

POLITECNICO DI MILANO

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING

Department of Mathematics

Master of Science in Mathematical Engineering

Performing Survival Analysis via Functional Cox-type Regression and a Machine Learning approach: an application to Heart Failure patients

Author:

Davide BURBA

Matr. 875290

Supervisor:

Prof. Francesca IEVA



Academic Year 2018/2019

"All models are wrong, but some are useful."

George E. P. Box

Contents

List of Figures	7
List of Tables	11
Sommario	13
Abstract	15
Introduction	17
1 Data	21
1.1 Heart Failure (HF)	21
1.2 Regione Lombardia dataset	22
1.3 Preprocessing data	24
1.4 Pipeline of the work	27
2 Methodologies	31
2.1 Survival analysis via Cox-type regression model with time-varying (functional) covariates	31
2.1.1 Introduction to Survival Analysis	31
2.1.2 Cox model for recurrent events	38
2.1.3 Functional Principal Component Analysis (FPCA)	46
2.1.4 Functional Cox-type regression model	47
2.2 Survival Analysis via a Machine Learning approach	50
2.2.1 Introduction to Deep Learning	51
2.2.2 Feedforward Neural Network (FNN)	54
2.2.3 DeepHit	55
2.2.4 Long-Short Term Memory Neural Network (LSTMNN)	57
2.2.5 Deep Recurrent Survival Analysis (DRSA)	60
2.3 Comparison among different models	63

3 Applications and Results	67
3.1 Modelling the clinical history	67
3.1.1 Features selection	68
3.1.2 Smoothing of the cumulative baseline hazard	74
3.1.3 Reconstruction of the compensators' trajectories	77
3.2 Summarise the information content of complex data	82
3.3 Predictive Survival Models	85
3.3.1 Functional Cox-type regression model	87
3.3.2 Machine Learning models	90
3.4 Performance Comparison	96
Conclusions	103
A Code	105
A.1 Fit compensators of stochastic processes	105
A.2 FPCA	109
A.3 Functional Cox-type regression model	111
A.4 DeepHit	113
A.5 DRSA	118
Bibliography	125

List of Figures

1.1	Timeline division for a hypothetical HF patient.	25
1.2	Pipeline applied on HF Data from Regione Lombardia [HFData-RF-2009-1483329].	29
2.1	Representation of an Artificial Neuron.	51
2.2	Representation of a Feedforward Neural Network (UC-R-Programming, 2018).	55
2.3	Long-Short Term Memory cell (modified version of Wikimedia-Commons, 2019).	60
3.1	Output of the cross-validation procedure used to select the best set of features for the Cox model for recurrent events describing the stochastic process relative to the purchase of ACE inhibitors.	70
3.2	Fitted cumulative baseline hazard of the Cox model for recurrent events describing the stochastic process of purchase of ACE inhibitors, fitted with the Breslow estimator and smoothed according to the procedure described in Baraldo et al., 2013.	75
3.3	Fitted cumulative baseline hazard of the Cox model for recurrent events describing the stochastic process of purchase of BB agents, fitted with the Breslow estimator and smoothed according to the procedure described in Baraldo et al., 2013.	75
3.4	Fitted cumulative baseline hazard of the Cox model for recurrent events describing the stochastic process of purchase of AA agents, fitted with the Breslow estimator and smoothed according to the procedure described in Baraldo et al., 2013.	76
3.5	Fitted cumulative baseline hazard of the Cox model for recurrent events describing the stochastic process of HF hospitalisations, fitted with the Breslow estimator and smoothed according to the procedure described in Baraldo et al., 2013.	76

3.6	Compensators of the stochastic process describing the purchase of ACE inhibitors, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.	78
3.7	Compensators of the stochastic process describing the purchase of BB agents, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.	78
3.8	Compensators of the stochastic process describing the purchase of AA agents, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.	79
3.9	Compensators of the stochastic process describing the HF rehospitalisations, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.	79
3.10	Residuals of the compensators of the stochastic process describing the purchase of ACE inhibitors, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.	80
3.11	Residuals of the compensators of the stochastic process describing the purchase of BB agents, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.	81
3.12	Residuals of the compensators of the stochastic process describing the purchase of AA agents, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.	81
3.13	Residuals of the compensators of the stochastic process describing the HF rehospitalisations, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.	82
3.14	First two Functional Principal Components of the compensator of the stochastic process describing the purchase of ACE inhibitors. In the top panels, the two Principal Components. In the bottom panels, the average compensator 10 times the Principal Components.	83

3.15	First two Functional Principal Components of the compensator of the stochastic process describing the purchase of BB agents. In the top panels, the two Principal Components. In the bottom panels, the average compensator 10 times the Principal Components.	84
3.16	First two Functional Principal Components of the compensator of the stochastic process describing the purchase of AA agents. In the top panels, the two Principal Components. In the bottom panels, the average compensator 10 times the Principal Components.	84
3.17	First two Functional Principal Components of the compensator of the stochastic process describing the purchase of HF hospitalisations. In the top panels, the two Principal Components. In the bottom panels, the average compensator 5 times the Principal Components.	85
3.18	Cross validated C-indexes for 14 possible sets of covariates for the Cox-type regression model used to predict survival time of HF patients.	88
3.19	Survival curves obtained by fitting a Cox-type regression model for a sample of 500 HF patients in the test set.	88
3.20	Loss function of the DeepHit model in the training phase, for each epoch. On the left, the loss function evaluated on the training set (dropout is applied). On the right, the loss function evaluated on the test set.	92
3.21	Predicted survival curves obtained by fitting a Deephit model for a sample of 500 HF patients in the test set.	93
3.22	Loss function of the DeepHit model in the training phase, for each epoch. On the left, the loss function evaluated on the training set (dropout is applied). On the right, the loss function evaluated on the test set.	95
3.23	Discrete hazard functions obtained by fitting a DRSA model for a sample of 500 HF patients in the test set.	95
3.24	Predicted survival curves obtained by fitting a DRSA model for a sample of 500 HF patients in the test set.	96

3.25	Survival curves obtained by fitting three different models for a sample of 500 HF patients in the test set. Panel (A): predictions obtained with the Cox-type regression model. Panel (B): predictions obtained with the DeepHit model. Panel (C): predictions obtained with the DRSA model.	98
3.26	Predicted survival curves for a sample of 500 HF obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models. The red curves correspond to the group of Females Over 75. .	100
3.27	Predicted survival curves for a sample of 500 HF obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models. The red curves correspond to the group of Females Under 75. .	100
3.28	Predicted survival curves for a sample of 500 HF obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models. The red curves correspond to the group of Males Over 75. . .	101
3.29	Predicted survival curves for a sample of 500 HF obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models. The red curves correspond to the group of Males Under 75. . .	101

List of Tables

1.1	Description of the main variables in the HF dataset [HFData-RF-2009-1483329].	23
1.2	Main groups of drugs for HF treatments and corresponding patterns for the relative ATC codes.	24
1.3	Description of variables in each of the four reformatted HF datasets (one for each considered process), regarding events happened in the observation period.	26
1.4	Reformatted HF hospitalisation dataset for patient ID 10000471.	27
1.5	Reformatted ACE inhibitors dataset for patient ID 10009476. .	27
3.1	Hazard ratios and corresponding 95% CI of the Cox model for recurrent events for the stochastic process describing the purchase of ACE inhibitors.	71
3.2	Hazard ratios and corresponding 95% CI of the Cox model for recurrent events for the stochastic process describing the purchase of BB agents.	72
3.3	Hazard ratios and corresponding 95% CI of the Cox model for recurrent events for the stochastic process describing the purchase of AA agents.	73
3.4	Hazard ratios and corresponding 95% CI of the Cox model for recurrent events for the stochastic process describing the HF rehospitalisations.	74
3.5	Description of the variables in the dataset used to predict survival time of individuals, enriched with the FPCA scores obtained in Section 3.2.	86
3.6	Fitted coefficients of the Cox-type regression model describing the survival time of HF patients.	89
3.7	Hyper-parameters choice for the training of the DeepHit model used to predict survival time of HF patients.	91
3.8	Hyper-parameters choice for the training of the DRSA model used to predict survival time of HF patients.	94

3.9 Sample Concordance Indexes for the 1,362 patients in the test set obtained with Cox-type regression, Deephit and DRSA models.	97
-------------------------------------------------------------------------------------------------------------------------------------------	----

Sommario

In questa tesi abbiamo sviluppato un'efficace metodologia che consente di introdurre l'informazione contenuta in particolari variabili tempo varianti - i compensatori dei processi di conteggio marcati - all'interno di modelli di sopravvivenza. L'elevata generalità di questi processi, ampiamente utilizzati per modellare eventi ricorrenti, permette di cogliere e modellare fenomeni complessi e quantificarne il contributo su altri processi (ad esempio, la sopravvivenza) per i quali si desidera fare previsione.

La metodologia proposta consiste nella ricostruzione della hazard function del processo di conteggio marcato che descrive l'evento tempo-variante di interesse, per poi riassumerla tramite la riduzione dimensionale consentita dall'Analisi delle Componenti Principali Funzionali (FPCA). Le nuove variabili così ottenute vengono poi introdotte in un modello previsivo tradizionale per un secondo processo, che si suppone dipendente e influenzato da quello per cui si è eseguita la procedura descritta.

L'introduzione di variabili tempo varianti legate alla realizzazione di processi stocastici marcati consente tra le altre cose di modellare comportamenti auto-eccitanti, per i quali il verificarsi di eventi nel passato aumenta la probabilità di un nuovo evento.

Abbiamo applicato questa metodologia all'analisi di dati forniti da Regione Lombardia in relazione alla sopravvivenza e all'utilizzo di farmaci di pazienti affetti da Scompenso Cardiaco. Nello specifico, abbiamo ricostruito le hazard relative ai processi di riospedalizzazione e acquisto di alcune categorie di farmaci e abbiamo quantificato il loro effetto sulla sopravvivenza di lungo periodo dei pazienti.

Per la modellazione della sopravvivenza, oltre al classico modello di Cox, abbiamo considerato due modelli di sopravvivenza basati su approcci di tipo Deep Learning (DeepHit e DRSA nel seguito) i quali, diversamente dal modello di Cox, non fanno assunzioni sulla forma del processo generativo della sopravvivenza dei pazienti. In tutti questi modelli abbiamo inserito le variabili relative ai processi di riospedalizzazione e acquisto dei farmaci sotto forma degli scores ottenuti dalla FPCA.

Infine, abbiamo confrontato le prestazioni dei tre diversi modelli per evidenziare punti di forza e di debolezza di ciascuno e dare indicazioni operative ai potenziali utilizzatori.

Tutte le analisi sono eseguite con i linguaggi di programmazione R e PYTHON, e il codice è disponibile alla repository <https://github.com/davideburba/thesis>.

Abstract

In this thesis we have developed an effective methodology that allows us to introduce the information contained in particular time varying variables - the compensators of marked counting processes - into survival models. The high generality of these processes, widely used to model recurrent events, allows us to capture and model complex phenomena and quantify their contribution to other processes (e.g., survival) for which we want to make forecasts.

The proposed methodology consists in the reconstruction of the hazard function of the marked counting process that describes the time-varying event of interest, to then summarise it through the dimensionality reduction allowed by Functional Principal Components Analysis (FPCA). The new variables thus obtained are then introduced into a traditional forecasting model for a second process, which is supposed to be dependent and influenced by the one for which the described procedure was performed.

Among other things, the introduction of time varying variables linked to the realisation of marked stochastic processes allows to model self-exciting behaviours, for which the occurrence of events in the past increases the probability of a new event.

We applied this methodology to the analysis of data provided by Regione Lombardia in relation to the survival and use of drugs from patients suffering from Heart Failure. Specifically, we reconstructed the hazards related to the processes of rehospitalisation and purchase of some categories of drugs, and we quantified their effect on the long-term survival of patients.

For survival modelling, in addition to the classic Cox-type regression model we considered two survival models based on Deep Learning (DeepHit and DRSA in the following) which, unlike the Cox-type regression model, do not make assumptions about the form of the generative process of patients' survival. In all these models, we inserted the variables relative to the processes of rehospitalisations and purchase of drugs in the form of the scores obtained through the FPCA.

Finally, we compared the performance of the three different models to

highlight the strengths and weaknesses of each one and to provide operational indications to potential users.

All the analyses are performed with R and PYTHON programming languages, and the code is available at the repository <https://github.com/davide-burba/thesis>.

Introduction

Within this work, we aim to propose an effective methodology to extract and summarise information from functional data intended as trajectories of suitable stochastic processes representing recurrent events, plugging them into survival models.

A recurrent event is an event which may occur more than once. Many situations can be modelled in the framework of recurrent events, for instance in the field of reliability, medicine, social sciences, economics and business.

Looking at the recurrent events for a set of individuals as particular stochastic processes, namely marked counting processes, we assumed the form of the generating model described in Andersen and Gill, 1982.

Several approaches have been proposed throughout literature to model recurrent events. Different approaches may differ in assumptions and in the interpretation of results. A common characteristic of recurrent events models is the necessity to account for correlation between repeated events regarding the same individual.

The extended Cox model proposed in Andersen and Gill, 1982 assumes that the correlation between event times for an individual can be explained by past events. Thus, the dependence could be captured by appropriate specification of time-dependent covariates which are functions of the realisation of past events, such as the number of previous events. The model is suitable when correlations among events for each individual are induced by measured covariates (Amorim and Cai, 2014).

The Prentice, Williams and Peterson (PWP) model introduced in Prentice, Williams, and Peterson, 1981 analyses ordered multiple events by stratification, based on the prior number of events during the follow-up period. The model can incorporate both overall and event-specific effects for each covariate. In practice, the data for the PWP model need to be limited to a specific number of recurrent events (Amorim and Cai, 2014).

The random effects approach (Kelly and Lim, 2000), also called the frailty model, introduces a random covariate into the model that induces dependence among the recurrent event times. This model assumes that recurrent

event times are independent conditional on the covariates and random effects. Through this model, dependencies among the interoccurrence times are incorporated, although it is not clear whether they may be appropriate dependencies (Peña and Hollander, 2004). When there is heterogeneous susceptibility to the risk of recurrent events, the frailty model can be applied (Amorim and Cai, 2014).

Another possible approach is the multi-state models (MSM) in the special case in which an individual moves from one state to another through time, and intermediate states are identified (Andersen and Keiding, 2002). A change of state is called a transition; the model is fully characterised through estimation of transition probabilities between states. The MSM for recurrent events is often applied in cases where there are different types of state; a typical example is given by patients whose intermediate states are hospitalisations and whose final state is death.

We decided to use the extended Cox model (Andersen and Gill, 1982) because of its simplicity and flexibility. Furthermore, it has been successfully used in similar contexts. For instance, this approach has been used in Baraldo et al., 2013 and S. Kennedy, 2001 to evaluate repeated occurrence of hospitalisations, which will also be the setting of the application we will consider.

Specifically, we are able to model the compensators (i.e. the predictable part) of these stochastic processes and plug them as time-varying covariates into a Cox-type regression model to predict individuals' survival. This innovative method relies on suitable dimensionality reduction and treatment of this data, namely Functional Principal Component Analysis (FPCA). In doing so, we allow the process under consideration to have a self-exciting behaviour and to depend on the values of its past marks. We perform a cross-validation procedure to carry out a feature selection over the covariates available for reconstructing the smoothed compensators of the counting processes.

In the second part of this work we considered two Machine Learning survival models which, opposite to the Cox-type regression model, do not make assumptions on the underlying stochastic process: DeepHit (C. Lee et al., 2018) and DRSA (Ren et al., 2018). Introducing the aforementioned synthesis of the time-varying covariates as predictors in these models, we used them to predict survival time of individuals as well.

We then compared the performances and predictions of the three different models.

