO penalty because it allows convex optimization and is interpretable in terms of variable selection.

First introduced in the context of linear regression by Tibshirani (1996) and then adapted to survival analysis (Tibshirani, 1997), the LASSO penalty corresponds to the L_1 norm of the regression

coefficients. The penalization term is denoted by:

$$pen(\lambda, \beta) = \lambda ||\beta||_1 = \lambda \left(\sum_{j=1}^{p} |\beta_j| \right).$$

The model's degree of parsimony or complexity depends on the penalization parameter λ: it varies from 0 (complete model including all biomarkers) to $+\infty$ (null model with no biomarkers). Thus this penalty can be used to select prognostic and/or predictive biomarkers. Estimating this parameter is essential to identify the sub-groups of biomarkers retained in the model correctly. λ is often estimated using cross-validation.

The estimator of the parameter β is obtained by solving the following problem:

$$\widehat{\beta}_{L_1} = \arg\min_{\beta} \{ -\mathcal{L}(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \}. \tag{3.23}$$

The optimization problem is convex in β and thus allows the use of convex optimization algorithms to estimate β. Therefore the optimization problem is equivalent to a minimization problem of the CoxPH partial log-likelihood (Cox, 1972) by adding the constraint: $\sum_{j=1}^{p} |\beta_j| \leq s$ with $s \in \mathbb{R}^+$. The resulting estimator $\widehat{\beta}$ is sparse, which means that a certain number of coefficients $\widehat{\beta}$ are null.

It has been shown that LASSO is sub-optimal, notably because it does not respect the oracle property or because of its shortcomings in the presence of strong correlations, since it tends to select only one biomarker among all the correlated biomarkers arbitrarily. It is also not consistent in terms of the selected variables. Therefore, other penalties are derived from the latter, such as the Ridge (Tibshirani, 1996), Elastic-Net (Zou et al., 2005), and Adaptive-LASSO (Fan et al., 2008).

3.4.2 . Random Survival Forests

A random survival forest model is a tree-based non-linear ensemble method first developed by Breiman (2001) and adapted to right-censored survival data by Ishwaran et al. (2008). It is constructed from an ensemble of decision trees.

The objective of decision trees is to create subgroups of individuals that are homogeneous in terms of the predicted variable. The construction of a decision tree is based on the determination of a sequence of nodes. Each interior node corresponds to a value taken by a variable, and each tree branch ends with a leaf (or terminal node) representing the values of the predicted variable. At each interior node, individuals are separated into subgroups according to the values of one of the variables.

Random forests are composed of a set of decision trees. Two sources of randomness are introduced to distinguish trees from one another: a random subset of the data is sampled before growing the tree (this is bootstrapping); at each node, the tree is grown using a splitting criterion based on subsets of randomly selected variables. The tree is built on the in-bag individuals, i.e., the fraction of n observations obtained by bootstrapping, and predictions are made on each individual of the remaining fraction or out-of-bag (OOB).

RSF (Ishwaran et al., 2008) is a specific type of random forest adapted to time-to-event data. The model does not rely on the proportional hazards hypothesis, and it allows the capture of nonlinear effects and high-order interactions in the data. As for classical random forests, M subsets of the data are randomly sampled. For each bootstrapped sample set, a survival tree is grown by recursively splitting the tree into nodes. At each tree node, a subset of the remaining variables is sampled. For

each node, a proposed split for a given variable X_p is of the form: $X_p \leq c$ and $X_p > c$, with c the associated splitting criterion. The variable with the splitting criterion that maximizes survival difference between individuals of the resulting nodes is chosen among all candidate variables. Here, the applied criterion is the log-rank splitting rule.

The tree is fully grown when each terminal node is greater than or equal to a specific number of unique deaths. For each terminal node of each tree, denoted as ν, the cumulative hazard function (CHF) is estimated using the non-parametric Nelson-Aalen estimator, $\hat{H}_\nu(t)$. Let $\hat{H}(t|X_i)$ be the conditional CHF of the terminal node that individual i belongs to based on its predictors X_i, and estimated as:

$$\hat{H}(t|X_i) = \hat{H}_\nu(t) \text{ if } X_i \in \nu. \tag{3.24}$$

All patients within ν have the same CHF. In Equation 3.24, the estimate is obtained using a single tree. Thus, the ensemble estimate is obtained by averaging these estimates over all trees. Let $\hat{H}_m(t|X)$ denote the CHF for a tree grown from the m^{th} bootstrap sample. Let $I_{i,m} = 1$ if i is an OOB case for the resampled set m. The OOB ensemble CHF is obtained by averaging the CHF of each tree:

$$\hat{H}_e(t|X_i) = \frac{\sum_{m=1}^{M} I_{i,m}\hat{H}_m(t|X_i)}{\sum_{m=1}^{M} I_{i,m}}. \tag{3.25}$$

The estimator is obtained by averaging over only the bootstrap samples in which i is an OOB sample.

Finally, Ishwaran et al. (2008) compute the error rate based on the concordance statistic (Harrell et al., 1982). $t_1^*, \ldots, t_N^*$ denote all unique event times in the data. To rank two cases i and j, where i has a worse predicted outcome than j if :

$$\sum_{k=1}^{N} \hat{H}_e(t_k^*|X_i) > \sum_{k=1}^{N} \hat{H}_e(t_k^*|X_j). \tag{3.26}$$

This equation defines the C-statistic C based on the OOB ensemble CHF. Finally, the prediction error corresponds to $1 - C$.

4 - Survival Predictions using Feedforward Neural Networks

4.1 . Context

In this chapter, we study neural network models adapted to high-dimensional time-to-event data, using specific ways to handle censored observations. We compare survival models based on neural networks with different loss functions, either in a continuous-time framework with CoxCC and CoxTime, or in a discrete-time framework with DeepHit and PLANN. We also investigate different ways of defining pseudo-observations and include neural networks based on pseudo-observations in the comparison study. Indeed, as defined in Chapter 3, in survival analysis, pseudo-observations provide a way to circumvent the complexity of censoring. We use RSF and LASSO penalization as a benchmark. We investigate the effect of pre-training using autoencoders (AE) and variational autoencoders (VAE) on the model's prediction in the case of real patient cohorts.

Synthetic data are simulated, enabling us to study and compare the operating characteristics of the models. We simulate data from a CoxPH model and an AFT model using the cumulative inverse method (Bender et al., 2005). We also use an AFT model proposed by Friedman (2001), with interactions and nonlinear effects of random subsets of variables. Generating survival time-to-event data from different models enables us to compare the models on data from models with different characteristics and underlying assumptions. The CoxPH model is a proportional hazards model, which is not the case for the AFT model. Moreover, with the AFT model proposed by Friedman (2001), we can compare the models on highly nonlinear and complex data. We further apply the methods to 2 real patient cohorts: the METABRIC breast cancer data set, including $1,960$ patients, 6 clinical variables and the expression of 869 genes, and a set of data on lung cancer, consisting of $4,120$ patients, 3 clinical variables, and $1,000$ genes.

4.2 . New ways of computing pseudo-observations

We explore here different ways of computing pseudo-observations. As in the DNNSurv model, we can also consider pseudo-observations in a discrete-time framework, as we can see in Equation 3.22. The time is divided into L intervals, and pseudo-conditional survival probabilities are computed for each interval. We implement this method, called **Pseudo-discrete** (PDisc). The discrete-time variable is added as input of the neural network with the other variables X. The model's predictions correspond to the conditional survival probability in each interval. The marginal survival probability for the l^{th} time is obtained by multiplying these conditional survival probabilities up to the l^{th} interval.

An obvious problem with these methods is that pseudo-observations can take values strictly above one or below 0, which cannot be interpreted in a legitimate sense. With the objective to interpret the pseudo-observations as the probability that $Z_i(t) = \mathbb{I}(T_i(t) > 0) = 1$, we enforced that it belongs to the $[0, 1]$ interval, with the same idea used by classical pseudo-observations to take advantage of the information contained in the leave-one-out KM estimators. Let us denote by $\bar{Z}(t) = \big(\bar{Z}_1(t), \cdots, \bar{Z}_N(t)\big)$ the new vector of pseudo-observations. It is the solution of the constrained minimization problem:

$$\bar{Z}(t) = \arg \min_{u \in [0;1]^N} \|Au - b\|^2, \tag{4.1}$$

with A an $(N+1)$ by N matrix, and b an (N+1)-dimensional vectors given by:

$$A = \begin{pmatrix} \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ 0 & \frac{1}{N-1} & \cdots & \frac{1}{N-1} \\ \frac{1}{N-1} & 0 & \cdots & \frac{1}{N-1} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{1}{N-1} & \cdots & \frac{1}{N-1} & 0 \end{pmatrix}, \; b = \begin{pmatrix} \hat{S}(t) \\ \hat{S}^{-1}(t) \\ \vdots \\ \hat{S}^{-N}(t) \end{pmatrix}. \tag{4.2}$$

Similarly, as for the classical pseudo-observations, if there were no censoring, the solution to the problem would be $(Z_1(t), \cdots, Z_N(t))$. We call this new method **Pseudo-Optim** (POpt). Again, we build a neural network including biological variables and the time variable as input. The output is the marginal survival probability for a given time.

Based on the idea that the pseudo-observation is an estimation of $Z_i(t)$, we can adopt a Bayesian point of view and take $\tilde{T}_i(t)$ as the conditional probability $\mathbb{P}\left(Z_i(t)|\mathcal{D}\right)$, where $\mathcal{D}$ denotes the observed (uncomplete) data, $\mathcal{D} = \left(\tilde{T}_i, D_i\right)_{1 \leq i \leq N}$. If $D_i = 1$ or if $D_i = 0$ and $t < \zeta_i$ (the censoring time for individual i), then $Z_i(t)$ is observed and we simply have: $\tilde{Z}_i(t) = Z_i(t)$. If $D_i = 0$ and $t \geq \zeta_i$, then, since individual survivals are supposed to be independent and follow the same distribution law:

$$\mathbb{P}\left(Z_i(t) \,|\, \mathcal{D}\right) = \mathbb{P}\left(T_i > t \,|\, T_i > \zeta_i, \mathcal{D}\right) = \mathbb{P}\left(T > t | T > \zeta_i, \mathcal{D}\right) = \frac{\mathbb{P}\left(T > t \,|\, \mathcal{D}\right)}{\mathbb{P}\left(T > \zeta_i, \mathcal{D}\right)}. \tag{4.3}$$

Finally, $\mathbb{P}\left(T > t|\mathcal{D}\right)$ can be estimated by the KM estimate, and thus, if $D_i = 0$ and $t \geq \zeta_i$:

$$\tilde{Z}_i(t) = \frac{\hat{S}(t)}{\hat{S}(\zeta_i)}. \tag{4.4}$$

We call this the **Pseudo-KM** (PKM) method.

4.3 . Combining FNNs with encoders

We can use transfer learning to help the FNN model distinguish between the information and noise contained in the molecular data, thus reducing overfitting. Indeed, transfer learning is applied in machine learning to reuse a model previously developed for a task in the context of a second and similar task. Here, we first train an autoencoder (AE) or a variational autoencoder (VAE). Then a combined model is built based on both the encoder part of the AE or the VAE and the FNN-based survival model. It is initialized with the weights of the pre-trained AE or VAE model and fine-tuned as previously.

4.3.1 . Autoencoders

An AE is a neural network that learns to output a reconstruction of the input data and produces a non-linear embedding at the bottleneck. The AE is made of 2 submodels: the encoder and the decoder.

The encoder is the first part of the model: it reduces the dimension of the input data by mapping it into a latent representation space. During compression, the AE learns salient features present in the input data. The aim is to encode a representation that captures characteristics that generalize well to new data. The second part is the decoder: it maps the compressed representation back to the original space, attempting to reconstruct the input data from the embedding.

Let us define the input $X \in \mathbb{R}^p$. Both the encoder and the decoder are composed of several non-linear layers of the form: $f(u) = \sigma(Wu + b)$, where σ is the activation function. We assume that the encoder and the decoder are symmetric, such that they have the same number of layers, $L_E = L_D = L$. The encoder part is represented by $E(.)$, parameterized by θ_E. The bottleneck representation of X is then written :

$$z = E(X, \theta_E). \tag{4.5}$$

If $D(.)$ is the decoder function parameterized by θ_D, it follows that the reconstructed input can be written as:

$$\hat{X} = D(z, \theta_D). \tag{4.6}$$

We have: $\theta_E = \{(W_{E,l}, b_{E,l}), l \in [\![1, L]\!]\}$ and $\theta_D = \{(W_{D,l}, b_{D,l}), l \in [\![L-1, 0]\!]\}$. The parameters $\langle \theta_E, \theta_D \rangle$ are learned together to output a reconstructed data sample that is ideally the same as the original input $X \approx \hat{X}$.

The loss function minimized during backpropagation to learn the AE parameters corresponds to the mean squared error between the input data X and the reconstructed data $\hat{X}$:

$$\mathcal{L}_{AE}(X; \theta_E, \theta_D) = \frac{1}{N} \sum_{i=1}^{N} \|F(X_i; \theta_E, \theta_D) - X_i\|_2^2. \tag{4.7}$$

4.3.2 . Variational Autoencoders

VAE is an extension of AE that is based on Bayesian inference. It not only reduces the dimension of the input data X but also models its underlying probability distribution by learning a generalized latent prior. As for the AE, the VAE is based on two submodels. The probabilistic encoder is an inference model where, given a data point X, it produces a distribution over the latent values z. Then, a probabilistic decoder produces a distribution over the possible corresponding values of X given a specific value of z.

Let $p(X)$ be the probability distribution of the data, $p(z)$ the probability distribution of the underlying latent variable, and $p(X|z)$ the distribution of generating data given the latent variable. Using the law of probability, we have the relation:

$$p(X) = \int p(X|z)p(z)dz. \tag{4.8}$$

We want to infer $p(z)$ based on $p(z|X)$. Variational inference allows us to approximate $p(z|X)$ using a distribution q that is easier to evaluate, e.g., using a gaussian distribution:

$$q(z|X) = \mathcal{N}(z; \mu, \sigma I), \tag{4.9}$$

where the mean μ and standard deviation σ are outputs of the encoder. Thus we have:

$$\mu(X) = W_\mu f(W_h X + b_h) + b_\mu, \tag{4.10}$$

$$\nu(X) = W_\nu f(W_h X + b_h) + b_\nu, \tag{4.11}$$

$$\sigma(X)^2 = e^{\nu(X)}, \epsilon \sim \mathcal{N}(0, 1), \tag{4.12}$$

where μ, ν, and σ^2 are the mean, log variance, and the variance of Gaussian distribution, respectively. W and b are trainable parameters of the VAE model. Moreover, the latent variable is supposed to

be Gaussian: $p(z) = \mathcal{N}(0, I)$. Thus, since the prior $p(z)$ and its representation $q(z|X)$ have both Gaussian distributions, the discrepancy between them is directly computed using the Kullback-Leibler (KL) divergence:

$$D_{KL} = -\sum_{k=1}^{n}(1 + \nu(X_k) - \mu(X_k)^2 - \sigma_k^2). \tag{4.13}$$

During the decoding phase, the latent variables are randomly sampled from the probability distribution outputted by the encoder to reconstruct the input variable. The reconstructed vector $\hat{X}$ is expressed as:

$$\hat{X} = W_r f(W_h^{'} z + b_h^{'}) + b_r. \tag{4.14}$$

However, the latent randomly sampled variables are not differentiable, which makes it difficult to calculate gradients in the context of backpropagation. Therefore, to optimize both the encoding and decoding weights, a reparametrization trick is applied: the variable ϵ that verifies $z = \mu(X) + \sigma(X)\epsilon$ is sampled from the standard Gaussian distribution. The error used as the objective function for the VAE model is defined as follows:

$$\hat{\theta}_{VAE} = \underset{\theta}{\operatorname{argmin}} \left(L(X, \hat{X})_{recon} + D_{KL}(X) \right), \tag{4.15}$$

where θ_{VAE} refers to the parameters of the VAE model and $\hat{X}_{recon}$ is the root mean squared error.

4.3.3 . Combined FNNs

AE and VAE have been typically used to reduce the dimensionality of the data by extracting the data structure information. They can be applied in the context of transfer learning, which consists in extracting features with a pre-trained model learned on a large database for a problem of smaller dimensions. Transfer learning prevents overfitting and improves the performances of the FNNs.

Several architectures based on VAE and FNN that handle time-to-event data have been introduced. For instance, the VAECox architecture (Kim et al., 2020) is a combination of the encoder layers of a VAE model and a CoxPH model. First, a VAE is trained. Then, the weights are transferred to the encoder layers of the VAECox model, and the remaining weights are randomly initialized. Another architecture, X-VAE (Simidjievski et al., 2019), can handle heterogeneous data sources, like multi-omics and clinical data, using individual branches for each data source. The branches are combined into one before the bottleneck layer.

Based on these complex architectures adapted to time-to-event data, we propose to combine the encoder part of a pre-trained VAE model with each neural network architecture compared in this work. Here, as shown on the left side **(1)** of Figure 4.1, an AE (resp. VAE) is trained using the genomic data as input of the model. The latent space corresponds to the orange rectangle in the middle. Then, the encoder part is transferred into a combined model with two inputs: the encoder with the genomic data, and the clinical variables (right side **(2)** of Figure 4.1). This combined model with two input branches is retrained. As the number of patients in our data is small compared to the number of variables, pre-training of the weights and encoding can help extract salient features in the molecular data.

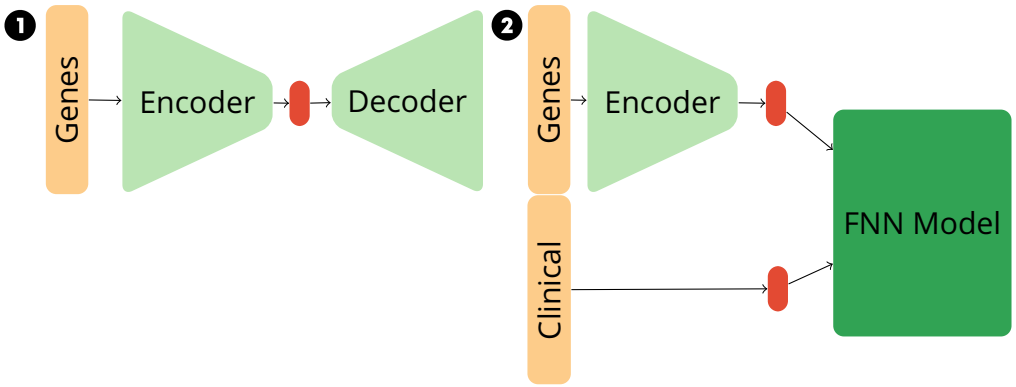


Figure 4.1: Construction and training of the AE and VAE models combined with a FNN. The scheme on the left part **(1)** represents the pre-training of the full AE/VAE model. The scheme on the right part **(2)** is the combined model with the pre-trained encoder part and the FNN model.

4.4 . Training Procedure

4.4.1 . Training Procedure of the FNN

To evaluate the performances of the different FNNs, the dataset is typically split into three sets (one training set, one validation set, and one test set). A CV procedure is also performed to obtain the hyperparameters of the FNNs.

A hyperparameter takes a value used to control the learning process, which cannot be estimated from data; the practitioner specifies it. Several hyperparameters are tuned to define the architecture of the model: the number of hidden layers, the number of neurons per layer, the activation function, and the optimization algorithm with different learning rates and batch sizes. To prevent overfitting and thus improve the accuracy of a deep learning model when new data is presented to the FNN, 2 regularization techniques are applied. The L_2 regularization, also called weight decay, extends the loss function by a regularization term defined as the euclidean norm of the weights ($pen_{L2} = ||W||_2^2$); the dropout consists of temporarily deactivating some neurons in the network during training. The range of values that will be tested for each hyperparameter is summarized in Table 4.1.

Hyperparameter	Value
Activation function	{elu, relu, tanh}
Batch size	$\{8, 16, 32, 64, 128, 256\}$
Dropout rate	$[0.01, 0.5]$
L_2 regularization	$[0.001, 0.1]$
Learning rate	$[0.001, 0.01]$
Number of hidden layers	$\{1, 2, 3, 4\}$
Number of neurons per layer	$[\![4, 128]\!]$
Optimization algorithm	{Adam, AdamAMSGRAD, RMSProp, SGDWR}

Table 4.1: Hyperparameter values tested for each FNN model.

We perform the hyperparameter search in the context of a 5-fold cross-valudation (CV) applied on the training set. First, the data is split into a training and a test set. Then we applied the 5-fold CV to the training set. It consists in randomly splitting the training data into $K = 5$ folds. The model is trained using $K - 1$ folds and validated on the remaining fold, outputting a score. The

training is repeated until each K-fold is used as a validation set. The average of the recorded scores is the performance metric of the model. Finally, the entire training set is used to train each network with the previously selected set of hyperparameters. The results are outputted on the test set.

We use the Tree-Parzen algorithm (Bergstra et al., 2011) to select hyperparameters iteratively in an informed manner (more details can be found in Appendix 8.2). We define a search space for each hyperparameter with specific distribution and boundary values. A set of hyperparameters is randomly sampled, and the model is scored on each of the five validation folds. These five validation scores are averaged, and a new set of hyperparameters is sampled based on the value of the average score. The sampling of hyperparameters sets is repeated 200 times.

For the real patient cohorts, we perform a double 5-fold CV on the entire dataset, as shown in Figure 4.2, because we have relatively small datasets and want to mimic an external test set. First, the real patient cohort is split into five folds: this is the outer loop. Then, we select one of the five folds as a test set and perform a 5-fold CV on the remaining data for each hyperparameter set: this is the inner loop. We choose the hyperparameters configuration with the minimum average validation loss obtained on the five folds of the inner loop. Finally, we fit the model with these optimal hyperparameters on the four folds of the outer loop and calculate predictions on the remaining test fold of the outer loop. This procedure is repeated on all the folds of the outer loop.

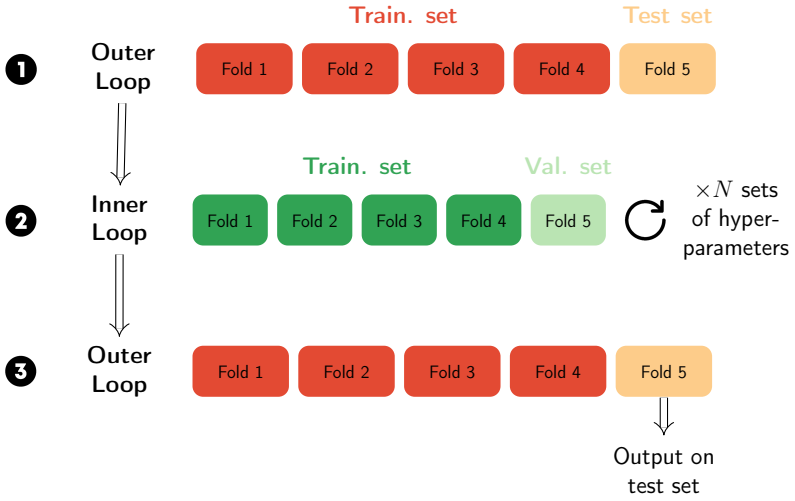


Figure 4.2: Double 5-fold CV.

4.4.2 . Training Procedure of the benchmark models

Two benchmark models are used, LASSO and RSF. The LASSO model is implemented in the R package biospear. The lasso penalty is chosen by maximum cross-validate likelihood. The RSF model is implemented in Python using the scikit survival package. This implementation is based on the work of Ishwaran et al. (2008) and the corresponding R package randomSurvivalForest (Ishwaran et al., 2007). 4 hyperparameters are tuned: the maximum depth of the tree, the minimum number of samples required at a leaf node, the number of randomly selected variables to look for the best split, and the number of trees grown in the forest. Table 4.2 reports the range of values tested for each hyperparameter by 5-fold CV using the prediction error. The splitting criteria used to split each node is the log-rank splitting rule.

Hyperparameter	Value
Max depth of each tree	$\{8, 32, 64, 128\}$
Minimum number of samples at a leaf node	$\{8, 10, 20, 50\}$
Number of variables per split	$\{2, 4, 6, 8, 10\}$
Number of trees	$\{10, 100, 250\}$

Table 4.2: Hyperparameter values tested for the RSF model.

4.5 . Simulation study to compare FNNs

The performance of the different neural network models is further evaluated and compared using a simulation study. The advantage of simulated data is that we know its true characteristics. Various types of data are simulated based on either a CoxPH or an AFT model, and on the distribution of the survival times. First, a CoxPH model is used in association with a Weibull distribution in order to simulate data respecting the proportional hazards assumption. Then, an AFT model is applied with a log-normal distribution as it does not rely on the proportionality assumption. To overcome this linear framework without interactions, we also simulate data using an AFT model with Friedman's random function generator. It enables us to simulate nonlinear data with high-order interactions.

4.5.1 . Simulations with proportional hazards

Following Bender et al. (2005), we generate survival data respecting the proportional risk assumption using the inverse cumulative distribution. Let T be the survival time from the CoxPH model, with distribution function F, then $U = F(Y)$ follows a uniform distribution $U \sim U_{[0,1]}$. Thus, it follows:

$$U = exp[-H_0(T)exp(\beta^T X)]. \tag{4.16}$$

If $U \sim U_{[0,1]}$, then $1 - U \sim U_{[0,1]}$. Furthermore, if $h_0(t) > 0$ for all t, then the cumulative baseline hazard function H_0 can be inverted and the survival time T of the CoxPH model can be expressed as :

$$T = H_0^{-1}\left[\frac{-log(1-U)}{exp(-\beta^T X)}\right]. \tag{4.17}$$

Moreover, we suppose that survival times are distributed according to a Weibull distribution $\mathcal{W}(a, \lambda)$, where $a > 0$ and $\lambda > 0$. Thus the baseline function is of the form $h_0(t) = a\lambda^a t^{a-1}$ and the cumulative baseline hazard function is written:

$$H_0(t) = \lambda t^a. \tag{4.18}$$

The inverse of this function is expressed as follows:

$$H_0^{-1}(u) = \left(\frac{u}{\lambda}\right)^{1/a}. \tag{4.19}$$

Based on this idea, a variable U is generated from a uniform distribution: $U \sim U_{[0,1]}$. The survival times are simulated as follows:

$$T = \frac{1}{\lambda^{1/a}}\left(\frac{-\log(1-U)}{\exp(\beta^T X)}\right)^{1/a}. \tag{4.20}$$

From the relation between the conditional survival function and the conditional hazard rate function, we have for $i = 1, \ldots, n,$:

$$S(t, X) = \exp\left[-\int_0^t h_0(s, X)ds\right] = \exp(-\lambda^a t^a \exp(\beta^T X)). \tag{4.21}$$

We call this method CoxPH-Weibull. To define a and λ, we take values so as to obtain survival times that are similar to real patient cohort survival time distribution. Based on the METABRIC data, we want to obtain an expectancy equal to 13.2 and a variance of 25.92.

The mean and the variance of simulated survival times rise with the increase in the number of variables q. A normalization term can be included in the Equation 4.20 to correct for this increase:

$$T = \frac{1}{\lambda^{1/a}}\left(\frac{-\log(1-U)}{\exp(\frac{1}{\sqrt{q}}\beta^T X)}\right)^{1/a}. \tag{4.22}$$

4.5.2 . Simulations with non-proportional hazards

Based on Leemis et al. (1990), we simulate data based on the AFT model that does not respect the proportional hazard assumption. The survival time of the AFT model can be expressed as follows:

$$T = \frac{H_0^{-1}\left[-\log(1-U)\right]}{\exp(\beta^T X)}.$$

We consider that survival times follow a log-normal distribution $\mathcal{LN}(\lambda, a)$, with $\lambda \in]-\infty, +\infty[$ and $a > 0$. The baseline hazard is of the form:

$$h_0(t) = \frac{\frac{1}{a\sqrt{2\pi t}}\exp\left(-\frac{(\log t - \lambda)^2}{2a^2}\right)}{1 - \Phi\left(\frac{\log t - \lambda}{a}\right)},$$

where $\Phi(t)$ is the cumulative distribution function of the standard normal distribution. The cumulative baseline hazard is written:

$$H_0(t) = -\log\left[1 - \Phi\left(\frac{\log t - \lambda}{a}\right)\right].$$

The survival times are simulated as follows:

$$T = \frac{1}{\exp(\beta^T X)\exp(a\phi^{-1}(U) + \lambda)}. \tag{4.23}$$

Finally, the survival function is written as follows:

$$S(t|X) = 1 - \phi\left[\frac{\log(t\exp(\beta^T X)) - \lambda}{a}\right]. \tag{4.24}$$

We refer to this method as AFT-LN.

