Neural networks for survival analysis and predictive maintenance

Achraf Bennis

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *14/01/2022*
par : **BENNIS Achraf**

Neural Networks for Survival Analysis and Predictive Maintenance

JURY

Mathieu Serrurier	Directeur de thèse
Sandrine Mouysset	Co-encadrante de thèse
Antoine Cornuéjols	Rapporteur
Yohann Foucher	Rapporteur
Jean-Marc Alliot	Examinateur
Sébastien Travadel	Examinateur - Président du Jury

École doctorale et spécialité :
Ecole Doctorale Mathématiques Informatique et Télécommunications de Toulouse (EDMITT)
Informatique et Télécommunications
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (IRIT)
Directeur de Thèse :
Mathieu Serrurier
Co-encadrante de Thèse :
Sandrine Mouysset
Rapporteurs :
Antoine Cornuéjols et *Yohann Foucher*

Résumé

Cette thèse se focalise sur le problème de l'analyse de survie via une approche d'apprentissage profond. L'objectif principal est d'estimer le risque d'un événement donné à l'échelle individuelle, puis de proposer une solution de maintenance pour minimiser ce risque compte tenu d'un budget limité. Nous utilisons la distribution de Weibull qui est particulièrement populaire et fréquemment utilisée dans l'analyse de survie car elle est adéquate pour modéliser le temps jusqu'à l'événement dans un cadre réel en utilisant une base de données quelle que soit sa taille, et est suffisamment flexible en raison de l'ensemble de formes, diverses et variées, de distribution déterminées par ses paramètres de forme et d'échelle. Cependant, la présence d'échantillons censurés (à droite) est fréquente dans les données de survie, et les ignorer induit un biais significatif dans l'estimation du risque.

Pour résoudre ce problème, nous étudions, dans ce travail, le problème de l'estimation du risque qu'un événement d'intérêt se produise chez un individu. Nous proposons, en premier lieu, une approche d'apprentissage profond, DeepWeiSurv, en supposant que la distribution temps-événement sous-jacente peut être modélisée par un mélange fini de distributions de Weibull dont les paramètres respectifs sont à estimer par le réseau. Nous présentons et décrivons l'architecture de ce réseau et la fonction de perte, avec laquelle il est entraîné, qui prend en compte les données censurées à droite. Des expériences sur des ensembles de données synthétiques et réelles montrent que cette approche offre une meilleure performance prédictive que les méthodes de l'état de l'art.

Cependant, la performance de ce modèle dépend de la taille du mélange qui est décrite comme un paramètre du modèle et cela peut être problématique dans un cadre réel.Pour résoudre ce problème, nous proposons une nouvelle approche, DPWTE, décrite comme une version étendue de DeepWeiSurv avec pratiquement la même architecture, qui ne fixe pas la taille du mélange mais fixe plutôt une limite supérieure (de la taille du mélange) suffisamment grande et trouve la combinaison optimale (en termes de taille, de paramètres et de coefficients de pondération) de distributions de Weibull pour modéliser la distribution temps-événement sous-jacente. Pour ce faire, nous introduisons une couche de multiplication par éléments, que nous appelons la couche *Sparse Weibull Mixture*, qui sélectionne par ses poids les distributions de Weibull qui ont une contribution significative à la modélisation de la distribution temps-événement. Pour stimuler ce processus de sélection, nous appliquons une régularisation *sparse* (en utilisant la norme $\ell_{0.5}$) sur cette couche en ajoutant un terme de pénalité à la fonction de perte. Nous validons ce modèle sur des ensembles de données simulées et réelles, en montrant qu'il permet d'améliorer les performances par rapport à DeepWeiSurv et aux méthodes les plus connues et performantes de l'état de l'art.

Par la suite, nous proposons une solution pour minimiser le risque détecté par ces deux approches tout en respectant les contraintes budgétaires. Pour ce faire, nous considérons un problème d'optimisation sous contrainte (budgétaire) qui consiste à minimiser la probabilité de risque d'un sujet donné à partir des données de survie, en supposant que chaque

modification de caractéristique (contrôlable) a un coût donné. Les contraintes de budget/coût étant modélisées par une boule ℓ_1 pondérée, nous proposons de résoudre ce problème numériquement en utilisant l'algorithme de *Descente du Gradient Projeté*. Nous considérons trois scénarios pour la fonction de risque de probabilité : la boîte noire pour laquelle nous utilisons LIME, la boîte semi-blanche, et la boîte blanche complète que nous rendons robuste contre l'instabilité numérique en utilisant une technique de régularisation du gradient. Nous réalisons une série d'expériences synthétiques pour évaluer les performances de la méthode dans trois scénarios différents et montrons que la version robuste de la boîte blanche complète obtient des résultats prometteurs et encourageants. Ce problème principalement motivé par l'analyse de survie, ou plus précisément la maintenance prédictive peut dépasser ce contexte et être posé pour tout problème de régression où le but est d'optimiser la sortie de la régression d'un individu donné sous des contraintes de boules (pondérées) ℓ_1.

Enfin, nous présentons et décrivons le projet *SmartOccitania* qui peut être considéré comme une application complète de l'analyse de survie dans le monde réel dont le but est de prédire le risque qu'une alimentation basse tension subisse une panne de courant, où nous testons et évaluons nos deux approches et proposons des solutions de maintenance.

Abstract

This work focuses on the problem of survival analysis via a deep learning approach. The main goal is to estimate the risk of an event of particular interest at the individual and then propose a maintenance solution to minimize this risk given a limited budget. We use the Weibull distribution which is particularly popular and frequently used in survival analysis since it is adequate for modeling the time-to-event of real-world events with small or large data, and is sufficiently flexible due to the range of distribution shapes determined by its shape and scale parameters. However, the presence of (right) censored samples is frequent in survival data, and ignoring them induces a significant bias in the risk estimation.

To address this problem, we investigate, in this work, the problem of estimating the risk of an individual to experience an event of interest. We propose, in the first place, a deep-learning approach, DeepWeiSurv, assuming that the underlying time-to-event distribution can be modeled by a finite mixture of Weibull distributions whose respective parameters are to be estimated by the network. We present and describe the architecture of this network and the loss function, with which it is trained, that takes into account the right-censored data. Experiments on synthetic and real-world datasets show that this approach offers better predictive performance than state-of-the-art methods.

However, the performance of this model depends on the size of the mixture which is described as a model parameter and this may be problematic in a real-world problem. To resolve this, we propose a novel approach, DPWTE, described as an extended version of DeepWeiSurv with practically the same architecture, that does not fix the size of the mixture but rather fixes an upper bound (of the mixture size) sufficiently large and finds the optimal combination (in terms of size, parameters, and weighting coefficients) of Weibull distributions to model the underlying time-to-event distribution. To accomplish this, we introduce an element-wise multiplication layer, which we call the *Sparse Weibull Mixture* layer, which selects through its weights the Weibull distributions that have a significant contribution to the time-to-event distribution modeling. To stimulate this selection process, we apply a sparse regularization (using $\ell_{0.5}$ norm) on this layer by adding a penalty term to the loss function. We validate this model on both simulated and real-world datasets, showing that it yields a performance improvement over DeepWeiSurv and the most known/performant state-of-the-art methods.

Afterward, we propose a solution to minimize the risk detected by these two approaches while respecting budgetary constraints. To do this, we consider a (budget) constrained optimization problem that consists of minimizing the probability risk of a given subject from the survival data, assuming that each (controllable) feature modification has a given cost. With the budget/cost constraints being modeled by a weighted ℓ_1 ball, we propose to solve this problem numerically using the *Projected Gradient Descent* algorithm. We consider three scenarios for the probability risk function: the black box for which we use LIME, the semi-white box, and the full-white box that we render robust against numerical instability using

a gradient regularization technique. We conduct series of synthetic experiments to evaluate the performance of the method in three different scenarios and show that the robust version of the full-white box achieves promising and encouraging results. This problem principally motivated by survival analysis, or more precisely the predictive maintenance can go beyond this context and be raised for any regression problem where the goal is to optimize the regression output of a given individual under (weighted) ℓ_1 ball constraints.

Finally, we present and describe the *SmartOccitania* project which can be described as a complete real-world application of survival analysis whose goal is to predict the risk that a low-voltage feeder experiences a power failure, where we test and evaluate our two approaches and propose maintenance solutions.

Contents

Introduction

The research work presented in this thesis has been carried out within the scope of *SmartOccitania*[1] project supported by *ADEME*[2] and *ENEDIS*[3] in cooperation with the Research Institute of Computer Science of Toulouse (IRIT) from the University of Toulouse III Paul Sabatier. The current work focuses on developing new deep-learning approaches for survival analysis using Weibull distributions and then proposes a maintenance solution under budget constraints using constrained optimization.

Survival analysis is a sub-field of statistics [1] where the goal is to analyze and model a type of data namely the so-called *survival data*, where the outcome is the time until experiencing an *event of interest*. More precisely, the main objective of survival analysis is to estimate the time of occurring an event of interest [2, 3, 1, 4]. One of the main challenges in this type of statistical analysis is the presence of instances whose event outcomes are non-observed during the monitoring period [5, 6, 7]. Such a phenomenon is called *censoring* and the concerned instances are described as *censored* observations which can be effectively handled using survival analysis techniques. Arguably, survival analysis has focused on interpretability, potentially at some cost of predictive accuracy which is the reason why binary classifiers based on machine learning are commonly used in applications in different areas where survival methods are applicable [8]. However, while the machine-learning-based classifiers can output predictions for one pre-determined duration, one loses interpretability and flexibility provided by the survival methodology that models the event time probabilities as a function of time. Furthermore, the binary classifiers typically ignore the censoring phenomenon [9, 10, 8], and therefore the use of survival models tends to be advantageous, if not essential.

0.1 Context

In many areas, clinical or industrial, some professionals have a question in the mind as to how much time will take for an event to happen. For instance:

- In a manufacturing plant, engineers and technicians are faced with the problem of estimating the time remaining until failure of a mechanical system or electronic devices to schedule an early maintenance [11, 12].
- Customer churn or attrition [13], i.e. *the percentage of customers that stop using a company's products or services* is one of the most important metrics for a business as it usually costs more to acquire new customers than it does to retain existing ones.

[1] https://www.enedis.fr/sites/default/files/field/documents/cp-smart-occitania-certifie.pdf
[2] https://www.ademe.fr/sites/default/files/assets/documents/smart_occitania_-_fiche_laureat.pdf
[3] https://www.enedis.fr/actualites/reseau-electrique-intelligent-en-milieu-rural-le-projet-smart-occitania-devoile-resultats

- Credit risk [14] refers to the likelihood that a borrower will not be able to repay a loan contracted by a lender. Therefore, financial institutions are under a duty to quantify that probability to limit their exposure.
- Employees retention [15] which costs a lot in terms of money, time to acquire the expertise (loss of knowledge when attrition happens).
- In the health field, doctors, sometimes, are led to conduct a clinical trial on a group of patients and then analyze the risk of death given their features and medical history [16].

These problems can be considered and resolved by survival analysis, also known as *Time-to-Event Analysis* (see Table 1). This branch of statistics that emerged in the 20^{th} century [17], is heavily used in industrial world [11, 12], economics and finance [18, 19], insurance [20], marketing [21], health field [16] and many more application areas [13, 14, 22, 23, 24, 25]. It is used to analyze and predict when an *event* can occur. An event in this context can be manifold since the application areas range from medicine to industry. We call *population* the group of subjects under the study, e.g. patients, mechanical systems, electronic devices, employees of a company, loaners, customers, etc. Furthermore, the Survival Analysis study needs to define a time frame in which this study is carried out. As in many cases, the given time period for the event to occur may be the same as each other. In the real world, some elements of the population on which the study is conducted do not get affected by the event of study before the end of the time frame. These subjects are described as *right censored* [5]. To recap, in a survival analysis study, we need to define the event to be observed and must have three main components for each element of the population: *time recorded, status and features* [26]. The time recorded is the *survival time* (also known as *time-to-event data*) for an *observed*, i.e. non-censored, subject and an underestimation of the survival time for the right-censored one. The variable *status* supplies with the information of whether the subject of study is right-censored or not. This triplet is what we call *survival data* or *time-to-event data* in this field. The real strength of survival analysis is its capacity to take into account the right-censored data in the event distribution modeling.

0.2 Questions Considered in Survival Analysis

In this section, we briefly describe a series of questions that should be considered when analyzing survival data.

0.2.1 Description

- *What is unique about survival data?* Survival data has a unique format since the outcome of interest has two key components: whether or not the event occurred and when if so [26]. Traditional methods of logistic and linear regression are not suited to be able to include both the event and time aspects as the outcome in the model [27, 28]. Traditional regression methods also are not equipped to handle censoring [9, 8] where the true time to event is underestimated. Special techniques for survival data, as will be discussed in the next chapter, have been developed [29, 30, 31, 32] to utilize the partial information on each subject with censored data and provide unbiased survival estimates.
- *What are important methodological considerations of survival data?* There are four main methodological considerations in the analysis of survival data. It is important

Table 1: Contribution of survival analysis for each case study described above (inspired by [4]).

Case Study	Subject	Event	Contribution
Predictive maintenance in a manufacturing plant.	Mechanical System or Electronic Device	Failure	Predict when the failure can happen and thus schedule the maintenance.
Customer Churn /Attrition	Customer	Churn	Predict when customers stop doing business, to find a specific strategy for customer retention.
Credit Risk	Loaner	Repayment	Predict the speed of repayment of a loan. This will help to mitigate losses due to bad debt, customize interest rates, etc.
Employee Retention	Employee	Retention	Estimates employee turnover, by predicting when an employee will quit.
Clinical Trial	Patient	Death	Estimates the lifespan of the patient/population.

to have a clear definition of the target event, the time origin, the time scale, and to describe how subjects will exit the study [26]. Once these are well-defined, then the analysis becomes more straightforward. Typically there is a single target event, but there are extensions of survival analyses that allow for multiple events [30].

- *What is the question of interest?* The choice of analytical tool should be guided by the research question of interest. With survival data, the research question can take several forms, that influence which function is the most relevant to the research question. Three different types of research questions that may be of interest for survival data include:

  1. What proportion of individuals will remain free of the event after a certain time?
  2. What proportion of individuals will have the event after a certain time?
  3. What is the risk of the event at a particular point in time, among those who have survived until that point?

  Each of these questions corresponds with a different type of function used in survival analysis:

  1. Survival Function, $S(t)$: the probability that an individual will survive beyond time t.
  2. Probability Density Function (PDF), $f(t)$, or the Cumulative Incidence Function (CIF), F(t): the probability that an individual will have a survival time less than or equal to t.

3. Hazard Function, $h(t)$: the instantaneous risk of experiencing an event at time t, conditional on having survived to that time.

4. Cumulative Hazard Function (CHF), $H(t)$: the integral of the hazard function from time 0 to time t, which equals the area under the curve $h(t)$ between time 0 and time t.

If one of these functions is known, the other ones can be calculated through the relationship between them, we will see this in more detail in Chapter Background.

- *What assumptions must be made to use standard techniques for survival data?* the main assumption in analyzing survival data is that of non-informative censoring [33]: individuals that are censored have the same probability of experiencing a subsequent event as individuals that remain in the study. Because the analysis will otherwise be biased [33, 5, 34]. There is no definitive way to test whether censoring is non-informative, though exploring patterns of censoring may indicate whether an assumption of non-informative censoring is reasonable. If informative censoring is suspected, sensitivity analyses, such as best-case and worst-case scenarios, can be used to try to quantify the effect that informative censoring has on the analysis.

0.3 Research Questions

This thesis focuses, in large part, on the problem of estimating the probability of occurring an event in the population of the study. The main research questions, that are being addressed, are as follows:

- How to model the relationship between the subject features (also known as *covariates*) and *survival times* at the individual level?

- How to take into account the right-censored sub-population in this modeling?

- What are the assumptions about the distribution of survival times that are the most appropriate to the real-world case studies?

For the second part of the thesis, we will devote it to the problem of optimizing the subject's survival time. In this context, we consider that the subject risk is already identified and we need to find the optimal way to maximize its mean lifetime. Several research questions were investigated:

- In the real-world setting, there is a limitation on the budget dedicated to maintenance, and some features could not be modified. Thus, the question is: how to take into account these two constraints?

- How to model mathematically this problem?

- What is the most appropriate technique that should be used to solve this problem?

- How to evaluate the performance of this technique in the real-world survival data?

0.4 Contributions

The work in this thesis focuses on developing methods for addressing the challenges raised in the two parts of the thesis described above: survival probability estimation and optimization of a mean lifetime at the individual scale. Regarding the first part, the main contributions are as follows:

- **DeepWeiSurv** [35]: A deep learning approach for survival analysis with censored data using Weibull Distributions. This model assumes that the survival times are samples drawn from a finite mixture of Weibull distributions whose parameters are to be estimated. The Weibull distribution is one of the most widely used lifetime distributions in survival analysis, as it is known to be able to correctly model the time-to-event of real-world data and flexible despite having only two parameters. A mixture of Weibull distributions can only be better. Since the model has a neural network basis, it does need to make any assumption on the nature of the relationship between the covariates and survival times, which guarantees free modeling of this relationship. DeepWeiSurv is consisted of two sub-networks and has three output layers (one *softmax* [36] and two ELU [37] layers). DeepWeiSurv is trained by minimizing a likelihood-based loss that takes into account the right-censored data. After training DeepWeiSurv, we obtain the estimated parameters of this mixture with which we can estimate the survival function S, the probability density function f, hazard function h, and cumulative hazard function H. This model can also estimate the mean lifetime unlike the other survival models cited and described in Chapter 1. This advantage enables DeepWeiSurv to be considered as a mean lifetime function of a subject of study. We will see this in the second part of the thesis.

- **DPWTE** [38] that stands for Deep Parsimonious Weibull Time-to-Event is considered as an extension of DeepWeiSurv. DPWTE has the same architecture as DeepWeiSurv, which means that it maintains the same assumptions and outputs the mixture parameters but the only main difference is that we add the so-called *Sparse Weibull Mixture* (SWM) layer after the softmax layer. The motivation behind this is to filter non-significant Weibull distributions, or in other words, to select, among the p ($p < \infty$ since it is a finite mixture) Weibull distributions, the ones which are significant in terms of contribution to distribution modeling, that is, whose weights in the mixture are relatively important. This means that we compose, through this layer, an optimal mixture of $q < p$ Weibull distributions selected from the initial set of p Weibull distributions. To stimulate this process, we penalize the SWM weights by adding a customized *regularization term* in the loss function which enforces the sparsity (hence the word *Parsimonious*) of the so-called Sparse Weibull Mixture layer.

Regarding the second part, namely the optimization of the mean lifetime (also known as expectation, expectancy, or lifespan), we make the following contribution:

- Predictive maintenance monitors the condition of an element in a system to reduce the likelihood of an unwanted event: failures, death, etc. This problem has drawn much attention with machine learning and especially the deep learning community. However, researchers are mainly focused on the problem of predicting the risk of experiencing the event under study. Here, we are rather interested in the problem of maximizing the life expectancy of a subject of interest given a certain budget. In this work, we consider a system defined by a set of features in which some of them are controllable (i,e. whose respective values can be modified) and the given associated regression problem

(network-based). The problem consists of maximizing the regression output under a budget constraint, assuming that each (controllable) feature modification has a given cost. This problem can be modeled as a constrained optimization problem where the budget constraint is a weighted ℓ_1 ball. We propose to solve this problem numerically using the *Projected Gradient Descent* algorithm. We consider three scenarios for the regression model: black-box where we use LIME [39] to estimate the gradients, a semi-white box in which architecture and the weights are known, and finally a full-white box when the model can be trained and, in this case, we introduce the gradient penalization in the training process to improve the robustness of the full-white box network.

0.5 Thesis Outline

This thesis is divided into four parts. The first one is dedicated to the basic concepts and definitions in survival analysis as well as the state of the art. In the second part, we present the two novel network-based approaches for survival analysis namely DeepWeiSurv and its extended approach DPWTE. We also dedicate a chapter in this part to experiments on real-world benchmark datasets. The third part presents a lifetime maximization method under budget constraints. In the fourth and last part, we describe the SmartOccitania project as a complete real-world application of the three approaches presented in the second and third parts. In this section, we will briefly describe the content of each part in this thesis.

- In the first part of this thesis, which only contains Chapter 1, we first describe the mathematical background required to understand the main problem considered in survival analysis. We will define the special features of survival analysis notably survival data, censored data, survival and hazard functions, and the likelihood function in the presence of censored data. We will also explain why traditional regression methods are not suitable for survival data and censored data, and show the advantage of survival analysis methods over the traditional statistical ones. The predictive performance scores, used to evaluate the survival analysis models, are presented and described as well in this chapter. Then, we present the state of the art of survival analysis. We start by describing the statistical methods namely non-parametric, semi-parametric and parametric approaches. We will move to the machine-learning-based approaches where statistical methods and machine learning were combined to improve the learning of the relationship between data and the event of interest.

- In the second part, we present DeepWeiSurv and DPWTE and describe experiments on real-world benchmark datasets. This part entails three chapters:

  - Chapter 2 presents a novel network-based approach for survival analysis, the so-called DeepWeiSurv, which models the event distribution with a finite mixture of Weibull distributions whose respective parameters are learned by the model network. In this chapter, we describe this approach as well as different simulated experiments conducted to test and evaluate some of its theoretical properties and its capacity to handle censoring at high levels.
  - Chapter 3 presents DPWTE as not only another network-based approach but also as an extension of DeepWeiSurv. This approach, unlike DeepWeiSurv, does not fix the size of the mixture with which we model the underlying distribution, but it rather fixes an upper bound of the number of Weibull, that can be used and tries to find the optimal mixture whose composing Weibull distributions have a significant contribution in the event-time distribution modeling. This model has an extra

layer considered as a special feature that selects these Weibull distributions. As in Chapter 1, we run different simulated experiments to evaluate this approach in terms of some mathematical properties as well as its capacity to handle highly censored settings.

  - In Chapter 4, we run experiments on real-world benchmark datasets to evaluate these two approaches, compare them and establish a comparison between both models and the most known competing methods. We also present another application of these approaches on simulated time-series data where the goal is to predict the remaining useful life of an engine. In this experiment, we compare a standard regression network to both DeepWeiSurv and DPWTE and show the out-performance of the latter.

- In the third part, we present our contribution for lifetime maximization (or more generally regression output maximization) under budget constraints. Chapter 5, describes the problem that we consider and make some assumptions to resolve it. To do this, we propose to resolve this problem by applying the projected gradient descend algorithm using the weighted ℓ_1-norm projection to maintain the budget constraints. We propose three types of network-based models described as regression networks, depending on the level of access to the network granted to us, namely the black, semi-, and full-white boxes. We use LIME to model the black box function and apply a gradient regularization technique to 'robustify' the full white box since we have access to its weights and gradients. We run simulated experiments to evaluate this method with these three defined scenarios. We also run an experiment on the real-world Wine Quality dataset (as an application of the regression output maximization with the real-world dataset) and discuss the different solutions provided to optimize the quality of a given wine.

- In the last part, we describe the SmartOccitania project, on which we worked for two years to fund my thesis, as a comprehensive application since on the one hand, we apply our two survival analysis models to predict the failure risk of a given feeder (subject under study) and on the other hand, we propose using our approach described in Chapter 5, a maintenance solution with respect to the budget. Chapter 6, in the first part, describes the survival analysis study conducted and presents the different results provided by the considered models (including ours). In the second part of this chapter, we propose different solutions, according to the budget dedicated to maintenance, to extend the lifetime of the most-at-risk (detected via DeepWeiSurv and DPWTE predictions) feeders.

Chapter 1

Survival Analysis

1.1 Introduction

Due to the development of various big data acquisition technologies, different disciplines have been interested in collecting a wide variety of data and monitor the observation over periods [4]. For the majority of the real-world applications, the main objective of collecting and monitoring this data is to estimate the time of occurring an event of interest [2, 3, 1, 4]. One of the main challenges present in such data, which we call survival data, is that usually there exist the so-called censored instances [5, 6, 7]. This means that the event of interest is not experienced by these instances, at least during the observation period, or more precisely, certain instances for which the event has occurred but the information about the specific time is only available until the end of the study. This challenge is the reason why it is not suitable to directly use classical statistical techniques or machine-learning-based predictive methods to analyze survival data [8]. Survival analysis provides a myriad of mechanisms to handle censored data problems [6]. Therefore, many works are realized for this purpose, combining survival analysis technique with machine learning especially deep learning frameworks in recent years [32, 31, 30, 40, 41, 42, 43, 44]. In this chapter, we will first review some necessary mathematical background used in survival analysis techniques, then we will describe the state-of-the-art methods and related works done for this purpose. We recall the basic notions on which such branch is based, in order to generate an intuitive understanding. The first thing to note is that *risk, survival, probability* mean the same thing in this context. In fact, we measure the risk or survival by probability. Second, survival relates to the membership in what we call *population*. This means that a population consists of a number of subjects and the survival times or probability is associated with each subject from this population composing what we call *survival data* [26, 2]. However, this membership is not constant. The change of membership is induced by an event of interest. The most common examples for a potential event are: death [16], mechanical system failure [11], crime [45], customer churn [13] and many other applications [22, 23, 24, 25]. Importantly, survival analysis originated from medical research [46] but is not limited to this application area and can also be applied to problems in engineering, marketing, finance, etc.

This chapter is organized as follows: in Section 1.2, we will give a brief review of the basic concepts and definitions that are necessary to understand of what consists survival analysis. Section 1.4 is dedicated to the evaluation metrics for survival models. Then, we describe the traditional statistical methods including non-parametric, semi-parametric and parametric models in Section 1.5. After that, we describe, in Section 1.6, several basic machine learning

approaches, including survival trees, ensemble learning methods, and network-based survival models, and summarize the state-of-the-art methods developed for survival analysis. Finally, we conclude this chapter in Section 1.7.

1.2 Definition of Survival Analysis

In this section, we first describe what survival data is and what makes it different from other types of data (Section 1.2.1). Then, we present a definition of censoring or censored data and explain what makes this phenomenon challenging and with which concept we can model censored data.

1.2.1 What is Survival Data?

Survival data is a term used for data measuring the time to a particular event of interest. It is defined as a set of individuals under an event study. These individuals are characterized by a set of features also known as covariates whose principal outcome in this context are the respective times recorded within the study. In the simplest case, *the event is death* [26, 47, 4], but the term also covers other events such as occurrence of a disease or a complication. This term is extended to other areas, for example in industrial applications, it can typically represent the time to failure of a unit or some of its components, while in economics, it can be time to acceptance of a job offer for an unemployed person, etc. The event can be described as a transition from one state to another (e.g., death is a transition from the state alive to the state dead). Depending on the context, we use words like *death, failure* or *event* to cover the same subject of interest namely what happens to the (concerned) individual or population at the response time.

In broad terms, what makes survival data special is that the outcomes are times and thus are not measured in the same way as other variables in regression problems. Generally, these usual outcome variables are measured almost instantaneously. Whereas, time is observed sequentially and this can lead to know the most common and important phenomenon in survival analysis namely *censoring* [6, 5, 7], that is, the presence of censored data. This phenomenon is what makes survival analysis approaches special and advantageous over traditional classifiers and regression models, where the latter does not handle and ignore this phenomenon leading to an eventual severe bias in the survival estimation [48].

1.2.2 Censored Data

Censored data means that the observations are not totally known and the unknown ones provide only an underestimation of their real survival time [5, 7]. This means that the censored individuals (whose events are not observed) provide only partial information about the time to the event. A common possible reason behind this phenomenon is that the individual under study has not experienced the event of interest when the study is evaluated, and this is all what is known about it. The presence of censored data is clearly a major technical challenge for modeling the underlying event distribution [2, 5, 7]. In this work, as in almost all survival data cases, censoring is right-censoring, i.e., observations are known to be larger than some given value. For instance, a patient has a censored survival time if the event has not yet occurred for this patient. To avoid confusion, let make clear one point with an example: *Patient 'A' has the time recorded t_A, if t_A is greater than the cut-off time, then we are talking about the censored time t_A and the censored Patient 'A*. We mean, by this example, that the word 'censored' can be used for both the subject and the time associated

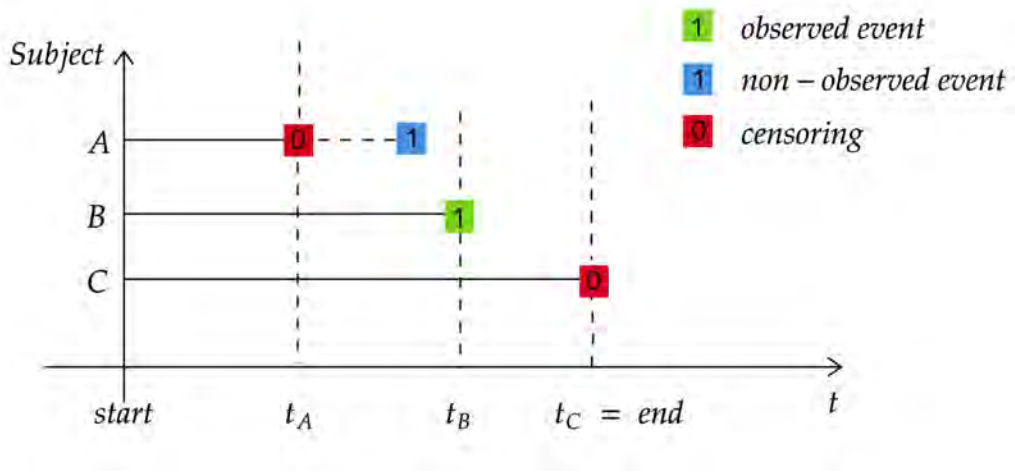


Figure 1.1: A visualization illustrating the meaning of the right-censoring (inspired from [47])

with the survival data. In this thesis, we focused on the right censoring setting. Here some cases where this patient is considered as right censored:

- The patient stops following an examination.
- The patient withdraws from the study.
- His associated event occurs after the time of study closure.

In Figure 1.1, we illustrate the phenomenon of censoring. For instance, subject 'B' experiences the event within the duration of the study unlike subject 'C' which experiences the event after the end of the study. Thus, this event is not observed by the study and the only observable information that we have about it is that, at the end of the study, the subject 'C' did not yet experience the event. We say, in this case, that 'C' is a censored subject. For subject 'A', we have a censored survival time as well, but for a different reason because, in this case, the study did not end yet. Formally, one calls the censoring for subject 'C' fixed and for the subject 'A' random, right-censoring. One important statement to point out in this context is that the occurrence of censoring must be unrelated to the future expectation [26]. This means that the probability density function of the residual lifetime for those censored must be exactly similar to that of those which are not censored and having the same distribution of covariates. To combine expressions for events and right-censored observations, it is common to use an event indicator variable [49, 50] that checks if the event is observed (returning one) or censored (zero otherwise). Survival data with censoring phenomenon seems to be complicated to analyze which is not the case. This is because the distribution likelihood function provides information quantified by terms corresponding to what we observe and thus, if the event is observed then it contributes with its density, or only with the probability that the time-to-event variable exceeds t if the event is censored at t [33]. We see this in detail in Section 1.2.4.

1.2.3 Problem Statement

Let $\mathbf{X} = \{(\mathbf{x}_i, t_i, \delta_i)_{i=1,..,N}\}$ denote a survival data containing N instances, each of them is represented by a triplet (x_i, t_i, δ_i), where :

- $\mathbf{x}_i \in \mathbb{R}^d$ is the feature vector,
- δ_i is the event indicator (or the so-called *status* as mentioned in the introduction of this thesis) which is equal to 1 if i^{th} instance is non-censored and 0 otherwise [50],
- and finally $t_i = min(y_i, c_i)$ which denotes the recorded time which is described as the survival time for non-censored (y_i) or censored time (c_i) otherwise.

Typically, we are unable to observe the variable of interest namely the survival times $y_1, .., y_N$ but we instead observe $(t_1, \delta_1), .., (t_N, \delta_N)$, where δ_i stands for the event indicator which can then be expressed as follows:

$$\delta_i = \begin{cases} 1 & \text{if } y_i \leq c_i \\ 0 & \text{otherwise.} \end{cases} \tag{1.2.1}$$

The main objective of survival analysis is thus to estimate the time-to-event $\hat{y}_j$ or the probability risk $\hat{p}_j$ of experiencing the event, for a particular instance of interest j given its feature values $\mathbf{x}_j$ [4]. It is highlighted that, in the survival analysis problem, the time variable is both continuous and non-negative [1]. For this purpose, we use survival and hazard functions considered as two crucial quantities in survival analysis that we will describe in Section 1.2.4.

1.2.4 Survival and Hazard Functions: General Characteristics

Let T be a non-negative random variable representing the times recorded for the survival data including the waiting time until experiencing an event for non-censored data and censoring time for censored data. We assume that T is a continuous random variable with probability density (p.d.f) function $f_T(t)$ and cumulative distribution function (c.d.f) $F_T(t) = \int_{-\infty}^{t} f_T(u)du = P(T \leq t)$. From now on, we will work with the complement of F, the survival function defined as follows:

$$S_T(t) = 1 - F_T(t) = \int_t^{\infty} f_T(u)du \tag{1.2.2}$$

which expresses the probability that the event of interest has not occurred by duration t. The function S_T has the following properties:

- S_T is defined on $\mathbb{R}^+$,
- S_T is non-increasing,
- $S_T(0) = 1$, i.e., the probability of surviving at the origin time is 1,
- $S_T(\infty) = 0$ since, by definition, $\lim_{t\to\infty} F_T(t) = \infty$.

The hazard function h_T, also known as instantaneous rate of occurrence of the event, can be described as an alternative function to characterize the distribution of T and considered as a crucial quantity of interest in survival analysis [51]. h_T is defined as

$$h_T(t) = \lim_{\delta t \to 0} \frac{P(t \leq T < t + \delta t | T \geq t)}{\delta t}. \tag{1.2.3}$$

The hazard function is described as 'instantaneous' because only subjects with $T \geq t$ and $T < t + \delta t$ for $\delta t \to 0$ are considered. It has the following properties:

[1] https://data.princeton.edu/wws509/notes/c7.pdf

- h_T is non-negative,
- h_T has no upper bound,
- $h_T(t) = 0$ means that there is no event happened in δt.

There is an important relation between h_T, f_T and S_T, and this means that they are not independent from each other. This relation, starting by the definition of h_T, can be derived as follows:

$$h_T(t) = \lim_{\delta t \to 0} \frac{P(t \leq T < t + \delta t | T \geq t)}{\delta t}. \tag{1.2.4}$$

Using the property of a conditional probability:

$$h_T(t) = \lim_{\delta t \to 0} \frac{P(t \leq T < t + \delta t)}{P(T \geq t)\delta t}. \tag{1.2.5}$$

and, by definition of S_T, we obtain:

$$h_T(t) = \lim_{\delta t \to 0} \frac{P(t \leq T < t + \delta t)}{S_T(t)\delta t}, \tag{1.2.6}$$

$$= \frac{f_T(t)}{S_T(t)}. \tag{1.2.7}$$

Note from Equation (1.2.2) that we can suggest rewriting Equation (1.2.7) as

$$h_T(t) = -\frac{d}{dt} log S_T(t). \tag{1.2.8}$$

By integrating this equation from 0 to t and by using the property $S_T(0) = 1$, the survival function S_T can be written as:

$$S_T(t) = exp\left\{ - \int_0^t h_T(u) du \right\}. \tag{1.2.9}$$

Hence, we obtain a formula for the probability of surviving to duration t as a function of the hazard at all durations up to t. These results show that the survival and hazard functions provide an alternative but equivalent characterizations of the distribution of T. The difference between them can be summarized as: the survival function focuses on surviving while the hazard one focuses on experiencing the event. Figure 1.2 shows the relationship between these functions.

1.2.5 Likelihood Function for Censored Data

Let n be the size of a population with lifetimes drawn from T governed by a survival function S_T with associated p.d.f denoted by f_T and hazard function h_T. t_i is the time during which the subject i is observed. If the subject i died at t_i, its contribution to the likelihood function [33] is

$$\mathcal{L}_i = f_T(t_i) = S_T(t_i) h_T(t_i). \tag{1.2.10}$$

However, if the subject is still alive at t_i, the only information that we have about this subject is that its lifetime exceeds t_i, under non-informative censoring [33, 5]. The contribution to the likelihood of the censored subject i, in this case is:

$$\mathcal{L}_i = S_T(t_i). \tag{1.2.11}$$

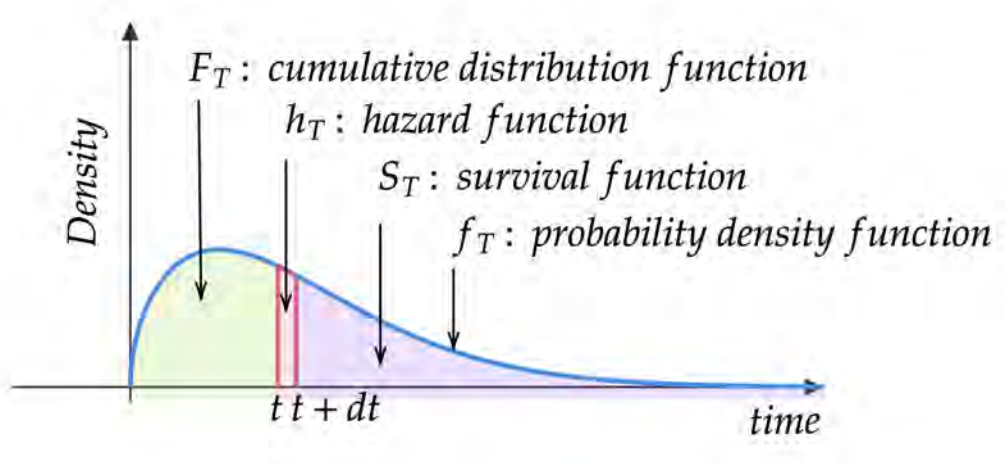


Figure 1.2: Relationship among different entities: f_T (blue curve), F_T (green shade), S_T (purple shade) and h_T (red shade).

One can notice that $S_T(t_i)$ contributes to the likelihood in both cases, which is obvious since the subject lived up to time t_i whether it is censored or observed. Therefore, the likelihood of the underlying distribution, denoted by $\mathcal{L}$, can be written as follows:

$$\mathcal{L} = \prod_{i=1}^{n} \mathcal{L}_i \tag{1.2.12}$$

$$= \prod_{i=1}^{n} S_T(t_i)^{\delta_i} h_T(t_i)^{\delta_i} \; S_T(t_i)^{1-\delta_i} \tag{1.2.13}$$

$$= \prod_{i=1}^{n} h_T(t_i)^{\delta_i} S_T(t_i) \tag{1.2.14}$$

Thus, the log-likelihood function for censored survival data can be expressed as:

$$log\ \mathcal{L} = \sum_{i=1}^{n} \delta_i log\ h_T(t_i) + log\ S_T(t_i) \tag{1.2.15}$$

1.2.6 Expectation of Life

By definition, the expectation value of T, denoted by μ, is given by:

$$\mu = \mathbb{E}(T) = \int_0^\infty t f_T(t) dt. \tag{1.2.16}$$

Integrating by parts, and using Equation (1.2.2) making use of the fact that -f_T is the derivative of S_T with the two last properties of S_T, namely $S_T(0) = 1$ and $S_T(\infty) = 0$, one can show that:

$$\mu = \int_0^\infty S_T(t) dt. \tag{1.2.17}$$

In other words, the survival function can be used to obtain the mean life expectation.

1.3 Why Traditional Regression Methods Are Not Suitable for Survival Data?

Survival data is unique because the variable of interest is not only whether or not an event occurred, but also when that event occurred. Traditional methods of logistic and linear regression are not fitted to take into account both the event of interest and time aspects as the predictand in the model. Traditional regression methods also are not equipped to handle censoring, whose occurrence biases the estimation of the true survival time. In fact, the censoring phenomenon results in survival residuals that are not quite normally distributed or at least as straightforward as they are in linear regression, mainly because, in the presence of censoring, some values of survival times are not observed and thus unknown, which yield a skewed residual distribution.

1.3.1 Naive Survival Estimators Yield Bias while Ignoring Censored Data

To illustrate how ignoring censored data will result in biased results, we propose to analyze three 'naive' estimators for the survival function while only taking into account the observed data (t_i, δ_i) [34]. Let use the notation used so far, with T is the random variable of observed times, and let Y, C denote the survival time and censoring variables respectively. We also note F_Y (resp. F_C) the cumulative distribution function of Y (resp. C) and denote by S_Y the survival function of Y. These naive estimators are defined as follows:

$$\hat{S}_T^1(t) := \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{t_i>t} \qquad\qquad \hat{S}_T^2(t) := \frac{1}{N}\sum_{i=1}^{N} \delta_i \mathbb{1}_{t_i>t}. \tag{1.3.1}$$

The first naive estimator is obtained by calculating the empirical cumulative survival function directly from all the observations. The second one is calculated as the empirical cumulative survival function over the non-censored observations. $\hat{S}_T^1(t)$ is a consistent estimator of $\mathbb{P}(T > t)$, that is, $\lim_{N\to\infty} \hat{S}_T^1(t) = \mathbb{P}(T > t)$. Under the independent censoring assumption [49] that states that Y is independent of C, we have :

$$\mathbb{P}(T > t) = \mathbb{P}(Y > t, C > t) = S_Y(t)(1 - F_C(t)). \tag{1.3.2}$$

This means that $\mathbb{P}(T > t) \geq S_Y(t)\ \forall t \geq 0$. We can notice that for smaller t, the bias is small as there is a lower risk to get censored contrary to larger values of t where the bias is increasing until reaching a maximum and then decreases to converge to 0 at ∞.

The second estimator $\hat{S}_T^2(t)$ is consistent of $\mathbb{P}(T > t, \prod_i \delta_i = 1)$. Censoring times are considered here as missing values, and thus we focus only on the observed survival times $(t_i, \delta_i = 1)$. This clearly deteriorates the sample size of the estimator and implies that one should have :

$$\mathbb{P}\left(T > t, \prod_i \delta_i = 1\right) = \mathbb{P}(Y > t). \tag{1.3.3}$$

However, using Fubini's theorem

$$\mathbb{P}\left(T > t, \prod_i \delta_i = 1\right) = \mathbb{P}(Y > t) = \int\int \mathbb{1}_{u>t}\mathbb{1}_{u\geq v}\mathrm{d}Y(u)\mathrm{d}C(v) \tag{1.3.4}$$

$$= \int \mathbb{1}_{u>t}\left(\int \mathbb{1}_{u\geq v}\mathrm{d}C(v)\right)\mathrm{d}Y(u) \tag{1.3.5}$$

$$= \int_t^\infty (1 - F_C(u))\mathrm{d}Y(u) = S_Y(t) - \int_t^\infty F_C \mathrm{d}F_Y \tag{1.3.6}$$

$$\leq \mathbb{P}(Y > t),\ \forall t \geq 0. \tag{1.3.7}$$

We note that, for $t = 0$, we obtain:

$$\mathbb{P}\left(\prod_i \delta_i = 1\right) = 1 - \int_0^\infty F_C \mathrm{d}F_Y. \tag{1.3.8}$$

This means that the estimator is already biased for $t = 0$ by the probability of being censored expressed by the quantity $\int_0^\infty F_C dF_Y$.

1.3.2 Biased Mean Squared Error of an Estimator

Let assume that $(t_1, \delta_1), .., (t_N, \delta_N)$ observed samples drawn from a single exponential distribution $Exp(\lambda)$. We choose this distribution for two simple reasons: first, exponential distribution is among the most widely used distribution in survival analysis; second, it has only one parameter and thus easy for calculations. Under the assumption of constant censoring times [52, 5], the likelihood can be written as follows

$$\mathcal{L}_\lambda = \prod_i h_T^{\delta_i}(t_i) S_T(t_i). \tag{1.3.9}$$

Since, for exponential distribution, $h_T = \lambda$ and $S_T(t) = exp(-\lambda t)$, then:

$$\mathcal{L}_\lambda = \lambda^{n_c} exp(-\lambda \sum_i t_i). \tag{1.3.10}$$

where $n_c = \sum_i \delta_i$. We linearize Equation (1.3.10) with the log function and consider the following optimization problem:

$$\max_\lambda log\ \mathcal{L}_\lambda. \tag{1.3.11}$$

Using the maximum likelihood estimation method, the solution to Problem 1.3.11, denoted by $\tilde{\lambda}$, should satisfy the following condition:

$$\frac{\mathrm{d}\ log(\mathcal{L}_\lambda)}{\mathrm{d}\ \lambda}(\tilde{\lambda}) = \frac{n_c}{\tilde{\lambda}} - \sum_i t_i = 0, \tag{1.3.12}$$

$$\implies \tilde{\lambda} = \frac{n_c}{\sum_i t_i}. \tag{1.3.13}$$

We thus obtain the mean survival time $\tilde{\mu}$:

$$\tilde{\mu} = \frac{1}{\tilde{\lambda}} = \frac{1}{n_c}\sum_i t_i. \tag{1.3.14}$$

Let denote the usual estimator for the mean by $\hat{\mu}$ and define it as the simple average of t_i's:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} t_i. \tag{1.3.15}$$

The mean squared error of $\hat{\mu}$ with respect to $\tilde{\mu}$, denoted by $MSE(\hat{\mu}, \tilde{\mu})$, can be calculated as follows:

$$\begin{aligned} MSE(\hat{\mu}, \tilde{\mu}) &= \mathbb{E}\left[(\hat{\mu} - \tilde{\mu})^2\right], && (1.3.16)\\ &= \mathbb{E}[\hat{\mu}^2] + \tilde{\mu}^2 - 2\tilde{\mu}\mathbb{E}[\hat{\mu}], && (1.3.17)\\ &= var(\hat{\mu}) + (\mathbb{E}[\hat{\mu}] - \tilde{\mu})^2. && (1.3.18) \end{aligned}$$

where $\mathbb{E}[z]$ denotes the expectation of z. The term $b = (\mathbb{E}[\hat{\mu}] - \tilde{\mu})^2$ quantifies the squared bias of $\hat{\mu}$. Since we have: $\mathbb{E}[\hat{\mu}] = \frac{n_c}{N}\tilde{\mu}$, then: $b = (\frac{n_c}{N} - 1)\tilde{\mu}$. Therefore, the estimator is unbiased when all the survival times are observed ($n_c = N$), otherwise the bias increases with the size of censored data (large biases for higher n_c's).

1.4 Predictive Performance Metrics

To evaluate and compare the predictive performance of all the methods considered in this experiment, we used two evaluation metrics that account for the censored individuals. In the following, we describe the two metrics: Concordance Index and Brier Score.

1.4.1 Concordance Index

In survival analysis, the concordance index or C-index [53] is one of the most commonly applied evaluation metrics. C-index is designed to calculate the number of *concordant* pairs of observations among all the *comparable* pairs, i.e. pairs of non-censored observations. In other words, C-index is described as the probability that, for a random pair of individuals, the predicted survival times of a pair of two individuals have the same ordering as their true survival times. Aside from the fact that C-index accounts for censoring, this metric can be interpreted as a misclassification probability and does not depend on a single fixed time for evaluation [54]. The concordance index is calculated using the following steps:

1. Generate all possible pairs of individuals over the dataset.

2. Remove the pairs whose shorter survival time is censored and those which have equal times unless at least one is non-censored.

3. For each comparable pair with different event times, count 1 if the shorter survival time has worse predicted output.

4. C-index is defined by the sum of counts against the total number of comparable pairs.

As the concordance index only depends on the ordering of the predictions, it is useful for evaluating linear-risk models since the ordering of these models does not change over time. However, it is not the case here, or at least for our two proposed approaches. In fact, the C-index cannot be applied for non-linear-risk models [55, 54]. Here, we will rather use a metric based on the time-dependent concordance index, denoted by C^{td}, proposed in [56]

which estimates the probability that a pair of individuals are concordant given they are comparable. This metric is defined as follows:

$$C^{td} = P(\hat{S}(t_i) > \hat{S}(t_j)|\ t_i > t_j \textbf{ and } j \text{ is non censored}) \tag{1.4.1}$$

$$= P(\hat{t}_i > \hat{t}_j|\ t_i > t_j \textbf{ and } \delta_j = 1) \tag{1.4.2}$$

$$= \frac{\sum_{i,j} \mathbb{1}_{t_i > t_j} . \mathbb{1}_{\hat{t}_i > \hat{t}_j} . \delta_j}{\sum_{i,j} \mathbb{1}_{t_i > t_j} . \delta_j}. \tag{1.4.3}$$

where $\hat{S}$ is the predicted survival function and $\hat{t}_i$ is the predicted time for the i^{th} individual. Similarly to the *Area Under Curve* score [57], $C^{td} = 1$ corresponds to the best model prediction. For proportional hazards models, the metric is equivalent to C-index.

1.4.2 Brier Score

The Brier score [58] noted BS is a metric used in binary classification problems that measures both discrimination and calibration of the estimates. BS is defined as follows:

$$BS = \frac{1}{N} \sum_i y_i - \hat{p}_i$$

where N is the number of observations, $y_i \in \{0, 1\}$ is the label of the i^{th} individual and $\hat{p}_i$ is the probability estimate of $p_i = P(y_i = 1)$. For survival data, we choose a threshold time t to label data according to whether a subject's event time is greater or smaller than t. Graf et al. [59] generalize this metric to take into account the censored events. Using the formula proposed in [60], BS can be written as follows:

$$BS(t) = \frac{1}{N} \sum_i \frac{\hat{S}(t|x_i)^2 \mathbb{1}_{t_i \leq t} \mathbb{1}_{\delta_i = 1}}{KM(t_i)} + \frac{(1 - \hat{S}(t|x_i))^2 \mathbb{1}_{t_i > t}}{KM(t)} \tag{1.4.4}$$

where $KM(t)$ is the Kaplan-Meier [61] estimate (described in detail in the next section) of the probability that t is greater than the censoring threshold. The generalized form of BS can be extended to an integrated BS that calculates the score for an interval:

$$IBS = \frac{\int_{t_0}^{t_1} BS(u)du}{t_1 - t_0} \tag{1.4.5}$$

IBS is calculated, in practice, by numerical integration.

1.5 Statistical Methods for Survival Analysis

In this section, we will introduce three main traditional statistical approaches to estimate the survival or hazard functions namely: non-parametric, semi-parametric, and parametric methods. In general, more than one approach can be correctly used in the same analysis.

1.5.1 Non-parametric Approaches

Non-parametric approaches do not rely on assumptions about the shape of the underlying distribution of the survival time. In this context, they are used to describe the survival data

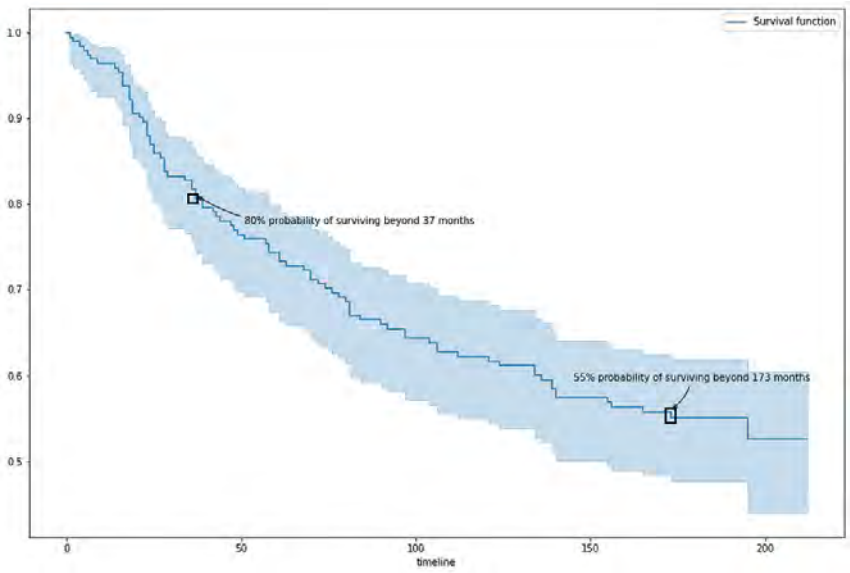


Figure 1.3: Survival function of a SEER Breast Cancer population sample (https://seer.cancer.gov/) estimated by Kaplan-Meier estimator [61].

by estimating the survival function, along with the median and quartiles of survival time. However, for censored subjects from the concerned population, these descriptive statistics cannot be calculated directly from the data, which underestimates the real survival time, leading to skewed estimates of these statistics. Non-parametric models are, in general, used as the first step in survival analysis works, in order to generate unbiased statistics. Let's describe the most known non-parametric approaches in survival analysis.

1.5.1.1 Kaplan-Meier Estimator

The most common non-parametric approach and among the first survival estimator in the literature is the Kaplan-Meier model [61]. The survival function estimator is given by:

$$\hat{S}_{KM}(t) = \prod_{t_i<t} \frac{n_i - d_i}{n_i}, \mathrm{t} > 0 \tag{1.5.1}$$

which depends only on two variables, namely n_i that denotes the subjects known to have survived up to time t_i and d_i the number of events experiences at time t_i. The main assumptions of this method, besides the random censoring, is that censoring occurs after experiencing the event and that there is no cohort effect on survival, i.e. no variation of the characteristics of the distribution over time, so subjects have the same survival probability regardless of when they came under study. In Figure 1.3, we visualize an example of Kaplan-Meier estimation of the survival function of a SEER Breast Cancer population sample [62]. SEER is a program that provides cancer incidence data from population-based cancer registries covering approximately 34.6% of the U.S. population. The estimated survival function from the Kaplan-Meier method can be plotted as a step-wise function with time on the X-axis, as shown in Figure 1.3. According to this figure, the model estimated for instance, that 80% of this population survive beyond 37 months, while almost half (55%) survive beyond 173 months. This plot can be used to estimate the median or quartiles of survival time.

1.5.1.2 Nelson-Aelen Estimator

Nelson-Aelen estimator [63], noted NAe, is considered as an alternative to Kaplan-Meier method [61], which is based on using a counting process approach to estimate the cumulative hazard function, H_T. The NAe estimator of the cumulative hazard function is given by:

$$H_T^{NAe}(t) = \sum_{t_i \leq t} \frac{d_i}{n_i} \tag{1.5.2}$$

which implies the definition of the NAe estimator of the survival function:

$$S_T^{NAe}(t) = exp\left(-H_T^{NAe}(t)\right) = exp\left(-\sum_{t_i \leq t} \frac{d_i}{n_i}\right) \tag{1.5.3}$$

As we see, the NAe estimator [64, 63] is an indirect estimator for the survival function unlike the Kaplan-Meier estimator.

A non-parametric model to the analysis of survival data is used to simply describe the distribution of survival times. More commonly, we are rather interested in the relationship between the subject's features/covariates and the time to the event which is not feasible with this approach. To do so, we use the semi- and fully-parametric approaches that allow the survival time to be analyzed with respect to the features. Let's start by describing the semi-parametric approach.

1.5.2 Semi-Parametric Approach Cox Proportional Hazard Model

A large family of models introduced by Cox [29] focuses directly on the hazard function. The simplest member of the family is the *Cox Proportional Hazards* Model [29], noted CPH, where the hazard at time t for a subject of covariates $\mathbf{x} \in \mathbb{R}^d$ is assumed to be defined as:

$$h_T(t|\mathbf{x}) = h_0(t)\ exp(\mathbf{x}^T\beta) \tag{1.5.4}$$

where $h_0(t)$ is called the baseline hazard. CPH is considered a semi-parametric because the model is consisted of a non-parametric component h_0 and a parametric one namely $exp(\mathbf{x}^T\beta)$. The baseline hazard serves as the 'default' hazard when the covariates are not taken into account, or $\mathbf{x} = 0$, and $exp(\mathbf{x}^T\beta)$ is the relative risk associate with the covariates $\mathbf{x}$. The variation of the relative risk is the same at all duration t. In this approach, no assumption about the baseline hazard h_0 is made. However, the model assumes:

- *Time independence of the covariates* $\mathbf{x}$, which means that there is a constant relationship between the survival time and $\mathbf{x}$.
- *Linearity in the covariates* $\mathbf{x}$: the implication of the first assumption is that the hazard function for any two subjects are proportional at any point in time.
- *Hazard proportionality*: the second assumption implies that the hazard curves for two subjects should be proportional.

In many situations, one is interested by the coefficients β_i rather than the shape of h_T. In order to obtain this, we apply the logarithm on the hazard ratio given by (Equation 1.5.4):

$$log\ \frac{h_T(t|\mathbf{x})}{h_0(t)} = \sum_i \beta_i x_i \tag{1.5.5}$$

which leads us to solve a problem of linear regression.

1.5.2.1 Regularized Cox Models

For high-dimensional data, it sometimes happens that the size of the covariate set is almost equal to or even greater than the size of the data. This makes building the prediction model with all the features a challenging task since the model might provide inaccurate results due to overfitting [65]. This problem motivates using sparsity techniques to select the most important features while assuming that most of them do not have a significant contribution in terms of prediction [66]. To do this, different penalty functions are used to develop prediction models using sparse learning techniques. ℓ_p-norm functions are the commonly used ones. Many researchers have used these techniques in Cox regression methods resulting in the following regularized Cox models:

- *Lasso-Cox*: Lasso [67] is a ℓ_1-norm regularizer known to be good at performing feature selection and estimating the regression coefficients simultaneously. Tibshirani in his work [68], incorporates the ℓ_1-norm regularizer into the log-partial likelihood to obtain the Lasso-Cox model. Several extensions of Lasso-Cox method notably fused Lasso-Cox [69], adaptive Lasso-Cox [70] and graphical Lasso-Cox [71].

- *Ridge-Cox*: Ridge regression was proposed by Hoerl and Kennard in [72] which is defined as ℓ_2-regularized regression model whose goal is to select the correlated features and distribute their values with each other. This regularization technique was incorporated in the context of Cox regression resulting in a ℓ_2-regularized Cox regression model [73].

- *EN-Cox*: Elastic-Net (EN) is a regularized regression method combining ℓ_1 and squared ℓ_2 penalties which have the potential to perform the feature selection and deal with the correlation between the features simultaneously [74]. Noah Simon et al. [75] proposed to introduce the EN penalty term into the log-partial likelihood function resulting in the so-called EN-Cox.

1.5.3 Parametric Approaches

One of the main advantages of using a semi-parametric model is that the baseline hazard does not need to be specified to estimate the hazard ratio that describes differences between groups in terms of relative hazard. However, the estimation of the baseline hazard itself may be of interest. In this case, a parametric approach is necessary and essential, where both the effect of the covariates and the hazard function are specified. This latter is estimated by making an assumption on the distribution of the underlying population. Due to the relation between the hazard and the survival functions, this also makes this latter a parametric model. Some specific models that are the most common and find frequent applications are listed below. For simplicity, a parameter λ defined as a function of $\mathbf{x}$ will be written λ rather than $\lambda(\mathbf{x})$.