We applied the methodologies developed within this thesis to a dataset of patients affected by Heart Failure (HF). This data is particularly rich since, besides storing patients variables like age, sex and survival time, it also contains observations related to drugs purchase and hospitalisations.

Any analysis aimed at better understanding and predicting the dynamic and the outcomes of HF may have a strong social and managerial impact. Our analysis aims at providing a joint description of the HF related survival and of the processes that may affect it (i.e. drugs intake and hospitalisations).

We considered for each individual four different recurrent events: purchases of three different types of drugs and rehospitalisations due to HF. By applying the previously described technique we obtained summarised information relative to the observation year and we used it to predict the patients' survival time.

In Chapter 1 we will provide a brief overview of HF and describe data. In Chapter 2 we introduce some theoretical background necessary to understand the novel methodology we developed and then present the methods for modelling recurrent events as realisations of marked counting processes. In Chapter 3 we apply our methods to the real case study of HF data.

We conclude our work with a discussion based on the results that we obtained and we lay a few proposals for future developments.

In Appendix A we reported the main parts of the code used for the analysis. Most of the work was developed in R, and it is highly dependent on the survival package (Therneau and Lumley, 2014). For the deep learning models we used the PyTorch framework (Paszke et al., 2017) in PYTHON.

Chapter 1

Data

In this Chapter we give an overview of Heart Failure (HF) and describe the administrative database analysed within this thesis. We also provide a pipeline of the entire work divided by steps performed in order to preprocess the data in preparation to the different analyses carried out in Sections 3.1, 3.2 and 3.3.

1.1 Heart Failure (HF)

Heart Failure (HF) is a syndrome whose symptoms and signs are caused by cardiac dysfunction, resulting in a reduced longevity (Mosterd and Hoes, 2007). It is highly lethal, with a median survival time of 1.7 years in men and 3.2 years in women and a 5-years survival rate of 25% in men and 38% in women. Hypertension and coronary heart disease are the two most common conditions predating its onset (Ho et al., 1993).

Healthcare expenditure on HF in developed countries consumes 1-2% of the total health care budget. The cost of hospitalisation represents the greatest proportion of total expenditure (Berry, Murdoch, and J. J. V. McMurray, 2001).

In Italy, there are about 80,000 new cases recorded per year (Maggioni and Spandonaro, 2014).

The treatment's goals are reduction in symptoms and prolongation of life. It is proven that treatments with Angiotensin Converting Enzyme (ACE) inhibitors, Beta-Blocking (BB) agents and Anti Aldosterone (AA) agents lead to substantial improvements in outcome (J. McMurray et al., 2005).

For all these reasons, any analysis aimed at better understanding and predicting the dynamic and the outcomes of this complex pathology may have a strong social and managerial impact. Our analysis aims at providing a joint

description of the HF related survival and of the process that may affect it (i.e. drugs intake and hospitalisations).

1.2 Regione Lombardia dataset

The data analysed within this thesis is provided by *Regione Lombardia - Healthcare Division*, within the project HFData [HFData-RF-2009-1483329] (Regione Lombardia, 2012). It contains anonymised information about patients hospitalised for Heart Failure (HF) from 2000 to 2012. For our work, we used a representative sample composed by 1,333,954 events related to 4,872 patients with their first HF hospitalisation happened during the period 2006-2012. The reason why 2006 is chosen as the beginning of the observation period is the following: years between 2000 and 2005 are used to check for patients hospitalised after 2006 to be "incident cases" of HF, i.e. with no previous contact with the healthcare system before that date. Then the follow-up of these incident cases of HF is considered up to the administrative censoring date of December 31st, 2012.

The dataset is composed by 73 variables; here we describe only the ones which have actually been used for the analysis.

Each row of the dataset corresponds to an event, to be intended as a hospitalisation or a pharmaceutical drug purchase; therefore we possibly have multiple rows for each patient.

Patients are identified with an ID code. In order to preserve privacy, each identification code was automatically converted into an anonymous code. The inverse process was prevented by deletion of the conversion table.

For each patient we have information like sex, age, the date of discharge from the first HF hospitalisation, the date of death/censoring and the final state of the patient. This can be "dead" if the patient died during the follow-up period, "truncated" if censored or "lost" if lost to follow up.

We also have some event specific information. Since events could be of two different types (hospitalisations or pharmaceutical prescriptions), some variables have different meanings, accordingly.

All the variables are described in Table 1.1.

The variable *tipo_prest* states the type of event, according to the aforementioned description. If the type of event is *hospitalisation*, the date of the event

states the date of discharge from the hospitalisation, and we have information about the length of the stay in the hospital and the "CCS-principal diagnosis" code, that is a code which clusters together similar patient diagnoses (Elixhauser, Steiner, and Palmer, 2008). If the type of event is *pharmacological*, the date of the event states the date of the purchase, and we have information about the coverage days of the prescription and the Anatomical Therapeutic Chemical (ATC), which is a code classifying active substances of drugs according to the organ or system on which they act and their therapeutic, pharmacological and chemical properties (Spreafico, 2017).

Variable name	Description	
<i>COD_REG</i>	Anonymous ID code for each patient	
<i>data_rif_ev</i>	Date of first discharge for HF event	
<i>SESSO</i>	Patient's gender	
<i>eta_Min</i>	Patient's age	
<i>data_studio_out</i>	Date of death/censoring	
<i>desc_studio_out</i>	Label at the end of the study	
	if the event is a "Pharmaceutical prescription"	if the event is a "Hospitalisation"
<i>tipo_prest</i>	Code identifying a pharmaceutical prescription	Code identifying a hospitalisation
<i>class_prest</i>	ATC code	CCS-principal diagnosis (Clinical Classification Software by CMS)
<i>data_prest</i>	Date of prescription	Date of discharge
<i>qt_prest_Sum</i>	Duration of the prescription	Length of stay at the hospital

TABLE 1.1: Description of the main variables in the HF dataset [HFData-RF-2009-1483329].

The belonging of a particular drug to a specific group of medicines can be verified by observing whether the ATC code of the drug contains the pattern corresponding to the group under examination.

In the following analysis we will consider the main types of drugs used for the treatment of HF, i.e. ACE inhibitors, BB agents, and AA agents. Their corresponding patterns for the ATC codes are shown in Table 1.2 (Spreafico, 2017).

Type of drug	ATC code pattern
<i>Angiotensin Converting Enzyme (ACE) Inhibitors</i>	C09A, C09B, C09X
<i>Beta-Blocking (BB) agents</i>	C07
<i>Anti Aldosterone (AA) agents</i>	C03D, C03E

TABLE 1.2: Main groups of drugs for HF treatments and corresponding patterns for the relative ATC codes.

1.3 Preprocessing data

One of the most important aspects of medical studies is the selection of an appropriate cohort of patients. Given the original data, we selected a cohort of patients survived for at least one year after their first hospitalisation. We then excluded 331 patients died within a year from discharge, decreasing the number of patients from 4,872 to 4,541. This is due to the need of defining an "observation period" of one year, where we may observe the processes of drugs intake and hospitalisations to evolve.

In the follow-up period that follows the observation period we observe the survival time of patients. Each of these times could be "censored", if the corresponding patient withdrew before the end of the study or survived to the administrative censoring date. In these cases we only know that the true survival time of the patient is greater than the observed one.

In this Section we explain how we prepared the datasets describing the events occurred to the patients in the one year observation period. In the following analysis, we will reconstruct the compensators of these stochastic processes and then, through FPCA, we will plug this information into survival models in order to predict survival time of patients after the observation period.

We considered four different types of events occurring in the observation period: the **hospitalisations** due to HF and the purchases of **ACE inhibitors**, **BB agents** and **AA agents**. The purchases refer to three different classes of drugs used to manage heart diseases, identified by the ATC codes (see Table 1.2 in Section 1.2).

In Figure 1.1 we show the timeline for a hypothetical patient. In the observation period this patient experienced two rehospitalsations, two purchases of BB agents, one purchase of ACE inhibitors and zero purchases of AA agents. Observe that we set the "time zero" at the end of the observation

period. Indeed, we will predict its survival time starting from this moment, using predictors obtained by analysing the information collected in the observation period.

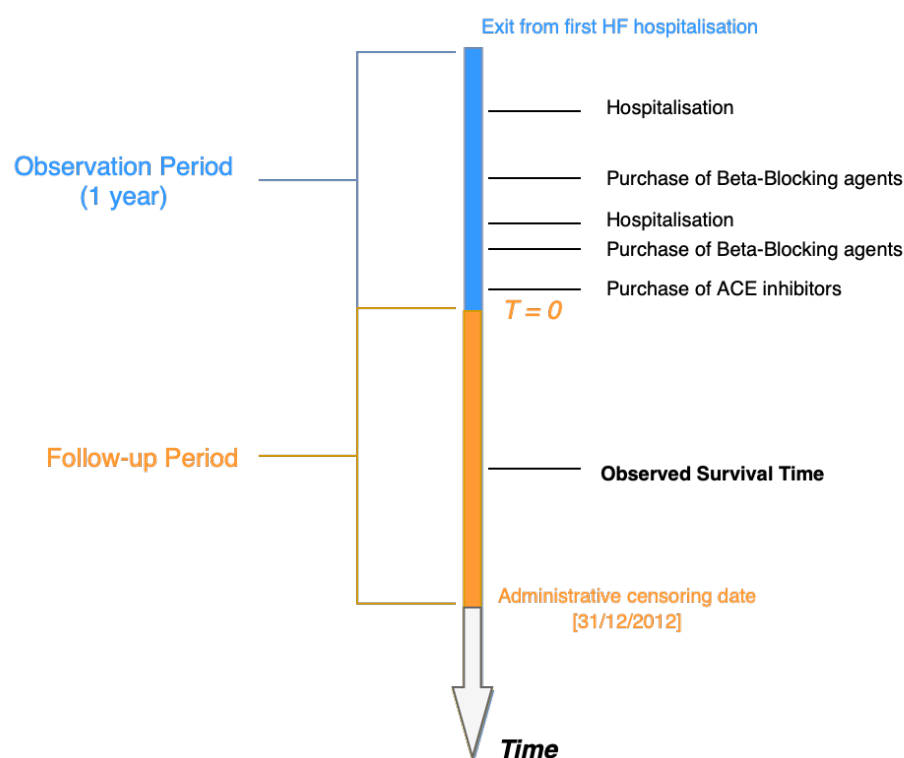


FIGURE 1.1: Timeline division for a hypothetical HF patient.

For each of the four considered types of event (HF rehospitalisations, purchase of ACE inhibitors, purchase of BB agents and purchase of AA agents) we built a dataset with the variables described in Table 1.3.

Variable name	Variable description
<i>id</i>	patient identifier (<i>COD_REG</i>)
<i>start</i>	time of the patient's previous event (equal to -0.5 if it is the first event)
<i>stop</i>	time of the event (equal to 366.5 if it is the censoring event)
<i>status</i>	type of the event indicator (0 if censored, 1 otherwise)
<i>sex</i>	sex indicator (0 if female, 1 if male)
<i>age_in</i>	age at discharge from the first hospitalisation (time = 0)
<i>Nm</i>	number of patient's previous events
<i>y</i>	sum of <i>past qt_prest_Sum</i> (i.e. sum of past days of hospitalisation/coverage)

TABLE 1.3: Description of variables in each of the four reformatted HF datasets (one for each considered process), regarding events happened in the observation period.

The reason why we used the $(start, stop]$ format in Table 1.3 is that it is required from the `coxph` function of survival R package (Therneau and Lumley, 2014).

Note that we introduced two new variables: Nm and y . Variable Nm denotes the number of patient's previous events. Variable y denotes the sum of the past qt_prest_Sum for the patient, thus taking different meanings according to the process which the dataset is describing. In the hospitalisation case, y denotes the number of days spent at the hospital in previous HF events, whilst for drugs purchases (ACE inhibitors, BB agents and AA agents) it denotes the days of coverage of previous drugs purchases.

Using Nm in a model for recurrent events implies the insertion of a self-exciting component, i.e. a dependency on the time to the next event from the number of past events. On the other hand, considering qt_prest_sum as an event's mark and introducing y in a model for recurrent events implies the insertion of a dependency of the time to the next event on the marks history.

In order to build the dataset, we had to use two expedients. Firstly, in case of simultaneous events of the same type for the same patient, we merged the events and we considered the sum of the values for qt_prest_Sum . Secondly,

in order to not have events at time zero or at censoring time, we considered time limits to be -0.5 and 366.5.

As examples, we reported the rehospitalisation dataset for one patient in Table 1.4 and the ACE dataset for another patient in Table 1.5. Note that, in both cases, the values of Nm and y for the first event are always 0 and that the last event has always *status* equal to 0.

id	start	stop	status	sex	age_in	Nm	y
10000471	-0.5	30	1	F	79	0	0
10000471	30	52	1	F	79	1	21
10000471	52	52	1	F	79	2	35
10000471	76	365.5	0	F	79	3	59

TABLE 1.4: Reformatted HF hospitalisation dataset for patient ID 10000471.

id	start	stop	status	sex	age_in	Nm	y
10009476	-0.5	21	1	F	82	0	0
10009476	21	65	1	F	82	1	56
10009476	65	137	1	F	82	2	131
10009476	137	213	1	F	82	3	206
10009476	213	365.5	0	F	82	4	281

TABLE 1.5: Reformatted ACE inhibitors dataset for patient ID 10009476.

1.4 Pipeline of the work

In this Section we show and describe the pipeline of the entire work, starting from the raw data up to the validation of the predictive models.

Figure 1.2 shows the pipeline that we followed for our application, which ensues the following steps.

Step 1. Selection of patients survived at least for one year, as described in Section 1.3.

Step 2. Selection of events happened in the observation period. Construction of the four datasets explained in Section 1.3 relative to the four considered stochastic processes (rehospitalisations, purchase of ACE inhibitors, purchase of BB agents and purchase of AA agents) with the features described in Table 1.3.

Step 3. Analysis of the four considered types of recurrent events through the application of the following steps to each of the four datasets built in the previous step.

- (a) Fitting of a Cox model for recurrent events (described in Section 2.1.2). Selection of the best set of features through cross validation.
- (b) Fitting of the cumulative baseline hazard using the Breslow estimator and the smoothing technique proposed in Baraldo et al., 2013. These methodologies are described in Section 2.1.2.
- (c) Reconstruction of the compensators of the counting processes representing the recurrent events, using Equation 2.35 in Section 2.1.2.
- (d) Summary of the fitted compensators trajectories through FPCA in order to obtain a dimensionality reduction of the functional data suitable to be inserted in a predictive model (Section 2.1.3).

The output of this step is a finite set of features (FPCA scores) representing the trajectories of the compensators of each of the four types of processes for each patient.

Step 4. Construction of a new dataset containing the following variables: "FPCA scores", "age", "sex", "survival time" and "censoring indicator". In this dataset, each observation represents one patient.

Step 5. Split of the dataset in 70% training set and 30% test set.

Step 6. Fitting of three predictive survival models: functional Cox-type regression (Section 2.1.4), DeepHit (Section 2.2.3) and DRSA (Section 2.2.5).

Step 7. Comparison of the predictions made by different models on the test set with the techniques described in Section 2.3.

Step 8. Selection of the best predictive model.

We have already shown how we applied steps 1 and 2 of the pipeline on the HF dataset from Regione Lombardia in previous Sections of this Chapter. The results obtained by applying the remaining steps of the pipeline are shown in Sections 3.1 (steps 3(a), 3(b) and 3(c)), 3.2 (step 3(d)), 3.3 (steps 4, 5 and 6) and 3.4 (steps 7 and 8).