As for CoxPH-Weibull, the mean and the variance of survival times rise with the increase in the number of variables q. Thus, we also add a normalization term in the Equation 4.5.2:

$$T = \frac{H_0^{-1}\left[-\log(1-U)\right]}{\exp(\frac{1}{\sqrt{p}}\beta^T X)}.$$

4.5.3 . Simulations with non-linearity and high-order interactions

We also applied Friedman's random function generator to simulate data (Friedman, 2001). It allowed us to generate random functions with high-order interactions between the survival data and the explanatory variables and strong nonlinear effects. It is based on an AFT model, that assumes that the relationship between the logarithm of survival time T and the variables is linear. The random function generator is defined as:

$$log\ T = m(X) + W \text{ with } W \sim \Gamma(2,1). \tag{4.25}$$

A vector of variables $X = (X_1, ..., X_{20})$ is generated with $X \sim \mathcal{N}(0, I)$. $m(X)$ is defined by Friedman's random function generator:

$$m(X) = \sum_{l=1}^{20} a_l g_l(R_l). \tag{4.26}$$

$\{a_l\}_1^{20}$ are randomly generated from a uniform distribution ($a_l \sim \mathcal{U}_{[-1,1]}$). R_l is a random subset of the input vector X of size n_l. The size of each subset, n_l, is itself random, with: $n_l = min(\lfloor 2 + r \rfloor, 20)$ and $r \sim \mathcal{E}(1/2)$. The expected number of input variables for each $g_l(R_l)$ is 4, implying high-order interaction effects. Hence, at least a few of the 20 functions $g_l(R_l)$ will involve high-order interactions, and $m(X)$ will highly probably be a function of all input variables. With Friedman's function generator, the input variables are associated with the survival time at different levels:

$$g_l(R_l) = \exp\{-\frac{1}{2}(R_l - \mu_l)^T V_l (R_l - \mu_l)\}. \tag{4.27}$$

Each mean vector $\{\mu_l\}_1^{20}$ is randomly generated with $\mu_l \sim \mathcal{N}(0,1)$. The matrix of variance-covariance V_l is also randomly generated: $V_l = U_l D_l U_l^T$ with U_l an orthonormal random matrix, $D_l = diag\{d_{l,1}, ..., d_{l,n_l}\}$ and $\sqrt{d_{lk}} \sim \mathcal{U}(u, v)$. u, v are chosen according to the distribution of the input variables X: there are the eigenvalue limits with $u = 0.1$ and $v = 2.0$. Finally, we obtain the survival times by applying the exponential: $\exp(\log(T))$. This is the AFT-Friedman method.

4.5.4 . Simulation scenarios

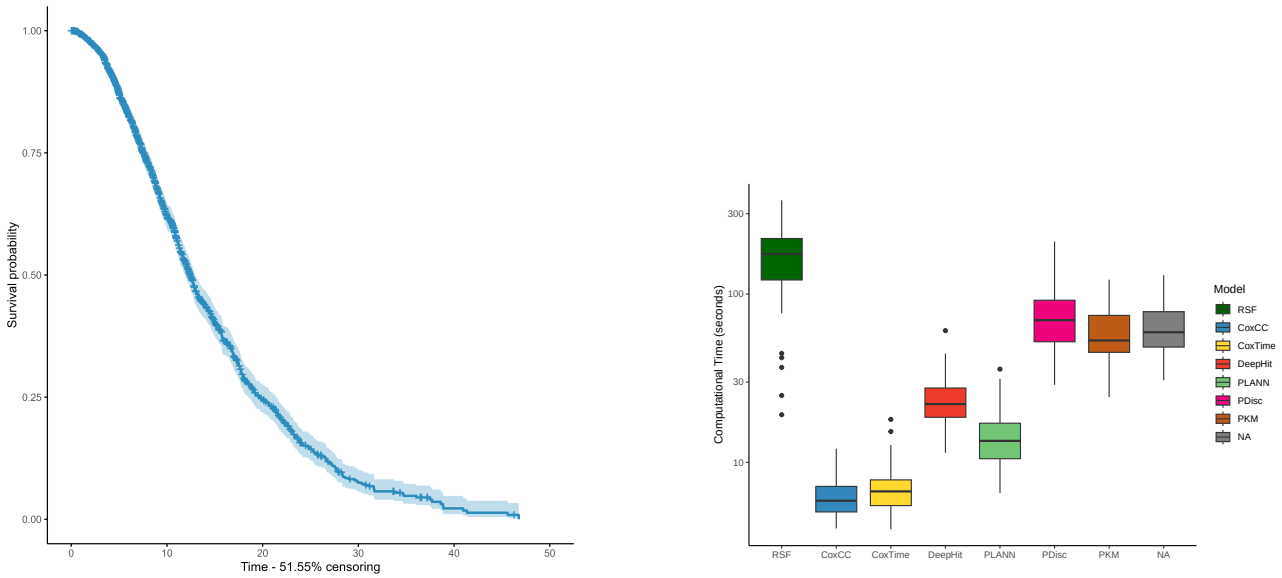
For the 2 first simulation types (CoxPH-Weibull, AFT-LN), a cohort of size n individuals ($n \in \{100, 1000\}$) and p variables ($p \in \{10, 100, 1,000\}$) is generated for the training set. Among the p variables, only q biomarkers are really prognostic and q increases as the total number of candidate biomarkers p increases. Thus, for $p = 10$, 100 and $1,000$ candidate biomarkers, $q = 2(20\%)$, $10(10\%)$ and $20(5\%)$ prognostic biomarkers are chosen. For each scenario, the relative risk reduction for the increase of one unit in a prognostic biomarker q is set to 50% corresponding to a hazard-ratio equal to 0.5 (i.e. $\beta_j = log(0.5) = -0.3$). For each n and p, the design matrix $X = (X_{i,j})_{1 \leq i \leq n, 1 \leq j \leq p}$ is simulated independently from a normal distribution (resp. from a uniform distribution) for the CoxPH-Weibull simulations (resp. for the AFT-LN simulations) . For each scenario, 100 replications are performed. All the test sets include $n = 1,000$ individuals. Censoring times are generated independently from the survival times via an exponential distribution $\mathcal{E}(\frac{1}{\gamma E(T)})$, where $\gamma > 0$ is a constant to be adjusted to the rate of censorship. We consider a large rate of censoring of 50% by taking $\gamma = 1.2$. For these two simulation settings, we know exactly the true underlying model that only includes the active biomarkers. Thus, we compute the reference C-statistic and IBS for this Oracle model, that is the unpenalized CoxPH model fitted to the truly related biomarkers.

For the last simulation type (AFT-Friedman), a cohort of size n individuals ($n \in \{100, 1000\}$) is generated for the training set. For each scenario, 100 replications are performed. All the test sets include $n = 1,000$ individuals. We consider two censoring rates: a moderate censoring rate of 20% by taking $\gamma = 4.5$ and a large censoring rate of 50% by taking $\gamma = 1.2$.

For each model, we perform a 5-fold CV on 200 different sets of parameters on the simulation data sets. Then we compute the C-statistic and the IBS with a fixed horizon at 5 years for the models trained on the finally chosen set of parameters.

4.5.5 . Results for CoxPH-Weibull data

The results reported here are the ones obtained using data simulated from a CoxPH model, in which the basis risk is modeled by a Weibull law. In this simulation scenario, data satisfy the proportional-hazards assumption. In Figure 4.3a, we can see the KM curve obtained based on one simulation set. We compared the computational times of the models trained in Python during the hyperparameter search performed with the Tree-Parzen algorithm. The hyperparameter search was computed on a CPU with 2.1 GHz and 4 GB RAM for each model and simulation set. Figure 4.3b shows 4 groups of models in terms of running times that correspond to different characteristics of the models. RSF was the slowest model in terms of computational time, followed by the pseudo-observation-based models, and the discrete-time models (DeepHit and PLANN), CoxCC, and CoxTime were the fastest. For instance, the median computing time for RSF is ten times longer than for DeepHit. The difference between the discrete-time models and the pseudo-observation-based models with the two fastest models can be explained by the structure of the models. Pseudo-observation-based FNNs and PLANN are trained using a larger n. RSF is built on an ensemble of decision trees, and not a single model, as is the case for the neural network models.



(a) Survival Curve

(b) Computational Time

Figure 4.3: On the left size, survival curve for one simulated set with the CoxPH-Weibull method, $q = 2$, $p = 10$ and $n = 2,000$. On the right side,computational time in seconds of the models on the logarithmic scale for CoxPH-Weibull data. Each boxplot displays 100 points corresponding to each simulated set (with $n = 2,000$, $p = 10$, $q = 2$). Each point represents the average over the 200 iterations of one 5-fold CV.

Table 4.3 reports the results for the different models in terms of C-statistic and IBS. The models were compared under 2 different training sample sizes per data set. The models performed

better when trained on a larger sample size ($n = 1,000$), which seems consistent, as machine learning algorithms usually perform well if the sample size is large. The models also obtained better performances when the number of prognostic variables q was close to the total number of variables p, that is, when the signal-to-noise ratio was high.

Overall, LASSO performed the best: the highest C-statistic was obtained with $n = 1,000$, $p = 10$, $q = 2$, while the lowest IBS was reached with $n = 1,000$, $p = 100$, $q = 10$. The model's performances were close to the Oracle model ones. These results are not surprising as the data is simulated based on a CoxPH model: the penalized CoxPH model, or LASSO, is well-suited to this context.

If we only consider the FNN-based models, CoxCC and CoxTime obtained the best performances. Specifically, CoxCC performances are close to the reference method ones: for instance, with $n = 1,000$, $p = 10$, and $q = 2$, the average C-statistic of CoxCC is 0.692, while it is 0.707 for the reference method. DeepHit and PLANN reached lower performances compared to CoxCC and CoxTime, with PLANN obtaining slightly better results than DeepHit. Pseudo-observation based models got systematically lower results, which can be explained by the increase in the number of input variables: the complexity of the networks increased, and their predictive ability decreased.

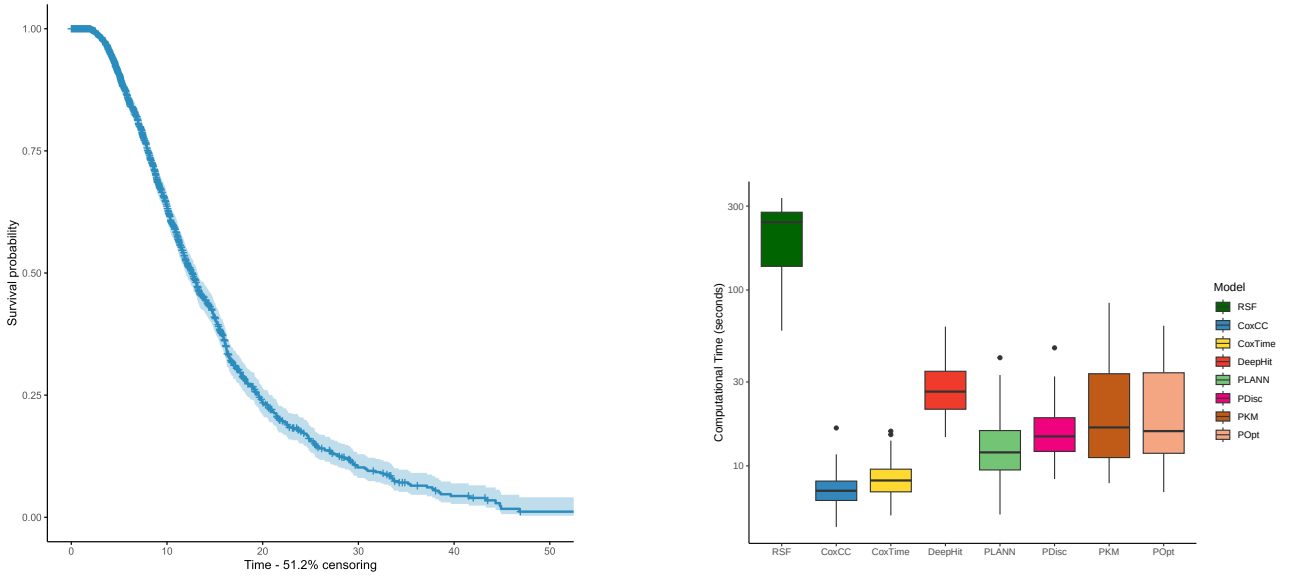
Table 4.3: Average C-statistic and IBS at 5 years across the 100 simulation sets for the CoxPH-Weibull data.

n		100			1,000		
p		10	100	1,000	10	100	1,000
q		2	10	20	2	10	20
Oracle	*C*	0.707 (±0.024)	0.710 (±0.026)	0.706 (±0.027)	0.707 (±0.024)	0.710 (±0.026)	0.706 (±0.027)
	IBS	0.050 (±0.006)	0.051 (±0.0052)	0.050 (±0.0048)	0.050 (±0.006)	0.051 (±0.0052)	0.050 (±0.0048)
LASSO	*C*	**0.699** (**±0.025**)	**0.607** (**±0.055**)	**0.516** (**±0.037**)	**0.705** (**±0.024**)	**0.699** (**±0.027**)	**0.655** (**±0.03**)
	IBS	**0.051** (**±0.006**)	**0.053** (**±0.006**)	**0.053** (**±0.005**)	0.053 (±0.006)	**0.051** (**±0.005**)	**0.052** (**±0.005**)
RSF	*C*	0.606 (±0.042)	0.531 (±0.034)	0.504 (±0.029)	0.662 (±0.03)	0.603 (±0.029)	0.513 (±0.029)
	IBS	0.056 (±0.008)	0.056 (±0.006)	0.056 (±0.006)	0.053 (±0.005)	0.054 (±0.005)	0.053 (±0.005)
CoxCC	*C*	0.629 (±0.047)	0.548 (±0.039)	0.505 (±0.028)	0.692 (±0.027)	0.651 (±0.032)	0.546 (±0.029)
	IBS	0.056 (±0.007)	0.056 (±0.006)	0.055 (±0.006)	**0.051** (**±0.006**)	0.053 (±0.006)	0.053 (±0.005)
CoxTime	*C*	0.625 (±0.044)	0.542 (±0.034)	0.508 (±0.033)	0.686 (±0.032)	0.648 (±0.035)	0.551 (±0.029)
	IBS	0.056 (±0.007)	0.114 (±0.012)	0.056 (±0.006)	**0.051** (**±0.006**)	0.053 (±0.006)	0.053 (±0.005)
DeepHit	*C*	0.588 (±0.058)	0.528 (±0.041)	0.508 (±0.027)	0.660 (±0.059)	0.602 (±0.054)	0.531 (±0.034)
	IBS	0.093 (±0.033)	0.089 (±0.014)	0.084 (±0.012)	0.082 (±0.011)	0.084 (±0.011)	0.083 (±0.01)
PLANN	*C*	0.605 (±0.054)	0.542 (±0.039)	0.508 (±0.032)	0.691 (±0.026)	0.634 (±0.035)	0.530 (±0.031)
	IBS	0.070 (±0.027)	0.073 (±0.03)	0.094 (±0.079)	0.060 (±0.005)	0.064 (±0.005)	0.066 (±0.007)
PDisc	*C*	0.503 (±0.028)	0.509 (±0.037)	0.514 (±0.034)	0.584 (±0.033)	0.519 (±0.034)	0.513 (±0.029)
	IBS	0.127 (±0.047)	0.118 (±0.04)	0.103 (±0.024)	0.107 (±0.021)	0.103 (±0.023)	0.089 (±0.018)
PKM	*C*	0.632 (±0.028)	0.506 (±0.027)	0.506 (±0.032)	0.633 (±0.029)	0.508 (±0.03)	0.514 (±0.029)
	IBS	0.129 (±0.053)	0.217 (±0.056)	0.284 (±0.067)	0.231 (±0.03)	0.249 (±0.019)	0.283 (±0.042)
POpt	*C*	0.502 (±0.03)	0.504 (±0.026)	0.506 (±0.031)	0.633 (±0.028)	0.509 (±0.03)	0.513 (±0.032)
	IBS	0.245 (±0.05)	0.217 (±0.051)	0.292 (±0.07)	0.233 (±0.026)	0.248 (±0.022)	0.284 (±0.043)

Notes. The value in brackets is the standard deviation across the 100 simulation sets. The highest C-statistic values and lowest IBS values are in bold.

4.5.6 . Results for AFT-LN data

For this second simulation setting, the data are simulated from an AFT model with a log-normal basis risk. Figure 4.4a represents the survival curve of a simulation set. In Figure 4.4b, we can observe 3 groups of models in terms of computation time. RSF is the slowest, and CoxCC the fastest. Discrete-time models and pseudo-observation based models are inbetween.



(a) Survival Curve

(b) Computational Time

Figure 4.4: On the left side, survival curve for one simulated set with the AFT-LN method, $q = 2$, $p = 10$ and $n = 2,000$. On the right side, computational time in seconds of the models on the logarithmic scale for CoxPH-Weibull data. Each boxplot displays 100 points corresponding to each simulated set (with $n = 2,000$, $p = 10$, $q = 2$). Each point represents the average over the 200 iterations of one 5-fold CV.

The results obtained with AFT-LN data were close to the ones obtained with CoxPH-Weibull data. It can be observed that all the models obtained better results when they were trained on more data. For instance, for CoxCC, the C-statistic went from 0.876 with $n = 100$, $p = 10$, and $q = 2$ synthetic patients per dataset to 0.912 with $n = 1,000$. LASSO's performances are close to the Oracle model. The LASSO model was able to identify the relevant prognostic variables (Appendix 8.3.1). For the FNN-based methods, the best-performing models were CoxCC and CoxTime. They obtained really close results. The pseudo-observation based models did not perform well.

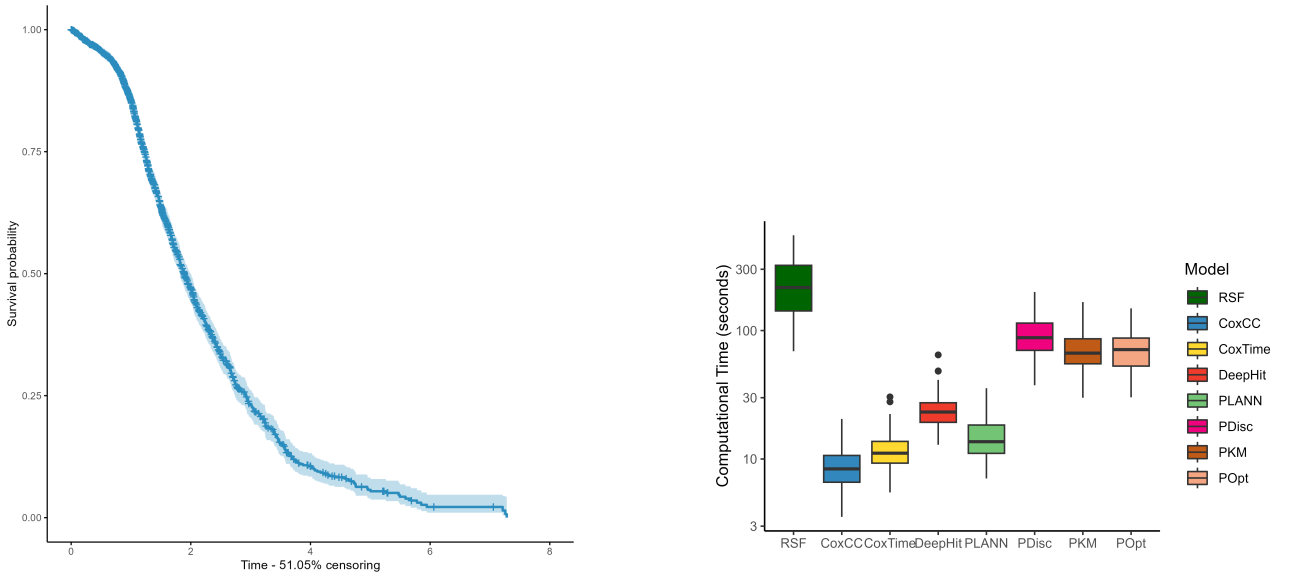
Table 4.4: Average C-statistic and IBS at 5 years across the 100 simulation sets for the AFT-LN data.

n		100			1,000		
p		10	100	1,000	10	100	1,000
q		2	10	20	2	10	20
Oracle	***C***	0.921 (±0.0104)	0.927 (±0.0089)	0.926 (±0.0116)	0.921 (±0.0104)	0.927 (±0.0089)	0.926 (±0.0116)
	IBS	0.0231 (±0.0025)	0.0215 (±0.0028)	0.0218 (±0.0029)	0.0231 (±0.0025)	0.0215 (±0.0028)	0.0218 (±0.0029)
LASSO	***C***	**0.918** (±**0.011**)	**0.895** (±**0.024**)	**0.658** (±**0.083**)	**0.920** (±**0.010**)	**0.922** (±**0.009**)	**0.909** (±**0.014**)
	IBS	0.027 (±0.003)	**0.029** (±**0.012**)	**0.033** (±**0.004**)	0.027 (±0.003)	0.027 (±0.003)	**0.028** (±**0.081**)
RSF	***C***	0.853 (±0.038)	0.628 (±0.056)	0.513 (±0.034)	0.904 (±0.015)	0.814 (±0.025)	0.59 (±0.035)
	IBS	**0.004** (±**0.03**)	0.034 (±0.004)	0.035 (±0.005)	**0.025** (±**0.003**)	0.032 (±0.004)	0.034 (±0.005)
CoxCC	***C***	0.876 (±0.025)	0.656 (±0.071)	0.523 (±0.043)	0.912 (±0.013)	0.886 (±0.02)	0.703 (±0.043)
	IBS	0.029 (±0.004)	0.034 (±0.005)	0.035 (±0.005)	**0.025** (±**0.003**)	**0.026** (±**0.004**)	0.033 (±0.005)
CoxTime	***C***	0.868 (±0.034)	0.656 (±0.073)	0.528 (±0.037)	0.912 (±0.011)	0.89 (±0.019)	0.705 (±0.037)
	IBS	0.029 (±0.004)	0.034 (±0.005)	0.035 (±0.005)	**0.025** (±**0.003**)	**0.026** (±**0.003**)	0.033 (±0.005)
DeepHit	***C***	0.827 (±0.071)	0.618 (±0.074)	0.533 (±0.042)	0.89 (±0.052)	0.828 (±0.081)	0.611 (±0.068)
	IBS	0.077 (±0.024)	0.073 (±0.011)	0.072 (±0.017)	0.07 (±0.017)	0.074 (±0.02)	0.072 (±0.025)
PLANN	***C***	0.847 (±0.054)	0.640 (±0.057)	0.539 (±0.04)	0.909 (±0.015)	0.858 (±0.025)	0.628 (±0.052)
	IBS	0.052 (±0.036)	0.060 (±0.047)	0.06 (±0.044)	0.045 (±0.003)	0.048 (±0.004)	0.052 (±0.006)
PDisc	***C***	0.506 (±0.034)	0.508 (±0.031)	0.516 (±0.038)	0.507 (±0.033)	0.549 (±0.052)	0.554 (±0.045)
	IBS	0.145 (±0.063	0.10 (±0.042)	0.086 (±0.023)	0.104 (±0.03)	0.086 (±0.028)	0.071 (±0.015)
PKM	***C***	0.507 (±0.035)	0.519 (±0.039)	0.518 (±0.033)	0.507 (±0.035)	0.544 (±0.035)	0.531 (±0.038)
	IBS	0.365 (±0.063)	0.236 (±0.065)	0.293 (±0.065)	0.367 (±0.042)	0.219 (±0.048)	0.266 (±0.022)
POpt	***C***	0.508 (±0.036)	0.52 (±0.037)	0.517 (±0.035)	0.507 (±0.035)	0.533 (±0.037)	0.543 (±0.034)
	IBS	0.365 (±0.068)	0.238 (±0.06)	0.296 (±0.067)	0.366 0.041	0.264 (±0.022)	0.328 (±0.042)

Notes. The value in brackets is the standard deviation across the 100 simulation sets. The highest C-statistic values and lowest IBS values are in bold.

4.5.7 . Results for AFT-Friedman data

For the third simulation setting, we considered an AFT model, including a flexible random functions generator. As shown in Figure 4.5a, survival times are shorter compared to the other simulations. Here, we did not choose the parameters in order to mimic a cohort of real patients. Instead, we kept the parameter values presented in the original paper (Friedman, 2001). Figure 4.5b shows 4 groups of models in terms of running times that correspond to different characteristics of the models. RSF was the slowest model in terms of computational time, followed by pseudo-observation-based models, and the discrete-time models (DeepHit and PLANN), CoxCC, and CoxTime were the fastest.



(a) Survival Curve

(b) Computational Time

Figure 4.5: On the left side, survival curve for one simulated set with the AFT-Friedman method, $r = 0.5$ and $n = 2,000$. On the right side, the computational time in seconds of the models on the logarithmic scale for CoxPH-Weibull data. Each boxplot displays 100 points corresponding to each simulated set (with $n = 2,000$, $r = 0.5$). Each point represents the average over the 200 iterations of one 5-fold CV.

In Table 4.5, the models are compared based on the C-statistic, and the IBS obtained at the median survival time of the test set.

The predictive and discriminative performances for all models improved when the censoring rate decreased. Overall, RSF obtained the best performances. Indeed, the best-performing model in terms of numerically highest C-statistic was RSF for both 20% censoring and 50% censoring. It is not surprising, as the simulation method implemented here was initially designed for RSF.

LASSO only outperformed the other methods, especially RSF, in the case of a small sample size and high censoring.

Regarding the FNN-based methods only, CoxCC, CoxTime, DeepHit, and PLANN obtained high C-statistics and low IBS for a larger sample size. The results were also improved when the censoring rate was low. The results were very similar for CoxCC and CoxTime. Pseudo-observation based methods performed the worst.

Table 4.5: Average C-statistic and IBS for the median survival time across the 100 simulation sets for the AFT-Friedman data.