1.5.3.1 Exponential Distribution

The exponential distribution assumes that $h_T(t)$ depends only on model coefficients and covariates and is constant over time. For this model, the hazard function, survival function

are given by:

$$h_T(t|\mathbf{x}) = \frac{1}{\eta} \tag{1.5.6}$$

$$S_T(t|\mathbf{x}) = exp\left(-\frac{t}{\eta}\right) \tag{1.5.7}$$

with η is a non-negative function of $\mathbf{x}$, called *rate parameter*. The expected life time is given by:

$$\mu(\mathbf{x}) = \eta. \tag{1.5.8}$$

1.5.3.2 Weibull Distribution

While the exponential distribution assumes a constant hazard, the Weibull distribution assumes a monotonic one that can either be increasing or decreasing. For the Weibull model, the hazard function and survival function are given by:

$$h_T(t|\mathbf{x}) = \left(\frac{\beta}{\eta}\right)^{\beta} \tag{1.5.9}$$

$$S_T(t|\mathbf{x}) = exp\left\{-\left(\frac{\beta}{\eta}\right)^{\beta}\right\} \tag{1.5.10}$$

where η is a non-negative function of $\mathbf{x}$, called *rate parameter* and β the *shape parameter*. h_T of Weibull distribution has the following properties:

- h_T is monotonously decreasing when $\beta < 1$, and monotonously increasing when $p > 1$
- h_T is constant when $\beta = 1$.

The expected life time is given by:

$$\mu(\mathbf{x}) = \eta\, \Gamma\left(1 + \frac{1}{\beta}\right) \tag{1.5.11}$$

with Γ is the Gamma function defined by: $\Gamma(s) = \int_0^{\infty} u^{s-1} exp(-u)du$. We will describe this distribution more in detail in the next chapter, as it is used in our approaches.

1.5.3.3 Log-Normal Model

For this model, h_T, S_T are given by:

$$h_T(t|\mathbf{x}) = \frac{1}{\sigma\sqrt{2\pi}t} exp\left(\frac{-\left(ln(t)-\mu\right)^2}{2\sigma^2}\right)\left(1 - \Phi\left(\frac{ln(t)-\mu}{\sigma}\right)\right)^{-1} \tag{1.5.12}$$

$$S_T(t|\mathbf{x}) = 1 - \Phi\left(\frac{ln(t)-\mu)}{\sigma}\right) \tag{1.5.13}$$

where Φ is is the cumulative distribution function of the standard normal distribution ($\mathcal{N}(0,1)$). The Log-normal has two parameters, since a normal distribution has two parameters $\mu \in \mathbb{R}$ and $\sigma > 0$ (the *mean* and the *standard deviation*, respectively, of the normal distribution associated). The expected life time is defined as follows:

$$\mathbb{E}(T) = exp\left(\mu + \frac{\sigma^2}{2}\right). \tag{1.5.14}$$

1.5.3.4 Log-Logistic Model

The hazard and the survival function for the Log-logistic model are given by the following formulas:

$$h_T(t|\mathbf{x}) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1}\left(1+\left(\frac{t}{\eta}\right)^{\beta}\right)^{-1}, \tag{1.5.15}$$

$$S_T(t|\mathbf{x}) = \left(1+\left(\frac{t}{\eta}\right)^{\beta}\right)^{-1}. \tag{1.5.16}$$

As we see, the Log-logistic model depends on two parameters namely $\beta > 0$ and $\eta > 0$ that denotes respectively the *shape* and the *scale* of this distribution. The behaviour of the hazard function depends on β:

- if $\beta < 1$, then h_T is monotonously decreasing from ∞,
- if $\beta = 1$, then h_T is monotonously decreasing from $\frac{1}{\eta}$,
- if $\beta > 1$, then h_T is concave. In this case: $h_T(t = 0) = 0$, and the maximum value of h_T is $h_T((\beta - 1)^{\frac{1}{\beta}})$.

The expected life time is defined (when $\beta > 1$) as follows:

$$\mathbb{E}(T) = \frac{\pi}{\beta\ \eta\ sin\left(\frac{\pi}{\beta}\right)}. \tag{1.5.17}$$

1.5.3.5 Comparison of the four parametric models: Exponential, Weibull, Log-Normal and Log-Logistic models

We show, in Figure 1.4, the four most common parametric models described above. We can notice that the shape of the hazard function varies from a model to another. The specific behavior of the hazard function decides which parametric model is appropriate for a particular problem. For the Exponential model, the hazard function is a constant, whereas the Weibull model of shape equal to 2, h_T is linearly increasing. We can use this model, for instance, when we analyze an event after an unsuccessful surgery or treatment. For the Log-logistic with $\beta < 1$, the hazard function is monotonous and decreasing, this can be useful in the case of a postoperative state of the patient after surgery or after a stock-market crisis. Regarding the Log-normal distribution, the hazard function has a concave form (slightly humped, in the shape of an inverted U).

1.5.3.6 Accelerated Failure Time

Accelerated Failure Time [76], noted AFT, models are a class of parametric survival models that can be linearized by taking the log of the survival time model. Since T must be non-negative, we might consider modeling its logarithm using a conventional linear model, say

$$log\ T = \mathbf{x}^{\beta} + \epsilon \tag{1.5.18}$$

where ϵ is the error term that follows a specific distribution. This model specifies the distribution of log-survival for a subject as a noised baseline distribution by the error term ϵ.

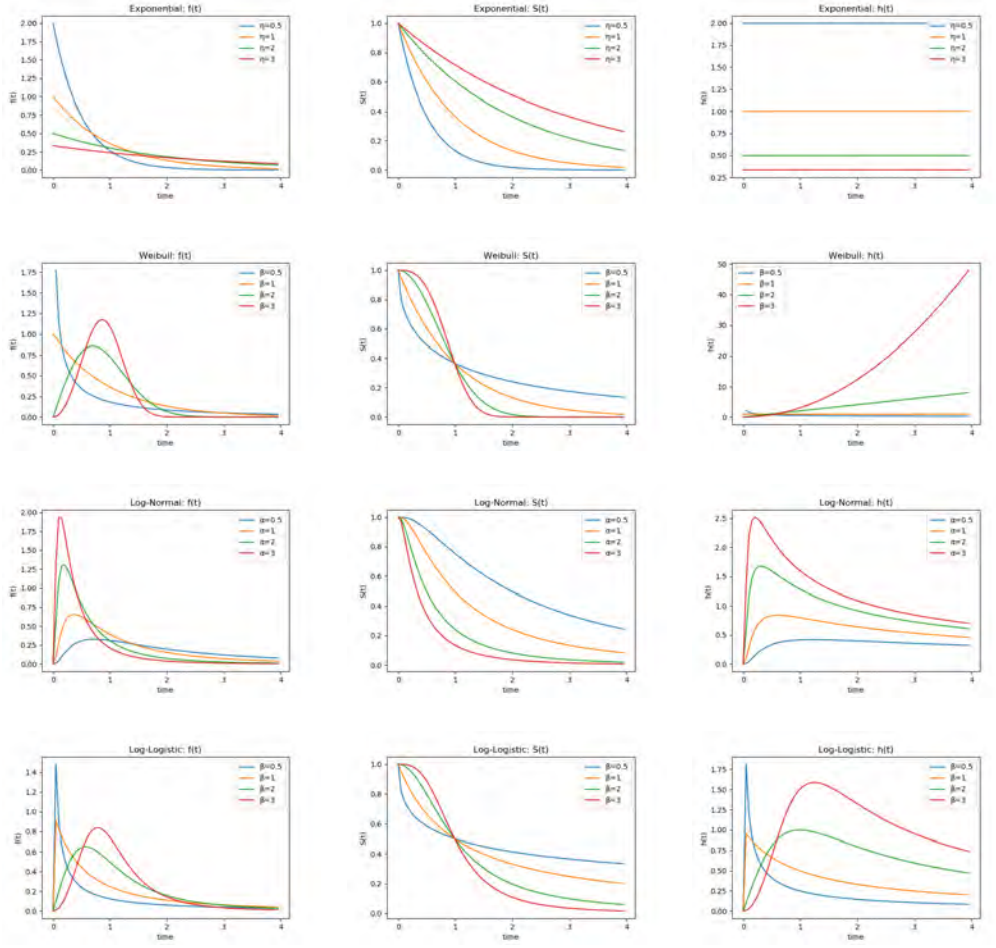


Figure 1.4: Comparison of the following parametric models: Exponential, Weibull, Log-Normal, and Log-Logistic models. For each model, we plot the density, survival, and hazard functions. For the Log-Normal model, the parameter α is the inverse of σ.

The main difference between AFT models and CPH models is that AFT models assume that effects of covariates are multiplicative on the time scale, while Cox models use the hazard scale as shown in Section 1.5.2.

There are many more parametric models notably:

- The *Gombertz distribution* [77] whose density expression is that of the log-Weibull distribution. The log of the hazard function is thus linear in t, hence, Gompertz distribution is a proportional hazard model. This kind of distribution is suitable for actuarial survival data, as the risk increases exponentially over time like the event rate of the Gompertz distribution.
- *Log-Logistic distribution* [78] which is an AFT model with ϵ following the standard logistic distribution. However, it is not considered a proportional hazard model.
- *Generalized Gamma distribution* [79]: a family of distributions that contains nearly all of the most commonly used distribution, including the exponential, Weibull, log-

Table 1.1: Pros and cons of each family of traditional approaches

Type	Advantages	Disadvantages
Non-parametric	-Efficient when no assumption about theoretical distribution.	- Not easy to interpret. - Over- or under-estimate survival time.
Semi-parametric	- Model the relationship between survival times and covariates. - No assumption on the underlying distribution.	- The distribution of the outcome is unknown. - Not easy to interpret
Parametric	- Easy to interpret. - More informative: estimate risk and predict survival time statistics. - Rely on full maximum likelihood to estimate parameters. - More efficient and yields more accurate estimates when parametric form is correctly specified.	- Inconsistent and yields non-optimal results when the distribution assumption is violated. - Not robust to misspecification.

normal, and gamma distribution. This allows comparisons among the different distributions.

- *Splines Approach* [80] that can be used for maximum flexibility in modeling the shape of the baseline hazard h_0, since the only general limitation of the specification of h_0 is that h_T is non-negative with respect time t.

The main advantages of using a parametric approach over non- and semi-parametric approaches are:

- Parametric models are more informative than the two other approaches. In addition to estimating the risk, they can also predict the survival time, hazard rates, mean, median survival times, etc.

- Parametric approaches rely on full maximum likelihood to estimate parameters.

- When the parametric form is correctly specified, parametric models have more power than semi-parametric models. They are also more efficient, leading to smaller standard errors and more precise estimates.

This family of models has also the main disadvantage namely the fact that it relies on the assumption that the underlying population distribution has been correctly specified. This may be risky because they are not robust to misspecification, that is why semi-parametric models are more common in the literature of survival analysis and are less risky to use there is uncertainty about the underlying population distribution [81]. Table 1.1 (inspired by [4]) summarizes both the advantages and disadvantages of each family of statistical survival methods.

1.6 Survival Analysis using Machine Learning

In the past several years, due to the advantages of machine learning techniques, notably their ability to learn non-linear functions and representations of the data and the improved quality of the overall predictions made, significant success has been achieved in various practice areas. In survival analysis, the main challenge of machine learning is to model the relationship between the covariates and survival times (not necessarily linear) while handling the censored information. It should be noted, that machine learning is effective when there are a large number of instances in a reasonable dimensional feature space. In this section, we will do a comprehensive review of commonly used machine learning methods in survival analysis.

1.6.1 Continuous-Time Models: Extension of Cox Model

Among the first network-based approaches for analyzing the survival data is the method of Faraggi and Simon [32] described as an extension of Cox Proportional Hazard model [29] that accommodates right-censored data. Their model is consisted of one hidden layer and estimates the network parameters, that replace the linear function in the Cox approach, by maximizing the likelihood. However, there is no performance improvement provided by this approach. Katzman et al. [31] revisited the Cox model in a deep learning framework, called *DeepSurv* which is a multi-layer perceptron predicting the probability of experiencing the defined event without subjecting itself to the linearity constraint. Katzman et al. [31] showed that their model outperforms CPH model [29] in terms of *concordance index* score [53, 56]. DeepSurv was then used in Zhu et al. [44, 82] works by replacing the multi-layer perceptron architecture of DeepSurv with convolution layers to analyze the pathological images. Other variants and extension of Cox are then proposed by and Yousefi et al. [40] (SurvivalNet, a framework for fitting proportional Cox models with neural networks and Bayesian optimization of the hyperparameters) and Kvamme et al. [60, 8] notably:

- *Cox-Time* [60] where the Cox relative risk function depends not only on the covariates but also on time. It also does not require a proportionality assumption and uses an alternative loss function scaling well non-linear cases to remedy this constraint,
- *CoxCC* [60]: a proportional version of the Cox-Time model,
- *PCHazard* [8] that assumes that the continuous-time hazard function is piece-wise constant.

1.6.2 Discrete-Time Models

An alternative approach to time-to-event prediction is to discretize the duration of the study and compute the hazard function of survival function on this predetermined time grid.

1.6.2.1 DeepHit

Lee et al. [30] use a deep neural network to learn the distribution of the time-to-event data without making any assumption about the form of the underlying distribution. DeepHit has the following properties:

- DeepHit learns the relationship between the covariates and the survival times while taking into account the variation of this relationship over time,

- It smoothly handles a single-type risk as well as competing risks,
- It consists of a single shared sub-network for all the competing risks that computes a latent representation of the data which is then used to feed the cause-specific sub-network of each of the competing risks.

DeepHit is considered as a probability mass function (since it outputs a vector of probabilities and hence the adjective 'discrete') approach with a ranking loss function that can handle the competing risks.

1.6.2.2 Logistic-Hazard

The Logistic-Hazard method parameterizes the discrete hazards and minimizes the survival negative log-likelihood. This approach is also called *Nnet-Survival* by Gensheimer et al. [83] and *Partial Logistic Regression* by Biganzoli et al. [84].

1.6.2.3 N-MTLR: The Neural Multi-Task Logistic Regression

Yu et al. [85] proposed a method called multi-task logistic regression (MTLR), to learn survival distribution at the individual level while taking into account the censored observations. It consists of modeling the survival function by combining multiple local dependent logistic regression models in order to handle the time-varying effects of the covariates. Afterward, another work was done by Fotso et al. [41]. They introduce a new network-based approach using the MTLR model as the base and a deep learning architecture as the core of the proposed model. This model, called N-MTLR, seems to outperform the standard MTLR as well as CPH.

1.6.2.4 BCE

The BCE method is an MLP with a similar network structure as the Logistic-Hazard, with each output node corresponding to a binary classifier at time t. These binary classifiers are constructed by minimizing the binary cross-entropy of the survival estimates at a set of discrete times and discarding the censored individuals. The removal of censored observations makes this approach biased since it underestimates the survival times of the non-observed events.

Other interesting works are done in Survival Analysis with a discrete-time approach. In fact, Luck et al. [42] proposed methods that are similar to DeepSurv [31], but with an additional set of discrete outputs for survival predictions. This approach consists of modeling the survival function instead of estimating the hazard function and jointly predicts the survival time as well as its rank using the Cox partial log-likelihood framework. Furthermore, Martinsson [43] proposed a recurrent network-based method called WTTE-RNN for sequential prediction of the time-to-event for both censored and non-censored events. WTTE-RNN has the main role of estimating the distribution of time of the next event as having a discrete or continuous Weibull distribution whose parameters are to be estimated by a recurrent neural network.

1.6.3 Survival Trees

Survival trees are trees performing regression or classification on survival data. The first advantage of survival trees over standard trees is that they are adapted to handle censored data. We recall that the basic intuition behind tree models is that data are recursively

partitioned based on a particular splitting condition, and elements that are similar to each other based on the event of interest will be placed in the same node. The difference between a survival tree and the standard one is in the choice of splitting criterion. The standard decision tree-based methods perform partitioning on the data by setting a threshold for each feature. However, they can neither consider the interactions between the features nor the censored information in the model. Two approaches are used as splitting criteria for survival trees:

- those that minimize within-node homogeneity using this latter in the loss function to train the method,
- those that maximize within-node heterogeneity.

Several metrics used in previous works to measure the homogeneity or heterogeneity by symmetry, notably Wasserstein metric [86], exponential log-likelihood [87].

1.6.4 Ensemble Learning Models for Survival Analysis

Ensemble learning techniques [88] learn a committee of classifiers that predict the class labels for a given data points by taking a weighted vote among the prediction results from all these classifiers. Breiman proposed bagging [89] and random forests [90], to perform the ensemble-based model building. Such models have been successfully adapted to the survival analysis problem.

1.6.4.1 Random Survival Forests

Random Survival Forests (RSF) proposed by Ishwaran et al. [54], is a random forest-based method for the analysis of the right-censored survival data. RSF computes a random forest [90] using the log-rank test as the splitting condition. Afterward, the model computes the cumulative hazards function of the leaf nodes of the random forest and averages over the ensemble. Below the main steps:

- Draw N bootstrap samples randomly from the survival data,
- for each bootstrap sample, build a survival tree by randomly selecting features and then split the node using the candidate ones that maximize the survival difference between the child nodes,
- build the full-size tree under the constraint that a leaf node must have no less than specific unique death,
- calculates the cumulative hazards function (CHF) for each tree, then average over the tree to obtain the ensemble CHF,
- and finally, calculate the prediction error for the ensemble CHF using the Out-of-Bag data (bootstrap samples).

RSF is known to be a very flexible continuous-time method that is not constrained by the proportionality assumption. Most previous works benchmark their methods against the random survival forests.

1.6.4.2 Bagging Survival Trees

The bagging method is one of the most commonly used ensemble methods, typically known to reduce the variance of the base models that are used. For bagging survival trees [91], the survival function can be calculated by averaging the predictions made by a single survival tree. Below, the three main steps:

- draw B bootstrap sample from the survival data,
- for each bootstrap sample, build the survival tree and guarantee that the number of events is at least equal to the given threshold, for all the leaf nodes,
- average the predictions made by the terminal nodes and then calculate the aggregated survival function. For each leaf node, the survival function is estimated using the Kaplan-Meier [61] estimator. This means that all the instances within the same leaf node are assumed to have the same survival function.

In contrast to aggregation by majority voting or averaging of the predictions in classical regression or classification problems, averaged point predictions are of minor interest in survival analysis. Instead, Hothorn et al. [91] did not aggregate point predictions but rather predict the conditional survival probability function by computing respective single Kaplan-Meier curves for leaves, each of which includes a sub-population of observations.

1.6.5 Summary

Broadly speaking, the survival analysis methods can be classified into two main categories: statistical methods and machine-learning-based methods. These two families of survival methods share the common goal to make predictions of the survival time or estimate the survival probability (in terms of experiencing the event of interest). However, statistical methods focus more on characterizing both the distributions of the event times and the statistical properties of the parameter estimation, while machine-learning-based survival methods focus more on the prediction of event occurrence at a given time point by incorporating these traditional statistical methods in machine learning frameworks such as survival trees and neural networks which takes advantages of the recent development in machine learning and optimization to learn dependencies between covariates and the event times in different ways. In addition, machine learning methods are usually applied to high-dimensional problems, unlike statistical methods. A summarized (but not exhaustive) taxonomy of the survival analysis methods described above, is shown in Figure 1.5.

Kaplan-Meier estimator [61] is considered as among the first estimators widely used for time-to-event prediction, but it doesn't incorporate individual covariates. In the time-to-event analysis that explores the relationship between features and both event and censored times, existing methods assume a linear dependence. The semi-parametric Cox Proportional Hazards [29] (CPH) model assumes the effect of covariates is a fixed and multiplicative covariate-dependent factor on the hazard rate (linear relationship) which may be too simplistic since, in the real-world data, the covariate effects are often non-monotonic.

Thanks to the ability of neural networks to learn nonlinear functions, many researchers tried to model the relationship between the covariates and the time-to-event data. An extension of CPH with neural networks was first proposed by Faraggi and Simon [32] who replaced the linear risk of the Cox regression model, with one hidden layer multi-layer perceptron but without performance improvement. Katzman et al. [31] revisited the Cox model in the framework of deep learning (DeepSurv), which removes the proportionality constraint and showed that it outperforms CPH in terms of concordance index score [53] which measures the

time ordering performance. Cox-Time [60] which is also a Cox extension, does not require this assumption and uses an alternative loss function scaling well non-linear cases to remedy this constraint. Most previous works benchmark their methods against the random survival forests (RSF) [54]. RSF computes a random forest using the log-rank test as the splitting criterion. It computes the cumulative hazards of the leaf nodes and averages them over the ensemble. Hence, RSF is a very flexible continuous-time method that is not constrained by the proportionality assumption. Other previous works are based on Cox regression such as SurvivalNet [40], a network-based model using Bayesian optimization of the hyperparameters, and Zhu et al. [44, 82] who proposed a convolutional neural network that replaces multi-layer perceptron architecture of DeepSurv and applied this methodology to pathological images. An alternative approach to time-to-event prediction is to discretize the duration and compute the hazard or survival function on this predetermined time grid. Lee et al. [30] proposed a method, called DeepHit, that estimates the probability distribution with a neural net and combines the log-likelihood with a ranking loss. Furthermore, the method has the added benefit of being applicable for competing risks. Fotso [41] proposed N-MTLR that used a Multi-Task Regression (MTLR) as the base with a neural network that calculates the survival probabilities on the points of the time grid. Another interesting work proposed by Martinsson [43] in which he presented WTTE-RNN, a model for sequential prediction of time-to-event for censored data whose main role is to estimate the distribution of time to the next event as having a discrete or continuous Weibull distribution with parameters being the output of a recurrent neural network. Unlike these discrete-time models, considered as probability mass functions, DPWTE [38] and DeepWeiSurv [35] model a time-to-event distribution and thus a continuous survival function that enables to estimate of the survival probability at any survival time horizon.

1.7 Conclusion

We presented in this chapter basic concepts of survival analysis and a detailed overview of the main state-of-the-art methods proposed as approaches to survival analysis. We first defined what survival data is and the phenomenon of censoring which makes the survival prediction task more challenging. Then we described the key functions that characterize an underlying time distribution corresponding to an event interest. We also showed how to compute the mean lifetime and the likelihood of distribution while taking into account the censored data. Then, we described the most common evaluation metrics used to assess the predictive performance of the survival methods. After that, we described the baseline statistical models starting with the non-parametric models which are among the first approaches, then the semi-parametric which is the most commonly used one namely Cox, and finally the parametric approaches where particular assumption on the underlying distribution is made. We compared the semi-parametric to parametric models by highlighting the advantages of the latter over semi-parametric approaches as well as the weakness provided by the distribution assumption to parametric models. We also described different categories of machine-learning-based methods namely continuous-time models, discrete-time models, and ensemble learning methods. Finally, we summarized the different contributions provided by the state-of-the-art methods, from non-parametric models to network-based parametric methods.

Figure 1.5: Taxonomy of the methods developed for survival analysis (inspired by [4]).

Chapter 2

Estimation of Conditional Mixture Weibull Distribution with Right-Censored Data using Neural Network for Time-to-Event Analysis

2.1 Introduction

In this chapter, we propose a novel approach based on neural networks for survival analysis with right-censored data. The model that we propose consists of the estimation of two-parameter Weibull distribution conditionally to the features. For this purpose, we describe the neural network architecture minimizing a loss function that takes into account the right-censored events in its modeling. We extend our network-based approach to a finite mixture of two-parameter Weibull distributions. We validate our model via synthetic experiments which will be carefully described. The main goals of experiments on simulated datasets are:

- *Assess the ability of the model to estimate the parameters of the conditional Weibull distributions with an acceptable accuracy.* More generally, the goal here is to verify if this network-based approach succeeds in modeling the relationship between the survival times and the covariates at the individual scale.

- *Check if the model behaves correctly in the case of a highly censoring setting*, i.e. when there is a significant portion of censored observations in the population or even when they make up the big majority of the population under study.

The second point permits us to show that our approach can consider any survival time horizons. This means that, once the network, which is the core of our model, is trained, we can estimate the survival function or density at different time t separated in time, even for the censored subjects.

In our novel approach, which we call *DeepWeiSurv*, we make the one and only assumption: we assume that the survival times distribution is modeled according to a finite mixture

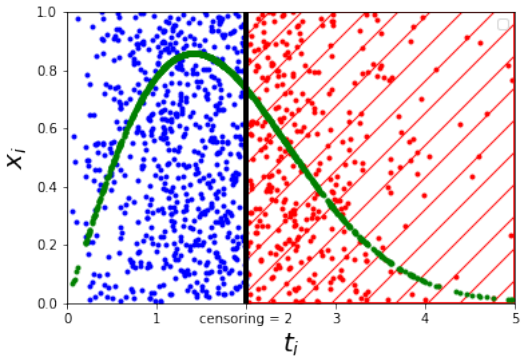


Figure 2.1: Weibull distribution right-censored at $t_c = 2$ with $x \in [0, 1]$ uniformly distributed. In this figure, the parameters of the law are independent with regard to x.

of Weibull distribution (at least one), whose parameters depend on the features, for a right-censored data. As Luck et al. [42], we propose a deep learning model that learns the survival function, but we will do this by estimating the Weibull's parameters. Unlike DeepHit [30], whose model consists in discretizing the time considering a predefined maximum time horizon, here, as we try to estimate the parameters, we can model a continuous survival function and thus estimate the risk at any given survival time horizon. To do so, we constructed a fully connected deep network model and a loss function to be minimized, which is the negative log-likelihood of the finite mixture of Weibull distributions taking into account the censored individuals.

This chapter is organized as follows. In Section 2.2, we discuss the necessary background of the survival analysis and mixture of Weibull distributions. In Section 2.3, we describe the architecture of DeepWeiSurv and the loss function used to train the model. Section 2.4 is dedicated to all the synthetic experiments that we conducted to evaluate the approach in different aspects namely: the ability to estimate the likelihood (Experiment 1) and more specifically, the parameters of the mixture (Experiment 2), the ability to model the relationship between parameters and baseline data covariates (Experiment 2), the ability to estimate the weighting coefficients (clustering problem in Experiment 3) and finally if the model can handle highly censoring setting (Experiment 4 where we consider two scenarios with different level of difficulty: uni-modal mixture and bi-modal mixture). We conclude in Section 2.5.

2.2 Weibull Mixture Distribution for Survival Analysis

In this section, we briefly review some basics in survival analysis and Weibull distributions.

2.2.1 Survival Analysis with Right-Censored Data

Let $X = \{(\mathbf{x}_i, t_i | 1 \leq i \leq n\}$ be a set of observations with $\mathbf{x}_i \in \mathbb{R}^d$, the i^{th} observation of the baseline data (covariates), $t_i \in \mathbb{R}$ its time recorded associated, and δ denotes the event indicator function defined as follows:

$$\begin{array}{rcccl} \delta & : & \mathbb{R}^{*+} & \to & \{0,1\} \\ & & t_i & \mapsto & \begin{cases} 0 & \text{if the event experienced by } i \text{ is censored,} \\ 1 & \text{if the event experienced by } i \text{ is observed.} \end{cases} \end{array} \tag{2.2.1}$$

For simplicity, we note $\delta_i := \delta(t_i)$. As shown in Figure 2.1, a blue point represents a non-censored observation $(\mathbf{x}_i, t_i, \delta_i = 1)$ and a red point represents a censored observation $(\mathbf{x}_i, t_i, \delta_i = 0)$. In order to characterize the distribution of the survival times $T = (t_i|\mathbf{x}_i)_{i \leq n}$, the aim is to estimate, for each observation, the probability that the event occurs after or at a certain survival time horizon t_{sth} defined by:

$$S(t_i|\mathbf{x}_i) = P(t_i \geq t_{sth}|\mathbf{x}_i).$$

Note that, t_{sth} may be different to the censoring threshold time t_c. We recall that it exists an alternative characterization of the distribution of T is given by the hazard function h that is defined as the event rate at time t conditional on survival at time t or beyond. Literature [92] has shown that h can be expressed as follows: $h(t) = \frac{f(t)}{S(t)}$, $f(t)$ being the density function.

Instead of estimating the survival value $S(t_i|\mathbf{x}_i)$, it is rather common to estimate directly the expectation of the survival time $\mathbb{E}(\hat{t}_i|\mathbf{x}_i)$.

2.2.2 Weibull Distribution for Censored Data

From now, we consider that the survival time variable T follows a finite mixture of two-parameter Weibull distributions, or at least a single Weibull distribution without a dependence to the covariates $\mathbf{x}_i$ (in the first place), which means: $S(t_i|\mathbf{x}_i) = S(t_i)\ \forall i = 1, .., n$. In this case, we know the analytical expressions of the survival function and the hazard function with respect to the mixture parameters. As the parameters of the mixture of Weibulls are the only unknown parameters, this necessarily leads to consider the problem of estimating the parameters of this mixture. Let's start by taking the case of a single Weibull distribution.

2.2.2.1 Single Weibull Distribution

Here, we are dealing with a particular case where the survival times variable T follows a single two-parameter Weibull distribution, denoted by $W(\beta, \eta)$, whose parameters are the shape, denoted by β, strictly positive, and the scale parameter, denoted by η, which is also necessarily strictly positive. These two parameters can be estimated by maximizing the likelihood of this distribution, a method known as *Maximum Likelihood Estimation* (MLE) defined as follows:

$$\underset{\beta>0,\eta>0}{arg\ max}\ \mathcal{L}\Big(\beta, \eta|t_i\Big) = \prod_{i=1}^{n} \Big(S_{\beta,\eta}(t_i).h_{\beta,\eta}(t_i)\Big)^{\delta_i} \Big(S_{\beta,\eta}(t_i)\Big)^{1-\delta_i}. \tag{2.2.2}$$

where $S_{\beta,\eta}$ and $h_{\beta,\eta}$ are defined as follows:

$$S_{\beta,\eta}(y) = exp\Big\{ -\Big(\frac{y}{\eta}\Big)^{\beta}\Big\}, \tag{2.2.3}$$

$$h_{\beta,\eta}(y) = \Big(\frac{\beta}{\eta}\Big)\Big(\frac{y}{\eta}\Big)^{\beta-1}. \tag{2.2.4}$$

Since the *log* function is an increasing function, this problem is equivalent to the log-likelihood maximization problem defined by the following equation:

$$\tilde{\beta}, \tilde{\eta} = \underset{\beta>0,\eta>0}{arg\ max}\ log\ \mathcal{L}\Big(\beta, \eta|t_i\Big) \tag{2.2.5}$$

$$= \underset{\beta>0,\eta>0}{arg\ max} \sum_{i=1}^{n} \Big[\delta_i\ log\Big\{S_{\beta,\eta}(t_i).h_{\beta,\eta}(t_i)\Big\} + (1-\delta_i)\ log\Big\{S_{\beta,\eta}(t_i)\Big\}\Big] \tag{2.2.6}$$

Using the estimation of $\tilde{\beta}$ and $\tilde{\eta}$, the mean lifetime $\tilde{\mu}$ can thus be estimated and can be expressed as a mean lifetime of the Weibull distribution $W(\tilde{\beta}, \tilde{\eta})$ [93]:

$$\tilde{\mu}_i = \tilde{\eta}_i.\Gamma\Big(1 + \frac{1}{\tilde{\beta}_i}\Big) \ \forall i \in [n] \tag{2.2.7}$$

where Γ is the Gamma function.

To ensure that the log-likelihood of Weibull distribution is concave, we made a choice to add another constraint: we assume that the shape parameter β is greater or equal one ($\beta \geq 1$). In the next section, we consider the general case of a finite mixture of Weibull distributions.

2.2.2.2 Finite Mixture of Weibull Distributions

Now, we suppose that the survival times variable T rather follows a mixture of p ($< \infty$) Weibull distributions $\mathcal{W}_p = \{W(\beta_k, \eta_k), \alpha_k | k = 1, .., p\}$, where β_k and η_k are respectively, the shape and scale parameters of the k^{th} Weibull distribution of the mixture, and α_k are the weighting coefficients that weigh the contribution of each Weibull distribution in the mixture. Let $\beta = (\beta_1, .., \beta_p)$, $\eta = (\eta_1, .., \eta_p)$ and $\alpha = (\alpha_1, .., \alpha_p)$. These weighting coefficients verify:

$$\alpha \geq 0, \ \forall k = 1, .., p \tag{2.2.8}$$

$$\sum_{k=1}^{p} \alpha_k = 1. \tag{2.2.9}$$

The density of the mixture $\mathcal{W}_p$, noted $f_{\mathcal{W}_p}$, is defined by:

$$f_{\mathcal{W}_p} = \sum_{k=1}^{p} \alpha_k \ f_{\beta_k, \eta_k} \tag{2.2.10}$$

$$= \sum_{k=1}^{p} \alpha_k \ S_{\beta_k, \eta_k}(t_i).h_{\beta_k, \eta_k} \tag{2.2.11}$$

where $f_{\beta_k, \eta_k}, S_{\beta_k, \eta_k}(t_i)$ and h_{β_k, η_k} are respectively the density, survival and hazard functions of $W(\beta_k, \eta_k)$. Therefore, the log-likelihood of the mixture $\mathcal{W}_p$ can be written as follows:

$$\begin{aligned} log \ \mathcal{L}\Big(\beta, \eta, \alpha | (t_i)_i\Big) = \sum_{i=1}^{n} \Bigg[\delta_i \ log \ \Bigg\{ \sum_{k=1}^{p} \alpha_k S_{\beta_k, \eta_k}(t_i).h_{\beta_k, \eta_k}(t_i) \Bigg\} \\ + (1 - \delta_i) \ log \ \Bigg\{ \sum_{k=1}^{p} \alpha_k S_{\beta_k, \eta_k}(t_i) \Bigg\} \Bigg]. \end{aligned} \tag{2.2.12}$$

Therefore, we need to estimate the weighting coefficients $(\alpha_k)_k$ and the Weibull parameters $(\beta_k)_k$ and $(\eta_k)_k$ that maximize the log-likelihood and thus the likelihood of the event time distribution $(t_i)_i$. We do this by solving the following optimization problem:

$$\tilde{\beta}, \tilde{\eta}, \tilde{\alpha} = \underset{\substack{\beta > 0, \eta > 0 \\ \alpha_k \geq 0, \ \sum_k \alpha_k = 1}}{arg \ max} \ log \ \mathcal{L}\Big(\beta, \eta, \alpha | (t_i)_i\Big). \tag{2.2.13}$$

Knowing the analytical expression of the estimated expectation of a single Weibull, and given that the estimated mean $\tilde{\mu}$ of a finite mixture of Weibull distribution weighted by $\tilde{\alpha}$, is the $\tilde{\alpha}$-weighted combination of the means of the Weibull distribution $\tilde{\mu}_k$ that compose this mixture, i.e. $\tilde{\mu} = \sum_{k=1}^{p} \tilde{\alpha}_k.\tilde{\mu}_k$, with $\tilde{\mu}_k$ is defined in Equation (2.2.7). Thus, the estimated mean lifetime [93] can be written as follows:

$$\tilde{\mu} = \tilde{\alpha}\ \mathbf{diag}\ (\tilde{\eta}_1, .., \tilde{\eta}_p)\ \Gamma(1 + \tilde{\beta}^{-1})^T \tag{2.2.14}$$

where T denotes the transpose of vector operator, and $\tilde{\beta}^{-1} = (\frac{1}{\tilde{\beta}_1}, .., \frac{1}{\tilde{\beta}_p})$. We thus consider $\tilde{\mu}_i$ as the survival time estimation of the subject i.

2.3 Methodology

Now, we consider that the Weibull mixture's parameters are modeled conditionally to the features, i.e. we model $t_i|\mathbf{x}_i$ with a mixture of Weibull distributions and this is the main contribution of DeepWeiSurv. Let's describe the model in the next section.

2.3.1 Description of DeepWeiSurv

We name g_p the multivariate function that models the relationship between a subject's covariates $\mathbf{x}_i$ and the parameters of the mixture of p Weibull distributions. g_p is defined as follows:

$$\begin{array}{rcccl} g_p & : & \mathbb{R}^d & \rightarrow & (\mathbb{R}^p)^2 \cup (\mathbb{R}^p)^3 \\ & & \mathbf{x}_i & \mapsto & \begin{cases} (\beta, \eta) & \text{if } p = 1, \\ (\alpha, \beta, \eta) & \text{otherwise.} \end{cases} \end{array} \tag{2.3.1}$$

This function also treats the particular case namely: single Weibull distribution, i.e. $p = 1$. In this case, it is not required to estimate α because it is a scalar equal to 1. Since we cannot represent this function analytically, we propose to use a deep learning framework for this purpose. The resulting network-based model, which we call DeepWeiSurv has a global architecture as illustrated in Figure 2.2. Therefore, by training DeepWeiSurv that maximize the likelihood of the mixture, we have a network representation of the function g_p which means that we can estimate the parameters β and η as well as α (if $p > 1$). Our proposed approach is therefore a multi-task network. As seen in Figure 2.2, DeepWeiSurv is consisted of a common sub-network, a regression sub-network denoted by *reg* and a classification sub-network (if $p > 1$) denoted by *clf* whose all hidden layers are fully connected and activated by ReLU [94] activation function :

- *The shared sub-network* calculates a common representation in a latent space. It takes as an input the survival data X of size n and calculates Z the latent representation of the data. For a mixture setting, i.e. $p = 1$, *clf* and *reg* sub-networks take Z as input towards outputting the estimate of α and that of the couple (β, η) respectively.
- For the *regression sub-network*, we use *Exponential Linear Unit* [37] (ELU) function with constant equal to 1:

$$ELU(y) = \begin{cases} y & \text{if } y > 0 \\ e^y - 1 & \text{otherwise.} \end{cases} \tag{2.3.2}$$

  We choosed this activation function for several reasons notably the fact that, unlike ReLU function, it produces negative outputs which help the network updating its

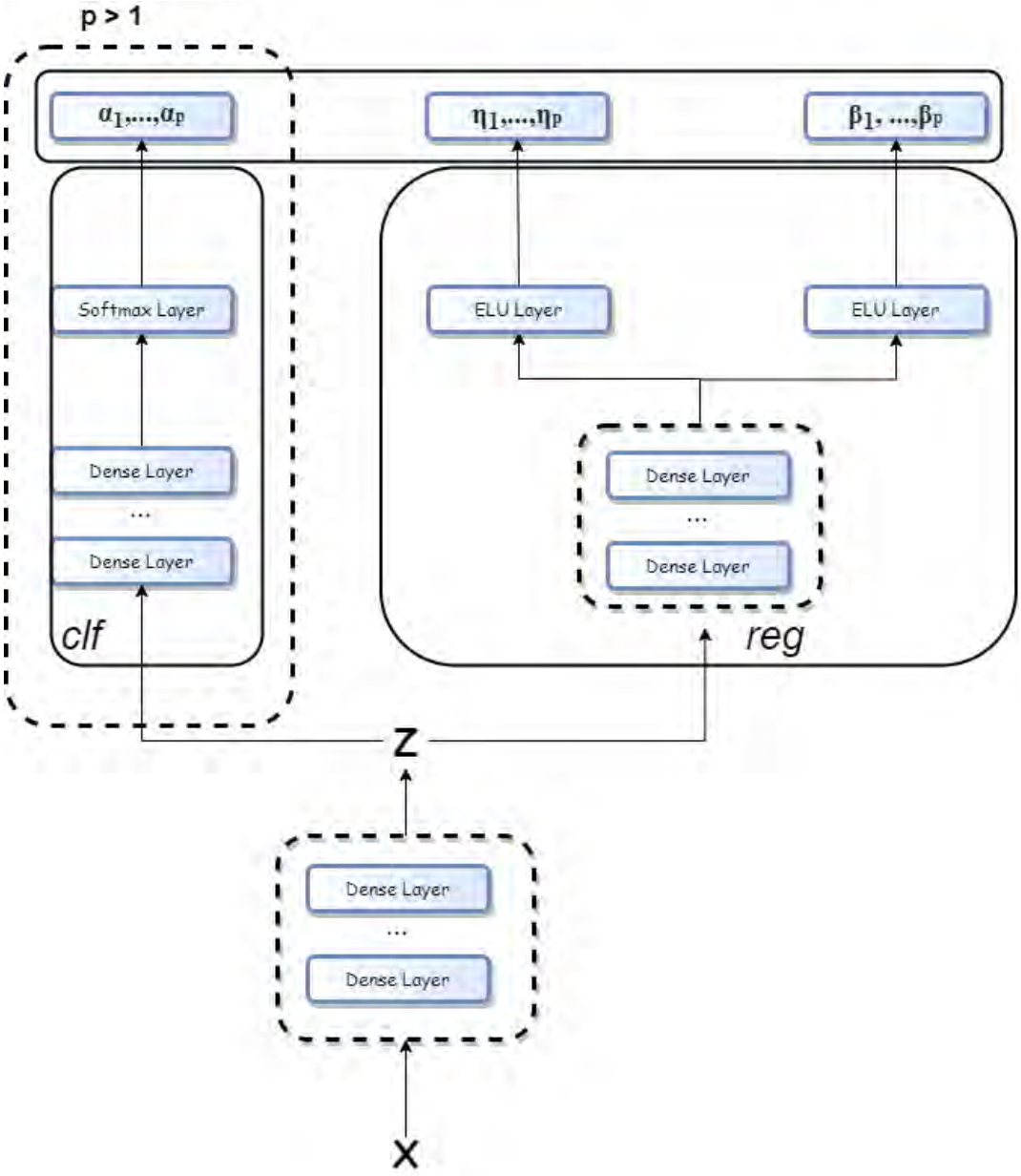


Figure 2.2: The architecture of DeepWeiSurv.

weights and biases and also produces activations instead of letting them be zero, when calculating the gradient in the network training phase. However, the regression sub-network is expected to outputs strictly positive values for η output layer and values greater than 1 for β output layer and as the co-domain of ELU is $]-1,+\infty[$, the network will rather learn:

$$\begin{cases} \beta - 2 \\ \eta - 1 - \epsilon,\ \epsilon > 0 \end{cases} \tag{2.3.3}$$

to get around this constraint. After the training step, these offset operations $\beta \leftarrow \beta + 2$ and $\eta \leftarrow \eta + 1 + \epsilon$ are then applied to recover the concerned parameters.

- Regarding the *classification sub-network*, this latter learn α while ensuring these two constraints:

$$\begin{cases} \sum_{k=1}^{p} \alpha_k = 1 \\ \alpha_k \in [0,1]\ \forall k \in [p]. \end{cases} \tag{2.3.4}$$

For this purpose we used a *softmax* [36] activation in the output layer of *clf*. Therefore, *reg* learns the p vectors of parameters $\beta_k = (\beta_{ik})_{i=1,..,n}$ and $\eta_k = (\eta_{ik})_{i=1,..,n}$, whereas *clf* learns the p $\alpha_k = (\alpha_{ik})_{i=1,..,n}$ where α_{ik} is defined as the estimated probability of the event $\{Y = t_i\}$ with Y is a random variable following $W(\beta_{ik}, \eta_{ik})$.

In the case of when we model the survival times by a single Weibull distribution, i.e. $p = 1$, we have $\alpha = \alpha_1 = 1$, thus we don't consider the classification sub-network.

2.3.2 Loss Function

We recall that the estimated parameters and coefficients are supposed to maximize the likelihood of the mixture distribution. Thus, to train DeepWeiSurv, we use the following loss function:

$$loss = -\frac{1}{n}\, log\, \mathcal{L}(\beta, \eta, \alpha | (t_i)_i) \tag{2.3.5}$$

$$\underset{(2.2.12)}{=} -\frac{1}{n}\Big(\mathcal{L}_1^T.\Delta + \mathcal{L}_2^T.(\mathbf{1}_{\mathbb{R}^n} - \Delta)\Big) \tag{2.3.6}$$

where $\mathbf{1}_{\mathbb{R}^n}$ a vector of ones of size n, $\Delta = \big(\delta_1, .., \delta_n\big)$ is the vector of event indicators of all the subjects and $\mathcal{L}_1$, $\mathcal{L}_2$ are defined as follows:

$$\mathcal{L}_1 = log\Big\{\mathbf{A}.\mathbf{diag}\ (\mathbf{M}^{\mathbf{f}}_{\beta,\eta}(t_1)^T, .., \mathbf{M}^{\mathbf{f}}_{\beta,\eta}(t_n)^T)\Big\} \tag{2.3.7}$$

$$\mathcal{L}_2 = log\Big\{\mathbf{A}.\mathbf{diag}\ (\mathbf{M}^{\mathbf{S}}_{\beta,\eta}(t_1)^T, .., \mathbf{M}^{\mathbf{S}}_{\beta,\eta}(t_n)^T)\Big\} \tag{2.3.8}$$

with $\mathbf{A} = (\alpha_{ik})_{\substack{1 \le i \le n \\ 1 \le k \le p}}$ and:

$$\mathbf{M}^{\mathbf{f}}_{\beta,\eta}(t_i) = \Big(h_{\beta_k,\eta_k}(t_i) S_{\beta,\eta_k}(t_i)\Big)_{1 \le k \le p} \tag{2.3.9}$$

$$\mathbf{M}^{\mathbf{S}}_{\beta,\eta}(t_i) = \Big(S_{\beta_k,\eta_k}(t_i)\Big)_{1 \le k \le p},\ \forall i \in [n] \tag{2.3.10}$$

$\mathcal{L}_1$ exploits the information provided by the observed events of non-censored data and thus use the density $h_{\beta_k,\eta_k} S_{\beta_k,\eta_k}$, whereas $\mathcal{L}_2$ exploits the censored subjects by extracting the

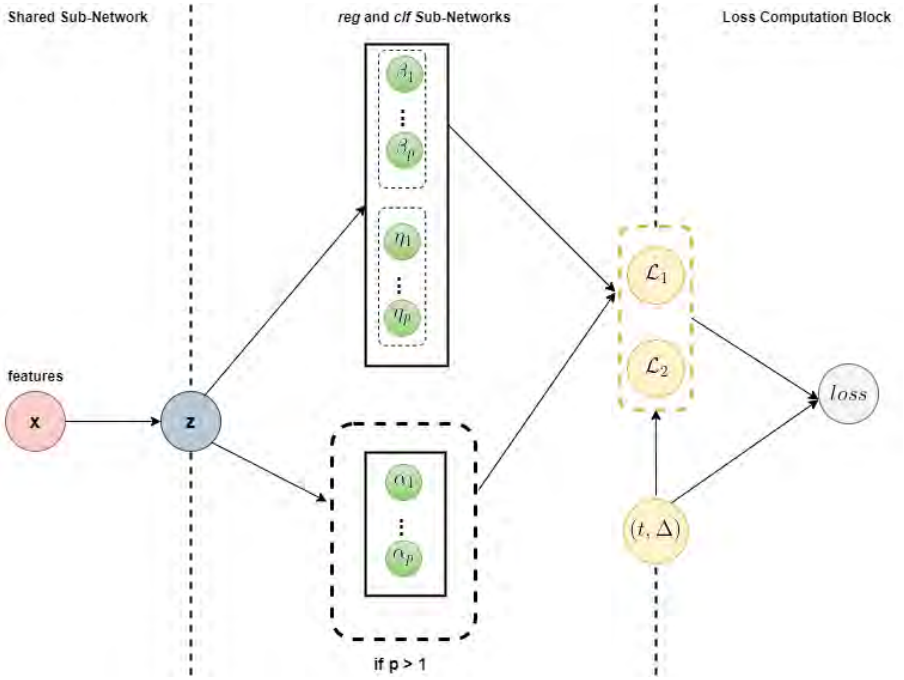


Figure 2.3: Computational graph of the loss function used to train DeepWeiSurv.

knowledge that their respective events will occur after the end of the study. This means that this information is the only one that we have about the right censored subjects hence the use of the survival function S_{β_k,η_k}. Figure 2.3 is an illustration of the computational graph of our loss function used for training DeepWeiSurv. The inputs are the baseline data of covariates x, the real values of survival times and event indicator vector are denoted by the couple (t, Δ) and the final outputs are the triplet (β, η, α) if $p > 1$ and (β, η) otherwise.

2.4 Experiments on Simulated Data

In this section, we run different experiments on synthetic datasets. The goal here is to evaluate and then validate DeepWeiSurv, that is, to show that this latter is able to estimate the mixture parameters. In the first experiment, the main idea is to estimate the likelihood of a simulated mixture of Weibull Distribution and compare the estimation with the real value for different forms that parameters can take. In the second experiment, we evaluate the ability of the model to reproduce the relationship between the parameters and the covariates. In the third experiment, we perform a clustering problem where the goal is to evaluate the ability of DeepWeiSurv in estimating the weighting coefficients α. For the fourth and last experiment, we observe the ability of our proposed approach in handling the highly censoring setting.

2.4.1 Network Configuration

DeepWeiSurv consists of three sub-networks: the shared sub-network is a 4-layer network, three of which are fully connected layers with 128,64,32 nodes respectively and the last one

is a batch normalization layer. The regression and classification sub-networks consist of two fully connected layers with 32 and 16 nodes respectively, and one batch normalization layer with two ELU and softmax output layers, respectively. The hidden layers of the three sub-networks are activated by ReLU. DeepWeiSurv is trained via Adam optimizer with a learning rate of 10^{-4}. The network is implemented in a Pytorch environment.

2.4.2 Experiment I: Likelihood Estimation

Let $\mathbf{X}$ be a set of $n = 1000$ one-dimensional observations generated from the uniform distribution $\mathcal{U}_{[0,1]}$ of support $[0, 1]$. In this experiment, we simulate the Weibull parameters conditionally to $\mathbf{X}$, train DeepWeiSurv (of parameter p, the mixture size) and compare the mean negative log-likelihood (*mnll*) in this experiment (defined in Equation (2.3.6)) estimated by DeepWeiSurv with the real value. The models considered here are: DeepWeiSurv with $p = 1, 2, 3, 4$ and 5 Weibull distributions. We propose four different case studies:

- Single Weibull Distribution of parameters β and η, dependent linearly to $\mathbf{X}$, defined by the following functions:

$$\beta = 3\mathbf{X} + 2$$
$$\eta = 2\mathbf{X} + 1$$

- A mixture of 2 Weibull distribution (70% for the first distribution, 30% for the second one) of parameters $(\beta^{0.7}, \eta^{0.7})$ and $(\beta^{0.3}, \eta^{0.3})$ respectively. These parameters are defined by the following functions:

$$\beta_1 = 3\mathbf{X} + 2 \qquad \eta_1 = 2\mathbf{X} + 1$$
$$\beta_2 = 1.5\mathbf{X} + 1 \qquad \eta_2 = 0.5\mathbf{X} + 0.5$$

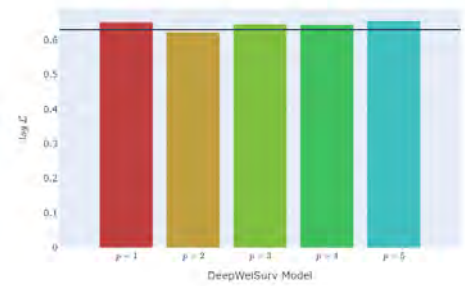
- Single Weibull Distribution of parameters β and η. Both of them have a quadratic form with respect X and defined as follows:

$$\beta = 2\mathbf{X}^2 + \mathbf{X} + 1$$
$$\eta = \mathbf{X}^2 + \mathbf{X} + 1$$

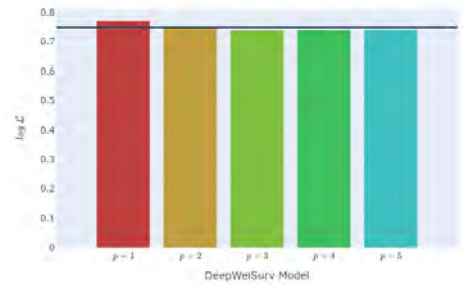
- A mixture of 2 Weibull distribution (70% for the first distribution, 30% for the second one) of parameters $(\beta^{0.7}, \eta^{0.7})$ and $(\beta^{0.3}, \eta^{0.3})$ respectively. These parameters are polynomial or order 2 and 3, and defined as follows:

$$\beta_1 = 2\mathbf{X}^2 + \mathbf{X} + 1 \qquad \eta_1 = \mathbf{X}^2 + 1$$
$$\beta_2 = 2\mathbf{X}^3 + 1 \qquad \eta_2 = \mathbf{X}^2 + \mathbf{X} + 1$$

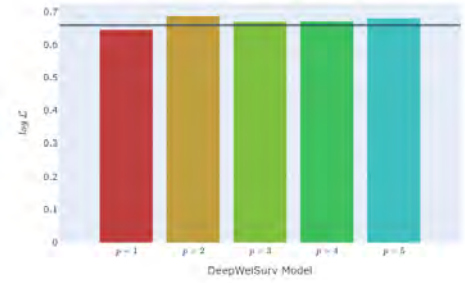
The results are displayed in bar plot in Figure 2.4. We can notice from the results shown in 2.4 that, for all scenarios, all the models have found a good approximation of the real value of the log-likelihood (the projection of the black horizontal line onto the y-axis). Still, some models perform better than others in each case study. For instance, in the first case (2.4a), the best approximations are provided by $p = 2$ and $= p = 4$ with *mnll* equal to 0.6193 and 0.634 respectively, whereas the real value is 0.629. In the second scenario (2.4b), given the true value of *mnll*: 0.7498, $p = 2$ has the best approximation with a value of 0.7402. $p = 1$ gives a slight overestimation, whereas $p \geq 3$ slightly underestimate the log-likelihood, but



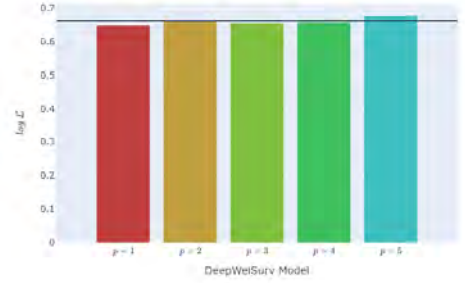
(a) Single Weibull - β, η linear.



(b) $p = 2, \alpha = (0.7, 0.3)$ - β_k, η_k linear.



(c) Single Weibull - β, η quadratic.



(d) $p = 2, \alpha = (0.7, 0.3)$ - β_k, η_k not linear.

Figure 2.4: Results of Experiment One: For each scenario, the mean negative log-likelihood is estimated by each considered model (in bar plot) and compared with the real value (black horizontal line).

these three models still outperform $p = 1$. These seem logical because we generate a mixture of 2 Weibull distributions, thus if DeepWeiSurv with $p = 2$ is well trained, it can only get the best approximation of *mnll*. In addition, with $p = 1$, we have more chance to poorly estimate the parameters of a mixture whose size is equal to 2 than with $p \geq 2$, especially when the two Weibull that compose the mixture are largely different. For the third scenario (2.4c), $p = 3$ has the best estimation with a value of 0.669 where the true value is 0.66. And finally, in the fourth scenario (2.4d) where the true value of the mean log-likelihood of the simulated distribution is 0.6615, $p = 2$ provides the best estimation with a value of 0.6609 (difference of order 10^{-4}) but the models with $p = 3$ and 4 respectively also get a good approximation with a value of 0.658 and 0.6585 respectively.

In this experiment, we evaluated the performance of DeepWeiSurv (with different values of p) in terms of estimating the likelihood of the distribution generated using different scenarios. Now, we will move to the next experiment, where the goal is to see more specifically if DeepWeiSurv can get a good estimation of the mixture's parameters.

2.4.3 Experiment II: Parameters-Features Relationship Modelling

In this experiment, we seek to investigate and evaluate DeepWeiSurv's ability to model the relationship between the baseline data features and the mixture parameters, Let X a vector of size $n = 5000$ of one-dimensional samples drawn from the uniform distribution $\mathcal{U}_{[0,1]}$ of support $[0, 1]$. We propose four scenarios where in each one of them, we try to reproduce the parameters of the distribution with which we generate the survival times used to train DeepWeiSurv. In this experiment, we consider that all the samples are non-censored. The case studies considered are as follows:

- We generate survival times following a single Weibull distribution whose parameters β, η are linear and defined as follows:

$$\beta = \mathbf{X} + 2 \qquad\qquad \eta = \frac{1}{2}(\mathbf{X} + 1).$$

- We generate survival time samples from a single Weibull distribution whose parameters are defined by the following functions:

$$\beta = \mathbf{X}^2 + 2\mathbf{X} + 2 \qquad\qquad \eta = \frac{1}{2}\mathbf{X}^2 + \mathbf{X} + \frac{1}{2}.$$

- We generate survival times drawn from a single Weibull distribution with parameters defined by the following equations:

$$\beta = 2\sin(2\mathbf{X} + 1)\sin(1 + e^{\mathbf{X}}) + \frac{7}{2} \qquad\qquad \eta = \cos(1 + \mathbf{X})\, e^{\mathbf{X}^2} + \frac{1}{2}.$$

- We generate survival times from a 50-50 mixture of 2 Weibull distributions whose parameters $\beta_i, \eta_i, i = 1, 2$ are defined by the following functions:

$$\beta_1 = \mathbf{X} + 2 \qquad\qquad \eta_1 = \frac{1}{2}\mathbf{X}^2 + \mathbf{X} + \frac{1}{2},$$
$$\beta_2 = \mathbf{X}^2 + 2\mathbf{X} + 2 \qquad\qquad \eta_2 = \frac{1}{2}(\mathbf{X} + 1).$$

In the three first scenarios, we use DeepWeiSurv with $p = 1$, whereas the last, we use DeepWeiSurv with $p = 2$. For each case study, we plot three figures: the estimate $\hat{\beta}$ vs. β,

$\hat{\eta}$ vs. η and the estimated mean $\hat{\mathbb{E}}$ vs. the exact mean. The results obtained are in Figure 2.5 (for Scenario 1, 2 and 3) and Figure 2.6 (for Scenario 4).

In Scenario 1 and 2 where we use a single Weibull distribution whose parameters have, respectively, linear and quadratic forms, we can notice that the estimate $\hat{\beta}$ and $\hat{\eta}$ almost coincide with β and η respectively. This means that the estimated mean $\hat{\mathbb{E}}$ practically coincides with $\mathbb{E}$ since this latter is a function of β and η and this is what we see in the plots 2.5g and 2.5h. For the third scenario, where the two parameters have a complex function, we can see that the model has difficulty in estimating the shape of β; nevertheless, $\hat{\beta}$ and β are both decreasing, in the same range values and the values of both of them are not far from each other, while $\hat{\eta}$ almost coincide with η which still a good thing because as we see in the plot 2.5i, the estimate of the mean lifetime $\hat{\mathbb{E}}$ is finally almost equal to $\mathbb{E}$. Therefore, this case study allows us to see, in these three case studies, that η has the biggest impact on the mean lifetime as this latter takes the shape of η and this is because $\Gamma(1 + \frac{1}{\beta})$ does not vary enough and does not stray too far from 1 which means that $\mathbb{E}$ is almost linear with respect η and, above all, they coincide in these cases as we see in 2.5g, 2.5h and 2.5i.

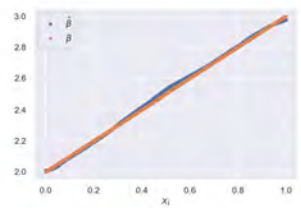
For the last case study (Figure 2.6), where we are dealing with a 50-50 mixture of 2 Weibull distributions, we can notice that $\hat{\beta}_i$ and $\hat{\eta}_i$ have practically the values of β_i and η_i respectively, except for $\hat{\beta}_1$ (see Figure 2.6a) which mostly under- or over-estimate the true values but still with a small deviation. This deviation does not substantially affect the quality of the mean estimate $\hat{\mathbb{E}}$ (see Figure 2.6e) since the latter largely depends on the $\hat{\eta}_i$.