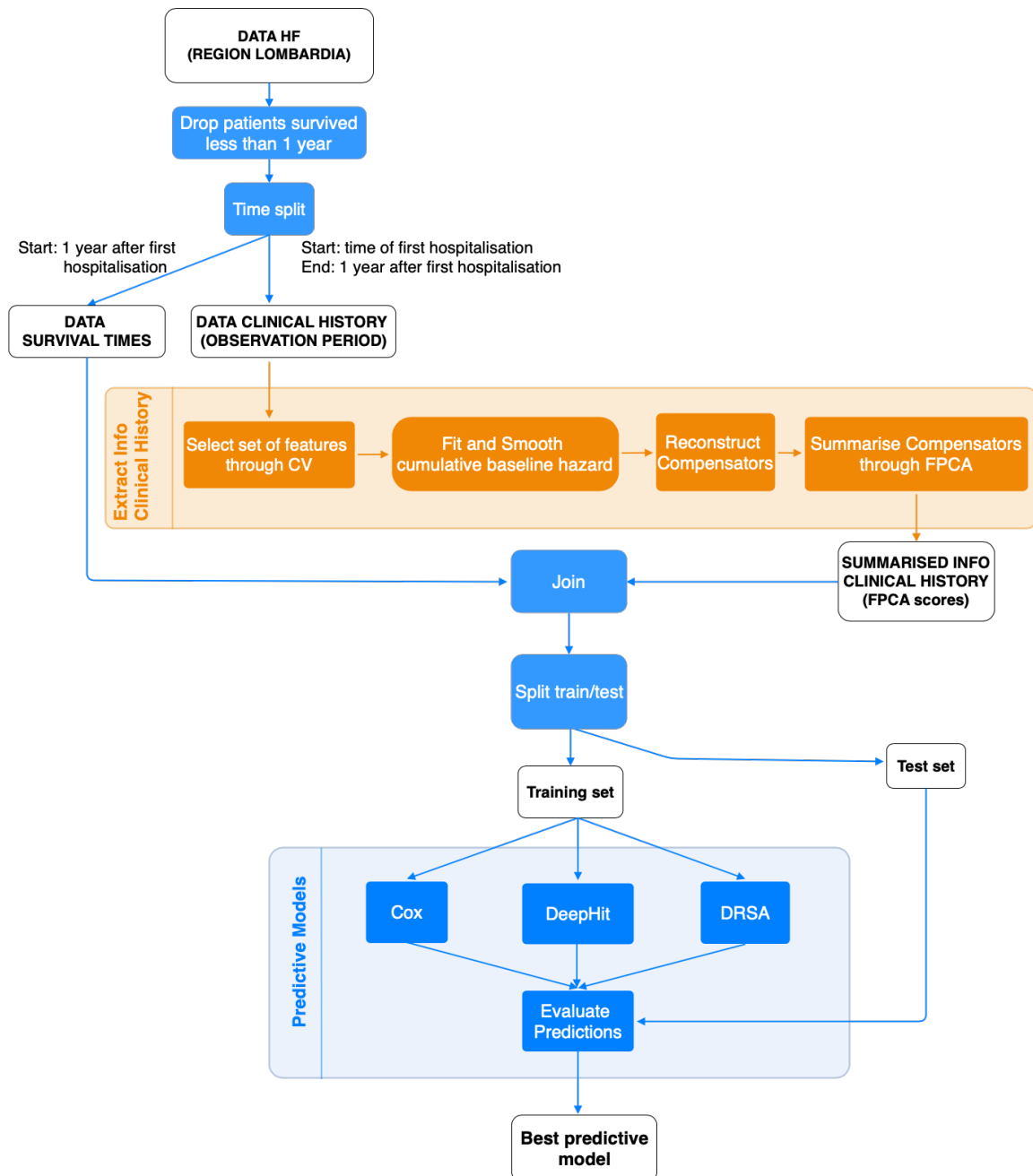


FIGURE 1.2: Pipeline applied on HF Data from Regione Lombardia [HFData-RF-2009-1483329].

Chapter 2

Methodologies

In this Chapter we will go into details of the methodologies adopted and developed to analyse the data and to set up predictive models.

We consider the realisations of the four types of events described in Section 1.3 (HF rehospitalisations, purchase of ACE inhibitors, purchase of BB agents, purchase of AA agents) and occurred in the observation period as time-varying covariates, to be dealt with functional data analysis techniques.

In particular, in Section 2.1 we show a methodology to build up functional data from the realisation of patients' drugs purchases/hospitalisations. Then, dimensionality reduction of such data is performed via Functional Principal Component Analysis, and the scores obtained by this step are introduced in a Cox-type regression model in order to predict patients survival.

In Section 2.2 we introduce two Deep Learning models, namely DeepHit and DRSA, to predict patients survival time in an alternative way.

Finally, in Section 2.3 we compare quantitatively and graphically the goodness of the three predictive survival models.

2.1 Survival analysis via Cox-type regression model with time-varying (functional) covariates

In Section 2.1.1 we give a short introduction to Survival Analysis. In Section 2.1.2 we get functional data from raw information. In Section 2.1.3 we introduce Functional Principal Component Analysis (FPCA). Section 2.1.4 introduces the final model, namely functional Cox-type regression model.

2.1.1 Introduction to Survival Analysis

Survival Analysis (Kleinbaum and Klein, 2013) is a collection of statistical procedures for data analysis for which the outcome variable of interest is the

time until an event occurs.

The aim of this section is to give a very short introduction of the basic concepts of the subject needed for the rest of this work.

Censoring

Most survival analysis problems have to deal with censored data; this is what makes survival analysis problems different from standard regression ones.

In our case, censored data is given by the fact that some patients survived the follow-up period or withdrew before the end of the study. The exact survival time of these subjects is unknown; we only know that it is greater than the censoring time. These observations are called *right-censored*.

Let us formalise this concept. For each patient i , let T_i be the survival time and C_i be the random variable corresponding to censoring time. What we actually observe is:

$$T_i = \min(T_i, C_i) \quad (2.1)$$

In addition, we also know the *status* of the observation, i.e. if T_i is equal to the censoring time or to the survival time. This is usually denoted as:

$$z_i = (T_i, C_i) \quad (2.2)$$

Therefore the observation for patient i corresponds to the pair (T_i, z_i) .

Survival and Hazard functions

Under absolute continuity hypothesis, the distribution of a survival time T can be characterised by a probability density function, like any other continuous random variable. However in Survival Analysis two other equivalent characterisations are often used. These are called the *survival function* and the *hazard function*. The three functions can be used to illustrate different aspects of the data (E. T. Lee and Wang, 2003).

Let T be a continuous random variable, representing the survival time of an individual. Let us assume that the distribution of T is absolutely continuous.

Definition 2.1.1. The **survival function** of T , denoted by $S(t)$, is defined as the probability that an individual survives longer than t

$$\begin{aligned} S(t) &= \text{(an individual survives longer than } t) \\ &= (T > t) \end{aligned} \quad (2.3)$$

By denoting with $F(t)$ the cumulative density function of T we have:

$$S(t) = 1 - F(t) \quad (2.4)$$

$S(t)$ is a nonincreasing function of time t such that:

$$\begin{aligned} S(0) &= 1 \\ \lim_{t \rightarrow \infty} S(t) &= 0 \end{aligned} \quad (2.5)$$

This means that the probability of surviving at least at the time zero is one and that of surviving an infinite time is zero.

The graph of $S(t)$ is called the *survival curve*. A steep survival curve, represents low survival rate or short survival time. A gradual or flat survival curve represents high survival rate or a longer survival time. The survival curve is used to find the median and other percentiles of survival time and to compare survival distributions of two or more groups (E. T. Lee and Wang, 2003).

Definition 2.1.2. The **hazard function** of T , denoted by $l(t)$, is defined as the limit of the probability that an individual fails in a short interval $[t, t + Dt)$, given that the individual has survived to time t :

$$\begin{aligned} l(t) &= \lim_{Dt \rightarrow 0} \frac{\text{P(an individual fails in } [t, t + Dt) \text{]he survived to } t)}{Dt} \\ &= \lim_{Dt \rightarrow 0} \frac{\text{P}(t < T < t + Dt | T > t)}{Dt} \end{aligned} \quad (2.6)$$

It can also be defined in terms of the survival function $S(t)$ and the probability density function $f(t)$ of the survival time T :

$$l(t) = \frac{f(t)}{S(t)} \quad (2.7)$$

The hazard function gives the *conditional failure rate*, i.e. the risk of failure per unit time during the aging process. It is a measure of the proneness to failure as a function of the age of the individual (E. T. Lee and Wang, 2003).

Another quantity that is often involved in survival analysis tasks is the *cumulative hazard function*.

Definition 2.1.3. Let $l(t)$ be the hazard function of the survival time T . The **cumulative hazard function** of T , denoted by $L(t)$, is defined as:

$$L(t) = \int_0^t l(s)ds \quad (2.8)$$

The relation between the cumulative hazard function and the survival function is given in Theorem 2.1.1.

Theorem 2.1.1. Let T be a survival time, $S(t)$ be its survival function and $L(t)$ be its cumulative hazard function. Then:

$$S(t) = e^{-L(t)} \quad (2.9)$$

for any $t \geq 0$.

We would like now to relate the survival time T_i of an individual i with a vector of predictors x_i .

Cox-type regression model

If there is no censored data, any regression method can be applied to model the survival time.

However, it is typical in these types of studies to have censored observations, given for instance by the fact that some subjects did not experience the event before the end of a study or dropped out before the event of interest occurred. Although these may seem to be cases of missing data as the time-to-event is not actually observed, these subjects are highly valuable since the observation that they went a certain amount of time without experiencing an event is itself informative.

One of the most important properties of survival methods is their ability to handle such censored observations which are ignored by standard regression models (Brandon, Seals, and Aban, 2014).

The most widely used survival regression model is the *Cox-type regression model* (Cox et al., 1972). We refer to this model also as *basic Cox model* because it has been extended in several ways, as we will see in Sections 2.1.2 and 2.1.4.

The basic Cox model investigates the relationship of a vector of predictors and the time-to-event through the hazard function. It assumes that the

predictors have a multiplicative effect on the hazard and that this effect is constant over time (Brandon, Seals, and Aban, 2014).

Let \mathbf{x}_i be a vector of covariates for individual i ; the basic Cox model assumes a hazard function for the survival time of the form:

$$l(t/\mathbf{x}_i) = l_0(t)e^{b^T \mathbf{x}_i} \quad (2.10)$$

where $l(t/\mathbf{x}_i)$ is the hazard at time t for a subject with a vector of predictors \mathbf{x}_i , $l_0(t)$ is an unspecified non-negative function called **baseline hazard** and b is the vector of coefficients describing the effect of the predictors on the overall hazard.

Despite the seeming ease at which the basic Cox model can be implemented, there is one significant assumption which must be checked (Brandon, Seals, and Aban, 2014). The basic Cox model assumes proportional hazards between the values of the predictors regardless of how the underlying hazard may change over time.

Indeed, the hazard ratio between two individuals:

$$\frac{l(t/\mathbf{x}_i)}{l(t/\mathbf{x}_j)} = e^{b^T (\mathbf{x}_i - \mathbf{x}_j)} \quad (2.11)$$

is constant over time. This assumption allow us to easily measure the effects of predictors. Indeed, the hazard ratio

$$\begin{aligned} HR_k &= \frac{l(t/x_1, \dots, x_k + 1, \dots, x_n)}{l(t/x_1, \dots, x_k, \dots, x_n)} \\ &= e^{b_k} \end{aligned} \quad (2.12)$$

denotes the hazard increase per unit of x_k . For instance, if $e^{b_k} = 1.1$, the increase of one unit of x_k leads to a 10% greater hazard function.

The proportionality assumption reflects on the survival function as follows. Using Theorem 2.1.1, the survival function for the basic Cox model can be written as:

$$\begin{aligned} S(t/\mathbf{x}_i) &= \exp \left\{ - \int_0^t l(s/\mathbf{x}_i) ds \right\} \\ &= \exp \left\{ - \int_0^t l_0(s) ds \right\} e^{-b^T \mathbf{x}_i t} \\ &= S_0(t)^{c_i} \end{aligned} \quad (2.13)$$

where $S_0(t) = \exp\left(-\int_0^t l_0(s) ds\right)$ and $c_i = e^{b^T x_i}$. Therefore, for the basic Cox model the survival curves are equal to a baseline function $S_0(t)$ to the power of a constant determined by the vector of predictors.

The proportionality hypothesis is often violated and it could lead to biased estimates of the underlying model (Moolgavkar et al., 2018).

In Section 2.1.4 we will introduce our extension of the basic Cox model designed for handling functional predictors. Also this model involves the proportionality assumption. The strictness of this hypothesis is the main reason that led us to consider more flexible models.

Alternative models not involving this assumption are, for instance, the two deep learning models DeepHit (C. Lee et al., 2018) and DRSA (Ren et al., 2018), described in Section 2.2. In Section 3.3 we will use all these three models (functional Cox-type regression, DeepHit and DRSA) to predict survival time of patients, and we will compare performances using techniques described in Section 2.3.

The basic Cox model is referred to as semiparametric model. This is due to the lack of a parametric form for the baseline hazard $l_0(t)$. This fact gives flexibility to the model since it does not constrain the baseline hazard to a given distribution, but it introduces a difficulty into the estimation procedure of the b coefficients. Indeed, standard estimation methods like maximum likelihood estimation cannot be applied.

To overcome this problem, Cox proposed a partial likelihood function, based on a conditional probability of failure (Cox et al., 1972).

Let T_i be the possibly censored observed survival time for the i th individual and let z_i be its censoring indicator (1 meaning uncensored).

Definition 2.1.4. The **partial likelihood** of the coefficients b takes the form:

$$PL(b) = \prod_{i=1}^N \frac{e^{b^T x_i} z_i}{\sum_{l \in \mathbf{R}(T_i)} e^{b^T x_l}} \quad (2.14)$$

where $\mathbf{R}(t) = \{i : T_i \geq t\}$ denotes the set of individuals whose survival time are at least t and it is called the *risk set*.

It follows that the estimate of the coefficients proposed by Cox is given by:

$$\hat{b} = \underset{b \in \mathbb{R}^n}{\operatorname{argmax}} PL(b) \quad (2.15)$$

The partial likelihood is typically maximised by applying the Newton-Raphson iterative procedure (E. T. Lee and Wang, 2003).

Another useful function for describing the underlying stochastic process is the **cumulative baseline hazard** function, defined as:

$$L_0(t) = \int_0^t l_0(s) ds \quad (2.16)$$

Typically, the non-parametric estimators of the basic Cox model do not directly estimate the baseline hazard, but rather the cumulative baseline hazard. The most common estimator for the baseline cumulative hazard is the nonparameteric *Breslow estimator* (Breslow, 1975).

Definition 2.1.5. The **Breslow estimator** for the baseline cumulative hazard is defined as:

$$\hat{L}_0(t) = \hat{a} \frac{z_i}{\sum_{j: T_j \leq t} \hat{a}_j 2\mathbf{R}(T_j) e^{\hat{b}^T \mathbf{x}_j}} \quad (2.17)$$

where \hat{b} is the maximum partial likelihood estimator of b .

The two estimators \hat{b} and \hat{L}_0 are consistent, asymptotically normal and asymptotically efficient (Lin, 2007).

Observe that, thanks to Theorem 2.1.1, we can estimate the survival functions $S_i(t)$ for a basic Cox model as:

$$\hat{S}_i(t) = e^{-\hat{L}_i(t)} = e^{-\hat{L}_0(t) e^{\hat{b}^T \mathbf{x}_i}} \quad (2.18)$$

The basic Cox model characterises a survival time, or more in general a "time-to-event" random variable. Therefore, it does not allow for modelling recurrent events settings. Furthermore, its predictors are assumed to be constants in time. However, this is not the case in several situations; for instance in longitudinal studies the value of some time-varying variables is often collected at each observation of the same individual.

Within our analysis of the recurrent events of patients, we will not use the basic Cox model, but we will take advantage of an extension described in Section 2.1.2, which allows to include time dependent covariates and to handle recurrent events.