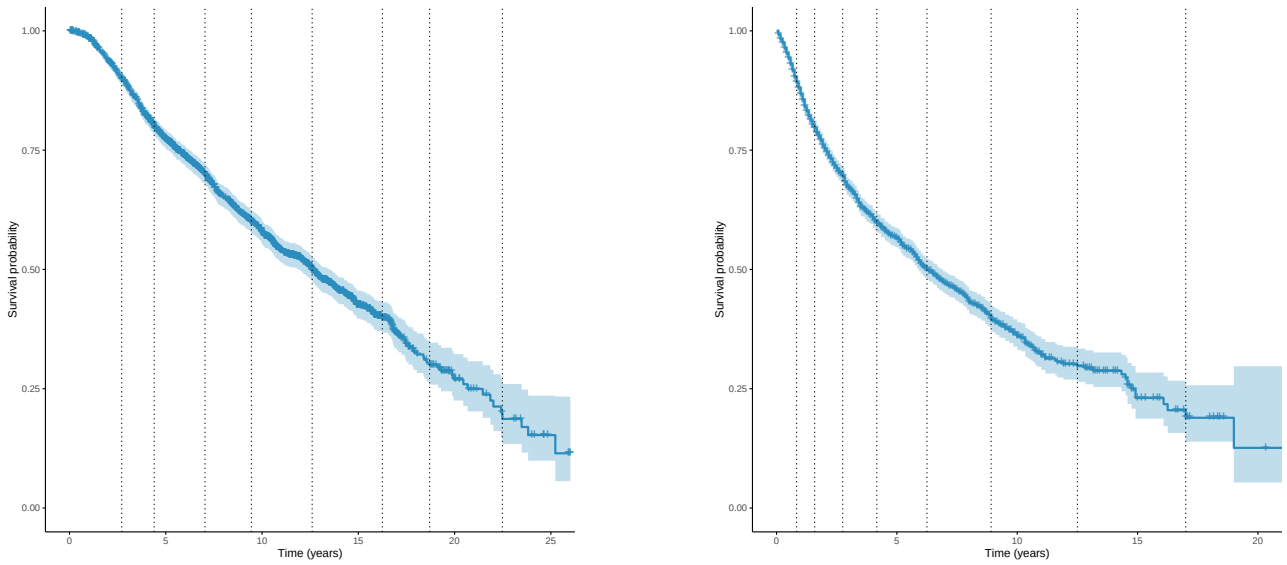
n		100		1,000	
Censoring		20%	50%	20%	50%
LASSO	*C*	0.705 (±0.067)	**0.634 (±0.061)**	0.645 (±0.123)	0.670 (±0.046)
	IBS	**0.144 (±0.023)**	**0.140 (±0.019)**	0.159 (±0.035)	0.134 (±0.016)
RSF	*C*	**0.709 (±0.052)**	0.609 (±0.050)	**0.822 (±0.034)**	**0.720 (±0.038)**
	IBS	0.146 (±0.020)	0.150 (±0.019)	0.112 (±0.019)	**0.126 (±0.013)**
CoxCC	*C*	0.657 (±0.062)	0.580 (±0.049)	0.816 (±0.046)	0.688 (±0.043)
	IBS	0.162 (±0.025)	0.266 (±0.032)	**0.106 (±0.019)**	0.13 (±0.013)
CoxTime	*C*	0.664 (±0.062)	0.585 (±0.053)	0.812 (±0.039)	0.687 (±0.044)
	IBS	0.160 (±0.026)	0.159 (±0.022)	0.107 (±0.019)	0.13 (±0.014)
DeepHit	*C*	0.645 (±0.074)	0.571 (±0.053)	0.760 (±0.065)	0.645 (±0.073)
	IBS	0.182 (±0.027)	0.172 (±0.02)	0.166 (±0.057)	0.167 (±0.061)
PLANN	*C*	0.647 (±0.062)	0.581 (±0.047)	0.778 (±0.053)	0.677 (±0.045)
	IBS	0.173 (±0.029)	0.170 (±0.031)	0.146 (±0.031)	0.144 (±0.014)
PDisc	*C*	0.501 (±0.04)	0.498 (±0.038)	0.504 (±0.046)	0.504 (±.038)
	IBS	0.274 (±0.051)	0.250 (±0.053)	0.278 (±0.043)	0.248 (±0.039)
PKM	*C*	0.504 (±0.05)	0.506 (±0.043)	0.505 (±0.051)	0.507 (±0.044)
	IBS	0.413 (±0.049)	0.383 (±0.042)	0.419 (±0.042)	0.391 (±0.030)
POpt	*C*	0.506 (±0.049)	0.506 (±0.043)	0.505 (±0.051)	0.506 (±0.045)
	IBS	0.414 (±0.048)	0.386 (±0.039)	0.427 (±0.043)	0.39 (±0.03)

Notes. The value in brackets is the standard deviation across the 100 simulation sets. The highest C-statistic values and lowest IBS values are in bold.

4.6 . Application on cancer cohorts

4.6.1 . Data description

The example data set we used is the METABRIC cohort, which can be extracted from the MetaGxBreast package (Gendoo et al., 2019). It consists of clinical variables and large-scale gene expression data of breast cancer patients obtained at surgery. We used a nonspecific filter independent from outcome based on standard deviation to select genes and increase the statistical power of the results (Bourgon et al., 2010), and retained the 863 genes with the highest standard deviation. If multiple probes corresponded to the same gene, we selected the probe with the highest variance. Regarding the clinical variables, we used the age at diagnosis, the grade, the tumor size, the number of invaded lymph nodes, the hormonal therapy indicator, and the chemotherapy indicator. We removed individuals with missing values for the survival time. Since one or more variable values were missing for 106 patients, we imputed the missing values using predictive mean matching for numerical variables and a multinomial logit model for categorical variables (Buuren et al., 2011). Then, we standardized the numerical variables using Z-score normalization and applied one-hot encoding to categorical variables. The final data set represents $1,960$ patients and 869 variables, with a median survival time of 88 months and a censoring rate of 54.6%.



(a) METABRIC

(b) Lung Cancer Explorer

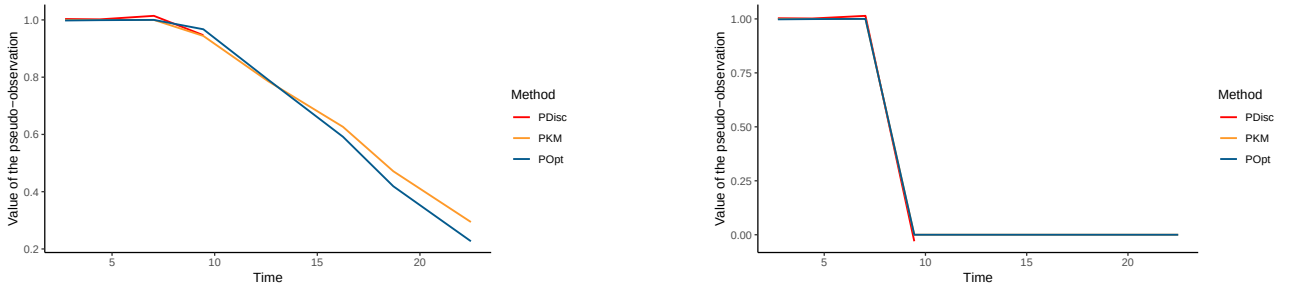
Figure 4.6: Kaplan Meier survival curved for the METABRIC data (Figure 4.6a) and the LCE data (Figure 4.6b). The dotted lines correspond to the percentiles of the empirical overall survival distribution (from the 10th to the 80th): these are the time points used to compute pseudo-observations.

We applied our methods to a second data set: the LCE. It is an online tool created by UT Southwestern Medical Center's Quantitative Biomedical Research Center, which is composed of more than $6,700$ patients over 56 data sets, including the Cancer Genomics Atlas and various Gene Expression Omnibus data sets. All the data sets provide gene expression and clinical data from lung cancer patients. The expression data stems from 23 genome-wide platforms and is predominantly composed of microarrays. It is reprocessed, normalized, and converted from probe to gene level. In our work, we selected two clinical variables: cancer stage and the age of the patient. For the transcriptomic data, only the genes with less than 5% of missing values were kept. Then, the $1,000$ genes with the highest standard deviation among the patients were conserved. The missing

values were replaced with the multiple imputation method using chained equations. More precisely, we used Bayesian polynomous regression for the stage categories and predictive mean matching for continuous variables. We standardized the numerical variables using Z-score normalization and applied one-hot encoding to the categorical variables. The final dataset includes $4,120$ patients.

4.6.2 . Results for the METABRIC cohort

In this section, the following models were compared: LASSO and RSF as benchmark models, CoxCC and CoxTime as continuous models, DeepHit and PLANN as discrete models, and PDisc, PKM, and POpt as models based on pseudo-observations. FNN-based methods and benchmark methods were applied to the METABRIC breast cancer cohort. All comparison measures were computed on the test set at 5 years. This time was chosen as this is a usual time of interest for clinical investigators for early breast cancer. Separate models are constructed with genomic and clinical variables and only clinical variables for each dataset.



(a) Patient with a censored survival time

(b) Patient with an observed event.

Figure 4.7: Values of the pseudo-observations computed for two patients of the METABRIC cohort. In Figure 4.7a, the survival time of the patient is censored at 8.5 years. In Figure 4.7b, the patient died at time 7.7.

Regarding C-indices, we see in Table 4.6 that the best-performing model is RSF. CoxCC obtains the numerically highest value for IBS. Overall, neural network models and RSF obtained relatively close results.

Figure 4.8a shows the Brier Score corresponding to each method over time. BS increases over time, regardless of the model considered. LASSO, PKM, POpt, and LASSO performed worse than the other models when all variables were included, especially after 5 years. LASSO obtained better performances when only the clinical variables were included.

All the models had better predictive performances and discriminative capacities at 5 years when the input variables were restricted to the clinical variables only. The best-performing model is PKM in terms of C-statistic and CoxTime in terms of IBS. Let us look at the weights associated with the variables selected by the LASSO when all variables are included in the model. We observe that the largest weights in absolute values are associated with the clinical variables. Hence, clinical variables do seem to contain a significant portion of the predictive information for survival.

Then, AE and VAE models were applied to the molecular data to synthesize information contained in transcriptomic data, and thus reduce the number of dimensions of the data while keeping the major part of the data structure information. An AE/VAE was trained on the genomic data, and the encoder part of the model was transferred into a combined model. As reported in Table 4.7, all the combined models performed better with the VAE-based model. With VAE pre-training, the best results were obtained with the CoxTime in terms of C-statistic and CoxCC for IBS. Overall,

CoxTime obtained the highest value in terms of C-statistic and the lowest value of IBS for all the configurations and for all the models considered.

Table 4.6: Average C-statistic and IBS at 5 years for METABRIC data.

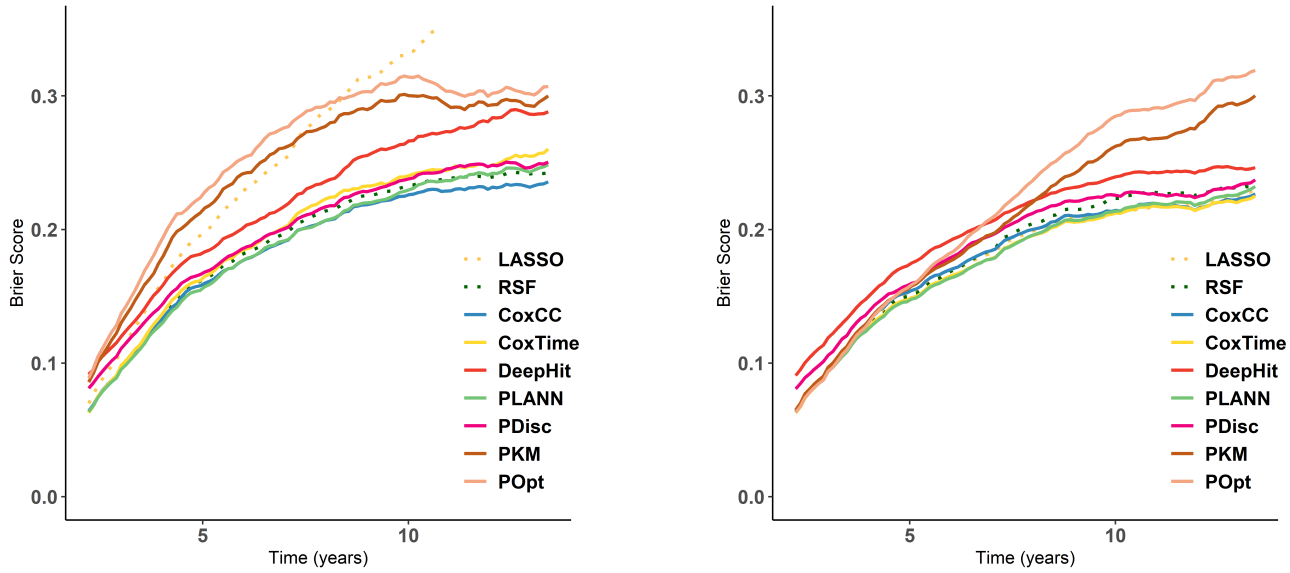
	All		Clinical	
Model	***C***	***IBS***	***C***	***IBS***
LASSO	0.558 (±0.03)	0.142 (±0.008)	0.680 (±0.032)	0.116 (±0.005)
RSF	**0.691** (±**0.036**)	0.121 (±0.006)	0.714 (±0.023)	0.116 (±0.005)
CoxCC	0.666 (±0.038)	0.120 (±0.005)	0.676 (±0.046)	0.118 (±0.008)
CoxTime	0.677 (±0.002)	0.122 (±0.007)	0.712 (±0.026)	**0.114** (±**0.007**)
DeepHit	0.660 (±0.078)	0.143 (±0.021)	0.695 (±0.03)	0.139 (±0.007)
PLANN	0.681 (±0.042)	**0.117** (±**0.006**)	0.714 (±0.03)	0.115 (±0.004)
PDisc	0.649 (±0.03)	0.132 (±0.009)	0.689 (±0.029)	0.126 (±0.007)
PKM	0.655 (±0.014)	0.161 (±0.018)	**0.718** (±**0.021**)	0.119 (±0.012)
POpt	0.640 (±0.018)	0.170 (±0.013)	0.709 (±0.028)	0.117 (±0.005)

Notes. The value in brackets is the standard deviation across the 5 folds. The highest C-statistic values and lowest IBS values are in bold.

Table 4.7: Average C-statistic and IBS at 5 years for METABRIC data.

	AE		VAE	
Model	***C***	***IBS***	***C***	***IBS***
CoxCC	0.624 (±0.033)	0.123 (±0.007)	0.707 (±0.047)	0.122 (±0.006)
CoxTime	0.617 (±0.029)	0.124 (±0.007)	**0.731** (±**0.030**)	**0.117** (±**0.004**)
DeepHit	0.650 (±0.043)	0.122 (±0.005)	0.708 (±0.037)	0.124 (±0.003)
PLANN	**0.667** (±**0.02**)	**0.121** (±**0.006**)	0.668 (±0.037)	0.117 (±0.005)
PDisc	0.643 (±0.015)	0.254 (±0.071)	0.685 (±0.026)	0.250 (±0.11)
PKM	0.648 (±0.027)	0.236 (±0.024)	0.659 (±0.019)	0.223 (±0.078)
POpt	0.645 (±0.023)	0.217 (±0.056)	0.644 (±0.015)	0.221 (±0.070)

Notes. The value in brackets is the standard deviation across the 5 folds. The highest C-statistic values and lowest IBS values are in bold.



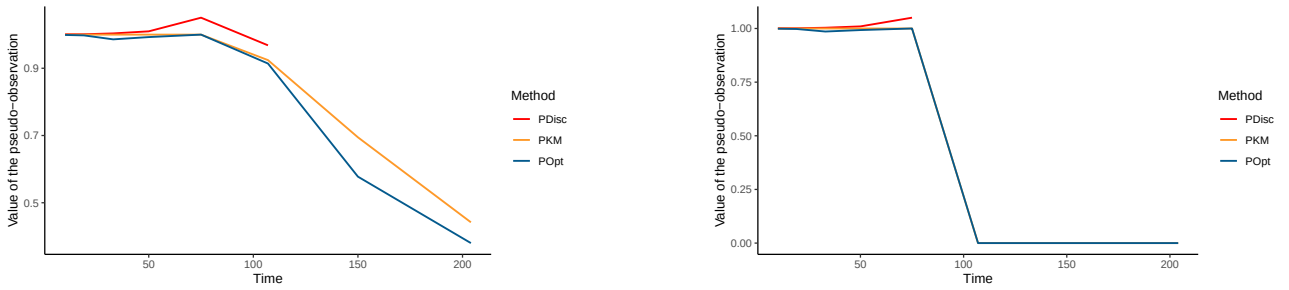
(a) All Variables

(b) Clinical Variables

Figure 4.8: Brier Score over time for FNNs (solide lines) and benchmark models (dotted lines), either trained on all variables of METABRIC data (Figure 4.8a), or restricted to clinical variables (Figure 4.8b).

4.6.3 . Results for the LCE cohort

FNN-based models and benchmark models were also applied to the LCE lung cancer cohort. All comparison measures were computed on the test set at 2 years.



(a) Patient with a censored survival time

(b) Patient with an observed event.

Figure 4.9: Values of the pseudo-observations computed for two patients of the LCE cohort. On Figure 4.9a, the survival time of the patient is censored after 98 months. On Figure 4.9b, the patient died at time 95.

With the LCE cohort, all the models performed better when genomic data was included. As we can see in Figure 4.10a, BS increases over time, regardless of the model considered. Considering the case when all variables were included, the performances of RSF, CoxCC, CoxTime and DeepHit were very close in terms of C-statistic, while pseudo-observation-based models obtained comparatively lower performances. The highest C-statistic was obtained with LASSO, and CoxCC had the lowest IBS. The VAE-based model improved the performances of all the models in terms of C-statistic.

Table 4.8: Average C-statistic and IBS at 2 years for LCE data.

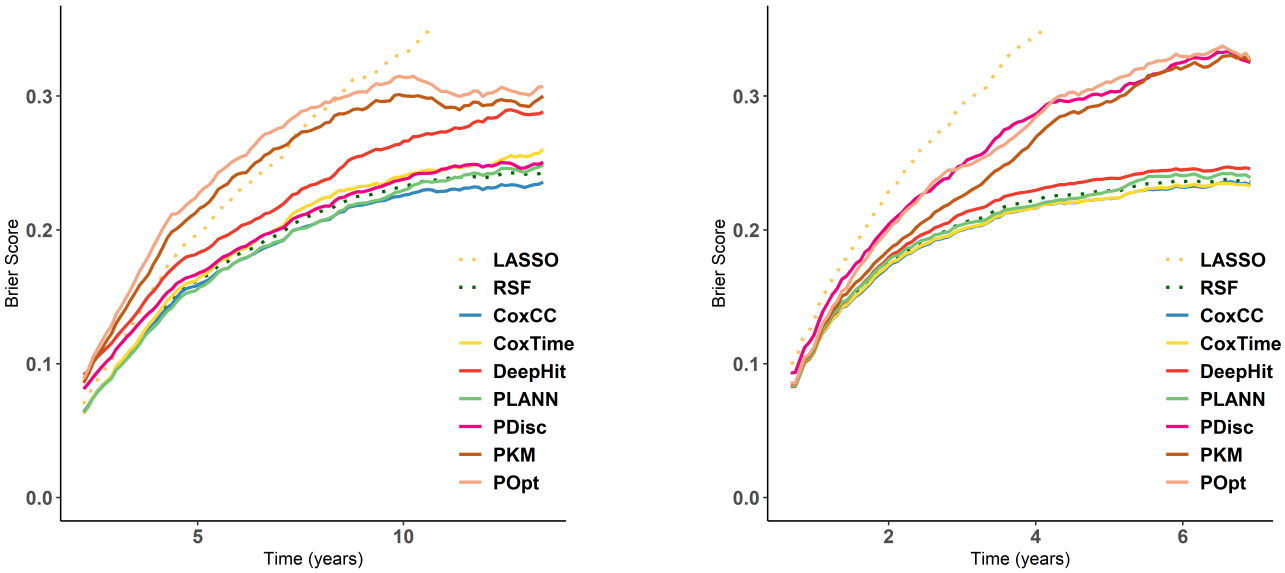
	All		Clinical	
Model	***C***	***IBS***	***C***	***IBS***
LASSO	**0.731** (±**0.025**)	0.135 (±0.004)	0.566 (±0.038)	0.168 (±0.007)
RSF	0.722 (±0.019)	0.122 (±0.003)	0.651 (±0.023)	0.135 (±0.003)
CoxCC	0.721 (±0.021)	**0.119** (±**0.003**)	**0.654** (±**0.023**)	**0.133** (±**0.005**)
CoxTime	0.711 (±0.02)	0.122 (±0.05)	0.652 (±0.022)	0.134 (±0.005)
DeepHit	0.712 (±0.015)	0.153 (±0.013)	**0.654** (±**0.021**)	0.137 (±0.004)
PLANN	0.683 (±0.006)	0.132 (±0.003)	0.655 (±0.018)	0.137 (±0.004)
PDisc	0.684 (±0.039)	0.146 (±0.008)	0.587 (±0.057)	0.154 (±0.016)
PKM	0.690 (±0.02)	0.155 (±0.013)	0.643 (±0.018)	0.141 (±0.006)
POpt	0.683 (±0.04)	0.158 (±0.01)	0.631 (±0.032)	0.149 (±0.015)

Notes. The value in brackets is the standard deviation across the 5 folds. The highest C-statistic values and lowest IBS values are in bold.

Table 4.9: Average C-statistic and IBS at 2 years for LCE data.

	AE		VAE	
Model	***C***	***IBS***	***C***	***IBS***
CoxCC	0.721 (±0.065)	**0.138** (±**0.003**)	0.772 ±0.056)	0.139 (±0.006)
CoxTime	0.662 (±0.112)	0.141 (±0.003)	**0.807** (±**0.033**)	0.138 (±0.008)
DeepHit	0.673 (±0.076)	0.147 (±0.012)	0.712 (±0.143)	0.142 (±0.005)
PLANN	0.723 (±0.017)	0.125 (±0.006)	0.740 (±0.028)	**0.119** (±**0.006**)
PDisc	0.685 (±0.111))	0.230 (±0.002)	0.738 (±0.148)	0.230 (±0.0062)
PKM	**0.764** (±**0.046**)	0.224 (±0.0083)	0.767 (±0.036)	0.235 (±0.0099)
POpt	0.659 (±0.041)	0.229 (±0.005)	0.719 (±0.114)	0.268 (±0.071)

Notes. The value in brackets is the standard deviation across the 5 folds. The highest C-statistics and lowest IBS are in bold.



(a) All Variables

(b) Clinical Variables

Figure 4.10: Brier Score over time for FNNs (solid lines) and benchmark models (dotted lines), either trained on all variables of LCE data (Figure 4.10a) or restricted to clinical variables (Figure 4.10b).

4.7 . Conclusion and Discussion

In this chapter, we compared different models based on neural networks to perform survival analysis. These models are not constrained by the proportionality assumption of the CoxPH model and are based on specific loss functions. In addition, we focused on the use of pseudo-observations to deal with censored observations and explored different ways of computing them, either in a discrete time, with a $[0, 1]$ constraint, or from a Bayesian point of view. The constraint has the advantage to see the pseudo-observation as an individual's survival probability at a given time point. Two benchmark models were also implemented. The models were compared on 3 different types of synthetic data and 2 cancer cohorts.

As a benchmark, we simulated data that respected the proportionality assumption using a CoxPH model with a Weibull risk function. As expected, FNN-based methods did not outperform the penalized CoxPH model in this context.
We also simulated data with an AFT model and a log-normal risk function. FNN-based methods did not outperform the LASSO benchmark model, which was able to identify the relevant prognostic variables.
As artificial neural networks can model complex relationships between the variables and event times, we simulated data with interactions and non-linearity using a random function generator based on an AFT model with different levels of censoring. Many of the proposed neural network models had overall somewhat similar performances. With 20% censoring and $n = 1,000$, all the neural networks performed better than the linear model LASSO and fitted to the complex interactions existing in the simulated data. There was a drop in performance for all the models at higher censorship. RSF obtained competitive results with the best performing FNNs, but always with a higher computational cost.
Regarding the FNNs only, the FNNs defined in a continuous time framework performed the best, with closed results between CoxCC and CoxTime. For the discrete time framework, PLANN obtained competitive results. The pseudo-observation based FNNs were the ones that got the worst results.

The machine learning methods were also applied to 2 real patient cohorts: the METABRIC study and the LCE. We obtained good performances in the example data of early breast cancer. Different neural network models obtained comparable 2-year discrimination performances on the lung cancer data, but with slightly lower values than RSF and LASSO. The real application results also showed that the VAE-based method could improve the performance of predictions by synthesizing the information contained in the transcriptomic data. The model that obtained the highest C-index and lowest IBS for both real patients cohorts was CoxTime.

In future research, we could explore the value of these neural network methods in the context of higher-dimensional data and thus further raise the number of variables, but this may need larger sample sizes. We could also challenge the results of our simulation study by simulating data differently, with other sources of complexity.

5 - Uncertainty Measures using Ensemble of Neural Networks

Machine learning models are increasingly used in the medical field and in survival or time-to-event analysis. Among the models applied by biostatisticians are FNNs (Faraggi et al., 1995); (Biganzoli et al., 1998), a type of neural network that can learn non-linear and complex relationships in patients' data in order to output survival probabilities. However, only a few neural network models for time-to-event data have addressed the assessment of prediction uncertainty, which is crucial in healthcare applications where practitioners could benefit from this uncertainty information. For example, an automated cancer detection system built on a FNN that is confronted with a patient's data that lies outside its data distribution might return unreasonable predictions and consequently bias the expert. In this case, if an uncertainty measure was associated with unreasonable predictions, the patients for whom the model is highly uncertain in its prediction could be treated as a particular case explicitly or even passed to a human to make a decision.

Deep learning models are confronted with multiple levels of uncertainty. One source of uncertainty comes from the noise in the data, which stems from technical issues involved with data collection and measurement. Another source comes from adversarial examples or dataset shifts that force deep learning models to extrapolate predictions far away from the observed data. These two uncertainties are called aleatoric uncertainty and are caused by irreducible structural relationships within the data. Another level of uncertainty exists in the model's parameters, hyperparameters, or even the underlying structure of the chosen model. This type of model uncertainty is referred to as epistemic uncertainty. Aleatoric and epistemic uncertainty can be combined to induce predictive uncertainty, i.e., the model's confidence in its prediction. Predictive uncertainty is often obtained by generating multiple functions from the model and corrupting them with noise.

Uncertainty sources can be deconstructed according to a specific model. Let us define the learning program as $Y = \phi(X)$ given data $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ for some unknown function $\phi(X)$. The output variable Y_i is predicted using the function $\phi(X_i)$ with the set of variables X_i. $\phi(X_i)$ is estimated by $\phi(X_i; \theta)$, the output of a neural network estimated from the data. We can write:

$$Y_i = \phi(X_i; \theta) + u_i. \tag{5.1}$$

u_i defines the approximation error with $u_i = \epsilon_i + \phi(X_i) - \phi(X_i; \theta)$ due to replacing $\phi(X_i)$ by $\phi(X_i; \theta)$. ϵ denotes the noise in the output variable that cannot be explained by the variables X and satisfies the conditional independence assumption $E[\epsilon_i | X_i]$. We can rewrite the empirical model as follows:

$$Y_i = \phi(X_i; \hat{\theta}) + e_i. \tag{5.2}$$

Additionally, the error term can be decomposed as:

$$\epsilon_i = \phi(X_i; \hat{\theta}) - \phi(X_i; \theta) + \phi(X_i; \theta) - \phi(X_i) + e_i. \tag{5.3}$$

$\phi(X_i; \hat{\theta}) - \phi(X_i; \theta)$ is the estimation error, $\phi(X_i; \theta) - \phi(X_i)$ defines the bias effect and e_i is the aleatoric error. The conditional variance of the output variable given the set of variables X, denoted

as σ_ϵ^2 satisfies that:

$$\sigma_\epsilon^2 = \sigma_{\hat{\theta}}^2(X_i) + \sigma_e^2, \tag{5.4}$$

with $\sigma_{\hat{\theta}}^2(X_i)$ the epistemic uncertainty and σ_e^2 the variance due to the aleatoric error. The uncertainty around the predictions is given by σ_ϵ^2.

The estimators used are FNNs (Roblin et al., 2020), which model time-to-event outcomes based on patient characteristics such as clinical, pathological, demographic, and molecular data. However, these estimators only return point predictions and do not provide a direct measure of uncertainty. Indeed, the weights of the FNN characterizing the predictions are usually fixed, implying that the output is deterministic. On the contrary, bayesian neural networks allow the networks' weights to be defined by a given probability distribution and can capture the posterior distribution of the output. In a frequentist framework, one way to overcome this issue is to use model ensembling, which aims to construct a predictive model by combining multiple learners. Several researchers have proposed different ways of obtaining ensembles of neural networks. One method consists in manipulating the training samples using a bootstrap strategy (Efron, 1979). Randomness can also be injected into the algorithm at different levels. With Deep Ensembles (Lakshminarayanan et al., 2016), multiple neural networks are trained by randomly initializing the neural network's weights. MCDrop (Gal et al., 2016) is a combination of models obtained by randomly applying dropout at test time, and FBMask (Mancini et al., 2020) is an extension of MCDrop that uses a fixed mask of units being dropped out. Different models' predictions are then ensembled to associate point predictions with uncertainty estimation, thus improving the accuracy of existing model predictions from individual neural networks. More importantly, the ensembling allows us to derive interval forecasts from approximate predictive distributions and thus assess the uncertainty about these ensembles of neural network models' expected survival predictions.