We can, therefore, conclude from this experiment, that DeepWeiSurv provides promising results in modeling the relationship between the parameters (β, η) and the covariates $\mathbf{X}$. Still, the performance of DeepWeiSurv depends on the complexity of this relationship (Scenario 3 more complex than 1 and 2) as well as the size of the mixture (Scenario 4 more complex than Scenario 1, 2, and 3).

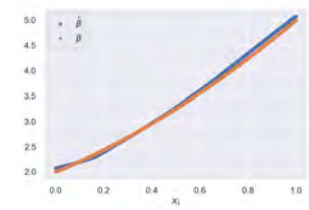
2.4.4 Experiment III: Clustering

Here, we want to test and evaluate the ability of DeepWeiSurv in estimating the weighting coefficients α. For this purpose, we conduct this experiment where we consider a problem of clustering. The main idea is to generate a point cloud in such a way that it forms two clusters that cannot be separated by a hyperplane. Each cluster is characterized by a Weibull distribution, which means that all the samples belonging to a given cluster have time outputs drawn from the same Weibull distribution. We draw samples with the function *make_moon* from a Python package called *scikit-learn* that generates two clusters in a shape of a moon in the waxing crescent phase (we add a standard deviation of Gaussian noise in the *make_moon* function). For each sample i of the first cluster, we draw t_i from $W(\beta_1 = 6.5, \eta_1 = 5)$ and for each sample j of the second cluster, we draw t_j from $W(\beta_2 = 1.5, \eta_2 = 0.5)$. The parameters of these Weibull distributions are chosen in a such way that they are markedly different (see Figure 2.7a). As in the previous experiment, here, we suppose that all the samples are non-censored. We model the time distributions of these two clusters with a mixture of 2 Weibull distributions (DeepWeiSurv with $p = 2$). The main objective is to check if the model can reproduce this clustering. For this purpose, we use, for each sample i, the estimate of $\alpha_i = (\alpha_{1i}, \alpha_{2i}) \in [0, 1]^2$ where α_{ij} is the probability that the sample i belong to the j^{th} cluster. We see the results in Figure 2.7b where we visualize the value of α_{2i} of each sample i. We use the colormap to represent the intensity of the variable that we visualize (between 0 and 1). Since $\alpha_{1i} + \alpha_{2i} = 1, \forall i$, α_{2i} equal or close to 0 (respectively 1) means that i belongs to the cluster one (respectively cluster two).

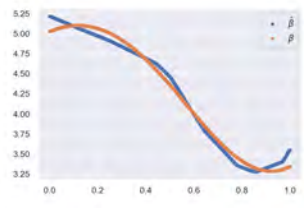
We notice from the figure 2.7b, the big majority of samples are assigned to their real cluster (the samples in red and purple as in Figure 2.7b). The samples that are at the edge



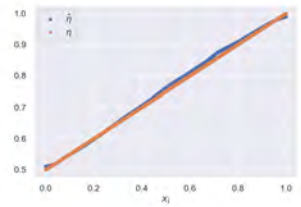
(a) $\hat{\beta}$ vs β - Scenario 1



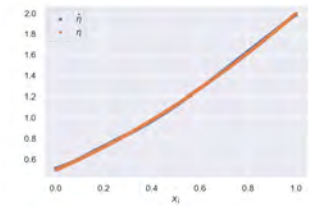
(b) $\hat{\beta}$ vs β - Scenario 2



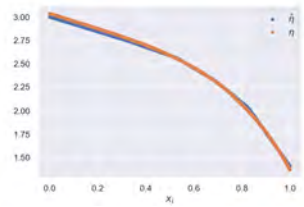
(c) $\hat{\beta}$ vs β - Scenario 3



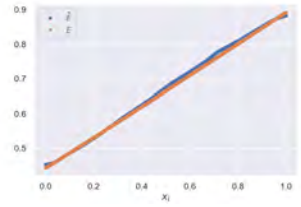
(d) $\hat{\eta}$ vs η - Scenario 1



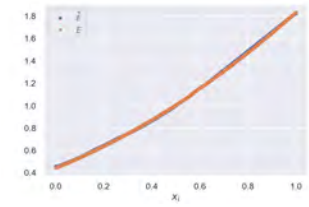
(e) $\hat{\eta}$ vs η - Scenario 2



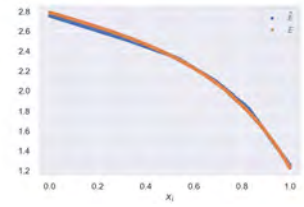
(f) $\hat{\eta}$ vs η - Scenario 3



(g) $\hat{\mathbb{E}}$ vs $\mathbb{E}$ - Scenario 1

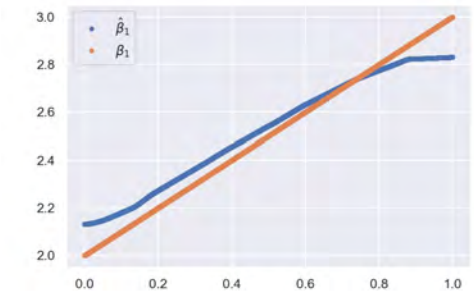


(h) $\hat{\mathbb{E}}$ vs $\mathbb{E}$ - Scenario 2

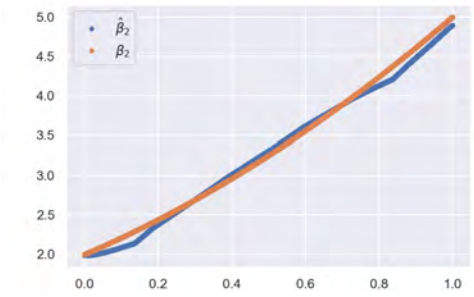


(i) $\hat{\mathbb{E}}$ vs $\mathbb{E}$ - Scenario 3

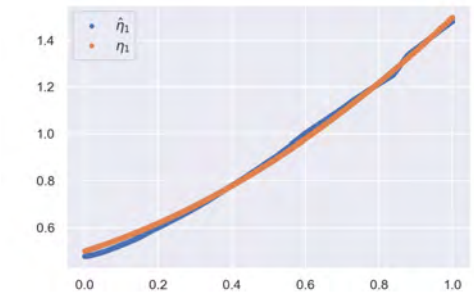
Figure 2.5: Results of Scenario 1 (first column) and 2 (second column): $\hat{\beta}$ vs β in the first row, $\hat{\eta}$ vs η in the second one and $\hat{\mathbb{E}}$ vs $\mathbb{E}$ in the last one.



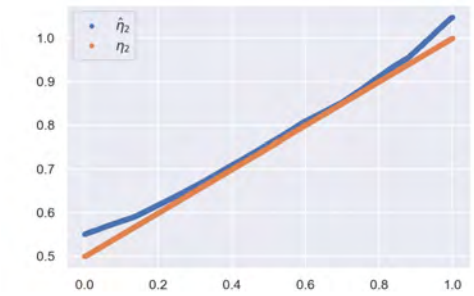
(a) $\hat{\beta}_1$ vs β_1



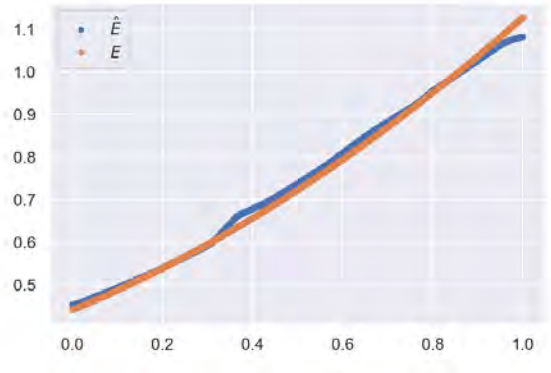
(b) $\hat{\beta}_2$ vs β_2



(c) $\hat{\eta}_1$ vs η_1

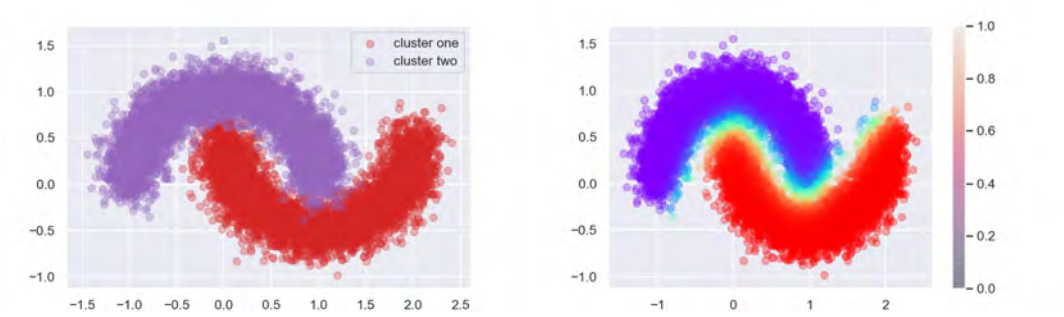


(d) $\hat{\eta}_2$ vs η_2



(e) $\hat{\mathbb{E}}$ vs $\mathbb{E}$

Figure 2.6: Results of Scenario 4: $\hat{\beta}_{i=1,2}$ vs $\beta_{i=1,2}$ in the first row, $\hat{\eta}_{i=1,2}$ vs $\eta_{i=1,2}$ in the second row and finally in the third row, the predicted mean $\hat{\mathbb{E}}$ and the true mean of the mixture $\mathbb{E}$.



(a) The drawn samples

(b) The samples coloured w.r.t α_{2i}

Figure 2.7: On the left, two generated clusters using the *make_moon* function from scikit-learn package. On the right, the clustering reproduced by DeepWeiSurv with $p = 2$ using the values of the second component of α.

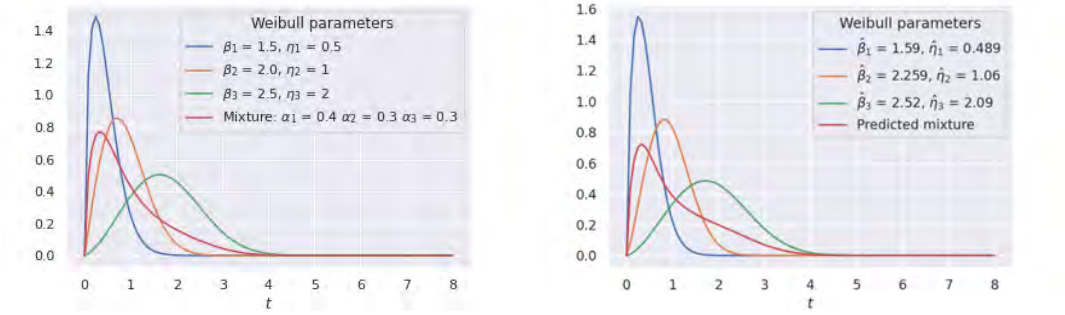
of their real cluster or close to the neighbor cluster have values between 0 and 1. Still, the border points of the first cluster have $\alpha_{2i} \to 0$ (blue-stained or colored in sky blue) and those of the second cluster have $\alpha_{2i} \to 1$ (have a color between yellow and orange). This means that if we apply a certain threshold on α_{2i} values for these border points, we finally reproduce the clustering of this sample. We, therefore, conclude, that DeepWeiSurv gives first promising results in the estimation of the weighting coefficients, at least when the simulated data is well chosen.

2.4.5 Experiment IV: Censoring Threshold Sensitivity

The main objective in this experiment is to evaluate the performance of DeepWeiSurv with respect to the censoring rate, denoted by r_c, present in the data, i.e. the size of censored samples against the size of the data. We aim at each scenario defined by a different value of r_c, reproduce the simulated distribution. Admittedly, the difficulty of modeling the distribution increases with the censoring rate but also varies with the shape of the distribution. For this purpose, we test two different shapes in this experiment: uni-modal and bi-modal mixture.

2.4.5.1 Uni-Modal Mixture

In this case study, we draw $n = 10000$ time samples from $\mathcal{W}_3$ a mixture of three Weibull distributions of parameters $(\beta_1 = 1.5, \eta_1 = 0.5)$, $(\beta_2 = 2, \eta_2 = 1)$ and $(\beta_3 = 2.5, \eta_3 = 2)$ respectively, with weighting coefficients of 0.4, 0.3 and 0.3 respectively. The composing Weibull and the mixture distributions are plotted in Figure 2.8a. At an initial stage, we train DeepWeiSurv with $p = 3$ on the samples without considering the censoring rate, i.e. we first consider that all the samples are non-censored, and obtain the predicted values of the triplets of parameters denoted $(\hat{\alpha}_i, \hat{\beta}_i, \hat{\eta}_i)_{i=1,2,3}$. We plot, as shown in Figure 2.8b, the predicted Weibull densities as well as the mixture of them using the parameters $(\hat{\beta}_i, \hat{\eta}_i)_{i=1,2,3}$ and the predicted weighting coefficients $(\hat{\alpha}_i)_{i=1,2,3}$. We can notice that DeepWeiSurv has reproduced the three composing Weibull distributions of the mixture considered. Now, we choose different values of the censoring rate r_c. This means that for each value of the censoring rate r_c, we have a censoring time t_c, described as a threshold, above which the recorded time is considered as censored. By applying the censoring rate r_c, the samples



(a) Simulated Weibull distributions

(b) Predicted Weibull distributions.

Figure 2.8: The density of Weibull distributions simulated on the left, and predicted ones on the right.

are split into non-censored instances (of portion 1-r_c) and censored ones (r_c). Then, for each case study, we train DeepWeiSurv with $p = 3$ on the samples (censored and non-censored) associated and compare the simulated mixture distribution $\mathcal{W}_3$ with the one which is predicted by the model since the goal is to estimate the overall distribution. We obtain the results for $r_c \in \{0, 0.25, 0.35, 0.45, 0.65, 0.75, 0.85, 0.95\}$ in Figure 2.9. The purple curve corresponds to the density of the mixture $\mathcal{W}_3$ simulated whereas the brown curve represents the density of the mixture predicted by DeepWeiSurv ($p = 3$). The censoring time t_c is represented in all the plots with a red vertical line.

The first thing that we can notice, regarding the censoring rate r_c, is that the higher is the censoring rate, the poorer is the prediction of the model which is logical since when we increase further the ratio of censored samples, we have less information about the overall distribution. Although, the model learns the general shape of the density in all cases. For $r_c = 0$, which means that all the time samples are non-censored, the density of the predicted mixture coincides with that of the mixture $\mathcal{W}_3$ simulated as seen in Figure 2.9a. The same goes for $r_c = 0.25$, 0.35, and 0.45, where the model keeps practically the same accuracy, as for $r_c = 0$, in estimating the mixture density. For $r_c = 0.65$ and $r_c = 0.75$, the respective predicted densities of the underlying distribution have approximately the same value of the peak as the real one and in the same position as well, but slightly overestimated shortly after the peak and then underestimated with earlier mitigation.

For $r_c = 0.85$ and $r_c = 0.95$, the peak of the density is slightly overestimated, and the slope of the decreasing part of the curve (shortly after the peak) is greater than that of the real one, which means that the mitigation of the respective densities is earlier than expected as shown in Figure 2.9g and Figure 2.9h. The model loses more accuracy because when 85% or 95% of the samples are censored, the model cannot have enough information about the tail of the underlying distribution and this renders the estimation more constraining. Still, the model correctly predicts the position of the peak. We can say that DeepWeiSurv provides promising results in handling the highly censoring setting in the case of uni-modal Weibull distribution.

2.4.5.2 Bi-Modal Mixture

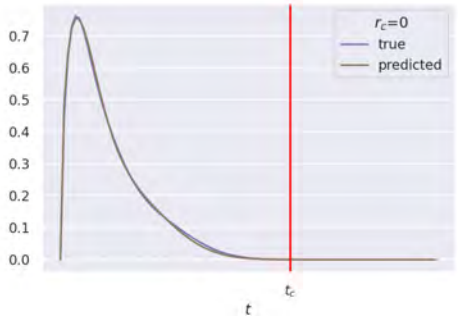
We conduct a similar experiment but we used three Weibull distributions in a such way that their mixture generates two peaks (bi-modal mixture). This example is more complicated

than the previous one for the simple reason that in the case of an important censoring rate, the model could have difficulty predicting the second peak especially if it is separated from the first one in time. Let $\mathcal{W}_3$ be a mixture of three Weibull distributions whose parameters are respectively, $(\beta_1 = 2, \eta_1 = 0.5), (\beta_2 = 2.5, \eta_2 = 2$ and $(\beta_3 = 4, \eta_3 = 3)$ with weighting coefficients of 0.4 0.2 and 0.4 respectively (see Figure 2.10a). In this experiment, we test the following values of the censoring rate r_c: 0, 0.3, 0.4, 0.65, 0.75 and 0.85. For $r_c = 0$ and 0.3, the model has a good estimation of the underlying distribution (the brown and purple curves, representing the predicted and the simulated distributions respectively). Now, when we set 40% of the samples to censored status, DeepWeiSurv slightly underestimates the second peak of the mixture (as seen in Figure 2.10d), and the more we increase the censoring ratio, the more the model has difficulty in estimating the position and the magnitude of the second peak until it ignores it, as we see for the case $r_c = 0.85$. This seems logical because for instance when we say that 50% of the time samples are censored, this means that all the time samples that are greater than the median time are not observed, thus if the second peak of the mixture is in this non-observed region, the model has difficulty to predict if there is another peak or more generally the shape of the density in this non-observed region. Therefore, this difficulty increases with the censoring rate. As we see in Figure 2.10e, the model further underestimates the position of the second peak with an earlier attenuation (the same goes for $r_c = 0.75$). Finally, in the case where we have 85% of the time samples are censored, the model only predicts the position of the first peak with a slight overestimation of its magnitude. In this scenario, the model has completely ignored the second peak and replaces it with earlier density mitigation. Certainly, in the case of the bi-modal mixture, DeepWeiSurv has more difficulties handling a highly censoring setting than in the uni-modal mixture case, but still manages samples with an important portion of censored ones.

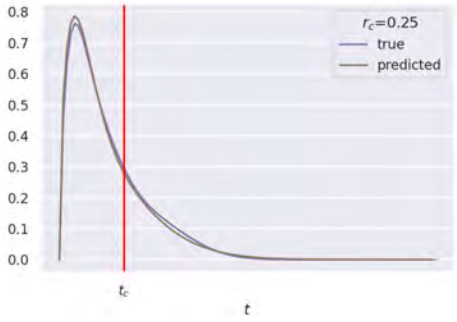
2.5 Conclusion

In this chapter, we presented a novel network-based approach to the survival analysis, called DeepWeiSurv. The main idea behind this approach is to assume that event times distribution can be modeled by a mixture of Weibull distributions whose parameters are functions of the covariates at an individual scale. DeepWeiSurv, therefore, learns, via its regression and classifier sub-networks, these parameters as well as the mixture weighting coefficients, respectively, by maximizing the log-likelihood of the mixture. We conducted four different simulated experiments on simulated data to assess different properties of the model. In the first experiment, the goal was to estimate the likelihood of a simulated mixture of Weibull distribution whose parameters take different shapes with respect to the features of the generated data and where DeepWeiSurv had good performance for all scenarios considered. Whereas in the second experiment, the main objective was to evaluate the ability of DeepWeiSurv to model the relationship or the function that links the covariates to the parameters of a mixture of Weibull distributions. To do so, we tested DeepWeiSurv with different nature of the relationship (linear, quadratic, and a trigonometric-based function) and in a case of single and a mixture of 2 Weibull distributions, where, even if the complex scenarios result in an overall loss of performance, it showed generally good performance. The third experiment was dedicated to the weighting coefficients estimation which tells us about the importance of each Weibull distribution that composes the mixture. We conducted a clustering experiment where each cluster is labeled by a Weibull distribution. The goal was to evaluate the ability of the model to estimate the Weibull distributions parameters of each sample from both clusters where samples from the same cluster are expected to have the same values of the parameters. DeepWeiSurv had easily and correctly estimated the parameters of the samples

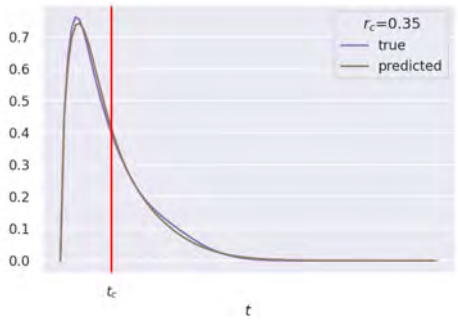
that are far from the neighbor cluster, but with a slight difficulty for the samples in the border. Finally, we evaluate the sensitivity of DeepWeiSurv towards the increasing of the censoring rate. We showed that in general, DeepWeiSurv handles a highly censoring setting for single-mode mixture, but for bi-modal mixture, the model tends to ignore a part of the distribution as the censoring rate increases, but the model still handles a significant portion of censored data. The last experiment showed that the approach allows considering many survival time horizons of prediction. In the next chapter, we will discover an extension of this approach where the final size mixture is to be estimated.



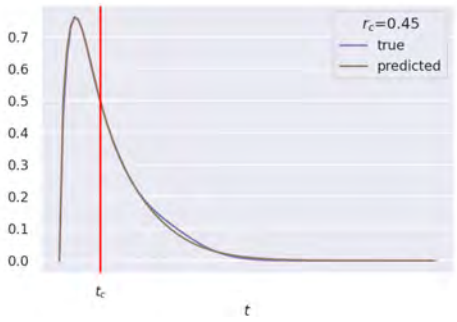
(a) $r_c = 0$



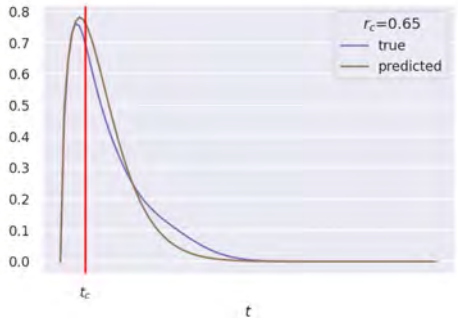
(b) $r_c = 0.25$



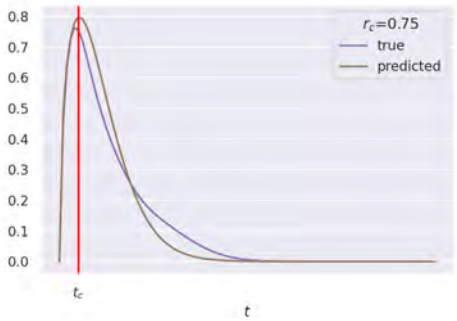
(c) $r_c = 0.35$



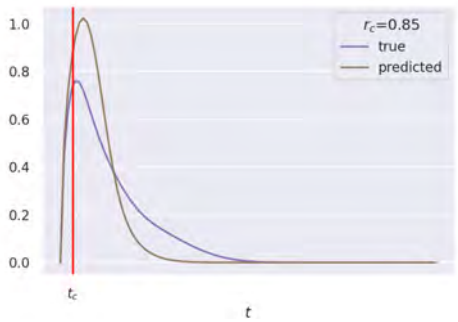
(d) $r_c = 0.45$



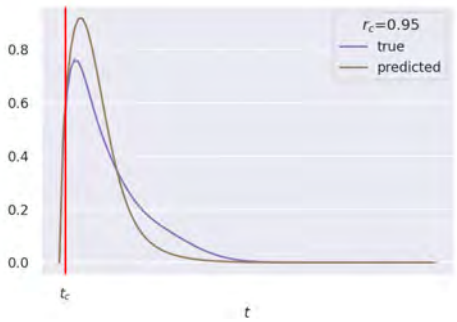
(e) $r_c = 0.65$



(f) $r_c = 0.75$

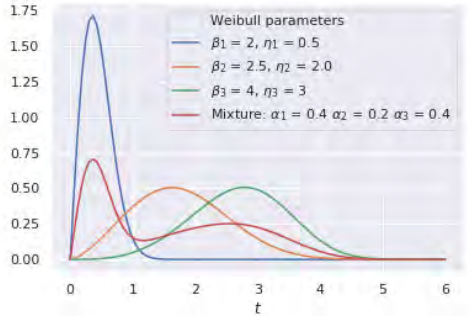


(g) $r_c = 0.85$

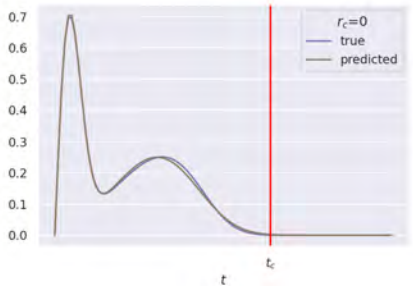


(h) $r_c = 0.95$

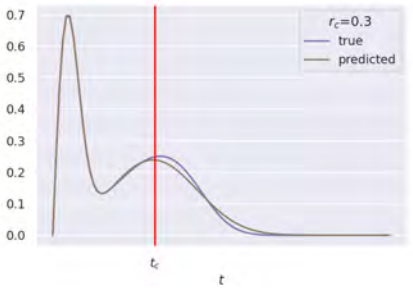
Figure 2.9: Results of the conducted experiment on the single-mode mixture repeated with different values of censoring rates r_c: densities of the simulated mixture vs. predicted mixture.



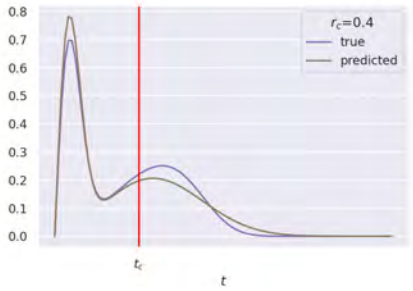
(a) Bi-modal mixture



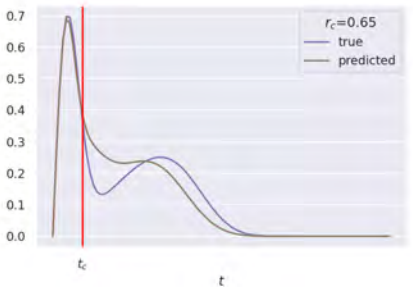
(b) $r_c = 0$



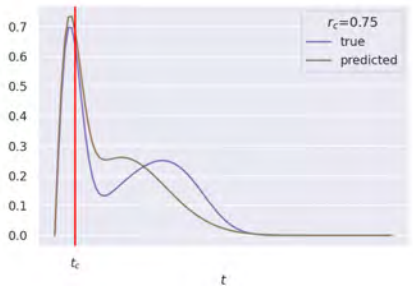
(c) $r_c = 0.3$



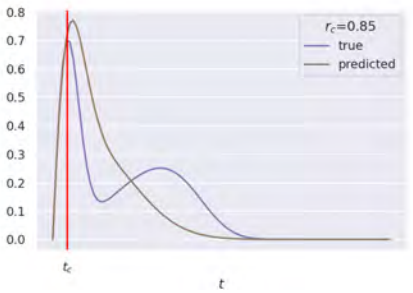
(d) $r_c = 0.4$



(e) $r_c = 0.65$



(f) $r_c = 0.75$



(g) $r_c = 0.85$

Figure 2.10: In the first row: density of $\mathcal{W}_3$ and those of the Weibull distributions that compose this mixture. In the remaining rows: results of the conducted experiment on the bi-modal mixture repeated with different values of censoring rates r_c: densities of the simulated mixture vs. predicted mixture.

Chapter 3

DPWTE: A Deep Learning Approach to Survival Analysis using a Parsimonious Mixture of Weibull Distributions

3.1 Introduction

In this chapter, we propose an extended version of DeepWeiSurv, the network-based approach to time-to-event analysis, that we have described in the previous chapter. As for DeepWeiSurv, we assume that the underlying event-time distribution can be modeled by a finite mixture of Weibull distributions whose parameters are to be estimated conditionally to the baseline data features at an individual level. In addition, no particular assumption about the nature of the relationship between the parameters and the features is made. The main difference here compared to DeepWeiSurv, is that we seek to estimate the optimal combination of the Weibull distributions assuming that the size of this optimal mixture is not fixed. This means that it is up to the model that we propose, which we call DPWTE standing for *Deep Parsimonious Weibull Time-to-Event*, to estimate the optimal number of Weibull distributions, as well as their respective parameters, it needs to combine to model the underlying event-time distribution, unlike to DeepWeiSurv where this number; the mixture size p; is fixed. This novel approach, therefore, guarantees more freedom in modeling the underlying distribution. For this purpose, given an upper bound of the size mixture, the model selects the distributions that have an important contribution to the final mixture, through the weights of a special layer that we call *Sparse Weibull Mixture* described as a filter. In order to stimulate this selection, we penalize these weights by applying a sparse regularization using the $\ell_{0.5}$ norm. The main contributions of this extended approach of DeepWeiSurv are summarized as follows:

- The survival times are assumed to be drawn from a random variable that follows a finite mixture of Weibull distributions.
- The right-censored observations are considered in the conception of the model.
- Without fixing the size of the mixture as done for DeepWeiSurv, but rather by setting

an upper bound p_{max} of the number of Weibull distributions with which we seek to model the underlying distribution, the network-based model learns, through the special layer called Sparse Mixture Weibull, the optimal combination of the Weibull distributions, by estimating the parameters as well as the weighting coefficients of these distributions that compose this mixture and thus the size of the latter (initially set to an upper bound p_{max}).

This chapter is organized as follows. In Section 3.2, we describe the architecture of the DPWTE's network, highlighting the role of the Sparse Weibull Mixture layer in this approach. We also describe our proposed loss function and justify this choice. In section 3.3, we conduct three different simulations on different simulated datasets tested in different scenarios where we evaluate the performance of DPWTE in terms of modeling the relationship covariates-parameters, estimating the weighting coefficients, and finally handling highly censoring setting. We conclude this chapter in Section 3.4.

3.2 Methodology

We remind that the goal is to model a time-to-event distribution with a finite mixture of Weibull distributions. The question that deserves to be asked is with how many Weibull distributions, we compose a mixture that could model a given event-time distribution. In other words, we need to know the estimated value of the mixture size p for the most accurate modeling of a specific survival time distribution. This problem was not identified in the DeepWeiSurv model since p is fixed. Using this approach, we will need to test with different values of p to find the optimal one in terms of the model's performance. In this section, we will present and describe the DPWTE model which proposes a solution to this issue.

3.2.1 Description of DPWTE

As discussed above, DeepWeiSurv fixes the value of p, the number of Weibull distributions used for distribution modelling. Hence, we don't know if this value is the optimal one, and this represents one of the limitations of this approach. DPWTE's approach addresses this issue, it has almost the same architecture and configuration as DeepWeiSurv, but the only difference is that, in the *clf* sub-network, we interleave a new layer called *Sparse Weibull Mixture* layer between the softmax layer and the output layer of *clf* as shown Figure 3.1 As for DeepWeiSurv, this network learns the triplet (α, β, η) that maximizes the likelihood of the mixture distribution. However, we will not necessarily use all the outputs to model the distribution, because we only combine the Weibull distributions whose weights, stored in the Sparse Weibull Mixture layer, are selected in the post-training steps (see Section 3.2.3). This weight selection is monitored by the sparse regularization that we apply on this layer (see the loss function in Section 3.2.4).

3.2.2 Sparse Weibull Mixture Layer

We recall that we seek to find the number of Weibull distributions with which we compose the mixture to have the most accurate modeling of the underlying event-time distribution. In other words, we assume here that p that denotes the mixture size is a variable that we need to estimate where the final mixture size estimate is denoted by $\tilde{p}$ initially set to an upper bound p_{max}, i.e. $p = p_{max}$ before training DPWTE. The latter is set, beforehand, at a sufficiently large value. For this purpose, we introduce the Sparse Weibull Mixture layer. This layer performs an element-wise multiplication of its weights by the softmax layer outputs in the *clf*

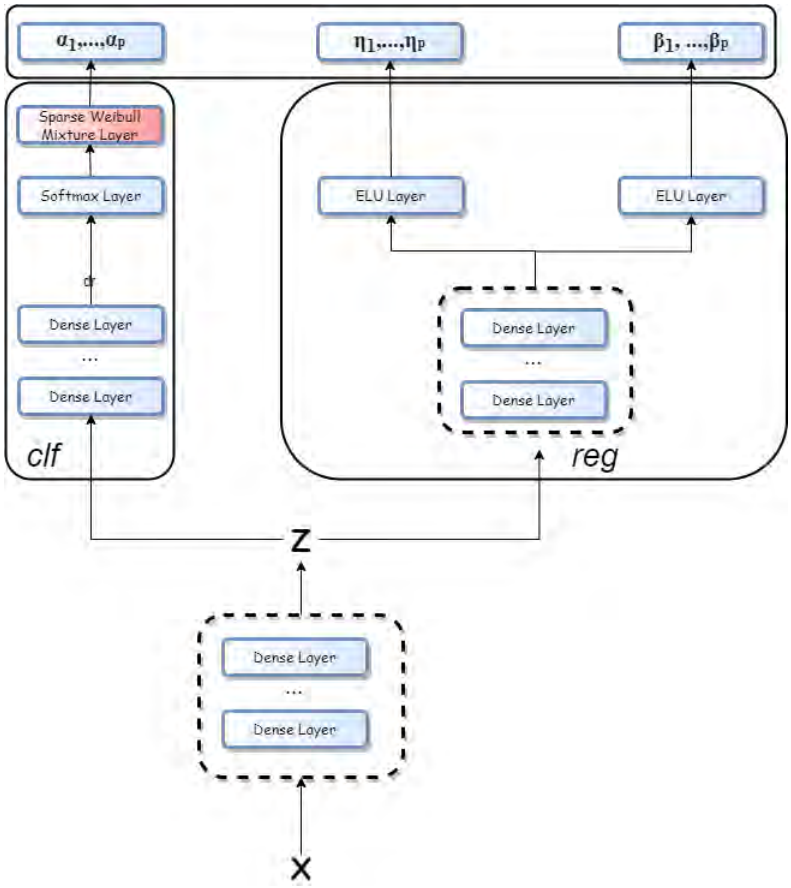


Figure 3.1: The architecture of DPWTE

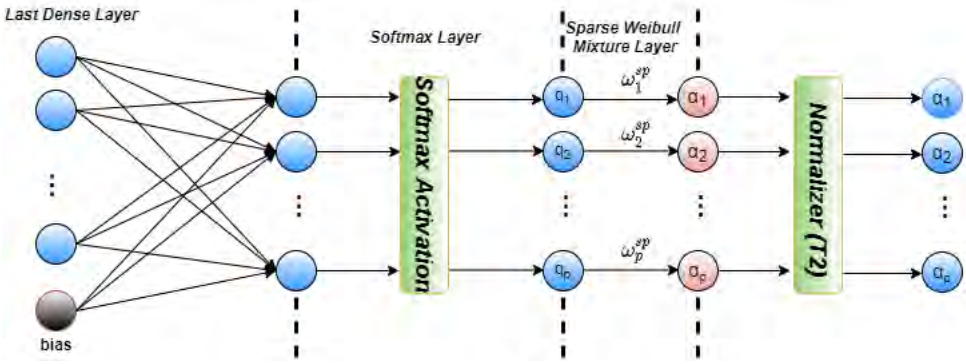


Figure 3.2: Softmax and Sparse Weibull Mixture layers of *clf* sub-network.

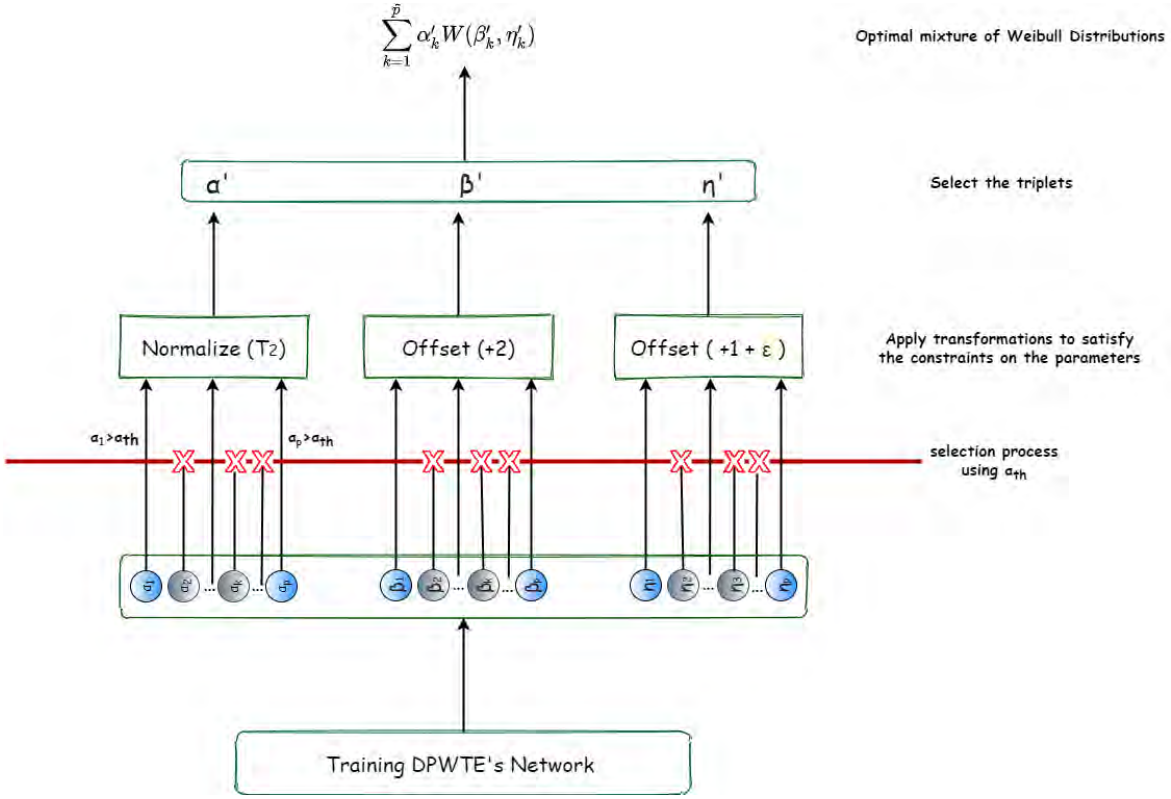


Figure 3.3: Schematic figure depicting the post-training steps of composing the optimal mixture of Weibull distributions.

sub-network. As we see in Figure 3.2 that describes the softmax and Sparse Weibull Mixture layers, we have $\alpha_k = \omega_k^{sp} \odot q_k$, where q_ks are the outputs of the softmax layer (in the case of DeepWeiSurv, $\tilde{p} = p$ and $q_k = \alpha_k, k = 1, .., p$), ω_k^{sp}s are the weights of the Sparse Weibull Mixture layer and $\odot$ denotes the element-wise multiplication. The weights ω_k^{sp}, through the importance of their values, can tell us if $W(\beta_k, \eta_k)$ has a significant contribution to the final mixture. In order to interpret the value of ω_k^{sp} to know which Weibull distributions must be used for the modelling, we need to satisfy the following conditions:

1. $\omega_k^{sp} \in [0, 1]$,
2. $\alpha_k \in [0, 1], k = 1, .., p$,
3. $\sum_{k=1}^{p} \alpha_k = 1$.

The constraint on ω_k^{sp} (1) can't be guaranteed even if we initialize them in that way. The constraints (2) and (3) are established to guarantee that α is a probability vector. As $\alpha_k = \omega_k^{sp} \odot q_k$ and $q_k \in [0, 1]$, we have (1) $\implies$ (2) and thus the latter is automatically checked if (1) is already checked. To ensure implicitly these constraints, we apply the following transformations:

$$(T_1) \quad \omega_k^{sp} \leftarrow \frac{|\omega_k^{sp}|}{\sum_{l=1}^{p} |\omega_l^{sp}|}, k = 1, .., p \tag{3.2.1}$$

$$(T_2) \quad \alpha_k \leftarrow \frac{\alpha_k}{\sum_{l=1}^{p} \alpha_l}, k = 1, .., p. \tag{3.2.2}$$

(T_1) is applied in the calculation of the Sparse Weibull Mixture layer's outputs (before normalizing α, $\alpha_k = T_1(\omega_k^{sp}) \odot q_k$). Whereas (T_2) is an operation performed in the architecture of the model (Normalizer step, see Figure 3.2) to train correctly the network (the loss function uses α as a probability vector) and in one of the post-training steps, after selecting the 'significant' distributions to compose the optimal mixture (α_k become α'_k, see Figure 3.3).

3.2.3 Post-Training Steps: Selection of Weibull Distributions to Combine for Time-to-Event Modelling

So far, we have not yet estimated the value of $\tilde{p}$. The learning phase is the same as for DeepWeiSurv (even if they don't have the same loss function, we will see this in detail in section 3.2.4). However, after the DPWTE's network is trained, we select the triplets $(\alpha_k, \beta_k, \eta_k)$ such as α_k is greater or equal a certain threshold denoted by α_{th} that we fix beforehand. As we change the distribution of α after this selection (before training: $\alpha = (\alpha_1, .., \alpha_p)$ and after training and selection process: $\alpha = (\alpha_k, \alpha_k \geq \alpha_{th})$ but we want to keep the probability constraint, we thus need to apply the transformation (T2) to the new α. Therefore if $\mathbf{A} = \{(\alpha_k, \beta_k, \eta_k) | \alpha_k \geq \alpha_{th}\}$ is the set of selected triplets for modelling, then:

1. $\tilde{p} = Card(\mathbf{A})$
2. $\alpha = (\alpha_k, \alpha_k \geq \alpha_{th}) \underset{T_2}{\longrightarrow} \alpha'$
3. $\beta = (\beta_k, \alpha_k \geq \alpha_{th}) \underset{offset_{(+2)}}{\longrightarrow} \beta'$
4. $\eta = (\eta_k, \alpha_k \geq \alpha_{th}) \underset{offset_{(1+\epsilon)}}{\longrightarrow} \eta'$

Algorithm 1 Post-Training Steps Algorithm

Require: dpwte, α_{th}
Ensure: α', β', η'.
1: $\hat{\alpha}$, β', $\eta' \leftarrow$ empty lists
2: Train DPWTE network dpwte
3: Output the mixture parameters using dpwte: α, β, η
4: **for** $k = 1, .., p$ **do**
5: **if** $\alpha_k \geq \alpha_{th}$ **then**
6: $\beta_k \leftarrow \beta_k + 2$
7: $\eta_k \leftarrow \eta_k + 1 + \epsilon$
8: α'.append(α_k)
9: β'.append(β_k)
10: η'.append(η_k)
11: **end if**
12: **end for**
13: Apply (T_2) on α'

where $offset_{(\gamma)}(\theta) = \theta + \gamma$. The underlying event-time distribution can therefore be modeled by:

$$\sum_{(\alpha_k, \beta_k, \eta_k) \in \mathbf{A}} \alpha'_k W(\beta'_k, \eta'_k).$$

This post-processing is described by Figure 3.3. As illustrated in this figure, the model learns the $2 \times p_{max}$ parameters β_k, η_k and the p_{max} weighting coefficients α_k, then using a threshold α_{th} and after applying the normalization (T_2) and offsets respectively on the weighting coefficients and the parameters, we obtain the selected triplets representing Weibull distributions whose mixture is the optimal one for modeling the underlying time distribution. The algorithm used to execute these post-training steps to find the selected parameters is given in Algorithm 1.

3.2.4 Loss Function

As discussed in previous sections, DPWTE is supposed to find the optimal combination of Weibull distributions. For these reasons, we propose, using notations established in the previous chapter, to train DPWTE with the following loss function:

$$loss = -log\ \mathcal{L}(\beta, \eta, \alpha | (t_i, \delta_i)_i) + \lambda ||\omega^{sp}||_{\frac{1}{2}}, \tag{3.2.3}$$

$$= -(\mathcal{LL}_{\delta=1} + \mathcal{LL}_{\delta=0}) + \lambda ||\omega^{sp}||_{\frac{1}{2}}. \tag{3.2.4}$$

where:

- $\mathcal{LL}_{\delta=1} = \sum_{i=1}^{n} \delta_i\ log \left\{ \sum_{k=1}^{p} \alpha_k S_{\beta_k, \eta_k}(t_i).h_{\beta_k, \eta_k}(t_i) \right\}$ is the contribution of the non-censored observations,
- $\mathcal{LL}_{\delta=0} = \sum_{i=1}^{n} (1 - \delta_i)\ log \left\{ \sum_{k=1}^{p} \alpha_k S_{\beta_k, \eta_k}(t_i) \right\}$ is the contribution of the censored observations,
- λ is the regularization parameter,

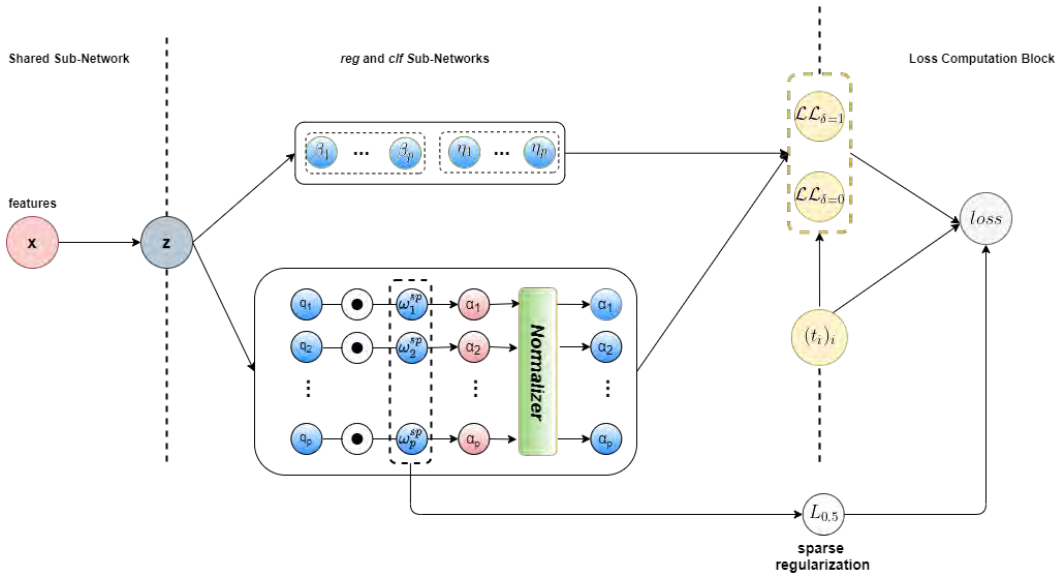


Figure 3.4: Computational graph of the loss function used to train DPWTE.

- and $||\omega^{sp}||_{\frac{1}{2}} = \sum_{k=1}^{p} \sqrt{|\omega_k^{sp}|}$.

The first element of the loss is the negative log-likelihood which is used as a loss function for DeepWeiSurv. We recall that $\mathcal{LL}_{\delta=1}$ is the partial log-likelihood of the recorded times which are non censored, while $\mathcal{LL}_{\delta=0}$ exploits the information provided by the censored times namely the knowledge that the concerned events do not occur before the end of the study.

To stimulate the triplet selection process discussed in the previous section, we apply a *sparse* regularization on $\omega^{sp} = (\omega_k^{sp})_{1 \leq k \leq p}$ the weights of *Sparse Weibull Mixture* layer by adding the penalty term $\lambda||\omega^{sp}||_{\frac{1}{2}}$ to the loss function, hence the name of Sparse Weibull Mixture layer and the word 'Parsimonious' in the name of the model. The purpose behind the sparse regularization is to enforce sparsity in the layer that contains the vector ω^{sp} or at least encourage some ω_k^{sp} to become almost zeros, and in this case, the threshold ω_{th} is applied. Clearly, the ℓ_0 regularizer is ideal for this purpose in the sense of yielding the most sparse weights. However, the ℓ_0 norm is non-differentiable, we, therefore, cannot incorporate it directly as a regularization term in the loss function. C. Louizos et al. [95] proposed a solution through the inclusion of a collection of non-negative stochastic gates, which collectively determine which weights to set to zero but it is a complex optimization problem that is difficult to be solved. The solutions of the ℓ_2 regularizer are smooth, but they do not possess the sparse property. While the ℓ_1 regularizer leads to a convex optimization problem, but it does not yield a sufficiently sparse solution. X. ZongBen et al. [96] proposed $\ell_{0.5}$ norm as the new regularizer which is more sparse than the ℓ_1 regularizer while it is still easier to be solved than the ℓ_0 regularizer. The sparsity property of $\ell_{0.5}$ regularization was demonstrated by Fan et al. [97]. In this approach, we opt for $\ell_{0.5}$-regularization as we see in the proposed loss function. Figure 3.4 is an illustration of the computation graph of the proposed loss function. The inputs are the baseline data of features $\mathbf{x}$ as well as the times recorded $(t_i)_i$ and the outputs are β_k, η_k and η_k used to calculate $\mathcal{LL}_{\delta=1}$ and $\mathcal{LL}_{\delta=0}$. The weights $\omega_1^{sp}, .., \omega_p^{sp}$ are used to calculate the penalty term using the $\ell_{0.5}$ norm.

3.3 Experiments on Simulated Data

In this section, we conduct three experiments on synthetic datasets where the main goal is to empirically investigate our proposed approach. These simulations are by no means exhaustive but are intended to verify that the methodology behind behaves as expected. In the first experiment, we evaluate the ability of DPWTE to learn the relationship between the mixture parameters and the baseline data features. Three scenarios are simulated with different functions of different levels of complexity as well as different mixture sizes. In the second simulation, we perform a clustering based on the weighting coefficient values, as done for DeepWeiSurv in the previous chapter, to evaluate the ability of DPWTE in estimating the weighting coefficients α and the size of the mixture needed to do so. We generate different shapes of datasets namely noisy moons, noisy circles, and noisy blobs. In the third and last experiment, we evaluate the ability of the network in handling the different levels of censoring settings. We test three different shapes of mixture: uni-modal (i,e. with one peak), bi-modal, and tri-modal mixtures. We would point out that these experiments were also run with different values of α_{th} $(10^k, k = -2, -3, -4, -5)$ to test different levels of tolerance and check whether the respective results are better. However, the same patterns were found in these settings, we therefore only keep the following value of the threshold $\alpha_{th} = 0.1$ in this chapter. We would also highlight that $\tilde{p}$ should not always equal the number of Weibull distributions with which we simulate survival times (see an example in Section 3.3.4).

3.3.1 Network Configuration

DPWTE consists of a shared sub-network which is a 4-dense-layer network where the batch normalization is applied immediately following the first fully connected layer. These four hidden layers have 128, 64, 32, and 16 nodes respectively. The regression sub-network has two dense layers with 16 and 8 nodes respectively, whose the second one is batch normalized and two ELU output layers, while the classifier sub-network is composed of 2-dense layers with 16 and 8 nodes respectively, a softmax layer followed by the Sparse Weibull Mixture layer whose weights are initially generated using the uniform distribution of support [0,1]. The hidden layers are activated using the ReLU function. The network is trained via Adam optimizer with a learning rate of 10^{-4}. We initialize the mixture size with $p_{max} = 10$, set the regularization parameter $\lambda = 10^{-4}$ and finally set the threshold $\alpha_{th} = 0.1$ for the post-training operations.

3.3.2 Experiment I: Parameters-Features Relationship Modelling

The purpose of this experiment is to investigate and evaluate DPWTE's ability to model the relationship between the baseline data features and the mixture parameters. Let $\mathbf{X}$ be a vector, of size $n = 5000$, of one-dimensional samples drawn from the uniform distribution $\mathcal{U}_{[0,1]}$ of support [0,1]. We consider three scenarios where we seek to reproduce the relationship between the covariates $\mathbf{X}$ and the mixture parameters with which we draw survival time samples, used to train DPWTE. We assume for this experiment that all the samples are non-censored. The scenarios considered here are defined as follows:

- The survival times samples are drawn from a single Weibull distribution where the parameters are defined as follows:

$$\beta = \mathbf{X}^2 + 2\mathbf{X} + 2 \qquad\qquad \eta = \frac{1}{2}\mathbf{X}^2 + \mathbf{X} + \frac{1}{2}.$$

- We generate survival times from a single Weibull distribution with parameters defined by the following functions:

$$\beta = 2\sin(2\mathbf{X}+1)\sin(1+e^{\mathbf{X}}) + \frac{7}{2} \qquad \eta = \cos(1+\mathbf{X})\, e^{\mathbf{X}^2} + \frac{1}{2}.$$

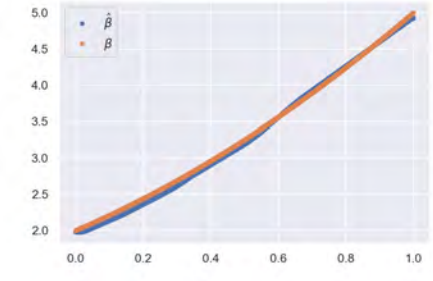
- We generate survival times from a 50-50 mixture of 2 Weibull distribution whose parameters $\beta_1, \eta_1, \beta_2, \eta_2$ are defined by the following functions:

$$\beta_1 = \mathbf{X} + 2 \qquad \eta_1 = \frac{1}{2}\mathbf{X}^2 + \mathbf{X} + \frac{1}{2}.$$
$$\beta_2 = \mathbf{X}^2 + 2\mathbf{X} + 2 \qquad \eta_2 = \frac{1}{2}(\mathbf{X}+1).$$

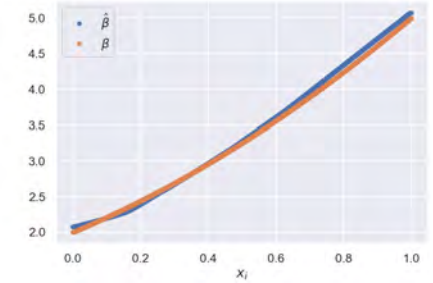
We compare the results of DPWTE and DeepWeiSurv. In the two first scenarios, we use DeepWeiSurv with $p = 1$, while in the last, we use DeepWeiSurv with $p = 2$. We set $p_{max} = 10$ for all scenarios. For each case study, we plot the simulated parameters β, η against their respective estimates. In these defined scenarios, we do not need to plot the estimated expectation since, using this data configuration, it has the same shape as the parameter η as shown in the second simulated experiment of the previous chapter. The results of the first scenarios are displayed in Figure 3.5 and those of Scenario 3 in Figure 3.6.

For the first scenario, we train DPWTE and obtain the mean of $\hat{\alpha}$ over the 5000 instances: $(2.1e{-}5, 4.15e{-}7, 2.9e{-}4, 9.976e{-}1, 3.15e{-}8, 1.8e{-}3, 9.97e{-}10, 4.8e{-}7, 2.57e{-}4, 3.107e{-}10)$, where, by applying the threshold α_{th}, the fourth distribution (average of $\hat{\alpha}_4 = 0.9976$) is the only distribution selected and thus we obtain $\tilde{p} = 1$ which seems logical, since the survival times are drawn from one Weibull distribution. The results of the estimate parameters $\hat{\beta} = \hat{\beta}_4, \hat{\eta} = \hat{\eta}_4$ associated are plotted in Figure 3.5a and Figure 3.5c respectively, and those of DeepWeiSurv associated are shown in Figure 3.5b and 3.5d. We can notice that DPWTE and DeepWeiSurv have practically the same performance. In the second scenario, where the relationship between the Weibull parameters and $\mathbf{X}$ are more complex than that in the first case study for no other reason than that we have a composition and product of exponential and trigonometric functions, whereas in the first scenario, the relationship is polynomial of degree 2. The trained network outputs α whose mean is equal to ($1.0333e^{-5}$, $7.7868e^{-7}$, $5.8331e^{-6}$, $1.5778e^{-4}$, $3.5203e^{-7}$, $9.9988e^{-1}$, $5.4700e^{-6}$, $7.9631e^{-7}$, $6.0515e^{-7}$, $2.6980e^{-5}$), therefore by applying the threshold α_{th}, we obtain $\tilde{p} = 1$ and the post-training steps only select the sixth distribution since α_6 is the only weighting coefficient above the threshold. The results of the estimate parameters $\hat{\beta} = \hat{\beta}_6, \hat{\eta} = \hat{\eta}_6$ associated are plotted in Figure 3.5e and 3.5g respectively, and those of DeepWeiSurv associated are shown in Figure 3.5f and 3.5h. As we notice, non-smooth parts of the curve aside, both models provide a good approximation of η (Figure 3.5g) as for DeepWeiSurv (Figure 3.5h), while for β, they had more difficulty to model the function (Figure 3.5e and Figure 3.5f for DPWTE and DeepWeiSurv respectively).

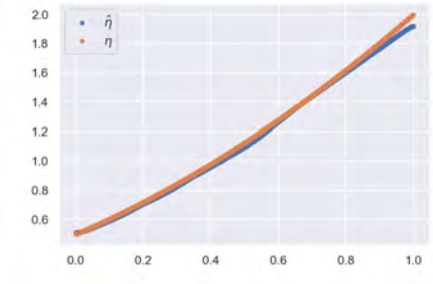
For the last case study, we train DPWTE and obtain the estimation of the weighting coefficients $\hat{\alpha}$ whose mean over the samples is (0.447, 0.46, $9.5e^{-4}$, $7.659e^{-4}$, $8e^{-4}$, 0.02, $5e^{-3}$, 0.065, $4.84e^{-4}$, $9.9e^{-8}$), which means that, after applying the threshold $\alpha_{th} = 0.1$, the process selects the two first distributions to model the simulated mixture, which means that $\tilde{p} = 2$. We notice that the average of $\hat{\alpha}_1$ and $\hat{\alpha}_2$ are close to each other and this implies that after normalizing, we obtain approximately a 50-50 mixture. Now let's describe the results of the mixture parameters selected shown in Figure 3.6. In this figure, we notice that DeepWeiSurv, as shown in Section 2.4.3 of Chapter 2, provides estimate values that



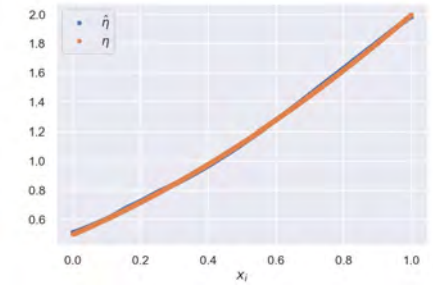
(a) DPWTE: β in Scenario 1



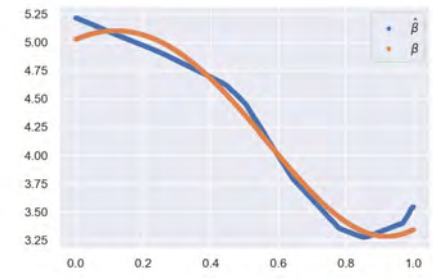
(b) DeepWeiSurv: β in Scenario 1



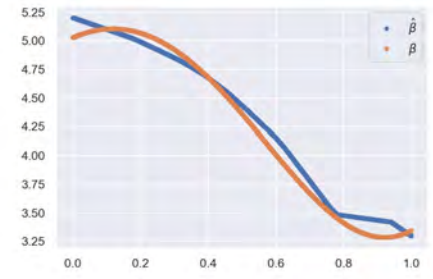
(c) DPWTE: η in Scenario 1



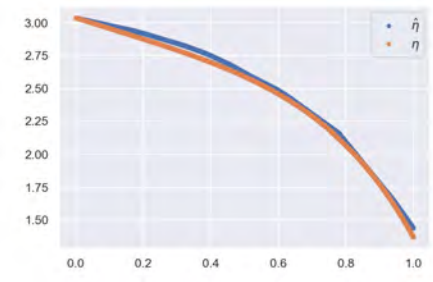
(d) DeepWeiSurv: η in Scenario 1



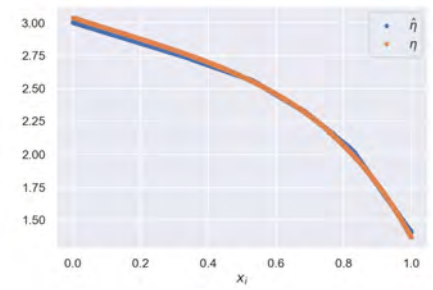
(e) DPWTE: β in Scenario 2



(f) DeepWeiSurv: β in Scenario 2

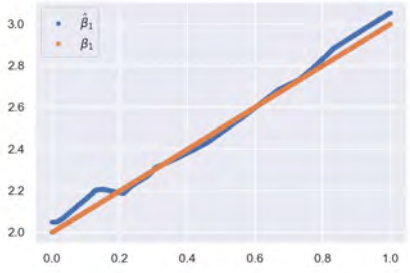


(g) DPWTE: η in Scenario 2

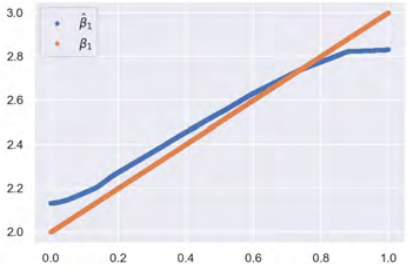


(h) DeepWeiSurv: η in Scenario 2

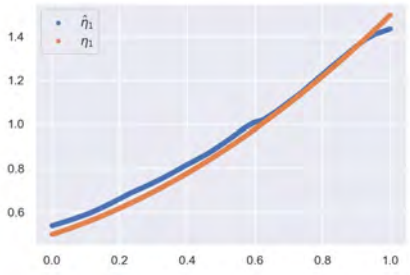
Figure 3.5: Results of DPWTE and DeepWeiSurv in Scenarios 1 and 2.