2.1.2 Cox model for recurrent events

The basic Cox model has been extended to handle recurrent events and time dependent variables through the counting process formulation given in Andersen and Gill, 1982.

To understand this new formulation, we briefly introduce the fundamental *Doob-Meyer decomposition* and the notion of *counting process* in the next sections.

Doob-Meyer decomposition

In order to introduce the Doob-Meyer decomposition, we need some preliminary definitions. We introduce the concepts of *filtration*, *stochastic basis* and *adapted process* which are necessary to handle stochastic processes rigorously. After that, we give the definition of a *stopping time* in order to introduce one of the processes that are involved in the decomposition, the *predictable process*. Finally, we define the two other types of processes which are involved: the *martingale* and the *submartingale*.

Let's fix a continuous time interval:

$$T = [0, t] \text{ or } [0, t) \quad (2.19)$$

with $0 < t < \infty$.

Definition 2.1.6. Let (W, F, P) be a probability space. A **filtration**:

$$(F_t, t \in T) \quad (2.20)$$

is an increasing family of sub- σ -algebras of F .

Definition 2.1.7. A complete probability space (W, F, P) with a filtration $(F_t, t \in T)$, which satisfies the following conditions $\forall t \in T$:

1. continuity from the right: $F_t = F_{t+}$
 2. completeness: F_t contains all subsets of sets $A \in F$ with P -measure 0
- is called a **stochastic basis** and it is denoted by $(W, F, \mathcal{F}_t; t \in T, P)$.

Given the definitions above, let X be a stochastic process, i.e. a time-indexed collection of random variables $(X(t) : t \in T)$. We can define an *adapted process* and a *stopping time* as follows.

Definition 2.1.8. The process X is called **adapted** to the filtration $(F_t, t \geq T)$ if $X(t)$ is F_t -measurable for each t .

Definition 2.1.9. A **stopping time** T is a random variable taking values in $\bar{T} = [0, t]$ such that:

$$t \geq T \implies t \geq T \text{ and } t \geq T \text{ for } t \geq T \quad (2.21)$$

Intuitively, the time T of a random event is a stopping time if at each fixed time t one can observe whether or not the event has already occurred.

We can now define the three types of processes which are involved in the Doob-Meyer decomposition: *predictable*, *martingale* and *submartingale*.

Definition 2.1.10. The process A is called **predictable** if $A(T)$ is F_T -measurable for all stopping times T .

The previous definition implies that the value of a predictable process at time T is fixed "just before" the time itself.

Definition 2.1.11. Let X be a right-continuous stochastic process with left-hand limits and $(F_t, t \geq T)$ a filtration, defined on a common probability space. Then X is called a **martingale** with respect to the filtration if:

1. X is adapted to the filtration
2. $E[|X(t)|] < \infty, \forall t \geq T$
3. $E[X(t+s)|F_t] = X(t)$ a.s. $\forall s \geq 0, t \geq T$

X is called a **submartingale** if property 3 is replaced by $E[X(t+s)|F_t] \geq X(t)$ a.s..

Given the previous definitions, we are ready to understand the Doob-Meyer decomposition theorem. Note that this version can be extended to hold for a wider class of processes through the idea of *localisation* (see for instance Andersen, Borgan, et al., 2012), but this goes beyond the scope of this thesis.

Theorem 2.1.2 (Doob-Meyer decomposition). *Let X be a right-continuous non-negative submartingale with respect to a stochastic basis $(W, F, \mathbb{F}_{F_t}; t \geq 0, P)$. Then there exists a right-continuous martingale M and an increasing right-continuous predictable process A such that $E[A(t)] < \infty$ and*

$$X(t) = M(t) + A(t) \text{ a.s.} \quad (2.22)$$

for any $t \geq 0$. If $A(0) = 0$ a. s., and if $X = M^0 + A^0$ is another such decomposition with $A^0(0) = 0$, then for any $t \geq 0$,

$$PfM^0(t) \ominus M(t)g = 0 = PfA^0(t) \ominus A(t)g. \quad (2.23)$$

Definition 2.1.12. The process A in Theorem 2.1.2 is called the **compensator** for the submartingale X .

Theorem 2.1.2 says that, under certain conditions, we can split a stochastic process in a random and in a systematic part. This is exactly what we want to exploit for the "clinical history" of the patients. In other words, we would like to include the main information of the stochastic process occurring in the observation period as time-varying covariates to be introduced into the functional Cox-type regression model after suitable dimensionality reduction.

Counting processes

We will now focus on a particular type of stochastic processes, namely counting processes.

Definition 2.1.13. A **counting process** is a stochastic process N adapted to a filtration $(F_t, t \geq T)$ with $N(0) = 0$ and $N(t) < \infty$ a.s., and whose paths are with probability one right-continuous, piecewise constant, and have only jump discontinuities, with jumps of size $+1$.

The hypothesis of Theorem 2.1.2 hold for this class of stochastic processes. Therefore, for any counting process N with finite expectation, there is a "unique" predictable process L so that $N = L + M$, M being a martingale, and therefore:

$$M = N - L \quad (2.24)$$

It is often useful to reformulate the Equation 2.24 in terms of the *intensity process* of the counting process.

Definition 2.1.14. Let N be a counting process adapted to a filtration $\mathcal{F}_{t,t}$. We call **intensity process** of N the stochastic process:

$$l(t) = \lim_{h \neq 0} \frac{1}{h} \mathbb{E}[N(t+h) - N(t) | \mathcal{F}_t] \quad (2.25)$$

Observe that the right-continuity property of counting processes ensures the existence of this limit from the right. The relation between the intensity process l and the predictable process L is given by:

$$L(t) = \int_0^t l(s) ds \quad (2.26)$$

Therefore we can rewrite Equation 2.24 as:

$$M(t) = N(t) - \int_0^t l(s) ds \quad (2.27)$$

Within our dataset we consider purchases of drugs and hospitalisations as realisation of counting processes, in which event times correspond to jump times of the processes. As we saw in Chapter 1, we also have a "mark" variable associated with each event, which denotes the duration of the prescriptions and the time spent at the hospital for the purchases of drugs and the hospitalisations, respectively. We would like to include marks in the formulation of the model through the notion of *marked point process*.

Marked point processes

In this Section we give a brief and informal introduction to marked point processes.

Sometimes, as in our case, we can associate a *jump mark* to each *jump time* of a counting process N . A typical example is given by earthquakes modelling, where to each earthquake we can associate its intensity (Harte et al., 2010).

these kind of processes are essentially known as *marked point processes* (Harte et al., 2010) and they are modelled through the *conditional intensity function* which represents the expected rate of events at time t with mark \mathbf{m} . Marked point processes can be seen as an extension of counting processes and the conditional intensity function represents the extension of the intensity function for counting processes.

It is typically assumed that the conditional intensity function has the form:

$$l(t, \mathbf{m} | F_t) = l_g(t | F_t) f(\mathbf{m} | F_t) \quad (2.28)$$

where F_t is the filtration of the process, l_g is the *intensity process* of the counting process and f is the multivariate density of the mark \mathbf{m} .

Typically F_t is interpreted as the history of realisations of the process. Note that, assuming the conditional independence of jump times and marks, the maximisation of the likelihood of the two parts can be done separately. Often the more restrictive hypothesis of independence from the past or dependence on time only are made for the mark distribution $f(\mathbf{m} | F_t)$ (Mancini and Paganoni, 2019), but it will not be necessary for our case, since we are interested in modelling only the process $l_g(t | F_t)$. This is the intensity of a counting process which may depend on the realisations of both events and marks.

We will consider recurrent events occurred in the observation year as trajectories of marked point process, and we will model their counting process part with a *Cox model for recurrent events*, which we describe in the following Section.

Counting process formulation of the Cox model for recurrent events

We are now ready to introduce the counting process formulation of the Cox model for recurrent events, which allows predictors to change in time and to handle recurrent events. This formulation exploits the Doob-Meyer decomposition for counting processes in the form given by Equation 2.27.

Let us consider a set of n individuals with possibly censored observations of multiple events. We can see these observations as the realisation of an n -component multivariate counting process (N_1, \dots, N_n) where N_i is the stochastic process which counts the observed events in the life of the i th individual.

We already saw that for each of these counting processes there is a unique decomposition:

$$M_i(t) = N_i(t) - \int_0^t l_i(s) ds \quad (2.29)$$

where M_i is a martingale and $L_i(t) = \int_0^t l_i(s) ds$ is a predictable increasing process.

The basic assumption made in Andersen and Gill, 1982 is that the random intensity process $l_i(t)$ has the form:

$$l_i(t) = Y_i(t) l_0(t) e^{b^T x_i(t)} \quad (2.30)$$

where b is a fixed vector of coefficients, x_i is the possibly time-dependent vector of covariates of the i th individual, l_0 a fixed underlying hazard function and Y_i is a predictable process taking values in $\{0, 1\}$ which set (by the value 1) when the i th individual is under observations. Note that Y_i takes the role of the censoring variable and that we still can define the *cumulative baseline hazard* as $L_0(t) = \int_0^t l_0(t) dt$.

With this new formulation the authors not only extended the class of Cox models, but also introduced two consistent estimators of the parameters b and of the cumulative baseline hazard L_0 analogous to the ones introduced in Section 2.1.1.

Again, the estimation of the parameters b is based on a partial likelihood function, based on a conditional probability of failure.

Definition 2.1.15. Let $T_i, i = 1, \dots, N$ be the observed event times and let us assume that $T_1 < \dots < T_N$. Then, the **partial likelihood** of the coefficients b takes the form:

$$PL(b) = \prod_{i=1}^N \frac{e^{b^T x_i(t)}^{DN_i(t)}}{\sum_{j=1}^N Y_j(t) e^{b^T x_j(t)}} \quad (2.31)$$

where $DN_i(t) = N_i(t) - N_i(t^-)$.

It follows that the estimate of the coefficients is given by:

$$\hat{b} = \underset{b \in \mathbb{R}^n}{\operatorname{argmax}} PL(b) \quad (2.32)$$

The partial likelihood is typically maximised by applying the Newton-Raphson iterative procedure (E. T. Lee and Wang, 2003).

The estimator for the baseline cumulative hazard L_0 is called again *Breslow estimator*.

Definition 2.1.16. The **Breslow estimator** for the baseline cumulative hazard is defined as:

$$\hat{L}_0(t) = \sum_{i=1}^N \int_0^t \frac{dN_i(u)}{\sum_{j=1}^N Y_j(u) e^{\hat{b}^T x_j(u)}} \quad (2.33)$$

where \hat{b} is the maximum partial likelihood estimator of b .

The two estimators \hat{b} and \hat{L}_0 are consistent, asymptotically normal and asymptotically efficient (Lin, 2007).

Observe that the Breslow estimator returns a step-function. However, since we are assuming that $L_0(t) = \int_0^t l_0(t)$, the true underlying L_0 function is absolutely continuous. A possible way to deal with this problem is to smooth the estimate given by the Breslow estimator. We will use the approach adopted in Baraldo et al., 2013 in which the authors used a constrained smoothing technique proposed in He and Ng, 1999, enforcing the smooth estimate of \hat{L}_0 to keep monotonicity and $\hat{L}_0(0) = 0$. The coefficients of the basis are estimated by minimising the distance induced by $\| \cdot \|_1$ of the smoothed evaluations, thus obtaining the desired regularised curve \tilde{L}_0 .

As we mentioned in Section 1.4, in our analysis we are interested in reconstructing the compensators of four (marked) counting processes, which represent the clinical history of patients. We will consider as possible predictors the variables described in Section 1.3 - "age", "sex", "number of previous events" and "sum of past marks" - and their interactions. The values of these features is piecewise constant, and they change only in correspondance of a new event for the patient, i.e. of a "jump time" of the counting process.

The compensator in this case can be written in a particular form (Baraldo et al., 2013). Let $N_i(t)$ be the counting process of individual i , and let $t_1^i, \dots, t_{N_i(t)}^i$ be its realised jump times, with t equal to the censoring time. In our case the censoring time is given by the end of the observation period, set to one year; therefore we can coherently set the censoring time t equal for all the individuals. Let $t_0^i = 0$ for all i . Then, we can write the compensator of individual i as follows:

$$\begin{aligned}
L_i(t) &= \int_0^t l_i(s) ds \\
&= \int_0^t l_0(s) e^{b^T x_i(s)} ds \\
&= \mathring{\mathbf{a}} \sum_{k=1}^{N_i(t)} \int_{t_{k-1}^i}^{\min(t_k^i, t)} l_0(s) e^{b^T x_i(t_{k-1}^i)} ds \\
&= \mathring{\mathbf{a}} \sum_{k=1}^{N_i(t)} e^{b^T x_i(t_{k-1}^i)} \int_{t_{k-1}^i}^{\min(t_k^i, t)} l_0(s) ds \\
&= \mathring{\mathbf{a}} \sum_{k=1}^{N_i(t)} e^{b^T x_i(t_{k-1}^i)} [L_0(\min(t_k^i, t)) - L_0(t_{k-1}^i)]
\end{aligned} \tag{2.34}$$

Therefore, we can express the realisations of each compensator L_i as a function of L_0 and b . An estimate of the compensator (Baraldo et al., 2013) can be obtained as:

$$\hat{L}_i(t) = \hat{\mathbf{a}} \sum_{k=1}^{N_i(t)} e^{\hat{b}^T x_i(t_{k-1}^i)} [\tilde{L}_0(\min(t_k^i, t)) - \tilde{L}_0(t_{k-1}^i)] \tag{2.35}$$

where \hat{b} is the estimated vector of coefficients and \tilde{L}_0 is the smoothed estimate of the cumulative baseline hazard.

In Chapter 1 we described the variables Nm and y which represent the number of past events and the sum of the past marks. In Section 3.1 we will consider these features, the age and the sex (*age_in* and *sex*) of the patients and we will model the cumulative intensity processes $L_i(t)$ of the counting processes representing their clinical history.

We will estimate the coefficients \hat{b} and the baseline hazard \hat{L}_0 for the intensity using the Partial likelihood estimator and the Breslow estimator, respectively. Then we will fit a smoothed version of \hat{L}_0 using the previously described technique (Baraldo et al., 2013) and we will exploit Equation 2.35 to obtain the estimated compensators \hat{L}_i of the counting processes for each patient.

Therefore, we will obtain a set of four functions ($\hat{L}_i^{(k)}(t)$, $k \in \{fACE, BB, AA, rehosp.g\}$) associated with each patient, corresponding to the compensators of the four stochastic processes of interest.

In the next Section we will see how to deal with and synthesize the trajectories of the fitted compensators in order to introduce them in predictive

models.

2.1.3 Functional Principal Component Analysis (FPCA)

In the previous Sections we showed how to extract a set of informative functions from the realisations of counting processes representing the clinical history of patients. In this Section we describe how to use *Functional Data Analysis* (FDA) methods to summarise each function to a finite set of covariates while loosing a minimum part of the information.

FDA (Ramsay and Silverman, 2013) is the branch of statistics dealing with *functional data*. We call *functional data* a sample where each observation is considered to be a function varying over a continuous domain, typically a time or space interval.

We will briefly introduce *Functional Principal Component Analysis* (FPCA), which is a fundamental tool of FDA.

Let $\{x_1, \dots, x_N\}$ be a sample of functions such that $x_i : S \rightarrow \mathbb{R}$ $i = 1, \dots, N$, where S is a continuum. We will set our analysis considering $\{x_1, \dots, x_N\}$ to be the realisations of the compensators for a specific counting process, $\{1, \dots, N\}$ to be the set of patients and S to be the one year observation period.