5.1 . Ensemble with neural networks

Ensemble methods amount to constructing several base models and combining them to produce an optimal predictive model. By constructing an ensemble, the searching space of the hypothesis to identify a local optimum in the space is more exhaustive and may provide a better approximation to the true unknown function than any individual model. In our context, an ensemble of M predictors is given by :

$$\bar{\phi}(X_i) = \frac{1}{M}\sum_{m=1}^{M} \phi_m(X_i; \hat{\theta}^{(m)}), \text{ for } i = 1, \ldots, n, \tag{5.5}$$

where $\phi_m(X_i; \hat{\theta}^m)$ denotes a set of M different prediction models based on deep neural network models; $\hat{\theta}^{(m)}$ denotes the estimates of the FNN model parameters and hyperparameters.

Many methods for constructing an ensemble have been developed, mainly based on decision trees like RSF. There are two classes of ensembles to induce diversity in the models. Some are based on bootstrap aggregating (bagging), where each model is trained on a different subset of the original training set. However, if the underlying base learner has multiple local optima, as is the case typically with FNNs, the bootstrap can sometimes hurt performances since the base learner is not trained using all data points. The second solution is based on randomization approaches. Indeed, if the model is applied to the same data each time but with different parameters, the resulting predictions are still different.

5.1.1 . Bootstrap

To build uncertainty measures based on confidence intervals, we first implement the naive bootstrap (Boot) approach (Efron, 1979). It is a commonly used method despite its computational requirements. M subsets of the training set are generated by random sampling with replacement from the training set. Each observation of the training set has the same probability of being extracted. Bootstrapped training sets are drawn with the same size as the original training set, and thus several training examples appear multiple times in the set. The neural network model is trained for each of these M subsets, allowing us to estimate M survival probabilities at time t for a given patient i of the test set, noted as $\hat{S}_{i(boot)}(t) = \{\hat{S}_{i(1)}(t), \ldots, \hat{S}_{i(M)}(t)\}$. Each bootstrap replicate of the original training set contains on average 63.2% of the initial training set, which can be a drawback as FNNs trained with less data are more biased.

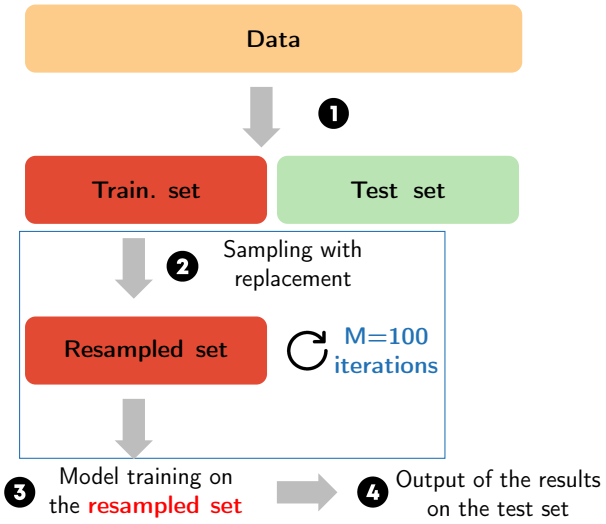


Figure 5.1: Ensemble of FNNs constructed on Boostraping. In the first step **(1)**, we split the data in a training set and a test set. Then **(2)**, the training set is resampled. The hyperparameter search is performed on the resampled set. The final model built using the selected hyperparameters is trained on the resampled set **(3)** and the results are outputted on the test set **(4)**. The resampling **(2)** is repeated M times.

We also implement the bootstrap-t (Boot-t) method (Tibshirani, 1997), a nested version of the bootstrap algorithm. First, we train the model on the entire training set, and then we compute the survival probability at time t $\hat{S}_i(t)$ for a given patient i of the test set. Next, the training set is randomly sampled M times, and we obtain the survival probabilities of patient i using these outer samples: $\hat{S}_{i(outer\ boot)}(t) = \{\hat{S}_{i(1)}(t), \ldots, \hat{S}_{i(M)}(t)\}$. For each $m \in \{1, \ldots, M\}$, the bootstrap sample is then resampled K times $(K = 50)$ and each inner sample set is used to fit the model. The survival probabilities for patient i computed with the inner samples for outer sample m are denoted by $\hat{S}_{i(m,inner\ boot)}(t) = \{\hat{S}_{i(m,1)}(t), \ldots, \hat{S}_{i(m,K)}(t)\}$. Using the K inner sample replicates $\hat{S}_{i(m,inner\ boot)}(t)$, we obtain the standard error of each $\hat{S}_{i(m)}(t)$. Thus we can compute the t-static as $t^{stud} = \frac{\hat{S}_{i(m)}(t) - \hat{S}_i(t)}{SE(\hat{S}_{i(m)}(t))}$, with SE the standard error, and build studentized confidence intervals. The major drawback of this algorithm is that it is computationally more expensive than Boot.

5.1.2 . DeepEnsemble

We compare Deep Ensembles (DeepEns) (Lakshminarayanan et al., 2016) to the Boot method. Here, unlike with Boot, the entire training set is used for training each model. This can represent

an advantage of this method, since FNNs achieve better performances when they are trained with more data.

In this approach, the randomness is introduced from within the algorithm as the parameters of the network are randomly set. Thus, an ensemble of M deterministic FNNs is trained by varying the random seed of the previously tuned set of hyperparameters. Indeed, simply changing the random seed is enough for FNNs to vary in their individual predicted probabilities. Another source of randomness is added by randomly shuffling all the data points of the training set, applying a permutation at each initialization of the model. For each iteration, the model is trained on a random set of parameters, and output a probability on the test set. The model outputs a different probability per patient of the test set for each different initialization. Let M denote the number of FNNs in the ensemble and $\{\theta_m\}_{m=1}^{M}$ denote the parameters of the ensemble. The method outputs a combination of predictions over M models to obtain a predictive distribution.

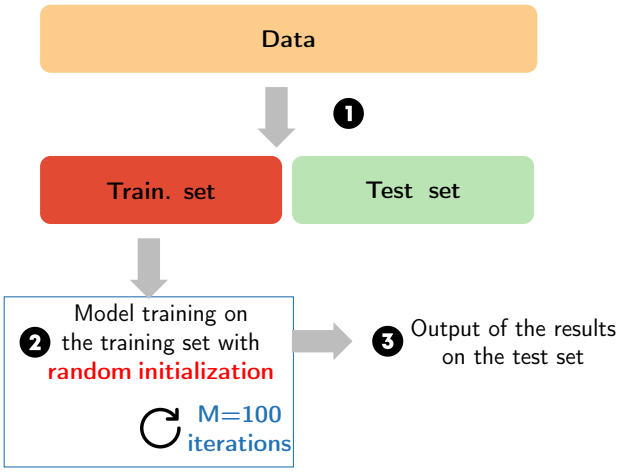


Figure 5.2: Deep Ensemble of FNNs. In the first step **(1)**, we split the data in a training set and a test set. Then **(2)**, the hyperparameter search is performed on the training set. The final model built using the selected hyperparameters is trained on the training set with a random seed **(2)** and the results are outputted on the test set **(4)**.

The authors also confront the model with adversial examples to smooth the predictive distributions. These adversial examples are points that are close to the original training examples, but are missclasiffied by the neural network. They are built using the fast gradient sign method (Goodfellow et al., 2014). The adversial perturbation creates a new training example by adding a perturbation along a direction which is likely to increase the loss. They are used to train the model on out-of-distribution examples and to increase the robustness of the model to miss-specification.

5.1.3 . Monte-Carlo Dropout

We also use the MCDrop method (Gal et al., 2016). Typically, dropout is a technique used to prevent overfitting, which occurs when FNN fits exactly the training dataset. FNN ends up learning noise and generalizes badly to hold-out samples, leading to poor performances. With dropout, units of the neural network are randomly excluded before each layer during training, with a chosen probability p. It forces the hidden nodes of the FNN not to co-adapt with the neighboring nodes.

Let us define a FNN with L hidden layers. For a given layer $l \in \{1, \ldots, L\}$, h_l is the vector of outputs from layer l and h_{zl} denotes the elements of the vector h_l for a given node $z = 1, \ldots, Z_l$. Let $r_l = [r_{1l}, \ldots, r_{zl}] \in \mathbf{R}^{Z_n}$ be a vector of realisations from the Bernoulli distribution with probability p. r_{zl} is a random variable with a probability p of being 1 (and $1-p$ of being 0). With dropout,

h_{zl} is transformed into $\bar{h}_{zl}$ by multiplying r_{zl} with the outputs of layer l, h_{zl}:

$$\bar{h}_{zl} = r_{zl}.h_{zl}. \tag{5.6}$$

Then, $\bar{h}_{zl}$ is used as input of the next layer, h_{zl+1}. Thus, a unit is dropped (i.e., its value is set to zero) for a given input if its corresponding binary variable r_{zl} takes value 0. With this method, an extra hyperparameter, the probability of retaining a unit p, has to be tuned. With $p = 1$, no dropout is applied, while a low p implies a high dropout level.

The application of dropout in a FNN with arbitrary depth and nonlinearities can be interpreted as a Bayesian approximation of a Gaussian process. As a FNN does not directly measure model uncertainty, this assimilation to a bayesian probabilistic model enables it to induce uncertainty estimates. With this new theoretical framework, Gal et al. (2016) introduced MCDrop, a method to estimate predictive uncertainty by simply applying dropout at test time. More precisely, M stochastic forward passes through the network are performed during test time. Each forward pass is multiplied by a random variable to generate a random sample of the approximated posterior distribution. In practice, this predictive distribution can be approximated using Monte Carlo methods. By sampling M sets of vectors of realisations from the Bernoulli distribution $\{r_l^{(m)}\}_{m=1}^{M}$ corresponding to M forward passes through the network, we obtain a different neural network at each iteration. Thus dropout may also be interpreted as an ensemble model combination.

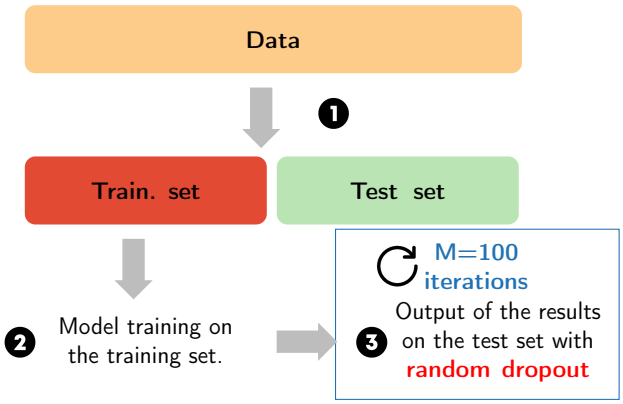


Figure 5.3: FNNs with Monte-Carlo Dropout. In the first step **(1)**, we split the data in a training set and a test set. Then **(2)**, the hyperparameter search is performed on the training set. The final model built using the selected hyperparameters is trained on the training set **(2)**. Then, some units of the FNN are randomly dropped before the results are outputted on the test set **(3)**. This drop out is randomly applied M times during the test phase.

We implement the MCDrop by randomly activating dropout during training and test time. After M iterations on the test set, we obtain an ensemble of M survival predictions at time t for all test set patients.

As noted by Levasseur et al. (2017), the construction of prediction intervals with correct empirical coverage probabilities using the MCDrop is highly dependent on the adequate tuning of the dropout rate p.

5.1.4 . Fixed Bernoulli Mask

An alternative to MCDrop is the FBMask (Mancini et al., 2020). Based on the extremely randomized trees (Geurts et al., 2006), it extends the MCDrop approach outside the bayesian framework. Mancini et al. (2020) define r_l as a Bernoulli mask and introduce the fixed Bernoulli

mask, noted as $\bar{r}_l$, a mask that is kept constant during training and testing. First the neural network architecture is defined. M fixed Bernoulli masks are defined where M sets of vectors are sampled from the Bernoulli distribution before training. Then, each FBMask is applied to the neural network architecture for each neural network model during both the training and test phases, with random initialization of the weights. We obtain M predictions $\phi_1(X_i; \hat{\theta}^{(1)}), \ldots, \phi_M(X_i; \hat{\theta}^{(M)})$ that we store to build prediction intervals. With FBMask, we obtain an ensemble of M neural networks with randomized weights and structures and no data resampling. The method does not rely on the assumption of independent and identically distributed observations.

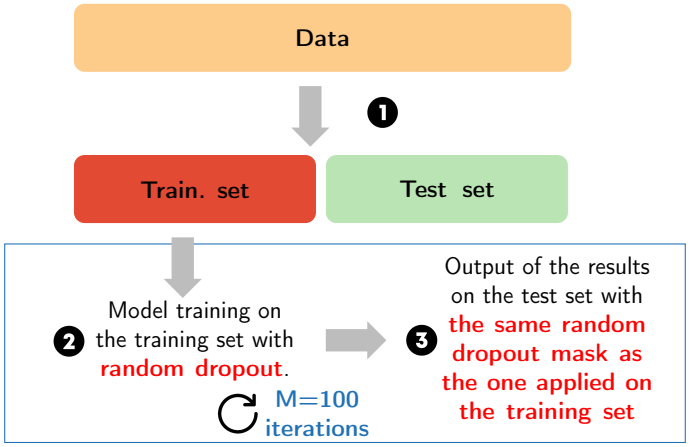


Figure 5.4: FNNs with FBMask. In the first step **(1)**, we split the data in a training set and a test set. Then **(2)**, the hyperparameter search is performed on the training set. Then, some units of the FNN are randomly dropped out: we call this new FNN architecture a fixed Bernoulli Mask. The final model built using the selected hyperparameters with the fixed Bernoulli mask is trained on the training set **(2)**, and the results are outputted on the test set **(3)** keeping the same architecture including the mask. The sampling of the mask is repeated M times.

5.2 . Uncertainty measure based on ensemble of predictions

5.2.1 . Percentile Confidence Intervals

It is not possible to evaluate the accuracy of predictions if we only output point estimates, whereas it is if we associate intervals with these forecasts. The percentile method is used to obtain confidence intervals at level $1-\theta$ for $\hat{S}_i(t)$ based on the distribution of $\hat{S}_{i(all)}(t)$:

$$IC_{1-\theta}\left(\hat{S}_i(t)\right) = \left[q_{\frac{\theta}{2}}\left(\hat{S}_{i(all)}(t)\right); q_{1-\frac{\theta}{2}}\left(\hat{S}_{i(all)}(t)\right)\right]. \tag{5.7}$$

Here, $\theta = 5\%$ and the 2.5^{th} and 97.5^{th} percentiles are computed using the empirical distribution of the M survival probabilities.

This interval represents a range of values that is likely to contain a future individual observation from the values of the input predictors that are taken into account in the model. There are 95% chances that the interval will include an individual's expected survival probability with the same specific levels of input predictor data. The interval depends on the quality of the model in this particular data region we are trying to study. It is always associated with a confidence level representing a degree of uncertainty. The more uncertainty associated with the forecast, the wider the intervals are.

For the Boot-t method, the confidence intervals are studentized. It is supposed to improve the coverage of percentile bootstrap confidence intervals, especially for smaller sample sizes. It is

computed as:

$$IC_{1-\theta}\left(\hat{S}_i(t)\right) = \left[\hat{S}_i(t) - \hat{s}q_{1-\frac{\theta}{2}}(t^{stud}), \hat{S}_i(t) - \hat{s}q_{\frac{\theta}{2}}(t^{stud})\right] \tag{5.8}$$

$\hat{s}$ is the standard error of $\hat{S}_i(t)$. It is computed with the M Boot estimates: $\hat{S}_{i(outer\ boot)}(t) = \{\hat{S}_{i(1)}(t), \ldots, \hat{S}_{i(M)}(t)\}$. The limit of this method is that it can produce estimates outside the range of plausible values.

5.2.2 . Coverage Rate

Evaluating the quality of predictive uncertainties is challenging as the ground truth uncertainty estimates are usually not available.

In order to evaluate the quality of the confidence intervals, the empirical coverage rate is calculated:

$$CR = \frac{1}{N}\sum_{i=1}^{N} I\left(q_{\frac{\theta}{2}}\left(\hat{S}_{i(all)}(t)\right) \leq S_i(t) \leq q_{1-\frac{\theta}{2}}\left(\hat{S}_{i(all)}(t)\right)\right) \tag{5.9}$$

The coverage rate is compared to the nominal value of 95%. For the Boot-t method, we compute the coverage rate using the studentized confidence intervals. The purpose is to build confidence intervals that have a coverage rate close to the chosen nominal value associated with a reasonable expected length.

All learning algorithms are run several times ($M = 100$ times) to build an ensemble of neural network models. Each method outputs M survival probabilities for individual i for a given time t. We then have: $\hat{S}(t)_{i(all)} = \{\hat{S}_{i(1)}(t), \ldots, \hat{S}_{i(M)}(t)\}$.

It is expected that the larger the sample size n, the closer the coverage probability of the prediction intervals to the nominal level of 95%.

5.3 . Simulation study

5.3.1 . Data Generation

Survival times are simulated according to the CoxPH model, with a log-logistic basis risk distribution (Lee et al., 1997). It enables to model non-monotonic risk rates. Based on the inverse cumulative distribution method (Bender et al., 2005), survival times are related to variables as follows:

$$T = \frac{1}{a}\{\exp[-\log(u)\exp(\beta f(X))] - 1\}^{\left(\frac{1}{b}\right)}. \tag{5.10}$$

First, we simulate the variable U according to a uniform distribution $\mathcal{U}(0,1)$. Then, $p+1$ variables are drawn independently: $X_1, \ldots, X_p$ following a normal distribution and z_1 is generated according to a Bernoulli law $\mathcal{B}(0.5)$. X is a sub-sample of 3 variables: $X = (X_1, X_2, X_3)$. These 3 variables are linked to the survival time with $f(X) = \exp(X^T V X)$ and $V = 0.05 \times \begin{pmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{pmatrix}$. It introduces non-linearity and interactions. The remaining variables are noise variables and do not contribute to the survival times. Here, $\beta = 0.5$, $\rho = 0.95$, $a = 1.25$ and $b = 0.9$.
Censoring times are generated with an exponential distribution, chosen to obtain around 20% of censoring among the data set. Survival times longer than 15 years are censored.
100 simulation data sets are generated, composed of $4,000$ individuals each. Each data set is split

into a train set ($2,000$ individuals) and a test set ($2,000$ individuals). The parameters are estimated from the training data, and the survival prediction and confidence intervals are computed on the test set.

5.3.2 . Oracle Model

To obtain theoretical survival probabilities for each individual i at a given time t, the truth-value of the log-logistic basis risk function at time t is estimated:

$$h_0(t) = ab(t)^{b-1}(1 + (at)^b)^{-1} \tag{5.11}$$

Using the baseline hazard (Equation 5.11), the baseline cumulative hazard is obtained with the relation $H_0(t) = \int_0^t h_0(s)ds = \log\left(1 + (at)^b\right)$. Then the survival function can be retrieved through $H_0(t)$:

$$S(t|X) = \exp\left[-H_0(t)\exp\left(\beta^T f(X_i)\right)\right] \tag{5.12}$$

These probabilities are used to obtain an oracle C-statistic and the coverage rate of the estimated confidence intervals.

5.3.3 . Results

In this section, the results of the simulation study are presented. The C-statistic was estimated M times for each data set using a fixed survival time as the horizon ($t = 0.5$), and the mean of the M values was computed. Then the mean of the 100 C-indices was obtained using all simulation data sets and is reported in Table 5.1. No matter what ensemble method chosen, the mean C-indices numerically closest to those from the oracle model were obtained either with CoxCC or CoxTime. The standard deviation of the boostrap values was much larger for the DeepHit method as compared to CoxCC and CoxTime.

Table 5.1: Mean of the M C-statistics and 95% Coverage Rates obtained on the 100 simulation test sets using a fixed survival time as horizon ($t = 0.5$).

		Boot	Boot-t	DeepEns	MCDrop	BMask
Oracle	***C***	0.743 (±0.007)		0.743 (±0.007)	0.743 (±0.007)	0.743 (±0.007)
CoxCC	***C***	0.712 (±0.01)		0.723 (±0.012)	0.718 (±0.015)	0.723 (±0.014)
	CR	0.964 (±0.033)	0.975 (±0.027)	0.889 (±0.066)	0.894 (±0.048)	0.880 (±0.068)
CoxTime	***C***	0.710 (±0.015)		0.722 (±0.014)	0.720 (±0.009)	0.722 (±0.012)
	CR	0.970 (±0.042)	0.970 (±0.030)	0.895 (±0.072)	0.898 (±0.041)	0.896 (±0.07)
DeepHit	***C***	0.659 (±0.072)		0.707 (±0.022)	0.720 (±0.008)	0.711 (±0.0016)
	CR	0.728 (±0.151)	0.861 (±0.114)	0.706 (±0.081)	0.786 (±0.073)	0.703 (±0.076)

Notes. The value in brackets is the standard deviation across the 5 folds. Highest C-statistics and lowest IBS are in bold.

Table 5.1 also shows the coverage rate of confidence intervals for survival probabilities at a fixed time ($t = 0.5$) for all patients in the test set. More precisely, this is the mean value of the coverage rates on all simulation test sets, and the value in brackets corresponds to the standard deviation. The CoxCC model and the Boot method achieved the closest value to the nominal value of 95%. On average, the Boot method yields 95% confidence intervals that contain the true survival probability 96.4% of the time, slightly above the pre-defined nominal value of 95%. The studentized correction increased the coverage rate, indicating that the method could be too conservative. DeepEns, MCDrop, and BMask had similar results with a good level of coverage, but on the anti-conservative side. A comparison of FNN models shows that DeepHit obtained lower average coverage rates for all the methods considered, and also much more variability across the M bootstrap resamples. Researchers (Kvamme et al., 2021) have indicated that DeepHit can reach good discrimination capacities at the cost of poorly calibrated survival estimates.

5.4 . Application on cancer cohorts

5.4.1 . Results

On the METABRIC cohort, we compared the results obtained using either all the input variables or only the clinical variables. In terms of discrimination (Table 5.2), the ensemble methods demonstrated higher mean C-indices for neural networks fitted to the clinical variables alone, as compared to those fitted to clinical plus molecular data, highlighting that the clinical variables contained already a large part of the predictive information for survival.

Table 5.2: Mean of the M C-statistics at 5 years on the METABRIC test set with all variables or clinical variables only.

	All				Clinical			
Model	**Boot**	**DeepEns**	**MCDrop**	**BMask**	**Boot**	**DeepEns**	**MCDrop**	**BMask**
CoxCC	0.622 (±0.027)	0.646 (±0.051)	0.660 (±0.02)	0.626 (±0.041)	0.660 (±0.032)	0.680 (±0.021)	0.642 (±0.02)	0.683 (±0.03)
CoxTime	0.619 (±0.026)	0.658 (±0.052)	0.656 (±0.017)	0.681 (±0.025)	0.657 (±0.033)	0.724 (±0.015)	0.723 (±0.014)	0.740 (±0.008)
DeepHit	0.632 (±0.036)	0.675 (±0.028)	0.670 (±0.02)	0.699 (±0.016)	0.623 (±0.038)	0.722 (±0.011)	0.733 (±0.006)	0.726 (±0.007)

Tables 5.3 reports the mean C-indices at 2 years using the LCE. In this cohort, adding the molecular data improved discrimination, regardless of the chosen neural network architecture or the ensemble technique. Thus, in this lung cancer data set, there is additional prognostic information in the molecular data which is captured by the neural network beyond the clinical variables.

5.4.2 . Confidence Intervals at the patient level

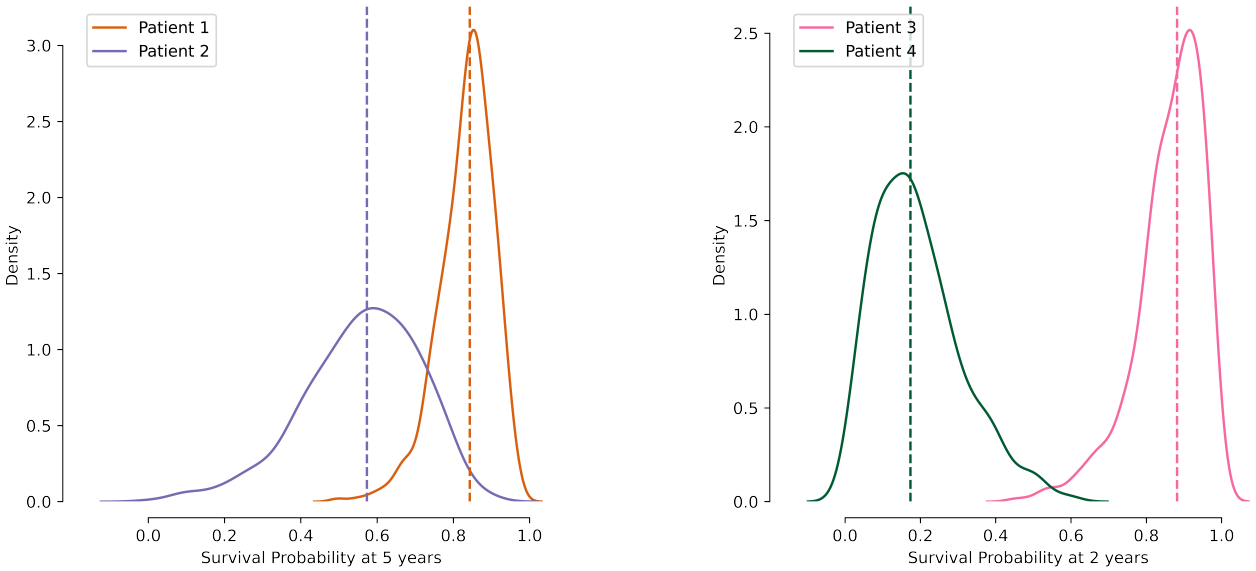
We further studied the point estimates of expected survival. We analyzed expected survival probabilities for 2 patients from the test set of the METABRIC data set, as if we were to advise a practitioner: Patient 1 has a survival time of 12 years and 6 months and his follow-up time is censored; Patient 2 has a survival time of 2 years with no censoring. Figure 5.5a represents the density of survival probabilities obtained with the MCDrop method applied on CoxTime. With this method, the mean survival probability at 5 years for patient 1 (orange line) was 0.834 with a confidence interval of $[0.667, 0.945]$. The second patient had a 0.560 mean survival probability at 5

Table 5.3: Mean of the M C-statistics at 2 years on the LCE test set with all variables or clinical variables only.

	All				Clinical			
Model	**Boot**	**DeepEns**	**MCDrop**	**BMask**	**Boot**	**DeepEns**	**MCDrop**	**BMask**
CoxCC	0.640 (±0.028)	0.697 (±0.016)	0.682 (±0.016)	0.664 (±0.012)	0.609 (±0.019)	0.644 (±0.004)	0.641 (±0.008)	0.643 (±0.004)
CoxTime	0.642 (±0.025)	0.671 (±0.018)	0.699 (±0.013)	0.678 (±0.016)	0.617 (±0.017)	0.636 (±0.011)	0.644 (±0.006)	0.641 (±0.005)
DeepHit	0.621 (±0.035)	0.680 (±0.026)	0.694 (±0.029)	0.680 (±0.017)	0.602 (±0.021)	0.649 (±0.009)	0.634 (±0.013)	0.640 (±0.007)

years, with a confidence interval of $[0.211,\ 0.804]$. The estimated mean survival probability was still relatively high for patient 2, even though the event already happened. However, the corresponding density is more dispersed and the confidence interval is wider than for patient 1. The model seems less confident in its survival predictions for the second patient.