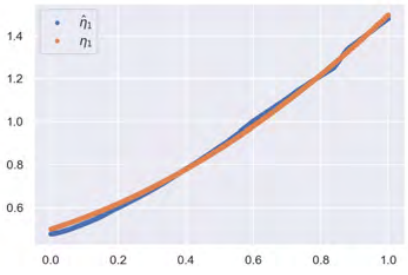
(a) DPWTE: $\hat{\beta}_1$ vs β_1



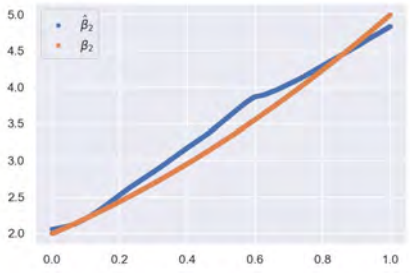
(b) DeepWeiSurv: $\hat{\beta}_1$ vs β_1



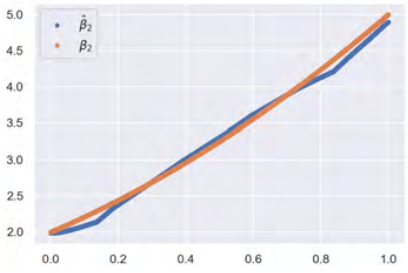
(c) DPWTE: $\hat{\eta}_1$ vs η_1



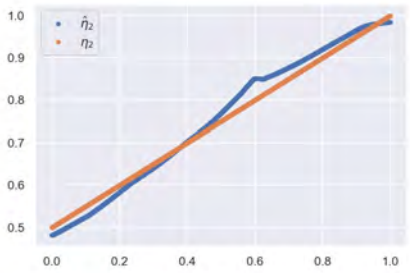
(d) DeepWeiSurv: $\hat{\eta}_1$ vs η_1



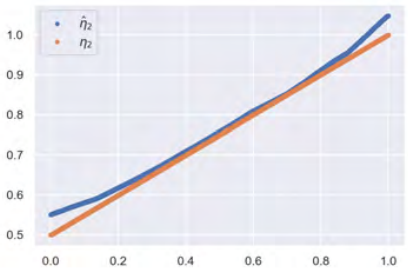
(e) DPWTE: $\hat{\beta}_2$ vs β_2



(f) DeepWeiSurv: $\hat{\beta}_2$ vs β_2



(g) DPWTE: $\hat{\eta}_2$ vs η_2



(h) DeepWeiSurv: $\hat{\eta}_2$ vs η_2

Figure 3.6: Results of DPWTE and DeepWeiSurv for Scenario 3.

practically coincide with the real ones. The same goes for DPWTE which has the same performance as DeepWeiSurv but the only difference is that the model started by modeling the mixture with p_{max}=10 Weibull distributions and found out, after training, that he only needs 2 Weibull distributions to do so which coincide with the exact size of the simulated mixture.

3.3.3 Experiment II: Clustering

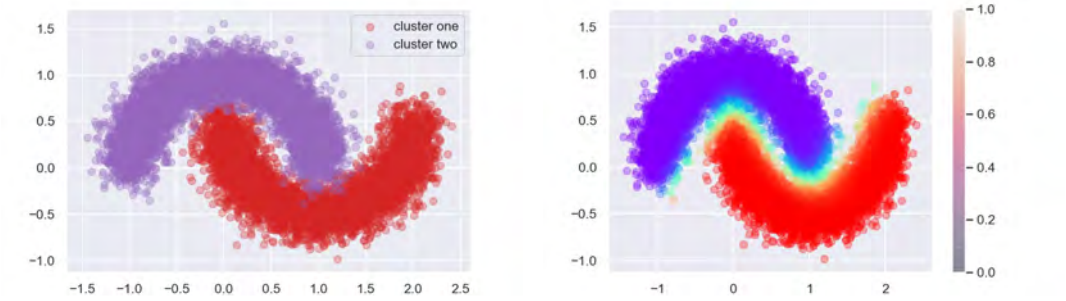
In this experiment, we want to evaluate the ability of DPWTE in estimating the weighting coefficients used to generate the simulated survival times. For this purpose, we consider the problem of clustering in this experiment. The main idea here is to generate m clusters $\mathcal{C}_1, .., \mathcal{C}_m$ not linearly separable. Each cluster $\mathcal{C}_i$ is labeled by a Weibull distribution. In other words, the samples that belong to a cluster $\mathcal{C}_i$ have their respective survival times that are drawn from the same single Weibull distribution of parameters β_i, η_i. We test three case studies:

- We draw 10000 samples using the function *make_moon* (Figure 3.7a) from the scikit-learn package of Python, half of which forms the first cluster $\mathcal{C}_1$ and the second half forms the second one $\mathcal{C}_2$. These clusters are in a shape of a moon in the waxing crescent phase hence the name of the function. We add a standard deviation of Gaussian noise to have thick clusters ($std = 0.15$). We draw for the samples of $\mathcal{C}_1$, time outputs from a single Weibull distribution whose parameters are $\beta_1 = 6.5, \eta_1 = 5$, whereas for the samples belonging to $\mathcal{C}_2$ have their survival times drawn from a Weibull distribution of parameters $\beta_2 = 1.5, \eta_2 = 0.5$.
- We draw 10000 samples using the function *make_circles* (Figure 3.8a) from scikit-learn package of Python, regularly distributed on two clusters $\mathcal{C}_1, \mathcal{C}_2$. These clusters are two nested circles, i.e. a large circle containing a smaller circle in 2d. We apply a standard deviation on 0.1 to generate noisy circles. For each sample i of the small cluster $\mathcal{C}_1$, we draw t_i the time output from a Weibull distribution $\beta_3 = 3.5, \eta_3 = 2.5$ and for each sample j from the bigger cluster $\mathcal{C}_2$, we draw t_j from $W(\beta_4 = 5, \eta_4 = 7.5)$.
- We draw 15000 samples using the function *make_blobs* (Figure 3.9a) from scikit-learn package to generate three isotropic clusters with a Gaussian noise (standard deviation of 0.8), fairly shared among the three clusters $\mathcal{C}_{i=1,2,3}$. $\mathcal{C}_1$ is characterized by $W(\beta_1, \eta_1)$, $\mathcal{C}_2$ is characterized by $W(\beta_2, \eta_2)$, while $\mathcal{C}_3$ is labeled by $W(\beta_3, \eta_3)$.

In each scenario, the Weibull distributions with we which we label the clusters are chosen in a such way that their respective densities are markedly separated in time. As in the previous experiment, we assume that all samples are non-censored. We recall that the objective of this experiment is to check if the model can reproduce the clustering. We train DPWTE in each scenario, and obtain the estimate $\hat{\alpha}_i = (\hat{\alpha}_{i1}, .., \hat{\alpha}_{i\tilde{p}})$ for each sample $i \in \bigcup_{k\in[m]} \mathcal{C}_k$ after applying the threshold $\alpha_{th} = 0.1$ and α-normalization, where $\tilde{p}$ is the estimated size of the mixture predicted and $\hat{\alpha}_{ik} = \mathbb{P}(i \in \mathcal{C}_k)$ is the probability that the sample i belongs to the cluster $\mathcal{C}_k$. Since $\sum_{k=1}^{\tilde{p}} \hat{\alpha}_{ik} = 1$ as well as $\hat{\alpha}_{ij}$ and $1 - \sum_{k\neq j} \hat{\alpha}_{ik}$ are complementary, the latter thus provide the same information. We use the colormap to represent the intensity of the variable to visualize (between 0 and 1).

3.3.3.1 Results and Discussion

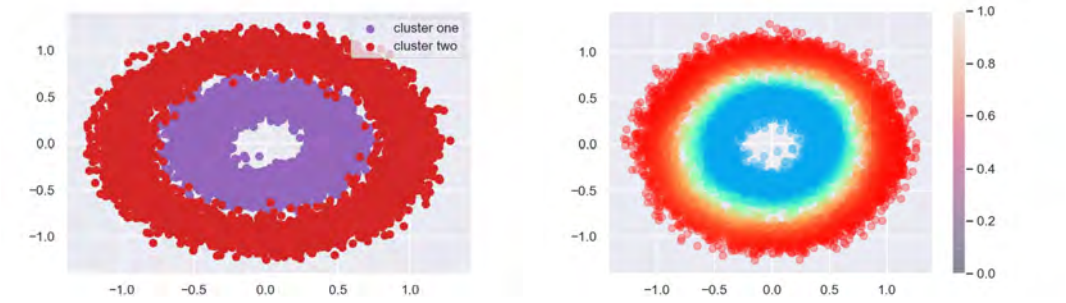
In the two first scenarios, we obtain $\tilde{p} = 2$, whereas, in the third scenario, we obtain $\tilde{p} = 3$, which means that DPWTE estimate exactly the optimal mixture size in these three cases.



(a) The drawn samples

(b) The samples coloured w.r.t $\hat{\alpha}_{i2}$

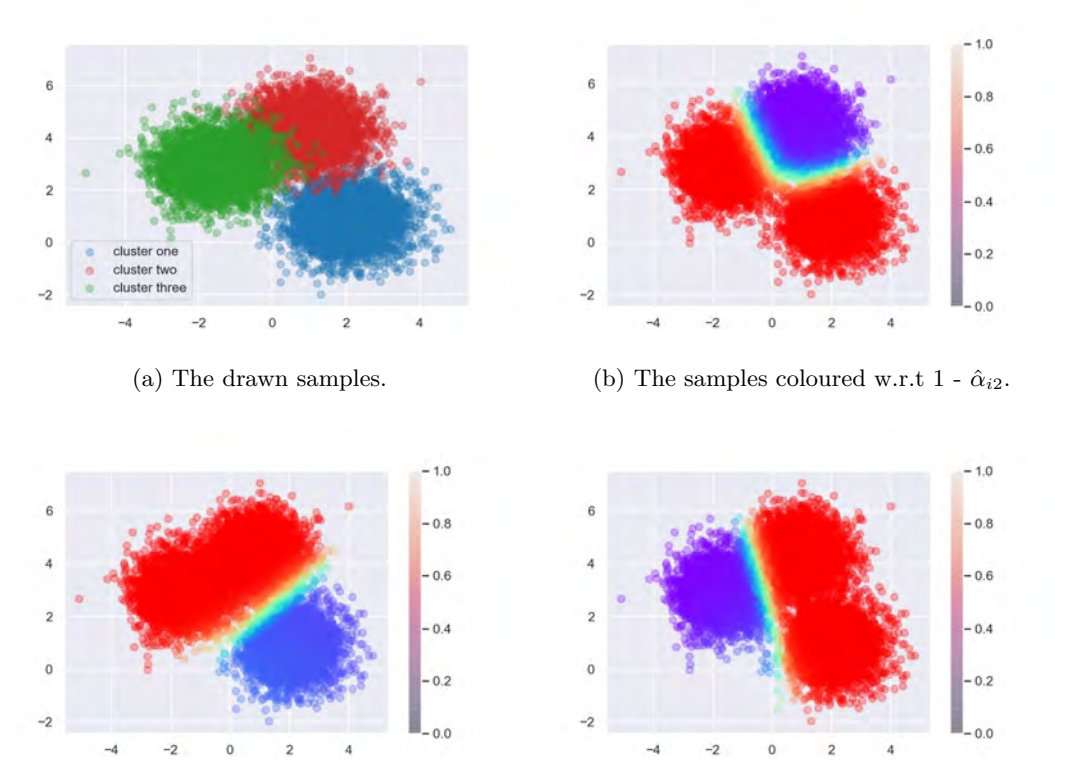
Figure 3.7: On the left, two noisy moons generated using the *make_moon* function from scikit-learn package. On the right, the clustering reproduced by DPWTE using the values of the second component of $\hat{\alpha}$ ($\tilde{p} = 2$).



(a) The drawn samples.

(b) The samples coloured w.r.t $\hat{\alpha}_{i2}$

Figure 3.8: On the left, two nested noisy circles generated using the *make_circle* function from scikit-learn package. On the right, the clustering reproduced by DPWTE using the values of the second component of $\hat{\alpha}$ ($\tilde{p} = 2$).



(a) The drawn samples.

(b) The samples coloured w.r.t 1 - $\hat{\alpha}_{i2}$.

(c) The samples coloured w.r.t 1 - $\hat{\alpha}_{i1}$.

(d) The samples coloured w.r.t 1 - $\hat{\alpha}_{i3}$.

Figure 3.9: Three clusters generated using the *make_blobs* function of scikit-learn package as well as their reproduced clustering performed by DPWTE ($\tilde{p} = 3$).

In Figure 3.7 and Figure 3.8 corresponding to Scenario 1 and 2 respectively, we visualize $(\hat{\alpha}_{i2}, i \in \mathcal{C}_1 \cup \mathcal{C}_2)$. We notice, for the moon-shaped clusters (3.7b), the big majority of samples from $\mathcal{C}_1$ and $\mathcal{C}_2$ have their value of $\hat{\alpha}_{i2}$ associated equal to 0 and 1 respectively. Only the samples that are in the edge of the cluster that faces towards the neighbor cluster have values between 0 and 1. Still, the border points in the edge of $\mathcal{C}_1$ are blue stained or colored in sky blue, which means that $\hat{\alpha}_{i2}$ converges to 0 (according to the colormap), and those of $\mathcal{C}_2$ is rather between yellow and orange, i.e $\hat{\alpha}_{i2}$ converges to 1. Compared to the same clustering performed by DeepWeiSurv in the previous chapter, DPWTE performs the same result starting with $p_{max} = 10$ Weibull distributions and finding the optimal mixture of size $\tilde{p} = 2$ as expected. For the nested-noisy-circle clusters, the samples of the outer noisy circle making $\mathcal{C}_2$ have almost all the value of $\hat{\alpha}_{i2}$ equal to 0.95, except those in the surface to the smaller circle whose $\hat{\alpha}_{i2}$ values are between 0.85 and 0.95 which still a good result (far from 0.5), whereas for the inner circle forming the cluster $\mathcal{C}_1$, the samples belonging to the latter have their respective values of $\hat{\alpha}_{i2}$ not far from zero (colored in a blue sky) except for those that find themselves in the edge facing towards the outer noisy circle which has greater values but still acceptable (between blue sky and green, i.e. $\hat{\alpha}_{i2}$ way below 0.5).

In the third scenario, we have two degrees of freedom since $\hat{\alpha}_{i1} + \hat{\alpha}_{i2} + \hat{\alpha}_{i3} = 1$. We thus visualize for each cluster $\mathcal{C}_k$ the value of $1 - \alpha_{ik}$ (Figure 3.9c for $\mathcal{C}_1$, Figure 3.9b for $\mathcal{C}_2$ and finally Figure 3.9d for $\mathcal{C}_3$). We can notice that the majority of the samples belonging to the two last clusters are correctly labeled (colored in purple) ($1 - \hat{\alpha}_{i2} = 0$ and $1 - \hat{\alpha}_{i3} = 0$ respectively) except the border points on the edge facing toward the two neighbors where the values are not zero but still not far from zero. Whereas the non-border samples of the first cluster are blue coloured, which means that their respective values of $1 - \hat{\alpha}_{i1}$ are not exactly zero ($1 - \hat{\alpha}_{i1}$ between 0.1 and 0.2) and the border points are blue sky coloured, still their respective values far below 0.5 which are acceptable values because by applying a threshold these samples are correctly labeled. We therefore conclude that DPWTE can have a good estimation of the weighting coefficients regardless the distribution of the baseline data and can also provide an improvement when compared to DeepWeiSurv (e.g. in the first scenario).

3.3.4 Experiment III: Censoring Threshold Sensitivity

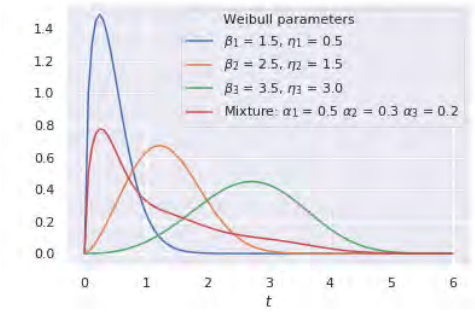
The main objective in this experiment is to evaluate the performance of DPWTE with respect to the censoring rate, denoted by r_c, present in the data, i.e. the size of censored samples against the size of the data. We aim at each scenario defined by a value of r_c, reproduce the distribution simulated. Admittedly, the difficulty of modelling the distribution increases with the censoring rate, but also varies with the shape of the distribution. For this purpose, we run this experiment on three mixture distributions of different shapes:

- *Uni-Modal Mixture*: we draw $m = 10000$ time samples from a 50-30-20 mixture of three Weibull distributions whose parameters are $(\beta_1 = 1.5, \eta_1 = 0.5)$, $(\beta_2 = 2.5, \eta_2 = 1.5)$ and $(\beta_3 = 3.5, \eta_3 = 3)$ respectively with a weighting of 0.5, 0.3 and 0.2 respectively:

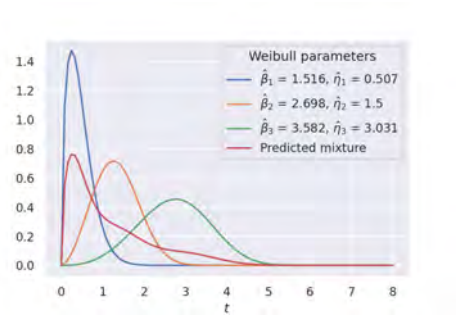
  $$\mathcal{W}_3^1 = 0.5W(\beta_1, \eta_1) + 0.3W(\beta_2, \eta_2) + 0.2W(\beta_3, \eta_3).$$

  The composing Weibull distributions and their mixture are illustrated in densities in Figure 3.10. For this mixture, we test the following values of censoring rate r_c: {0, 0.5, 0.7, 0.85, 0.95, 0.99}.

- *Bi-Modal Mixture*: we draw $m = 10000$ survival time samples from a 40-20-40 mixture of three Weibull distributions of parameters $(\beta_1 = 1.5, \eta_1 = 0.5)$, $(\beta_2 = 2.5, \eta_2 = 1.5)$



(a) Simulated Weibull distributions $\mathcal{W}_3^1$



(b) Predicted Weibull distributions

Figure 3.10: The density of composing Weibull distributions simulated on the left, and those of predicted ones.

and $(\beta_3 = 4.5, \eta_3 = 3.5)$ respectively with a weighting of 0.4, 0.2 and 0.4 respectively:

$$\mathcal{W}_3^2 = 0.4W(\beta_1, \eta_1) + 0.2W(\beta_2, \eta_2) + 0.4W(\beta_3, \eta_3).$$

The composing Weibull distributions and their mixture are illustrated in densities in Figure 3.12a. For this mixture, we test the following scenarios of the value of censoring rate r_c: {0, 0.25, 0.45, 0.55, 0.65, 0.75, 0.85}.

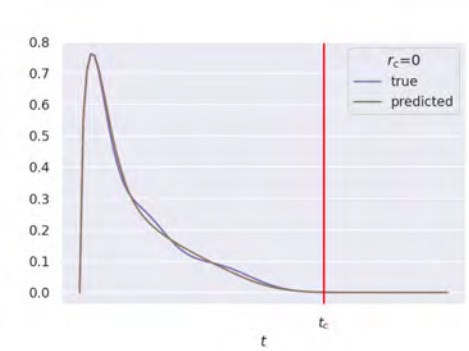
- *Tri-Modal Mixture*: we draw $m = 10000$ samples from a 40-30-30 mixture of three Weibull distributions of parameters $(\beta_1 = 1.5, \eta_1 = 0.5)$, $(\beta_2 = 5.5, \eta_2 = 1.5)$ and $(\beta_3 = 3.5, \eta_3 = 3)$ respectively with a weighting of 0.4, 0.3 and 0.3 respectively:

$$\mathcal{W}_3^3 = 0.4W(\beta_1, \eta_1) + 0.3W(\beta_2, \eta_2) + 0.3W(\beta_3, \eta_3)$$

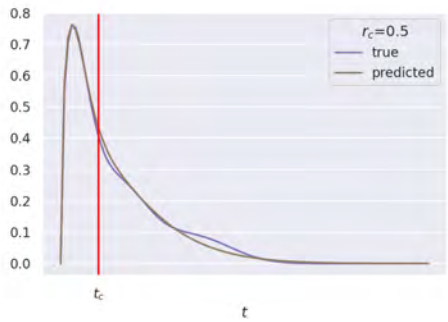
The composing Weibull distributions and their mixture are illustrated in densities in Figure 3.13. For this mixture, we test the following scenarios of the value of censoring rate r_c: {0, 0.1, 0.2, 0.3, 0.45, 0.55, 0.65, 0.85}.

3.3.4.1 Uni-Modal Mixture

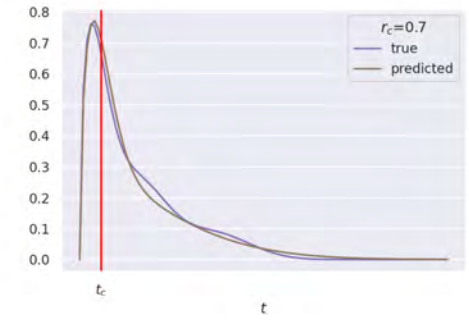
At an initial stage, we train DPWTE on the samples without considering the censoring rate, which means that we assume that all the time samples are non-censored, and obtain the predicted values of the triplets of parameters denoted by $(\hat{\alpha}_i, \hat{\beta}_i, \hat{\eta}_i)_{i=1,2,3}$. We plot, as shown in Figure 3.10b, the predicted Weibull densities as well as the mixture of them using the parameters $(\hat{\beta}_i, \hat{\eta}_i)_{i=1,2,3}$ and the predicted weighting coefficients $(\hat{\alpha}_i)_{i=1,2,3}$. As we can notice in Figure 3.10b and Figure 3.11a, the three distributions are reproduced and the predicted mixture coincides with the simulated one, which means that the mixture parameters and their weighting coefficients are correctly estimated by DPWTE. Now, let's see when we switch a portion of r_c of the data into a censored status. This means that the samples are split into non-censored sub-population (of size $\lfloor (1 - r_c) \times m \rfloor$) and censored sub-population (of size $\lfloor r_c \times m \rfloor + 1$). Then for each scenario defined by a value of the censoring rate, we train DPWTE on the resulting data and compare the mixture distribution with the one predicted by the model, since the goal is to estimate the underlying distribution. We obtain the results for all the values of r_c tested in Figure 3.11. The purple curve corresponds to the probability density function of the mixture simulated while the brown curve is that



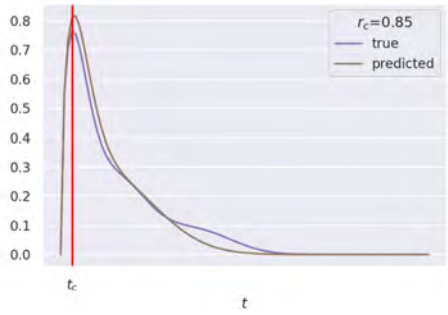
(a) $r_c = 0, \tilde{p} = 3$



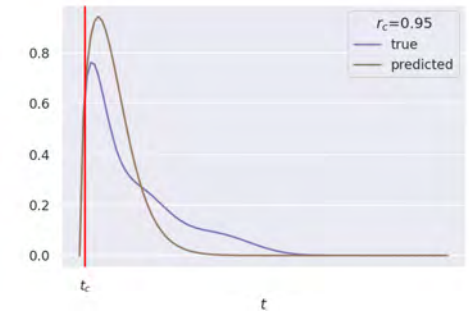
(b) $r_c = 0.5, \tilde{p} = 4$



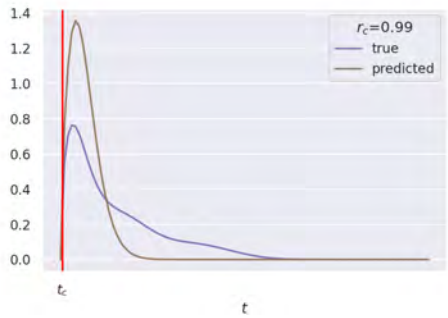
(c) $r_c = 0.7, \tilde{p} = 3$



(d) $r_c = 0.85, \tilde{p} = 2$



(e) $r_c = 0.95, \tilde{p} = 2$



(f) $r_c = 0.99, \tilde{p} = 1$

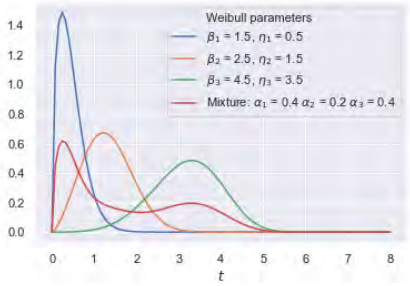
Figure 3.11: Results of the conducted experiment on the uni-modal mixture repeated with different values of censoring rates r_c: densities of the simulated mixture vs. predicted mixture.

of the predicted mixture. We recall that t_c corresponds to the censoring time, defined as the $(1 - r_c)$-th quantile of the vector of simulated times, above which a recorded time is considered as censored. This censoring time r_c is represented in the plots by a red vertical line.

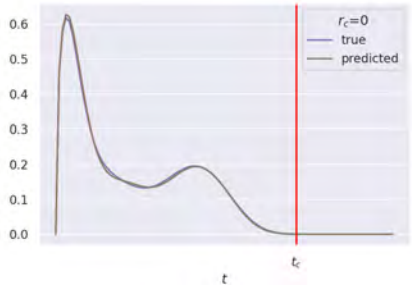
The first thing to notice is that we do not have the same value of the estimate $\tilde{p}$ from a scenario to another. This may be due to the fact that sometimes, the network apportions a Weibull distribution to more than one node or because another Weibull distribution resembles that of the mixture with a weak weighting coefficient but shortly above the threshold α_{th}. We also suspect the decreasing of $\tilde{p}$ (while the censoring rate increases) comes from the fact that the more we increase the censoring rate the more the network ignores a part of a mixture and thus model the mixture with less Weibull distributions than it should be. We will see this example more clearly in the next scenario (bi-modal mixture). Another interesting thing that we can notice is that the precision of prediction decreases as the censoring rate increases which is expected since the increase of the censoring rate implies a loss of information about the overall distribution. Still, the model learns the shape of the distribution regardless of the value of r_c. As we noticed before, when $r_c = 0$, the two curves match up perfectly. For $r_c = 0.5$ and $r_c = 0.7$, the two densities almost coincide with each other. The same goes for $r_c = 0.85$ even the model does not have information about the tail of the distribution, but it predicts earlier density mitigation. For the extremely highly censoring setting namely the cases $r_c = 0.95$ and $r_c = 0.99$ (see Figure 3.11e and Figure 3.11f), the peak is overestimated which implies an early density attenuation. Still, the peak is well located even if it is not observed (the red vertical line is placed before the peak in these two cases). We can conclude, as in the experiment of the uni-modal mixture conducted by DeepWeiSurv in the previous chapter, that DPWTE can provide promising results in terms of handling the highly censoring setting as shown in this experiment.

3.3.4.2 Bi-Modal Mixture

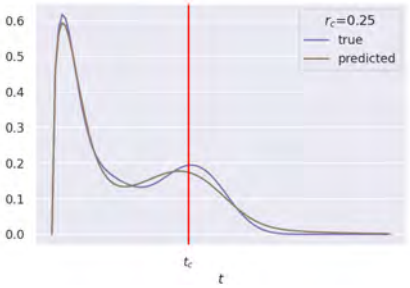
We conduct a similar experiment on the bi-modal mixture $\mathcal{W}_3^2$ described above, but with different values of r_c. We notice from Figure 3.12b that the simulated and predicted densities coincide with each other when $r_c = 0$. In this scenario, the model combines the same number of Weibull distributions to model the simulated one ($\tilde{p} = 3$). In the case where we have 25% of censored samples, DPWTE estimates the density of the mixture using three Weibull distributions ($\tilde{p} = 3$) with a very slight degradation of the precision (the second peak is slightly underestimated and slightly shifted on the left), however, it learns the underlying shape of the distribution and the positions of the peaks. For $r_c = 0.45$, the model predicted only the first peak which seems logical as the second peak is way located after the censoring time t_c. In fact, the model learned the first two distributions (hence $\tilde{p} = 2$) but completely ignored the second peak. Actually, the third highest value of α before normalization was of the order of 0.06 (less than the threshold 0.1), which corresponds to the third distribution that the model could not learn because of the high censoring. Still, the observed curve (before t_c) is perfectly estimated. The same goes for $r_c = 0.55$ and $r_c = 0.65$ where the model used 2 distributions, but it further ignores the tail of the distribution as the censoring time t_c further backward. For the last two cases, the respective predicted densities still ignoring the second peak while having earlier density mitigation. Compared to the uni-modal scenario, the degradation of the model performance (in terms of handling highly censoring setting) occurs before in this scenario. This is because, when the ratio of censored samples is important, the bi-modal mixture is considered more complex to learn than the uni-modal especially when the two peaks are largely separated in time.



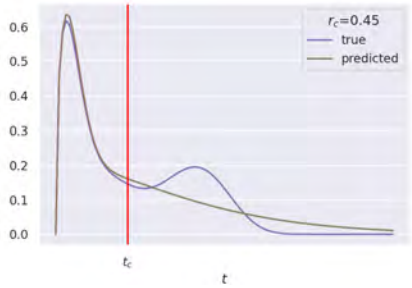
(a) Densities of $\mathcal{W}_3^2$.



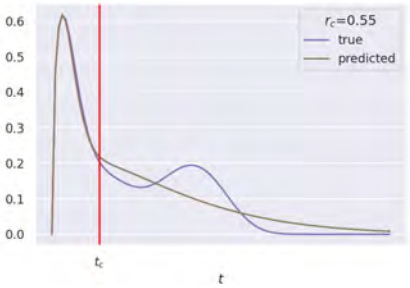
(b) $r_c = 0., \tilde{p} = 3$



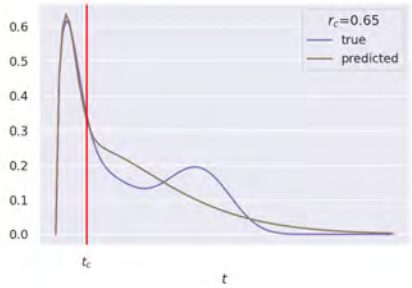
(c) $r_c = 0.25, \tilde{p} = 3$



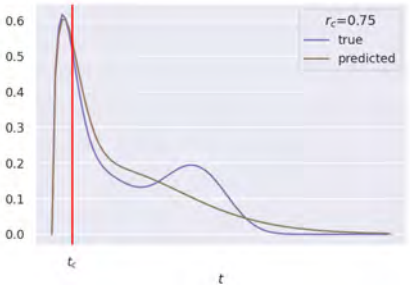
(d) $r_c = 0.45, \tilde{p} = 2$



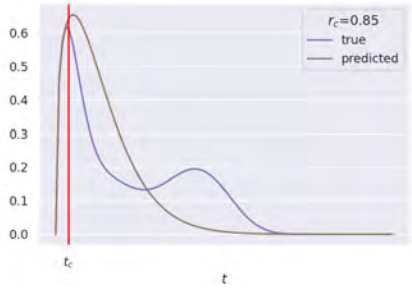
(e) $r_c = 0.55, \tilde{p} = 2$



(f) $r_c = 0.65, \tilde{p} = 2$



(g) $r_c = 0.75, \tilde{p} = 2$



(h) $r_c = 0.85, \tilde{p} = 1$

Figure 3.12: Results of the conducted experiment on the bi-modal mixture repeated with different values of censoring rates r_c: densities of the simulated mixture vs. predicted mixture.

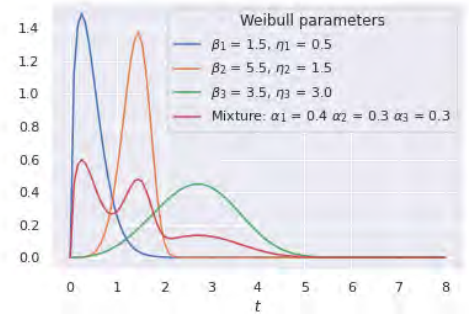


Figure 3.13: Densities of $\mathcal{W}_3^3$ and its composing distributions.

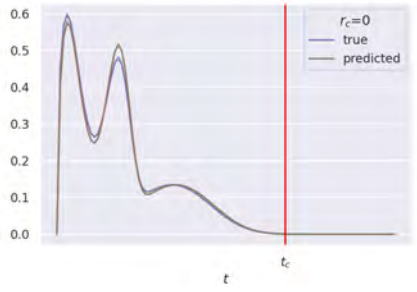
3.3.4.3 Tri-Modal Mixture

The same experimental protocol is executed on the tri-modal mixture $\mathcal{W}_3^3$ with different values of r_c. This means that the complexity of the problem grows even further. The third peak here appears but with a sharpness less important as seen in Figure 3.13. In the scenario where all the samples are non-censored, the model estimates with good precision the mixture with its three peaks correctly located. For $r_c = 0.1$, the model correctly predicted the peaks with their respective magnitudes and positions in time. For $r_c = 0.2$ and $r_c = 0.3$, the third peak starts to disappear, while the density values before this peak are perfectly estimated (the purple and brown curves coincide from t=0 to t=2 which corresponds to the interval that contains the first two peaks). For the cases $r_c = 0.45$ and $r_c = 0.55$, even if the second peak is not observed (it is placed after the red vertical line) the model predicts the latter with a very slight shift on the right but a significant overestimation of the magnitude which speeds up the density mitigation. For the last two cases, the precision of the density estimation continues to decline after the first peak, but still perfectly estimating the part of the curve corresponding to the observed times.

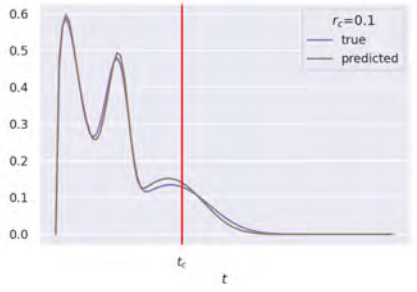
3.3.4.4 Summary Results

To sum up, these three datasets represent different levels of difficulties in terms of modeling the mixture in a highly censoring setting. The difficulty of modeling in the presence of censored data depends on the number of peaks, their respective positions in time, and also their respective magnitudes. For example, let's take an example of two bi-modal Weibull distributions whose peaks have respectively the values (0.6,0.2) and (0.4,0.4). Since the magnitude of the second peak of the second Weibull is greater than that of the first Weibull. The latter is more likely to be ignored than that of the second Weibull because for a given value of r_c, the t_c associated is lower in the first case and thus the model further loses information.

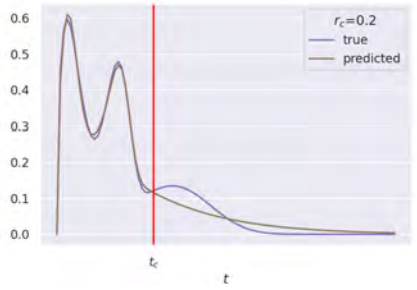
In general, we can say that DPWTE handles censoring samples at a varying portion that depends on the shape of the distribution. For instance, in the uni-modal mixture, even with 99% of censoring, the model is considered to be well-performing until $r_c = 0.85$ but still learn the underlying shape of the distribution as well as the position of the peak, while in the bi-modal the deterioration in the quality of estimates starts earlier with notably $r_c = 0.45$. Finally, for the tri-modal mixture, the model has difficulties even earlier (it ignored the last peak from the case $r_c = 0.2$ and overestimated the magnitude of the second peak from the case $r_c = 0.45$ until it ignored it in the case $r_c = 0.85$), but in most cases, the position of



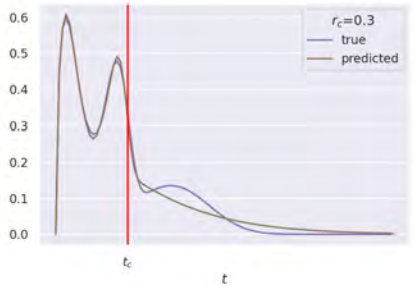
(a) $r_c = 0, \tilde{p} = 3$



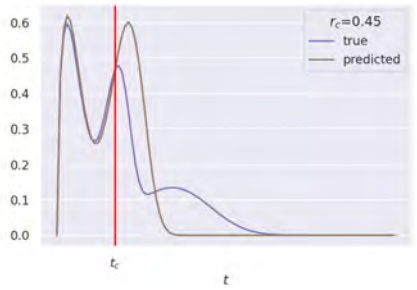
(b) $r_c = 0.1, \tilde{p} = 3$



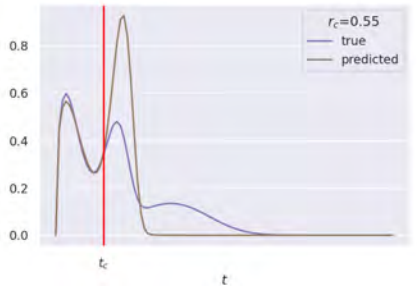
(c) $r_c = 0.2, \tilde{p} = 3$



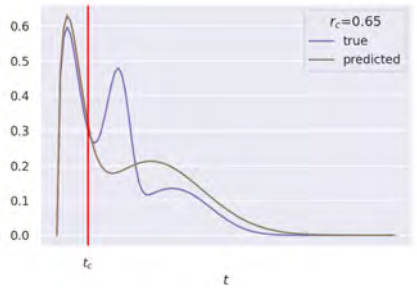
(d) $r_c = 0.3, \tilde{p} = 3$



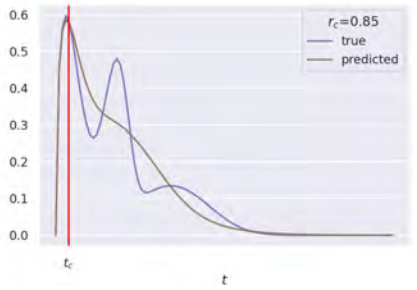
(e) $r_c = 0.45, \tilde{p} = 2$



(f) $r_c = 0.55, \tilde{p} = 2$



(g) $r_c = 0.65, \tilde{p} = 2$



(h) $r_c = 0.85, \tilde{p} = 2$

Figure 3.14: Results of the conducted experiment on the tri-modal mixture repeated with different values of censoring rates r_c: densities of the simulated mixture vs. predicted mixture.

the second peak is well predicted even if it is not observed.

3.4 Conclusion

In this chapter, we presented an extended approach of DeepWeiSurv for survival analysis called DPWTE. This network-based approach can model complex relationships between the covariates of the baseline data and the event times, assuming that the latter is drawn from a mixture of Weibull distributions. This means that DPWTE leverages Weibull advantages, namely the fact that these distributions are known to be a good representation for survival time distribution and it also to consider any time horizon, we indirectly learn a continuous probability density function, via the mixture parameters that the model learns by maximizing the likelihood of the mixture. The main contribution of DPWTE relative to DeepWeiSurv is that it does not fix the size of the Weibull mixture with which we model the underlying distribution but rather initializes it by an upper bound and finds the optimal combination of Weibull distributions using a 'filter' layer that we call Sparse Weibull Mixture layer. To stimulate the process of selecting the significant contribution distributions, we applied a sparse regularization on the weights of this layer. We conducted three experiments with different scenarios. In the first experiment, we evaluated the performance of DPWTE in terms of learning the relationship between the covariates and the mixture parameters and compared it with that of DeepWeiSurv. To do so, we tested three functions with different levels of difficulty and found that DPWTE provides an improvement in estimation over DeepWeiSurv, especially in the most complex function where the variance of the estimation decreased. The second experiment was dedicated to the weighting coefficients estimation which gives us information about the importance of the contribution of each Weibull distribution in the mixture. We ran a clustering experiment using three simulated clustering benchmark datasets with different shapes. The goal was to evaluate the ability of the proposed model to estimate the weighting coefficients of the samples considering that all samples from a given cluster should have the same weighting coefficient. DPWTE showed an improvement in relation to DeepWeiSurv (illustrated by the moon cluster). Finally, the last experiment was conducted with the aim of evaluating the sensitivity of the model towards the importance of the censoring rate. We tested three shapes of distribution namely uni-modal, bi-modal, and tri-modal mixture where DPWTE showed that it handles the extremely highly censored setting in the uni-modal distribution, however, for bi-modal, the model performance is altered because the model tends to ignore a part of the distribution (notably the second peak) when the censoring rate increases and the same goes for the tri-model with more alteration for the same reasons. Still, the model can handle a significant rate of censoring. So far, we described the two approaches: DeepWeiSurv in the previous chapter and DPWTE in the current one, we conducted different experiments on simulated data to verify different properties. In the next chapter, we will describe other experiments that we have done on real-world benchmark datasets where we compare the respective performances of DeepWeiSurv and DPWTE with those of the state-of-the-art. We will also see an application of these two approaches in time series.

Chapter 4

Experiments on Real-World Benchmark and Simulated Time-Series Datasets

4.1 Introduction

In Chapter 2 and Chapter 3, we presented two deep learning approaches to survival analysis namely DeepWeiSurv as well as its extended approach DPWTE. We described their respective architectures and conducted different simulated experiments to test and evaluate the different properties of these two models. In this chapter, we will run experiments on real-world datasets where we compare the predictive performance of the two network-based approaches, one with another as well as with that of the most known state-of-the-art models. To compare our models with the competing models that we propose here, we use two evaluation metrics: concordance index [53, 56] and Brier score [58, 59]. We will also conduct a time-series experiment on NASA Turbofan Jet Engine Data Set including Run-to-Failure simulated data from turbofan jet engines. The goal of this dataset is to predict the remaining useful life (RUL) of each engine in the data. To do so, we propose to combine Long-Short Term Memory (LSTM) with DeepWeiSurv and DPWTE respectively for this experiment where we compare the predictive performance of these two combinations not only with each other but also with the predictive performance of a fully connected deep network combined with LSTM. This chapter is split into two parts:

1. *Experiments on Real-World Benchmark Data Sets*: we conduct two sets of experiments, with the same experimental protocol, to compare the proposed models with the state-of-the-art competing models:

    - In the first instance, we apply the experimental protocol (that we will define) on all models with the proposed benchmark datasets and compare their respective predictive performances in each dataset.
    - Second, we apply the same protocol as before but with changing the censoring rate of the datasets. This changes the distribution of the censored and observed time events and thus may bring an additional challenge to risk prediction. In this experiment, we aim (as in the last experiments of both Chapter 2 and Chapter

3 but with real-world datasets) to evaluate the sensitivity of the models towards the highly censored settings.

2. *Experiments on Simulated Time Series*: we conduct an experiment on a simulated data called Turbofan Jet Engine Dataset using three models which are a combination of LSTM with a fully connected network, DeepWeiSurv and DPWTE respectively. We compare these models, by evaluating their RUL predictions on test data as well as their *Mean Absolute Error* values.

This chapter is organized as follows: in Section 4.2, we describe the experiments conducted on real-world benchmark datasets. Section 4.3 is devoted to the simulated experiments ran on Turbofan Jet Engine dataset. We conclude in Section 4.4.

4.2 Experiments on Real-World Data

In this section, we evaluate our proposed models on real-world datasets and compare their predictive performance with that of the competing models from the state-of-the-art methods. The datasets considered here are the following: SEER Breast Cancer, SEER Heart Disease, METABRIC, SUPPORT and FLCHAIN.

4.2.1 Description of the Real-World Datasets

Here, we briefly define the real-world datasets used in this section and we also visualize the respective distributions of the most important variables for each dataset. We give an overview on descriptive statistics of these five datasets in Table 4.1.

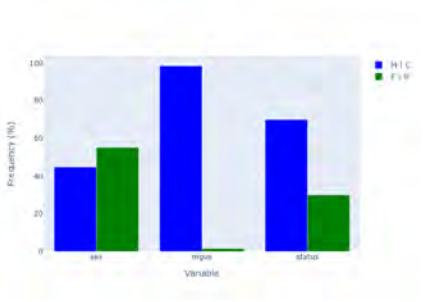
4.2.1.1 FLCHAIN

Assay of Serum Free Light Chain [98] is a database used to study the relationship between serum free light chain (FLC) and mortality. We used a stratified random sample containing 50% of the subjects from this database. We extracted 8 variables which are the following: age, sex, kappa and lambda portion of serum free light chain, FLC group for the subject, serum creatinine and monoclonal gammapothy indicator denoted by mgus which checks if the subject has been diagnosed with mgus. The respective distributions of all these variables as well as that of the event time target variable called futime are shown in Figure 4.1.

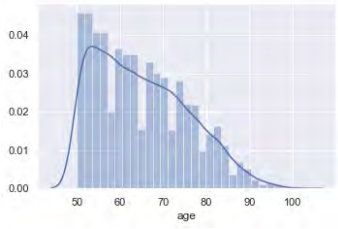
4.2.1.2 METABRIC

The Molecular Taxonomy of Breast Cancer International Consortium[1] (METABRIC) database is a Canada-UK Project which contains targeted sequencing data of 1980 primary breast tumor samples of which 888 are non-censored [99]. This database contains gene expressions and clinical features including age at diagnosis, stage and size of the tumor, positive and removed lymph nodes, and the Nottingham prognostic index (NPI) which is used to determine prognosis following surgery calculated using the size of the tumor, the number of lymph nodes and the grade of the tumor. It also contains categorical variables such as ER immunological historical chemistry (IHC) status, Progesterone Receptor status (PR), etc. In figure 4.2, we show the most important variable distributions as well as the distribution of the event time target variable.

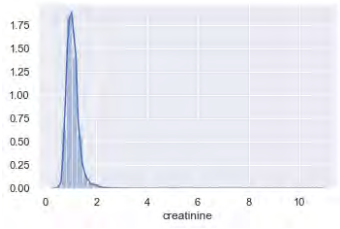
[1] https://ega-archive.org/studies/EGAS00000000083



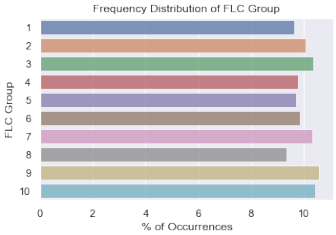
(a) Sex, mgus and status variables



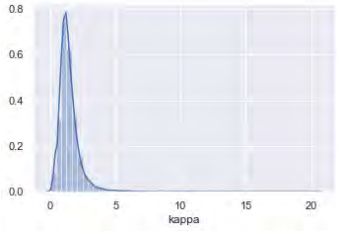
(b) Age



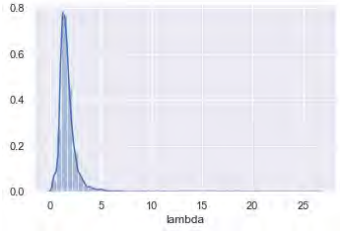
(c) Creatinine



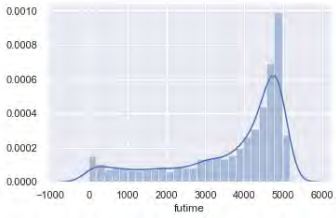
(d) FLC Group



(e) Kappa



(f) Lambda

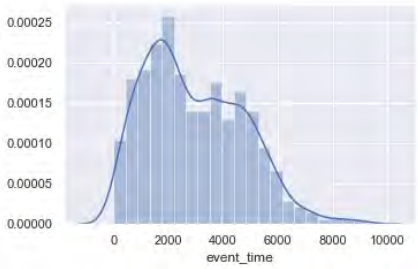


(g) Event times

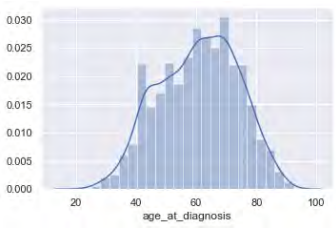
Figure 4.1: The distributions of FLCHAIN variables.

Table 4.1: Descriptive Statistics of Real-World Datasets

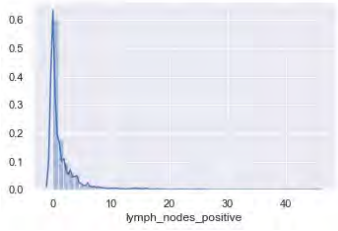
Datasets	No. Uncensored	No. censored	No. Features	Censoring Time			Event Time		
				min	max	mean	min	max	mean
SEER BC	9152(42.8%)	12221 (57.2%)	34	1	227	181.5	1	226	63.7
SEER HD	12014 (49.6%)	12221 (50.4%)					1	224	76.7
FLCHAIN	2169(27.6%)	5705(72.4%)	8	1	5215	4226.2	0	4998	2174.5
SUPPORT	5844(68.1%)	2735(31.9%)	36	344	2029	1060.2	3	1944	206.0
METABRIC	888 (44.8%)	1093 (55.2%)	21	1	308	116.0	1	299	77.8



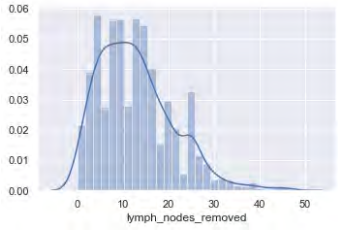
(a) Event times



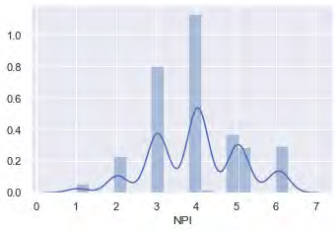
(b) Age



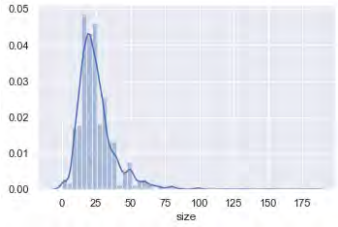
(c) Positive lymph nodes.



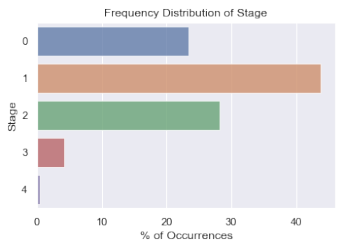
(d) Removed lymph nodes.



(e) NPI

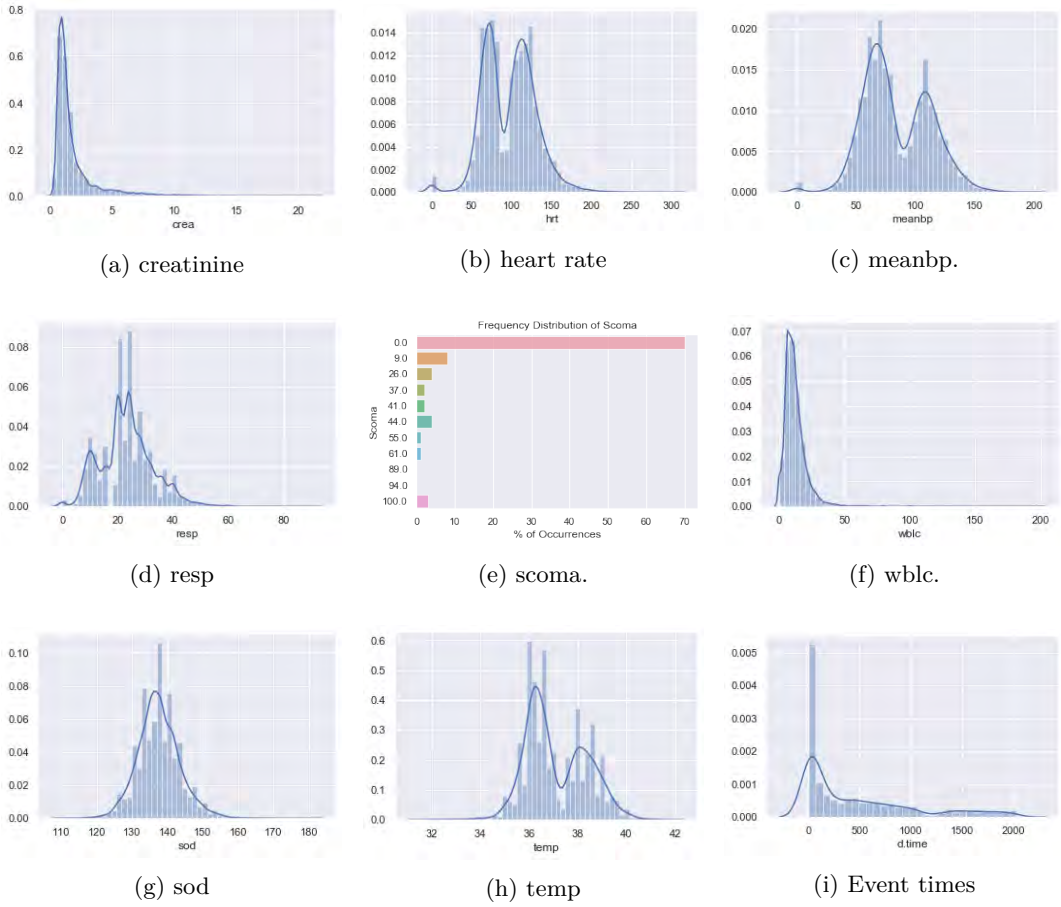


(f) Tumor size



(g) Stage

Figure 4.2: The distributions of METABRIC variables.



(a) creatinine (b) heart rate (c) meanbp.

(d) resp (e) scoma. (f) wblc.

(g) sod (h) temp (i) Event times

Figure 4.3: The distributions of SUPPORT variables.

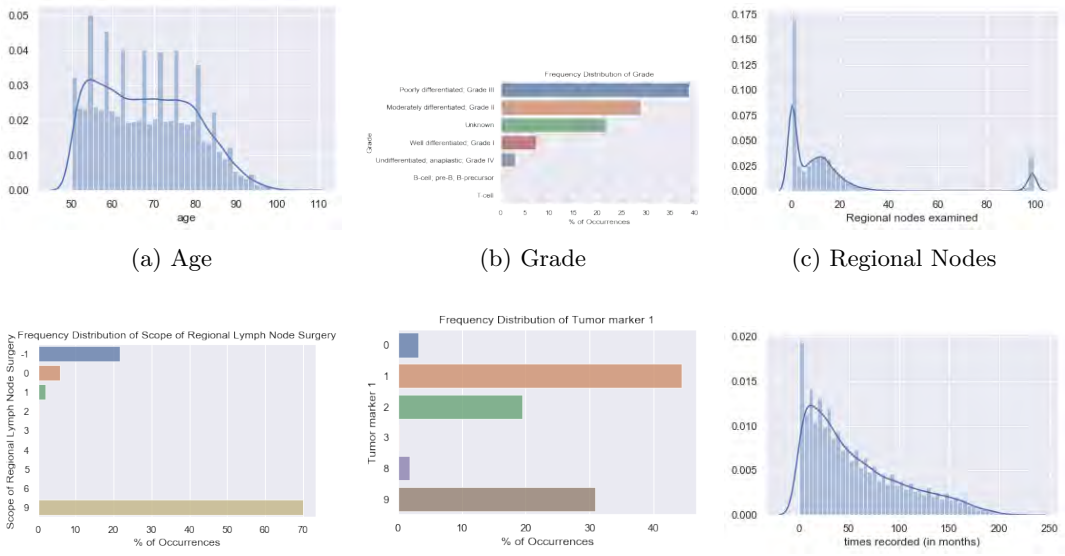
4.2.1.3 SUPPORT

SUPPORT which stands for Study to Understand Prognoses Preferences Outcomes and Risks of Treatment is a larger study that researches the survival time of seriously ill hospitalized adults [100]. The SUPPORT dataset consists of 9105 patients under study, of which 68.1% died during the survey with a median death time of 58 days, described with 36 non-correlated attributes including age, mean arterial blood pressure (meanbp), heart rate, temperature (temp), respiration rate (resp), white blood cell count (wblc), Glasgow coma scale (scoma), serum's sodium (sod) and serum's creatinine. The distributions of these variables are displayed in Figure 4.3.

4.2.1.4 SEER

The Surveillance, Epidemiology and End Results (SEER)[2] [62] Program provides cancer incidence data from population-based cancer registries covering approximately 34% of the U.S. population. Here, we focused on the patients recorded between 1998 and 2002 who have Breast Cancer (BC), Heart Disease (HD), or who have survived to the end of this period.

[2]https://seer.cancer.gov



(a) Age (b) Grade (c) Regional Nodes

(d) Scope of RLNS (e) Tumor marker 1 (f) Event times

Figure 4.4: The distributions of SEER BC variables.