Let us assume that we already removed the temporal mean from the original functions, i.e.:

$$\int_0^1 x_i(s) ds = 0 \quad \forall i = 1, \dots, N \quad (2.36)$$

FPCA may be interpreted as follows: assume we want to find a set of K orthonormal functions x_1, \dots, x_K such that the expansion of each curve in terms of these basis functions is "as close as possible to the original curves". This may be pursued solving the following optimisation problem:

$$\begin{aligned} \min_{x_1, \dots, x_K} \quad & \sum_{i=1}^N \int_0^1 [x_i(s) - \hat{x}_i(s)]^2 ds \\ \text{s.t.} \quad & \int_0^1 x_i(s) x_k(s) ds = 0 \quad i \neq j, \quad i, j = 1, \dots, K \\ & \int_0^1 x_k(s) ds = 1 \quad k = 1, \dots, K \\ & \hat{x}_i = \sum_{k=1}^K f_{ik} x_k \quad i = 1, \dots, N \end{aligned} \quad (2.37)$$

The goal of solving this problem is to find the closest K -dimensional representation of the original functions using the terms f_{ik} , which are called *functional principal component scores*. For our application, this translates in finding a set of K features representing the compensators of the counting processes, and therefore the clinical history in the observation period of patients. This will allow us to introduce the information of compensators in survival models as standard covariates.

For a deeper tractation of FPCA, see for instance Ramsay and Silverman, 2013.

A question that naturally arises is how to choose the dimension K of the basis. In standard principal component analysis (PCA) we usually choose the minimum number such that the percentage of variance explained by the PCA representation is above a given threshold. With FPCA we can do something similar. Let us define the *total variation* of the sample x_1, \dots, x_N as:

$$N^{-1} \sum_{i=1}^N \|x_i\|^2 \quad (2.38)$$

and the *explained variation* of the first K principal components as:

$$N^{-1} \sum_{i=1}^N \|\hat{x}_i^{(K)}\|^2 \quad (2.39)$$

where $\hat{x}_i^{(K)}$ is the projection of x_i on the K -dimensional principal component basis. Then, we can choose the number of components such that the ratio between the explained variation and the total variation goes above a given threshold.

Using FPCA in Section 3.2 we will find a finite set of features that represent and summarise our original functional data.

We would like to use a model that incorporates functional data from a theoretical point of view, so as to be interpretable. The basic Cox model described in Section 2.1.1 has been generalised to handle functional data. We describe this extended version in the next Section.

2.1.4 Functional Cox-type regression model

In this Section we describe the *functional Cox-type regression model*, i.e. the Cox-type regression model extended so as to deal with functional covariates.

Let us consider a set of individuals and let \mathbf{x}_i be the vector of predictors for the individual i . Let us assume that each individual i also has a functional predictor $z_i(s)$ taking value in a domain S .

In this case, the functional Cox-type regression model takes the following form:

$$l(t|\mathbf{x}_i, z_i) = l_0(t) \exp\{b^T \mathbf{x}_i + \int_S z_i(s) g(s) ds\} \quad (2.40)$$

where, in addition to the usual baseline hazard l_0 and the vector of coefficients b there is a functional coefficient g .

Through FPCA we can approximate the functional covariates with a finite sum of K orthonormal basis:

$$z_i(s) = \sum_{k=1}^K z_{ik} x_k(s) \quad (2.41)$$

where z_{ik} is the FPCA score of individual i for the k th orthonormal base, as in 2.37. Therefore, we can approximate the integral in Equation 2.40 as:

$$\begin{aligned} \int_S z_i(s) g(s) ds &= \int_S \sum_{k=1}^K z_{ik} x_k(s) g(s) ds \\ &= \sum_{k=1}^K z_{ik} \int_S x_k(s) g(s) ds \\ &= \sum_{k=1}^K z_{ik} g_k \end{aligned} \quad (2.42)$$

where g_k is the scalar representing the quantity $\int_S x_k(s) g(s) ds$.

Introducing the approximation 2.42 in Equation 2.40 we obtain:

$$l(t|\mathbf{x}_i, z_i) = l_0(t) \exp\{b^T \mathbf{x}_i + \sum_{k=1}^K z_{ik} g_k\} \quad (2.43)$$

We observe that, using the FPCA approximation, the model described by Equation 2.40 is equivalent to the basic Cox model where the principal components are treated as standard covariates.

The extension to the case with multiple functional covariates is trivial. In fact, let us consider the case in which to each individual i it corresponds a vector of features \mathbf{x}_i and a set of m functions $z_i^{(1)}, \dots, z_i^{(m)}$. The model described by Equation 2.40 takes the following form:

$$l(t|\mathbf{x}_i, z_i^{(1)}, \dots, z_i^{(m)}) = l_0(t) \exp \{ \mathbf{b}^T \mathbf{x}_i + \mathop{\mathring{\mathbf{a}}}\limits_{j=1}^m \int_S z_i^{(j)}(s) g^{(j)}(s) ds \} \quad (2.44)$$

Similarly to Equation 2.41 we can approximate each functional covariate with a finite sum of K orthonormal basis:

$$z_i^{(j)}(s) = \mathop{\mathring{\mathbf{a}}}\limits_{k=1}^{K_j} z_{ik}^{(j)} x_k^{(j)}(s) \quad (2.45)$$

We can then approximate the integrals in Equation 2.44 as:

$$\begin{aligned} \int_S z_i^{(j)}(s) g^{(j)}(s) ds &= \int_S \mathop{\mathring{\mathbf{a}}}\limits_{k=1}^{K_j} z_{ik}^{(j)} x_k^{(j)}(s) g^{(j)}(s) ds \\ &= \mathop{\mathring{\mathbf{a}}}\limits_{k=1}^{K_j} z_{ik}^{(j)} \int_S x_k^{(j)}(s) g^{(j)}(s) ds \\ &= \mathop{\mathring{\mathbf{a}}}\limits_{k=1}^{K_j} z_{ik}^{(j)} g_k^{(j)} \end{aligned} \quad (2.46)$$

where $g_k^{(j)}$ is the scalar representing the quantity $\int_S x_k^{(j)}(s) g^{(j)}(s) ds$.

Introducing the approximation 2.46 in Equation 2.44, we can express the hazard as:

$$l(t|\mathbf{x}_i, z_i^{(1)}, \dots, z_i^{(m)}) = l_0(t) \exp \{ \mathbf{b}^T \mathbf{x}_i + \mathop{\mathring{\mathbf{a}}}\limits_{j=1}^m \mathop{\mathring{\mathbf{a}}}\limits_{k=1}^{K_j} z_{ik}^{(j)} g_k^{(j)} \} \quad (2.47)$$

Defining the following quantities:

$$\begin{aligned} \mathbf{w} &= [\mathbf{b}^T, [g_1^{(1)}, \dots, g_{K_1}^{(1)}]^T, \dots, [g_1^{(m)}, \dots, g_{K_m}^{(m)}]^T]^T \\ \mathbf{y}_i &= [\mathbf{x}_i^T, [z_{i,1}^{(1)}, \dots, z_{i,K_1}^{(1)}]^T, \dots, [z_{i,1}^{(m)}, \dots, z_{i,K_m}^{(m)}]^T]^T \end{aligned} \quad (2.48)$$

and substituting them in Equation 2.47 we can express the functional Cox-type regression model with the FPCA approximation as a basic Cox model:

$$l_i(t) = l_0(t) \exp \{ \mathbf{w}^T \mathbf{y}_i \} \quad (2.49)$$

Using the FPCA approximation in a functional Cox-type regression model, all the properties and estimators of the basic Cox model shown in Section

2.1.1 still hold. Therefore we can estimate the vector of coefficients \mathbf{w} by maximising the Partial likelihood function, and we can reconstruct the cumulative hazard function L_0 by using the Breslow estimator. Then, using the estimates $\hat{\mathbf{w}}$ and \hat{L}_0 we can reconstruct the survival curves as:

$$\hat{S}_i(t) = e^{-\hat{L}_0(t)e^{\hat{\mathbf{w}}^T y_i} \quad (2.50)$$

In Section 3.3 we will use the functional Cox-type regression model to predict survival time of patients and to reconstruct their survival curves.

The proportionality assumption still holds for the functional Cox-type regression model. In Section 2.2 we describe two models that do not make this assumption.

2.2 Survival Analysis via a Machine Learning approach

In Section 2.1 we have illustrated a methodology to extract and to reduce dimensionality of informative functional data from the "clinical history" of the patients, and to use this summarised information for predicting survival time of patients in a functional Cox-type regression model.

In order not to assume the restrictive hypothesis of proportionality made by the functional Cox-type regression model, we consider two models enabling estimates of survival that do not make any assumption on the underlying stochastic process, namely the two Deep Learning models *DeepHit* and *DRSA*. These models claim to be the state-of-the-art in terms of goodness of predictions (C. Lee et al., 2018 and Ren et al., 2018).

As a drawback, these models do not allow us to give a theoretical justification to the use of FPCA scores and to interpret the effect of the predictors as for the functional Cox-type regression model; on the other hand, they may be smarter in catching the real behaviour of risks in our dataset.

In Section 2.2.1 we introduce some preliminary Deep Learning concepts. In Section 2.2.2 we introduce the Feedforward Neural Network, that is the network used by the *DeepHit* model described in Section 2.2.3. We then introduce in Section 2.2.4 the Long-Short Term Memory Neural Network, which is used by the *DRSA* model described in Section 2.2.5.

2.2.1 Introduction to Deep Learning

In this Section we give a brief introduction to Deep Learning, only for what concerns this thesis work. For a complete explanation and overview, see for instance Goodfellow, Bengio, and Courville, 2016.

Deep learning is a field of machine learning based on the usage of artificial Neural Networks (NN), which are computing systems inspired by the biological neural networks that constitute animal brains.

A NN is based on a collection of connected units or nodes called artificial neurons. A standard artificial neuron is a function $a : \mathbb{R}^n \rightarrow \mathbb{R}$ which is a composition of a nonlinear function and a linear functional:

$$a_{\mathbf{w},b}(\mathbf{x}) = f(b + \mathbf{w}^T \mathbf{x}) \quad (2.51)$$

A representation of an artificial neuron is given in Figure 2.1.

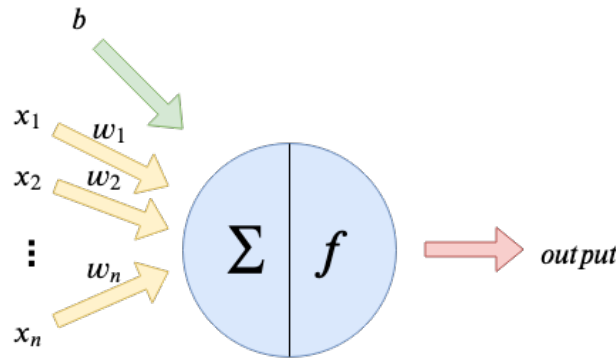


FIGURE 2.1: Representation of an Artificial Neuron.

Usually nodes in a NN are divided into groups, with each node of the same group receiving the same input vector. Let us consider a group of m artificial neurons $a_{\mathbf{w}_1, b_1}^{(1)}, \dots, a_{\mathbf{w}_m, b_m}^{(m)}$, such that $a_{\mathbf{w}_i, b_i}^{(i)} : \mathbb{R}^n \rightarrow \mathbb{R} \forall i = 1, \dots, m$. As a result, the group of the artificial neurons can be seen as a function $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$:

$$g(\mathbf{x}) = [a_{\mathbf{w}_1, b_1}^{(1)}(\mathbf{x}), \dots, a_{\mathbf{w}_m, b_m}^{(m)}(\mathbf{x})]^T \quad (2.52)$$

Observe that the function g consists in a composition of an affine transformation A and a non-linear function f . The affine transformation $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is given by:

$$\begin{aligned} A(\mathbf{x}) &= [\mathbf{w}_1^T \mathbf{x} + b_1, \dots, \mathbf{w}_m^T \mathbf{x} + b_m]^T \\ &= \mathbf{W}\mathbf{x} + \mathbf{b} \end{aligned} \quad (2.53)$$

where $W = [\mathbf{w}_1, \dots, \mathbf{w}_m]^T$ and $\mathbf{b} = [b_1, \dots, b_m]^T$.

The non-linear function $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$, which takes in input the output $\mathbf{z} = A(\mathbf{x})$ of the affine transformation A , is given by:

$$f(\mathbf{z}) = [f_1(z_1), \dots, f_m(z_m)]^T \quad (2.54)$$

where f_i is the non-linear function of the i th artificial neuron.

The function f is referred to as *activation function*. The most commonly used activation functions are:

$$\begin{aligned} \text{ReLU}(\mathbf{z})_i &= \max(0, z_i) \\ \text{Tanh}(\mathbf{z})_i &= \frac{e^{z_i} - e^{-z_i}}{e^{z_i} + e^{-z_i}} \\ \text{Sigmoid}(\mathbf{z})_i \text{ or } S(\mathbf{z})_i &= \frac{1}{1 + e^{-z_i}} \\ \text{SoftMax}(\mathbf{z})_i &= \frac{e^{z_i}}{\sum_j e^{z_j}} \end{aligned} \quad (2.55)$$

Observe that the *SoftMax* activation function normalises the output. Therefore, since the i th value of the output does not depend only on the i th value of the input, it represents a small extension of Equation 2.54. Since the output of this function sums up to 1, it is often used to represent probability measures.

Connections between artificial neurons can transmit a signal from one node to another; each node that receives a signal can process it and then transmit it to other connected nodes. Therefore, the vector representing the output of a group becomes the input of the groups connected to it.

We can see a NN as a sequence of high-level building blocks, namely *layers*. A layer can be seen as a function that usually receives weighted input, transforms it with a set of mostly non-linear functions and then passes these values as output to the next layer (Mihajlovic, 2018).

Each layer "contains" artificial neurons, or nodes, and the nodes of one layer are connected to those of the next.

The simplest possible layer is the one given by Equation 2.52 and it is called *fully connected layer*. It is the one used in Feedforward NNs, which we will explain in Section 2.2.2. More complex layers are used, for instance, in Long-Short Term Memory NNs, which we will explain in Section 2.2.4.

Within this thesis we will consider only a supervised framework: the role of the NN will be to predict some unknown output given a known input. The

parameters of the NN are given by the weights in linear transformations (i.e. weights of connections) and, after initialisation, they are updated through an iterative procedure. During this training phase, the quality of the network is measured through a *loss function* L . The goal is to find an "optimal" set of parameters so as to minimise the loss function.

The parameters are usually initialised at random and updated iteratively through a gradient-based optimisation algorithm, e.g. the Adam method (Kingma and Ba, 2014). Let $\mathbf{w} = [w_1, \dots, w_K]^T$ contain all the weights of the NN. To apply a gradient-based optimisation algorithm, we need to evaluate the gradient of the loss function with respect to each weight:

$$\nabla_{\mathbf{w}} L = \left[\frac{\partial L}{\partial w_1}, \dots, \frac{\partial L}{\partial w_K} \right]^T \quad (2.56)$$

The computation can be done with the *back-propagation* algorithm (LeCun et al., 2012), which is based on the chain rule of calculus. It gets its name because the weights are updated backwards, from output towards input.

In theory, we could use the entire training set to compute the loss function at each iteration step of the optimisation algorithm. However it is common practice to compute the gradient of the loss function at each iteration only for a fixed number of samples. This indeed shortens training time and decreases computational needs (Goodfellow, Bengio, and Courville, 2016). The sample used to compute the loss is called *batch*. An *epoch* corresponds to the entire training set used to update parameters once.

There exist many different NN architectures (Goodfellow, Bengio, and Courville, 2016). In our work, we will focus only on two particular ones, namely the Feedforward Neural Network (FNN) and the Long-Short Term Memory Neural Network (LSTMNN), with the aim of comparing their ability in predicting survival time with standard methods deriving from Cox-type regression models. The application of Deep Learning methods to Survival Analysis without making assumptions on the underlying probability distribution is a relatively new field of application, mentioned only in recent articles like Katzman et al., 2018, C. Lee et al., 2018 and Ren et al., 2018.