Two patients were also sampled from the LCE test set: Patient 3 was followed for 5 years and 2 months before being unavailable for follow-up; Patient 4 has a survival time of 8 months without censoring. Applying the same method as previously, we obtained a mean predicted survival probability at 2 years of 0.948 with a confidence interval of $[0.848, 0.993]$ for patient 3 (pink line on Figure 5.5b). The estimated mean survival probability for patient 4 was 0.526 with a confidence interval of $[0.262,\ 0.774]$. Both densities are centered around their corresponding median prediction value in Figure 5.5b.



(a) METABRIC (b) LCE

Figure 5.5: Density estimate of expected survival probabilities at 5 years for two patients with breast cancer in Figure 5.5a and at 2 years for two patients with lung cancer in Figure 5.5b, using the MCDrop method with CoxTime. The vertical line represents the median of the M predictions.

5.5 . Conclusion and Discussion

We developed here confidence intervals for expected survival probabilities using ensemble of artificial neural networks. Our simulation study assessed the performance of different uncertainty measures by comparing various models' survival predictions to the oracle value. These methods were evaluated for two cohorts. We also described expected survival estimates for two patients of each data set, representing uncertainty at the patient level.

The neural network models CoxCC and CoxTime performed similarly. As the simulated data did not include time-dependent variables, there was no reason for CoxTime to perform better. CoxTime obtained the highest C-indices for the cohort studies. Boot obtained encouraging results for the simulation study, but was computationally more costly; unfortunately, the computational coast could be a drawback when training deep neural networks. The coverage rate above 95% indicates that Boot could be slightly too conservative. DeepEns, MCDrop, and BMask performed well in terms of empirical coverage. MCDrop was the second best method after Boot, although it was slightly anti-conservative in the simulations. We recommend using MCDrop for deep neural networks since it performed well both for the simulation study and the applications and is less computer-intensive than Boot.

In the METABRIC breast cancer cohort, the same results were obtained with the ensemble of M neural network models as in Chapter 3 with a single neural network. Indeed, the ensemble of neural networks had difficulty capturing additional prognostic information from the molecular data, suggesting the clinical variables alone are sufficient, while in the LCE cohort, the models led to substantially stronger discrimination values when adding molecular data to the clinical variables. For the real patient cohorts, we could further interrogate the quality uncertainty estimates by evaluating the models at various levels of confidence.

Based on the results obtained in Chapter 3, we could build an ensemble of combined VAE model to help capture the features of the transcriptomic data. We could even implement a VAE-based uncertainty measure based on a sampling strategy in the latent space z. Indeed, in a VAE, if we sample multiple points from the latent distribution $p(z|X)$, we can create an ensemble of M neural networks. Then, M survival predictions per patient from the test set can be outputted using the resulting M combined VAE-FNNs. However, if we build our VAE based uncertainty measure on the combined VAE structure presented in 3, as the VAE part is trained using only the genomic data, it would measure the uncertainty portion of the survival prediction that is dependent on this type of data. By only quantifying uncertainty related to the molecular data, the method would underestimate the actual uncertainty. As seen with the METABRIC cohort, in many biomedical datasets the clinical variables contain a significant portion of the predictive information for survival that would not be taken into account with this method.

Other sources of uncertainty have the potential to be analyzed in future work. Applying data set shift to a model using adversarial training (Lakshminarayanan et al., 2016), for example, can reveal variations in either the input or the output distributions, especially between the training and test sets. We could also investigate the variability of the confidence intervals across patient sub-

groups for each ensemble method.

The different ensembling methods investigated here could also be applied to other types of neural network architectures such as Convolutional Neural Network or Recurrent Neural Network (RNN). For instance, Dusenberry et al. (2020) built an ensemble of RNNs using Deep Ensembles.

6 - Personalized Treatment Recommendations

6.1 . Context

Stratified medicine consists in using biomarkers to differentiate subpopulations of patients with a similar disease profile or who will respond to a targeted treatment. Subgroups of patients are determined by searching for potential interactions between available treatment options and prognostic variables. In cancer research, for instance, practitioners want to identify treatments expected to be beneficial for a subset of patients whose cancers display specific molecular or cellular characteristics. An increasing interest in medicine is devoted to identifying patients more likely to benefit from the treatment and designing individualized optimal treatment options: this is precision medicine.

The treatment benefit is estimated in the context of a RCT and computed as an average across the study population. In a RCT, patients are randomly assigned to two groups of treatment: one who receives the experimental treatment, called the treatment group (for example, receiving trastuzumab added to chemotherapy), and the second group which does not receive it, called the control group (for example, receiving chemotherapy only). Both groups are otherwise comparable. Thus, any observed changes on average should result from the intervention. The RCT aims to control variables that are not under direct experimental control and determine whether a cause-effect exists between the intervention and the outcome. In secondary analyses, it amounts to identifying biomarkers-by-treatment interactions, also called treatment-effects modifiers or predictive biomarkers. A combination of multiple treatment-effect modifiers can be determined with RCTs, for example, a gene signature based on 8 genes for trastuzumab in early breast cancer (Pogue-Geile et al., 2013).

A general methodological framework for identifying treatment-effect modifiers is the CoxPH model, including the main effects of both the treatment and the biomarkers and the biomarker-by-treatment interactions. However, when the data is high-dimensional, for instance, when using molecular biomarkers, a model like CoxPH can be non-identifiable. A solution consists of selecting a sparse set of treatment-effect modifiers among many candidate variables (Ternès et al., 2017), for example, with the LASSO. One can also apply machine learning models as they were developed to handle such data types.

This work investigates the possibility of using neural network models to make individualized treatment recommendations based on patient characteristics. We observe the effect of the recommendations made on the survival rate of patients and set up a survival analysis with counterfactuals. We compare the results obtained with standard methods with those obtained with FNNs.

6.2 . Assessing treatment effects

Let us consider n patients from population $\mathcal{P}$. Let τ be the treatment variable of value $\{0, 1\}$ corresponding to the treatment options, where $\tau = 0$ stands for control and $\tau = 1$ is for treatment. Throughout, we assume that there is a unique version of each treatment, that each patient has a nonzero probability of assignment to each treatment level, and that the treatment level is independent of the potential outcomes, possibly conditional on observed variables. Such assumptions are met in randomized experiments.

6.2.1 . Potential outcome framework

To determine the treatment effect, that is, the effect of τ on the outcome variable Y, we use the potential outcome framework (Rubin, 1974). The potential outcome of treatment $\tau = 1$ or $\tau = 0$ for patient i is noted as $Y_i(1)$ or $Y_i(0)$. Since Y is the observed outcome, we have:

$$Y = \begin{cases} Y(1) & \text{if } \tau = 1 \\ Y(0) & \text{if } \tau = 0. \end{cases} \tag{6.1}$$

Thus, when $\tau = 1$, $Y(0)$ is not observed: it is the counterfactual. It follows that the causal effect of treatment τ for unit i is defined as:

$$\kappa = Y_i(1) - Y_i(0). \tag{6.2}$$

It corresponds to the treatment effect of changing patient i from control to treatment group. Here, the challenge is to infer quantities about unobserved counterfactuals using only the observed ones. The Average Treatment Effect (ATE) for treatment τ is:

$$\mathbf{E}^*(\kappa) = \mathbf{E}^*(Y_i(1) - Y_i(0)). \tag{6.3}$$

It measures the expected treatment effect τ for a random patient in a population. To estimate $\mathbf{E}^*(\kappa)$, the treatment τ needs to be unconfounded ; that is, it needs to verify $(Y(1), Y(0)) \perp\!\!\!\perp \tau$. A confounder is a variable that can impact τ and Y simultaneously. An exogenous procedure must be implemented for the treatment assignment to isolate confounders' effects, which is the case for RCTs. In this context, we can write:

$$\mathbf{E}(Y|\tau = 1) = \mathbf{E}^*(Y(1)|\tau = 1) = \mathbf{E}^*(Y(1)).$$

Similarly, we have:

$$\mathbf{E}(Y|A = 0) = \mathbf{E}^*(Y(0)).$$

It follows that:

$$\mathbf{E}^*(\kappa) = \mathbf{E}^*(Y(1) - Y(0)) = \mathbf{E}(Y|\tau = 1) - \mathbf{E}(Y|\tau = 0), \tag{6.4}$$

which corresponds to the ATE estimated as the difference between two conditional expectations involving only observables. Outside the framework of RCTs, causal inference must handle the existence of confounding factors.

While the main focus of clinical trials is to evaluate the average effect of a particular treatment, personalized medicine focuses on characterizing the variability in patients' response to treatment and estimating individualized treatment effects. The purpose is to identify the optimal treatment recommendations for a patient based on their current and previous characteristics. This amounts to determining predictive biomarkers in a RCT.

Survival models, like the CoxPH model, are used to explore and understand the relationships between patients' variables (e.g., clinical and genetic variables) and the effectiveness of various treatment options. However, they require extensive feature engineering or prior medical knowledge to model treatment interaction at an individual level. While nonlinear survival methods (FNN, RSF) can inherently model these high-level interaction terms, they have yet to be shown as effective

treatment recommender systems.

Here, we present two methods that were specifically designed to handle interactions, and that are used as a benchmark against the FNNs: the full biomarker-by-treatment interaction model subject to the LASSO penalty; Interaction Forest (IF), which is a specific type of random forest that models quantitative and qualitative interactions between variable pairs.

6.2.2 . LASSO penalty on a full biomarker-by-treatment interaction model

Rothwell (2005) put forward that the only reliable approach for assessing the predictiveness of biomarkers is to test their interaction with the treatment. To do so, the following model which contains the treatment, the biomarkers, and their interactions, can be implemented:

$$h(t|\tau, X) = h_0(t) \exp\left(\alpha\tau + \sum_{i=1}^{p} \beta_{i1} X_i + \sum_{i=1}^{p} \beta_{i2} X_i \tau\right), \tag{6.5}$$

with α, β_1 and β_2 the regression coefficients for the treatment τ (1 the experimental treatment and 0 the control arm), the standardized biomarkers $X_i, i = 1, \ldots, p$ and their interactions $X_i\tau$, respectively. The second sum is the component estimating the biomarker-dependent treatment effect.

In a high-dimensional setting, with $2p+1 \gg n$, the full biomarker-by-treatment interaction model is non identifiable. To overcome this issue, we introduce a LASSO penalisation $pen(\lambda, \beta_1, \beta_2)$. The penalized regression maximizes the penalized log-likelihood $\mathcal{L}(\beta_1, \beta_2) - pen(\lambda, \beta_1, \beta_2)$. The penalization term is written:

$$pen(\lambda, \beta_1, \beta_2) = \lambda \left(\sum_{i=1}^{p} |\beta_{i1}| + \sum_{i=1}^{p} |\beta_{i2}|\right) \tag{6.6}$$

Of note, the penalization is not applied to α. The main effects and the interactions are equally penalized (same shrinkage parameter λ). This approach lacks the hierarchy constraint: the main effect of a biomarker can be discarded ($\beta_{i1} = 0$) irrespective of whether the associated interaction β_{i2} is.

6.2.3 . Interaction Forests

IF (Hornung et al., 2022) is a type of random forest specifically designed to model quantitative and qualitative interaction effects in bivariate splits. Two variables are said to interact if the effect of one variable on the outcome depends on the value of the other variable. It is particularly interesting in this chapter, as the aim is to focus on biomarker-by-treatment interactions in the setting of a RCT. Modeling interactions can also improve the predictive performances of the model.

We focus on biomarker-by-treatment interactions in the setting of a RCT. Two variables are said to interact if the effect of one variable on the outcome depends on the value of the other variable. Here, two types of interactions are accounted for: the quantitative interactions and the qualitative interactions, as categorized by Peto (1982). For a quantitative interaction, the treatment effect is different depending on the biomarker value, but it remains in the same direction as the biomarker value (e.g., the treatment effect improves when the biomarker value increases). Conversely, when considering a qualitative interaction, the direction of the treatment effect depends on the biomarker value. As they are based on multivariable splitting, IFs are specifically developed to model these interactions. With RSF, the variables' interaction is considered if at least one of the two variables

has a strong marginal effect. It follows that the model does not take into account two variables that only have an effect when they both interact (Wright et al., 2016). Compared to RSFs, IFs enable the determination and the ordering of interaction effects.

As RSF, IF is an ensemble of decision trees. Each tree is learned using a random subset of the training set, and the remaining OOB samples are used to estimate the prediction error. The tree is built using multivariable splitting, unlike RSF (Ishwaran et al., 2007) which is built using univariable splitting. 6 types of splits are considered at each node, including univariate and bivariate splits. The candidate split set is composed of sampled splits from each split type. For each candidate split, a criterion is computed: the selected split is the one associated with the best value of the criterion. For survival outcomes, the criterion is the log-rank test statistic. Once all trees are grown, the predictions of all the trees are averaged. A new importance measure, the Effect Importance Measure, is introduced. It allows the ordering of pairs of variables according to the importance of their interaction effects on prediction. For datasets composed of more than 100 biomarkers, only $5,000$ variable pairs are pre-selected, and the tree depths are restricted when the dataset includes more than $1,000$ patients. Unordered categorical variables are converted into ordered variables.

It has been shown that the performance of random forests is relatively insensitive to changes in their hyperparameter values (Probst et al., 2019). Therefore, the default hyperparameter values are used in the implementation of IF. The number of variable pairs to sample for each split equals $\sqrt{p}/2$. The number of trees constituting each forest is set to $2,000$.

6.3 . Modeling treatment effect using FNNs

To model the treatment effect using a FNN, the categorical variable representing the patient's treatment assignment is added as one of the input variables of the FNN. This enables the neural network to learn the relationship between the treatment variable and the individual's variables.

Here, we introduce another way to handle the treatment variable. It consists in using Conditional Variational Autoencoder (CVAE). CVAE is an extension of VAE that includes a condition variable. We define this condition as the treatment variable and apply CVAE in the context of real patients' cohorts and a combined model with FNNs, as we did in Chapter 4 with AE and VAE.

Additionally, we develop a treatment recommendation rule specifically designed for FNNs and based on the work of Katzman et al. (2016).

6.3.1 . Combining FNNs with CVAE

CVAE is an extension of VAE that enables the sampling of data under a specific label and the generation of data with specific attributes. To do so, it simply adds a condition to the entire generative process. Thus, CVAE overcomes one of the limits of VAE. Indeed, VAE does not have any control over what kind of data it generates. It assumes that some random hidden process generates the data. The encoder part tries to learn the hidden representation of data X, $Q(z|X)$, and the decoder tries to learn $P(X|z)$. z is the hidden representation, and it is assumed that z is generated from some prior distribution $P(z)$. With CVAE, a condition is added to the VAE model: given an observation τ, z is drawn from the prior distribution $P(z|\tau)$. The encoder and decoder are conditioned on τ: $Q(z|X,\tau)$ and $P(X|z,\tau)$. Here, we define τ as the treatment variable.

6.3.2 . Treatment recommendation rule using a FNN

Using the survival probability outputted by the FNN model, we are able to formulate a stratified treatment recommendation based on the treatment recommendation system introduced by Katzman et al. (2016). With the FNN model DeepSurv, Katzman et al. (2016) provided a personalized recommendation method based on the personal risk of prescribing one treatment option over another. They demonstrated that these personalized treatment recommendations increased the median survival time of a set of patients. They placed their work in the context of a RCT: all patients are assigned to one of n treatment groups $\{0, 1, \ldots, n-1\}$.

Let us suppose that each individual has the same baseline hazard function $h_0(t)$. The estimation of the treatment recommendation system then proceeds in two stages. First, the patient i is assigned to treatment group k, and the network predicts the associated risk of being prescribed this given treatment k. Then, the patient i is assigned to treatment group l, and the associated risk is computed. Finally, the log of the hazards ratio of the two risks is calculated: it represents the personal risk of being prescribed a treatment over another. The recommender function, or rec_{kl}, is thus computed as:

$$\begin{aligned} rec_{kl}(X_i) &= \log\left(\frac{h(t|X_i, \tau = k)}{h(t|X_i, \tau = l)}\right) \\ &= \log\left(\frac{h_0(t).e^{\phi_k(X_i)}}{h_0(t).e^{\phi_l(X_i)}}\right) \\ rec_{kl}(X_i) &= \phi_k(X_i) - \phi_l(X_i). \end{aligned} \tag{6.7}$$

If $rec_{kl}(X_i) > 0$, then $\phi_k(X_i) > \phi_l(X_i)$, meaning that treatment k leads to a higher risk of death than treatment l. Hence, the patient i should be prescribed treatment l.

We extend this function and apply it to survival probabilities at a pre-specified time point. Furthermore, we add a threshold ϵ to the recommendation rule, as the gain in survival time must be clinically significant. We obtain the following treatment recommendation rule:

$$\tilde{rec}_{kl}(X_i) = S(t|X_i, \tau = k) - S(t|X_i, \tau = l) \text{ for a given time } t. \tag{6.8}$$

If $S(t|X_i, \tau = k) - S(t|X_i, \tau = l) > \epsilon$, then the patient i is prescribed the treatment k over the treatment l.

6.4 . Simulation study

We present a simulation study including treatment effect modifiers. We evaluate the ability of the FNN models to predict the treatment effect. Two models are used as benchmarks (LASSO and IF).

Two types of simulations are developed. First, we implement a full biomarker-by-treatment interaction model, as presented by Rothwell (2005), and include non-linear interactions. Secondly, we generate data using Friedman's random function generators, as in Chapter 4. A second generator is included in the AFT model to account for the treatment effect.

6.4.1 . Simulation with non-linearity

Based on this full biomarker-by-treatment interaction model, we can simulate data according to the following model:

$$h(t|\tau, X) = h_0(t) \exp(\alpha_\tau \tau + \sum_{i=1}^{p} \beta_{i1} f_i(X_i) + \sum_{i=1}^{p} \beta_{i2} f_i(X_i)\tau). \tag{6.9}$$

Here, compared to the original setting, we add nonlinearity with the choice of the function f, which enables to build complex interactions with the treatment. Haller et al. (2019) proposed to simulate such interactions. They considered the case of a single biomarker and simulated different biomarker-treatment interactions. Here, we set $\beta_{i1} = 0.4$, $f_i(X) = (2X - 1)^2$ and $\beta_{i2} = -0.5$. α was chosen as $\alpha = ln(0.75) \approx -0.288$. The biomarkers are generated from a uniform distribution: $X_i \sim \mathcal{U}_{[0,1]}$. Since we intend to simulate a RCT, a total of n patients per data set are randomly assigned $(1:1)$ to the experimental or control arm, with $P(\tau = 1) = P(\tau = 0) = \frac{1}{2}$.

Survival times are generated from a Weibull distribution. Thus, the risk can be written as follows:

$$h(t|\tau, X) = b^{-a} a t^{a-1} \exp\left(\alpha_\tau \tau + \sum_{i=1}^{p} \beta_{i1} f_i(X_i) + \sum_{i=1}^{p} \beta_{i2} f_i(X_i)\tau\right), \tag{6.10}$$

with a the shape parameter and b the scale parameter. For all scenarios, we generate exponential survival times (corresponding to a shape parameter $a = 1$) with a time-constant baseline hazard rate. We generate independent censoring.

6.4.2 . Simulation with nonlinearity and high-order interactions

In this section, we simulate survival data based on Friedman's random function generator, as in Chapter 4. (Henderson et al., 2020) adapted this flexible random function generator to the analyse of treatment effect by introducing a second generator interacting with the treatment variable. Thus, the AFT-Friedman model can be rewritten as:

$$\log T = m_1(X) + \tau \times m_2(X) + W \text{ with } W \sim \Gamma(2, 1), \tag{6.11}$$

where the vector of variables $X = (X_1, ..., X_{20})$ is generated with $X \sim \mathcal{N}(0, I)$, and $m_1(X)$ and $m_2(X)$ are defined as :

$$m_1(X) = \sum_{i=1}^{20} a_{1l} g_{1l}(R_{1l}) \text{ and } m_2(X) = \sum_{i=1}^{20} a_{2l} g_{2l}(R_{2l}). \tag{6.12}$$

The coefficients a_{1l} and a_{2l} are randomly generated from a uniform distribution, with $a_{1l} \sim \mathcal{U}_{[-1,1]}$, and $a_{2l} \sim \mathcal{U}_{[-0.2,0.3]}$. $R_{jl,j\in\{1,2\}}$ is a random subset of the input vector X of size n_{jl}. The size of each subset, n_{jl}, is itself random, with: $n_{jl} = min(\lfloor 2 + r \rfloor, 20)$ and $r \sim \mathcal{E}(1/2)$.

With Friedman's function generator, the input variables are associated with the survival time at different levels:

$$g_{jl}(R_{jl}) = \exp\{-\frac{1}{2}(R_{jl} - \mu_{jl})^T V_{jl}(R_{jl} - \mu_{jl})\}. \tag{6.13}$$

Each mean vector $\{\mu_{jl}\}_1^{20}$ is randomly generated with $\mu_l \sim \mathcal{N}(0, 1)$. The matrix of variance-covariance V_l is also randomly generated: $V_{jl} = U_{jl} D_{jl} U_{jl}^T$ with U_{jl} an orthonormal random matrix, $D_{jl} = diag\{d_{jl,1}, ..., d_{jl,n_{jl}}\}$ and $\sqrt{d_{jlk}} \sim \mathcal{U}(u, v)$. u, v are chosen according to the distribution of

the input variables X: there are the eigenvalue limits with $u = 0.1$ and $v = 2.0$. We finally obtain the survival times by applying the exponential: $\exp(\log(T))$. This simulation framework allowed us to generate random functions that have high-order interactions with the treatment variable τ and strong nonlinear effects. As for the first simulation setting, a total of n patients per data set were randomly assigned $(1:1)$ to the experimental or control arm, with $P(\tau = 1) = P(\tau = 0) = \frac{1}{2}$.

6.4.3 . Simulation scenarios

For the two simulation scenarios, we generate 100 training sets per scenario with different sample sizes ($n \in \{100, 1,000\}$), and 100 test set with $n = 1,000$. For both simulation studies, the treatment variable τ is drawn from a Bernoulli distribution with $p = 0.5$.

In the first simulation study, p variables ($p \in \{10, 100, 1,000\}$) are generated for the training set. We varie the number of non-zero main effects and true biomarker-by-treatment interactions. Indeed, among the p variables, only q biomarkers are really prognostic and interact with the treatment. As in Chapter 4, for $p = 10, 100$ and $1,000$ biomarkers, $q = 2(20\%)$, $q = 10(10\%)$, and $q = 20(5\%)$. Censoring times are generated independently from the survival times via an exponential distribution $\epsilon(\frac{1}{\gamma E(T)})$, where $\gamma > 0$ is a constant to be adjusted for the rate of censoring. We consider a large rate of censoring of 50% by taking $\gamma = 1.2$. In the second simulation study, we evaluate performance differences between the models according to the level of independent censoring in the data. We compare two censoring rates: 20% and 50%.

6.4.4 . Measures

To further analyze treatment recommendations, we compute the difference between the predicted survival probability at a given time point for a patient from the test set who received the treatment and the predicted survival probability for the same patient who did not receive the treatment. The difference is averaged over all patients of the test set and noted as **Mean Diff**. If the individual difference is superior to a certain threshold ϵ (here, $\epsilon = 0.02$), the treatment is recommended to the patient. The proportion of patients for whom we recommend the treatment in the test set is also computed (**Prop. Benef**). Here, the value ϵ is chosen as an example. The threshold value should be chosen as a value above which treating patients is worthwhile. It should be clinically meaningful and take into account costs and side effects, as outlined by (Efthimiou et al., 2022).

As in Chapter 4, the C-statistic and IBS are also measured to assess the performances of the models in terms of discrimination and calibration.

6.4.5 . Results for CoxPH-Weibull data

Table 6.1 reports the results for the first simulation setting based on the full biomarker-by-treatment interaction model. The results were computed at a fixed timepoint (the average median time of all the test sets) and averaged over 100 simulation sets.

Overall, benchmark models obtained higher C-statistics and lower IBS. IFs and FNNs performed better when they were trained on more data. Indeed, both C-statistics and IBS were better with $n = 1,000$ compared to $n = 100$. For instance, CoxCC obtained an average C-statistic of 0.512 and an average IBS of 0.201 for $n = 1,000$, $p = 100$, and $q = 10$, while it obtained 0.502 and 0.223 for $n = 100$. For $p = 1,000$ and $q = 20$, all the models obtained a lower IBS than the reference model. It was also the case for LASSO and IF with $p = 100$ and $q = 10$.

Regarding neural networks, CoxCC and CoxTime performed better than DeepHit. The two models reached very close results. This seems consistent with the fact that we did not simulate

time-dependent variable effects.

For IF and LASSO, the mean difference between survival probabilities for patients with or without treatment was larger than for the FNN models. The mean difference was also larger than the reference model. The closest model to the reference varies according to n, p, q. For instance, it is CoxTime for $n = 100$, $p = 1,000$ and $q = 20$, while it is IF for $n = 1,000$, $p = 1,000$ and $q = 20$.

6.4.6 . Results for AFT-Friedman data

In Table 6.2, the results from the AFT-Friedman simulation are presented. They were computed at the median time for each test set and are averaged over 100 simulation sets.

All the models performed better for a lower percentage of censoring (with more events in the data). The drop in performance was more important for IF: for instance, the average C-statistic went from 0.751 at 20% censoring to 0.502 at 50% censoring for $n = 100$. Compared to the other methods, LASSO was less impacted by censoring.

Overall, IFs achieved the best performances in terms of highest C-statistic (0.795 for $n = 1,000$ and 20% censoring) and lowest IBS (0.134 for $n = 1,000$ and 20% censoring). IFs performed better when they were trained on more individuals.

FNNs were the second-best performing methods. The best results were obtained when FNNs were trained using a larger sample size. The performances of CoxCC and CoxTime were very close in terms of the C-statistic or IBS. For a larger percentage of censoring, CoxTime performed better than CoxCC, and vice versa.

LASSO and IFs predicted more distinct survival predictions for individuals with treatment compared to the same individuals without treatment, as we can see with a larger mean difference in survival predictions. For the FNN models, this difference was smaller. For instance, the average difference is 0.051 for CoxTime with 50% censoring and trained using 100 patients, while it is 0.105 for IF. Logically, the proportion of individuals benefiting from the treatment is larger when the average difference is bigger.