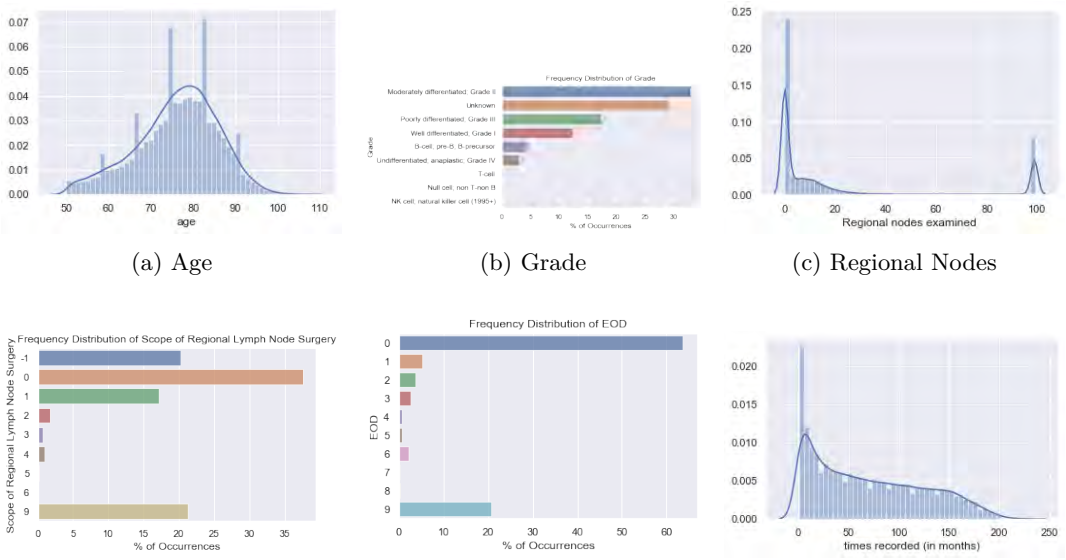
We kept 34 covariates including:

- Age of patient.
- Tumor grade.
- Tumor size.
- Regional nodes examined: records the total number of regional lymph nodes that were removed and examined by the pathologist.
- Scope of RLNS: scope of Regional Lymph Node Surgery describes the procedure of removal, biopsy, or aspiration of regional lymph nodes performed during the initial work-up or first course of therapy at all facilities.
- Tumor marker 1: records prognostic indicators for breast cancer cases.
- EOD 10-nodes: records the highest specific lymph node chain that is involved by the tumor.
- Progesterone and Estrogen Receptor Status.

We generated from this database two single-event datasets namely: SEER BC and SEER HD corresponding to the Breast Cancer and Heart Disease events respectively. The distributions of the most important variables in SEER BC and SEER HD as well as that of the event times variable are displayed in Figure 4.4 and Figure 4.5 respectively.

4.2.2 Methods

In Chapter 2 and Chapter 3, we presented two new survival methods based on a Weibull distribution assumption and neural networks namely DeepWeiSurv and its extended version DPWTE. In the following experiments, we consider the models above:



(a) Age (b) Grade (c) Regional Nodes

(d) Scope of RLNS (e) EOD (f) Event times

Figure 4.5: The distributions of SEER HD variables.

- Cox Proportional Hazards Model (CPH) [29] with a penalty term in the order of 10^{-1}.
- Weibull Accelerated Failure Time Model (Weibull AFT).
- Random Survival Forests (RSF) [54] with number of trees set to 100.
- DeepSurv [31] with 2 layers of 32 nodes.
- DeepHit [30] with a dropout probability of 0.6 between all the hidden layers. We used the TensorFlow implementation.
- DeepWeiSurv: We test DeepWeiSurv with two possible values of $p = 1, 10$.
- DPWTE: trained with $p_{max} = 10$ and $\lambda = 10^{-4}$.

We will evaluate the performance of these models and compare our approaches with those of the state-of-the-art. We do not test the method proposed by Luck et al. [42] because it belongs to the proportional hazard models family and is therefore restricted in all the same ways as CPH and other proportional methods. For the network-based methods, we standardize the numerical covariates and encode the categorical ones, while for RSF, we directly used the categorical variables without any transformations. For DeepWeiSurv and DPWTE, we use the same network configurations as described in Chapter 2 and Chapter 3 respectively. All the methods are trained via Stochastic Gradient Descent optimizer with a learning rate of 10^{-4}.

4.2.3 Experiment One

In the first experiment, we will apply the experimental setting, as described below, executed on all the considered models (our proposed methods and the competing ones) with the benchmark datasets described above and will discuss their respective performances in terms of survival time predictions ordering and error of probability predictions.

4.2.3.1 Experimental Protocol

For evaluation, we applied 5-fold cross-validation: the data is randomly split into the training set and validation set (80-20 split). For each iteration, the models are fitted by the corresponding training set (4 folds) and evaluated on the validation set (1 fold) by calculating C^{td} and IBS using the duration of the validation set as the time span. Once all iterations are executed, we obtain for each method and for each dataset, two vectors (of size 5) containing C^{td} and IBS for each iteration.

4.2.3.2 Results and Discussion

The results are summarized in Table 4.2 and Table 4.3 where we calculated the confidence interval and the average of the concordance index scores and IBS scores respectively, over the five cross-validation folds. In METABRIC, DeepHit and our proposed models provided a significant improvement in terms of concordance when compared to other competing methods, especially DPWTE, using $\tilde{p} = 1$ Weibull distributions that have the mean concordance index slightly exceeding that of DeepHit and DeepWeiSurv ($p = 1, 10$), but it still has wider interval confidence. We can say that for METABRIC, DeepHit and DPWTE have practically the same ordering performance (when we take into account the trade-off between the mean and the variance of C^{td}). For the SUPPORT dataset, DeepHit outperforms the other models in terms of times ordering, but both DPWTE, using $\tilde{p} = 3$ Weibull distributions, and DeepSurv minimized the difference between their respective concordance and DeepHit's one compared to RSF, CPH, and Weibull AFT. The same remark goes for FLCHAIN, but with a slight decrease of the scores for all the models and DPWTE only used $\tilde{p} = 1$ Weibull distribution. In the SEER dataset, for Breast Cancer and Heart Disease populations alike, we can notice that both DPWTE, with $\tilde{p} = 2$, and DeepWeiSurv, showed a large significant out-performance compared to the competing methods, with a slight improvement from DeepWeiSurv with $p = 1$ to DPWTE. We can also remark that the standard deviation of C^{td} for METABRIC data is relatively greater than that of SEER, FLCHAIN, and SUPPORT datasets. We suspect that this could come from the small size of METABRIC data regarding the other datasets. Furthermore, another thing to point out is that for all the datasets, except METABRIC, the respective confidence intervals of DPWTE and DeepWeiSurv are narrower than those of the competing methods, which means that our proposed methods produced a more stable estimation. DPWTE seems to have the best discriminative performance overall. Now, let's analyze the calibration of the survival estimates. From Table 4.3, where we show the average and the deviation of the integrated Brier score, we can notice that DeepHit and DPWTE seem to generally perform the best. DeepHit outperforms in FLCHAIN, SUPPORT, and METABRIC datasets with a very slight improvement, if it is not significant, compared to DPWTE, whereas the latter outperforms all the competing methods in SEER data with significant difference compared to the competing methods (except DeepHit). We can also notice that DPWTE has a stable estimation as shown by the narrowness of its confidence intervals. As we see DPWTE estimated different values of $\tilde{p}$ for the five datasets. In the case of SUPPORT, we have $\tilde{p} = 3$ but this does not necessarily mean that it used 3 disjoint Weibull distribution which is the case here. In fact, among these 3 Weibull distributions, 2 have practically the same estimate parameter values with different values of weighting coefficient estimates. Technically, DPWTE modeled the SUPPORT distribution with 2 disjoint and separated Weibull distributions.

Table 4.2: Comparison of C^{td} performance tested on the five benchmark datasets (mean and 95% confidence interval)

Models	**Datasets**				
	SEER BC	SEER HD	FLCHAIN	SUPPORT	METABRIC
CPH	0.831 (0.824 - 0.839)	0.785 (0.781 - 0.788)	0.789 (0.783 - 0.794)	0.805 (0.799 - 0.813)	0.661 (0.635 - 0.687)
Weibull AFT	0.832 (0.825 - 0.839)	0.785 (0.78 - 0.79)	0.789 (0.784 - 0.795)	0.807 (0.802 - 0.814)	0.659 (0.634 - 0.684)
DeepSurv	0.841 (0.836 - 0.847)	0.786 (0.784 - 0.787)	0.79 (0.78 - 0.8)	0.826 (0.811 - 0.831)	0.662 (0.635 - 0.69)
RSF	0.838 (0.829 - 0.848)	0.755 (0.744 - 0.765)	0.75 (0.708 - 0.791)	0.783 (0.78- 0.789)	0.667 (0.636 - 0.699)
DeepHit	0.875 (0.867-0.883)	0.846 (0.842-0.851)	**0.816 (0.809-0.821)**	**0.835 (0.83-0.842)**	0.821 (0.805-0.827)
DeepWeiSurv $p = 1$	0.877 (0.864-0.891)	0.857 (0.85-0.866)	0.78 (0.75-0.79)	0.802 (0.79-0.809)	0.805 (0.782-0.829)
DeepWeiSurv $p = 10$	0.908 (0.906 - 0.909)	0.863 (0.86 - 0.868)	0.79 (0.78 - 0.797)	0.815 (0.79 - 0.82)	0.819 (0.812 - 0.837)
DPWTE	**0.912 (0.911 - 0.914)**	**0.871 (0.865 - 0.878)**	0.812 (0.79 - 0.82)	0.83 (0.812 - 0.843)	**0.829 (0.808 - 0.849)**
$\tilde{p}$	2	2	1	3	1

Table 4.3: Comparison of IBS performance tested on the five benchmark datasets (mean and 95% confidence interval)

Models	**Datasets**				
	SEER BC	SEER HD	FLCHAIN	SUPPORT	METABRIC
CPH	0.15 (0.14-0.2)	0.17 (0.15-0.182)	0.192 (0.189-0.197)	0.167 (0.161-0.169)	0.224 (0.219-0.23)
Weibull AFT	0.158 (0.14-0.19)	0.169 (0.155-0.182)	0.191 (0.188-0.197)	0.164 (0.161-0.167)	0.227 (0.22-0.234)
DeepSurv	0.152 (0.15-0.16)	0.162 (0.16-0.174)	0.185 (0.183-0.189)	0.16 (0.158-0.162)	0.219 (0.215-0.224)
RSF	0.164 (0.162-0.17)	0.168 (0.166-0.17)	0.188 (0.186-0.192)	0.164 (0.159-0.166)	0.221 (0.217-0.226)
DeepHit	0.149 (0.145-0.151)	0.151 (0.147-0.153)	**0.174 (0.172-0.179)**	**0.152 (0.15-0.155)**	**0.196 (0.194-0.199)**
DeepWeiSurv $p = 1$	0.152 (0.148-0.155)	0.155 (0.151-0.158)	0.184 (0.182-0.188)	0.158 (0.156-0.161)	0.209 (0.206-0.214)
DeepWeiSurv $p = 10$	0.147 (0.145-0.15)	0.15 (0.148-0.153)	0.179 (0.177-0.183)	0.155 (0.152-0.157)	0.208 (0.206-0.212)
DPWTE	**0.142 (0.139-0.145)**	**0.149 (0.147-0.15)**	0.175 (0.177-0.181)	0.153 (0.149-0.157)	0.198 (0.195-0.201)

Table 4.4: Distribution of METABRIC observations (censored/non-censored) for each censoring threshold in $Q_{METABRIC}$.

t_c	No. censored	No. non-censored	Added portion
$t_{METABRIC}$	1093	888	-
$q_{0.5}$	1285	696	17.6%
$q_{0.45}$	1411	570	29%
$q_{0.35}$	1559	422	42.6%
$q_{0.25}$	1670	311	52.8%

Table 4.5: Distribution of FLCHAIN observations (censored/non-censored) for each censoring threshold in $Q_{FLCHAIN}$.

t_c	No. censored	No. non-censored	Added portion
$t_{FLCHAIN}$	5705	2169	-
$q_{0.65}$	6237	1637	9.3%
$q_{0.55}$	6534	1340	14.5%
$q_{0.4}$	6820	1054	19.5%
$q_{0.3}$	7086	788	24.2%

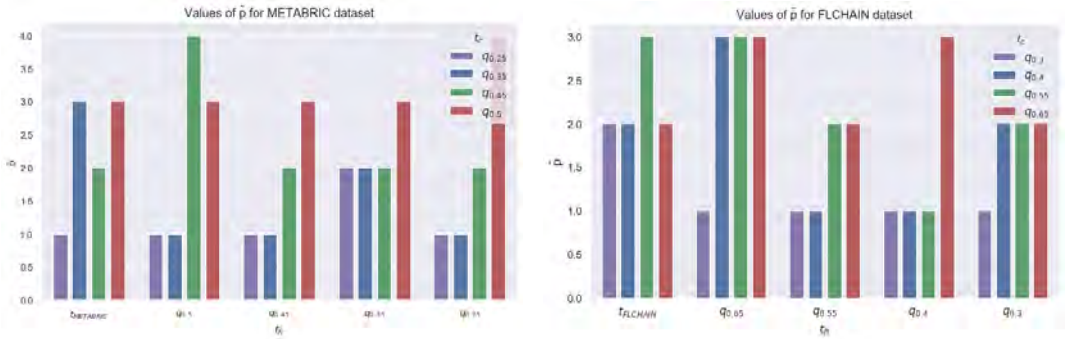
4.2.4 Experiment Two: Censoring Threshold Sensitivity

In the previous experiment, the models learned on the original version of the considered benchmark datasets. This means that the censoring threshold is fixed in each dataset. In this experiment, for a given dataset $\mathcal{X}$, we train the models with different censoring thresholds that are greater than the initial one. In other words, let t_{c_0} denotes the initial censoring threshold for a given dataset $\mathcal{X}$, we choose k censoring thresholds $\{t_{c_i} | t_{c_i} > t_{c_0},\ i = 1, ..k\}$ where for each t_{c_i}, the original is transformed to a new set $\mathcal{X}_{c_i} = \{t_i \mid \delta(t_i) = 1 \text{ if } t_i < t_{c_i} \text{ and } 0 \text{ otherwise}\}$ with which the models are fitted. Then, for each censoring threshold t_{c_i}, we evaluate the associated models on all the $\mathcal{X}_{c_i}$. The goal here is to investigate the ability of the models to handle the highly censored settings. We conduct this experiment on FLCHAIN and METABRIC by testing the 4 best models that show a good performance in the previous experiment namely: DeepSurv, DeepHit, DeepWeiSurv with $p = 10$, and finally DPWTE. We choose METABRIC because of its size which is relatively small compared to the others which render the task more challenging, while FLCHAIN is selected because DeepHit has shown the best performance for this dataset and thus represents a fair choice.

4.2.4.1 Experimental Protocol

For each considered dataset in this experiment, the censoring thresholds are expressed in quantiles q_α of the event time variable and chosen in such a way that each threshold provides a significant portion of censoring compared to the one that precedes it, or in other words, changes significantly the time distribution. For METABRIC and FLCHAIN, we respectively choose the following censoring threshold vectors: $Q_{METABRIC} = (q_{0.5}, q_{0.45}, q_{0.35}, q_{0.25})$ and $Q_{FLCHAIN} = (q_{0.65}, q_{0.55}, q_{0.4}, q_{0.3})$. Table 4.4 and Table 4.5 give, for each censoring threshold, the associated distribution of censored/non-censored samples for METABRIC and FLCHAIN datasets respectively. The 'Added portion' column represents the percentage (out of the initial distribution) of data whose status switch from non-censored to censored.

For each scenario defined by a couple: censoring threshold t_{c_i} and associated dataset $\mathcal{X}_{c_i}$,



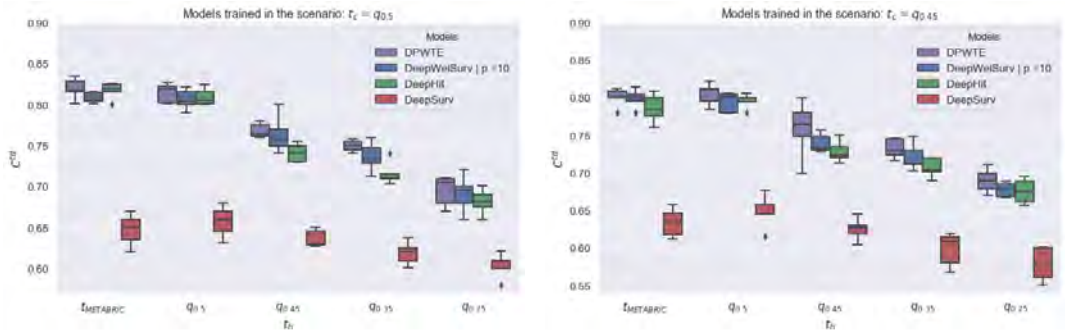
(a) METABRIC (b) FLCHAIN

Figure 4.6: Mean values of the estimate $\tilde{p}$ calculated over the 5-fold cross validation for each censoring threshold t_c in both METABRIC (left) and FLCHAIN (right).

we apply the five-fold cross-validation where, in each iteration, we train the models $\mathcal{M}$ on the associated training folds (4/5 of the global size) of $\mathcal{X}_{c_i}$ then evaluate these models on the associated validation fold of $(\mathcal{X}_{c_k})_{c_k \in Q_{\mathcal{X}}}$ by calculating the concordance index C^{td}. We obtain at the end of each scenario, a vector of five scores (calculated over the 5 iterations) for each model evaluation and for each validation fold.

4.2.4.2 Results and Discussion

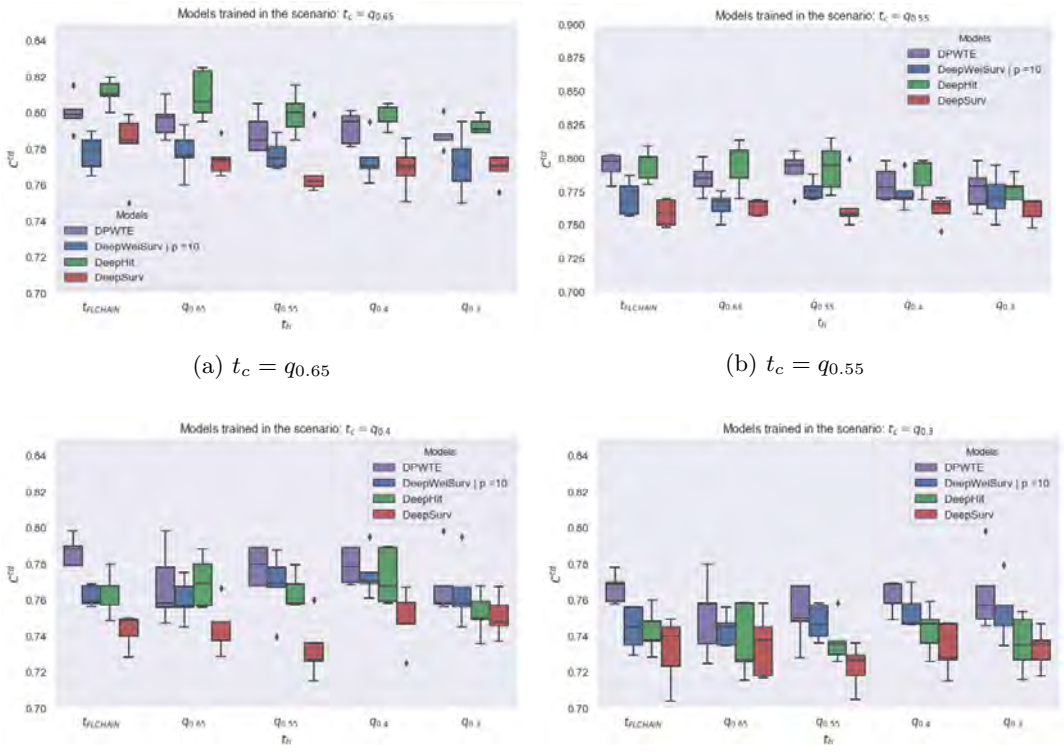
To highlight the mean and the standard deviation of the scores obtained, we use box plots. The results are shown in Figure 4.7 and Figure 4.8 for METABRIC and FLCHAIN datasets respectively where each scenario as described above is represented by a sub-figure. We call the time horizon t_h, the censoring threshold in the case of prediction. Firstly, it is considered overall, that the smaller the censoring threshold, i.e. the tighter the observation period, the weaker is the predictive performance of the models, which is (for the same reason as in the simulated experiments discussed in Chapter 2 and Chapter 3) normal since the quantity of knowledge about the distribution increases with the size of observed (non-censored) samples. For the METABRIC dataset, DPWTE outperforms in all scenarios whereas DeepSurv has the worst performance which means that it has the most difficulty in handling highly censored settings. As regards the two other models namely DeepHit and DeepWeiSurv, they still have a good performance with a normal drop for small thresholds. Still, the models generally perform an acceptable confidence interval. For FLCHAIN, DeepSurv gets closer to other models but still in the last of the ranking in terms of predictive performance. DeepHit starts (in the two first scenarios, i.e., for $t_c = q_{0.65}, q_{0.55}$) by outperforming the rest of models as in the previous experiments, but we notice that it has more difficulty than DPWTE and DeepWeiSurv for $t_c = q_{0.4}, q_{0.3}$. Furthermore, DeepHit performs standard deviations greater than those of DPWTE and DeepWeiSurv. We can conclude that DPWTE and DeepWeiSurv are the best in handling highly censored settings with a slight improvement provided by DPWTE. We suspect that their performance comes from the fact that the Weibull distribution best fits the respective underlying distributions of the benchmark datasets. Another thing to notice, from Figure 4.6, is that the estimate $\tilde{p}$ globally decreases, regardless of the survival time horizon t_h, while decreasing the censoring threshold (hence the censoring rate is increasing). We suspect this comes from the fact that the more we increase the censoring rate the more the network ignores a part of the underlying distribution and thus model the latter with an



(a) $t_c = q_{0.5}$ (b) $t_c = q_{0.45}$

(c) $t_c = q_{0.35}$ (d) $t_c = q_{0.25}$

Figure 4.7: Box plots of the concordance index scores calculated, for each censoring threshold $t_c \in Q_{METABRIC}$ over the five-fold cross validation for METABRIC dataset.



(a) $t_c = q_{0.65}$ (b) $t_c = q_{0.55}$

(c) $t_c = q_{0.4}$ (d) $t_c = q_{0.3}$

Figure 4.8: Box plots of the concordance index scores calculated, for each censoring threshold $t_c \in Q_{FLCHAIN}$ over the five-fold cross validation for FLCHAIN dataset.

insufficient combination of Weibull distributions (in terms of mixture size in this case).

4.3 Experiments on Simulated Time Series: Remaining Useful Life Prediction on NASA Turbofan Jet Engine Dataset using LSTM-Based Networks

In this section, we will conduct an experiment on Turbofan Jet Engine which is a run-to-failure simulated time-series data, whose goal is to predict the remaining useful life (RUL). We test and evaluate the performance of three network-based models namely DeepWeiSurv, DPWTE, and a fully connected network (FCN) where all of them are combined with the LSTM network.

4.3.1 Introduction

The remaining useful life (RUL) is a technical term that is defined by the length of time a machine is likely to operate before it requires repair or replacement [101]. In other words, RUL is used to describe the progression of a machine failure in order to schedule maintenance. In this way, RUL estimation has the potential to prevent critical faults, avoid unplanned downtime as well as optimize operating efficiency [102]. In the context of this experiment, where we use a Kaggle version of the very well known public data set for asset degradation modeling from NASA called Turbofan Jet Engine, the RUL of each engine is equivalent to the number of flights that remained for the engine after the last data point recorded [103]. Here, we seek to investigate the predictive performance of the three models cited above (that we will briefly describe in Section 4.3.2) in terms of RUL prediction.

4.3.2 Description of the Models

To perform the RUL prediction, we use the three networks one of which is a standard classifier and the remainder namely our two approaches are survival methods:

- Fully Connected Network (FCN): a network consisted of two hidden layers and one output layer that calculates the RUL values. ReLU activation function is applied on all the layers. FCN is trained by minimizing the mean squared error loss function.

- DeepWeiSurv (with the same configuration as described in Chapter 2) with $p = 1, 2$ and 3: since DeepWeiSurv learns the parameters of the Weibull mixture, it can estimate the RUL variable considered as the mean lifetime which only depends on these mixture parameters. We thus add a new output layer that takes, as an input, the outputs of the two previous layers namely the mixture parameters and calculates the RUL values.

- DPWTE, $p_{max} = 10$: with the same network configuration described in Chapter 3, it can do the same regression task as DeepWeiSurv by adding the extra layer.

Since we deal here with time series, we choose to combine individually LSTM with these three networks, to obtain the final models used for RUL prediction denoted by LSTM+FCN, LSTM+DeepWeiSurv, and LSTM+DPWTE respectively. Figure 4.3.2 describes the global architecture of these models. The inputs $(x^t)_t$ of the models as shown in this figure are sequences of time series obtained via a method described in Section 4.3.3.1.

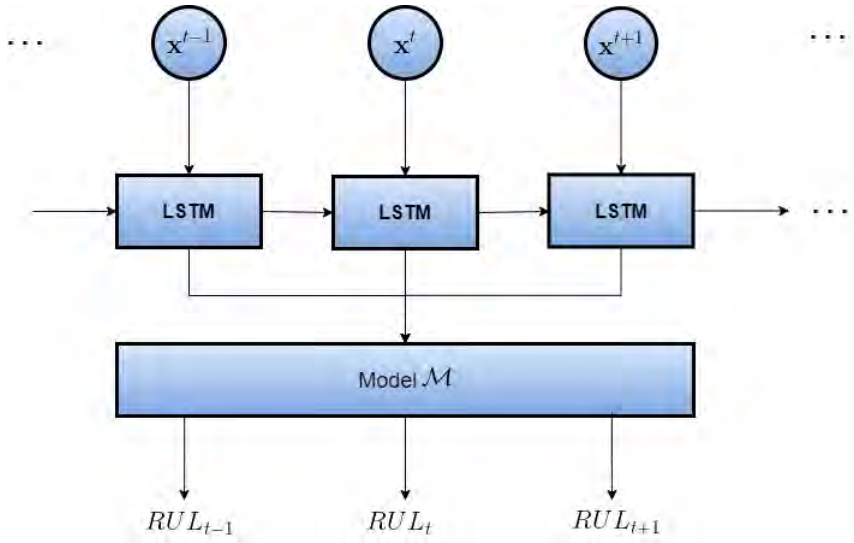


Figure 4.9: Global architecture of the three models considered in the experiment, with $\mathcal{M}$ = DeepWeiSurv, DPWTE or FCN.

4.3.3 Experimental Protocol

The Commercial Modular Aero-Propulsion System Simulation (C-MAPSS [104]) turbo aero-engine dataset is widely used in the predictive maintenance area. This family of datasets simulated the degradation of aero-engine in four working environments to get the corresponding datasets with different fault modes [105]: FD00X, X=1,2,3,4. The samples of FD001 suffered high-pressure compressor failure under a single operating condition, while FD002 describes the same fault mode but under six operating conditions. For FD003, the samples suffered high-pressure compressor as well as fan failure under a single operating condition, while in FD004, there are also these two fault modes but under six operating conditions. Here, we only conduct the experiment on the first working environment namely FD001. This dataset is further divided into training and test subsets that consist of multiple multivariate time series of different engines constituting a fleet of engines of the same type with their remaining useful life values. The training samples degrade until engine failure, and the time associated with the last record of a given engine unit is considered as the failure time. In the test set, the time series ends sometime prior to system fault and the failure period is recorded only for verification. The objective of this experiment is to predict, via three network-based models considered, the number of remaining operational cycles before failure in the test set.

4.3.3.1 Data Preprocessing

FD001 dataset describes the evolution of 100 engines (identified by the 'unit number' variable) where each engine x is recorded n_x times. The monitoring data in each sample consists of three operational settings, that jointly determine the system's working mode and have a significant effect on engine performance, as well as 26 noisy sensor measurements including total temperature, pressure at fan inlet, physical core speed, bleed enthalpy and other quantitative variables (see [103] for more details about variable description). Different sensors have different physical meanings and different range values. For this purpose, we standardize

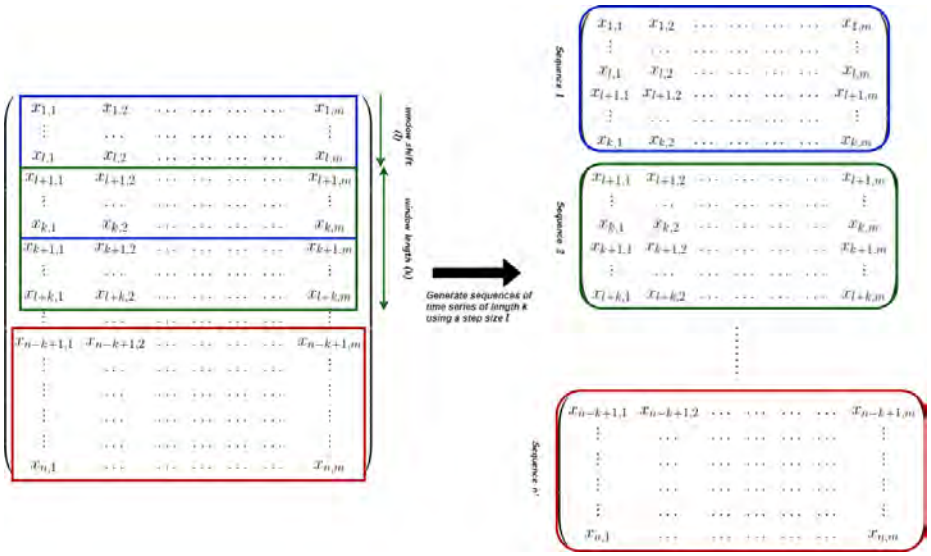


Figure 4.10: Diagram illustrating the sequence generation of an engine x with n records. For sequence generation, a sliding window is applied on the data with a window of size k and shifting by l, to obtain n' sequences with $n' = \lfloor \frac{n-k}{l} \rfloor + 1$.

the data to adjust the range of each sensor and by doing this, we eliminate the influence of ranges on the degree of contribution in the RUL prediction. Since we use LSTM, we transform the data into a set of sequences of time series using a sliding window, of a fixed length k, large enough to span relevant contextual information that is necessary to estimate the RUL variable (see Figure 4.10 for illustration). The same transformation is thus applied to the target variable. We point out that each sequence is from the same engine, which means that the window stops at the last record of the current engine, and a new window is applied on the next engine records to avoid sequences with more than one engine. Therefore, the model inputs are written as follows: $((X_1, Y_1), ..., (X_{100}, Y_{100}))$ where $X_j = (X_1^j, ..., X_{n'_j}^j)$ and $Y_j = (Y_1^j, ..., Y_{n'_j}^j)$ correspond to the records of the engine of unit number j, sensor measurements X_j and RUL values Y_j.

4.3.3.2 Evaluation Metrics

Let y and $\hat{y}$ denote the real RUL and the predicted RUL variables respectively. In the context of predictive maintenance, the late predictions ($\hat{y}_i > y_i$) are considered to have more serious consequences than the early ones ($\hat{y}_i \leq y_i$). Thus to evaluate the predictive performance of the proposed models, we calculate the two following scores: mean absolute error (MAE) and the averaged weighted sum of RUL error (WRE). Mean Absolute Error gives an equal penalty to late and early predictions, as we see in the following equation:

$$MAE = \frac{1}{|y|} \sum_i |y_i - \hat{y}_i| \tag{4.3.1}$$

where $|y|$ is the size of predictions. The second metric WRE, proposed by [103], is defined as an asymmetric function which penalizes late predictions more than the early ones. WRE is nothing but a weighted sum of remaining useful life errors averaged over the predictions,

Table 4.6: MAE and WRE scores for each model in each scenario.

Models	MAE			WRE		
	$k = 10$	$k = 25$	$k = 50$	$k = 10$	$k = 25$	$k = 50$
LSTM+FCN	26.72	21.17	17.91	45.5	25.6	16.8
LSTM+DeepWeiSurv	12.49	10.18	9.81	15.03	9.16	4.95
LSTM+DPWTE | $\tilde{p} = 3$	**11.43**	**7.06**	**6.89**	**9.87**	**6.3**	**2.22**

and defined by the following equation:

$$WRE = \frac{1}{|y|}\sum_i s_i,$$
$$s_i = \begin{cases} e^{-\frac{\hat{y}_i - y_i}{13}} - 1 & \text{if } \hat{y}_i < y_i \\ e^{\frac{\hat{y}_i - y_i}{10}} - 1 & \text{otherwise} \end{cases}$$

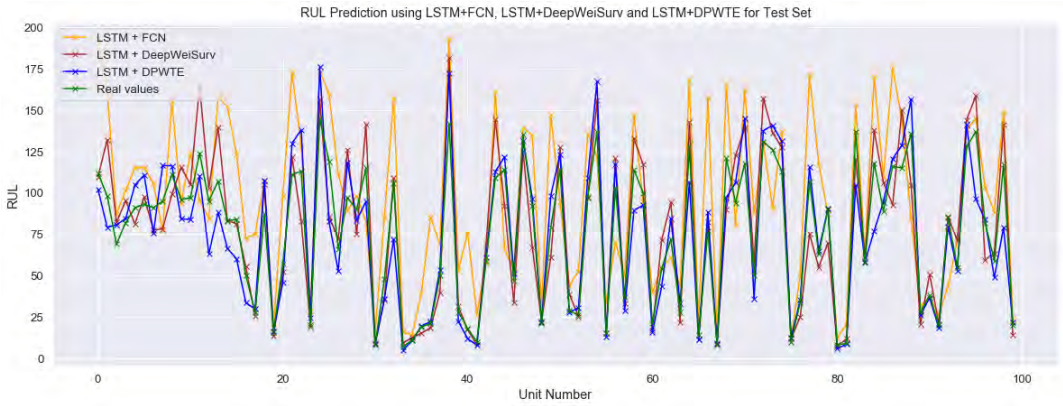
A perfect model would have a score equal to zero.

4.3.3.3 Experimental Settings

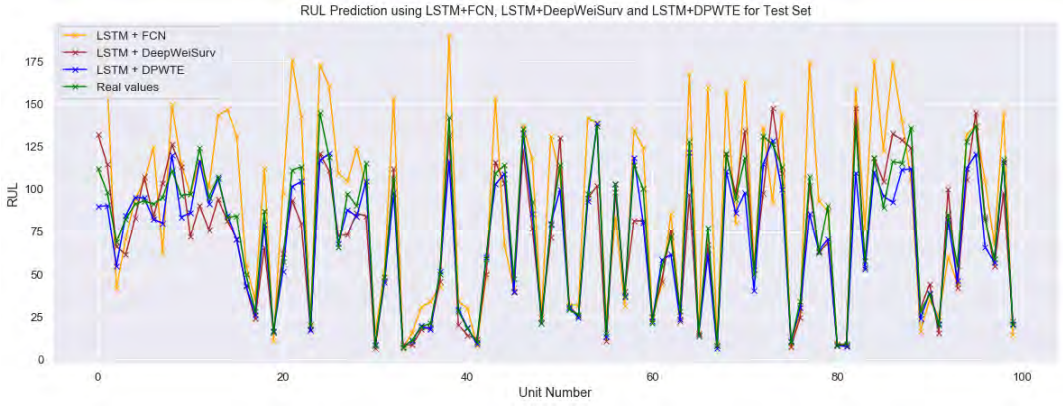
We evaluate the models in three scenarios: we train the models on the sequences generated from the training data by applying a sliding window with a size of $k = 10$, then $k = 25$ and finally $k = 50$ and we choose to set the step size l to 1 for these three cases. We set $p = 10$ for DeepWeiSurv and $p_{max} = 10$ for DPWTE. The models are trained via Adam optimizer with a learning rate of 10^{-4}. For each scenario, we calculate the RUL predictions performed by the models and plot them with the real values of the target variable. We also calculate two evaluation metrics namely the mean squared error (MAE) and the averaged weighted sum of RUL errors (WRE).

4.3.4 Results and Discussion

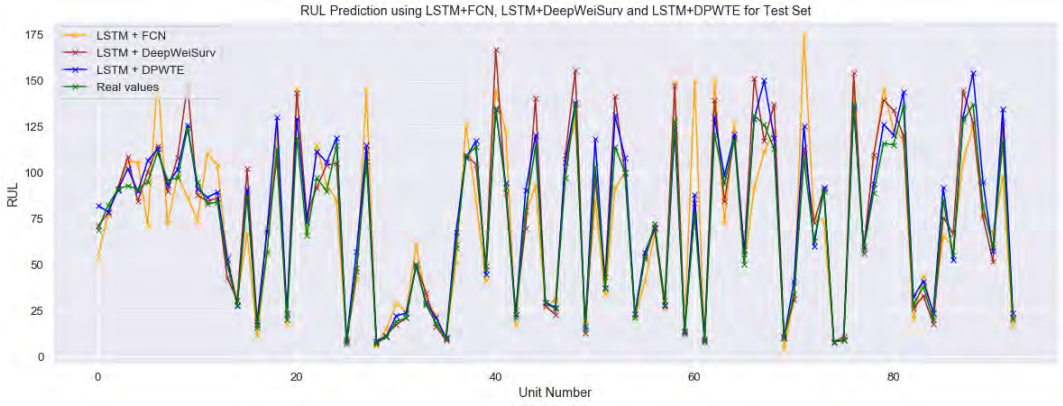
The results of RUL predictions for different scenarios are shown in Figure 4.11 and the evaluation scores are summarized in Table 4.6. The first thing that we can notice, is that increasing length boosts the predictive performance and the models achieve lower error scores on the sequences of size 50 than those calculated on sequences of size 25 and even less than those on sequences of 10 samples. This observation confirms that investigating fault progression in short sequences is more challenging and this is because we lose some contextual information in short sequences. This increasing of length implies a significant performance boost in terms of RUL prediction for LSTM+FCN as shown in Figure 4.11 and quantified in Table 4.6. Still, LSTM+DPWTE, with an optimal mixture of 3 Weibull distributions, outperforms the two other models but not with the same difference. In fact, as we see in this figure, the RUL predictions performed by the DPWTE-based model almost coincide with the real values in the best case ($k = 50$). In addition, DeepWeiSurv performance is getting closer to that of DPWTE in the best case namely $k = 50$, and thus still acceptable. Concerning the two error scores, we can notice that our two proposed approaches decrease substantially the MAE and WRE compared to FCN. This means that the two proposed models have RUL estimations much more accurate and prevent well from late predictions and even better when the length of the sequences is sufficiently large.



(a) $k = 10$



(b) $k = 25$



(c) $k = 50$

Figure 4.11: RUL prediction results: we evaluate the performance of LSTM+{FCN, DeepWeiSurv, DPWTE} in three cases defined by the values of the window length k tested namely $k = 10$, $k = 25$ and $k = 50$.

4.4 Conclusion

In this chapter, we investigated the performance of DeepWeiSurv and DPWTE in two experiments different than those conducted (with simulated data) and described in Chapter 2 and Chapter 3. In the first experiment, we first evaluated the performance of our approaches as well as that of the well-known state-of-the-art methods and compared between them using two evaluation metrics namely the time-dependent Concordance Index (C^{td}) and Integrated Brier Score (IBS). We showed that DPWTE, DeepWeiSurv, and DeepHit have practically the same performance but our approaches still have a larger gap, in terms of predictive performance measured by C^{td}, when they outperform for a given dataset (e.g. 0.912/0.908 for DPWTE/DeepWeiSurv vs. 0.875 for DeepHit in SEER BC) while the gap is not statistically significant when DeepHit outperforms, especially between the latter and DPWTE (e.g. 0.812, 0.83 for DPWTE vs. 0.816, 0.835 for DeepHit, in FLCHAIN and SUPPORT respectively). The difference between model performances is even more marked in the second part of the first experiment where we evaluated the model sensitivity towards a highly censored setting. In fact, DeepWeiSurv and DPWTE handle better the higher censoring rate than the other models, and the difference in performance increases with this ratio. Then, in the second experiment, we explored the predictive performance of our two approaches combined with LSTM in a run-to-failure simulated time-series experiment using two error scores (RMSE and WRE) and compared them with that of LSTM combined with a dense network. In this experiment, our two models largely outperform the other model in terms of RUL prediction and late prediction penalty score and this means that DPWTE and DeepWeiSurv prevent way better the late predictions whose consequences are important in this context.

Chapter 5

Deep Learning Approach for the Maximization of a Regression Output under Budget Constraints

5.1 Introduction

Predictive maintenance (PdM) techniques are designed to estimate or predict when equipment or machine failure will occur so that the required maintenance can be scheduled. Among the first models widely used in this field is the Kaplan-Meier estimator [61]. However, it is limited in the sense it does not incorporate the observations' covariates. Another model called semi-parametric Cox Proportional Hazards [29] takes into account the covariates but makes a simplistic assumption where it considers that the probability of failure is a linear function of covariates. After that, several works have been carried out that propose a network-based approach notably Faraggi-Simon network [32], DeepSurv[31] (deeper in terms of layers), Luck's model [42] and deepHit [30] (these two models learn the survival function which makes them more general, i.e., we can estimate the failure probability at any given time) and DeepWeisurv [35] which assumes that the survival times variable follows a mixture of Weibull distributions allowing, through the learned parameters of this mixture, to estimate the survival function, the mean lifetime, the median, etc.

Many, if not most of the works in survival analysis focused on estimating the risk of experiencing an event. However, once the model is able to estimate this risk, the problem is to find the most optimal way to maximize the life expectancy given a certain budget. Admittedly, the optimization of the mean lifetime in a predictive maintenance context motivates our work, but the problem can go beyond this context and can be raised for any regression problem where optimizing the output makes sense and the subjects of study have some controllable (i.e., modifiable) features. Let's take an example of the problem of modeling wine preferences [106] where the goal is to estimate the quality (0 to 10) of the wine (subject) given its physico-chemical features. Since some features are controllable such as alcohol, pH, and volatile acidity, we can raise the problem of wine quality optimization for wines whose qualities are under 10 for example. For this purpose, we consider, for a given subject, a constrained-optimization problem whose objective function is $f(\mathbf{x})$, with f is the regression

model used and $\mathbf{x}$ the variable that represents the features of the concerned subject, and whose solution is the optimal modification of its controllable features while respecting the budget constraint. In other words, the solution belongs to a weighted ℓ_1 ball that represents this constraint to which the solution must submit. This optimization problem is similar to that of the model explication/interpretation because we need to estimate the importance of each (controllable) feature, or in other words, the (positive or negative) contribution of each feature in the model output for output optimization purpose. To solve this problem while respecting the budget constraint, we use the *Projected Gradient Descent* (PGD) algorithm which requires to calculate the gradients of f, at each iteration. Furthermore, this problem can be formulated as a an adversarial attack problem in a regression setting since a translation of the current iterate in the update step of the PGD algorithm could be considered (especially when the network is not robust, i.e. not trained against adversarial instances) as an adversarial input to the network f. Therefore, we consider three scenarios in which the regression network f can be found :

- *Semi-White Box* when the regression model f is a given neural network already trained with which we can only calculate the gradients with respect to the inputs. This means that we cannot re-train the network, or modify the weights of the network layers.
- *Full-White Box* when f is a white-box neural network that we can train, that is, we have access and control on the weights. To robustify the network, we introduce a gradient penalization technique to train f in such a way to resist adversarial inputs, or in other words, to reduce the numerical instability of the network;
- *Black Box* when f is a black box, which means we cannot calculate the gradients of f and have access to the weights. In this case, we need to interpret the black-box model, which means we need to locally estimate the contribution of each covariate in approximating f, that is calculating the gradients with respect to covariates and locally model the relationship between f and the covariates using the inputs and outputs of the black-box model f. To do so, we use the LIME approach, which is a linear-regression-based method to model f locally.

This chapter is organized as follows: Section 5.2 is dedicated to describe the model-agnostic interpretability method for black-models namely LIME. In Section 5.3 we present and describe the mechanism defense used to make a network robust against adversarial examples. In Section 5.4, we describe how to calculate the projection onto a weighted ℓ_1 ball that we use in the PGD algorithm. In Section 5.5, we describe the main problem considered in this work. Section 5.6 describes the PGD algorithm as well as the three scenarios of the regression model. In Section 5.7, we conduct two simulation studies (one is dedicated to the three scenarios, and another one only for the robust full-white box). In Section 5.8, we run an experiment on Wine Quality dataset [106] where we discuss the performance of the robust full-white box network. We conclude and propose perspectives in Section 5.9.

5.2 Estimating Gradient of Black-Box Networks using LIME

The model-agnostic interpretability methods [107] are methods used to explain already trained supervised machine learning models, notably the neural networks. These techniques explain predictions of these machine learning models, assumed to be black boxes, without

inspecting the internal model parameters (e.g. model's weights). Model-agnostic interpretability methods enable models to be interpreted either globally or locally regardless of the complexity. Global interpretability generates explanations of the overall behavior over the entire population under study. For instance, explaining which of the covariates have a significant contribution in the construction of the model or describing the (positive or negative) impact of each feature, on average, on the predictions of the model. While local interpretability methods, also known as local surrogate models, are interpretable models that are used to explain individual predictions of the model. The principle of this approach is to assume that model predictions in the neighborhood of a given instance can be approximated by a simple with-box interpretable model such as a regularized linear regression model (LASSO or Ridge). In the next section, we present and describe the local model-agnostic model that we use in this work namely the so-called LIME whose role is to estimate the feature contributions and linearly combine them to model the black-box network.

Description of LIME

In this work, we are interested in local surrogate models and specifically, the Local Interpretable Model-Agnostic Explanation method (LIME) [39] to handle the third scenario as described in the Introduction. In this case, f is a black-box network-based model which can only be used to take input data points and provide predictions. The goal is therefore to capture the link between inputs and outputs at the individual level, that is, we seek to understand why the black-box model made a certain prediction for a given individual. The concept of LIME is to investigate the prediction change when we give variations or perturb data into the model. For this purpose, LIME generates for a particular instance a new dataset consisting of the instance itself and its neighborhood as well as the corresponding predictions of the model f. Many methods are possible to use to generate this neighborhood such as Gaussian and Latin Hyper-cube (LHS) [108] sampling methods. On this new dataset, LIME then trains an interpretable model (e.g. Lasso) weighted by the proximity of the neighborhood individuals to the instance of interest. We use the Euclidean distance and the Gaussian kernel π to measure the proximity and calculate the weight of each element of the neighborhood respectively. Mathematically, LIME can be expressed as follows:

$$LIME(\mathbf{x}) = \underset{g \in L}{argmin}\ \mathcal{L}(f, g, \pi, \mathbf{x}) + \mathcal{R}(g) \tag{5.2.1}$$

where g is the explanation model for instance $\mathbf{x}$, expected to minimize the loss function $\mathcal{L}$ (e.g., mean squared error) which measures the gap between explanation and prediction provided by the original model f. L is the family of all possible linear regression models and $\mathcal{R}(g)$ is the penalty term used to apply a regularization on the weights of g. The proximity metric π defines how large the neighborhood around a particular instance $\mathbf{x}$ is that we consider performing the explanation. In this approach, one has to think about the number of features n_v to have in the conception of g. The lower n_v, the easier it to interpret the model. However, we risk losing precision since higher n_v potentially produces models with higher fidelity. Several methods can be used to select n_v best feature candidates notably:

- Forward selection: which is a method taking as input n_v and selects the n_v highly contributing variables by iteratively adding features that bring significant improvement to predictions.

- Lasso model which is trained on the dataset to select the features that have a non-zero contribution on the predictions, by calculating their respective weights. This method

is unsupervised in the sense it does not require n_v as a parameter, which means that it can return a set of features whose size is less than n_v.

- , In the same way, we can use Ridge regression to calculate the (negative or positive) contribution of the features through the model weights and select those that have non-zero weights (or above a given threshold).

To summarize the process of training LIME below is the recipe:

1. Select the instance of interest $\mathbf{x}$ for which we want to have an explanation of its black box prediction $f(\mathbf{x})$.
2. Perturb the instance using a Gaussian or LHS method to generate the neighborhood $\mathcal{V}(\mathbf{x})$.
3. Weight the new samples of $\mathcal{V}(\mathbf{x})$ according to their proximity to the instance $\mathbf{x}$ using the kernel π and the euclidean distance.
4. Apply the feature selection method, to choose the n_v (or less) features to model LIME.
5. Train LIME on the perturbed dataset.
6. Use LIME to explain prediction by investigating the weights, predict a new instance drawn from the neighborhood distribution, or calculate the gradients which are nothing but the weights of this explanation model.

Finally, LIME as a solution to Problem 5.2.1 can be written as follows:

$$LIME(\mathbf{x}) = g^*(\mathbf{x}) = \sum_{i=1}^{n_v} a_i x_i,\ a_i \in \mathbb{R}^* \tag{5.2.2}$$

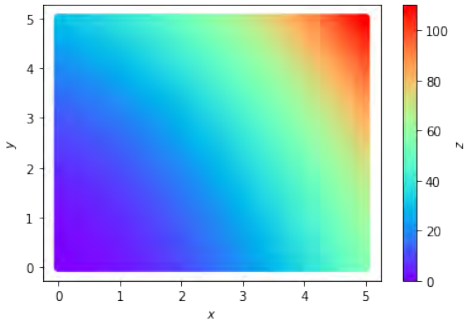
where x_i are the features selected to explain predictions and a_i the weights associated. Thus, the partial derivative of f in the direction x_i can be approximated by:

$$\frac{\partial f}{\partial x_i} \approx \frac{\partial g^*}{\partial x_i} = a_i \tag{5.2.3}$$

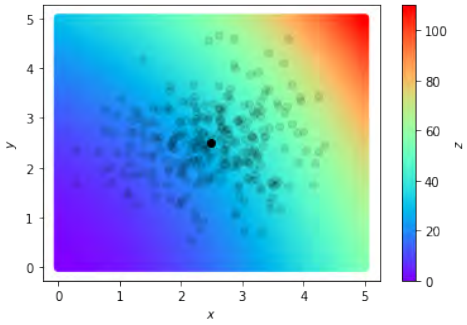
Therefore, we can approximate, using LIME solution (i.e., g^*), the output og the black-box model f corresponding to the input $\mathbf{x}$ as well as its respective local gradients. Figure 5.1 visually illustrates how sampling and local model training works. We use an example of an instance $\mathbf{x}$ of two dimension characterized by the covariates x and y where the analytical function to model locally is defined as: $z = f(x, y) = 2x^2 + xy + y^2 + x + y$. The output z is defined by the colormap. LIME currently uses an exponential smoothing kernel to define the proximity of the neighborhood $\mathcal{V}(\mathbf{x})$: $\pi\big(\mathbf{x}' \in \mathcal{V}(\mathbf{x})\big) = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|_2}{2}}$. This smoothing kernel is a function that takes the particular instance of interest as well as an element of the neighborhood and returns a proximity measure.

5.3 Robustifying Networks using Gradient Regularization against Adversarial Regression Attacks

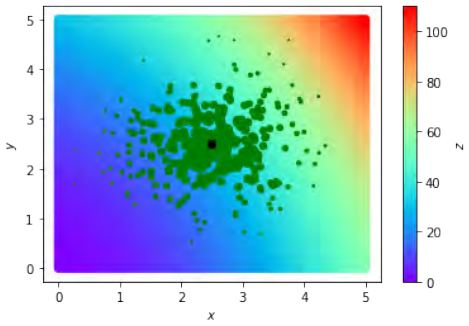
In this section, we start by synthesizing the state-of-the-art works that dealt with this problem. Then, we describe a method that penalizes adversarial inputs in the training phase of a particular network.



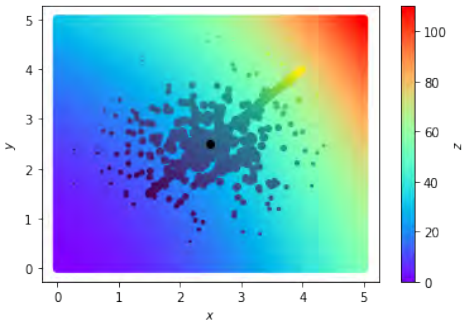
(a) z values given features x and y



(b) $\mathbf{x}$ and $\mathcal{V}(\mathbf{x})$



(c) Assign π values on $\mathcal{V}(\mathbf{x})$



(d) Train linear model for $\mathcal{V}(\mathbf{x})$

Figure 5.1: LIME algorithm for 2D data. (a) z values for different feature values x and y from $[0, 5]$. (b) An instance of interest $\mathbf{x}$ (in big dot) and its neighborhood $\mathcal{V}(\mathbf{x})$ sampled from a normal distribution (shaded small dots). (c) Assign weights to the neighborhood using π (higher weights for the nearest neighbors and lower weights for the farthest ones. (d) Learn the local model using Ridge regression taking into account the weights of the neighborhood data.

5.3.1 State of the Art

Adversarial attacks in machine learning, i.e. augmented data points generated by perturbation of input samples against neural networks, has been a research area for over a decade [109], but have recently drawn much attention with the machine learning and data mining community [110, 111, 112, 113, 114, 115]. However, adversarial attacks in a regression setting are understudied so far. In fact, Biggio and Roli provided in [109] an extensive survey of the history of adversarial machine learning and current research directions, but surprisingly little work has been done on adversarial regression problems [111].

To explain why adversarial examples exist, Szegedy et al. [116] suspected this phenomenon comes from the fact that the network output is a highly nonlinear function of inputs and reported that in [117] that concatenation of non-linear layers in between the input and the output nodes is a way for the network to encode non-local generalization prior over the input space. This means that one assumed that it is possible to assign non-significant probability values to regions of the input space whose vicinity is free of training instances. Whereas Goodfellow et al. argued in [114] that this particular phenomenon could arise naturally from high dimensional linearity. Similarly, Fawzi et al. [118] support that non-linearity is not the fundamental reason behind adversarial attacks and reported that the ability to generalize adversarial examples is due to the resemblance of network models to linear classifiers. To attenuate the impact made by adversarial inputs, Gu et al. [119] penalized the Jacobian matrix based on a series of approximations of adversarial noise. Still, their model leads to an accuracy drop aside from the training downturn caused by the additional cost. Szegedy et al. [116] have demonstrated that adversarial training is the suitable solution, that is, injecting those perturbed samples back in training and take them into account in the loss function. Goodfellow et al. [114] followed and extended this idea to the fast gradient sign method where, using this approach, the models are more robust against adversarial attacks.

Despite this newfound investment in adversarial attacks in machine learning, few works investigate the regression case. Among the most known works on this problem are respectively those of Nguyen and Raff in [111], Simon-Gabriel et al. in [112], Lyu et al. in [110] and Yu et al. in [113] who took the perspective that adversarial attacks in a regression setting (and can be applied in a classification setting, e.g. [110]) are likely caused by numerical instability in learned networks. Another work performed by Singh et al in [120] where it is assumed that the weights of individual layers of a network are ill-conditioned and they are one of the contributing factors in neural network's susceptibility towards adversarial attacks. This means they take the perspective of numerical instability from a layer view, while others are concerned with the numerical stability of the network as a whole function and this is what we exploit in our work.

5.3.2 Gradient Regularization against Numerical Instability

In this section, we will describe the method, inspired by [110] and initially proposed in [114], that we use to build a robust version of a full-white box model f in order to handle the numerical instability.

5.3.2.1 General Description of Gradient Regularization Method

Let $\mathcal{L}(\mathbf{x};\theta)$ denote the loss function where $\mathbf{x}$ is the baseline data and θ the parameters of a given model f. To train f on $\mathbf{x}$, it usually proposed to solve the following optimization

problem:

$$\underset{\theta}{argmin}\ \mathcal{L}(\mathbf{x};\theta) \tag{5.3.1}$$

However this problem does not handle adversarial examples (i.e. perturbed instances) and thus the model is highly vulnerable to numerical instability. In fact, it is desirable that the loss function $\ell(.)$ should have no great change in output for small perturbations ϵ. Usually, these adversarial instances are translation of the original data (i.e, in the form of $\mathbf{x}+\epsilon$). Goodfellow et al. [114] proposed to train on adversarial examples, that is train the model f with an adversarial loss function. The idea can be formalized as follows. Instead of solving 5.3.1, we seek to build a robust model against numerical instability by solving the following min-max optimization problem:

$$\underset{\theta}{argmin}\ \underset{\|\epsilon\|_p \leq \sigma}{argmax}\ \mathcal{L}(\mathbf{x}+\epsilon;\theta) \tag{5.3.2}$$

where $\|.\|_p$ is the ℓ_p norm and σ is the radius of the ℓ_p ball to which ϵ should belong. The norm constraint of the inner problem says that we only require our model f to be robust against certain perturbation. This means that the maximization portion finds the perturbation ϵ that maximizes the loss given the data and constraints. Then, the outer minimization attempts to alleviate this loss by optimization model parameters θ. Since, this problem is non-convex with respect to ϵ and θ, it is difficult to find an exact solution. Alternatively, we propose to solve the problem using an approximation technique [110]. To do so, we approximate $\mathcal{L}(\mathbf{x};\theta)$ by its first-order Taylor series at $\mathbf{x}$. For the sake simplicity, $\mathcal{L}(\mathbf{x};\theta)$ will be henceforth shorthanded as $\mathcal{L}$. The inner max problem can thus be written as follows:

$$\underset{\|\epsilon\|_p \leq \sigma}{argmax}\ \mathcal{L} + \nabla_{\mathbf{x}}\mathcal{L}^T\epsilon \tag{5.3.3}$$

The problem becomes linear and hence convex with respect to ϵ. Since $\mathcal{L}(\mathbf{x})$ is independent of ϵ. The Lagrange dual associated is defined by the following equation:

$$\underset{\epsilon}{argmax}\ \nabla_{\mathbf{x}}\mathcal{L}^T\epsilon - \lambda(\|\epsilon\|_p - \sigma) \tag{5.3.4}$$

Using the Karush-Kuhn-Tucker (KKT) theorem [121], we can show that necessary and sufficient conditions for ϵ to be an optimum are:

$$\nabla_{\mathbf{x}}\mathcal{L} - \lambda \overbrace{|\epsilon|^{p-1}\|\epsilon\|_p^{p-1}\mathbf{sgn}(\epsilon)}^{\nabla_\epsilon \|\epsilon\|_p} = 0 \tag{5.3.5}$$

$$\lambda(\|\epsilon\|_p - \sigma) = 0 \tag{5.3.6}$$

where **sgn** is the *signum* vector function. Assume that the optimal ϵ has a norm strictly lower than σ, that is, $\|\epsilon\|_p < \sigma$. Let $\epsilon' = \frac{\sigma}{\|\epsilon\|_p}\epsilon$. Therefore $\|\epsilon'\|_p = \frac{\sigma}{\|\epsilon\|_p}\|\epsilon\|_p = \sigma$ and hence ϵ' is a feasible solution. In addition,

$$\nabla_{\mathbf{x}}\mathcal{L}^T\epsilon' = \frac{\sigma}{\|\epsilon\|_p}\nabla_{\mathbf{x}}\mathcal{L}^T\epsilon > \nabla_{\mathbf{x}}\mathcal{L}^T\epsilon$$

We thus constructed a feasible solution ϵ' which attains a loss value greater than that of ϵ. This leads us to a contradiction. Therefore, we are set to solve the following equation

system:

$$\begin{cases} \nabla_{\mathbf{x}}\mathcal{L} - \lambda|\epsilon|^{p-1}\left\|\epsilon\right\|_p^{p-1}\mathbf{sgn}(\epsilon) & = 0 \\ \left\|\epsilon\right\|_p & = \sigma \end{cases} \tag{5.3.7}$$

Combining the two equations of System (5.3.7), we have:

$$\nabla_{\mathbf{x}}\mathcal{L} = \lambda|\epsilon|^{p-1}\sigma^{1-p}\mathbf{sgn}(\epsilon) \tag{5.3.8}$$

$$\nabla_{\mathbf{x}}\mathcal{L}^{\frac{p}{p-1}} = \lambda^{\frac{p}{p-1}}\mathbf{sgn}(\epsilon)^{\frac{p}{p-1}}\frac{|\epsilon|^p}{\sigma^p} \tag{5.3.9}$$

Let $p^* = \dfrac{p}{p-1}$ denote the dual of p, i.e., $\dfrac{1}{p^*} = 1 - \dfrac{1}{p}$. Then, by applying the absolute value function and summing over two sides of Equation (5.3.9), we have:

$$\sum|\nabla_{\mathbf{x}}\mathcal{L}|^{p^*} = |\lambda|^{p*}\sum\frac{|\epsilon|^p}{\sigma^p} \tag{5.3.10}$$

$$\implies \|\nabla_{\mathbf{x}}\|_{p^*}^{p^*} = |\lambda|^{p^*}\frac{\|\epsilon\|_p^p}{\sigma^p} = |\lambda|^{p^*} \tag{5.3.11}$$

where $\|.\|_{p^*}$ is the dual norm associated to the ℓ_p norm. Combining System (5.3.8) and Equation (5.3.11), we obtain:

$$|\nabla_{\mathbf{x}}\mathcal{L}| = |\lambda|\sigma^{1-p}|\epsilon|^{p-1} \tag{5.3.12}$$

$$= \|\nabla_{\mathbf{x}}\mathcal{L}\|_{p^*}\sigma^{1-p}|\epsilon|^{p-1} \tag{5.3.13}$$

Finally, one finds the solution of the inner problem which is written as follows:

$$\epsilon = \sigma\ \mathbf{sgn}(\nabla_{\mathbf{x}}\mathcal{L})\left(\frac{|\nabla_{\mathbf{x}}\mathcal{L}|}{\|\nabla_{\mathbf{x}}\mathcal{L}\|_{p^*}}\right)^{\frac{1}{p-1}} \tag{5.3.14}$$

5.3.2.2 Inner Problem Solution with Respect to the ℓ_p Norm

In the following, we analyze the solution of the inner maximization problem in different cases defined by the norm constraints. Here, we only examine two norm constraints namely when $p = 1$ and $p = \infty$.