A typical problem with studies based on NNs is that NN models are likely to overfit training data due to the high number of parameters they are characterised by. To prevent overfitting, it is often useful to use a method called *early stopping* (Yao, Rosasco, and Caponnetto, 2007), which consists in using a

fraction of training data as validation set and interrupting the training when the loss of the validation set stops decreasing. We then keep the parameters corresponding to the best loss for the validation set.

Another technique to prevent overfitting is called *dropout* (Srivastava et al., 2014) and consists in randomly removing nodes and their connections at each training stage, so that we are left with a reduced network.

Given these notions, we are now able to introduce two families of models which might be intended as competitors of the model proposed in Section 2.1.4.

2.2.2 Feedforward Neural Network (FNN)

As we stated above, a Feedforward Neural Network (FNN) defines a parametric function $f_q(\mathbf{x}) = \mathbf{y}$ which maps an input \mathbf{x} to an output \mathbf{y} , where f_q is a composition of fully connected layers.

Each of these layers is a function, which is a further composition of an activation function and an affine transformation. The weights of the affine transformation correspond to the parameters of the network and must be "learned" from the data.

In a FNN we distinguish three types of layers: *input layer*, *hidden layers* and *output layer*. The *input layer* is the layer that represent the known input. Indeed it has the same size of the input and it does not perform transformations on it; it is used only to bring the input inside the network. The *hidden layers* are "standard layers"; they receive the input from a previous layer and their output becomes the input of the consecutive layer. The *output layer* is the last layer of the network. It is a "standard layer", the only difference with a hidden layer is that its output is not further processed, but it represent the unknown output we are interested in.

FNN models are typically represented with directed acyclic graphs describing how the layers are composed together. In Figure 2.2 we show a representation of a FNN with three hidden layers. Nodes represent the activation functions (the identity for the input layer) and connections represent affine transformations.

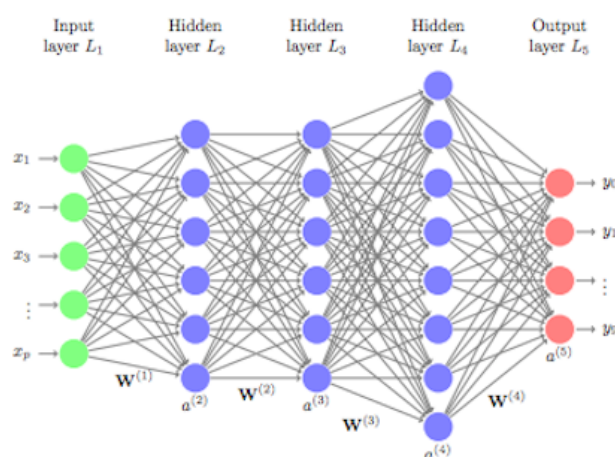


FIGURE 2.2: Representation of a Feedforward Neural Network (UC-R-Programming, 2018).

FNNs have found many applications; for instance they are a fundamental part of *Convolutional Neural Networks*, a class of NNs which is widely used in computer vision and which stacks together *convolutional layers* and *fully connected layers* (Goodfellow, Bengio, and Courville, 2016).

Furthermore, FNNs have good theoretical properties. They are known indeed for being universal function approximators (Hecht-Nielsen, 1992).

2.2.3 DeepHit

Among FNNs, DeepHit (C. Lee et al., 2018) may be used to predict the distribution of survival time for a patient given its vector of predictors. In this Section we briefly explain the *DeepHit* survival model introduced in C. Lee et al., 2018, which was developed to model a "competitive framework", i.e. a framework in which there could be more than one type of event. We will consider only the simple case in which there is only one event of interest.

In this framework we consider a sequence of K discrete times t_1, \dots, t_K , where t_K is considered to be the maximum survival time.

The DeepHit model consists of a FNN which takes as input the features vector \mathbf{x} of an individual and gives as output a probability distribution $\mathbf{y} = [y_1, \dots, y_K]$, where the i th element corresponds to the probability that survival time of the individual is equal to $t_i, i = 1, \dots, K$.

Observe that, using the DeepHit model for our purposes, the proportionality assumption made by the functional Cox-type regression model is no

more necessary. Indeed no assumption about the underlying stochastic process is made.

The net is composed by 3 fully-connected hidden layers. The activation function of the output layer is the *SoftMax* function, since the output is a discrete probability distribution. The activation function for the hidden layers is the *ReLU* function, which is widely used due to its good computational properties (Ramachandran, Zoph, and Le, 2017). The authors set a number of nodes for hidden layers as 3, 5, and 3 times of the covariate dimension for the layer 1, 2, and 3, respectively.

Let us denote by \mathbf{x}_i the features of individual i , by T_i its observed survival time, by z_i its censoring indicator (1 meaning uncensored) and by $\mathbf{y}^{(i)}$ its predicted probability distribution. Let N be the total number of patients.

The loss function L_{total} to be minimised is specifically designed to handle censored data and it is the sum of two terms:

$$L_{total} = L_1 + L_2 \quad (2.57)$$

where:

$$L_1 = \sum_{i=1}^N \left[(z_i=1) \log(y_{T_i}^{(i)}) + (z_i=0) \left(1 - \prod_{j:t_j < T_i} y_j^{(i)} \right) \right] \quad (2.58)$$

is the negative log-likelihood of the joint distribution of survival times modified to take account of the right-censoring of the data, and:

$$L_2 = a \sum_{i \notin j} (z_i=1, T_i < T_j) \exp \left[\frac{1}{S} \left(\prod_{j:t_j < T_i} y_j^{(j)} - \prod_{j:t_j < T_i} y_j^{(i)} \right) \right] \quad (2.59)$$

penalises incorrect ordering of comparable pairs. Parameters a and S have to be chosen through cross-validation.

The authors trained the NN by back-propagation via *Adam* optimizer (Kingma and Ba, 2014) with a batch size of 50, a learning rate of 10^{-4} , a dropout probability of 0.6 and *Xavier* initialization (Koturwar and Merchant, 2017) for all the layers.

Given a vector of predictors \mathbf{x}_i , the DeepHit model returns a vector $\mathbf{y}^{(i)}$, with $y_j^{(i)} = \mathbf{P}(T_i = t_j)$. Thus, the predicted mean survival time can be easily

obtained as:

$$\mathbb{E}[T_i] = \sum_{k=1}^K t_k y_k^{(i)} \quad (2.60)$$

Predictions of the DeepHit model can be easily converted into survival functions. Indeed, let $S^{(i)}(t)$ be the survival function of survival time T_i . We can construct the survival function $S^{(i)}(t)$ using the probability distribution vector $\mathbf{y}^{(i)}$ as follows.

$$\begin{aligned} S^{(i)}(t_j) &= P(T_i > t_j) \\ &= 1 - P(T_i \leq t_j) \\ &= 1 - \sum_{k=1}^j P(T_i = t_k) \\ &= 1 - \sum_{k=1}^j y_k^{(i)} \end{aligned} \quad (2.61)$$

As we stated before, this model does not make assumptions on the underlying stochastic process, therefore allowing us to discard the proportionality assumption made by the Cox-type regression models. However, the model also has some drawbacks. The effects of the covariates on the predictions is not easily interpretable as in the functional Cox-type regression model. Furthermore, the DeepHit model regards the event probability estimation as a pointwise prediction problem, ignoring the reasonable sequential patterns within consecutive predictions. Therefore it could occur that the model predicts "non-smooth" densities.

There is a class of NNs which is able to handle the dependencies between predictions consecutive in time, namely the *Recurrent neural networks* (RNNs). The Deep Recurrent Survival Analysis (DRSA) model is based on a particular type of RNN, namely the *Long-Short Term Memory Neural Network* (LSTMNN), which we explain in Section 2.2.4.

2.2.4 Long-Short Term Memory Neural Network (LSTMNN)

Recurrent neural networks (RNNs) are a family of NNs specialised for processing sequential data (Goodfellow, Bengio, and Courville, 2016). They contain a recursive layer whose behaviour can be described with an Equation of the

form:

$$\mathbf{h}(t) = f(\mathbf{x}(t), \mathbf{h}(t-1)) \quad (2.62)$$

where $\mathbf{x}(t)$ is the input at time t and $\mathbf{h}(t)$ is called *hidden state* or *cell output*. Typically, $\mathbf{h}(0) = \mathbf{0}$ and the final output is a non-linear transformation of the cell output.

As we mentioned before, to update the parameters during the training phase the gradient of the parameters with respect to some loss function is computed through the chain rule of calculus. This involves products of many terms and, due to the finite precision of computers, these products can rapidly converge to 0 (or more rarely "explode"). This phenomenon is known as *vanishing gradient* (Goodfellow, Bengio, and Courville, 2016).

The *Long-Short Term Memory Neural Network* (LSTMNN) is a particular RNN that tries to solve the vanishing gradient problem, introducing self-loops where the gradient can flow for long durations. To do that, it introduces a vector variable \mathbf{c} which is called *cell state*.

A standard LSTMNN is composed by stacking one or more *Long-Short Term Memory* (LSTM) layers with one or more fully connected layers. LSTM layers are also known as LSTM *cells*. We will consider only the case in which there is only one LSTM layer followed by one fully connected layer. The behaviour of a LSTM layer can be described by equations of the general form:

$$\begin{aligned} \mathbf{h}(t) &= f_1(\mathbf{x}(t), \mathbf{h}(t-1), \mathbf{c}(t)) \\ \mathbf{c}(t) &= f_2(\mathbf{x}(t), \mathbf{h}(t-1), \mathbf{c}(t-1)) \end{aligned} \quad (2.63)$$

where the variable \mathbf{c} is called *cell state*.

More specifically, the functions f_1 and f_2 in Equations 2.63 usually take the following form:

$$\begin{aligned} \mathbf{h}(t) &= \mathbf{o}(t) \tanh(\mathbf{c}(t)) \\ \mathbf{c}(t) &= \mathbf{f}(t)\mathbf{c}(t-1) + \mathbf{i}(t) \tanh(W_{ig}\mathbf{x}(t) + W_{rg}\mathbf{h}(t-1) + \mathbf{b}_{ig}) \end{aligned} \quad (2.64)$$

where the terms $\mathbf{o}(t)$, $\mathbf{f}(t)$ and $\mathbf{i}(t)$ are called *gate units*.

Definition 2.2.1. The terms $\mathbf{o}(t)$, $\mathbf{f}(t)$ and $\mathbf{i}(t)$ in Equation 2.64 are called, respectively, **output gate unit**, **forget gate unit** and **input gate unit**. They are defined as follows:

$$\begin{aligned}
\mathbf{o}(t) &= s(W_{io}\mathbf{x}(t) + \mathbf{b}_{io} + W_{ro}\mathbf{h}(t-1) + \mathbf{b}_{ro}) \\
\mathbf{f}(t) &= s(W_{if}\mathbf{x}(t) + \mathbf{b}_{if} + W_{rf}\mathbf{h}(t-1) + \mathbf{b}_{rf}) \\
\mathbf{i}(t) &= s(W_{ii}\mathbf{x}(t) + \mathbf{b}_{ii} + W_{ri}\mathbf{h}(t-1) + \mathbf{b}_{ri})
\end{aligned} \tag{2.65}$$

The terms in Equation 2.65 take values in $(0, 1)$ since their activation functions is the sigmoid:

$$s(\mathbf{z})_i = \frac{1}{1 + e^{-z_i}} \tag{2.66}$$

They are called gates since their purpose is to modulate the effects of the terms they multiply in Equation 2.64 on the update of the cell state $\mathbf{c}(t)$ and the hidden vector $\mathbf{h}(t)$.

Intuitively, the output gate modules how much the internal state of the cell is used to compute the cell output, the input gate modules the extent to which a new value flows into the cell and the forget gate modules how much a value remains in the cell.

As we mentioned above, a LSTMNN is composed by stacking a LSTM layer with a fully connected layer. The LSTM layer receives the input, processes it and updates the two internal states. The fully connected layer receives the updated hidden state, process it and returns the output. Then, the LSTMNN receives and processes the input subsequent in time, using the values of the hidden vectors from the previous computations and updating them again. Therefore, each prediction corresponds to a single time step, and a temporal dependency among predictions is introduced.

The weights of the affine transformations in the LSTM layer (Equations 2.64 and 2.65) and in the fully connected layer represent the parameters of the LSTMNN, which are to be learned from data.

A representation of a LSTM layer is given in Figure 2.3. The three products represent the effects of the three previously described gates. The incoming horizontal rows represent the internal states in input from the previous computation, and the outgoing horizontal rows represent the input for the following one. The incoming vertical line represent the external input (the vector of predictors), and the outgoing vertical line corresponds to the input for the fully connected layer.

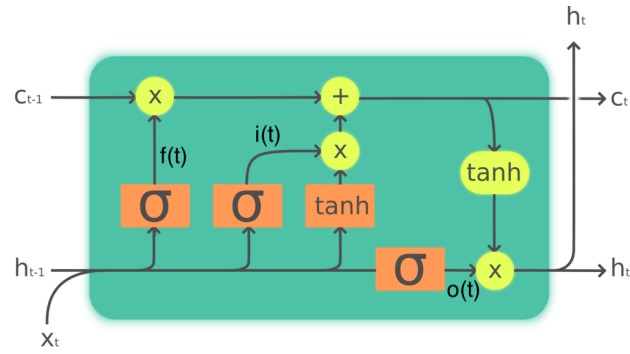


FIGURE 2.3: Long-Short Term Memory cell (modified version of Wikimedia-Commons, 2019).

In Section 3.3.2 we will predict the hazard rates of patients using a DRSA model, which is a survival model based on a LSTMNN. In Section 2.2.5 we introduce the DRSA model.

2.2.5 Deep Recurrent Survival Analysis (DRSA)

In this Section we describe the *Deep Recurrent Survival Analysis* (DRSA) model introduced in Ren et al., 2018.

As in Section 2.2.3, we consider a sequence of K discrete times t_1, \dots, t_K , where t_K is considered to be the maximum survival time.

Let us denote by \mathbf{x}_i the features of individual i , by T_i its possibly censored observed survival time, and by z_i its censoring indicator (1 meaning uncensored). Let N be the total number of patients.

Let us introduce the *discrete hazard function* for a discrete survival time T , which approximates the continuous hazard function and it is defined as:

$$l(t_l) = P(T = t_l | T > t_{l-1}) \quad l = 1, \dots, K \quad (2.67)$$

The DRSA model is based on a LSTMNN architecture. The input of the network at time t for individual i is the vector $[t, \mathbf{x}_i]$, and the output is the discrete hazard:

$$l^i(t) = \text{Sigmoid}(\mathbf{a} \mathbf{h}_t^i + b) \quad (2.68)$$

where \mathbf{a} and b are parameters and \mathbf{h}_t^i is the recursively computed hidden state of the cell at time t for individual i . Considering together the hidden state and the cell state as $\mathbf{r}_t = [\mathbf{h}_t, \mathbf{c}_t]$ and denoting the entire net by $f_{q,\mathbf{a},b}$ we

can express the whole model as:

$$l^i(t) = f_{q,a,b}(\mathbf{x}_i, t | \mathbf{r}_{t-1}^i) \quad (2.69)$$

where \mathbf{r}_{t-1} is recursively computed by the LSTM layer. Observe that the vector \mathbf{r} is used to model the conditional probability over time. This is a quite natural choice and it is widely been used (Bahdanau, Cho, and Bengio, 2014).

For our analysis, we decided to make the initial values of the hidden vector and the cell state learnable, by setting:

$$\begin{aligned} \mathbf{h}_0^i &= \text{Relu}(A_1 \mathbf{x}_i + \mathbf{b}_1) \\ \mathbf{c}_0^i &= \text{Relu}(A_2 \mathbf{x}_i + \mathbf{b}_2) \end{aligned} \quad (2.70)$$

where values of matrices A_1 , A_2 and vectors \mathbf{b}_1 and \mathbf{b}_2 have to be learned from data.