Table 6.1: Average C-statistic, IBS Mean Diff. and Prop. Benef. across the 100 simulation sets for the CoxPH-Weibull data.

n		100			1,000		
p		10	100	1,000	10	100	1,000
q		2	10	20	2	10	20
Oracle	***C***	0.594 (±0.014)	0.632 (±0.015)	0.662 (±0.015)	0.594 (±0.014)	0.632 (±0.015)	0.662 (±0.015)
	IBS	0.161 (±0.012)	0.215 (±0.008)	0.247 (±0.013)	0.161 (±0.012)	0.215 (±0.008)	0.247 (±0.013)
	Mean Diff.	0.039 (±0.001)	0.156 (±0.001)	0.087 (±0.005)	0.039 (±0.001)	0.156 (±0.001)	0.087 (±0.005)
	Prop. Benef.	0.893 (±0.009)	0.996 (±0.002)	0.647 (±0.029)	0.893 (±0.009)	0.996 (±0.002)	0.647 (±0.029)
LASSO	***C***	**0.568** (±**0.013**)	**0.570** (±**0.013**)	**0.570** (±**0.013**)	0.568 (±0.012)	**0.569** (±**0.012**)	**0.567** (±**0.013**)
	IBS	**0.189** (±**0.012**)	**0.193** (±**0.010**)	**0.193** (±**0.010**)	**0.191** (±**0.005**)	**0.192** (±**0.006**)	**0.195** (±**0.004**)
	Mean Diff.	0.179 (±0.084)	**0.184** (±**0.081**)	0.162 (±0.071)	0.180 (±0.06)	**0.181** (±**0.044**)	0.171 (±0.029)
	Prop. Benef.	0.918 (±0.274)	0.957 (±0.203)	0.980 (±0.114)	0.923 (±0.266)	0.976 (±0.151)	0.998 (±0.001)
Interaction Forests	***C***	0.553 (±0.02)	0.550 (±0.021)	0.517 (±0.018)	**0.571** (±**0.015**)	0.582 (±0.013)	0.550 (±0.019)
	IBS	0.194 (±0.011)	0.200 (±0.009)	0.205 (±0.008)	0.193 (±0.005)	0.198 (±0.004)	0.206 (±0.003)
	Mean Diff.	0.111 (±0.047)	0.035 (±0.021)	0.003 (±0.005)	0.141 (±0.029)	0.069 (±0.018)	**0.011** (±**0.006**)
	Prop. Benef.	0.904 (±0.142)	0.691 (±0.406)	0.018 (±0.107)	0.962 (±0.039)	0.997 (±0.011)	0.069 (±0.222)
CoxCC	***C***	0.514 (±0.021)	0.502 (±0.016)	0.500 (±0.012)	0.556 (±0.018)	0.512 (±0.016)	0.502 (±0.014)
	IBS	0.215 (±0.02)	0.223 (±0.024)	0.224 (±0.025)	0.195 (±0.006)	0.201 (±0.007)	0.203 (±0.005)
	Mean Diff.	0.085 (±0.064)	0.02 (±0.012)	0.005 (±0.004)	0.140 (±0.046)	0.028 (±0.029)	0.003 (±0.003)
	Prop. Benef.	0.653 (±0.280)	0.288 (±0.255)	0.028 (±0.005)	0.972 (±0.113)	0.438 (±0.445)	0.002 (±0.017)
CoxTime	***C***	0.512 (±0.019)	0.502 (±0.014)	0.504 (±0.015)	0.556 (±0.018)	0.512 (±0.016)	0.500 (±0.014)
	IBS	0.214 (±0.02)	0.221 (±0.021)	0.224 (±0.022)	0.195 (±0.006)	0.201 (±0.007)	0.203 (±0.007)
	Mean Diff.	**0.075** (±**0.05**)	0.021 (±0.013)	**0.006** (±**0.004**)	0.140 (±0.048)	0.031 (±0.026)	0.003 (±0.004)
	Prop. Benef.	0.65 (±0.29)	0.296 (±0.255)	0.027 (±0.074)	0.970 (±0.143)	0.527 (±0.437)	0.019 (±0.084)
DeepHit	***C***	0.504 (±0.017)	0.502 (±0.014)	0.501 (±0.014)	0.506 (±0.02)	0.501 (±0.01)	0.502 (±0.012)
	IBS	0.224 (±0.026)	0.235 (±0.025)	0.241 (±0.031)	0.245 (±0.013)	0.253 (±0.017)	0.261 (±0.014)
	Mean Diff.	0.002 (±0.008)	0.001 (±0.002)	0.001 (±0.002)	**0.005** (±**0.033**)	0.001 (±0.006)	0.00 (±0.00)
	Prop. Benef.	0.017 (±0.096)	0.001 (±0.014)	0.002 (±0.018)	0.034 (±0.166)	0.005 0.053	0.00 (±0.00)

Notes. The value in brackets is the standard deviation across the 100 simulation sets. Highest C-statistic values and lowest IBS values are in bold. The closest value of Mean Diff to the Oracle model is in bold.

Table 6.2: Average C-statistic, IBS Mean Diff. and Prop. Benef. at median survival time across the 100 simulation sets for the AFT-Friedman data with treatment interaction.

n		100		1,000	
Censoring rate		20%	50%	20%	50%
LASSO	*C*	0.690 (±0.061)	**0.627** (±**0.058**)	0.713 (±0.052)	0.666 (±0.045)
	IBS	0.155 (±0.016)	**0.147** (±**0.013**)	0.149 (±0.016)	0.140 (±0.013)
	Mean Diff.	0.249 (±0.090)	0.214 (±0.095)	0.256 (±0.073)	0.226 (±0.074)
	Prop. Benef.	0.739 (±0.209)	0.737 (±0.246)	0.709 (±0.177)	0.695 (±0.181)
Interaction Forests	*C*	**0.751** (±**0.041**)	0.502 (±0.015)	**0.795** (±**0.034**)	0.503 (±0.016)
	IBS	**0.151** (±**0.012**)	0.167 (±0.017)	**0.134** (±**0.011**)	0.172 (±0.019)
	Mean Diff.	0.099 (±0.093)	0.105 (±0.098)	0.127 (±0.108)	0.136 (±0.113)
	Prop. Benef.	0.758 (±0.358)	0.768 (±0.356)	0.768 (±0.352)	0.782 (±0.346)
CoxCC	*C*	0.635 (±0.060)	0.569 (±0.047)	0.753 (±0.060)	0.670 (±0.048)
	IBS	0.173 (±0.019)	0.165 (±0.017)	0.135 (±0.017)	**0.139** (±**0.011**)
	Mean Diff.	0.069 (±0.072)	0.047 (±0.032)	0.113 (±0.145)	0.106 (±0.086)
	Prop. Benef.	0.543 (±0.324)	0.483 (±0.305)	0.776 (±0.258)	0.745 (±0.324)
CoxTime	*C*	0.632 (±0.06)	0.576 (±0.051)	0.750 (±0.06)	**0.671** (±**0.047**)
	IBS	0.174 (±0.016)	0.166 (±0.017)	0.135 (±0.019)	**0.139** (±**0.012**)
	Mean Diff.	0.061 (±0.057)	0.051 (±0.039)	0.142 (±0.115)	0.104 (±0.084)
	Prop. Benef.	0.576 (±0.319)	0.502 (±0.320)	0.747 (±0.285)	0.722 (±0.33)
DeepHit	*C*	0.608 (±0.074)	0.551 (±0.053)	0.713 (±0.091)	0.583 (±0.075)
	IBS	0.208 (±0.061)	0.169 (±0.023)	0.196 (±0.033)	0.177 (±0.025)
	Mean Diff.	0.019 (±0.038)	0.006 (±0.014)	0.041 (±0.066)	0.009 (±0.031)
	Prop. Benef.	0.15 (±0.301)	0.031 (±0.119)	0.372 (±0.406)	0.054 (±0.199)

Notes. The value in brackets is the standard deviation across the 100 simulation sets. Highest C-statistics and lowest IBS are in bold.

6.5 . Application on cancer cohorts: trastuzumab

6.5.1 . Data description

The models were compared on data from a RCT funded by the National Cancer Institute and the National Institute of Health. This randomized phase III clinical trial evaluated the effect of adding trastuzumab to adjuvant chemotherapy in early breast cancer patients in Her2$^+$ breast cancer (Pogue-Geile et al., 2013). There exist 4 subtypes of breast cancer based on the immunohistochemistry expression of estrogen receptor (ER), progesterone receptor (PR), or human epidermal growth factor receptor 2 (Her2): ER/PR$^+$, Her2$^+$; ER/PR$^+$, Her2$^-$; ER/PR$^-$, Her2$^+$; and ER/PR$^-$, Her2$^-$ (or Triple negative). Trastuzumab is a monoclonal antibody that is directed against Her2 protein overexpressed in approximately 20% of breast cancer patients with proven efficacy for both macro disease (metastatic and neoadjuvant setting) and micrometastatic disease (adjuvant setting).

As illustrated in Figure 6.2, $n = 1,574$ patients were included in the randomized trial: $n = 795$ patients received chemotherapy, while $n = 779$ others received chemotherapy and trastuzumab. For each patient, the ER status and number of positive nodes was available, as well as the expression of $p = 462$ genes. The nCounter was used for gene expression profiling. As the number of genes included in an nCounter assay is limited to less than 50, a microarray-based screening was performed beforehand for initial candidate gene discovery. The nCounter assay was designed with 462 probes to include candidate prognostic and predictive genes of trastuzumab benefit in the adjuvant setting.

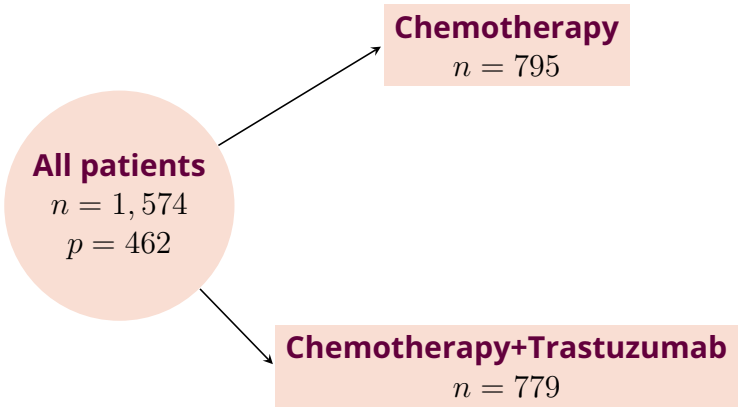


Figure 6.1: Randomized Control Trial to test for the evaluation of adding trastuzumab to chemotherapy. n represents the number of patients in each group, p is the number of genes.

72% of the survival times are censored. The distant recurrence-free survival at 5 years equals 0.75 (95% CI: $[0.73, 0.77]$). On average, there is a benefit in adding trastuzumab to chemotherapy in terms of relapse-free survival in the study population, with a hazard ratio $HR = 0.46$ (95% CI: $[0.38, 0.56]$). The difference in terms of survival time between the two treatment groups is represented in Figure 6.2.

6.5.2 . Results

The results are presented in Table 6.3. They are computed at 5 years and averaged over the 5 folds.

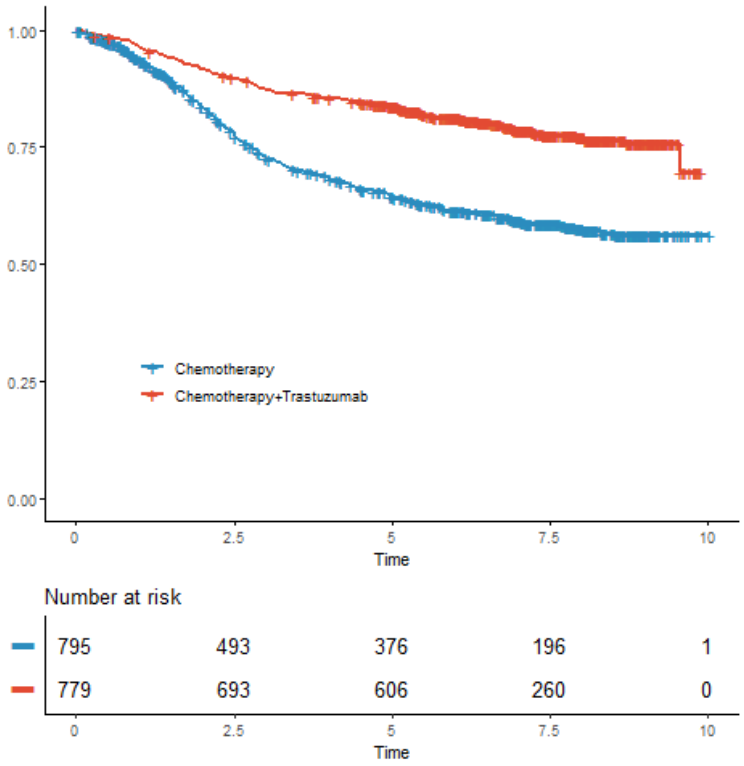


Figure 6.2: Kaplan Meier survival curves with stratification per treatment group.

LASSO achieved the best performance with the highest C-statistic and lowest IBS, and IFs obtained the second-best results. Regarding the FNNs, CoxCC and CoxTime achieved close results, with a higher C-statistic for CoxCC and a lower IBS for CoxTime. The mean difference between survival probabilities for a patient with treatment and the same patient without treatment was larger when the FNN model achieved better performances.

Table 6.3: 5 folds average measures at 5 years.

Model	C	IBS	Mean Diff.	Prop. Benef.
LASSO	**0.689** (±**0.039**)	**0.122** (±**0.012**)	0.152 (±0.034)	0.998 (±0.004)
Interaction Forests	0.616 (±0.045)	0.133 (±0.007)	0.021 (±0.01)	0.497 (±0.048)
CoxCC	0.589 (±0.040)	0.146 (±0.015)	0.039 (±0.020)	0.570 (±0.180)
CoxTime	0.572 (±0.032)	0.141 (±0.009)	0.028 (±0.024)	0.472 (±0.340)
DeepHit	0.565 (±0.028)	0.212 (±0.073)	0.022 (±0.018)	0.440 (±0.283)

Notes. The value in brackets is the standard deviation across the 100 simulation sets. The highest C-statistics and lowest IBS are in bold.

In Table 6.4, we see that the introduction of the encoder part of a pre-trained CVAE improved the performances of all the FNNs. The biggest improvement concerns DeepHit, which performs the best in this context. This outlines the fact that the FNNs have difficulty capturing the salient features contained in the data.

Table 6.4: 5 folds average measures at 5 years with CVAE.

Model	C	IBS	Mean Diff.	Prop. Benef.
CoxCC	0.588 (±0.069)	0.137 (±0.013)	0.024 (±0.016)	0.477 (±0.057)
CoxTime	0.610 (±0.065)	0.136 (±0.017)	0.031 (±0.026)	0.507 (±0.033
DeepHit	**0.612** (±**0.056**)	**0.133** (±**0.012**)	0.031 (±0.009)	0.504 (±0.093)

Notes. The value in brackets is the standard deviation across the 5 folds. The highest C-statistics and lowest IBS are in bold.

6.6 . Conclusion and Discussion

In this chapter, we focused on the possibility of formulating individualized treatment recommendations based on patient characteristics. To do so, we used a counterfactual approach with FNNs and identified an individualized treatment effect in the context of a RCT (no confounding factor). Based on the work of Katzman et al. (2016), we formulated a recommendation treatment rule built using the difference in the expected probability at a fixed time point of an event in treatment minus that in control. The difference was compared to a common treatment threshold for all patients. Two simulation sets were used to compare the FNNs method: one introducing a nonlinear interaction with the treatment and one built on Friedman's random function generator. LASSO and IF were used as benchmarks. We illustrated our recommended treatment rule with a detailed analysis of a RCT for the treatment of early breast cancer. As in Chapter 4, we used VAE-based methods to try capturing the relevant features in the transcriptomic data. More specifically, we used a CVAE that includes a condition variable (here, the treatment). We saw an improvement in the results when the CVAE pre-training was included.

Efthimiou et al. (2022) worked on the question of how to assess the ability of prognostic models to accurately predict treatment benefit, developing measures of discrimination and calibration adapted to this context. They extended the work of Klaveren et al. (2017), who introduced a C-for-benefit specifically adapted to the evaluation of treatment benefit. Their work could be adapted to time-to-event data. In future work, we could also associate an uncertainty measure with our individualized treatment recommendations, using the ensemble methods introduced in Chapter 4.

We could also further work on clinical variables. Indeed, we did not adjust the models on clinical variables in our simulation study. Such variables could be considered prognostic factors or they could be used as candidates for treatment-variable interactions, just like the biomarkers.

In this Chapter, we assumed that the threshold for our treatment recommendation rule was common for all patients. As suggested by Efthimiou et al. (2022), we could explore cases where the threshold varies with different types of patients. For instance, if patients are at a higher risk of an adverse event, the threshold for treatment might be higher. The treatment recommendation rule could even include patient variables to account for this variability.

Another area of interest is the relative contribution of the input variables, as it could enable us to identify specific biomarkers interacting with the treatment. Hence, we could determine the biomarker-treatment score for patients defined as the cross-product between the coefficient of the interactions retained in the training set and their biomarkers. Thereafter, we could estimate a criterion based on the strength of interaction between gene signatures and measuring the concordance between the biomarker-treatment score and the survival time in each treatment arm, as presented in Ternès et al. (2017). Determining the relative importance of the different input variables is challenging in the case of FNNs. As it is a nonlinear type of model, simply looking at the resulting weights and biases outputted by the model is not possible. Also, FNNs do not have an assumption of non-colinearity of the inputs. There could be informative interactions among groups of variables. To overcome this issue, different methodologies have been defined to assess variable contributions in FNNs that focus on the weights of the model (Garson, 1991). We could use the Effect Importance Measure associated with IF as a benchmark.

7 - Conclusion and Discussion

This PhD addressed the question of survival analysis in the context of high-dimensional data. In this case, standard methods like the Cox proportional hazards model (Cox, 1972) cannot be implemented. Several methods have been proposed to reduce the dimension of the data. For instance, LASSO is based on the Cox partial likelihood and adds a penalisation term (Tibshirani, 1996). Here, we focused on neural networks, as their complex architecture offers great flexibility and makes it possible to overcome the rigidity of the assumptions of the CoxPH model. They can take into account the complex and non-linear relationships between the variables and the level of risk of the individual.

The aim of this work was to study the use of neural network models with time-to-event data, using specific ways to handle censorship. We gave an overview of survival analysis, and discussed state-of-the-art techniques. We detailed survival analysis in the context high dimensional data, and introduced FNNs. We estimated expected survival probabilities for individual patients applying FNNs on time-to-event data, and compared FNNs in terms of calibration and discrimination capacities. Then, we associated a measure of uncertainty with survival predictions based on an ensemble of neural networks. Finally, we predicted the expected treatment effect at the patient level in the context of a randomized clinical trial.

7.1 . Contribution

First, we compared existing survival models based on FNNs and using specific loss functions. In a continuous time framework, CoxCC (Kvamme et al., 2019) uses a loss based on a case-control approximation; CoxTime is an extension of CoxCC that includes the time variable. For the models defined in a discrete time framework, PLANN (Biganzoli et al., 1998) is built on the binary cross-entropy, and DeepHit (Lee et al., 2018) combines a log-likelihood with a ranking loss. DNNSurv (Zhao et al., 2019) uses pseudo observations. We proposed new ways of computing pseudo-observations. We used random survival forests by Ishwaran et al. (2008) and LASSO penalization as a benchmark, as these are commonly used benchmark models and usual competitors to neural networks. We studied the operating characteristics of the models in a simulation study in a high-dimensional setting. Data were simulated using different models (CoxPH with a Weibull risk function, AFT with a log-normal risk function, AFT including a random function generator). Simulated data had different levels of complexity. Indeed, we varied the number of variables, the sample size, the level of censoring, and the sparsity. A higher censoring rate reduced FNNs' performances. The FNNs defined in a continuous time framework performed the best, with closed results between CoxCC and CoxTime. For the discrete time framework, PLANN obtained competitive results. The pseudo-observation based FNNs were the ones that got the worst results. FNNs outperformed LASSO only for the AFT model, including a random function generator, that is, for complex and non linear data, and more specifically, when censoring was low and when the FNNs were trained on a large n. RSF obtained competitive results with the best performing FNNs, but always with a higher computational cost. Also, the FNNs had lower results when the number of variables increased, suggesting a need to perform variable selection and dimension reduction before training the FNNs model, as confirmed with real patient cohorts.

We also applied the neural network models to 2 patient cohorts. An important issue is integrating both clinical and genomic data in prediction models. Indeed, how the two types of variables are combined can impact the quality of model predictions. Here, we studied the performance differences between models that included all the variables and models that included clinical variables only. With the METABRIC breast cancer cohort, we initially obtained better results when using only clinical variables as input of the model. We were able to improve the results of the models with all the data in input of the model when using a variational-autoencoder to synthesize the information contained in the transcriptomic data. The Lung Cancer Explorer dataset obtained the best results by including all the variables. The results were improved when using a VAE. Overall, the model that obtained the best results on both real cohorts was CoxTime.

Second, we developed confidence intervals for expected survival probabilities based on the percentile method and using an ensemble of artificial neural networks. With Bootstrap (Efron, 1979), a training subset is randomly drawn with replacement in order to obtain M neural networks. To build Deep Ensembles (Lakshminarayanan et al., 2016), the weights of each of the M models are initialized randomly during training. Gal et al. (2016) propose to apply dropout randomly during the test phase (M draws): this is the Monte-Carlo Dropout. The idea of the fixed Bernoulli mask (Mancini et al., 2020) is to previously sample M dropout masks applied at the time of training and testing the model. We assessed the performance of the different ensemble methods with a simulation study, by comparing various models' survival predictions to the oracle value. Data was generated using the inverse cumulative method with a log-logistic risk function in a CoxPH model. The methods were also evaluated for the 2 cohorts, and we described expected survival estimates for two patients of each data set, representing uncertainty at the patient level. The neural network models CoxCC and CoxTime performed similarly. As the simulated data did not include time-dependent variables, there was no reason for CoxTime to perform better. CoxTime obtained the highest C-indices for the cohort studies. The bootstrap obtained encouraging results for the simulation study, but was computationally more costly; unfortunately, the computational cost could be a drawback when training deep neural networks. The coverage rate above 95% indicates that Bootstrap could be slightly too conservative. DeepEns, MCDrop, and BMask performed well in terms of empirical coverage. MCDrop was the second-best method after Boot, although it was slightly anti-conservative in the simulations. We recommend using MCDrop for deep neural networks since it performed well both for the simulation study and the applications and is less computer-intensive than Boot.

Third, we developed prediction models with biomarkers, called treatment-effect modifiers, to predict how much benefit individual patients would derive from specific treatments so as to take a therapeutic decision that best fits each individual patient. Our work was placed in the context of high-dimensional data. We developed a recommender system consisting of a difference in expected survival probabilities at a specific time point between a future patient who received the experimental treatment and the same patient who did not. We compared FNNs with two benchmark models: a penalized CoxPH model and Interaction Forest, which are RSF that specifically capture interactions between biomarkers and treatment. The models were compared using two simulation sets based on the AFT model: one introducing a nonlinear interaction with the treatment and one built on Friedman's random function generator. Then, we investigated for a potential benefit from treatment in a randomized clinical trial in a high-dimensional setting. Indeed, we applied the FNNs to a large

randomized phase III clinical trial that evaluated the effect of adding trastuzumab to chemotherapy in $1,574$ early breast cancer patients, for which the expression of 462 genes was measured. We also used CVAE to analyze the real patient cohort. It is a VAE-based method that specifically handles the treatment variable.

7.2 . Limits and Perspectives

During this Ph.D. research, we have been focusing on comparing different FNNs adapted to time-to-event data. We also benchmarked FNNs with LASSO, RSF, and Interaction Forests. For instance, in the second chapter, a total of 4 FNNs based on specific loss functions, 3 FNNs built using pseudo-observations, and 2 benchmark models were compared. However, this comparison was not exhaustive. The FNNs were chosen after conducting a literature search. The different FNNs were chosen so as to present a wide range of characteristics. Other existing FNNs could have been included in the study, like DeepSurv (Katzman et al., 2016) or Cox-nnet (Ching et al., 2018). Concerning the benchmark models, we picked LASSO and RSF, as the majority of the papers used these methods as benchmarks. We also wanted to compare our FNNs against credible competitors that behave well in the high-dimensional setting. We could have applied other benchmark methods, like GAMLASSO. We excluded more complex scenarios, such as competing risks and multi-state models, and did not compare the performances of FNNs in the context of classification. Bailly et al. (2022) presented such a comparison with the aim of diagnosing a disease.

The interest of simulation studies is to evaluate the performances of different methods on data sets where we know the underlying characteristics of the data (for instance, the effect of the biomarkers or the expected survival probability). Although there is no simulation study that can cover all possible cases, it is important that the simulation study shares characteristics with real patient cohorts. This is not always the case. Here, for instance, we showed that FNNs outperformed other methods that did not model complex nonlinear interactions only in the context of data simulated with Friedman's random function generator. Thus, these simulations seem relevant to understand the performances of FNNs in optimal situations, including highly complex interactions in the dataset, but the generalization of the results can remain delicate.

Another issue concerns the integration of clinical variables along with transcriptomic data. The inclusion of clinical variables in simulation studies is not straightforward, raising the question of how to combine low-dimensional clinical data with high-dimensional transcriptomic data. Here, we did not simulate clinical variables along with molecular data. We could have explored simulation settings presented in De Bin et al. (2019). For the 2 real patient cohorts, we investigated specific ways of handling both clinical and transcriptomic data in the same neural network using auto-encoders and variational auto-encoder structures inside a combined FNN. Specific uncertainty measures could be developed to handle both types of data based on these combined structures.

One of the limits concerning our ensembling strategies is that we did not investigate the uncertainty linked to the choice of the model. We could have accounted for this type of uncertainty by creating an ensemble of predictions based on different types of models. This amounts to combining predictions from multiple models. The predictions made by each model could have been weighted by some criterion, and the weighted mean of the predictions could have been reported. Also, we could have stacked the predictions with the help of a meta-model.

Theoretically, using machine-learning techniques, specifically FNNs, poses the problem of how

to make sure that the model does not become a “black box”, a complicated model that is not easily comprehensible. It raises the question of "interpretable machine learning" (Molnar et al., 2020). Here, we tried to develop an uncertainty measure based on an ensemble of neural networks to see how confident the model was in its predictions, trying to add some insights on the predictions outputted by the FNNs. Another challenge is to determine the relative of the different input variables in the case of FNNs. This would be particularly relevant in the case of RCTs, where we want to identify specific treatment modifiers and further develop gene signatures that are sparse. To overcome this issue, some methodologies have been tested but not in the context of time-to-event data. For example, connection weights methods focus on the weights of the models (Garson, 1991). While these techniques do show some interesting results, they are not always intuitive and not always stable over time. They are not the equivalent of regression coefficients and t-stats in linear models.

Another area of interest that we did not investigate concerns multi-task learning to handle diverse types of data in the context of survival prediction. These models present the advantage to learn on large datasets. Their structures include both common parts and specific parts, allowing them to adapt to each study case. Herrmann et al. (2020) worked on survival time predictions using multi-omics data. Goncalves et al. (2020) showed an improvement in the performance of 5-year survival prediction using multi-task learning applied to a set of 10 different datasets, each set concerning a type of cancer. But these different types of cancer are all related to human papillomavirus. In the context of breast cancer, we could combine cohorts corresponding to different subtypes, with a common part of the model and a specific part for each of the 4 breast cancer subtypes. Multi-task learning could even include autoencoders to extract salient features in multi-omics data, as is the case with CustOmics (Benkirane et al., 2022).

The acquisition of a large amount of molecular data allows us to characterize patients in great detail, which gives rise to great hope for the development of new diagnostic, prognostic, and treatment approaches. Similarly, new models and deep learning methods open up new perspectives in terms of precision and capacity. Nevertheless, the limited size of the cohorts poses statistical and numerical problems of interest for the years to come.

All this PhD work has been coded using Python software (and more specifically PyTorch package) for the implementation of the FNNs and RSF, and R sotfware for the simulations and the implementation of LASSO and IF. The different approaches discussed are easily usable for other applications. Moreover, the code associated with the second and the third chapter are publicly available on github, having been submitted for publications. One perspective of this work is to develop and to make publicly available a unified framework with all the codes and simulation methods of this PhD.