ℓ_1-Norm Constraint In this case, we have $p^* \to \infty$ and thus the perturbation that maximizes the loss becomes:

$$\lim_{p\to 1}\epsilon = \sigma\ \mathbf{sgn}(\nabla_{\mathbf{x}}\mathcal{L})\ \lim_{p\to 1}\left(\frac{|\nabla_{\mathbf{x}}\mathcal{L}|}{\|\nabla_{\mathbf{x}}\mathcal{L}\|_{p^*}}\right)^{\frac{1}{p-1}} \tag{5.3.15}$$

$$= \sigma\ \mathbf{sgn}(\nabla_{\mathbf{x}}\mathcal{L})\left(\frac{|\nabla_{\mathbf{x}}\mathcal{L}|}{\|\nabla_{\mathbf{x}}\mathcal{L}\|_{\infty}}\right)^{\infty} \tag{5.3.16}$$

Since we are constrained by the ℓ_1 norm of the perturbation ϵ whose value is regardless of the directions being penalized, it is intuitive to choose one direction that maximize the overall perturbation. Thus, a suitable solution will be written as:

$$\epsilon_j = \begin{cases} \sigma\ \mathbf{sgn}(\nabla\mathcal{L}_j) \text{ if } \|\nabla_{\mathbf{x}}\mathcal{L}\|_{\infty} = \nabla\mathcal{L}_j \\ 0, \text{ otherwise.} \end{cases} \tag{5.3.17}$$

where $\nabla\mathcal{L}_j = \dfrac{\partial\mathcal{L}}{\partial x_j}$. The loss function can therefore be approximated as follows:

$$\mathcal{L} + \nabla_{\mathbf{x}}\mathcal{L}^T\epsilon = \mathcal{L} + \sigma\nabla\mathcal{L}_{i_0}\mathbf{sgn}(\nabla\mathcal{L}_{i_0}) \tag{5.3.18}$$
$$= \mathcal{L} + \sigma|\nabla\mathcal{L}_{i_0}| \tag{5.3.19}$$
$$= \mathcal{L} + \sigma\left\|\nabla_{\mathbf{x}}\mathcal{L}\right\|_\infty \tag{5.3.20}$$

where $i_0 = argmax\ |\nabla\mathcal{L}_i|$ is the penalized direction and $\sigma\left\|\nabla_{\mathbf{x}}\mathcal{L}\right\|_\infty$ is the regularization term. However, the main drawback in using this regularization term is that it only penalized one direction, and thus it is not interesting to use the ℓ_1-norm constraint in our context.

ℓ_∞-Norm Constraint In this case, where $p \to \infty$ and thus $p^* \to 1$, the best perturbation (in terms of maximizing the loss) is written as follows:

$$\lim_{p\to\infty}\epsilon = \sigma\ \mathbf{sgn}(\nabla_{\mathbf{x}}\mathcal{L})\ \lim_{p\to\infty}\left(\frac{|\nabla_{\mathbf{x}}\mathcal{L}|}{\left\|\nabla_{\mathbf{x}}\mathcal{L}\right\|_{p^*}}\right)^{\frac{1}{p-1}} \tag{5.3.21}$$
$$= \sigma\ \mathbf{sgn}(\nabla_{\mathbf{x}}\mathcal{L})\left(\frac{|\nabla_{\mathbf{x}}\mathcal{L}|}{\left\|\nabla_{\mathbf{x}}\mathcal{L}\right\|_1}\right)^0 \tag{5.3.22}$$

By obviously assuming that $|\nabla_{\mathbf{x}}\mathcal{L}| > 0$, we therefore have:

$$\epsilon = \sigma\ \mathbf{sgn}(\nabla_{\mathbf{x}}\mathcal{L}) \tag{5.3.23}$$

This special case corresponds therefore to *fast gradient sign method* proposed in [114]. The loss function is thus written as follows:

$$\mathcal{L} + \nabla_{\mathbf{x}}\mathcal{L}^T\epsilon = \mathcal{L} + \sigma\nabla_{\mathbf{x}}\mathcal{L}^T\mathbf{sgn}(\nabla_{\mathbf{x}}\mathcal{L}) \tag{5.3.24}$$
$$= \mathcal{L} + \sigma\left\|\nabla_{\mathbf{x}}\mathcal{L}\right\|_1 \tag{5.3.25}$$

where $\sigma\left\|\nabla_{\mathbf{x}}\mathcal{L}\right\|_1$ is the associated regularization term. Mathematically, the ℓ_1-norm regularization tends to shrink the less important feature's gradients to zero, this mean that this regularization encourages sparsity and thus only selects the important features.

5.4 Projection onto the Weighted ℓ_1 Ball

In this section, we describe the calculation of the projection of a vector onto a weighted ℓ_1 ball. We firstly show how to calculate this projection without constraining the feature values (Section 5.4.1). Secondly, we assume that the features have bounded range values and calculate the projection accordingly (Section 5.4.2).

5.4.1 Box-Constraint-Free Projection

Let $\mathcal{X}$ denote a subspace of $\mathbb{R}^d$. Given two vectors $\mathbf{x} \in \mathcal{X}$, $\mathbf{w} \in (\mathbb{R}^{*+})^d$, and a real number $b_0 > 0$, let $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ denote the $\mathbf{w}$-weighted ℓ_1-ball of center $\mathbf{x}^0$ and radius b_0:

$$\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0} = \{\mathbf{h} \in \mathcal{X} \mid \left\|\mathbf{w} \odot (\mathbf{h} - \mathbf{x}^0)\right\|_1 \leq b_0\} \tag{5.4.1}$$

and $\mathbf{x}^b$ the projection of $\mathbf{x}$ onto $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ the solution of the following minimization problem:

$$\underset{\mathbf{y}\in\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}}{\operatorname{argmin}}\frac{1}{2}\left\|\mathbf{y} - \mathbf{x}\right\|_2^2 \tag{5.4.2}$$

We assume that $\mathcal{X} \neq \emptyset$ to guarantee feasibility and existence of a solution. Many works [122, 123, 124, 125] on unweighted or weighted ℓ_1-constrained problem used a well-known property that allows to calculate the projection onto a weighted ball as long as one can project onto the simplex associated. The following lemma presents a reduction of Problem 5.4.2 to the simplex-constrained problem given in Equation (5.4.3).

Lemma 1 ([122], Lemma 3)**.** *Let $b_0 > 0$, $\mathbf{w} \in (\mathbb{R}^{*+})^d$, $\mathbf{x}^0 \in \mathcal{X}$ and $\mathbf{x} \in \mathcal{X}$ be the vector to be projected. Let $\mathbf{u}$ be a vector obtained by taking the absolute value of each component of $\mathbf{x} - \mathbf{x}^0$, i.e., $u_i = |x_i - x_i^0|$ and $\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ denote the simplex :*

$$\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0} = \{\mathbf{h} \in \mathcal{X} \mid \mathbf{w}^T(\mathbf{h} - \mathbf{x}^0) = b_0 \text{ and } \mathbf{h} \geq \mathbf{x^0}\}$$

Let $\mathbf{x}^s$ be the solution of the following minimization problem:

$$\underset{\mathbf{y} \in \Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{u}\|_2^2 \tag{5.4.3}$$

then the solution to Problem 5.4.2 is

$$\mathbf{x}^b = \mathbf{x}^0 + \mathbf{sgn}(\mathbf{x} - \mathbf{x}^0) \odot (\mathbf{x}^s - \mathbf{x}^0), \tag{5.4.4}$$

where **sgn** *is the signum vector function.*

Proof. The proof is given by [122] with $\mathbf{x}^0 = \mathbf{0}_{\mathbb{R}^d}$, where $\mathbf{0}_{\mathbb{R}^d}$ is the zero vector of $\mathbb{R}^d$ (for simplicity, $\mathbf{0}_{\mathbb{R}^d} = \mathbf{0}$ in the following text). For $\mathbf{x}^0 \neq \mathbf{0}$, we apply the same reasoning to $\mathbf{x} - \mathbf{x}^0$. □

Therefore, we need to solve the problem defined in Equation (5.4.3) whose solution is simpler to compute. The following lemma presents the analytical expression of the solution.

Lemma 2. *Let $b_0 > 0$, $\mathbf{w} \in (\mathbb{R}^{*+})^d$, $\mathbf{x}^0 \in \mathcal{X}$ and $\mathbf{x} \in \mathcal{X}$ be the vector to be projected. Let $\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ denote the simplex : $\{\mathbf{h} \in \mathcal{X} \mid \mathbf{w}^T(\mathbf{h} - \mathbf{x}^0) = b_0$ and $\mathbf{h} \geq \mathbf{x^0}\}$ and let $\mathbf{x}^s$ and $\mathbf{x}^b$ the solutions to Problem 5.4.3 and Problem 5.4.2 respectively. We recall that if $\mathbf{x} \in \Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ then $\mathbf{x}$ is the solution, i.e., $\mathbf{x}^s = \mathbf{x}$ and the same goes for when $\mathbf{x} \in \mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$. If $\mathbf{x}$ is outside $\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ then there exists a unique real λ^* such that:*

$$x_i^s = x_i^0 + \max\{x_i - (w_i\lambda^* + x_i^0), 0\},\ \forall i \in [d] \tag{5.4.5}$$

Using Lemma 1, we thus have; when $\mathbf{x} \notin \mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$; the solution to Problem 5.4.2 defined as follows:

$$x_i^b = x_i^0 + sign(x_i - x_i^0)\ (x_i^s - x_i^0) \tag{5.4.6}$$

where sign is the signum function.

Proof. See Section A.1 in Appendix A. □

We use a *direct* algorithm to compute this projection since d is considered as not large enough to use $\omega - pivot^F$ or $\omega - bucket^F$ that are described as efficient algorithms for high dimensional vectors proposed in [124]. This algorithm which is a generalization of [126] is given in Algorithm 2. The solution provided here can be used in a simulated experiment where feature values are not bounded. However, this cannot represent real-world problems in which the numerical features are mainly bounded in range values.

Algorithm 2 Projection onto the weighted ℓ_1 ball

Require: $\mathbf{x}$, $\mathbf{x}^0$, $\mathbf{w}$, b_0
Ensure: $\mathbf{x}^b = Projection_{\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}}(\mathbf{x})$, the projection of $\mathbf{x}$ onto $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$

$\mathbf{z} \leftarrow \left\{ \frac{x_i - x_i^0}{w_i} | i = 1, .., d \right\}$
$\delta \leftarrow \text{sort} \uparrow (\mathbf{z})$ return the index permutation δ
$\mathbf{z} \leftarrow \{ z_{\delta(i)} | \forall i = 1, .., d \}$
$\varrho \leftarrow \left\{ \frac{-b_0 + \sum_{j=i+1}^{d} w_{\delta(j)} \left(x_{\delta(j)} - x^0_{\delta(j)} \right)}{\sum_{j=i+1}^{d} w_{\delta(j)}} | i = 1, .., d \right\}$
$\varrho\_\mathbf{z} \leftarrow \left\{ \varrho_i - z_{\delta(i)} | i = 1, .., d \right\}$
if $\varrho\_\mathbf{z}.size() \geq 1$ **then**
    $i_{max} \leftarrow \underset{i=1,..,d}{\arg\max} \left\{ \frac{-b_0 + \sum_{j=i+1}^{d} w_{\delta(j)} \left(x_{\delta(j)} - x^0_{\delta(j)} \right)}{\sum_{j=i+1}^{d} \mathbf{w}^2_{\delta(j)}} > z_{\delta(i)} \right\}$
    $\lambda^* \leftarrow \frac{-b_0 + \sum_{j=i_{max}+1}^{d} w_{\delta(j)} \left(x_{\delta(j)} - x^0_{\delta(j)} \right)}{\sum_{j=i_{max}+1}^{d} w^2_{\delta(j)}}$
else
    $\lambda^* \leftarrow \frac{-b_0 + \sum_{j=1}^{d} w_{\delta(j)} \left(x_{\delta(j)} - x^0_{\delta(j)} \right)}{\sum_{j=1}^{d} w^2_{\delta(j)}}$
end if
for $i = 1..d$ **do**
    $x_i^s \leftarrow x_i^0 + \max\{x_i - (w_i \lambda^* + x_i^0), 0\}$
    $x_i^b \leftarrow x_i^0 + sgn(x_i - x_i^0)(\mathbf{x}_i^s - x_i^0)$
end for

5.4.2 Upper- and Lower-Bounded Projection

We modify the problem described above to a box-constrained scenario containing the upper and lower bounds on the vector to be projected. In this case, using the same notations, we need to calculate the projection of a vector $\mathbf{x}$ on a box-constrained ℓ_1-ball $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ of center $\mathbf{x}^0$ and radius b_0. Mathematically, $\mathbf{x}^b$ that denotes the projection of $\mathbf{x}$ on $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ is the solution of the following optimization problem:

$$\underset{\mathbf{y} \in \mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}}{\text{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 \text{ s.t. } \mathbf{a} \leq \mathbf{y} \leq \mathbf{b} \tag{5.4.7}$$

where a_i and b_i are lower and upper bound of the i^{th} feature of $\mathbf{x}$. In this work, we make one assumption on these box constraints: all the intervals $[a_i, b_i]$ are not containing 0, which means that there does not exist j such that $a_j \leq 0 \leq b_j$. This case study is treated by Gupta et al. in [127]. This assumption implies that the respective range values in which y_i should lie can be characterized as $[\mathbf{a}_i, \mathbf{b}_i] \subset \mathbb{R}^{*-}$ or $\mathbb{R}^{*+}$. These two remaining cases are assessed to be equivalent under sign flip (ℓ_2 distance preservation, ℓ_1 norm preservation and range transformation under sign flip). Therefore, we can assume here that all the boundaries are positive. The idea here, is to apply a change of variables on $\mathbf{y}$, $\mathbf{x}$ and $\mathbf{b}$:

$$\hat{\mathbf{y}}, \hat{\mathbf{x}}, \hat{\mathbf{x}}^{\mathbf{0}}, \hat{\mathbf{b}} := \{\mathbf{y}, \mathbf{x}, \mathbf{x}^0, \mathbf{b}\} - \mathbf{a} \tag{5.4.8}$$

Using this transformation, Problem 5.4.7 and its equivalent simpler problem respectively become:

$$\underset{\hat{\mathbf{y}}\in\mathcal{B}^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}}{\operatorname{argmin}} \frac{1}{2}\left\|\hat{\mathbf{y}}-\hat{\mathbf{x}}\right\|_2^2 \text{ s.t. } \mathbf{0}\leq\hat{\mathbf{y}}\leq\hat{\mathbf{b}} \tag{5.4.9}$$

$$\underset{\hat{\mathbf{y}}\in\Delta^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}}{\operatorname{argmin}} \frac{1}{2}\left\|\hat{\mathbf{y}}-\hat{\mathbf{x}}\right\|_2^2 \text{ s.t. } \mathbf{0}\leq\hat{\mathbf{y}}\leq\hat{\mathbf{b}} \tag{5.4.10}$$

where $\mathcal{B}^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}$ and $\Delta^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}$ are respectively the $\mathbf{w}$-weighted ℓ_1 ball of center $\hat{\mathbf{x}}^\mathbf{0}$ and radius b_0 and its simplex associated. The solution to 5.4.9 can therefore be derived from 5.4.10, i.e. the projection onto the simplex associated $\Delta^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}$. This leads to propose the following lemma that presents the final solution to Problem 5.4.7.

Lemma 3. *Let use the notation defined so far, and let $\hat{\mathbf{x}}^b$ and $\hat{\mathbf{x}}^s$ be the solutions to 5.4.9 and 5.4.10 respectively. If $\mathbf{x}\notin\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ which implies $\hat{\mathbf{x}}\notin\mathcal{B}^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}$, then there exists a unique real λ^* such that:*

$$\hat{x}_i^s=\begin{cases}\hat{x}_i^0 & \text{if } \hat{x}_i-\hat{x}_i^0\leq\lambda^* w_i\\ \hat{b}_i & \text{if } \hat{x}_i-\hat{b}_i\geq\lambda^* w_i\\ \hat{x}_i-\lambda^* w_i & \text{if } \hat{x}_i-\hat{b}_i<\lambda^* w_i<\hat{x}_i-\hat{x}_i^0\end{cases} \tag{5.4.11}$$

Based on Lemma 1 and using the inverse transformation associated to 5.4.8, the projection onto $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ $\mathbf{x}^b$ can be written as follows:

$$x_i^b=\hat{x}_i^b+a_i=a_i+sign(x_i-x_i^0)(x_i^s-x_i^0) \tag{5.4.12}$$

where $\mathbf{x}^s=\hat{\mathbf{x}}^s+\mathbf{a}$ and sign is the signum function.

Proof. See Section A.2 in Appendix A. □

From Equation (A.2.9), we can notice that b_0 is a decreasing function of λ, that we denote by g, delimited by $2d$ values namely $\hat{x}_i-\hat{b}_i$ and $\hat{x}_i-\hat{x}_i^0$. As we highlighted in the proof above, we used an approach resulting to an algorithm inspired from [127] to calculate the optimal sub-sets L^*, U^* and C^* with which we can find λ^*. The pseudo code that calculates λ^* is given in Algorithm 3. To do this, we find the uncertainty interval, denoted by $[\lambda^L,\lambda^U]$ for λ, initially set to $[min(\hat{\mathbf{x}}-\hat{\mathbf{b}}),max(\hat{\mathbf{x}}-\hat{\mathbf{x}}^\mathbf{0})]$ (line 2) and iteratively reduced by bisection at a selected pivot from the merged vectors $\hat{\mathbf{x}}-\hat{\mathbf{b}}$ and $\hat{\mathbf{x}}-\hat{\mathbf{x}}^\mathbf{0}$ while belonging to the current uncertainty interval (lines 10-16). As in [128], we set the pivot to the median (line 6) of the uncertainty interval and apply the partitioning around the pivot before each dichotomy step (line 7). Let λ^{pivot} be the current pivot at a particular iteration and $b_0^{pivot}=g(\lambda^{pivot})$. This means:

$$\begin{aligned} b_0^{pivot} &= S_{\mathbf{w},\hat{\mathbf{x}}^0}-\lambda^{pivot}\left\|\mathbf{w}\right\|_2+\lambda^{pivot}\sum_{i\in U\cup L}w_i^2 \\ &\quad -\sum_{i\in U\cup L}w_i(\hat{x}_i-\hat{x}_i^0)+\sum_{i\in U}w_i(\hat{b}_i-\hat{x}_i^0) \qquad (5.4.13)\\ &= S_{\mathbf{w},\hat{\mathbf{x}}^0}-\lambda^{pivot}\left\|\mathbf{w}\right\|_2-\sum_{i\in L}w_i(\hat{x}_i-\hat{x}_i^0)-\sum_{i\in U}w_i(\hat{x}_i-\hat{b}_i) \\ &\quad +\lambda^{pivot}\sum_{i\in U\cup L}w_i^2 \qquad (5.4.14)\end{aligned}$$

Let define the two partial sums that will be used in the algorithm:

$$\Sigma_L = \sum_{\hat{x}_i - \hat{x}_i^0 \leq \lambda^L} w_i(\hat{x}_i - \hat{x}_i^0) \tag{5.4.15}$$

which represents the sum for all projection components that are guaranteed to converge to zero, i.e. the sum over L^*, and:

$$\Sigma_U = \sum_{\hat{x}_i - \hat{b}_i \geq \lambda^U} w_i(\hat{x}_i - \hat{b}_i) \tag{5.4.16}$$

representing the sum for all projection components that are guaranteed to converge to the upper bounds $\hat{b}_i$'s, i.e. the sum over U^*. Using these two partial sums b_0^{pivot} can be expressed as follows:

$$\begin{aligned} b_0^{pivot} = S_{\mathbf{w},\hat{\mathbf{x}}^0} - \lambda^{pivot} \left\|\mathbf{w}\right\|_2 - \Sigma_L - \Sigma_U \\ - \sum_{\lambda^L < \hat{x}_i - \hat{x}_i^0 \leq \lambda^{pivot}} w_i(\hat{x}_i - \hat{x}_i^0 - \lambda^{pivot} w_i) \\ - \sum_{\lambda^{pivot} \leq \hat{x}_i - \hat{b}_i < \lambda^U} w_i(\hat{x}_i - \hat{b}_i - \lambda^{pivot} w_i) \end{aligned} \tag{5.4.17}$$

Let $0 < b_0^{target} < \sum_{i=1}^d w_i(\hat{b}_i - \hat{x}_i^0)$. If $b_0^{pivot} > b_0^{target}$, then the current uncertainty interval is replaced by $[\lambda^{pivot}, \lambda^U]$, and L^*, Σ_L are also updated:

$$L^* = L^* \cup \{i \mid \lambda^L < \hat{x}_i - \hat{x}_i^0 \leq \lambda^{pivot}\} \tag{5.4.18}$$

$$\Sigma_L = \Sigma_L + \sum_{\lambda^L < \hat{x}_i - \hat{x}_i^0 \leq \lambda^{pivot}} w_i(\hat{x}_i - \hat{x}_i^0) \tag{5.4.19}$$

Otherwise, the uncertainty interval is replaced by $[\lambda^L, \lambda^{pivot}]$ and U^*, Σ_U are updated:

$$U^* = U^* \cup \{i \mid \lambda^{pivot} \leq \hat{x}_i - \hat{b}_i < \lambda^U\} \tag{5.4.20}$$

$$\Sigma_U = \Sigma_U + \sum_{\lambda^{pivot} \leq \hat{x}_i - \hat{b}_i < \lambda^U} w_i(\hat{x}_i - \hat{b}_i) \tag{5.4.21}$$

The optimal sub-sets L^*, U^* and C^* are thus found when the uncertainty interval is reduced to two end points without points of discontinuity between them. The unique λ^* is therefore found and the projection is then calculated using Lemma 5.

5.5 Problem Statement

For simplicity, we will adopt a general term, referring to the variable to maximize under a budget constraint as 'expectation' such as the life expectancy in survival analysis, quality variable in wine preference problem, price in the house pricing problem, etc. Consider a regression model f which takes an input $\mathbf{x}^0 \in \mathcal{X} \subset \mathbb{R}^d$ and returns its expectation $f(\mathbf{x}^0) \in \mathbb{R}^+$. For instance, in the context of wine preference modeling, f could be a regression network that estimates the wine quality (positive scalar) with respect to its physico-chemical features (input).

We recall that the main goal is to find the optimal update of $\mathbf{x}^0$ that maximizes the expectation. In addition, in the real-world cases, each feature modification has its own cost and there is a limited budget $b_0 < \infty$ devoted to these modifications. In this work, we make the following assumptions:

Algorithm 3 Box-constrained projection onto the weighted ℓ_1 ball

Require: $\hat{\mathbf{x}}$, $\hat{\mathbf{x}}^0$, $\mathbf{w}$, , $\hat{\mathbf{b}}$ b_0, b_0^{target}
Ensure: $\hat{\mathbf{x}}^s$
merged_data $\leftarrow$ merge($\hat{\mathbf{x}} - \hat{\mathbf{x}}^{\mathbf{0}}, \hat{\mathbf{x}} - \hat{\mathbf{b}}$)
$id\_\lambda^L, id\_\lambda^U \leftarrow 0, 2d-1$ \\ respective indices of λ^L and λ^U
define $S_{\mathbf{w},\hat{\mathbf{x}}^0}$
$\Sigma_L, \Sigma_U \leftarrow 0, 0$
while $id\_\lambda^U > id\_\lambda^L + 1$ **do**
    $\lambda^{pivot} \leftarrow$ median(merged_data, id_λ^L, id_λ^U)
    partition(merged_data, id_λ^L, id_λ^U, λ^{pivot})
    $id\_\lambda^{pivot} \leftarrow$ index(λ^{pivot})
    Calculate b_0^{pivot} using (5.4.17)
    if $b_0^{pivot} > b_0^{target}$ **then**
        $id\_\lambda^L \leftarrow id\_\lambda^{pivot}$
        Update L^* and Σ_L using (5.4.18) and (5.4.19)
    else
        $id\_\lambda^U \leftarrow id\_\lambda^{pivot}$
        Update U^* and Σ_U using (5.4.20) and (5.4.21)
    end if
end while
Calculate λ^* using (A.2.11)
for $i = 1, .., d$ **do**
    Calculate $\hat{x}_i^s$ using (A.2.1)
end for

- *uncontrollable variables :* the features that we cannot modify are frozen, i.e., whose values are not modifiable. Thus we only consider the controllable ones in the problem formulation. Let $\mathcal{C} = \{i | i^{th}$ feature is controllable$\}$, where $[d]$ is the set of integers in the interval $[1, d]$.
- *cost linearity :* the cost of transforming a feature value x_i^0 to $x_i^1 = x_i^0 + \alpha$ is linear with respect to $|\alpha|$.
- *Independent cost variables :* the costs of modifying x_i^0 and x_j^0; $i \neq j$; are independent from each other.

Let $w_i > 0$, for $i \in \mathcal{C}$, be the cost factor associated to the controllable feature of index i, which means that the cost of transforming x_i^0 to $x_i^1 = x_i^0 + \alpha$ is $w_i|\alpha|$. Considering the budget constraints and the assumptions, we seek to solve the following optimization problem:

$$\begin{aligned} & \arg\min_{\mathbf{x}\in\mathbb{R}^d} \text{-}f(\mathbf{x}) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{C}} w_i|x_i - x_i^0| \leq b_0 \\ \text{and} \quad & x_i = x_i^0 \ \ \forall\, i \notin \mathcal{C} \end{aligned} \tag{5.5.1}$$

If we define by extension $w_i = c > 0$ for $i \notin \mathcal{C}$ as indefinitely large such as the modification of the feature i is impossible, the constraint: $\sum_{i\in\mathcal{C}} w_i|x_i - x_i^0| \leq b_0$ will be nothing but a weighted ℓ_1 norm of $\mathbf{x} - \mathbf{x}^0$ bounded above by b_0. Therefore, the solution must be in the weighted

ℓ_1 ball of center $\mathbf{x}^0$ and radius b_0, i.e, $\mathbf{x} \in \{\mathbf{h} \mid \left\|\mathbf{w} \odot \left(\mathbf{h} - \mathbf{x}^0\right)\right\|_1 \leq b_0\}$ where $\odot$ denotes the component-wise multiplication. For this purpose, we use the PGD algorithm to solve Problem 5.5.1. The extra step compared to the classical gradient descent is the projection of the iterate onto the weighted ℓ_1 ball, and this is what ensures that the constraints of the problem are respected.

5.6 Methodology

Algorithm 4 PGD algorithm to solve 5.5.1 for an instance $\mathbf{x}^0$.

Require: ∇f, $\mathbf{x}^0$, $\mathbf{w}$, b_0, n_e \\ ∇f : gradient of f
Require: $\mathbf{a}$, $\mathbf{b}$ \\ when using box-constrained projection
Ensure: $\mathbf{x}^{(n_e)}$, the solution to Problem 5.5.1
1: **if** *box_constrained* **then**
2: $P \leftarrow$ Projection onto $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ with $[a_i, b_i]$ box constraints (Algo 3)
3: **else**
4: $P \leftarrow$ Projection onto $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ (Algo 2)
5: **end if**
6: $\mathbf{x}^{(0)} = \mathbf{x}^0$
7: **for** $k = 1..n_e$ **do**
8: $\alpha_k \leftarrow \frac{b_0}{\|\mathbf{w}\|_1 \sqrt{k}}$
9: **grad** $\leftarrow \nabla f(\mathbf{x}^{(k-1)})$
10: \\ freeze the non-modifiable features
11: $\mathbf{p}_{k-1} \leftarrow \{\ sgn(grad_i)$ if $i \in \mathcal{C}$ else $0\ \}$
12: $\mathbf{y}^{(k)} \leftarrow \mathbf{x}^{(k-1)} + \alpha_k\ \mathbf{p}_{k-1}$
13: $\mathbf{x}^{(k)} \leftarrow Projection(\mathbf{y}^{(k)})$
14: **end for**

Now we are able to calculate the projection of a vector $\mathbf{x}$ onto $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$, we can apply PGD to solve the problem 5.5.1 which is given in Algorithm 4. The algorithm has $\mathbf{x}^0$ as an initial guest (line 1), then calculates n_e iterates where the last one is supposed to be the solution. We use a *diminishing* step size α_k ($\lim_{k\to\infty} \alpha_k = 0$, $\sum_{k\in\mathbb{R}^+} \alpha_k = \infty$) in a such way that, regardless of the number of iterations, the first iterate $\mathbf{x}^{(k)}$ reaches the $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ surface ($||\mathbf{w} \odot (\mathbf{x}^{(1)} - \mathbf{x}^{(0)})||_1 = b_0$) on which the solution is located when f is concave (line 3). Thus, We choose n_e sufficiently large in order to circulate on the surface if f is concave or search the solution inside the ball otherwise. To ensure that the non-controllable features do not change, we activate only the directions provided by the gradient of f on the controllable features (line 6).

We can apply PGD on a regression model f since we can calculate $f(\mathbf{x})$ the expectation of $\mathbf{x}$ and $\nabla f(\mathbf{x})$ the gradient of f with respect to $\mathbf{x}$ for $\mathbf{x} \in \mathcal{X}$. In this work, we recall that we consider three possible scenarios, and thus three groups of models: semi-white box, full-white box and black box.

5.6.1 Semi-White Box

In this scenario, we assume that the network is *semi-white box* which means that the network is already trained and can be only used to calculate outputs and gradients in the PGD

algorithm but we cannot retrain it or modify its architecture. This network, denoted by f_{net}, is constructed as follows:

- f_{net} is consisted of four hidden layers with 128, 64, 32 and 16 nodes respectively.
- Hidden output layers are activated by the ReLU function.
- We use the mean squared error (MSE) as the loss function to minimize, and Adam optimizer to update f_{net} weights.

5.6.2 Full-White Box

We recall that in this case study, we assume that we have the possibility to train the network. The way in which this problem is dealt with in this problem is similar to that of the problem of adversarial attacks [114, 115]. In fact, apart from the weights of $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$, PGD algorithm as described in Algorithm 4 is known as one of the most efficient methods to exploit the weakness of a classical network such as f_{net} to generate attacks. This means that PGD can find a 'fake' solution whose expectation is far from the truth. Actually, updating the initial input $\mathbf{x}^0$ in the PGD algorithm can generate a linear perturbation of f_{net}. This perturbation can be caused by an inherent numerical instability of the learned network [111], which describes the degree to which f_{net}'s output changes after the update step. For instance, $f_{net}(\mathbf{x}^{(k)})$ can be relatively far from the value it should take (and in this case $\mathbf{x}^{(k)}$ is considered as an adversarial input [111, 110]). For this purpose, we introduce a regularization-based technique proposed in [110] as a defense to improve the robustness and the stability of f_{net}. This method consists in penalizing the gradients of the network in the training phase. Choosing the ℓ_1 norm to measure this perturbation, the new loss function ℓ_σ is defined as follows:

$$\ell_\sigma(.) = \ell(.) + \sigma \left\| \frac{\partial \ell(.)}{\partial \mathbf{x}} \right\|_1 \tag{5.6.1}$$

where $\ell(.)$ is the MSE loss function and σ is a regularization strength parameter. We use the same architecture and training protocol as f_{net}. This model is denoted by $f_{net+\sigma}$.

5.6.3 Black Box

Here, we assume that we have the model f as a black box, which means that we cannot change its weights or calculate directly the gradients of f. For this purpose, we use LIME [39] which is a method that builds a sparse linear model around each prediction of f to learn its local behavior. In this scenario, we assume that f_{net} and $f_{net+\sigma}$ are already trained and will be used as black boxes. Below are the steps for fitting LIME to model locally $f \in \{f_{net}, f_{net+\sigma}\}$ around an instance $\mathbf{x}$:

- Generate a vicinity $\mathcal{V}(\mathbf{x})$ of the instance $\mathbf{x}$ following a Gaussian distribution and get their predictions using f.
- Weight the new samples by their proximity to $\mathbf{x}$ using the euclidean distance.
- Fit the linear model on the weighted vicinity and their predictions using Ridge regression.

As LIME is a linear regression model, the gradients are nothing but the regression coefficients (without the intercept term), thus we can apply PGD to this model. By $lime \circ f$, we denote the LIME approach to model locally f.

5.7 Synthetic Experiments

In this section, we will evaluate the precision of the approaches, described above, in calculating the optimal solution of the problem 5.5.1. For this purpose, we run two different experiments on a simulated data generated by a given exact function with which we compare our methods. The synthetic dataset is a matrix $\mathbf{X} = (\mathbf{x}^i)_{i\in[n]} \in \mathbb{R}^{n\times d}$ of size $n = 5000$ with $d = 5$ columns which are all controllable. The samples are drawn from a uniform distribution of support $[0, 1]$. g denotes the exact function that takes $\mathbf{x} \in \mathbb{R}^d$ as an input and calculates $g(\mathbf{x})$ the 'expectation' that we seek to estimate by our models and then maximize it under budget-constraint. We thus generate the expectation vector (target) of $\mathbf{X}$ by g and we train f_{net} and $f_{net+\sigma}$. PGD is then applied on g and each considered model to solve (5.5.1) for $m < n$ instances of indices $J \subset [n]$ picked randomly from $\mathbf{X}$. We obtain for each model f, $\{\hat{\mathbf{x}}^i, f(\hat{\mathbf{x}}^i)\}_{i\in J}$, and for g, $\{\mathbf{x}^{*i}, g(\mathbf{x}^{*i})\}_{i\in J}$ the solution to Problem 5.5.1 for each instance in J. We then evaluate the performance of f by comparing its results with those of g. Once the solutions are calculated, we then evaluate the performance of the eleven models by calculating the following metrics:

- $m_1 = \frac{\|\mathbf{x}^* - \hat{\mathbf{x}}\|_2}{\|\mathbf{x}^*\|_2}$ which measures the gap between the exact solution $\mathbf{x}^*$ and the solution found by f, i.e, how much is the solution of f, far from the exact one;
- $m_2 = \frac{|f(\mathbf{x}^*) - g(\mathbf{x}^*)|}{|g(\mathbf{x}^*)|}$ which measures the degree of precision in which f estimates the exact value of the optimal expectation;
- $m_3 = \frac{|f(\hat{\mathbf{x}}) - g(\hat{\mathbf{x}})|}{|g(\hat{\mathbf{x}})|}$ which evaluates the precision of the estimated value $f(\hat{\mathbf{x}})$;
- $m_4 = \frac{|g(\hat{\mathbf{x}}) - g(\mathbf{x}^*)|}{|g(\mathbf{x}^*)|}$ explains if $\hat{\mathbf{x}}$ can be considered as a solution for an optimal value of g;
- $m_5 = \frac{|f(\hat{\mathbf{x}}) - g(\mathbf{x}^*)|}{g(\mathbf{x}^*)}$ which measures the gap between the optimal expectation calculated $f(\hat{\mathbf{x}})$ using f, and the exact value of the optimal expectation $g(\mathbf{x}^*)$.

Then we plot the mean of these metrics over J for each model. In this section, we use box-constraint-free projection as we are dealing with simulated data and thus do not need to use real-world box constraints.

5.7.1 Experiment 1: Comparing the Black, Semi- and White-Box Performances

5.7.1.1 Experimental Protocol

In this experiment, we consider the following models: f_{net} (semi-white box), $lime \circ f_{net}$ (black box), $f_{net+\sigma}$ (full-white box) and $lime \circ f_{net+\sigma}$ (black box) for $\sigma = 0.1, 0.25, 0.5$ and 0.75. In addition, we test g in two forms: linear and quadratic (the two functions are defined in the caption of Figure 5.2). Therefore, in both cases, we have 10 models to fit (trained in 2500 epochs with a learning rate $lr = 1e-4$ and Adam optimizer), on which we apply the experimental protocol described above. For simplicity, we set $\sigma = 0$ for the semi-white box network $f_{net} = f_{net+0}$. For the black box models, we set the size of the vicinity $\mathcal{V}$ to 5000, generated following a Gaussian distribution of mean $\mathbf{x}$ and standard deviation that of the training set distribution. We choose to set m to 100 in this experiment because LIME-based models take too much time in terms of execution in comparison with the other models, and this is because we need to fit LIME-based models as many times as the number of epochs

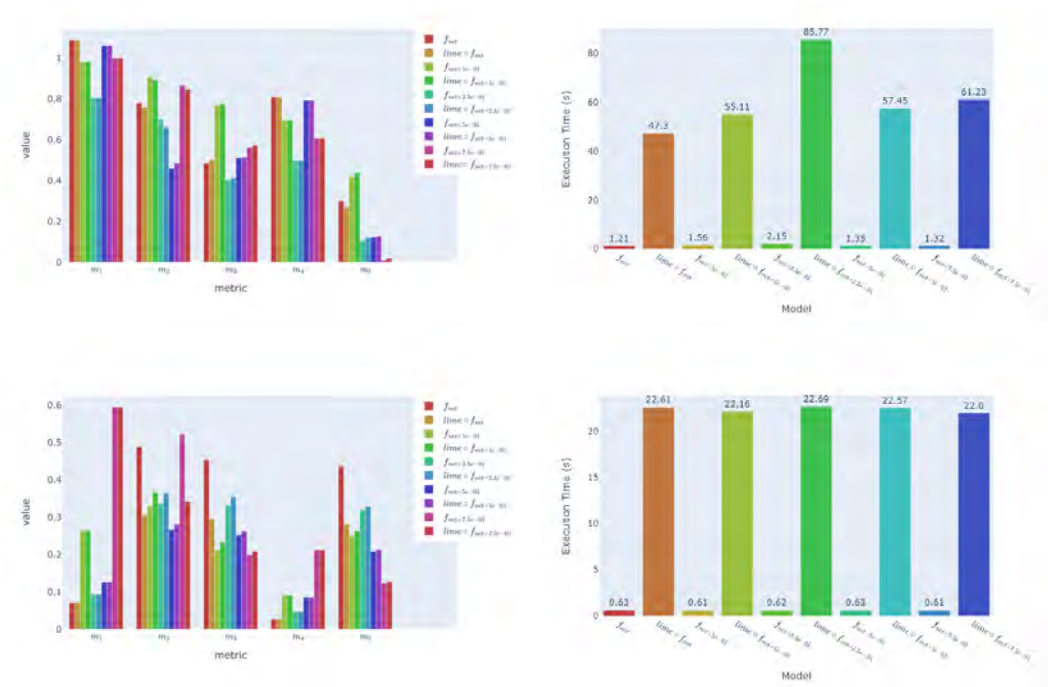


Figure 5.2: The results of Experiment 1: mean values of the five metrics calculated for the 10 models for the linear and quadratic settings. In the top, we set $g(\mathbf{x}) = trace(\mathbf{D} \cdot diag(\mathbf{x})) + b_g$, $g(\mathbf{x}) = \mathbf{x}.\mathbf{D}.\mathbf{x}^T + b_g$ for the linear and quadratic shapes respectively, for each x_i of index $i \in J$, where $\mathbf{D} = diag(\frac{5}{2}, 1, \frac{3}{2}, 1, \frac{1}{2})$ and $b_g = 1e^{-4}$.

n_e, since they are defined locally. For the constraints, we set the cost coefficients of the d columns to $\mathbf{w} = [15, 20, 10, 17, 25]$ and the budget b_0 is fixed at 150. We run PGD with $n_e = 350$ iterations. The results are displayed in Figure 5.2.

5.7.1.2 Results and Discussion

For the case where g has a linear form, we notice, from the values of m_1, that $\hat{\mathbf{x}}$ and $\mathbf{x}^*$ do not coincide, and are relatively far from each other and this is valid for all the tested models. This result implies the fact that m_4 is relatively far from zero for the simple reason that g is linear (and thus has a unique optimum). In addition, according to the values of m_2, $\sigma = 0.5$ provides the best estimation $f(\mathbf{x}^*)$ of the exact optimal expectation $g(\mathbf{x}^*)$, whereas $f_{net+\sigma}$ and $lime \circ f_{net+\sigma}$ with $\sigma = 0.25$ have the closest value to the exact expectation of $\hat{\mathbf{x}}$ according to the values of m_3, and this means that $\sigma = 0.25$ affords more robustness to compared with other values of σ. However, the optimal expectation $f(\hat{\mathbf{x}})$ provided by $\sigma = 0.75$ almost coincide with the exact optimal expectation $g(\mathbf{x}^*)$ ($m_5 = 0.005, 0.02$ for $f_{net+0.75}$, $lime \circ f_{net+0.75}$ respectively) unlike the other values of σ, the semi-white box model and its LIME-based variant ($lime \circ f_{net}$). Therefore, in this scenario, the models did not find the exact optimal solution ($\hat{\mathbf{x}} \neq \mathbf{x}^*$) since m_1 is far from zero; nevertheless, $\sigma = 0.75$ practically provides the same optimal expectation as the exact one $g(\mathbf{x}^*)$.

Concerning the case where g has the quadratic form, the semi-white box model ($\sigma = 0$)

and its LIME variant ($lime \circ f_{net}$) find the closest solution $\hat{\mathbf{x}}$ to the exact one $\mathbf{x}^*$ ($m_1 < 0.1$) but they do not provide the closest optimal expectation $f(\hat{\mathbf{x}})$ to the exact one $g(\mathbf{x}^*)$ which is done by $\sigma = 0.1$. Furthermore, this value of σ provides the most accurate value of the expectation of $\hat{\mathbf{x}}$ according to the metric m_3, but the models with this value of σ have approximately the same performance, in estimating the expectation of the exact solution $\mathbf{x}^*$ as the other models. We can notice, in this case, that the models have more difficulties (in comparison with the linear case) to estimate the optimal expectation.

Performance Comparison Now, let's compare the performance of the full-white box models $f_{net+\sigma}$ with their LIME-based versions $lime \circ f_{net+\sigma}$. For g linear, we notice that both $lime \circ f_{net+\sigma}$ and $\circ f_{net+\sigma}$ models have practically the same scores which means that LIME provides no improvement. Whereas for the quadratic form setting, $lime \circ f_{net+\sigma}$ do not have the same scores as $f_{net+\sigma}$. In fact, for $\sigma = 0$, that is, for the non-robust version, $lime \circ f_{net}$ outperforms f_{net} and thus provides improvement over the black-box model contrary to the linear case. This means that LIME improves the black-box predictions through its local modeling approach. For non-zero values of σ, we can remark that robust versions of the full-white box networks outperform LIME-based methods. This means the gradient regularization improves further robustness against the local modeling of LIME. We can say that LIME and gradient regularization improve the numerical stability of the network predictions with a slight out-performance of the gradient regularization technique that showed a significant improvement, but the latter depends on the value of the strength regularization parameter σ. Furthermore, according to the execution time results of both cases (see Figure 5.2), LIME-based variants take the most time compared to the network-based method (whose PGD execution takes less than one second). When running PGD on the couple ($f_{net+\sigma}$, $lime \circ f_{net+\sigma}$), $f_{net+\sigma}$ take in average 4.2% of the time while LIME-based one take 95.8% of the total time consumed. We can conclude, through Experiment 5.7.1, that both LIME and gradient regularization provide improvement in terms of numerical stability of the networks, but LIME still largely expensive in terms of execution time which affects the speed of PGD calculations that directly depends on the number of LIME calls (number of epochs n_e).

5.7.2 Experiment 2: Effect of Increasing the Weighted ℓ_1-Ball Radius on the Semi- and Full-White Box Performance

5.7.2.1 Experimental Protocol

As discussed in Experiment 5.7.1, for a fixed value of σ, $lime \circ f_{net+\sigma}$ showed practically the same performance as $f_{net+\sigma}$. Here, we apply the same experimental protocol as before to $m = 1000$ samples with five full-white box network models: $\{f_{net+\sigma}\}_\sigma$, $\sigma = \{0, 0.1, 0.25, 0.75, 2.5\}$ trained with the same learning process. Given the multitude of case studies to be tested, not all of which can be included in this work, we keep, for this experiment, the case where g is linear (as defined in the caption of Figure 5.2). We set $\mathbf{w} = (16, 20, 15, 19, 12)$, which means that the most expensive ($\propto 20$) and the cheapest ($\propto 12$) modifications are those of the second feature and the last feature respectively, and run the PGD-based algorithm with three different values of b_0: 150, 350 and 500. The main goal in this experiment is to assess the impact of the radius of $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ on the performance of the models. As in Experiment 5.7.1, once we have the solutions, we calculate and then plot the five metrics $(m_i)_{i=1,..5}$ (Figure 5.3) and we present two additional results:

- For each model $f_{net+\sigma}$, we plot $g(\mathbf{x}^*)$ with respect to $f_{net+\sigma}(\hat{\mathbf{x}})$. This result is more

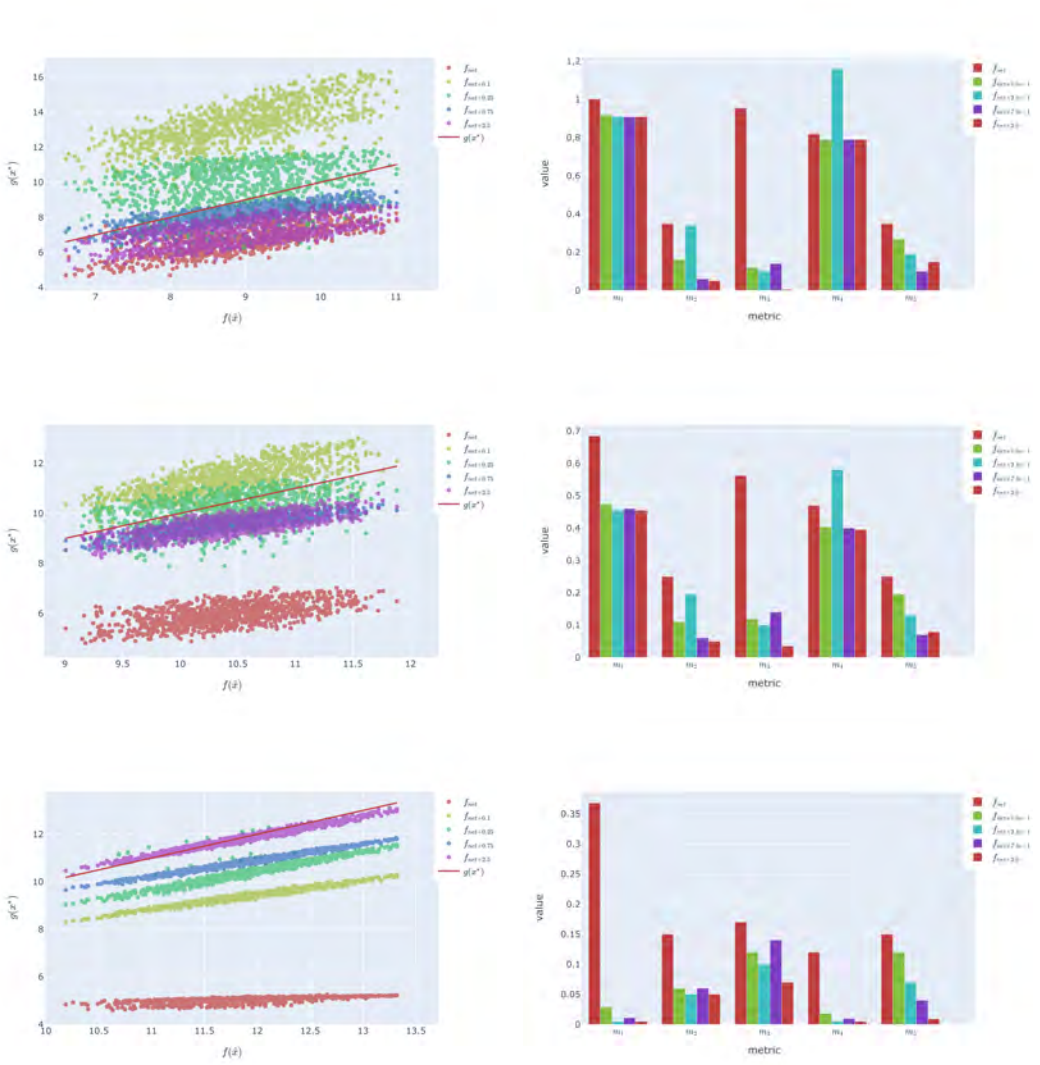


Figure 5.3: The two first results of Experiment 5.7.2 for $b_0 = 300$, 150 and 75 (from top to bottom): visualization of $g(\mathbf{x}^*)$ with respect to $f_{net+\sigma}$ in the first column and mean values of the five metrics associated in the second column.

informative than m_5, because it does not only give the information about the distance between $f_{net+\sigma}(\hat{\mathbf{x}})$ and $g(\mathbf{x}^*)$ but also the nature of the relationship between these two outputs (see Figure 5.3).

- Using the Principal Component Analysis (PCA), we plot the solutions $\hat{\mathbf{x}}$ and $\mathbf{x}^*$ in a reduced 2d space.

Table 5.1: Solutions provided by the exact function g and $f_{net+\sigma}$ for the instance $i_0 \in J$.

Features	1	2	3	4	5	**E**
$\mathbf{x}^{0i_0}$	0.46	0.96	0.32	0.37	0.02	2.97
$\hat{\mathbf{x}}^{i_0}_{\sigma=0}$	0.46	0.96	7.48	0.37	3.57	5.75
$\hat{\mathbf{x}}^{i_0}_{\sigma=0.1}$	0.46	0.96	6.06	0.37	5.33	10.3
$\hat{\mathbf{x}}^{i_0}_{\sigma=0.25}$	5.44	0.96	3.81	0.37	1.5	8.3
$\hat{\mathbf{x}}^{i_0}_{\sigma=0.75}$	**1.82**	**0.96**	**7.51**	**0.37**	**1.71**	**9.2**
$\hat{\mathbf{x}}^{i_0}_{\sigma=2.5}$	1.02	0.96	7.25	0.37	3.11	8.3
$\mathbf{x}^*$	0.46	0.96	0.32	0.37	12.52	9.22

5.7.2.2 Results and Discussion

As illustrated in Figure 5.3, for $b_0 = 300$, $\sigma = 0.75$ (blue points) provides the closest values to the exact ones ($g(x^*)$ denoted by the red line) since the blue scatter plot has a more or less linear shape with a low dispersion, half of which is crossed by the exact straight line in red and the other half forms a relatively small angle with it. $f_{net+\sigma}$ with $\sigma = 0.25$ or 0.1 overestimates the optimal expectation whereas $\sigma = 0$ and 2.5 are underestimating it and for these four values of σ, $f_{net+\sigma}(\hat{\mathbf{x}})$ outputs higher-dispersion scatter plots with different values of dispersion ($\sigma = 0.25, 0.1$ have practically the highest dispersion). Concerning the five metrics calculated for this case study, we notice that any of the models reached the optimal solution $\mathbf{x}^*$ provided by g according to m_1 and m_4. In addition, m_2 and m_3 tell us that $f_{net+2.5}$ is the most robust since it has, in average, the most accurate expectation of $\hat{\mathbf{x}}$ and $\mathbf{x}^*$ but $f_{net+0.75}$ outperforms it in terms of estimating the optimal expectation as shown by m_5 values.

Now, when we shorten the weighted ℓ_1 ball's radius by 150, i.e. $b_0 = 150$, the first thing that we can notice is that the dispersion of the point clouds generated by $f_{net+\sigma}$ have decreased and form a kind of beams with practically the same slope as g. Furthermore, it is apparent that a significant proportion of the green scatter plot ($f_{net+0.25}$ outputs) is crossed by the straight line in red, however, $\sigma = 0.75$ and 2.5 have the best performance (with a slight improvement provided by $\sigma = 0.75$), as illustrated by the values of m_5 in the five-metrics plot associated. Still, $f_{net+2.5}$ is more accurate in calculating $f(\mathbf{x}^*)$ and $f(\hat{\mathbf{x}})$. Similarly to the previous case study, the models do not reach the exact optimal solution $\mathbf{x}^*$ as shown by m_1 and m_4 values.

Finally, when we shorten the radius further by 75 ($b_0 = 75$), the scatter plot that represents $f(\hat{\mathbf{x}})$ become practically straight lines with approximately the same slope as the exact expectation function g (except f_{net}). We notice from the scatter plot, with $\sigma = 2.5$, that $f_{net+2.5}(\hat{\mathbf{x}}) \simeq g(\mathbf{x}^*)$ and for the other non-zero values of σ, $f_{net+\sigma}(\hat{\mathbf{x}}) \simeq g(\mathbf{x}^*) - \epsilon_\sigma$, $\epsilon_\sigma > 0$, which means that these three models underestimate the optimal expectation with different biases. Unlike the two previous case studies, except the semi-white box model, the models $f_{net+0.1}$ and $f_{net+0.75}$ have almost reached the exact solution with $m_1 < 3\%$ and $m_1 < 1\%$ respectively, whereas the other models namely $f_{net+0.25}$ and $f_{net+2.5}$ have reached the exact solution (with $m_1 < 0.1\%$ for both of them). However, $\sigma = 2.5$ is largely outperforming the other models in terms of estimating the expectation of both $\hat{\mathbf{x}}$ and $\mathbf{x}^*$ according to m_2 and m_3 values as well as in terms of estimating the optimal expectation as shown by m_5 values of the associated five-metrics plot.

Discussion of the Impact of Increasing $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ Radius What can be noticed from these results is that the more the radius b_0 of the weighted ℓ_1 ball $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ is increased, which means that the optimization constraint becomes wider, the more the solutions of the network-based models $f_{net+\sigma}$ are less precise (higher dispersion and therefore it looks less and less like a straight line), which is logical since a larger budget implies a greater variability of the features and therefore we are more likely that the distribution of $\hat{\mathbf{x}}$ is significantly different from that of the inputs X^0. In addition, this experiment illustrates a significant improvement, in terms of quality of estimation, provided by the gradient penalization which is usually used as a robustifying technique. Still, the value or the range of adequate σ values depends on the conditions of the experiment, notably: the input distribution, the budget, the standard mode, etc. In figure 5.4, we visualize the expected solutions $\hat{\mathbf{x}}$ and the exact ones $\mathbf{x}^*$ for the $m = 1000$ samples, after applying on them the linear dimensionality reduction using PCA. For simplicity, we mean by $\hat{\mathbf{x}}_\sigma$ the solution $\hat{x}$ provided by the model $f_{net+\sigma}$, for a given σ. For $b_0 = 75$, we notice that almost all the solutions are contained in a large cluster around which 6 small red clusters are formed and which correspond to a significant portion of the solutions provided by $\sigma = 0$, as well as a green cluster even smaller (located in $[0, 20] \times [-60, -40]$) which corresponds to some solutions of $\sigma = 0.25$. These red clusters justify the m_1 score of f_{net} since a substantial proportion of $\sigma = 0$ solutions are relatively far from the exact solutions (in purple) which, all of them, are contained in the big cluster. In the cases where $b_0 = 150$ and 300, we can notice that only the exact function g produced a large and unique cluster (in purple). The other generated models rather several clusters, some of which overlapped with the 'exact' cluster, which explains the m_1 values of all the models.

Discussion of the Solutions Now, we will dig a little deeper in our analysis and look at the different solutions $\hat{\mathbf{x}}^i$ proposed by the models $f_{net+\sigma}$ and compare them with that of the exact function, g for a given instance $i \in J$. Let's look at the case $b_0 = 150$ where the solutions $\hat{\mathbf{x}}$ are in average relatively far from $\mathbf{x}^*$, and select randomly an instance i_0 to analyze its different proposed solutions. The solutions are displayed in Table 5.1. The five first columns represent the feature values and the last column denoted by $\mathbf{E}$ is the expectation. The first row represents the initial values of the features and the expectation calculated by g as follows: $g(\mathbf{x}^{0i_0}) = trace\left(\mathbf{D} \cdot diag(\mathbf{x}^{0i_0})\right) + b_g = \frac{5}{2} \times 0.46 + 0.96 + \frac{3}{2} \times 0.32 + 0.37 + \frac{1}{2} \times 0.02 + b_g = 2.97$. The first thing that we can remark is that for all the models as well as g, the feature values 2 and 4 did not change and this may be due to their relatively high cost coefficients namely 20 and 19 respectively. While, g has only modified the feature 5 as we see in the last row of the table, $f_{net+\sigma}$ changes two or three features (3-5 or 1-3-5). $\sigma = 0$ has the worst performance which was expected given the results in Figure 5.3. $\sigma = 0.1$ modifies the features (3,5) and overestimates the optimal expectation ($f_{net+0.1} = 10.3 > g(\mathbf{x}^*) = 9.22$), whereas $\sigma = 0.25, 2.5$ modify the features (1,3,5) differently and underestimate it with an optimal expectation equal to 8.3 for both of them. Still, these last 3 values of σ have a small relative error in the order of 10% (between 9.7% and 11.7%) unlike $\sigma = 0$ (37.6%). For $\sigma = 0.75$, $f_{net+\sigma}$ found the exact value of the optimal expectation ($f_{net+0.75}(\hat{\mathbf{x}}^{i_0}_{\sigma=0.75}) \simeq g(\mathbf{x}^*) = 9.22$), even they did not change the instance i_0 in the same way. In fact, g spent the entire budget b_0 on the modification of the 5^{th} feature, whereas $f_{net+0.75}$ distributed the budget to the modifications of the features 1,3 and 5, which implies that having $\hat{\mathbf{x}}$ different from $\mathbf{x}^*$, i.e. m_1 relatively far from 0, does not necessarily mean that the optimal expectation is underestimated or overestimated by $f_{net+\sigma}$. The last thing to check, is that for each solution $\mathbf{s} \in \{\hat{\mathbf{x}}^{i_0}_{\sigma=0}, \hat{\mathbf{x}}^{i_0}_{\sigma=0.1}, \hat{\mathbf{x}}^{i_0}_{\sigma=0.25}, \hat{\mathbf{x}}^{i_0}_{\sigma=0.75}, \hat{\mathbf{x}}^{i_0}_{\sigma=2.5}, \mathbf{x}^*\}$: $\mathbf{w}^T(\mathbf{s} - \mathbf{x}^{0i_0}) = 150 = b_0$, which means that the budget is totally consumed.