The DRSA model returns the predicted hazard one time step at a time. It is therefore quite different from the DeepHit model which, given the vector of predictors, returns the predicted density on the entire discrete time space.

To obtain the predicted hazard for a patient i for our purposes, we sequentially give in input the same vector of predictors \mathbf{x}_i and the time to predict t . The output sequence corresponds to the predicted hazard for the patient.

Observe that, using the DRSA model, we avoid again the proportionality assumption made by Cox-type regression models, since no assumption about the underlying stochastic process is made. However, the effects of the covariates on the predictions is not easily interpretable as in the functional Cox-type regression model.

This model solves the lack of time dependency among consecutive predictions that we saw in the DeepHit model. Indeed it is incorporated through the recursively computed hidden vector among the same patient.

The loss function L_{total} to be minimised is specifically designed to handle censored data and it is a weighted sum of three losses, denoted by L_z , $L_{uncensored}$ and $L_{censored}$, and it is given by:

$$L_{total} = aL_z + (1 - a)(L_{uncensored} + L_{censored}) \quad (2.71)$$

where a is a parameter.

The loss L_z is defined as:

$$\begin{aligned} L_z &= \log \prod_{i:z_i=1} \tilde{\mathbf{O}} \mathbf{P}(T = T_i | \mathbf{x}_i) \\ &= \mathop{\mathring{\mathbf{a}}}\limits_{i:z_i=1} [\log(l^{(i)}(T_i))] + \mathop{\mathring{\mathbf{a}}}\limits_{j:t_j < T_i} \log(1 - l^{(i)}(t_j)) \end{aligned} \quad (2.72)$$

and it is the negative log-likelihood of the true event times over the uncensored observations.

The loss $L_{uncensored}$ is defined as:

$$\begin{aligned} L_{uncensored} &= \log \prod_{i:z_i=1} \tilde{\mathbf{O}} \mathbf{P}(T > T_i | \mathbf{x}_i) \\ &= \mathop{\mathring{\mathbf{a}}}\limits_{i:z_i=1} \log[1 - \prod_{j:t_j > T_i} \tilde{\mathbf{O}} (1 - l^{(i)}(t_j))] \end{aligned} \quad (2.73)$$

and it is the negative partial log-likelihood of the event rate over the uncensored observations.

The loss $L_{censored}$ is defined as:

$$\begin{aligned} L_{censored} &= \log \prod_{i:z_i=0} \tilde{\mathbf{O}} \mathbf{P}(T > T_i | \mathbf{x}_i) \\ &= \mathop{\mathring{\mathbf{a}}}\limits_{i:z_i=0} \mathop{\mathring{\mathbf{a}}}\limits_{j:t_j > T_i} \log(1 - l^{(i)}(t_j)) \end{aligned} \quad (2.74)$$

and it is the partial log-likelihood of the censored observations.

Predictions of the DRSA model can be easily converted into survival functions. Indeed, let $S^{(i)}(t)$ be a survival function of a survival time T_i and let $l^{(i)}(t)$ be its predicted hazard function taking values in t_1, \dots, t_K . We can build the survival function $S^{(i)}$ as follows.

$$\begin{aligned}
S^{(i)}(t_j) &= P(T_i > t_j) \\
&= P(T_i \notin t_1, \dots, T_i \notin t_j) \\
&= \prod_{k=1}^j P(T_i \notin t_k | T_i > t_{k-1}) \\
&= \prod_{k=1}^j (1 - P(T_i = t_k | T_i > t_{k-1})) \\
&= \prod_{k=1}^j (1 - l^{(i)}(t_k))
\end{aligned} \tag{2.75}$$

We can then reconstruct the predicted mean survival time as follows.

$$\begin{aligned}
E[T_i] &= \sum_{j=1}^K t_j P(T_i = t_j) \\
&= \sum_{j=1}^K t_j P(T_i = t_j | T_i > t_{j-1}) P(T_i > t_{j-1}) \\
&= \sum_{j=1}^K t_j l^{(i)}(t_j) S^{(i)}(t_j) \\
&= \sum_{j=1}^K t_j l^{(i)}(t_j) \prod_{k=1}^j (1 - l^{(i)}(t_k))
\end{aligned} \tag{2.76}$$

In Section 3.3.2 we will use the DRSA model to predict hazard functions for each patient and we will compare its predictions with the ones of other models using predicted mean survival times and predicted survival functions.

To compare predictions of survival times obtained with the three models we described (functional Cox-type regression, DeepHit and DRSA) we will use the techniques described in Section 2.3.

2.3 Comparison among different models

In the previous Sections we described three models, namely functional Cox-type regression, DeepHit and DRSA, that allow us to predict survival time of patients. We would like to compare the performances and the differences

between the models by using their output on the test set, exploiting both quantitative and graphical tools. Due to censoring, we cannot rely on standard regression goodness-of-fit measures, like the Mean Squared Error.

We would like a good model to distinguish, given two patients, which one will survive more time. This quality is measured by the Concordance Index (Pencina and D'Agostino, 2004), which is the most common evaluation index used in Survival Analysis (C. Lee et al., 2018). In order to define the Concordance Index, we give the definition of *usable pair* (Pencina and D'Agostino, 2004).

Definition 2.3.1. A pair (T_i, T_j) of possibly censored survival times for two individuals is said to be **usable** if $T_i \neq T_j$ and the lower survival time is uncensored.

Usable pairs of observed survival times are pairs for which we can assess which individual survived for a longer time.

Let us introduce the risk notion. We call *risk* a general quantity that gives a relative measure of the propensity of individuals to fail.

Let M be a model that given a vector of patient's predictors x_i returns the patient risk r_i .

Definition 2.3.2. The **Concordance Index** of a model M is the probability that M correctly orders a random pair of survival times (T_i, T_j) conditioned to the fact that the pair is usable.

$$C = \mathbf{P} \{ \hat{r}_i > r_j, T_i < T_j \text{ or } \hat{r}_i < r_j, T_i > T_j \mid (T_i, T_j) \text{ is usable} \} \quad (2.77)$$

The C index measures the probability that, given two comparable patients, the patient who survives longer will have the lower predicted risk. Therefore, it measures how well a model predicts the ordering of sample event times.

An estimator of the Concordance Index is given in the following definition.

Definition 2.3.3. Let r_i be the predicted risk for the i th individual, T_i its observed survival time and z_i its censoring indicator. The **sample Concordance Index** is defined as:

$$\hat{C} = \frac{\sum_{(i,j) \in W} \mathbb{1}_{\{\hat{r}_i > r_j\}}}{|W|} \quad (2.78)$$

where $W = \{(i, j) : T_i < T_j, z_i = 1\}$

For our analysis we will use as a measure of risk the linear prediction $b^T \mathbf{x}_i$ for the functional Cox-type regression model and the negative expected survival time $-E[T_i]$ for the DeepHit and the DRSA models.

For the Concordance Index, the only thing that matters is the order of risks. Given two patients i and j , the expected value of survival time predicted by the functional Cox-type regression model for patient i is greater than that of patient j if and only if the linear prediction $b^T \mathbf{x}_i$ is less than $b^T \mathbf{x}_j$. Therefore we would obtain the same Concordance Index if we used the negative expected survival times instead of the linear predictions as measures of risk for the functional Cox-type regression model.

Given two patients i and j , we thought it reasonable to consider i at greater risk than j if the expected survival time of i is lower than that of j .

Under random censoring assumption, the concordance index estimator \hat{C} given in Definition 2.3.3 is asymptotically normal and it converges to the true concordance index C (Pencina and D'Agostino, 2004). Therefore, given two models M_1 and M_2 , the difference among the Concordance Index Estimates $\hat{C}_1 - \hat{C}_2$ obtained by predicting risks for the same set of individuals with the two models is also normal and it converges to the true difference $C_1 - C_2$. Using the estimate for the variance of $\hat{C}_1 - \hat{C}_2$ given in Kang, Chen, et al., 2015 and assuming a sufficiently large number of observations we can perform the test:

$$\begin{aligned} H_0 : C_1 - C_2 &= 0 \\ H_1 : C_1 - C_2 &> 0 \end{aligned} \tag{2.79}$$

We can therefore test if the estimated differences of performances between two models is significant.

In Section 3.4 we will assess the performance of our models in predicting survival time of patients, evaluating and testing the concordance index with the previously described techniques.

The Concordance Index allow us to rank models, based on a scalar which summarises each prediction. Since the predictions given by our models are functions, we would like also to compare them graphically. We already saw that the predictions of the three models (functional Cox-type regression, DeepHit and DRSA) can be converted into survival functions. In Section 3.4 we will graphically compare the predicted survival curves obtained with these three models.

Chapter 3

Applications and Results

In this Chapter we outline how we applied the techniques described in Chapter 2 to analyse data presented in Chapter 1.

In Section 3.1 we modelled the events happened in the observation period (rehospitalisations, purchase of ACE inhibitors, purchase of BB agents, purchase of AA agents) as realisations of stochastic processes, and we estimated the corresponding compensators. In Section 3.2 we summarised these compensators through the scores obtained by applying FPCA. We used this information in predictive models for survival time in a traditional Cox-type regression model in Section 3.3.1 and with a Machine Learning based approach in Section 3.3.2. Finally, in Section 3.4 we compared performances of the models in terms of Concordance Indexes on a test set and we explored the predictions graphically.

3.1 Modelling the clinical history

We refer to the recurrent events happened to patients in their one year observation period as to their "clinical history". We recall that we are considering four types of events: rehospitalisations due to HF and purchases of Angiotensin Converting Enzyme (ACE) inhibitors, Beta-Blocking (BB) agents and Anti Aldosterone (AA) agents.

Following the example of Baraldo et al., 2013, we described the processes of rehospitalisation and drugs purchases using a recurrent events framework. In particular we treated them as data generated from a Marked Point Process (MPP) and we reconstructed the realisations of the corresponding compensators, focusing our analyses on these objects.

In Section 3.1.1 we modelled each compensator with a Cox model for recurrent events using suitable covariates chosen through a cross-validation technique. In Section 3.1.2 we fitted and smoothed the resulting cumulative

baseline hazard functions. Finally, in Section 3.1.3 we reconstructed the realisations of the compensators of the MPPs.

The code that we developed to obtain the results presented in Sections 3.1.1, 3.1.2 and 3.1.3 is available in Section A.1 of the Appendix.

3.1.1 Features selection

Following the example of Baraldo et al., 2013, we used a Cox model for recurrent events (as in Section 2.1.2) to fit the compensators of the processes described in Section 1.3.

We remind that we see the clinical history of each patient i as the realisation of marked counting processes $N_i^{(k)}$, $k \in \{ACE, BB, AA, rehosp.g\}$ and that for each of these processes there is a unique decomposition:

$$M_i^{(k)}(t) = N_i^{(k)}(t) - \int_0^t l_i^{(k)}(s) ds \quad (3.1)$$

where $\int_0^t l_i^{(k)}(s) ds$ is the compensator of the process $N_i^{(k)}$ and $M_i^{(k)}$ is a martingale. In Section 2.1.2 we have seen that the Cox model for recurrent events assumes the following form for the random intensity process $l_i(t)$.

$$l_i(t) = Y_i(t) l_0(t) e^{b^T x_i(t)} \quad (3.2)$$

where b is a fixed vector of coefficients, x_i is the possibly time-dependent vector of covariates of the i th individual, l_0 a fixed underlying hazard function and Y_i is a predictable process taking values in $\{0, 1\}$ which sets (by the value 1) when the i th individual is under observations.

We now want to identify meaningful predictors for this model. Therefore, given two or more Cox models for recurrent events fitted using different sets of covariates, we need a metric to assess which one is "better". Since we are dealing with stochastic processes and recurrent events, we cannot rely on standard regression metrics, like Mean Squared Error.

A possible way to measure the "goodness of fit" for these models is given by functions of the residuals (Therneau, Grambsch, and Fleming, 1990). In fact, smaller residuals correspond to a greater predictive power of the model.

Once we estimated the compensators \hat{L}_i of the counting processes N_i for each of the four considered stochastic process, we can reconstruct the residuals as:

$$\hat{M}_i = \hat{L}_i - N_i \quad (3.3)$$

We denote the residuals by \hat{M}_i because they estimate the realisations of the martingale M_i involved in the Doob-Meyer decomposition, described in Section 2.1.2.

To compare models obtained by using different features, we would like to use the Mean Absolute Martingale Residual (MAMR):

$$MAMR = \frac{1}{N} \sum_{i=1}^N \int_0^T \hat{M}_i(s) ds \quad (3.4)$$

where T represents the length of the observation year.

To correctly compute the MAMR we should first compute the compensator using the techniques described in Section 2.1.2 and then evaluate the residuals in a grid of points. This implies a high computational cost (several hours of computations on a common laptop) solely for the sake of choosing a suitable set of features. Since we want to use this quantity only to rank models fitted with different sets of predictors, we decided to rely on the estimate:

$$MAMR = \frac{1}{\sum_{i=1}^N n_i} \sum_{i=1}^N \sum_{j=1}^{n_i} \hat{M}_i(t_j^{(i)}) \quad (3.5)$$

where \hat{M}_i is the residual obtained by fitting the compensator without smoothing the baseline hazard for patient i , n_i is the total number of events he experienced and $t_j^{(i)}$ is the time instant in which he experienced the j th event.

This estimate is not accurate since the residuals are evaluated only when events happen (rather than on the continuous interval corresponding to the one year observation period) and because the estimate is done by reconstructing the compensators without the smoothing of the baseline hazard. However, it allows to rank models while limiting computational needs (refer to residuals function of survival R package, Therneau and Lumley, 2014).

Let us refer to the features described in Table 1.3. In all the models, if significant, we used features *age_in* and *sex*. Then, we considered the number of events occurred in the past Nm , the sum of the corresponding marks y and the interaction of the previous two terms $Nm \cdot y$, in order to account for possible interaction among the two.

For Nm and y we tried to use both the original values and their logarithmic transformations shifted away from 0 ($\log(Nm + 1)$ and $\log(y + 1)$). This

allows the process to depend more or less intensively on large values of Nm and y .

A list of these models and an example of the output of this procedure is given in Figure 3.1.

The estimates of b coefficients are retrieved using the `coxph` function of survival R package (Therneau and Lumley, 2014).

In all the cases we performed a ten folds cross-validation at a patient level, i.e. in each fold we trained the model on events regarding 90% of the patients and we evaluated it on the events of the remaining 10%.

```

***** Cross validation mean absolute Martingale residual *****
|-----|-----|
| MEAN ABS. RESIDUAL | FORMULA | |
|---|---|---|
| 0.689 | age_in + Nm + cluster(id)" |
| 0.801 | age_in + y + cluster(id)" |
| 0.689 | age_in + Nm + y + cluster(id)" |
| 0.833 | age_in + Nm:y + cluster(id)" <----- WORST MODEL" |
| 0.657 | age_in + Nm + Nm:y + cluster(id)" |
| 0.801 | age_in + y + Nm:y + cluster(id)" |
| 0.625 | age_in + Nm + y + Nm:y + cluster(id)" |
| 0.565 | age_in + log(Nm + 1) + cluster(id)" |
| 0.564 | age_in + log(Nm + 1) + y + cluster(id)" |
| 0.806 | age_in + log(Nm + 1):y + cluster(id)" |
| 0.563 | age_in + log(Nm + 1) + log(Nm + 1):y + cluster(id)" |
| 0.801 | age_in + y + log(Nm + 1):y + cluster(id)" |
| 0.563 | age_in + log(Nm + 1) + y + log(Nm + 1):y + cluster(id)" |
| 0.624 | age_in + log(y + 1) + cluster(id)" |
| 0.586 | age_in + Nm + log(y + 1) + cluster(id)" |
| 0.722 | age_in + Nm:log(y + 1) + cluster(id)" |
| 0.656 | age_in + Nm + Nm:log(y + 1) + cluster(id)" |
| 0.597 | age_in + log(y + 1) + Nm:log(y + 1) + cluster(id)" |
| 0.568 | age_in + Nm + log(y + 1) + Nm:log(y + 1) + cluster(id)" |
| 0.565 | age_in + log(Nm + 1) + cluster(id)" |
| 0.624 | age_in + log(y + 1) + cluster(id)" |
| 0.562 | age_in + log(Nm + 1) + log(y + 1) + cluster(id)" |
| 0.608 | age_in + log(Nm + 1):log(y + 1) + cluster(id)" |
| 0.563 | age_in + log(Nm + 1) + log(Nm + 1):log(y + 1) + cluster(id)" |
| 0.59 | | age_in + log(y + 1) + log(Nm + 1):log(y + 1) + cluster(id)" |
| 0.559 | age_in + log(Nm + 1) + log(y + 1) + log(Nm + 1):log(y + 1) + cluster(id) <----- BEST MODEL"

```

FIGURE 3.1: Output of the cross-validation procedure used to select the best set of features for the Cox model for recurrent events describing the stochastic process relative to the purchase of ACE inhibitors.