8 - Appendix

8.1 . Résumé substantiel en français

En oncologie, l'analyse de survie permet d'estimer la survenue d'évènements comme le décès, la récidive tumorale, ou encore l'apparition de métastases chez un ou plusieurs groupes d'individus. Elle a des applications multiples : étude de la survie d'un groupe de patients, comparaison des effets de différents traitements, etc. Une des caractéristiques des données de survie est l'existence d'observations incomplètes, et en particulier de la censure à droite. On utilise alors traditionnellement un modèle à risques proportionnels de Cox (Cox, 1972) pour les analyser. C'est un modèle multiplicatif du risque qui s'exprime par le produit d'une fonction de risque de base (identique pour tous les individus) et d'une fonction de variables explicatives permettant d'ajuster la fonction de risque aux caractéristiques des individus. Il s'appuie sur deux hypothèses, à savoir la proportionnalité des risques et la log-linéarité. Dans de nombreuses applications, ces hypothèses sont trop restrictives et ne permettent pas de prendre en compte facilement l'analyse d'un nombre toujours croissant de facteurs pronostiques et prédictifs, ni d'inclure toutes leurs interactions ou leurs effets non-linéaires. En particulier, le contexte actuel de multiplication et de diversification des données (cliniques, génétiques, imagerie) soulève de nouveaux enjeux. Le défi actuel est d'intégrer cette masse d'information incluant des données plus variées, de nature hétérogène (expression, nombre de copies, niveau de gris de voxels, etc) et de structure complexe (voie biologique, répartition spatiale sur l'image, etc). L'identification de marqueurs spécifiques au sein de ces données présente des difficultés en termes d'intégration de données de différentes natures, mais aussi de modélisation de la grande dimension. Grâce aux progrès techniques et à la puissance de calcul actuelle, il est possible d'implémenter des algorithmes complexes, notamment les réseaux de neurones. Ces algorithmes ont une capacité à traiter des problèmes divers (classification, prédiction, analyse de données). On trouve de nombreux exemples de leur utilisation : en reconnaissance d'images, en traitement du langage naturel, ou encore dans le domaine biomédical. Ils permettent en particulier de traiter des problèmes non structurés, c'est-à-dire sur lesquels on ne dispose d'aucune information préalable. Ils représentent un outil de modélisation des relations entre des données ou des fonctions particulièrement complexes. En biostatistique, les modèles de survie basés sur les réseaux de neurones sont une alternative aux modèles standards. En effet, l'architecture complexe des réseaux de neurones offre une grande flexibilité et permet de surmonter la rigidité des hypothèses du modèle de Cox. Ils peuvent notamment prendre en compte les relations complexes et non linéaires entre les facteurs pronostiques et le niveau de risque de l'individu, ou encore fournir une recommandation personnalisée basée sur ce risque. L'objectif de cette thèse est de se baser sur ces modèles et de les adapter au contexte des données censurées. Dans un premier temps, nous implémentons des réseaux de neurones répondant à la question de la censure à droite et les utilisons pour effectuer des prédictions de survie. Dans un second temps, nous mettons en place des intervalles de confiance construits à l'aide d'ensemble de réseaux de neurones et afin de mesurer l'incertitude des prédictions réalisées. Dans un troisième temps, nous effectuons des recommandations de traitement individualisées avec des réseaux de neurones adaptés à la survie et dans le cadre d'un essai clinique contrôlé randomisé.

Tout d'abord, nous appliquons des modèles de réseaux de neurones aux données de survie, en

utilisant des méthodes particulières pour traiter les observations censurées, comme par exemple les pseudo-observations ou des fonctions de perte spécifiques. Différents modèles de réseaux de neurones sont comparés. CoxCC et CoxTime (Kvamme et al., 2019) s'appuient sur une fonction de perte construite à partir de l'approximation cas-contrôle. DeepHit (Lee et al., 2018) est un modèle qui estime la fonction de masse de probabilité et associe la log-vraisemblance à une contrainte de rang. DNNSurv (Zhao et al., 2019) pallie le problème de la censure en utilisant des pseudso-observations. Dans notre travail, nous proposons également d'autres façons de calculer les pseudo-observations : en imposant une contrainte pour que la pseudo-observation soit comprise entre 0 et 1 (POpt), en construisant la pseudo-observation à l'aide de l'estimateur de Kaplan-Meier (PKM). Deux modèles de référence sont utilisés: le modèle de Cox pénalisé (Tibshirani, 1996) et les forêts aléatoires de survie (Ishwaran et al., 2007). La capacité prédictive des modèles est analysée en utilisant trois types de simulations. Les deux premières méthodes sont basées sur la méthode de la distribution cumulée inverse (Bender et al., 2005). La génération des temps de survie est d'abord effectuée à partir d'un modèle de Cox où l'on suppose que la fonction du risque de base du modèle suit une loi de Weibull, impliquant des risques proportionnels. Ensuite, les temps de survie sont simulés selon un modèle AFT associé à une loi log-normale, impliquant des risques non proportionnels. Enfin, une troisième méthode de simulmation introduite par Friedman (2001) est implémentée. Elle se base également sur un modèle AFT et inclue des interactions par paires de variables et des effets non linéaires pour des sous-ensembles aléatoires de ces variables. Les modèles sont comparés à l'aide de l'indice de concordance (Uno et al., 2011) et du score de Brier intégré. Plusieurs paramètres de simulations varient: le nombre de variables pronostiques parmi le nombre total de variables, le nombre d'individus utilisé pour entraîner le modèle, le taux de censure pour le troisième type de simulation (20% ou 50%). On simule 100 bases de données par scénario. Pour le premier type de données, les résultas du modèle de Cox pénalisé sont meilleurs que ceux des réseaux de neurones. Pour le deuxième ensemble de données simulées, le modèle de Cox pénalisé obtient à nouveau les meilleurs performances. Cela souligne le fait que les données simulées n'ont pas un effet non proportionnel fort. Pour le troisième type de simulations, la plupart des modèles de réseaux neuronaux proposés obtiennent des performances assez proches. Avec une censure de 20% et $n = 1000$ individus, tous les réseaux neuronaux sont plus performants que le modèle LASSO. Par ailleurs, on constate une baisse de performance pour tous les modèles lorsque la censure est plus élevée. Nous appliquons également ces méthodes au jeu de données METABRIC qui porte sur la survie après une chirurgie pour le cancer du sein. Il est composé de 1960 patientes, de 6 variables cliniques et de l'expression de 863 gènes. Les différents modèles de réseaux de neurones permettent d'obtenir des résultats très proches à 5 ans. Les résultats des réseaux de neurones sont meilleurs pour nos deux indicateurs si le modèle n'est entraîné que sur les variables cliniques. Les variables cliniques semblent contenir l'essentiel de l'information nécessaire à la prédiction. Mais si l'on construit un modèle en prétraitant les données de transcriptomique avec l'encodeur d'un autoencoder variationnel, les résultats des réseaux de neurones combinés sont meilleurs. Cela montre l'intérêt d'une méthode de réduction des données pour prétraiter les données de transcriptomique. Nous étudions finalement l'ensemble des modèles sur une base de données qui réunit diverses sources de données portant sur le cancer du poumon : c'est le Lung Cancer Explorer. Elle comporte 4120 patients, 3 variables cliniques et l'expression de 1000 gènes. Pour les données de cancer du poumon, les résultats sont comparables à 2 ans. Les résultats obtenus avec les forêts aléatoires de survie et le modèle de Cox pénalisé sont légèrement meilleurs pour les données de cancer du sein et de cancer du poumon.

Nous étudions ensuite la stabilité des résultats obtenus avec ces réseaux de neurones, point essentiel pour le praticien qui a besoin d'un indicateur de la confiance qu'il peut avoir dans les prédictions apportées par le modèle. Nous développons des méthodes permettant d'appréhender l'incertitude inhérente à l'apprentissage profond. Nous nous penchons sur la question de l'erreur de prédiction, mais aussi sur le niveau d'incertitude de la prédiction de survie du modèle obtenu. Différentes méthodes sont comparées. Tout d'abord, nous adoptons une méthode non paramétrique de rééchantillonnage. Nous générons M échantillons tirés au sort à partir de l'échantillon initial et estimons le modèle au sein de chaque échantillon : c'est la méthode du boostrap (Efron, 1979). Une deuxième méthode consiste à combiner les prédictions issues de M modèles entraînés avec initialisation aléatoire des paramètres : c'est la méthode de DeepEnsemble (Lakshminarayanan et al., 2016). Gal et al. (2016) proposent d'utiliser la technique du dropout (initialiement proposée pour pallier le sur-apprentissage) au cours de la phase de test et d'effectuer M tirages de façon aléatoire: c'est le Monte-Carlo Dropout. L'idée du masque de Bernoulli fixé (Mancini et al., 2020) est de générer préalablement M masques de dropout appliqués au moment de l'entraînement et du test du modèle. Une étude de simulation et deux applications nous permettent de comparer et d'évaluer les ensembles. Nous mettons en oeuvre une étude de simulation afin d'évaluer le taux de couverture des intervalles de confiance obtenus. Nous simulons des données avec des interactions non-linéaires et en utilisant la méthode de la distribution cumulée inverse (Bender et al., 2005). Les temps de survie sont simulés selon le modèle de Cox, avec pour distribution du risque de base une fonction log-logistique. 100 bases de données sont simulées, composées chacune de 4 000 individus (séparés en 2 000 individus pour l'entraînement et 2 000 pour le test), avec un taux de censure à 20%. Les paramètres sont estimés sur les données d'entraînement et les prédictions et intervalles de confiance à 95% estimés sur la base de test. La qualité de l'estimation de la survie prédite est évaluée à travers le biais, celle des intervalles de confiance est évaluée grâce au taux de couverture. L'indice de concordance le plus proche du modèle oracle est obtenu avec la méthode du MCDropout, en particulier avec le modèle CoxTime. Les masques de Bernoulli fixés permettent d'obtenir le taux de couverture le plus élevé, en étant appliqué au modèle CoxTime. Nous appliquons également les modèles aux données réelles présentées en première partie. Nous constituons ensuite des intervalles de prédiction au niveau individuel sur deux bases de données réelles. Grâce aux méthodes d'ensembles, on obtient des intervalles de confiance pour les prédictions de survie avec réseau de neurones. En nous basant sur les résultats de notre étude, nous préconisons l'utilisons de CoxTime, avec la méthode du MCDropout ou celle du masque de Bernoulli fixé, ces deux méthodes ayant des résultats proches.

Finalement, nous nous intéressons à la possibilité de formuler des recommandations individualisées de traitement en fonction des caractéristiques du patient. On s'intéresse plus particulièrement aux biomarqueurs prédictifs permettant de prédire l'efficacité du traitment proposé. Se met alors en place une médecine stratifiée, qui utilise ces facteurs, ces caractéristiques biologiques (et en particulier génétiques), pour choisir le traitement le plus adapté au patient. En effet, la médecine stratifiée est une approche thérapeutique dont l'objectif est de sélectionner les patients auxquels administrer un traitement en fonction d'un marqueur, afin de ne traiter que la sous population susceptible de recevoir un bénéfice du traitement. Pour ce faire, nous utilisons des méthodes d'inférence causale pour mesurer l'effet d'un traitement en tenant compte d'un ensemble de facteurs d'influence représentés

par des variables observables caractérisant le patient (approche contrefactuelle). Nous construisons une règle de décision basée sur une différence de probabilité de survie attendue à un temps donné entre la probabilité de survie avec l'administration du traitement 1 et la probabilité de survie avec l'administration du traitement 0 pour le même individu donné. Nous utilisons deux types de bases de données simulées à l'aide d'un modèle AFT, l'un introduisant une interaction non linéaire avec le traitement et l'autre construit à partir d'un générateur de fonctions aléatoires de Friedman. Le modèle de Cox pénalisé et les forêts d'interaction ont été utilisés comme modèles de référence. Nous avons illustré notre règle de recommandation de traitement par une analyse détaillée d'un essai clinique randomisé pour le traitement du cancer du sein de stade précoce. Cette base qui contient les données des 1 574 patientes participant à un essai randomisé promu par le National Cancer Institute / National Institute of Health et évaluant l'effet du trastuzumab adjuvant dans le cancer du sein. L'expression de 462 gènes est disponible en plus des variables clinico-pathologiques telles que les statuts ER et ganglionnaire, ou encore la taille de la tumeur. Comme dans la première partie, nous avons utilisé des méthodes basées sur la VAE pour essayer de capturer l'essentiel des caractéristiques des données transcriptomiques. Plus précisément, nous avons utilisé un autoencodeur variationnel conditionnel, qui inclut une variable de condition (ici, le traitement). Nous constatons une amélioration des résultats avec le pré-entraînement CVAE.

L'intérêt des études de simulation est d'évaluer les performances de différentes méthodes sur des données dont nous connaissons les caractéristiques sous-jacentes (par exemple, l'effet des biomarqueurs ou la probabilité de survie attendue). Bien qu'aucune étude de simulation ne puisse couvrir toutes les situations possibles, il est important que l'étude de simulation partage des caractéristiques avec les ensembles de données réelles. Ce n'est pas toujours le cas. Ici, par exemple, nous avons montré que les réseaux de neurones artificiels surpassaient d'autres méthodes ne modélisant pas les interactions non linéaires complexes uniquement dans le contexte de données simulées avec le générateur de fonctions aléatoires de Friedman. Ainsi, ces simulations semblent pertinentes pour comprendre les performances des réseaux de neurones dans des situations optimales pour eux, incluant des interactions très complexes dans le jeu de données, mais la généralisation des résultats peut rester délicate.

Une autre question concerne l'intégration des variables cliniques avec les données transcriptomiques. L'inclusion de variables cliniques dans les études de simulation n'est pas simple. Ici, nous n'avons pas simulé de variables cliniques. Aucun travail n'a été trouvé sur la façon de simuler des données contenant à la fois des variables cliniques et génomiques. Pour les ensembles de données réelles, nous avons travaillé sur des méthodes spécifiques de traitement des données cliniques et transcriptomiques dans le même réseau neuronal en utilisant des structures d'auto-encodeurs et d'auto-encodeurs variationnels dans un réseau de neurones combiné. Des mesures d'incertitude spécifiques pourraient être développées pour traiter les deux types de données sur la base de ces structures combinées.

Théoriquement, l'utilisation de techniques d'apprentissage automatique, en particulier les réseaux de neurones, pose le problème de savoir comment s'assurer que le modèle ne devienne pas une "boîte noire", un modèle compliqué et difficilement compréhensible. Ici, nous avons essayé de développer une mesure d'incertitude basée sur un ensemble de réseaux neuronaux afin de déterminer le degré de confiance du modèle dans ses prédictions. Un autre défi consiste à déterminer le poids des différentes variables d'entrée du réseau de neurones. Cela serait particulièrement intéressant dans le cas de l'essai

clinique contrôlé randomisé, grâce auquel nous pourrions identifier les biomarqueurs interagissant avec le traitement et développer des signatures de gènes. Pour surmonter ce problème, certaines méthodologies ont été testées, mais pas dans le contexte de données de survie. Par exemple, les méthodes de poids de connexion se concentrent sur les poids du modèle (Garson, 1991). Bien que ces techniques donnent des résultats intéressants, elles ne sont pas toujours intuitives et stables dans le temps.

Par ailleurs, nous ne nous sommes pas intéressés ici à la question du traitement de différentes sources de données. L'apprentissage multi-tâche pourrait permettre de traiter de façon conjointe plusieurs types de jeu de données différents mais comparables, avec dans les modèles des parties communes (qui peuvent profiter de jeux de données plus importants pour l'apprentissage) et des parties spécifiques permettant de s'adapter à chaque cas d'étude. Goncalves et al. (2020) a montré une amélioration de performance de la prédiction de survie à 5 ans en utilisant un apprentissage multi-tâche appliqué sur un ensemble de 10 jeux de données différents dont chaque jeu concerne un type de cancer. Mais ces différents types de cancer sont tous liés aux papillomavirus. Par exemple, pour le cancer du sein, nous pourrions combiner des cohortes correspondant aux différents sous-types HER+, ER+/HER−, Triple Négatif, avec une partie commune du modèle et une partie spécifique.

8.2 . Tree-Parzen Algorithm

The Tree-structured Parzen Estimator (TPE) is a sequential model-based optimization algorithm: it uses a surrogate model and an acquisition function to find the best model by iteratively selecting the most promising hyperparameters in the search space to approximate the actual objective function. Let x represents the hyperparameter configuration and y the associated quality score. TPE uses Bayes rule to compute $p(x, y)$ by first determining $p(y)$ and $p(x|y)$. Multiple sets of x are randomly sampled to get an initial set of loss scores y. The scores y are separated into the best loss scores and the worst ones according to some threshold y^* (for instance, a percentile). Two densities $l(x)$ and $g(x)$ are constructed based on these two groups of values: $l(x)$ is modeled by $p(x|y < y^*)$, and $g(x)$ by $p(x|y \geq y^*)$. With each iteration of the TPE algorithm, a hyperparameter configuration is sampled and evaluated, and the densities are adjusted. The densities $l(x)$ and $g(x)$ are modeled using Parzen estimators. Such estimators are organized in a tree structure. It means that for each hyperparameter in the configuration, a fit for each density is provided. One of the great drawbacks of such a tree structure is that it does not take into account model interactions between the hyperparameters. In each iteration of the algorithm, the hyperparameter configuration is chosen based on the the acquisition function. The function used here is the Expected Improvement (EI). The idea is to choose hyperparameters from $l(x)$. Thus, the EI is maximized where $\frac{l(x)}{g(x)}$ is maximized.

1. Definition of a search space for each hyperparameter with a specific distribution and boundary values. Definition of a performance score y.
2. Creation of an objective function that takes the hyperparameters as input and provides a score as output.
3. Scores computation by randomly sampling a set of hyperparameters.
4. Classification of the obtained scores in two groups according to a certain threshold: the first group with the best scores and the second group with the remaining scores.
5. Construction (draw) of two densities $l(x)$ and $g(x)$ by the Parzen-Rosenblatt method (kernel estimation).
6. Construction of an hyperparameter set based on $l(x)$ and selection of the set that achieves the minimum value according to the ratio $\frac{l(x)}{g(x)}$, which is the largest expected improvement.
7. Evaluation of this set of hyperparameters with the objective function and update of the list of scores obtained in 3.
8. Repeat steps 4 to 7 N times, with N the number of trials. Here $N = 200$.

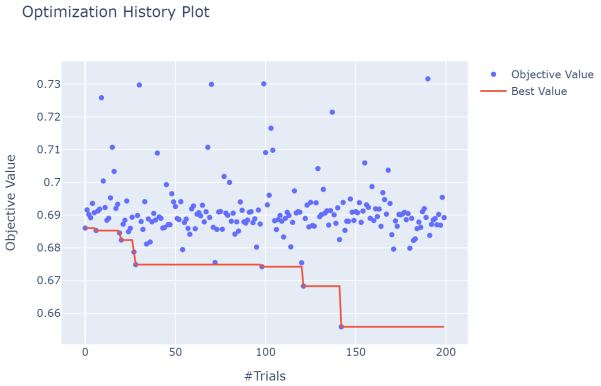
In Table 8.1, we see an example of selected hyperparameters for CoxCC trained on one dataset simulated with the CoxPH-Weibull method.

The hyperparameters in table 8.1 were selected based on 200 repepetions of the TPE algorithm, with 200 hyperparameter configurations being sampled. We used the optuna package in Python. Figure 8.1 represents different aspects of this hyperparameter search. Figure 8.1a represents the progress of the optimization throughout the trials: we see that the objective value does not go down after 150 trials. 200 trials was probably more than necessary. Figure 8.1b shows which of the

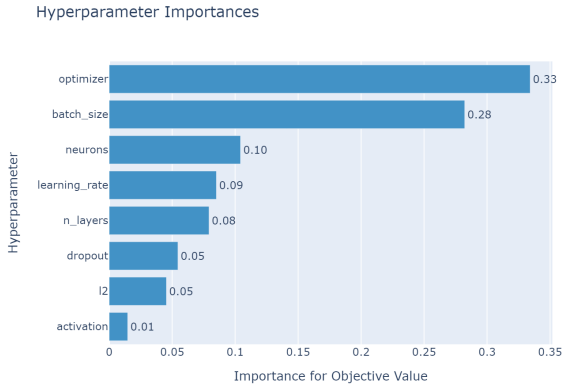
Hyperparameter	Value
Activation function	tanh
Batch size	8
Dropout rate	0.19
L_2 regularization	0.04
Learning rate	0.003
Number of hidden layers	3
Number of neurons per layer	99
Optimization algorithm	Adam

Table 8.1: Hyperparameter values selected for one simulated set.

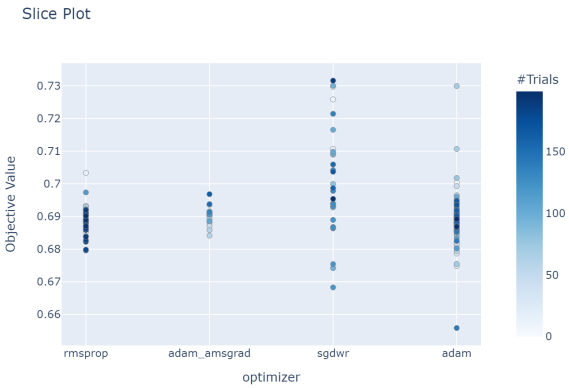
hyperparameter affects the performance the most: here, it is the optimization algorithm. Figure 8.1c and 8.1d describe the optimization of two specific hyperparameters, that are the number of layers and the optimization algorithm. We see how the TPE algorithm chooses the optimal value throughout the trials with the darkening color of the points.



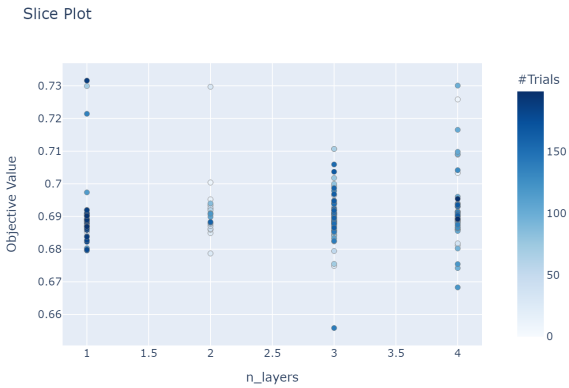
(a) Objective value



(b) Parameters Importance



(c) Trials for "Number of layers"



(d) Trials for "Optimiser"

Figure 8.1: Illustration of the hyperparameter search using the Tree-Parzen algorithm with the optuna package in Python.

8.3 . Simulation study

In Chapter 4, we did a simulation study with 3 simulation settings, using either a CoxPH-Weibull model, an AFT-LN model, or an AFT-Friedman model. Here, we present further results of this

simulation study.

8.3.1 . Selected variables with LASSO

With the simulation study, we know which variables are really prognostic. Based on this information, we define the variables selected by LASSO that are actually prognostic as True Positive (TP). A False Positive (FP) is a biomarker that is incorrectly selected by the LASSO procedure. We compute TP and FP for each simulation set. Precision can be computed based on these 2 numbers. The precision is the ratio of correctly positively identified prognostic biomarkers by LASSO to all selected biomarkers:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{8.1}$$

Tables 8.2 and 8.3 show the average TP, FP and Precision over the 100 simulation sets. We can see that all the prognostic biomarkers are identified by the LASSO for $q = 2$, and almost all for $q = 10$ and $q = 100$ on average. A few FP are also selected, especially when p is larger.

Table 8.2: Average TP, FP and Precision score for CoxPH Weibull data with LASSO.

n	**100**			**1,000**		
p	**10**	**100**	**1,000**	**10**	**100**	**1,000**
q	**2**	**10**	**20**	**2**	**10**	**20**
TP	2.0 (±0.0)	5.68 (±2.963)	10.42 (±1.534)	2.0 (±0.0)	10.0 (±0.0)	18.82 (±1.252)
FP	3.219 (±1.964)	9.21 (±6.759)	0.102 (±0.178)	3.56 (±1.768)	25.55 (±6.786)	70.28 (±16.343)
Precision	0.453 (±0.205)	0.41 (±0.224)	0.102 (±0.178)	0.404 (±0.153)	0.293 (±0.065)	0.218 (±0.04)

Notes. The value in brackets is the standard deviation across the 100 simulation sets.

Table 8.3: Average TP, FP and Precision score for AFT-LN data with LASSO.

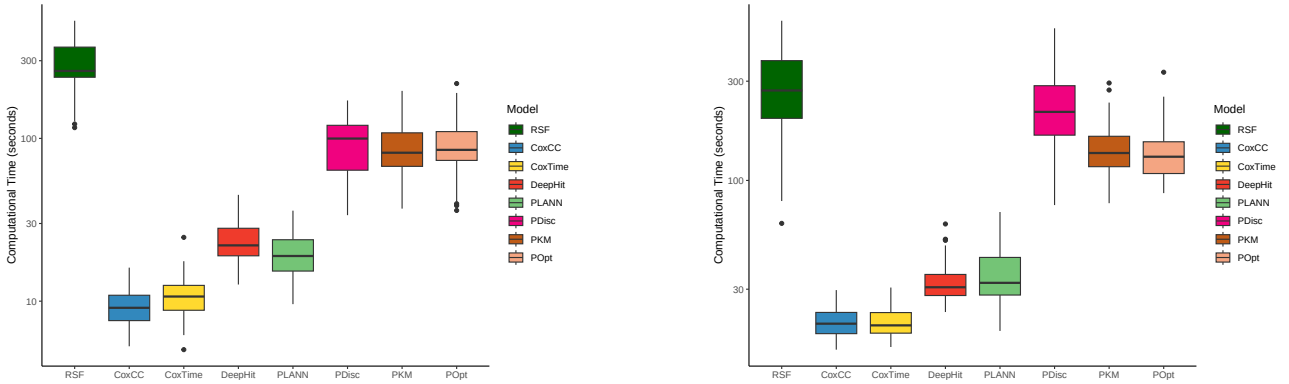
n	**100**			**1,000**		
p	**10**	**100**	**1,000**	**10**	**100**	**1,000**
q	**2**	**10**	**20**	**2**	**10**	**20**
TP	2.0 (±0.0)	9.97 (±0.222)	7.02 (±4.152)	2.0 (±0.0)	10.0 (±0.0)	20.0 (±0.0)
FP	3.79 (±1.824)	21.16 (±5.449)	15.56 (±12.175)	3.86 (±1.643)	29.85 (±6.983)	111.68 (±14.421)
Precision	0.388 (±0.148)	0.331 (±0.061)	0.323 (±0.205)	0.374 (±0.127)	0.259 (±0.048)	0.154 (±0.017)

Notes. The value in brackets is the standard deviation across the 100 simulation sets.

8.3.2 . Computational Time

In Figures 8.2a and 8.2b, we compared the running times of the different models for the hyperparameter search based on the Tree-Parzen algorithm and using a 5-fold cross-validation in Python. The ordering of the models according to the computation times is the same, no matter what parameters are used to simulate the CoxPH-Weibull data. We have 4 groups of models, as already seen in

Figure 4.3a. The fastest models are CoxCC and CoxTime, followed by PLANN and DeepHit. The pseudo-observation based models are the third group of models in terms of computing time. Finally, RSF is the slowest model.

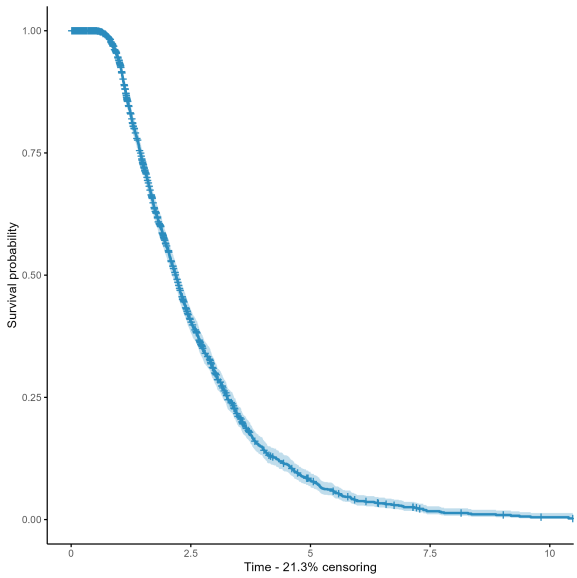


(a) $q = 10$, $p = 100$

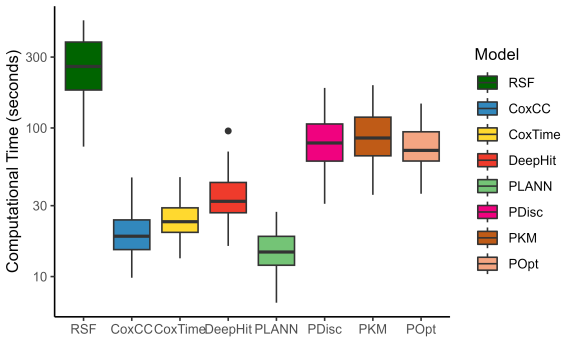
(b) $q = 100$, $p = 1000$

Figure 8.2: On the left size, computational time in seconds of the models on the logarithmic scale for CoxPH-Weibull data. Each boxplot displays 100 points corresponding to each simulated set (with $n = 2,000$, $p = 100$, $q = 10$). Each point represents the average over the 200 iterations of one 5-fold cross validation. On the right side, same figure with $n = 2,000$, $p = 1000$, $q = 20$.