Table 5.2: Descriptive Statistics of physico-chemical characteristics for both red and white wine

Covariate	Red Wine			White Wine		
	min	max	mean	min	max	mean
fixed acidity (g/dm^3)	4.6	15.9	8.3	3.8	14.2	6.9
volatile acidity (g/dm^3)	0.1	1.6	0.5	0.1	1.1	0.3
citric acid (g/dm^3)	0	1	0.3	0	1.7	0.3
residual sugar (g/dm^3)	0.9	15.5	2.5	0.6	65.8	6.4
chlorides (g/dm^3)	0.01	0.61	0.08	0.01	0.35	0.05
free sulfur dioxide (mg/dm^3)	1	72	14	2	289	35
total sulfur dioxide (mg/dm^3)	6	289	46	9	440	138
density (g/cm^3)	0.99	1.004	0.996	0.987	1.039	0.994
pH	2.7	4	3.3	2.7	3.8	3.1
sulfates (g/dm^3)	0.3	2.0	0.7	0.2	1.1	0.5
alcohol (% vol.)	8.4	14.9	10.4	8	14.2	10.4

5.8 Real-World Dataset Experiment

In this section, we apply our method on two real-world datasets namely: red and white *vinho verde* wine samples [106]. These datasets contain many quantitative variables that describe the physico-chemical characteristics of each wine from both (red and white) types of wines and have a target variable called *quality* which measures the human wine taste preferences. In this experiment, we model the wine preferences under a regression approach using $f_{net+\sigma}$ models described in the previous section, and see how they maximize the quality given the cost coefficient of each modifiable physico-chemical variable and a certain budget. In other words, we use $f_{net+\sigma}$ to solve the problem 5.5.1 where $f \in \{f_{net+\sigma}, \sigma\}$ predicts the wine quality. The goal, here in this section, is just to analyze the optimization solutions provided by the network-based models in a real-world application. To do so, we proceed as follows: first of all, we analyze the wine datasets, by giving brief descriptive statistics of the data per wine type, and calculating the variable importance for both datasets, highlight the different correlations between the variables, that occur for each type of wine. Then, we fit the models and present their performances in order to validate them and thus use them to optimally improve the wine quality. And finally, compute the solutions provided by the models, analyze and compare their respective solutions. Clearly, we use the box-constrained projection in the PGD calculations.

5.8.1 Wine Data

The *vinho verde* wine dataset, is data about wines produced in a Portuguese region called *vinho verde*. In this experiment, we use the two most common variants, white and red wine. During the pre-processing step, the database was transformed in such a way that all observations of a given sample are aggregated into a single row. Furthermore, only the most common physicochemical tests were selected. Since the red and white wines are different in terms of taste, two analyses are performed separately. The red and white wine datasets were built with 1599 and 4898 samples respectively. Cortez et al. [106] give more details about the wine data. The descriptive statistics of both datasets are given in Table 5.2.

Concerning the quality variable, each sample was evaluated by a minimum of three sensory assessors, which graded the wine with a score from 0 to 10. The final score, which is

Table 5.3: Confusion matrix for $t_h = 0.5$ and the two metrics $prec_{t_h}$ and rec_{t_h} for $t_h = 0.5$ and1, calculated using f_{net+1}

Actual	Red Wine						White Wine						
Quality	3	4	5	6	7	8	3	4	5	6	7	8	9
3	1	2	2	0	0	0	3	3	2	0	0	0	0
4	1	12	3	3	0	0	0	30	21	4	0	0	0
5	0	20	151	45	1	0	0	28	315	169	0	0	0
6	0	0	65	134	24	2	0	0	135	576	31	0	0
7	0	0	2	14	48	6	0	0	12	100	173	8	0
8	0	0	1	2	2	3	0	0	3	9	21	24	0
9	-	-	-	-	-	-	0	0	0	0	0	1	1
$prec_{t_h=0.5}$	50	58.8	67.4	67.7	64	27.7	100	49.2	64.5	67.1	76.9	72.7	100
$rec_{t_h=0.5}$	20	63.2	69.6	59.5	68.6	37.5	37.5	54.5	61.5	77.6	59	42.1	50
$prec_{t_h=1.}$	88.2	92.1	95	96	89.8	98.7	100	80.8	84.2	88.4	93	90.9	100
$rec_{t_h=1.}1$	66	83.5	85.6	72.3	84	70.3	71.3	73.3	79.7	94.2	75.6	72.8	81.2

represented by the variable *variable*, is given by the median of these evaluations.

5.8.2 Model Performance

In this section, we evaluate the performance semi- and robust white-full box in wine quality prediction. We first describe the experimental setting used to perform the wine quality classification experiments (Section 5.8.2.1) on the concerned models. Then, we compare the respective performances of the models considered in this experiment (Section 5.8.2.2).

5.8.2.1 Experimental Setting

As the aim is to predict the quality variable which has 6 and 7 possible classes for red and white wines respectively, we could consider the multi-class classification problem. However, since we need to calculate the optimal wine quality after solving Problem 5.5.1, we will rather adopt the regression approach using $f_{net+\sigma}$ whose performances are to be assessed before moving to the next and the final step which is the goal of this section. For this purpose, we need to convert the latter to a float variable, which means that the actual classes of the quality variable become floats and the predictions as well. This approach preserves the order of the wine quality variable. For instance, if the true score is 5, then a model that predicts a value from the range $[5, 6[$ is better than one that predicts a value from $[7, 8[$. Here is the experimental protocol: we use the 2/3 of the data randomly selected to train $f_{net+\sigma}$, and the remaining 1/3 is used for test. Let y_i denote the actual quality of an instance i, $\hat{y}_i$ the predicted value and t_h a threshold value. To assess the performance of the models in the test data, we compute the following metrics:

- *mean absolute error (mae)* defined as follows: $\frac{1}{N}\sum_i |\hat{y}_i - y_i|$, where N the size of the dataset,
- we compute an adapted confusion matrix $C_m^{t_h}$, where the predicted class associated to $\hat{y}_i$, denoted by $C_{\hat{y}_i}$, is given by:

$$C_{\hat{y}_i} = \begin{cases} y_i, \text{ if } |y_i - \hat{y}_i| \leq t_h \\ \underset{y \in \{\lfloor \hat{y}_i \rfloor, \lceil \hat{y}_i \rceil\}}{argmin} |y - \hat{y}_i| \text{ otherwise} \end{cases} \tag{5.8.1}$$

  where $\lfloor . \rfloor$ and $\lceil . \rceil$ denote the floor and ceil functions respectively. This means that we

Table 5.4: Mean Absolute Error and the accuracy acc_{t_h} for two value of t_h: 0.5 and 1, calculated for each value of σ.

σ	Red Wine			White Wine		
	mae	acc_{t_h} (%)		mae	acc_{t_h} (%)	
		$t_h = 0.5$	$t_h = 1.$		$t_h = 0.5$	$t_h = 1.$
0.001	0.55	54.1	79.1	0.58	52.3	80
0.1	0.58	53.23	80.54	0.59	50.2	82.5
0.25	0.45	61.34	89.5	0.46	62.25	88.1
1	**0.39**	**64.15**	**91.24**	**0.45**	**67.3**	**88.9**
2.5	0.5	58.22	86.78	0.53	59.2	87

assign to $\hat{y}_i$ its closest class if $|y_i - \hat{y}_i| > t_h$ otherwise, we assign the class y_i to it. We show the confusion matrix in Table 5.3 of $f_{net+\sigma}$ that provides the best performance.

- Once $C_m^{t_h}$ is calculated, we calculate the precision $prec_{t_h}$ and recall rec_{t_h} for each class (below the confusion matrix in Table 5.3), and the classification accuracy acc_{t_h} (Table 5.4).

We apply this experimental protocol to $f_{net+\sigma}$ with σ= 0.001, 0.1, 0.25, 1 and 2.5.

5.8.2.2 Results

From Table 5.4, we notice that f_{net+1} has the lowest thus the best mae for both datasets, and also outperforms the other models in terms of accuracy, especially for $t_h = 0.5$. Furthermore, $\sigma = 0.25$ provides the second best performance whose results are close to those of $\sigma = 1$ especially for white wine dataset, and $\sigma = 2.5$ slightly under-performs the two last models, but has significantly better results than σ= 0.001 and 0.1. From the values of $prec_{t_h}$ and rec_{t_h} confusion matrix $C_m^{t_h=0.5}$ calculated for the best model, we notice that $f_{net+\sigma}$ which is a regression approach perform a good multi-class classification especially when $t_h = 1$. Therefore, to solve the problem 5.5.1 for both datasets, we use $f_{net+\sigma}$ with σ=1, 0.25 and 2.5. To calculate the global variable importance (Figure 5.5) for both datasets, we use f_{net+1} as the black-box model for LIME.

We can notice from Figure 5.5 that the contributions of the variables are different within each wine type. For instance, the citric acid and residual are more important in white wine, whereas pH level has a big impact on the quality of the red wine (color, taste, and smell). However, sulfates have the largest (positive) contribution and an increase of alcohol tends to result in a higher quality for both cases.

5.8.3 Wine Quality Optimization

In Section 5.8.2, we assessed the performance of the semi- and robust full-white box in terms of wine quality prediction. Now, we will apply our methodology to wine instances (red and white) and analyze and discuss the results. In this experiment, since we are dealing with real-world datasets, we will use the box-constrained projection method in PGD calculations. We first describe the experimental setting (Section 5.8.3.1). Then, we present, discuss and compare the results of different models (Section 5.8.3.2).

Table 5.5: Modifications provided by $f_{net+\sigma}$, $\sigma = 0.25$, 1 and 2.5 in terms of volatile acidity, pH and alcohol to improve the quality of the instance ins_{red}.

	Volatile Acidity	*pH*	*Alcohol*	*Quality*
initial	0.59	3.52	11.4	5
$\sigma = 0.25$	0.31	4.08	14.16	6.42
$\sigma = 1$	0.505	3.62	15.93	6.98
$\sigma = 2.5$	0.45	4.07	14.46	6.1

5.8.3.1 Experimental Setting

We recall that particular wine regardless of the type (red or white), which is recorded in the data is defined by a vector containing values of physico-chemical variables. Some of these physico-chemical variables can be controlled to improve the (individual) wine quality. For example, alcohol concentration can be increased or decreased by monitoring the grape sugar concentration prior to the harvest [106]. Also, by suspending the sugar fermentation, the residual sugar could be increased in white wine. It is also possible to monitor pH levels periodically throughout the wine production process for the red wine that should be less acidic (pH between 3.4 and 3.8). In addition, it is possible to remediate the volatile acidity using Reverse Osmosis that lowers the acetic acid concentration. However, even the total sulfur dioxide has a big contribution in the red wine, controlling it is no easy task since it implies controlling sulfur-free in the wine, and the portion that is bound to other chemicals. To optimize the wine quality, we solve the problem 5.5.1 using $f_{net+\sigma}$, σ=0.25, 1 and 2.5. For the red wine, we choose to control pH, alcohol, and volatile acidity whose cost coefficients are, respectively, set to 3, 1, and 2 whereas for the white wine, we choose alcohol, residual sugar and citric acid with cost coefficients: 1, 1 and 3 respectively. We set the budget b_0 to 5. The values of cost coefficients represent nothing in reality and we assume that variables that are hard to control are more expensive (relatively higher cost coefficient value) such as pH which is not easy to monitor. For *Alcohol, Citric Acid, pH, Residual Sugar and Volatile Acidity*, we set respectively the box constraints while providing a real-world setting[1][2]: $\mathbf{a} = (11, 0.1, 3.4, 11, 0.3)$ and $\mathbf{b} = (16.5, 0.35, 4.2, 16, 0.61)$. We randomly take two instances ins_{red} and ins_{white} from the red and white wine datasets respectively whose quality variable value is 5. Then, we apply PGD using box-constrained projection (with box constraints defined by $\mathbf{a}$ and $\mathbf{b}$) on $f_{net+\sigma}$ to find the optimal modifications in ins_{red} and ins_{white}. The results are shown in Table 5.5 for ins_{red} and Table 5.6 for ins_{white}. For both tables, the first row corresponds to the initial values of the instance, whereas the three last rows correspond to the modifications and the optimized quality provided by $f_{net+0.25}$, f_{net+1} and $f_{net+2.5}$ respectively.

5.8.3.2 Results and Discussion

The first thing that we can notice, is that the three models increase the alcohol level in both wines. These modifications are acceptable since an increase in alcohol (which is among the most relevant attribute for both types of wine) tends to improve the wine quality. For the red wine, the models decrease the volatile acidity, which is a good thing because it has a negative impact on the quality since the acetic acid makes the taste more vinegary [106].

[1] https://winemakermag.com/
[2] https://www.homebrewit.com/

Table 5.6: Modifications provided by $f_{net+\sigma}$, $\sigma = 0.25$, 1 and 2.5 in terms of citric acid, residual sugar and alcohol to improve the quality of the instance ins_{white}.

	Citric Acid	*Residual Sugar*	*Alcohol*	*Quality*
initial	0.24	12.1	9.5	5
$\sigma = 0.25$	0.15	14.76	11.56	7.88
$\sigma = 1$	0.24	12.1	14.5	7.38
$\sigma = 2.5$	0.24	14.86	11.74	6.98

They also increase pH which means that the wine becomes less acidic and thus tastier. Of the three models, f_{net+1} finds the best pH (3.62) for a good red wine because it decreases acidity while staying in the recommended range namely: 3.4-3.8. By decreasing volatile acidity and increasing alcohol and pH, if we round the optimal quality values, $f_{net+0.25}$ and $f_{net+2.5}$ could improve the quality by 1 and f_{net+1} by 2. For the white wine, ins_{white}, f_{net+1} only modifies increase the alcohol by 5% to improve the quality by 2 (rounded), while $f_{net+2.5}$ increases the residual sugar and alcohol by roughly 2 to provide practically (by rounding) the same improvement. On the other hand, $f_{net+0.25}$ not only increases; in the same way as $f_{2.5}$; the alcohol and the residual sugar but also decrease the citric acid, which is an interesting result since the citric acid and residual sugar levels are more important in white wine, where the balance between the freshness (controlled by the citric acid) and sweet taste (provided by the residual sugar) is more appreciated.

5.9 Conclusion

This work, principally motivated by the predictive maintenance world, presents an approach for any regression problem where the goal is to maximize the regression output with respect to the controllable features under given budget constraints. The main idea is to translate this problem into an optimization problem under weighted ℓ_1-constraints whose solution is the optimal update of the controllable features that maximize the output. We solved this optimization problem numerically using PGD. We proposed three scenarios namely: semi-white box model pre-trained, full-white box 'robustified' using gradient regularization, and a black box case modeled linearly and locally LIME. In the synthetic experiments, we have shown, by comparing with the exact functions, that the robust network outperformed the two other models. Still, the performance of the robust network depends on the value of the regularization strength parameter. In the real-world experiment, we discussed the different solutions provided by different values of the regularization parameter. An important direction for future works would be to take into account the case where the lower and upper bounds of the box constraints have opposite signs, which means box constraints containing zero without necessarily being an endpoint. Because, actually in some real-world datasets, there are some variables that can have both positive and negative values (e.g., temperature). This scenario can be important since we could have a case where the values required to optimize the output are not in the range values defined by the box constraints.

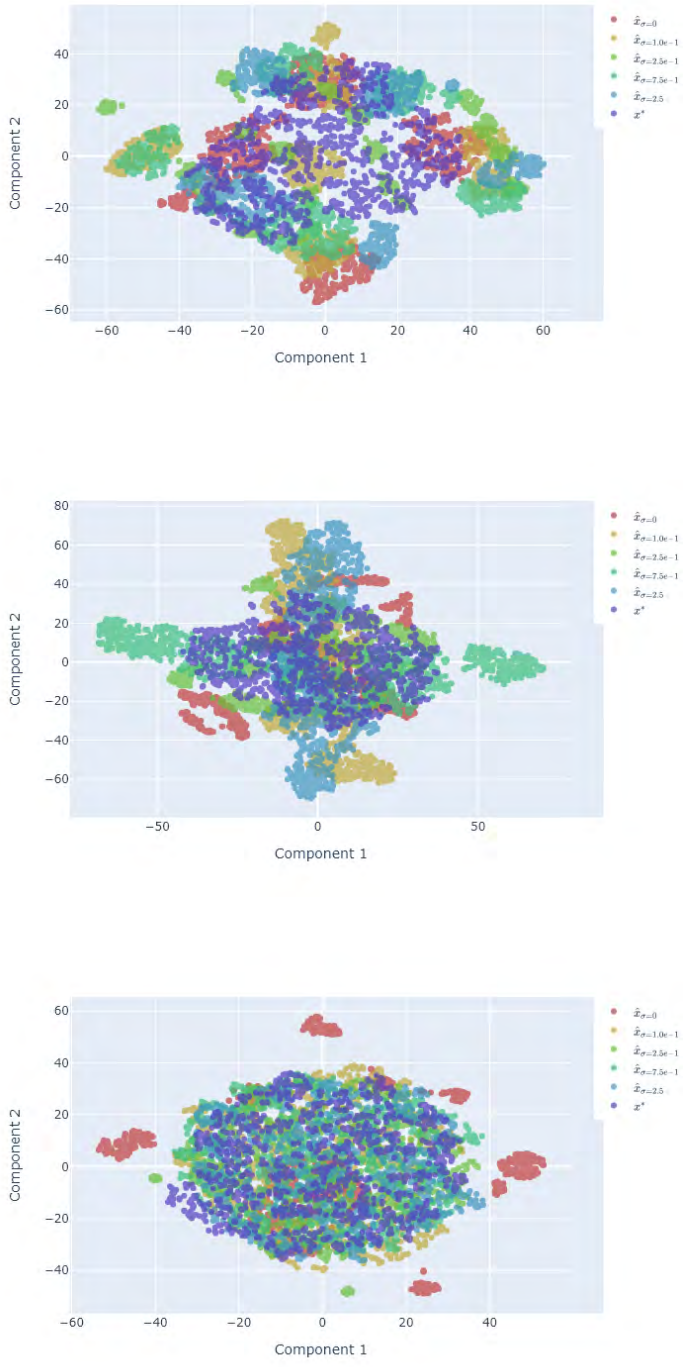


Figure 5.4: Visualization (in two dimensions using PCA) of the solutions $\hat{\mathbf{x}}$ of the four σ-regularizer models and the semi-white box one, as well as the exact solutions $\mathbf{x}^*$ for three case studies (from top to bottom): $b_0 = 300$, 150 and 75.

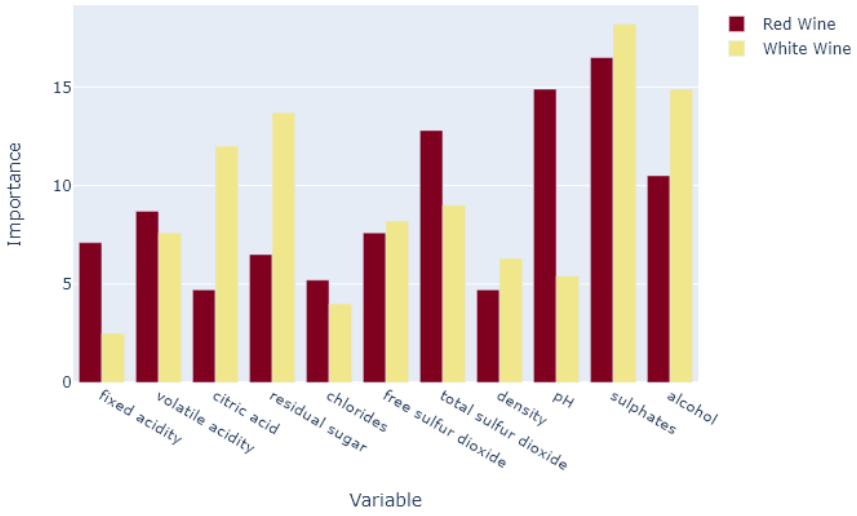


Figure 5.5: Global variable importance using LIME with f_{net+1} for the red and white wine datasets.

Chapter 6

Real-World Application : SmartOccitania Project

6.1 Introduction

In parallel with the thesis, more precisely in the first two years, we worked on a multi-stack-holder project, called Smart Occitania. The problem that we studied in this project which is: developing AI algorithms for predictive maintenance on low-voltage feeders, can be considered as concrete and complete application of the main problems addressed in this thesis, from a survival analysis study to a maintenance strategy proposition. In fact, the main objective of this part of the project was to identify locally the most-at-risk rural distribution networks (of the region of Occitania) in terms of power failures, or in other words, those with the highest probability of experiencing a power failure in the next three years and detect the respective associated causes. We use, for this purpose, the information provided by the communicating objects placed on the electrical networks as well as the inherent characteristics of the low-voltage feeders. In this chapter, we conduct a full study where, first of all, we analyze the sets of data provided for this purpose and conduct an experiment to evaluate the most common models as well as our two approaches in terms of estimating the failure risks, taking into account the cause behind. Then, we move to the second stage, where we run an experiment, using our methodology as described in the last chapter, to propose the optimal modification to optimize the mean lifetime of the most-at-risk low-voltage feeders.

This chapter is organized as follows. In Section 6.2, we globally describe the project, e.g. its major challenges, the principal actors, objectives, etc. Section 6.3 is devoted to the survival analysis study, where we first analyze the sets of data provided for this purpose, then conduct an experiment, using four network-based models including our two approaches namely: DeepWeiSurv and DPWTE, to evaluate their performance in terms of estimating the risk and identifying the most-at-risk feeders. For the latter, we introduce a novel score that we describe in the same section. In Section 6.4, we conduct an experiment where we apply our constrained-optimization-based methodology to propose locally, i.e. at the low-voltage feeder level, the optimal modifications that should be operated to maximize the mean lifetime. We conclude this chapter in Section 6.5.

6.2 Description of Smart Occitania Project

The Smart Occitania project is a demonstrator of smart electrical networks in rural areas based on the infrastructure of public electricity distribution networks enriched with an additional telecom infrastructure[1]. Enedis[2] in the region of Occitania and its partners have won the call for projects launched by ADEME[3] as part of the "Intelligent Electric Networks" program of the future investments. Around a coherent set of experiments, the project integrates a logic of industrial development, regional development, and energy planning. The Occitania region is particularly well suited for this type of project because, on the one hand, the potential for renewable energies is important, and on the other hand, the electrical network is extensive and overhead, which makes it sensitive to climatic hazards. The Smart Occitania project answers two strong stakes:

- Participate in the success of the strategic project of the Occitania region to become the 1st positive energy region in 2050.
- Improve the quality of electricity supply by reducing the average time of interruption per customer, that is, by deploying an efficient predictive maintenance strategy to avoid long interruptions at the customer level.

This project is expected to be built around three experimental studies of a technical, industrial, scientific, and societal nature, carried out in all the departments of the Occitanie region. The respective objectives of these three studies can be summarized as follows:

- Renewable Energy Deployment (RED): the main objective is to facilitate the integration of electricity production from renewable sources, which is intermittent by nature. The output of the RED project is the increased capacity of the electricity networks to receive renewable energy networks. To perform this, the production and consumption of sites with non-electric storage (e.g., methane plants) will be regulated according to the constraints of the network.
- Involving all stakeholders in the energy transition: the objective is to raise awareness among private customers and small professionals on the topic of the energy transition. To do this, the Smart Occitania project launched an important survey to evaluate the level of knowledge of the residents on the subject of the energy transition. Then, the project team put at the disposal of the customers and communities an educational program on the energy transition adapted to their level of knowledge. The first feedbacks will allow feeding a network of local companies, acting in favor of the energy transition, and participating in the calls for projects of the region of Occitania.
- Improve the observability of the public distribution network in rural areas: composed of several sub-studies, the technical part of Smart Occitania will allow working on several themes linked to the advanced functions of a rural distribution network. Among these sub-studies was one whose goal is to highlight the role of AI in predictive maintenance. In fact, the electricity suppliers seek to identify the most-at-risk areas, that is, those which contain low-voltage feeders that are likely to experience a power failure within a handful of years.

We, as researchers, were brought to realize one of the technical projects of Smart Occitania, namely the use of AI to implement an algorithm on predictive maintenance. More precisely, we used a deep learning framework for this purpose.

[1] https://www.smartgrids-cre.fr/projets/smart-occitania
[2] https://www.enedis.fr/enedis-en-midi-pyrenees-sud
[3] https://www.occitanie.ademe.fr/

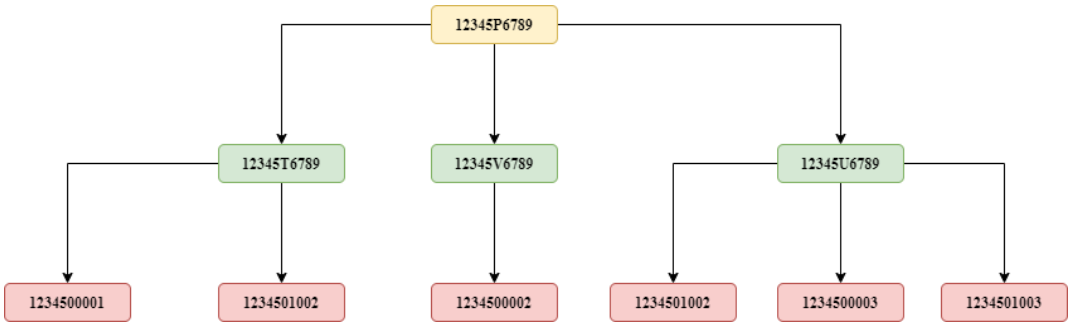


Figure 6.1: Illustration of an electrical network structure as defined in the region of Occitania.

6.3 Detection of Competing Risks of Power Outage

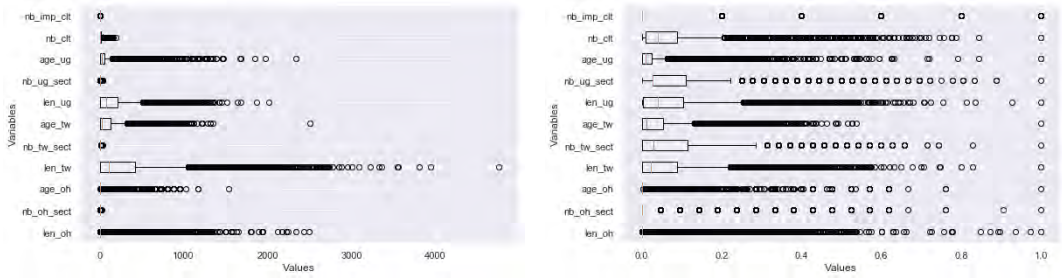
In this section, we will describe a study on a survival analysis problem that we carried as a part of a collaborative project, the so-called *Smart Occitania*, driven by ENEDIS[4]. One of the goals was to improve, using a deep learning framework, the estimation of the risk of a power failure and its potential cause, on the low-voltage side of a given feeder, in rural areas of the region of Occitania. This means, that we are dealing with a competing risks problem, i.e., the failure events can be due to different causes from a feeder to another, unlike all the experiments conducted in the previous chapters. The reason behind this motivation, since the budget for repairs is limited, is to have the ability to schedule maintenance for the most-at-risk low-voltage feeders and thus minimize the inferred costs. For this purpose, we conduct an experiment on the data provided by ENEDIS where we test (in competing risk setting), evaluate, and compare the respective performances of our two approaches as well as those of DeepHit and a fully connected network (FCN).

6.3.1 Data Description

As said above, our main goal in this project was to predict the risk of power failure at the lower-voltage feeder level. Before describing the data that we have at our disposal, we briefly describe the structure of the electricity network deployed in the region of Occitania. In fact, the low-voltage feeders, which are the subjects of study, form geographical clusters each of whom is connected to a high-voltage-to-low-voltage transformer (HV/LV) substation. The transformer sub-stations are in turn linked to different high-voltage stations. Figure 6.1 is an illustration (does not represent a real example for purposes of confidentiality) of the electrical system structure deployed in this region. As shown in this figure, the high-voltage station (in yellow) is connected to three HV/LV transformer substations (in green) which are in turn connected to two, one, and three respectively (in red).

To construct the link in the data, each low-voltage (LV) feeder, high-voltage to low-voltage (HV/LV) transformer sub-station, and high-voltage (HV) station is defined by an identifier that we call GDO code. For a low-voltage feeder, a GDO code is nothing but a string of roughly 9 to 10 digits, whereas, for HV/LV transformer sub-station and high-voltage station, a GDO code is defined in the format 'XXXXXYXXXX' with X a number from 0 to 9, $Y = P$ to designate HV station and $Y \in \{R, S, T, U, V\}$ for HV/LV transformer sub-station.

[4]https://www.enedis.fr/actualites/reseau-electrique-intelligent-en-milieu-rural-le-projet-smart-occitania-devoile-resultats



(a) Before normalization (b) After normalization

Figure 6.2: Distribution of the variables describing low-voltage feeders before and after normalization

To conduct this study, we were provided with three types of data within the same period 2011-2016:

- Data on all the low-voltage feeders describing the respective characteristics and the respective number of clients. Each low-voltage feeder in the data, is defined by an identifier consisted of a string of numbers.
- Aggregate data at high-voltage station level.
- History of power failures that occurred during the period under study.
- Additional data on the pruned zones: this data contains information about the zones whose trees are pruned and cleaned up. This includes the dates of the operations, the concerned feeders, the size of of the pruned portions, etc.

6.3.1.1 Data at the Low-Voltage Feeder Level

This dataset contains, aside from the GDO code of the feeders and those of their respective corresponding lines and stations, information about the characteristics of the low-voltage feeders and the number of clients per feeder, recorded during a period of 6 years (2011-2016), which means that each feeder is recorded once per year. A feeder can consist (but not exclusively) of overhead ('oh'), twisted ('tw'), or underground ('ug') sections. Table 6.1 gives a description of each numerical variable recorded. We point out that we removed variables that are functions of some of those considered. We also removed variables whose values are mainly blank (NaN values). As we see in Figure 6.2a, the variables do not have the same order of range values which can be problematic because these differences will affect the training of the networks. For this purpose, we normalize the data and thus obtain the new distributions as seen in Figure 6.2b.

6.3.1.2 Aggregate Data at the High-Voltage Station Level

We have at our disposable 6 sub-datasets correspondings to data on high-voltage stations recorded each year during the period 2011-2016. This data describes the modification done, during this period, at the high-voltage station level. The reason behind this modification is to minimize the risk of a power failure. Aside from the GDO code, the data has four additional attributes that describe the state of the station, namely: the length of the overhead,

Table 6.1: Description of the low-voltage feeder variables.

Variables	Description
len_oh, len_tw, len_ug	total length of the overhead, twisted and underground sections respectively
nb_oh_sect, nb_tw_sect, nb_ug_sect	number of overhead, twisted and underground sections respectively
age_oh, age_tw, age_ug	age of overhead, twisted and underground sections respectively
nb_imp_clt	number of important clients
nb_clt	number of current clients

Table 6.2: Failure rate of grp1 and grp2 for each year from the period under study

Year	rate_grp1 (%)	rate_grp2(%)
2012	2.12	6.6
2013	1.02	3.43
2014	0.8	2.92
2015	0.44	1.71

underground and twisted wires as well as the length of the 'weak' section of the overhead wire. This data is considered as aggregate data because these attributes are aggregated, that is, each value of a variable for a station in this data is nothing but the sum of the values of the corresponding variables over the feeders that belong to that station. In order to minimize the risk for the coming years, three possible modifications can be done, but not exclusively: reduce the overhead wire, add underground or twisted sections. To check the impact of these modifications on the failure rate, we have done the following test: for a given year ye, we divided the population of this data into two sub-groups, the first one (denoted by grp1 of size 5000) contains stations experiencing a modification in this year and the second one (denoted by grp2) is a similarly-sized sub-group of stations randomly selected from the complementary of the first one. We then calculated the failure rate over the coming years ($> ye$) in the period and obtained the results as shown in Table 6.2. As we see, it is significantly less likely that the failure occurred in grp1 than in grp2. As with the first data, we apply normalization on these four variables to eliminated the biased contributions of the high range values.

6.3.1.3 History of Power Failures Recorded in the Period under Study

This data contains all the records of failures that occurred during the period 2011-2016. It has the following attributes: the date of the incident, the GDO code of the concerned low-voltage feeder, the cause behind the incident, and the date of the last maintenance. The latter is useful to implicitly have the information of the mean lifetime. The time observed t that we use for modeling the failure risk is the difference between the date of incident and that of the last maintenance (in days). Two insights to highlight :

- The first point is that we have different causes behind the failure (hence the 'cause' variable), and this means that we are in a competing risk setting.
- The second point is that the feeders recorded in this data are (relatively small) sub-set of the set of feeders recorded in the first data, which means that failure, regardless of

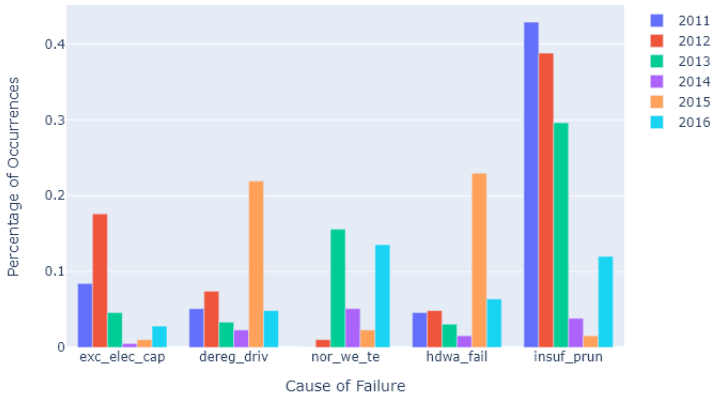


Figure 6.3: Failure distribution of the period under study.

the cause, is a rare event and thus we're dealing with censored data.

These are two challenges that we are going to face in this study. Five possible causes can be found in this data, namely:

- Exceeding electrical capacities (exc_elec_cap).
- Deregulated drivers (dereg_driv).
- Normal wear and tear (nor_we_te).
- Hardware failure (hdwa_fail).
- Insufficient pruning (insuf_prun).

In this study, we are interested only in the internal causes, i.e. the failures that are due to an internal dysfunction (e.g., hardware failure) and anomaly around the feeders (namely insufficient pruning). Other causes such as weather conditions are deleted due to lack of data. In Figure 6.3, we show the distribution of different causes of failures recorded during the period under study. We notice that the numbers of occurrences of the failure, regardless of the cause behind it, are very small (in the order of $10^{-2}, 10^{-1}$) which confirm the fact that a failure is a rare event.

6.3.1.4 Combined Data

As a reminder, we have at our disposal 4 different data: feeder data, station data, history data, and extra data on pruned zones. The last is only used when we want to estimate the risk of failure of cause: insufficient pruning. In order to exploit the evolutionary aspect of the stations, we merge the first two data using the feeder GDO code as a primary key resulting in data used as the baseline data for our experiment in this section. Then for each year from the period under study, we have a set of low-voltage feeder observations, consisting of the features of the merged data and a target object consisted of the history associated, with five

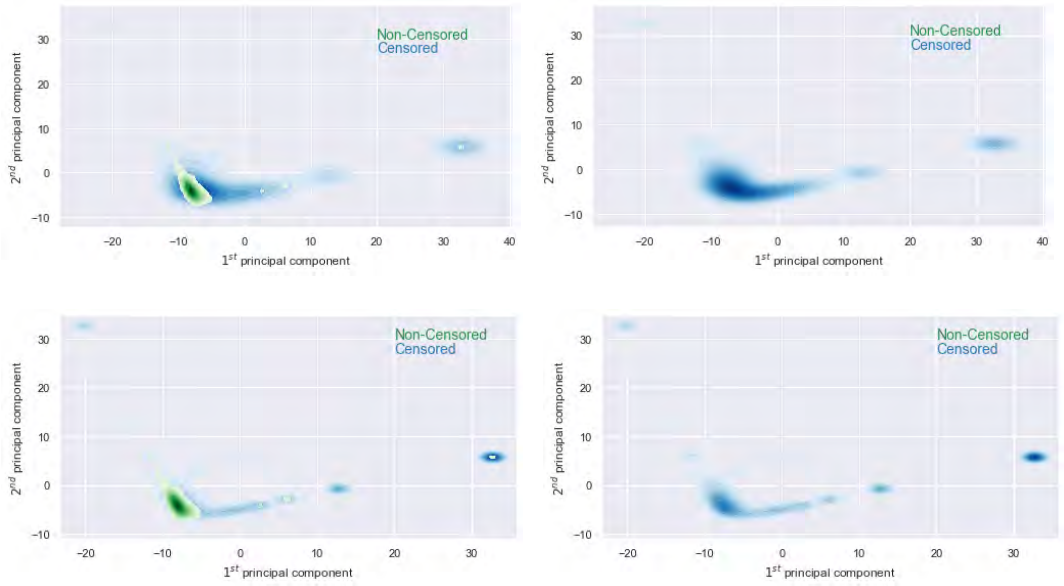


Figure 6.4: Distribution of censored and non-censored populations in the two defined scenarios: in the top, distribution of the sub-populations without station-level data. At the bottom, their corresponding distributions using station-level data. Since, in each scenario, there is an overlap between the respective density modes of the two sub-populations, we plot the result twice but with highlighting a different density mode in each time.

features (target matrix) corresponding to the five potential causes considered in this study, or one column (target variable) in the case of power failure estimation regardless of the cause behind. To show the contribution of the high-voltage station data to the predictability of the failure risk, regardless of cause, we run the following test considering two scenarios: one where we use only the low-voltage feeder data, and the second where we use the combination. In each scenario, we divide the data into two censored and non-censored sub-population and then calculate the kernel density estimation for each sub-population to visualize the spatial distribution of the censored and non-censored populations (as shown in Figure 6.4). To reduce the dimensionality of the data for visualization, we apply PCA[5] on the data.

As we see in Figure 6.4, in the first scenario where we use only the observations at the feeder level, the two density modes (located in $[-10, 0] \times [-10, 0]$) coincide, which means that there is practically any indicator that helps the models to predict the risk and thus the estimation will be done almost randomly. Whereas in the second scenario, the respective modes are separated in space (the non-censored one is located in $[-10, 0] \times [-10, 0]$ while the censored one in $[30, 40] \times [0, 10]$) and this means thatt the data at the station level provides a non-zero contribution to the predictability of the risk. However, the density of the non-censored population is totally covered by that of the censored one which means that the failure estimation task is still challenging.

[5]Principal Component Analysis

6.3.2 Experiment: Power Failure Prediction

In this section, we describe the experimental scenario that we apply to the models tested in this study, then we define the metric used to evaluate the performance of these models. Next, we define the evaluation strategy, and finally, we discuss the results of this experiment.

6.3.2.1 Problem Statement

As a reminder, the main objective of this study is to predict the probability that a given low-voltage feeder experiences a power failure, or in other words, model the run-to-failure probability distribution at the individual scale. For this purpose, we consider two scenarios: first, we model this probability distribution regardless of the type of cause behind the failure. It will therefore be a binary classification problem. Second, we model the run-to-failure probability distribution for each type of cause, i.e. we will predict not only the probability that a given feeder will break down but also the likelihood of each cause considered. As seen in Section 6.3.1.3, the power failures, all causes combined, that were recorded during the period under study are considered as rare events, which is challenging the models to predict the likelihood of failures and even more when it comes to predicting the cause (events, when classified by cause, are even rarer).

6.3.2.2 Models

Since we are in a competing risk setting with five different potential causes where we seek to learn the joint distribution and not the marginal one, we are dealing with a multi-class (but not multi-label) classification problem. This means that the models are expected to output five probabilities, each one represents the probability that a given feeder experiences a failure of a given cause. For this purpose, we use in this experiment four network-based models, one of which is a standard classifier while the remainder is survival methods:

- *Fully Connected Network FCN* : consisted of four hidden layers with 64, 32, 16, and 8 nodes respectively, and a softmax output layer. For the first scenario, we set 1 node in the output layer and 5 (corresponding to the number of potential causes) for the second scenario.
- *DeepHit* [30] : for the purpose considered in this study, DeepHit will consist of a shared sub-network linked to m cause-specific sub-networks where each sub-network k learns the probability of experiencing the k^{th} cause. A single softmax layer is used as an output layer that stores the probabilities learned by the m cause-specific sub-networks. We set m at 1 and 5 for the first and second scenarios respectively.
- *DeepWeiSurv* : by definition, DeepWeiSurv handles competing risks. The (multi-class) classification is performed using the weighting coefficients. We set p the mixture size at five for both cases in this study.
- *DPWTE* : like DeepWeiSurv, DPWTE can be used in a competing risk setting via its weighting coefficients. We set $p_{max} = 10$ and $\lambda = 10^{-4}$ for both scenarios.

We apply the ReLU activation function on all the hidden layers of the models cited above.

6.3.2.3 Evaluation Metrics

We recall that the reason behind this study is to detect the most-at-risk low-voltage feeders to make them a priority in the maintenance program, given the limited budget allocated for

this purpose. This means that we have to evaluate the predictive performance of the models taking into account this real-world budget constraint. In this case, the time-dependent concordance index C^{td} [56] is not the most suitable evaluation metric, or at least not sufficient, because it only describes the likelihood of concordance of pairs of failures. This means that it evaluates the probability that for a pair of failures, the respective predicted time-to-failure have the same ordering as the real ones, but this score does not establish any priority between the feeder with respect to the failure probability. Still, this score is useful to evaluate the model performance in the first stage for the binary classification problem.

The same goes for the Top-k Accuracy classification score in the multi-class classification problem, but we use it to assess the performance of the models in the second scenario. We recall that Top-k Accuracy[6] (denoted here by Top$_A$-k) computes the number of times where the correct class is among the top k labels predicted, in decreasing order of probability.

Let $\hat{y} \in [0,1]^{n \times m}$ is the prediction matrix where n is the size of the population under study and m corresponds to the number of possible classes (of size m-1) plus the last column that indicates if the instance is predicted to experience the event of interest (e.g, if the i^{th} sample is expected to experience an event, a perfectly correct prediction of the i^{th} value of the last column would be 1, or at least not far from 1). Let $y \in \{1, m\}^n$ denote the index vector of the true labels, i.e. $y_i = j$ means that the i^{th} sample has the j^{th} label. Then Top$_A$-k is defined as follows:

$$\text{Top}_A\text{-k} = \frac{1}{n} \sum_{i=0}^{n-1} \sum_{j=1}^{k} \mathbb{1} \left(\underset{i \in [m] \setminus S_j^i}{\text{argmax}} \ \hat{y}_{i,:} = y_i \right)$$

$$S_j^i = \begin{cases} \emptyset & \text{if j = 1} \\ \underset{i \in [m] \setminus S_{j-1}^i}{\text{argmax}} \ \hat{y}_{i,:} & \text{otherwise} \end{cases}$$

where $\hat{y}_{i,:}$ is the i^{th} row of $\hat{y}$ corresponding to the joint probability distribution of the i^{th} sample and $\mathbb{1}$ is the indicator function. Top$_A$-1 is the accuracy where true class matches with the most probable predicted classes, which is the same as a standard accuracy.

Score of Most-at-Risk Samples Detection under Budget Constraint Let $B \in \mathbb{N}^*$ the budget devoted to maintenance interventions. In this study, we make the following assumptions:

1. For the first scenario, a budget B allows B maintenance interventions, which means that the cost of intervention is assumed to be 1.

2. In the second scenario, all the failure causes have the same intervention cost which is 1.

3. Each year a budget B is allocated for maintenance in the first scenario, and a budget of $5B$ fairly shared over the five failure causes for the second scenario.

The idea here is to classify the low-voltage feeders, from the set to evaluate, through decreasing order of predicted failure probability, then take the first B most-at-risk feeders according to the predictions (since only B maintenance interventions are possible given the budget) and count those which have actually experienced the power failure. This allows us to have the information of the percentage of failures that we would have avoided over a year if we

[6] See details in: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.top_k_accuracy_score.html

Table 6.3: Second scenario result: C^{td} scores calculated for each potential cause using the four models considered in this study.

Models	exc_elec_cap	dereg_driv	nor_we_te	hdwa_fail	insuf_prun
FCN	0.608	0.616	0.604	0.629	0.665
DeepHit	**0.631**	0.631	0.612	**0.648**	0.67
DeepWeiSurv	0.624	0.631	616	0.642	0.672
DPWTE	0.626	**0.633**	**0.617**	0.645	**0.678**

have spent a budget B for maintenance interventions on the feeders that are the most likely to experience the power failure.

Let $\hat{y} \in [0,1]^{n\times m}$ is the prediction matrix on the evaluation set of size n. For the second scenario, m corresponds to the number of potential causes (of size m-1) plus the predict event indicator column. For the first scenario, $m = 1$. Let $y \in \{0,1\}^{n\times m}$ denote the matrix of true labels. Then, the score of the first B most-at-risk feeders, denoted by Top$_R$-B, is defined as follows:

$$\text{Top}_R\text{-B} = \frac{1}{B}\sum_{i=1}^{B} \mathbb{1}\left(y_{\sigma_j(i),j} = 1\right), \text{ j=1,..,m} \tag{6.3.1}$$

where σ_j is the permutation corresponding to the decreasing order of $\hat{y}_{:,j}$ the j^{th} column of $\hat{y}$, that is, $\hat{y}_{\sigma_j(1),j} > \hat{y}_{\sigma_j(2),j} > .. > \hat{y}_{\sigma_j(n),j}$. Clearly, the Top$_R$-B takes into account the budget B which represent a real-world constraint. A perfect score of Top$_R$-B is therefore 1 corresponding to predicting B most-at-risk feeders, that is, using 100% of the budget for B most-at-risk feeders.

6.3.2.4 Evaluation Strategy

We recall that we have at our disposal all the data recorded during the period 2011-2016. However, we cannot use the data recorded in 2016 because we do not have a failure history of the following year namely 2017. We choose therefore to use the combined data recorded between 2011 and 2014 and history recorded between 2012 and 2015 as the training, while the combined data and history recorded in 2015 and 2016 respectively, will be used for evaluation. To have an idea about the size of the divided data: the training data contains 175000 observations whose 1700 are non-censored regardless of cause, whereas the evaluation set contains 44000 instances with 320 observed failures. For both scenarios, we train the competing models via Adam optimizer with a learning rate of 10^{-4}. After the training phase, we calculate the concordance index score C^{td} for all the models in both scenarios, and Top$_A$-k accuracy only for the multi-class problem. We also calculate Top$_R$-B for different values of $B = [20, 30, 40, 50, 2500]$ for both all-cause and competing risks scenarios.

6.3.2.5 Results and Discussion

The concordance index scores calculated in the first scenario as well as the Top$_A$-k scores ($k = 1, 2$) calculated in the second scenario are displayed in Table 6.4, whereas the concordance index scores for each cause under study (exceeding electrical capacities, deregulated drivers, normal wear and tear, hardware failure, and insufficient pruning) are calculated using the four models as shown in Table 6.3. For the budget-based score, Figure 6.6 and Figure 6.5 show, respectively, the Top$_A$-B scores calculated in the first and the second scenarios.

Table 6.4: Top$_A$-2 and Top$_A$-1 calculated for the second scenario, and C^{td} scores calculated for the first scenario, using the four models considered in this study.

Models	Top$_A$-2	Top$_A$-1	C^{td}
FCN	0.689	0.644	0.675
DeepHit	**0.717**	0.658	0.699
DeepWeiSurv	0.712	0.651	0.704
DPWTE	0.715	**0.661**	**0.709**

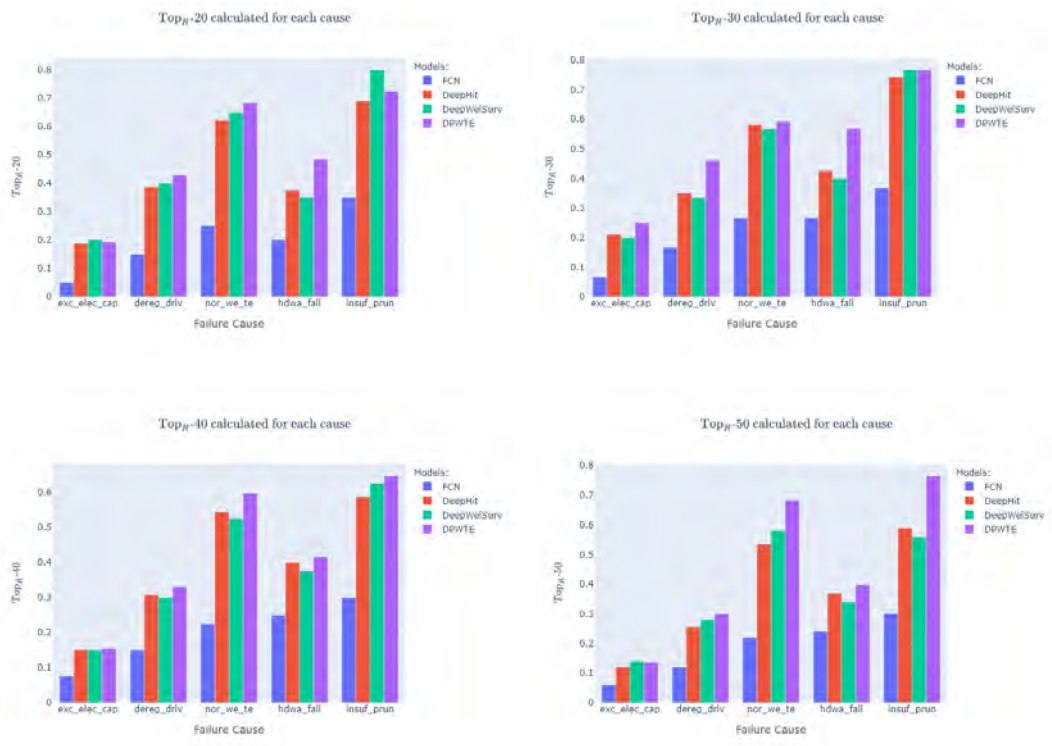


Figure 6.5: Second scenario result: Top$_R$-B scores calculated, for each cause, for different values of B using the four models considered

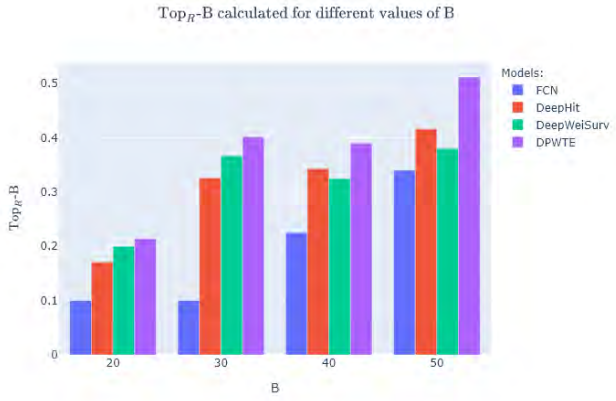


Figure 6.6: First scenario result: Top$_R$-B scores calculated for different values of B using the four models considered.

Classification Results From Table 6.4, we can notice that DeepHit very slightly outperforms, in terms of Top$_A$-2, DPWTE and DeepWeiSurv but outperforms FCN with a statistically significant difference. Whereas, in terms of Top$_A$-1, DPWTE outperforms the other models, with a slight improvement over DeepHit. This means DPWTE and DeepHit have the highest performance in terms of multi-class classification in this experiment. Since, as we remark in this table, Top$_A$-2 scores are greater than Top$_A$-1, this means that the models learned useful latent information from the combined data. However, the performances of the models are still low, this may be due to the fact that they (except FCN which is a classical classifier model) are described as survival methods and thus have focused on interpretability, potentially at some cost of predictive accuracy. In addition, FCN is under-performing even if it is described as a classifier whose goal is the maximization of the accuracy by minimizing the categorical cross-entropy in order to approximate the underlying event time distribution. Concerning the C^{td} scores calculated for the first scenario as shown in the same table, DPWTE outperforms the other models with a slight difference compared to DeepWeiSurv and DeepHit.

In the second scenario, DPWTE and DeepHit globally outperform DeepWeiSurv (slight difference) and FCN (with a statistically significant difference). The models have the lowest performance predicting the 'normal wear and tear' cause (nor_we_te) compared to other causes, while for the insufficient pruning cause, the models have the highest performance. This may be due to the extra data on the pruned zones that we merged with the combined data. Still, the scores are globally low (compared to the scores calculated in the experiment on the benchmark datasets, in Chapter 4). We suspect these predictive performances, measured by C^{td} and Top$_A$-k, come from the lack of some important characteristics and data at the low-voltage feeder level notably the electrical power, meteorological data as well as the modifications are done at the feeder level instead of aggregate data on the station level. We also suspect this underperformance is due to the short history (in terms of time duration) and the period where the data and high-voltage station modifications are recorded as well as the history size, that, is the number of observed events which is relatively small compared to the size of the electrical network under study.

Results of Most-At-Risk Samples Detection under Budget Constraint Regarding the Top_R-B that measures the portion of the most-at-risk feeders priority detected, DPWTE outperforms the other models in the first scenario with an increasing average score differential with respect to B. For instance, for $B = 10$, DPWTE and DeepWeiSurv outperform DeepHit and even further FCN hitting practically the same score, whereas, for $B = 30$, 40, and 50, DPWTE is outperforming the other models. For the latter, Top_R-50 = 0.52, which means that using a budget of 50, one could save 52%, that is, roughly 25 to 26 most-at-risk feeders that would have experienced a power failure in the coming year. We can notice that with $B = 30$, DPWTE doubled his score compared to $B = 20$ with 40% against 20%, that is, moving from $B = 20$ to $B = 30$ provides a ratio of 2. Whereas, $B = 40$ provides a slope a ratio of 1 compared to $B = 20$ ($\frac{0.4-0.2}{40-20} \times 100 = 1$). In the same way, $B = 50$ multiplied the score of $B = 20$ by $\frac{5}{2}$, providing a ratio of 1. This shows that the Top_R-B score is not linear as formulated in Equation (6.3.1).

In the second scenario, FCN is still under-performing the other models, while DPWTE is slightly outperforming in practically all the cases (except in the case of predicting 'insufficient pruning' when $B = 20$, where DeepWeiSurv outperforms). We can also notice that the scores not only vary from a value of B to another, but also between the five potential causes. We suspect this variation comes from the distribution of events (per cause) recorded in the training period. For instance, 'insufficient pruning' and 'normal wear and tear' are the two causes that have the highest scores for all values of B, and this might be due to the number of occurrences of these causes in the history and/or the extra file that stores all the pruning information (GDO code, geographical location, length of the pruned limbs, etc.) for the 'insufficient pruning' cause. Whereas, 'exceeding electrical capacities' cause has the lowest Top_R-B scores. This under-performance, aside from the model imperfections and the short training period, may come from the eventual lack of information about the total low-voltage (or at least high-voltage) electrical power, and the electrical voltage at different levels of the electrical structure (LV feeder, HV/LV transformer sub-station and HV station). Yet, this cause can be considered as the most feasible to oversee thanks to the corresponding sensors, and then monitor the electrical power and voltage for example. In this experiment, we can summarize the results as follows:

- The three survival methods namely: DeepHit, DeepWeiSurv, and DPWTE outperform the classifier FCN in terms of most-at-risk detection, time ordering, and even classification accuracy. We expect this comes from the fact that FCN is ignoring the censoring phenomenon which severely penalizes its performance since we are dealing with a highly censoring setting. Hence, the survival methodology is the most suitable solution to this problem.

- DPWTE outperforms, in most cases, the other considered models.

- Some causes are easier to detect than others. The reasons behind this phenomenon may be multi-fold (they only represent some hypotheses): the single-event distributions in the training history are different, the lack of information about some important characteristics for some causes (e.g. electrical power and voltage for 'exceeding electrical capacities' and 'hardware failure' causes), some sensor information for 'deregulated drivers', etc.

- The C^{td}, Top_A-k and Top_R-B scores are relatively low in all cases and for all models. This might be due to, in addition to the reasons cited above, the relative rarity of incidents which induces a huge ratio of censored events which, makes the task more challenging. Another complementary explanation to this under-performance would be:

the variables or attributes stores in the provided are not sufficiently relevant or simply not sufficient to distinguish between the most-at-risk feeders and "safe" ones. This can be supported by the analysis that we performed above, illustrated in Figure 6.4. And last but not least, the data provided describes only the state of the stations/feeders of a part of the region under study, while it is more interesting to collect data of other areas in this region even for local predictions.

Now after having described this survival study and discussed its results, we will consider the next stage namely the maintenance strategy. We will propose, using our methodology, given a budget for a given feeder, the optimal modifications to perform in a maintenance intervention in the feeder of interest, in order to maximize its mean lifetime.

6.4 Low-Voltage Feeder Mean-Lifetime Optimization under Budget Constraint

In the previous section, we conducted a survival analysis study on low-voltage feeders in the region of Occitania. Now in this section, we will see, by applying the methodology described in Chapter 5 what are the optimal modification given a budget, that it could be done for a particular feeder of interest in order to maximize its mean lifetime. Clearly and logically, we will be interested in the most-at-risk feeders detected using the models evaluated in the study described in the previous section. In the previous section, the fully connected network FCN showed a significant underperformance compared to DeepHit DeepWeiSurv and DPWTE. In addition, contrary to DeepWeiSurv and DPWTE, DeepHit is described as a discrete-time model, which means that we cannot use it as a continuous regression function to calculate the survival function at any given point in time. Therefore, we only test our two approaches in this experiment. As these network-based models are described as full-white boxes, we have the possibility to make them robust against numerical instability via the ℓ_1 gradient regularization technique described in Chapter 5. To do this, we need to re-train these networks but this time with their respective adversarial losses associated (called adversarial training, since we train the network to be robust against adversarial examples).This experiment is organized as follows: we first briefly describe the problem statement (Section 6.4.1). Then, we describe the calculation steps of the respective adversarial losses of each model (Section 6.4.2). Finally, we describe the experimental protocol and discuss the results (Section 6.4.3).