For Figures 8.3a and 8.3b, we considered the AFT model, including a flexible random functions generator. The survival times here have a range of values close to the ones represented in Figure 4.5a. For the computational times, the ordering of the models is the same for models trained on data with 20% censoring compared to data with 50% censoring, except for PLANN, which is the fastest model.



(a) Survival Curve



(b) Computational Time

Figure 8.3: On the left size, survival curve for one simulated set with the AFT-Friedman method, $r = 0.2$ and $n = 2,000$. On the right side,computational time in seconds of the models on the logarithmic scale for CoxPH-Weibull data. Each boxplot displays 100 points corresponding to each simulated set (with $n = 2,000$, $r = 0.2$). Each point represents the average over the 200 iterations of one 5-fold cross validation.

8.4 . Application on cancer cohorts

In this section, we present the results obtained for the models compared in Chapter 4, either at 10 years on the METABRIC cohort, or at 5 years on the LCE cohort.

8.4.1 . Results for the METABRIC cohort

Table 8.4 shows the results at 10 years for the FNNs and the benchmark models. All the models performed better at 5 years compared to 10 results. As for the results at 5 years, all the models performed better when trained on clinical variables only. RSF obtained the best C-statistic and PLANN the best IBS when the models were trained with all the variables. These are the same results as the ones obtained at 5 years. For the models including only clinical variables, CoxTime had the highest C-statistic, and PLANN had the lowest IBS. Overall, CoxTime, PLANN, POpt, and PKM obtained close results. It was already the case at 5 years.

Table 8.4: Average C-statistic and Integrated Brier Score at 10 years for METABRIC data.

	All		Clinical	
Model	***C***	***IBS***	***C***	***IBS***
LASSO	0.595 (±0.023)	0.227 (±0.014)	0.658 (±0.025)	0.163 (±0.007)
RSF	**0.644** (±**0.028**)	0.175 (±0.006)	0.655 (±0.015)	0.161 (±0.005)
CoxCC	0.637 (±0.02)	0.171 (±0.004)	0.656 (±0.033)	0.165 (±0.012)
CoxTime	0.641 (±0.006)	0.179 (±0.006)	**0.668** (±**0.025**)	0.161 (±0.008)
DeepHit	0.628 (±0.062)	0.198 (±0.031)	0.633 (±0.025)	0.187 (±0.006)
PLANN	0.630 (±0.016)	**0.170** (±**0.005**)	0.663 (±0.025)	**0.131** (±**0.004**)
PDisc	0.624 (±0.021)	0.182 (±0.007)	0.642 (±0.024)	0.175 (±0.01)
PKM	0.626 (±0.017)	0.230 (±0.025)	0.666 (±0.023)	0.179 (±0.011)
POpt	0.614 (±0.021)	0.242 (±0.025)	0.665 (±0.015)	0.187 (±0.013)

Notes. The value in brackets is the standard deviation across the 5 folds. Highest C-statistics and lowest IBS are in bold.

8.4.2 . Results for the LCE cohort

Figure 8.5 shows the C-statistic and IBS computed at 5 years. All the models performed better at 5 years compared to 2 years. Overall, the C-statistic was lower, and the IBS was higher for models trained on clinical variables only compared to models trained with all variables. LASSO (CoxCC respectively) obtained the highest C-index when the models were trained on all the variables (on the clinical variables only, respectively), as was the case with the results at 2 years. Here, PDisc obtained the lowest IBS score at 5 years for the model with only clinical variables, when it was CoxCC at 2 years.

Table 8.5: Average C-statistic and Integrated Brier Score at 5 years for LCE data.

	All		Clinical	
Model	***C***	***IBS***	***C***	***IBS***
LASSO	**0.701** (±**0.013**)	0.235 (±0.01)	0.568 (±0.031)	0.275 (±0.012)
RSF	0.695 (±0.013)	0.173 (±0.007)	0.636 (±0.012)	0.189 (±0.003)
CoxCC	0.687 (±0.013)	**0.172** (±**0.002**)	**0.643** (±**0.023**)	0.185 (±0.005)
CoxTime	0.683 (±0.013)	0.175 (±0.006)	0.642 (±0.011)	0.186 (±0.004)
DeepHit	0.624 (±0.035)	0.205 (±0.009)	0.636 (±0.011)	0.195 (±0.002)
PLANN	0.667 (±0.012)	0.183 (±0.003)	0.642 (±0.018)	0.188 (±0.003)
PDisc	0.653 (±0.032)	0.194 (±0.008)	0.599 (±0.043)	**0.154** (±**0.016**)
PKM	0.657 (±0.014)	0.233 (±0.013)	0.631 (±0.015)	0.216 (±0.008)
POpt	0.663 (±0.028)	0.228 (±0.014)	0.631 (±0.01)	0.231 (±0.018)

Notes. The value in brackets is the standard deviation across the 5 folds. Highest C-statistics and lowest IBS values are in bold.

8.4.3 . Results for Trastuzumab

The models obtained higher C-statistics and lower IBS at 5 years, compared to 8 years. LASSO performeds the best, followed by IFs, as was the case at 5 years.

Table 8.6: 5 folds average measures at 8 years.

Model	*C*	*IBS*	*Mean Diff.*	*Prop. Benef.*
LASSO	**0.662** (±**0.04**)	**0.148** (±**0.015**)	0.183 (±0.031)	0.998 (±0.004)
Interaction Forests	0.601 (±0.05)	0.162 (±0.007)	0.018 (±0.011)	0.49 (±0.047)
CoxCC	0.573 (±0.048)	0.198 (±0.023)	0.052 (±0.026)	0.704 (±0.133)
CoxTime	0.506 (±0.114)	0.198 (±0.022)	0.038 (±0.029)	0.535 (±0.405)
DeepHit	0.490 (±0.073)	0.289 (±0.089)	0.002 (±0.001)	0.032 (±0.044)

Notes. The value in brackets is the standard deviation across the 5 folds. Highest C-statistics and lowest IBS are in bold.

Bibliography

[1] O. Aalen. “Nonparametric Inference for a Family of Counting Processes”. In: *The Annals of Statistics* 6.4 (1978), pp. 701 –726. doi: 10.1214/aos/1176344247.

[2] P. K. Andersen and M. P. Perme. “Pseudo-observations in survival analysis”. In: *Statistical Methods in Medical Research* 19 (1 Feb. 2010), pp. 71–99. issn: 09622802. doi: 10.1177/0962280209105020.

[3] A. Atkinson et al. “Biomarkers and surrogate endpoints: Preferred definitions and conceptual framework”. In: *Clin Pharmacol Ther* 69 (Mar. 2001), pp. 89–95.

[4] A. Bailly et al. “Effects of dataset size and interactions on the prediction performance of logistic regression and deep learning models”. In: *Computer Methods and Programs in Biomedicine* 213 (2022), p. 106504. issn: 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2021.106504.

[5] R. Bender, T. Augustin, and M. Blettner. “Generating survival times to simulate Cox proportional hazards models”. In: *Statistics in Medicine* 24 (11 June 2005), pp. 1713–1723. issn: 02776715. doi: 10.1002/sim.2059.

[6] H. Benkirane et al. *CustOmics: A versatile deep-learning based strategy for multi-omics integration*. 2022. doi: 10.48550/ARXIV.2209.05485.

[7] J. Bergstra et al. “Algorithms for Hyper-Parameter Optimization”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011.

[8] E. Biganzoli et al. “Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach”. en. In: *Statistics in Medicine* 17.10 (1998), pp. 1169–1186. issn: 1097-0258. doi: 10.1002/(SICI)1097-0258(19980530)17:10<1169::AID-SIM796>3.0.CO;2-D.

[9] A. Boileve et al. “Les organoïdes en oncologie digestive : état des lieux et perspectives. Hépato-Gastro & Oncologie Digestive”. In: *Hépato-Gastro & Oncologie Digestive* 9 (27 2020), pp. 865–872. doi: doi:10.1684/hpg.2020.2046.

[10] R. Bourgon, R. Gentleman, and W. Huber. “Independent filtering increases detection power for high-throughput experiments”. In: *Proceedings of the National Academy of Sciences of the United States of America* 107 (21 2010), pp. 9546–9551. issn: 00278424. doi: 10.1073/pnas.0914005107.

[11] L. Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32. issn: 0885-6125. doi: 10.1023/A:1010933404324.

[12] N. Breslow. “Discussion on Professor Cox’s Paper”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 34.2 (1972), pp. 202–220. doi: https://doi.org/10.1111/j.2517-6161.1972.tb00900.x. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1972.tb00900.x.

[13] C. C. Brown. *On the Use of Indicator Variables for Studying the Time-Dependence of Parameters in a Response-Time Model*. 1975, pp. 863–872.

[14] S. van Buuren and K. Groothuis-Oudshoorn. “mice: Multivariate imputation by chained equations in R”. In: *Journal of Statistical Software* 45 (3 2011), pp. 1–67. issn: 15487660. doi: 10.18637/jss.v045.i03.

[15] M. Charachon et al. “Leveraging Conditional Generative Models in a General Explanation Framework of Classifier Decisions”. In: *Future Gener. Comput. Syst.* 132.C (2022), 223–238. issn: 0167-739X. doi: 10.1016/j.future.2022.02.020.

[16] T. Ching, X. Zhu, and L. X. Garmire. “Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data”. In: *PLoS Computational Biology* 14 (4 Apr. 2018). issn: 15537358. doi: 10.1371/journal.pcbi.1006076.

[17] D. R. Cox. “Regression Models and Life-Tables”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 34.2 (1972), pp. 187–220. issn: 0035-9246.

[18] R. De Bin et al. “Combining clinical and molecular data in regression prediction models: insights from a simulation study”. In: *Briefings in Bioinformatics* 21.6 (Nov. 2019), pp. 1904–1919. issn: 1477-4054. doi: 10.1093/bib/bbz136. eprint: https://academic.oup.com/bib/article-pdf/21/6/1904/34672071/bbz136.pdf.

[19] M. W. Dusenberry et al. “Analyzing the Role of Model Uncertainty for Electronic Health Records”. In: *Proceedings of the ACM Conference on Health, Inference, and Learning*. CHIL ’20. New York, NY, USA: Association for Computing Machinery, 2020, 204–213. isbn: 9781450370462. doi: 10.1145/3368555.3384457.

[20] B. Efron. “Bootstrap Methods: Another Look at the Jackknife”. In: *The Annals of Statistics* 7.1 (1979), pp. 1–26. issn: 00905364.

[21] O. Efthimiou et al. “Measuring the performance of prediction models to personalize treatment choice”. In: *Statistics in Medicine* n/a.n/a (2022). doi: https://doi.org/10.1002/sim.9665. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.9665.

[22] J. Fan and J. Lv. “Sure independence screening for ultrahigh dimensional feature space”. en. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.5 (2008), pp. 849–911. issn: 1467-9868. doi: 10.1111/j.1467-9868.2008.00674.x.

[23] D. Faraggi and R. Simon. “A neural network model for survival data”. In: *Statistics in medicine* 14.1 (1995), pp. 73–82.

[24] J. H. Friedman. “Greedy function approximation: A gradient boosting machine”. In: *Annals of Statistics* 29 (5 2001), pp. 1189–1232. issn: 00905364. doi: 10.2307/2699986.

[25] Y. Gal and Z. Ghahramani. “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning”. In: *33rd International Conference on Machine Learning, ICML 2016* 3 (2016), pp. 1651–1660.

[26] G. D. Garson. “Interpreting Neural-Network Connection Weights”. In: *AI Expert* 6.4 (1991), 46–51. issn: 0888-3785.

[27] D. M. Gendoo et al. “MetaGxData: Clinically Annotated Breast, Ovarian and Pancreatic Cancer Datasets and their Use in Generating a Multi-Cancer Gene Signature”. In: *Scientific Reports* 9 (1 2019), pp. 1–14. issn: 20452322. doi: 10.1038/s41598-019-45165-4.

[28] P. Geurts, D. Ernst, and L. Wehenkel. “Extremely randomized trees”. In: *Machine Learning* 63 (1 Apr. 2006), pp. 3–42. issn: 08856125. doi: 10.1007/s10994-006-6226-1.

[29] A. Goncalves et al. “Improving five-year survival prediction via multitask learning across HPV-related cancers”. In: *PLoS ONE* 15 (11 November Nov. 2020). issn: 19326203. doi: 10.1371/journal.pone.0241225.

[30] I. J. Goodfellow et al. “Generative Adversarial Networks”. In: (June 2014).

[31] E. Graf et al. “Assessment and comparison of prognostic classification schemes for survival data”. In: *Statistics in Medicine* 18 (1718 1999), pp. 2529–2545. issn: 02776715. doi: 10.1002/(sici)1097-0258(19990915/30)18:17/18<2529::aid-sim274>3.3.co;2-x.

[32] B. Haller, K. Ulm, and A. Hapfelmeier. “A Simulation Study Comparing Different Statistical Approaches for the Identification of Predictive Biomarkers”. In: *Computational and Mathematical Methods in Medicine* 2019 (2019).

[33] J. Harrell Frank E. et al. “Evaluating the Yield of Medical Tests”. In: *JAMA* 247.18 (May 1982), pp. 2543–2546. issn: 0098-7484. doi: 10.1001/jama.1982.03320430047030.

[34] N. C. Henderson et al. “Individualized treatment effects with censored data via fully nonparametric Bayesian accelerated failure time models”. In: *Biostatistics* 21 (1 2020), pp. 50–68. issn: 14684357. doi: 10.1093/biostatistics/kxy028.

[35] M. Herrmann et al. “Large-scale benchmark study of survival prediction methods using multi-omics data”. In: *Briefings in Bioinformatics* 22.3 (Aug. 2020). bbaa167. issn: 1477-4054. doi: 10.1093/bib/bbaa167. eprint: https://academic.oup.com/bib/article-pdf/22/3/bbaa167/38657335/response_2_bbaa167.pdf.

[36] R. Hornung and A.-L. Boulesteix. “Interaction forests: Identifying and exploiting interpretable quantitative and qualitative interaction effects”. In: *Computational Statistics & Data Analysis* 171 (2022), p. 107460. issn: 0167-9473. doi: https://doi.org/10.1016/j.csda.2022.107460.

[37] J. T. G. Hwang and A. A. Ding. “Prediction Intervals for Artificial Neural Networks”. In: *Journal of the American Statistical Association* 92.438 (1997), pp. 748–757. doi: 10.1080/01621459.1997.10474027.

[38] H. Ishwaran and U. Kogalur. “Random survival forests for R”. In: *R News* 7.2 (2007), pp. 25–31.

[39] H. Ishwaran et al. “Random survival forests”. In: *Annals of Applied Statistics* 2 (3 2008), pp. 841–860. issn: 19326157. doi: 10.1214/08-AOAS169.

[40] G. Kantidakis et al. “A Simulation Study to Compare the Predictive Performance of Survival Neural Networks with Cox Models for Clinical Trial Data”. In: *Computational and Mathematical Methods in Medicine* 2021 (2021). issn: 17486718. doi: 10.1155/2021/2160322.

[41] E. L. Kaplan and P. Meier. “Nonparametric Estimation from Incomplete Observations”. In: *Journal of the American Statistical Association* 53.282 (1958), pp. 457–481. doi: 10.1080/01621459.1958.10501452. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1958.10501452.

[42] J. Katzman et al. “DeepSurv: Personalized Treatment Recommender System Using A Cox Proportional Hazards Deep Neural Network”. In: (June 2016). doi: 10.1186/s12874-018-0482-1.

[43] W. K. Kelly and S. Halabi. *Oncology clinical trials : successful design, conduct, and analysis*. Demos Medical Publishing, 2010.

[44] S. Kim et al. “Improved survival analysis by learning shared genomic information from pan-cancer data”. English. In: *Bioinformatics* 36 (2020). issn: 1367-4803. doi: 10.1093/BIOINFORMATICS/BTAA462.

[45] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2014. doi: 10.48550/ARXIV.1412.6980.

[46] D. Klaveren et al. “The proposed ‘concordance-statistic for benefit’ provided a useful metric when modeling heterogeneous treatment effects”. In: *Journal of Clinical Epidemiology* 94 (Nov. 2017). doi: 10.1016/j.jclinepi.2017.10.021.

[47] M. Krzywinski and N. Altman. “Points of significance: Importance of being uncertain”. In: *Nature methods* 10 (Sept. 2013), pp. 809–10. doi: 10.1038/nmeth.2613.

[48] H. Kvamme and O. Borgan. “Continuous and discrete-time survival prediction with neural networks”. In: *Lifetime Data Analysis* 27 (Oct. 2021), pp. 1–27. doi: 10.1007/s10985-021-09532-6.

[49] H. Kvamme, O. Borgan, and I. Scheel. *Time-to-Event Prediction with Neural Networks and Cox Regression*. 2019. arXiv: 1907.00825 [stat.ML].

[50] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Dec. 2016), 6405––6416.

[51] E. Lander et al. “Initial sequencing and analysis of the human genome. Nature”. In: *Nature* 409 (Jan. 2001), pp. 860–921.

[52] C. Lee et al. “DeepHit: A deep learning approach to survival analysis with competing risks”. In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018* (2018), pp. 2314–2321.

[53] E. T. Lee and O. T. Go. “SURVIVAL ANALYSIS IN PUBLIC HEALTH RESEARCH”. In: *Annual Review of Public Health* 18.1 (1997), pp. 105–134. doi: 10.1146/annurev.publhealth.18.1.105.

[54] L. M. Leemis, L.-H. Shih, and K. Reynertson. “Variate generation for accelerated life and proportional hazards models with time dependent covariates”. en. In: *Statistics & Probability Letters* 10.4 (Sept. 1990), pp. 335–339. issn: 01677152. doi: 10.1016/0167-7152(90)90052-9.

[55] L. P. Levasseur, Y. D. Hezaveh, and R. H. Wechsler. “Uncertainties in Parameters Estimated with Neural Networks: Application to Strong Gravitational Lensing”. In: *The Astrophysical Journal* 850 (1 Nov. 2017), p. L7. issn: 20418213. doi: 10.3847/2041-8213/aa9704.

[56] K. Liestbl, P. K. Andersen, and U. Andersen. “Survival analysis and neural nets”. en. In: *Statistics in Medicine* 13.12 (1994), pp. 1189–1200. issn: 1097-0258. doi: 10.1002/sim.4780131202.

[57] Y.-H. Lo, K. Karlsson, and C. J. Kuo. “Applications of organoids for cancer biology and precision medicine”. In: *Nature Cancer* 1 (8 2020), pp. 761–773. issn: 26621347. doi: 10.1038/s43018-020-0102-y.

[58] T. Mancini, H. F. Calvo-Pardo, and J. Olmo. “Prediction intervals for Deep Neural Networks”. In: *ArXiv* abs/2010.04044 (2020).

[59] D. R. Mani et al. “Statistics and data mining techniques for lifetime value modeling”. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’99. New York, NY, USA: Association for Computing Machinery, 1999, pp. 94–103. isbn: 978-1-58113-143-7. doi: 10.1145/312129.312205.

[60] M. Margulies et al. “Genome Sequencing in Microfabricated High-Density Picolitre Reactors”. In: *Nature* 437 (Oct. 2005), pp. 376–80. doi: 10.1038/nature03959.

[61] C. Molnar, G. Casalicchio, and B. Bischl. “Interpretable Machine Learning – A Brief History, State-of-the-Art and Challenges”. In: *ECML PKDD 2020 Workshops*. Springer International Publishing, 2020, pp. 417–431. doi: 10.1007/978-3-030-65965-3_28.

[62] K. P. Murphy. “Machine learning - a probabilistic perspective”. In: *Adaptive computation and machine learning series*. 2012.

[63] W. Nelson. “Hazard Plotting for Incomplete Failure Data”. In: *Journal of Quality Technology* 1.1 (1969), pp. 27–52. doi: 10.1080/00224065.1969.11980344.

[64] R Peto. “Statistical aspects of cancer trials”. In: *Treatment of cancer* (1982), pp. 867–871.

[65] K. L. Pogue-Geile et al. “Predicting degree of benefit from adjuvant trastuzumab in NSABP trial B-31”. In: *Journal of the National Cancer Institute* 105 (23 Dec. 2013), pp. 1782–1788. issn: 00278874. doi: 10.1093/jnci/djt321.

[66] P. Probst, A.-L. Boulesteix, and B. Bischl. “Tunability: Importance of hyperparameters of machine learning algorithms”. In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 1934–1965.

[67] V. C. Raykar et al. “On Ranking in Survival Analysis: Bounds on the Concordance Index”. In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. NIPS’07. Red Hook, NY, USA: Curran Associates Inc., 2007, 1209–1216. isbn: 9781605603520. doi: 10.5555/2981562.2981714.

[68] E. Roblin, P.-H. Cournede, and S. Michiels. “On the Use of Neural Networks with Censored Time-to-Event Data”. In: *Mathematical and Computational Oncology*. Ed. by G. Bebis et al. Cham: Springer International Publishing, 2020, pp. 56–67. isbn: 978-3-030-64511-3. doi: 10.1007/978-3-030-64511-3_6.

[69] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”. In: *Psychological Review* 65 (1958), pp. 386–408.

[70] P. M. Rothwell. “Treating individuals 2: Subgroup analysis in randomised controlled trials: Importance, indications, and interpretation”. In: *Lancet* 365 (9454 Jan. 2005), pp. 176–186. issn: 01406736. doi: 10.1016/S0140-6736(05)17709-5.

[71] P. Royston and W. Sauerbrei. “Interaction of treatment with a continuous variable: Simulation study of significance level for several methods of analysis”. In: *Statistics in Medicine* 32 (22 Sept. 2013), pp. 3788–3803. issn: 02776715. doi: 10.1002/sim.5813.

[72] D. B. Rubin. “Estimating causal effects of treatments in randomized and nonrandomized studies.” In: *Journal of Educational Psychology* 66 (1974), pp. 688–701.

[73] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536.

[74] F. Sanger, S. Nicklen, and A. R. Coulson. “DNA sequencing with chain-terminating inhibitors”. In: *Proceedings of the National Academy of Sciences* 74.12 (1977), pp. 5463–5467. doi: 10.1073/pnas.74.12.5463. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.74.12.5463.

[75] M. Schena et al. “Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray”. In: *Science* 270.5235 (1995), pp. 467–470. doi: 10.1126/science.270.5235.467. eprint: https://www.science.org/doi/pdf/10.1126/science.270.5235.467.

[76] N. Simidjievski et al. “Variational Autoencoders for Cancer Data Integration: Design Principles and Computational Practice”. In: *Frontiers in Genetics* 10 (2019). issn: 1664-8021. doi: 10.3389/fgene.2019.01205.

[77] R. Singh and K. Mukhopadhyay. “Survival analysis in clinical trials: Basics and must know areas”. In: *Perspectives in clinical research* 2 (Oct. 2011), pp. 145–8. doi: 10.4103/2229-3485.86872.

[78] W. N. Street. “A Neural Network Model for Prognostic Prediction”. In: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML ’98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 540–546. isbn: 978-1-55860-556-5.

[79] N. Ternès, F. Rotolo, and S. Michiels. “Biospear: An R package for biomarker selection in penalized Cox regression”. In: *Bioinformatics* 34 (1 Jan. 2018), pp. 112–113. issn: 14602059. doi: 10.1093/bioinformatics/btx560.

[80] N. Ternès et al. “Identification of biomarker-by-treatment interactions in randomized clinical trials with survival outcomes and high-dimensional spaces”. In: *Biometrical Journal* 59.4 (2017), pp. 685–701. doi: https://doi.org/10.1002/bimj.201500234. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bimj.201500234.

[81] R. Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. issn: 0035-9246.

[82] R. Tibshirani. “The Lasso Method for Variable Selection in the Cox Model”. en. In: *Statistics in Medicine* 16.4 (1997), pp. 385–395. issn: 1097-0258. doi: 10.1002/(SICI)1097-0258(19970228)16:4<385::AID-SIM380>3.0.CO;2-3.

[83] T. Tieleman and G. Hinton. *Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude.* 2012.

[84] H. Uno et al. “On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data”. In: *Statistics in Medicine* 30 (10 May 2011), pp. 1105–1117. issn: 02776715. doi: 10.1002/sim.4154.

[85] L. van ’t Veer et al. “Gene expression profiling predicts clinical outcome of breast cancer”. In: *Nature* 415 (Feb. 2002), pp. 530–6. doi: 10.1038/415530a.

[86] J. Venter et al. “The Sequence of the Human Genome”. In: *Science* 291 (Feb. 2001), pp. 1304–1351.

[87] M. N. Wright, A. Ziegler, and I. R. König. “Do little interactions get lost in dark random forests?” In: *BMC bioinformatics* 17 (Mar. 2016), p. 145. issn: 14712105. doi: 10.1186/s12859-016-0995-8.

[88] L. Yates et al. “The European Society for Medical Oncology (ESMO) Precision Medicine Glossary”. In: *Annals of Oncology* 29.1 (2018). Focus on liquid bicopsy, pp. 30–35. issn: 0923-7534. doi: https://doi.org/10.1093/annonc/mdx707.

[89] L. Zhao and D. Feng. “DNNSurv: Deep Neural Networks for Survival Analysis Using Pseudo Values”. In: (2019), pp. 1–15.

[90] H. Zou and T. Hastie. “Regularization and Variable Selection via the Elastic Net”. In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320. issn: 1369-7412.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Cancérologie : biologie - médecine - santé (CBMS)

Titre: Développements des réseaux de neurones pour données censurées: prédiction de survie associée à une mesure d'incertitude et recommandation de traitement individualisée.....
Mots clés: Analyse de survie, Réseaux de neurones, Incertitude, Recommandation de traitement

Résumé : En oncologie, l'analyse des données de survie a des applications multiples : analyse de la survie d'un groupe de patients, comparaison des effets de différents traitements, etc. Les données de survie présentent la caractéristique de ne pas être observées complètement, l'événement d'intérêt n'étant pas nécessairement survenu pour tous les patients à la fin de la période d'observation : les données sont dites censurées. Dans les modèles pronostiques, on utilise traditionnellement un modèle à risques proportionnels de Cox pour analyser les données de survie. Ce modèle repose sur des hypothèses restrictives qui ne permettent pas de prendre en compte facilement l'analyse d'un nombre toujours croissant de facteurs pronostiques et prédictifs, ni d'inclure toutes leurs interactions ou leurs effets non linéaires. De nouveaux modèles s'appuyant sur les réseaux de neurones artificiels peuvent être employés pour faire face à ces limites. L'objectif de cette thèse est d'utiliser ces modèles et de les adapter au contexte des données censurées. Dans un premier temps, nous implémentons des modèles de réseaux de neurones permettant d'effectuer des prédictions de survie. Nous comparons des méthodes existantes à nos méthodes proposées et construites à l'aide de pseudo-observations. Dans un second temps, nous établissons une mesure de l'incertitude des prédictions réalisées avec ces réseaux de neurones. Elles prennent la forme d'intervalles de confiance. Enfin, nous mettons en place des recommandations de traitement individualisées.

Title: Survival Prediction using Artificial Neural Networks on Censored Data.....
Keywords: Survival analysis, Neural networks, Uncertainty, Treatment recommendation

Abstract: In oncology, analyzing survival data is of primary importance in epidemiological studies and clinical research. The most commonly used approach to deal with the censored observations present in this type of study is the Cox proportional hazards model. The model is based on particular assumptions which may be hard to assume for a high number of candidate covariates to evaluate (proportional hazards, linearity of effects, interactions between covariates). Recently a new family of models, Artificial Neural Networks, has been proposed to model complex patterns and prediction problems, and may be used in the context of survival prediction. They have been widely used in different areas of science. This Ph.D. aims to provide new methods of survival analysis using neural networks. The objective of the first part of the Ph.D. is to compare existing neural network models adapted to survival with our proposed method based on pseudo observations as a solution to the problem of censored data. The second part focuses on measuring the uncertainty of survival predictions using confidence intervals. Finally, a treatment recommendation rule based on a neural network is set up in order to provide treatment recommendations based on the patient's prognostic features.