6.4.1 Problem Statement

We note that, in this experiment, we use the considered models to estimate the survival function of a feeder at a given point in time t. We also highlight that, as we are limited by the lack of some potential variables that could affect failure probabilities of some cause, we are interested only in the minimization of the failure probability, and thus the maximization of the survival function, in a general way, i.e. regardless of the cause behind. For simplicity we formulate the problem at the individual level, that is we consider the minimization failure probability problem for a most-at-risk low-voltage feeder of particular interest. Let $S^p_{dws}, S^{\tilde{p}}_{dpwte}$ denote the survival functions learned by DeepWeiSurv (of parameter p) and DPWTE (with $\tilde{p}$ is the size of the optimal combination of Weibull distribution) respectively. We also note $\mathcal{C}$ the set of controllable features of the feeder data, $\mathbf{c}$ the cost factor vector associated to the features (where for non-controllable ones, the respective cost factors are defined as indefinitely large so their values do not change), and B the individual budget

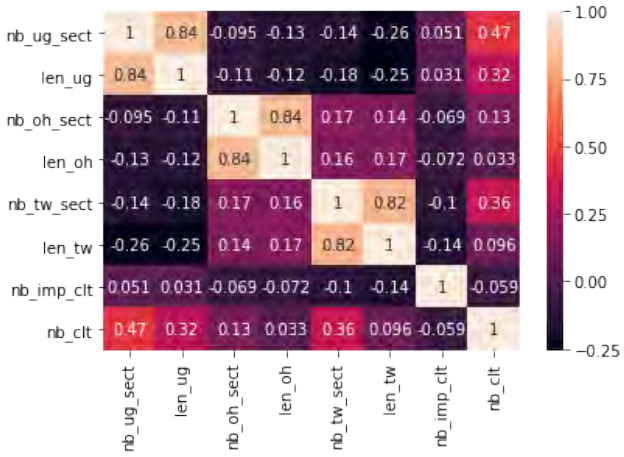


Figure 6.7: Correlation between feeder variable (except age variable).

dedicated for maintenance. Under the assumptions stated in Section 5.5 of Chapter 5, the problem that we consider here, for a particular feeder $\mathbf{x}$ of interest, can be formulated as follows:

$$\begin{aligned} & \arg\max_{\mathbf{y}\in\mathbb{R}^d} S(t,\mathbf{y}) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{C}} c_i|y_i - x_i| \leq b_0 \\ \text{s.t.} \quad & a_i \leq y_i \leq b_i \;\; \forall\, i \in \mathcal{C} \quad \text{and } y_i = x_i \;\; \forall\, i \notin \mathcal{C} \end{aligned} \tag{6.4.1}$$

where $S \in \{S^p_{dws}, S^{\tilde{p}}_{dpwte}\}$ is a survival function learned by DeepWeiSurv or DPWTE and d is the number of features of feeder data. The solution to Problem 6.4.1, denoted by $\mathbf{y}^*$ lower and upper bound by $\mathbf{a} = (a_i)$, $\mathbf{b} = (b_i)$ respectively, is nothing but the optimal modification of the given feeder (in terms of controllable feature values) that best minimizes the failure risk under the defined budget constraint. We recall that the feeder data have the features described in Table 6.1. Clearly, the values of the 'age' variables cannot be modified. Furthermore,we assume that removing a an important client from a feeder is so costly that it is not feasible. We therefore assume that important client variable is not controllable. We also choose to not let our methodology operate on the section total length variables for the sheer fact that there exists a highly strong positive correlation (of 84%, see Figure 6.7) between the number of section variable and the total length variable of the same type (e.g. the correlation between the 'number of overhead sections' variable denoted by *nb_oh_sect* and the 'total length of overhead section' variable denoted by *len_oh* is equal to 0.84), and more precisely increasing the number of sections of a given type implies the increasing the total length of the sections of the same type. These assumptions and choices made leave us with a resulting controllable features set ($\mathcal{C}$) on which our methodology can operate which is consisted of the following variables: the number of current clients as well as the number of overhead, underground and twisted sections variables, i.e $\mathcal{C} = \{nb\_clt, nb\_oh\_sect, nb\_ug\_sect, nb\_tw\_sect\}$. To solve the optimization problem described in (6.4.1), we first robustify both DeepWeiSurv and DPWTE networks against adversarial inputs, i.e. numerical instability through ℓ_1-norm gradient regularization.

6.4.2 Robustifying DeepWeiSurv and DPWTE Against Adversarial Inputs

Let f^p_{dws} and $f^{p\to\tilde{p}}_{dpwte}$ denote respectively the function learned by DeepWeiSurv of parameter p and the function learned by DPWTE as trained previously on the feeder data, where $\tilde{p}$ is the mixture size estimate initially set to an upper bound p. We also note ω the weights of the Mixture Weibull Sparse (MWS) layer of DPWTE. Then, using the notation established in Chapter 2 and Chapter 3, f^p_{dws} and $f^{p\to\tilde{p}}_{dpwte}$ are solutions of the following loss minimization problems

$$f^p_{dws} = \underset{\beta\geq 1,\eta>0,\alpha\geq 0,\|\alpha\|_1=1}{\operatorname{argmin}} \mathcal{L}^p_{dws} = -log\ \mathcal{L}(\beta,\eta,\alpha|\mathbf{X}), \tag{6.4.2}$$

$$f^{p\to\tilde{p}}_{dpwte} = \underset{\beta\geq 1,\eta>0,\alpha\geq 0,\|\alpha\|_1=1}{\operatorname{argmin}} \mathcal{L}^p_{dpwte} = -log\ \mathcal{L}(\beta,\eta,\alpha|\mathbf{X}) + \lambda||\omega||_{\frac{1}{2}}. \tag{6.4.3}$$

where $\mathbf{X} = (\mathbf{x}^1, .., \mathbf{x}^N)$ is the training data and λ is the regularization parameter to penalize MWS weights. These two networks which respectively minimize Equation (6.4.2) and Equation (6.4.3), may be vulnerable against some adversarial inputs generated by PGD algorithm. Thus, we propose a robust version of them using ℓ_1 gradient regularization, resulting to two functions that we denote $f^p_{dws+\sigma}$, $f^{p\to\tilde{p}}_{dpwte+\sigma}$ defined as follows:

$$f^p_{dws+\sigma} = \operatorname{argmin}\ \mathcal{L}^p_{dws+\sigma} = \mathcal{L}^p_{dws} + \sigma\left\|\nabla_{\mathbf{X}}\mathcal{L}^p_{dws}\right\|_1 \tag{6.4.4}$$

$$f^{p\to\tilde{p}}_{dpwte+\sigma} = \operatorname{argmin}\ \mathcal{L}^p_{dpwte+\sigma} = \mathcal{L}^p_{dpwte} + \sigma\left\|\nabla_{\mathbf{X}}\mathcal{L}^p_{dpwte}\right\|_1 \tag{6.4.5}$$

where σ is the strength parameter. After the learning phase (and the post-training steps for DPWTE), we construct the respective survival functions, that we respectively denote $S^p_{dws+\sigma}$ and $S^{\tilde{p}}_{dpwte+\sigma}$, of the robust versions of DeepWeiSurv and DPWTE from the learned Weibull parameters in $f^p_{dws+\sigma}$ and $f^{p\to\tilde{p}}_{dpwte+\sigma}$ respectively. In this experiment, we use the following (robust) objective function $S(t,.)$, where $S \in \{S^p_{dws+\sigma}, S^{\tilde{p}}_{dpwte+\sigma}\}$ instead of $\{S^p_{dws}, S^{\tilde{p}}_{dpwte}\}$.

6.4.3 Experiment : Feeder-Level Survival Function Maximization under Budget Constraint using Robustified DeepWeiSurv and DPWTE

We recall that in order to optimize the survival function of a given low-voltage feeder which potentially can experience a failure event, we need to do some modifications by operating on its intrinsic features as well as the number of clients that use the feeder of interest. Furthermore, we point out that in this experiment, for the sake of confidentiality, we hide the identity of the feeder. In the same way, as in Section 6.3.2, we note that the values of the cost vector $\mathbf{c}$ do not reflect reality, but for the sake of consistency, we make the following assumptions:

- A modification at the intrinsic feature level has the same cost (that we set at 1) regardless of the type of section (underground, overhead or twisted).
- A modification of the number of clients is more expensive for the simple reason that an obligation to decrease the number of clients from a most-at-risk feeder may imply other modifications such as the change of the electrical capacity or other electrical features (whose details are not at our disposal), or construction of a new feeder to host each part of clients from most-at-risk feeders, etc.

Still, these interpretations rely on assumptions that may not sufficiently reflect the reality on the ground, but the results are expected to be at least consistent and open to interpretation since the performance of the models are acceptable. We choose therefore regardless of the feeder under study, the following configurations:

- We set the respective cost factors for the intrinsic features at 1, that of the number of current clients at 5:

$$c_i = \begin{cases} 1 \text{ if } i \in \{nb\_oh\_sect, nb\_ug\_sect, nb\_tw\_sect\} \\ 5 \text{ if } i = nb\_clt \\ c' >> b_0 \text{ otherwise} \end{cases}$$

- We set the budget $b_0 = 50$.
- Another way to freeze the non-controllable features is to set: $b_i = a_i = x_i,\ \forall i \notin \mathcal{C}$, where $\mathbf{x} = (x_i)$ is the features values of the feeder of interest and $\mathcal{C}$ is the set of the controllable features.
- For controllable features, we set $a_i = 0,\ \forall i \in \mathcal{C}$, since the respective range values of the concerned variables necessarily belong to $\mathbb{R}^+$, and we set $b_i = 50$ for intrinsic features and $b_i = 90$ for the number of current clients variable.

6.4.3.1 Experimental Scenario

To train the robust versions of DeepWeiSurv and DPWTE, represented respectively by the functions $f^p_{dws+\sigma}$ and $f^{p\to\tilde{p}}_{dpwte+\sigma}$, we apply the same protocol as in Section 6.3.2.4. In this experiment, we test different values of $\sigma = 0.5, 1, 2.5$ (other values of σ are tested but they are not included in the results of this experiment for the sheer fact that they provide similar performances). Once these two groups of networks are trained, we numerically solve the problem defined in Equation (6.4.1) for given a most-at-risk feeder $\mathbf{x}$ from the test set (year 2016). To calculate the optimal modifications in the feeder of features $\mathbf{x}$, we apply, using the configurations fixed above, the projected gradient descent method defined in Algorithm 4 (Chapter 5) using a box-constrained and weighted ℓ_1-ball projection as defined in Algorithm 3 (Chapter 5) with $\mathbf{a}$ and $\mathbf{b}$ the lower and upper bounds respectively. As results, we obtain for each value of σ both the optimal modifications as well as the optimal survival value associated $S(t,.)$ provided by $f^p_{dws+\sigma}$ and $f^{p\to\tilde{p}}_{dpwte+\sigma}$. The time t (in days) for which we calculate the survival function output corresponds to the day in which the feeder of interest experienced the failure in 2016 (as indicated in the history data), which means that maximizing $S(t,.)$ by operating on the feeder covariates gives us an idea about how long we could have extended the life span of the feeder. To summarize the protocol that we describe above, let's briefly describe the steps applied to find the optimal modification in a feeder of covariates $\mathbf{x}$:

1. Train the networks $f^p_{dws+\sigma}$ and $f^{p\to\tilde{p}}_{dpwte+\sigma}$, for each selected value of σ, using the experimental protocol described in 6.3.2.4.
2. Apply the PGD algorithm using a box-constrained projection to calculate the optimal modifications $\hat{\mathbf{x}}$ and the survival function value $S(t, \hat{\mathbf{x}}), S \in \{S^p_{dws+\sigma}, S^{\tilde{p}}_{dpwte+\sigma}\}$ for each network and for each value of σ.

As mentioned above, we test feeders that experienced a failure in the last year of the study period namely: 2016. We then see what are the modifications that needed to be made at the feeder level and which could have maximized the survival function and thus prolonged its mean lifetime. We choose for this experiment two feeders from this group:

Figure 6.8: The mean global variable importance calculated over all the models considered in this experiment.

- The first feeder, of covariates $\mathbf{y}$, is totally underground with the following feature values: $nb\_oh\_sect = 0$, $nb\_ug\_sect = 15$, $nb\_tw\_sect = 15$ and $nb\_clt = 82$.
- The second one, of covariates $\mathbf{z}$, is practically overhead with $nb\_oh\_sect = 11$ overhead sections, one underground and one twisted sections ($nb\_ug\_sect = 1$ and $nb\_tw\_sect = 1$) feeding 44 current clients ($nb\_clt = 44$).

We will apply the experimental setting described in this section on these two feeders. Then, we analyze and discuss the different solutions provided by the models considered.

6.4.3.2 Results and Discussion

Before discussing and analyzing the results of this experiment, we note that using each model, we calculated the global importance of all the features, or in other words, we quantified the (negative or positive) contribution, of each feature, to the survival prediction. We then average the global importance of these features over the models considered here and obtain the mean global importance of all the variables (used to train the models as shown in Figure 6.8). As we see in this figure, only four variables (2 of which are controllable namely the number of underground sections nb_ug_sect and the twisted sections nb_tw_sect) have positive contributions to the prediction of survival. This means that increasing the values of these features can only increase the survival value, i.e. extend the lifetime of the feeder of interest. Whereas, 7 out of 11 variables have a negative contribution, three of which are controllable, namely the number of the current client (nb_clt) as well as the number (nb_oh_sect) and the length (len_oh_sect) of the overhead sections (the last feature is not taken into account since there is a significant correlation between it and nb_oh_sect), and whose values should be decreased to extend the lifespan. We assess this statement in the results presented in the following discussion.

The results of the experiment applied on both feeders described above are displayed in Table 6.5 and Table 6.6. For both tables, each row corresponds to the respective values of the controllable features of the feeder of interest. The last row in each table represents the current controllable-feature values taken by the feeder before an eventual maintenance.

Table 6.5: Propositions of modifications (raw solutions) on the feeder $\mathbf{y}$ provided by the robust versions of DeepWeiSurv and DPWTE with $\sigma = 0.5$, 1 and 2.5.

	nb_oh_sect	*nb_ug_sect*	*nb_tw_sect*	*nb_clt*
$\hat{\mathbf{y}}^{dws}_{\sigma=0.5}$	0	22.91	21.59	74.9
$\hat{\mathbf{y}}^{dpwte}_{\sigma=0.5}$	0	19.83	20.17	74
$\hat{\mathbf{y}}^{dws}_{\sigma=1}$	0	41.37	38.63	82
$\hat{\mathbf{y}}^{dpwte}_{\sigma=1}$	0	46.1	33.9	82
$\hat{\mathbf{y}}^{dws}_{\sigma=2.5}$	0	39.02	26.68	79.14
$\hat{\mathbf{y}}^{dpwte}_{\sigma=2.5}$	0	34.24	24.01	77.65
$\mathbf{y}$	0	15	15	82

Table 6.6: Propositions of modifications (raw solutions) on the feeder $\mathbf{z}$ provided by the robust versions of DeepWeiSurv and DPWTE with $\sigma = 0.5$, 1 and 2.5.

	nb_oh_sect	*nb_ug_sect*	*nb_tw_sect*	*nb_clt*
$\hat{\mathbf{z}}^{dws}_{\sigma=0.5}$	0	30.09	10.91	44
$\hat{\mathbf{z}}^{dpwte}_{\sigma=0.5}$	1.02	24.09	8.98	42.21
$\hat{\mathbf{z}}^{dws}_{\sigma=1}$	2.58	13.46	10.12	40
$\hat{\mathbf{z}}^{dpwte}_{\sigma=1}$	0	20.78	20.22	44
$\hat{\mathbf{z}}^{dws}_{\sigma=2.5}$	0.38	13.57	9.01	40.24
$\hat{\mathbf{z}}^{dpwte}_{\sigma=2.5}$	0.37	15.7	11.42	41.15
$\mathbf{z}$	11	1	1	44

For instance, $\hat{\mathbf{y}}^{dws}_{\sigma=0.5}$ is a solution to Problem 6.4.1 for the feeder $\mathbf{y}$ provided by the robust version of DeepWeiSurv of $\sigma = 0.5$. All the results meet the budget constraint namely: $\|\mathbf{c} \odot (\mathbf{x} - \hat{\mathbf{y}})\|_1 \leq b_0$. In those instances, during PGD calculations, the iterate was projected onto the $\mathbf{c}$-weighted ℓ_1 sphere of center $\mathbf{x}$ and radius b_0, which means that all the budget was consumed to calculate the numerical (raw) modifications, e.g., for the feeder $\mathbf{y}$, the solution $\hat{\mathbf{y}}^{dws}_{\sigma=0.5}$ meet the budget constraint: $1\times(22.91-15)+1\times(21.59-15)+5\times(82-74.9) = 50 = b_0$.

However, as we notice in these two tables, the feature values of the solutions are float numbers (raw format of the solutions as calculated in PGD algorithm), while the features from the controllable set $\mathcal{C}$ only take integer values. We, therefore, in the first instance, round the feature values to get the final solutions. We choose to round the solutions to the nearest integer below their respective current values (*floor* denoted by $\lfloor . \rfloor$) for the features with positive contributions and above their respective values (*ceil* denoted by $\lceil . \rceil$) for the negative-contribution features, for the simple reason that otherwise, we could exceed the amount in the budget especially if the cost of the feature concerned is greater than 1. This first post-process results in some budget residual b_{res}, which can then be shared between all the feasible modifications or devoted to extra modification on the most important variables (with respect to the respective cost factors).

To illustrate this, if we take the modifications on the feeder $\mathbf{z}$ provided by DPWTE's robust version of $\sigma = 0.5$, we can propose to operate the following modifications :

- Reduce the number of the overhead section to 2 ($11 \rightarrow \lceil 1.02 \rceil = 2$).
- Add 23 underground section ($1 \rightarrow \lfloor 24.09 \rfloor) = 24$).

Table 6.7: Final solutions (in bold) for the feeder **y** provided by the robust versions of DeepWeiSurv and DPWTE with $\sigma = 0.5$, 1 and 2.5.

	nb_oh_sect	*nb_ug_sect*	*nb_tw_sect*	*nb_clt*	*budget residual*	*survival*
$\hat{\mathbf{y}}^{dws}_{\sigma=0.5}$	0	22.91	21.59	74.9	0	-
	0	22	21	75	2	-
	0	**23**	**22**	**75**	**0**	**0.63**
$\hat{\mathbf{y}}^{dpwte}_{\sigma=0.5}$	0	19.83	20.17	74	0	-
	0	19	20	74	1	-
	0	**20**	**20**	**74**	**0**	**0.59**
$\hat{\mathbf{y}}^{dws}_{\sigma=1}$	0	41.37	38.63	82	0	-
	0	41	38	82	1	-
	0	**42**	**38**	**82**	**0**	**0.68**
$\hat{\mathbf{y}}^{dpwte}_{\sigma=1}$	0	46.1	33.9	82	0	-
	0	46	33	82	1	-
	0	**47**	**33**	**82**	**0**	**0.66**
$\hat{\mathbf{y}}^{dws}_{\sigma=2.5}$	0	39.02	26.68	79.14	0	-
	0	39	26	80	5	-
	0	**42**	**28**	**80**	**0**	**0.63**
$\hat{\mathbf{y}}^{dpwte}_{\sigma=2.5}$	0	34.24	24.01	77.65	0	-
	0	34	24	78	2	-
	0	**36**	**24**	**78**	**0**	**0.65**
y	0	15	15	82 0	-	0.36

Table 6.8: Final solutions (in bold) for the feeder **z** provided by the robust versions of DeepWeiSurv and DPWTE with $\sigma = 0.5$, 1 and 2.5.

	nb_oh_sect	*nb_ug_sect*	*nb_tw_sect*	*nb_clt*	*budget residual*	*survival*
$\hat{\mathbf{z}}^{dws}_{\sigma=0.5}$	0	30.09	10.91	44	0	-
	0	30	10	44	1	-
	0	**31**	**10**	**44**	**0**	**0.67**
$\hat{\mathbf{z}}^{dpwte}_{\sigma=0.5}$	1.02	24.09	8.98	42.21	0	-
	2	24	8	43	6	-
	0	**26**	**10**	**43**	**0**	**0.64**
$\hat{\mathbf{z}}^{dws}_{\sigma=1}$	2.58	13.46	10.12	40	0	-
	3	13	10	40	1	-
	2	**13**	**10**	**40**	**0**	**0.58**
$\hat{\mathbf{z}}^{dpwte}_{\sigma=1}$	0	20.78	20.22	44	0	-
	0	20	20	44	1	-
	0	**21**	**20**	**44**	**0**	**0.69**
$\hat{\mathbf{z}}^{dws}_{\sigma=2.5}$	0.38	13.57	9.01	40.24	0	-
	1	13	9	41	5	-
	0	**15**	**11**	**41**	**0**	**0.62**
$\hat{\mathbf{z}}^{dpwte}_{\sigma=2.5}$	0.37	15.7	11.42	41.15	0	-
	1	15	11	42	6	-
	0	**18**	**13**	**42**	**0**	**0.65**
z	11	1	1	44	-	0.26

- Add 7 twisted sections ($1 \rightarrow \lfloor 8.98 \rfloor = 8$).
- Remove 1 client ($44 \rightarrow \lceil 42.21 \rceil = 43$)

These modifications will therefore cost $1 \times (11-2)+1 \times (24-1)+1 \times (8-1)+5 \times (44-43) = 44 = b_0 - b_{res}$, which means that $b_{res} = 6$ can be fairly shared between the three first features (of cost 1) and thus used to reduce *nb_oh_sect* to zero and add extra underground and twisted sections (2 each).

Once the final modifications are calculated, we estimate the survival value $S(t, \hat{\mathbf{x}}), S \in \{S^{p}_{dws+\sigma}, S^{\tilde{p}}_{dpwte+\sigma}\}$ for each solution $\hat{\mathbf{x}}$ with the corresponding model. We illustrate these steps and obtain the final solutions in Table 6.7 and Table 6.8 for both feeders $\mathbf{y}$ and $\mathbf{z}$ respectively. The first insight that we can extract from these two tables is that all the modifications are translated by a lifespan extension as we see in the column *'survival'* which stored the values of $S(t, .)$. In fact, through our methodology, both feeders $\mathbf{y}$ and $\mathbf{z}$ could multiply their respective survival value by a factor of 1.7 and 2.4 respectively (the final survival is equal to 0.64 on average for both of them while the initial values of survival are respectively 0.36 and 0.26 for $\mathbf{y}$ and $\mathbf{z}$. This means that, by applying the proposed modifications, both feeders have up to 64% chance to survive in the coming year). For each feeder, the survival values predicted by the models using their respective solutions induce a tight standard deviation which can explain the numerical stability of the robust versions. The second thing to notice, if we look at what happens on a feature scale, is that all the models tend to decrease the values of both *nb_oh_sect* and *nb_clt* features, whereas they increase *nb_ug_sect* and *nb_tw_sect*. This fits with the fact that both the number of overhead sections and the number of clients have a negative impact on the lifespan, while the number of underground and twisted sections have positive contributions on the feeder survival as indicated by the global importance of the controllable features represented in Figure 6.8. This result is in line with what was stated and operated by ENEDIS in the previous section and was expected, in the case of the use of reliable models, since as we have seen in Section 6.3.1.2, the only possible modifications done during the study period are: reducing the number of overhead sections, increasing the number of underground sections or increasing the number of twisted sections.

6.5 Conclusion

In this chapter, we have seen concrete, real-world and complete application of the main problems developed in this thesis namely the project on which we worked with ENEDIS. The aim of this project during these two years was to use different nature of data at their disposal to implement an algorithm, using a machine learning framework whose role is to predict the failure on an individual scale. In the first instance, we were interested in only predicting the failure regardless of the cause behind it. Then, we extend the problem to a multi-class prediction where the nature of the risk is also a subject of study. We started by analyzing the data provided, including the failure history data, and describing the main pre-processing steps that result in clean pre-processed data. After that, we briefly described the main experiment that we conducted during the project, where we described the network-based models including our two approaches DeepWeiSurv and DPWTE used for failure prediction and the evaluation metrics which we consider relevant to evaluate these models. We also defined the experimental settings, presented and discussed the result of our two approaches as well as the competitive methods which globally were outperformed by our models in terms of classification, most-at-risk feeders detection, and the most likely failure ranking. The overall performance is still subject to optimization in the future by feeding

further the database with more information notably new relevant variables (e.g., consumed power, weather quantitative information, etc.), data about feeders from other zones of the region, and the failure history associated to strengthen the models against biases. Finally, we completed the study (not performed during these two years on which we worked with ENEDIS) by proposing a different solution, for a given most-at-risk feeder, to increase or extend its lifespan. For this purpose, we used our optimization-based methodology presented in Chapter 5, combined by ℓ_1-regularized versions of DeepWeiSurv and DPWTE to get around the numerical instability induced by the PGD algorithm. We conducted a simulation experiment on two anonymous feeders that experienced a failure (regardless of the cause behind for the sake of simplicity) and discussed the modifications proposed by our models that could have been operated at the controllable feature level to increase the survival value at the time of the experienced failure and thus to extend their respective lifespans. As a result, both groups of models have proposed, for a given feeder and subject to budget constraint, to reduce both the number of overhead sections and the number of the current clients directly linked to the feeder of interest as well as to increase both the number of underground and twisted sections. This result is almost perfectly in line with the strategy of ENEDIS performed at the high-voltage station level to reduce the failure risks.

Conclusion

In this thesis, we first investigated the problem considered in survival analysis namely, given survival data, the quantification of the risk of experiencing an event of interest at the individual scale while taking into account censoring in the data. We proposed, to tackle this problem, two deep learning approaches called DeepWeiSurv (Chapter 2) and DPWTE (Chapter 3) respectively, that assume that the underlying event time distribution can be modeled by a finite mixture of Weibull distributions whose parameters are estimated by a neural network maximizing the likelihood of this mixture. We empirically showed, using simulated tabular datasets, that both models learn the respective parameters as well as the weighting coefficients of the Weibull distributions composing the mixture. We also conducted experiments on time-series datasets and real-world datasets, testing several methods (our two models and the most known state-of-the-art methods described in Chapter 1) and showed that our approaches (especially the so-called DPWTE) are globally outperforming the competing methods at different degrees. Apart from the predictive performance, another advantage of these two models over the competing ones, is that we can estimate the (individual) mean lifetime as well as the survival probability at any point of time.

The main difference between our two approaches is that DeepWeiSurv takes the size of the mixture denoted by p as a parameter which means that the performance of this method depends on the value of p, while DPWTE tackles this problem by only taking as input an upper bound, sufficiently large, of the size and find the optimal mixture (whose size is lower than the upper bound) of Weibull distributions to model the underlying distribution. We showed, through simulated experiments described in Chapter 3, that DPWTE finds the size of the mixture as well as the parameters and the weighting coefficients with which we generated the datasets given that the model takes an upper bound largely greater than the size of the generated mixture. In addition, we showed, through the experiments on the real-world data described in Chapter 4, that DPWTE has at least the same, if not better, performance as DeepWeiSurv. The approach behind these two models can also be applied on other statistical distributions (while adapting the loss function) notably the mixture of normal distributions.

After having tackled the problem of survival analysis, we proposed, in Chapter 5, a hypothesis-based approach to maximize the lifespan, i.e. minimize the risk of the event by operating on individual covariates under budget constraints. This approach consisted of solving the constrained optimization problem where the solution is the set of modifications to be operated on the concerned features. We considered, in our experiments, three scenarios for the objective function (that calculates the risk/lifespan) namely the black, the semi-, and the full-white boxes, where the first one is interpreted by LIME [39] and the last one is robustified using the gradient regularization technique to address the numerical instability. We empirically showed that the robust version of the full-white box family outperforms compared to the first two scenarios. We also showed, through a simulation, that the performance

of the method depends on the radius of the weighted ball that models the budget/cost constraints. Since the proposed method, admittedly motivated by the predictive maintenance field, can be applied for any regression problem whose objective is to maximize the output under linear constraints as assumed in this work, we run an experiment on the wine dataset, where the goal was to maximize the quality of a certain wine given its physico-chemical characteristics, using the robust version of the full-white box and discuss the results. We noticed that the provided solutions make sense in reality.

However, in addition to the research directions explored in this work, we believe that there are still some perspectives that must be taken into account in the future and deserve further investigation. We recall that the approach presented in Chapter 5 is developed under the assumption that limits its scope of application. In reality, the cost linearity assumption is not always correct and thus the assumption of a non-linear relationship between the cost and the feature modification must be made resulting in non-linear budget/cost constraints. The same goes for the independent cost variables in which case, in reality, a modification of a variable may imply a change in another variable and thus induces additional costs that must be taken into account in the modeling. Regarding the box constraints, an interesting direction for future works would be to consider the case where the upper and lower bounds have opposite signs which may be possible for some variables such as the temperature, a force in physics, etc.

In the last chapter (Chapter 6), we described the study conducted within the SmartOccitania project supported by ENEDIS and ADEME. This study, considered as a complete application of our approaches developed during this thesis, aims to model the risk that a low-voltage feeder experiences a power failure. For this purpose, we used DPWTE and DeepWeiSurv as well as DeepHit considered as the best competing method according to the experiments described in Chapter 4 whose results are acceptable but need improvement and this is, in part, due to the quality and the quantity (limited space and short duration) of the provided baseline and history. Then, we applied the approach presented in Chapter 5 to propose optimizing solutions for the most-at-risk feeders.

Appendices

Appendix A

Proofs of Lemmas used to Calculate the Projection onto the Weighted ℓ_1 Ball

A.1 Box-Constraint-Free Projection

Lemma 4. *Let $b_0 > 0$, $\mathbf{w} \in (\mathbb{R}^{*+})^d$, $\mathbf{x}^0 \in \mathcal{X}$ and $\mathbf{x} \in \mathcal{X}$ be the vector to be projected. Let $\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ denote the simplex : $\{\mathbf{h} \in \mathcal{X} \mid \mathbf{w}^T(\mathbf{h} - \mathbf{x}^0) = b_0$ and $\mathbf{h} \geq \mathbf{x^0}\}$ and let $\mathbf{x}^s$ and $\mathbf{x}^b$ the solutions to Problem 5.4.3 and Problem 5.4.2 respectively. We recall that if $\mathbf{x} \in \Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ then $\mathbf{x}$ is the solution, i.e., $\mathbf{x}^s = \mathbf{x}$ and the same goes for when $\mathbf{x} \in \mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$. If $\mathbf{x}$ is outside $\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ then there exists a unique real λ^* such that:*

$$x_i^s = x_i^0 + \max\{x_i - (w_i\lambda^* + x_i^0), 0\},\ \forall i \in [d] \tag{A.1.1}$$

Using lemma 1, we thus have; when $\mathbf{x} \notin \mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$; the solution to Problem 5.4.2 defined as follows:

$$x_i^b = x_i^0 + sign(x_i - x_i^0)\ (x_i^s - x_i^0) \tag{A.1.2}$$

where sign is the signum function.

Proof. Let $b_0 > 0$, $\mathbf{w} \in (\mathbb{R}^{*+})^d$ $\mathbf{x}^0 \in \mathcal{X}$ and $\mathbf{x} \in \mathcal{X}$ be the vector to be projected. Let $\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ denote the simplex : $\{\mathbf{h} \in \mathcal{X} \mid \mathbf{w}^T(\mathbf{h} - \mathbf{x}^0) = b_0$ and $\mathbf{h} \geq \mathbf{x^0}\}$. We suppose that $\mathbf{x}$ is outside $\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$. The projection onto $\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}$ is formulated as:

$$\underset{\mathbf{y} \in \Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 \tag{A.1.3}$$

The Lagrangian of the problem above is:

$$\mathcal{L}(\mathbf{y}, \lambda, \mu) = \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 + \lambda(\mathbf{w}^T(\mathbf{y} - \mathbf{x}^0) - b_0) - \mu^T(\mathbf{y} - \mathbf{x^0})$$

where $\lambda \in \mathbb{R}$ is a Lagrangian multiplier and $\mu \in (\mathbb{R}^{*+})^d$ is a vector of non-negative Lagrange multipliers. Using Kuhn-Tucker [121] conditions, the optimum satisfies:

$$\begin{cases} \nabla_{\mathbf{y}}\mathcal{L} = \mathbf{y} - \mathbf{x} + \lambda\mathbf{w} - \mu = 0 \\ \mu(\mathbf{y} - \mathbf{x}^0) = \mathbf{0}. \end{cases} \tag{A.1.4}$$

By defining $\mathbf{y}$ and μ such as $\forall i \in [d]$:

$$y_i = x_i^0 + \max\{x_i - (w_i\lambda + x_i^0), 0\}, \tag{A.1.5}$$
$$\mu_i = \max\{w_i\lambda + x_i^0 - x_i, 0\}. \tag{A.1.6}$$

the two conditions defined in System (A.1.4) still respected. By definition, $\mathbf{y} \geq \mathbf{x}^0$. Let $g : \mathbb{R} \mapsto \mathbb{R}$ a function defined as follows:

$$g(\lambda) = \sum_{i\in[d]} w_i(y_i - x_i^0) \tag{A.1.7}$$
$$= \sum_{i\in[d]} w_i \max\{x_i - (w_i\lambda + x_i^0), 0\} \tag{A.1.8}$$

We aim at finding the value of λ^* such that $g(\lambda^*) = b_0$. From the definition, g is a positive part function and piece-wise linear whose breakpoints $\mathbf{z}$ are defined by the following set:

$$\mathbf{z} = \{\frac{x_i - x_i^0}{w_i} \mid \forall i \in [d]\}. \tag{A.1.9}$$

Let δ denote the permutation of $\mathbf{x} - \mathbf{x}^0$ and $\mathbf{w}$ such that $z_{\delta(1)} \leq z_{\delta(2)} \leq \cdots \leq z_{\delta(d)}$. Thus, the values of g at the breakpoints are:

$$\begin{aligned} g(z_i) &= \sum_{j=1}^{d} w_j \max\{x_j - x_j^0 - w_j z_{\delta(i)}, 0\} \\ &= \sum_{j=i+1}^{d} w_{\delta(j)}(x_{\delta(j)} - x_{\delta(j)}^0 - w_{\delta(j)} z_{\delta(i)}) \end{aligned}$$

For any $z_i < \lambda^*$, $y_i = x_i^0$ (A.1.5) and $g(z_i) > b_0$ (since g is non-increasing). This implies:

$$\begin{aligned} &\sum_{j=i+1}^{d} w_{\delta(j)}(x_{\delta(j)} - x_{\delta(j)}^0 - w_{\delta(j)} z_{\delta(i)}) > b_0 \\ \implies &\frac{-b_0 + \sum_{j=i+1}^{d} w_{\delta(j)}(x_{\delta(j)} - x_{\delta(j)}^0)}{\sum_{j=i+1}^{d} w_{\delta(j)}^2} > z_{\delta(i)} \end{aligned}$$

Thus, $y_i = x_i^0$ for $i \in [i_{max}]$, with:

$$i_{max} := \arg\max\{i \mid \frac{-b_0 + \sum_{j=i+1}^{d} w_{\delta(j)}(x_{\delta(j)} - x_{\delta(j)}^0)}{\sum_{j=i+1}^{d} w_{\delta(j)}^2} > z_{\delta(i)}\}$$

By calculating i_{max}, we obtain λ^* as follows:

$$\begin{aligned} g(\lambda^*) &= \sum_{j=i_{max}+1}^{d} w_{\delta(j)}(x_{\delta(j)} - x_{\delta(j)}^0 - w_{\delta(j)}\lambda^*) = b_0 \\ \implies \lambda^* &= \frac{-b_0 + \sum_{j=i_{max}+1}^{d} w_{\delta(j)}(x_{\delta(j)} - x_{\delta(j)}^0)}{\sum_{j=i_{max}+1}^{d} w_{\delta(j)}^2} \end{aligned}$$

This proof shows that there exists a unique solution $\mathbf{x}^{\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}}$ to Problem A.1.3 such that:

$$x_i^{\Delta^{b_0}_{\mathbf{w},\mathbf{x}^0}} = x_i^0 + \max\{x_i - (w_i\lambda^* + x_i^0), 0\}, \forall i \in [d]. \tag{A.1.10}$$

□

A.2 Upper- and Lower-Bounded Projection

Lemma 5. *Let use the notation defined so far, and let $\hat{\mathbf{x}}^b$ and $\hat{\mathbf{x}}^s$ be the solutions to 5.4.9 and 5.4.10 respectively. If $\mathbf{x} \notin \mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ which implies $\hat{\mathbf{x}} \notin \mathcal{B}^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}$, then there exists a unique real λ^* such that:*

$$\hat{x}_i^s = \begin{cases} \hat{x}_i^0 & \text{if } \hat{x}_i - \hat{x}_i^0 \leq \lambda^* w_i \\ \hat{b}_i & \text{if } \hat{x}_i - \hat{b}_i \geq \lambda^* w_i \\ \hat{x}_i - \lambda^* w_i & \text{if } \hat{x}_i - \hat{b}_i < \lambda^* w_i < \hat{x}_i - \hat{x}_i^0 \end{cases} \tag{A.2.1}$$

Based on lemma 1 and using the inverse transformation associated to 5.4.8, the projection onto $\mathcal{B}^{b_0}_{\mathbf{w},\mathbf{x}^0}$ $\mathbf{x}^b$ can be written as follows:

$$x_i^b = \hat{x}_i^b + a_i = a_i + sign(x_i - x_i^0)(x_i^s - x_i^0) \tag{A.2.2}$$

where $\mathbf{x}^s = \hat{\mathbf{x}}^s + \mathbf{a}$ and sign is the signum function.

Proof. Let assume that $\hat{\mathbf{x}}$ the vector to be projected is outside $\mathcal{B}^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}$. The bounded projection onto $\Delta^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}} \subset \mathcal{B}^{b_0}_{\mathbf{w},\hat{\mathbf{x}}^\mathbf{0}}$ is formulated in Equation (5.4.10). The Lagrangian of this optimization problem can be written as follows:

$$\mathcal{L} = \frac{1}{2}\left\|\hat{\mathbf{y}} - \hat{\mathbf{x}}\right\|_2^2 + \lambda(\mathbf{w}^T(\hat{\mathbf{y}} - \hat{\mathbf{x}}^\mathbf{0}) - b_0) - \mu^T(\hat{\mathbf{b}} - \hat{\mathbf{y}}) - \gamma^T(\hat{\mathbf{y}} - \hat{\mathbf{x}}^\mathbf{0}) \tag{A.2.3}$$

where $\lambda \in \mathbb{R}$ is a Lagrangian multiplier and $\mu, \gamma \in (\mathbb{R}^{*+})^d$ are two vectors of non-negative Lagrange multipliers. Using KKT conditions [121], the optimum satisfies:

$$\nabla_y \mathcal{L} = \hat{y}_i - \hat{x}_i + \lambda w_i + \mu_i - \gamma_i = 0 \tag{A.2.4}$$

$$\mu_i(\hat{b}_i - \hat{y}_i) = 0 \tag{A.2.5}$$

$$\gamma_i(\hat{y}_i - \hat{x}_i^0) = 0 \tag{A.2.6}$$

The second complementary slackness KKT conditions in Equation (A.2.6), implies that $\hat{y}_i = \hat{x}_i^0$ when $\hat{x}_i - \hat{x}_i^0 = \lambda w_i - \gamma_i$. Since $\gamma_i > 0$, then $\hat{y}_i = \hat{x}_i^0$ if $\hat{x}_i - \hat{x}_i^0 \geq \lambda w_i$. The first complementary slackness KKT condition in Equation (A.2.5) implies that $\hat{y}_i = \hat{b}_i$ when $\hat{b}_i - \hat{x}_i + \lambda w_i + \mu_i = 0$. As $\mu_i > 0$ thus $\hat{y}_i = \hat{b}_i$ when $\hat{x}_i - \hat{b}_i \geq +\lambda w_i$. Now, using both complementary slackness KKT conditions, we have $\mu_i = \gamma_i = 0$ when $\hat{x}_i^0 < \hat{y}_i < \hat{b}_i$ which implies that : $\hat{y}_i = \hat{x}_i - \lambda w_i$ when $\hat{x}_i - \hat{b}_i < \lambda w_i < \hat{x}_i - \hat{x}_i^0$. Therefore, for a given λ, $\hat{y}_i$'s are divided into three disjoint sub-sets denoted by L, U, C respectively:

$$\hat{y}_i = \begin{cases} \hat{x}_i^0 & \text{if } \hat{x}_i - \hat{x}_i^0 \leq \lambda w_i : L \\ \hat{b}_i & \text{if } \hat{x}_i - \hat{b}_i \geq \lambda w_i : U \\ \hat{x}_i - \lambda w_i & \text{if } \hat{x}_i - \hat{b}_i < \lambda w_i < \hat{x}_i - \hat{x}_i^0 : C \end{cases} \tag{A.2.7}$$

The budget constraint can therefore be expressed as,

$$b_0 = \sum_{i=1}^{d} w_i(\hat{y}_i - \hat{x}_i^0) \tag{A.2.8}$$

$$= \sum_{i \in U} w_i(\hat{b}_i - \hat{x}_i^0) + \sum_{i \in C} w_i(\hat{x}_i - \hat{x}_i^0) - \lambda \sum_{i \in C} w_i^2 \tag{A.2.9}$$

$$= \overbrace{\sum_{i=1}^{d} w_i(\hat{x}_i - \hat{x}_i^0)}^{S_{\mathbf{w},\hat{x}^0}} - \sum_{i \in U \cup L} w_i(\hat{x}_i - \hat{x}_i^0) + \sum_{i \in U} w_i(\hat{b}_i - \hat{x}_i^0) - \lambda \sum_{i \in C} w_i^2 \tag{A.2.10}$$

To find the optimal boundaries of the sub-sets L, U and C which therefore allow to estimate λ, we use the approach proposed by Gupta et al. in [127]. Denoting the optimal sub-sets by L^*, U^* and C^* as well as the optimal λ by λ^*, we obtain,

$$\lambda^* = \frac{S_{\mathbf{w},\hat{\mathbf{x}}^0} - \sum_{i \in U^* \cup L^*} w_i(\hat{x}_i - \hat{x}_i^0) + \sum_{i \in U^*} w_i(\hat{b}_i - \hat{x}_i^0) - b_0}{\sum_{i \in C^*} w_i^2} \tag{A.2.11}$$

By definition, L^*, U^* and C^* are unique, hence the same goes for λ^*. □

Bibliography

[1] G David Kleinbaum. Survival analysis: A self-learning text (statistics in the health sciences) springer verlag. *New York, New York*, 1996.

[2] David Roxbee Cox and David Oakes. *Analysis of survival data*. Chapman and Hall/CRC, 2018.

[3] Melinda Mills. *Introducing survival and event history analysis*. Sage, 2010.

[4] Ping Wang, Yan Li, and Chandan K Reddy. Machine learning for survival analysis: A survey. *ACM Computing Surveys (CSUR)*, 51(6):1–36, 2019.

[5] Kwan-Moon Leung, Robert M Elashoff, and Abdelmonem A Afifi. Censoring issues in survival analysis. *Annual review of public health*, 18(1):83–104, 1997.

[6] John P Klein and Melvin L Moeschberger. *Survival analysis: techniques for censored and truncated data*. Springer Science and Business Media, 2006.

[7] John P Klein, Hans C Van Houwelingen, Joseph G Ibrahim, and Thomas H Scheike. *Handbook of survival analysis*. CRC Press, 2016.

[8] Håvard Kvamme and Ørnulf Borgan. Continuous and discrete-time survival prediction with neural networks. *arXiv preprint arXiv:1910.06724*, 2019.

[9] Peter C Austin and Jeffrey S Hoch. Estimating linear regression models in the presence of a censored independent variable. *Statistics in medicine*, 23(3):411–429, 2004.

[10] Roberto Rigobon and Thomas M Stoker. Estimation with censored regressors: Basic issues. *International Economic Review*, 48(4):1441–1467, 2007.

[11] John H Bompas-Smith. Mechanical survival. 1973.

[12] Paul D Allison. *Event history and survival analysis*. Routledge, 2018.

[13] Junxiang Lu. Predicting customer churn in the telecommunications industry—-an application of survival analysis modeling using sas. In *SAS User Group International (SUGI27) Online Proceedings*, volume 114, 2002.

[14] Maria Stepanova and Lyn Thomas. Survival analysis methods for personal loan data. *Operations Research*, 50(2):277–289, 2002.

[15] John R Mattox II and Darryl L Jinkerson. Using survival analysis to demonstrate the effects of training on employee retention. *Evaluation and Program Planning*, 28(4):423–430, 2005.

[16] Jeffrey L Carson, Mark A Kelley, Amy Duff, John G Weg, William J Fulkerson, Harold I Palevsky, J Sanford Schwartz, B Taylor Thompson, John Popovich Jr, Thomas E Hobbins, et al. The clinical course of pulmonary embolism. *New England Journal of Medicine*, 326(19):1240–1245, 1992.

[17] Thomas R. Fleming and D. Y. Lin. Survival analysis in clinical trials: Past developments and future directions. *Biometrics*, 56(4):971–983, 2000.

[18] Daniela-Emanuela Danacica and Ana-Gabriela Babucea. Using survival analysis in economics. *survival*, 11:15, 2010.

[19] Adrian Gepp and Kuldeep Kumar. The role of survival analysis in financial distress prediction. *International research journal of finance and economics*, 16(2008):13–34, 2008.

[20] RE Bogacki, RJ Hunt, M Del Aguila, and WR Smith. Survival analysis of posterior restorations using an insurance claims database. *Operative dentistry*, 27(5):488–492, 2002.

[21] Jake Ansell, Tina Harrison, and Tom Archibald. Identifying cross-selling opportunities, using lifestyle segmentation and survival analysis. *Marketing Intelligence and Planning*, 2007.

[22] Ryan C Fields, Klaus J Busam, Joanne F Chou, Katherine S Panageas, Melissa P Pulitzer, Dennis H Kraus, Mary S Brady, and Daniel G Coit. Recurrence and survival in patients undergoing sentinel lymph node biopsy for merkel cell carcinoma: analysis of 153 patients from a single institution. *Annals of Surgical Oncology*, 18(9):2529–2537, 2011.

[23] Robert T Perri, Robert P Hebbel, and Martin M Oken. Influence of treatment and response status on infection risk in multiple myeloma. *The American journal of medicine*, 71(6):935–940, 1981.

[24] WS Kendal. Suicide and cancer: a gender-comparative study. *Annals of Oncology*, 18(2):381–387, 2007.

[25] Michael D Lesem, Tram K Tran-Johnson, Robert A Riesenberg, David Feifel, Michael H Allen, Robert Fishman, Daniel A Spyker, John H Kehne, and James V Cassella. Rapid acute treatment of agitation in individuals with schizophrenia: multicentre, randomised, placebo-controlled study of inhaled loxapine. *The British Journal of Psychiatry*, 198(1):51–58, 2011.

[26] Philip Hougaard. Fundamentals of survival data. *Biometrics*, 55(1):13–22, 1999.

[27] Manfred S Green and Michael J Symons. A comparison of the logistic risk function and the proportional hazards model in prospective epidemiologic studies. *Journal of chronic diseases*, 36(10):715–723, 1983.

[28] Sunao Moriguchi, Yoshio Hayashi, Yoshiaki Nose, Yoshihiko Maehara, Daisuke Korenaga, and Keizo Sugimachi. A comparison of the logistic regression and the cox proportional hazard models in retrospective studies on the prognosis of patients with castric cancer. *Journal of surgical oncology*, 52(1):9–13, 1993.

[29] Cox D.R. Regression models and life tables (with discussion). *Journal of the Royal Statistical Society. Series B. 34*, pages 187–220, 1972.

[30] Changhee Lee, William R Zame, Jinsung Yoon, and Mihaela van der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deep survival: A deep cox proportional hazards network. *stat*, 1050:2, 2016.

[32] David Faraggi and Richard Simon. A neural network model for survival data. *Statistics in medicine*, 14(1):73–82, 1995.

[33] Glen H Lemon. Maximum likelihood estimation for the three parameter weibull distribution based on censored samples. *Technometrics*, 17(2):247–254, 1975.

[34] Olivier Bouaziz. The effect of ignoring censoring in survival analysis: theoretical and practical considerations.

[35] A. Bennis, S. Mouysset, and M. Serrurier. Estimation of conditional mixture weibull distribution with right censored data using neural network for time-to-event analysis. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2020.

[36] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[37] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[38] Achraf Bennis, Sandrine Mouysset, and Mathieu Serrurier. Dpwte: A deep learning approach to time-to-event analysis using a sparse weibull mixture layer. 2021.

[39] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[40] Safoora Yousefi, Fatemeh Amrollahi, Mohamed Amgad, Chengliang Dong, Joshua E Lewis, Congzheng Song, David A Gutman, Sameer H Halani, Jose Enrique Velazquez Vega, Daniel J Brat, et al. Predicting clinical outcomes from large scale cancer genomic profiles with deep survival models. *Scientific reports*, 7(1):1–11, 2017.

[41] Stephane Fotso. Deep neural networks for survival analysis based on a multi-task framework. *arXiv preprint arXiv:1801.05512*, 2018.

[42] Margaux Luck, Tristan Sylvain, Héloïse Cardinal, Andrea Lodi, and Yoshua Bengio. Deep learning for patient-specific kidney graft survival analysis. *arXiv preprint arXiv:1705.10245*, 2017.

[43] E. Martinsson. Weibull time to event recurrent neural network. *Doctoral dissertation, Chalmers University of Technology University of Gothenburg*, 2016.

[44] Xinliang Zhu, Jiawen Yao, and Junzhou Huang. Deep convolutional neural network for survival analysis with pathological images. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 544–547. IEEE, 2016.

[45] Brent B Benda. Gender differences in life-course theory of recidivism: A survival analysis. *International Journal of Offender Therapy and Comparative Criminology*, 49(3):325–342, 2005.

[46] Rupert G Miller Jr. *Survival analysis*, volume 66. John Wiley and Sons, 2011.

[47] Frank Emmert-Streib and Matthias Dehmer. Introduction to survival analysis in practice. *Machine Learning and Knowledge Extraction*, 1(3):1013–1038, 2019.

[48] Anastasios Tsiatis. *Semiparametric theory and missing data*. Springer Science and Business Media, 2007.

[49] JS Williams and SW Lagakos. Models for censored survival analysis: Constant-sum and variable-sum models. *Biometrika*, 64(2):215–224, 1977.

[50] Jonathan Buckley and Ian James. Linear regression with censored data. *Biometrika*, 66(3):429–436, 1979.

[51] Maxim Finkelstein. *Failure rate modelling for reliability and risk*. Springer Science and Business Media, 2008.

[52] Stephen W Lagakos. General right censoring and its impact on the analysis of survival data. *Biometrics*, pages 139–156, 1979.

[53] Frank E Harrell, Robert M Califf, David B Pryor, Kerry L Lee, and Robert A Rosati. Evaluating the yield of medical tests. *Jama*, 247(18):2543–2546, 1982.

[54] Hemant Ishwaran, Udaya B Kogalur, Eugene H Blackstone, Michael S Lauer, et al. Random survival forests. *The annals of applied statistics*, 2(3):841–860, 2008.

[55] Thomas A Gerds, Tianxi Cai, and Martin Schumacher. The performance of risk prediction models. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 50(4):457–479, 2008.

[56] Laura Antolini, Patrizia Boracchi, and Elia Biganzoli. A time-dependent discrimination index for survival data. *Statistics in medicine*, 24(24):3927–3944, 2005.

[57] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[58] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[59] Erika Graf, Claudia Schmoor, Willi Sauerbrei, and Martin Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, 18(17-18):2529–2545, 1999.

[60] Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and cox regression. *Journal of Machine Learning Research*, 20(129):1–30, 2019.

[61] Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

[62] Surveillance Research Program National Cancer Institute, DCCPS. Surveillance, epidemiology, and end results (seer) program (www.seer.cancer.gov) seer*stat database: Incidence - seer 18 regs research data + hurricane katrina impacted louisiana cases, nov 2018 sub (1975-2016 varying), 2019. Linked To County Attributes - Total U.S., 1969-2017 Counties, released April 2019, based on the November 2018 submission.

[63] Odd Aalen. Nonparametric inference for a family of counting processes. *The Annals of Statistics*, pages 701–726, 1978.

[64] Wayne Nelson. Theory and applications of hazard plotting for censored failure data. *Technometrics*, 14(4):945–966, 1972.

[65] Hans van Houwelingen and Hein Putter. *Dynamic prediction in clinical survival analysis*. CRC Press, 2011.

[66] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[67] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[68] Robert Tibshirani. The lasso method for variable selection in the cox model. *Statistics in medicine*, 16(4):385–395, 1997.

[69] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[70] Hao Helen Zhang and Wenbin Lu. Adaptive lasso for cox's proportional hazards model. *Biometrika*, 94(3):691–703, 2007.

[71] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[72] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[73] Pierre JM Verweij and Hans C Van Houwelingen. Penalized likelihood in cox regression. *Statistics in medicine*, 13(23-24):2427–2436, 1994.

[74] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[75] Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of statistical software*, 39(5):1, 2011.

[76] John D Kalbfleisch and Ross L Prentice. *The statistical analysis of failure time data*, volume 360. John Wiley and Sons, 2011.

[77] John H Pollard and Emil J Valkovics. The gompertz distribution and its applications. *Genus*, pages 15–28, 1992.

[78] Steve Bennett. Log-logistic regression models for survival data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 32(2):165–171, 1983.

[79] E Webb Stacy and G Arthur Mihram. Parameter estimation for a generalized gamma distribution. *Technometrics*, 7(3):349–358, 1965.

[80] Patrick Royston and Mahesh KB Parmar. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in medicine*, 21(15):2175–2197, 2002.

[81] Alessandra Nardi and Michael Schemper. Comparing cox and parametric models in clinical studies. *Statistics in medicine*, 22(23):3597–3610, 2003.

[82] Xinliang Zhu, Jiawen Yao, Feiyun Zhu, and Junzhou Huang. Wsisa: Making survival prediction from whole slide histopathological images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7234–7242, 2017.

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

[84] Elia Biganzoli, Patrizia Boracchi, Luigi Mariani, and Ettore Marubini. Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. *Statistics in medicine*, 17(10):1169–1186, 1998.

[85] Chun-Nam Yu, Russell Greiner, Hsiu-Chin Lin, and Vickie Baracos. Learning patient-specific cancer survival distributions as a sequence of dependent regressors. *Advances in Neural Information Processing Systems*, 24:1845–1853, 2011.

[86] Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969.

[87] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in statistics*, pages 66–70. Springer, 1992.

[88] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[89] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[90] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[91] Torsten Hothorn, Berthold Lausen, Axel Benner, and Martin Radespiel-Tröger. Bagging survival trees. *Statistics in medicine*, 23(1):77–91, 2004.

[92] GS Watson and MR Leadbetter. Hazard analysis. i. *Biometrika*, 51(1/2):175–184, 1964.

[93] Narayanaswamy Balakrishnan, Norman L Johnson, and Samuel Kotz. *Continuous univariate distributions*. 1994.

[94] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[95] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[96] Zongben Xu, Hai Zhang, Yao Wang, XiangYu Chang, and Yong Liang. L 1/2 regularization. *Science China Information Sciences*, 53(6):1159–1169, 2010.

[97] Jianqing Fan, Heng Peng, et al. Nonconcave penalized likelihood with a diverging number of parameters. *The Annals of Statistics*, 32(3):928–961, 2004.

[98] Terry M Therneau and Thomas Lumley. Package 'survival'. *R Top Doc*, 128(10):28–33, 2015.

[99] Christina Curtis, Sohrab P Shah, Suet-Feung Chin, Gulisa Turashvili, Oscar M Rueda, Mark J Dunning, Doug Speed, Andy G Lynch, Shamith Samarajiwa, Yinyin Yuan, et al. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*, 486(7403):346–352, 2012.

[100] William A Knaus, Frank E Harrell, Joanne Lynn, Lee Goldman, Russell S Phillips, Alfred F Connors, Neal V Dawson, William J Fulkerson, Robert M Califf, Norman Desbiens, et al. The support prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of internal medicine*, 122(3):191–203, 1995.

[101] W Kalgren Patrick, Carl Byington, Michael Roemer, et al. Defining phm, a lexical evolution of main-tenance and logistics. In *IEEE AUTOTESTCON 2006 Conference Record. Anaheim, California*, pages 353–358, 2006.

[102] André Listou Ellefsen, Emil Bjørlykhaug, Vilmar Æsøy, Sergey Ushakov, and Houxiang Zhang. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering and System Safety*, 183:240–251, 2019.

[103] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE, 2008.

[104] Dean K Frederick, Jonathan A DeCastro, and Jonathan S Litt. User's guide for the commercial modular aero-propulsion system simulation (c-mapss). 2007.

[105] Jun Peng, Shengnan Wang, Dianzhu Gao, Xiaoyong Zhang, Bin Chen, Yijun Cheng, Yingze Yang, Wentao Yu, and Zhiwu Huang. A hybrid degradation modeling and prognostic method for the multi-modal system. *Applied Sciences*, 10(4):1378, 2020.

[106] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[107] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[108] Boxin Tang. Orthogonal array-based latin hypercubes. *Journal of the American statistical association*, 88(424):1392–1397, 1993.

[109] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[110] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *2015 IEEE International Conference on Data Mining*, pages 301–309. IEEE, 2015.

[111] Andre T Nguyen and Edward Raff. Adversarial attacks, regression, and numerical stability regularization. *arXiv preprint arXiv:1812.02885*, 2018.

[112] Carl-Johann Simon-Gabriel, Yann Ollivier, Léon Bottou, Bernhard Schölkopf, and David Lopez-Paz. Adversarial vulnerability of neural networks increases with input dimension. 2018.

[113] Fuxun Yu, Zirui Xu, Yanzhi Wang, Chenchen Liu, and Xiang Chen. Towards robust training of neural networks by regularizing adversarial gradients. *arXiv preprint arXiv:1805.09370*, 2018.

[114] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[115] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[116] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[117] Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[118] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, 2018.

[119] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[120] Abhishek Sinha, Mayank Singh, and Balaji Krishnamurthy. Neural networks in an adversarial setting and ill-conditioned weight space. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 177–190. Springer, 2018.

[121] Morgan A Hanson. On sufficiency of the kuhn-tucker conditions. *Journal of Mathematical Analysis and Applications*, 80(2):545–550, 1981.

[122] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279, 2008.

[123] Laurent Condat. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, 158(1-2):575–585, 2016.

[124] Guillaume Perez, Sebastian Ament, Carla Gomes, and Michel Barlaud. Efficient projection algorithms onto the weighted l1 ball. *arXiv preprint arXiv:2009.02980*, 2020.

[125] Yannis Kopsinis, Konstantinos Slavakis, and Sergios Theodoridis. Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls. *IEEE Transactions on Signal Processing*, 59(3):936–952, 2010.

[126] Michael Held, Philip Wolfe, and Harlan P Crowder. Validation of subgradient optimization. *Mathematical programming*, 6(1):62–88, 1974.

[127] Mithun Das Gupta, Sanjeev Kumar, and Jing Xiao. L1 projections with box constraints. *arXiv preprint arXiv:1010.0141*, 2010.

[128] Clifford Stein, T Cormen, R Rivest, and C Leiserson. Introduction to algorithms. *The MIT Press*, 31(77):13, 2001.