Finally, we checked for significance of the selected features in the model fitted on the entire training set.

In Table 3.1 we show the variables selected for describing the stochastic process of drugs purchase in the case of ACE inhibitors, the corresponding Hazard Ratios (HRs) and the 95% Confidence Interval (CI).

The HR relative to the age (age_in), being less than 1, indicates that younger patients are most likely to buy ACE-inhibitors than older ones.

The sex variable (sex) is not significant to predict purchases of ACE inhibitors.

We observe that the HRs related to the number of past events $\log(Nm + 1)$ and to the sum of the past marks $\log(y + 1)$ are greater than 1. This can be interpreted as a “self-exciting” behaviour: many purchases of ACE inhibitors in the past and the purchase of big quantities of ACE inhibitors increase the risk of a new purchase.

The HR relative to the interaction term $\log(Nm + 1) \log(y + 1)$ is lower than 1. This means that the increase in risk is softened in case of several purchases of ACE inhibitors and/or a great quantities of ACE inhibitors purchased. Let a patient i buy ACE inhibitors, having bought them a few times and in small quantities in the past. Let another patient j buy at the same time the same amount of ACE inhibitors, but having made several purchases and/or purchases of large quantities of ACE inhibitors in the past. For both patients the risk of purchasing other ACE inhibitors in the future increases but the increase of patient i is greater than the one of patient j .

Variable name	HR ($e^{\hat{b}_k}$)	CI (2.5%)	CI (97.5%)
age_in	0.9967	0.9957	0.9978
$\log(Nm + 1)$	4.5216	4.1633	4.9107
$\log(y + 1)$	1.1036	1.0872	1.1202
$\log(Nm + 1) \log(y + 1)$	0.9141	0.9025	0.9258

TABLE 3.1: Hazard ratios and corresponding 95% CI of the Cox model for recurrent events for the stochastic process describing the purchase of ACE inhibitors.

In Table 3.2 we show the variables selected for describing the stochastic process of drugs purchase in the case of BB agents, the corresponding HRs and the 95% CI.

The selected features and the sign of the fitted coefficients is the same as for the ACE inhibitors process. Therefore we can give a similar interpretation to the effect of predictors.

Younger patients are most likely to buy BB agents than older ones, and the sex variable is not significant to predict purchases of BB agents.

We still observe a “self-exciting” behaviour: frequent purchases of BB agents in the past and the purchase of big quantities of BB agents increase the risk of a new purchase.

The increase in risk is softened in case of frequent purchases of BB agents and/or a great quantities of BB agents purchased.

Variable name	HR ($e^{\hat{\beta}_k}$)	CI (2.5%)	CI (97.5%)
<i>age_in</i>	0.9928	0.9917	0.9939
$\log(Nm + 1)$	5.5360	5.2147	5.8770
$\log(y + 1)$	1.1340	1.1144	1.1540
$\log(Nm + 1) \quad \log(y + 1)$	0.8283	0.8161	0.8406

TABLE 3.2: Hazard ratios and corresponding 95% CI of the Cox model for recurrent events for the stochastic process describing the purchase of BB agents.

In Table 3.3 we show the variables selected for describing the stochastic process of drugs purchase in the case of AA agents, the corresponding HRs and the 95% CI.

Differently from the previous two processes, the *sex* variable is significant, while the age variable *age_in* is not. The HR of the *sex* variable, being lower than one and significant, indicates that females are most likely to buy AA agents than males.

As for the previous two models, we selected the variables $\log(Nm + 1)$, $\log(y + 1)$ and their interaction $\log(Nm + 1) \quad \log(y + 1)$. Since the sign of the relative coefficients are the same as for the previous two processes, we may give a similar interpretation. We still observe a “self-exciting” behaviour: many purchases of AA agents in the past and the purchase of big quantities of AA agents increase the risk of a new purchase. The increase in risk is softened in case of many purchases of AA agents and/or great quantities of AA agents purchased in the past.

Variable name	HR ($e^{\hat{b}_k}$)	CI (2.5%)	CI (97.5%)
<i>sex</i> (Male)	0.9435	0.9073	0.9811
$\log(Nm + 1)$	9.8781	8.6116	11.3310
$\log(y + 1)$	1.2023	1.1722	1.2332
$\log(Nm + 1) \quad \log(y + 1)$	0.7780	0.7561	0.8005

TABLE 3.3: Hazard ratios and corresponding 95% CI of the Cox model for recurrent events for the stochastic process describing the purchase of AA agents.

In Table 3.4 we show the variables selected for describing the stochastic process of rehospitalisations due to HF, the corresponding HRs and the 95% CI.

For this process, age and sex variables (*age_in* and *sex*) are both significant. The HR relative to *age_in* is lower than 1, which means that younger patients are most likely to be rehospitalised than older ones. The HR relative to *sex* is greater than 1, which means that females are less likely to be hospitalised than males.

For the hospitalisation process, the cross-validation procedure selected the original features Nm , y and $Nm \quad y$ instead of their logarithmic transformations. This is probably due to the fact that hospitalisations are rarer than drug purchases, so they may have a greater effect in increasing the risk of experiencing a new event.

We found that the HRs related to Nm and y are greater than 1. This can be interpreted again as a “self-exciting” behaviour: being hospitalised often in the past, and having spent long periods of time at the hospital both increase the risk of a new hospitalisation.

The HR for the interaction term $Nm \quad y$ is lower than 1. This means that the increase in risk is softened in case of many hospitalisations and/or in the case of a long time spent at the hospital in the past. Let a patient i experience a new hospitalisation having being hospitalised only a few times and for a short time. Let another patient j experience the same event at the same time, but having being hospitalised many times and/or for longer periods in the past. For both patients the risk of experiencing a new hospitalisation in the future increases but the increase of patient i is greater than the one of patient j .

Variable name	HR ($e^{\hat{b}_k}$)	CI (2.5%)	CI (97.5%)
<i>age_in</i>	0.9957	0.9934	0.9979
<i>sex</i> (Male)	1.1510	1.0854	1.2207
<i>Nm</i>	1.4319	1.3809	1.4848
<i>y</i>	1.0083	1.0051	1.0116
<i>Nm y</i>	0.9976	0.9968	0.9985

TABLE 3.4: Hazard ratios and corresponding 95% CI of the Cox model for recurrent events for the stochastic process describing the HF rehospitalisations.

We observed that for all four stochastic processes, the cross-validation procedure selected models which include features relative to the number of events in the past Nm , the sum of past marks y and their interaction, and their coefficients are always significantly different from 0. Furthermore, the signs of the fitted coefficients relative to these three types of features are consistent throughout the four processes, allowing us to give similar interpretations.

Now that we fitted the coefficients of the four Cox model for recurrent events describing the four considered stochastic processes, we can fit and smooth the corresponding cumulative baseline hazards.

3.1.2 Smoothing of the cumulative baseline hazard

Once we estimated the coefficients of each of the four Cox model for recurrent events, we found the estimated cumulative baseline hazard $\hat{\Lambda}_0$ with the Breslow estimator described in Section 2.1.2.

To deal with the fact that the Breslow estimator returns a step-function, we used the smoothing technique described in Section 2.1.2, as it was done in Baraldo et al., 2013. We used 20 knots for the spline basis; since our timeline starts at -0.5, we put the constraint $\tilde{\Lambda}_0(-0.5) = 0$.

Figures 3.2, 3.3, 3.4 and 3.5 show the fitted baseline cumulative hazard for the four considered processes. We show both the estimate obtained with the Breslow estimator $\hat{\Lambda}_0$ and the corresponding smoothed estimate $\tilde{\Lambda}_0$.

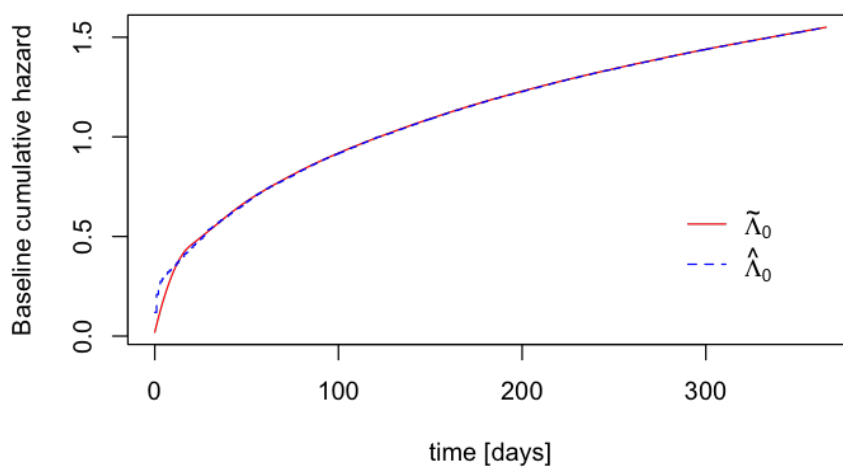


FIGURE 3.2: Fitted cumulative baseline hazard of the Cox model for recurrent events describing the stochastic process of purchase of ACE inhibitors, fitted with the Breslow estimator and smoothed according to the procedure described in Baraldo et al., 2013.

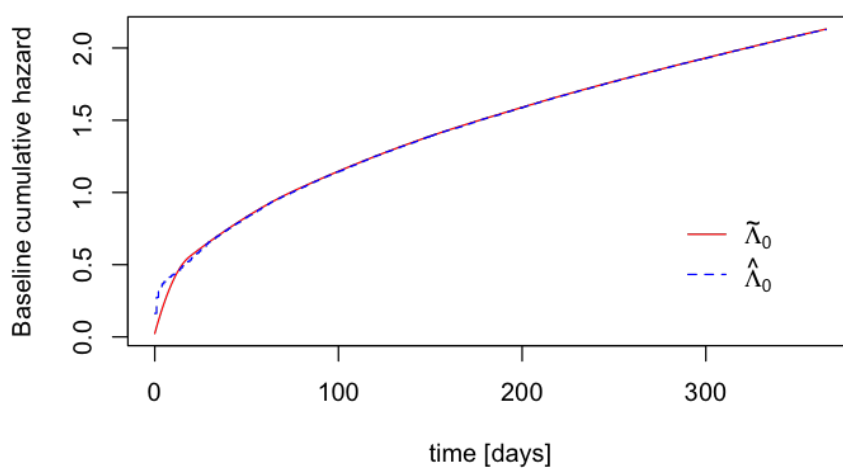


FIGURE 3.3: Fitted cumulative baseline hazard of the Cox model for recurrent events describing the stochastic process of purchase of BB agents, fitted with the Breslow estimator and smoothed according to the procedure described in Baraldo et al., 2013.

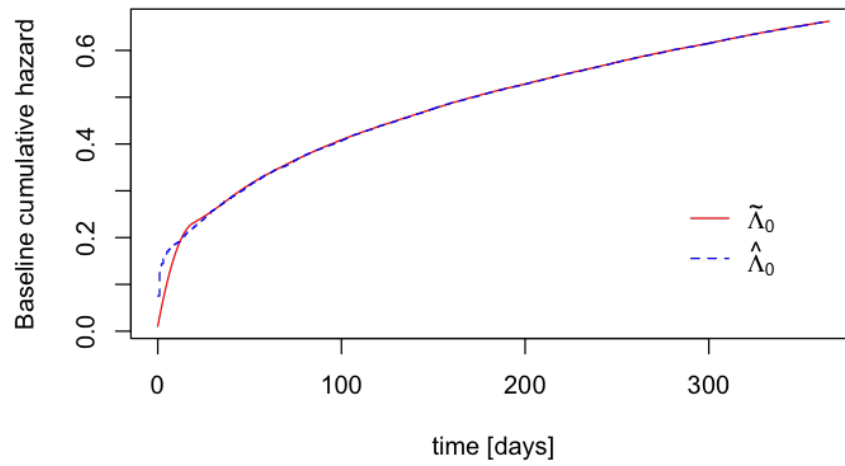


FIGURE 3.4: Fitted cumulative baseline hazard of the Cox model for recurrent events describing the stochastic process of purchase of AA agents, fitted with the Breslow estimator and smoothed according to the procedure described in Baraldo et al., 2013.

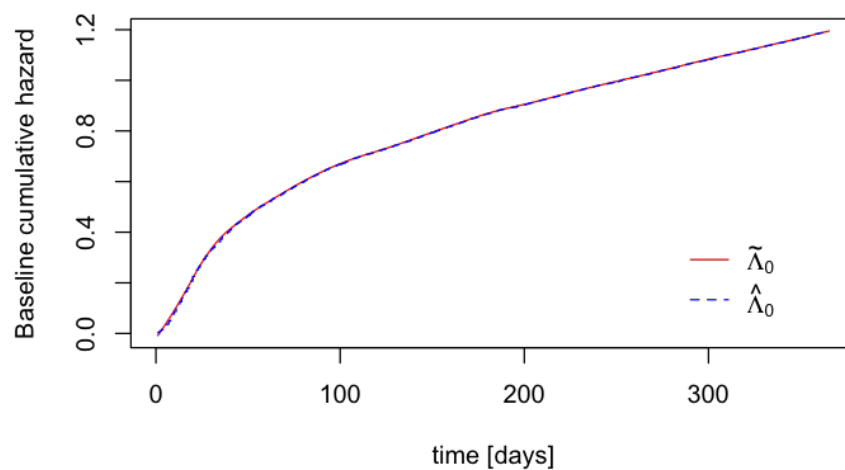


FIGURE 3.5: Fitted cumulative baseline hazard of the Cox model for recurrent events describing the stochastic process of HF hospitalisations, fitted with the Breslow estimator and smoothed according to the procedure described in Baraldo et al., 2013.

We can see that the estimates of the cumulative baseline hazards are monotonically increasing and that they take value 0 at time equal to -0.5.

Once we reconstructed the cumulative baseline hazards, we can proceed to the reconstruction on the compensators of the four considered stochastic processes.

3.1.3 Reconstruction of the compensators' trajectories

Afterwards, we built the trajectories of the compensators of the four counting processes representing the clinical history of patients. Since the values of the considered covariates (Tables 3.1, 3.2, 3.3 and 3.4) are step-wise constants and change only in correspondence of the jump times of the corresponding patient, we can exploit Equation 2.35 (reported here in Equation 3.6 to ease the notation recall).

$$\hat{\Lambda}_i(t) = \mathring{\mathbf{a}} \prod_{k=1}^{N_i(t)} e^{\hat{\mathbf{b}}^T \mathbf{x}_i(t_k - 1)} [\tilde{\Lambda}_0(\min(t_k^i, t)) - \tilde{\Lambda}_0(t_{k-1}^i)] \quad (3.6)$$

This Equation allows to estimate the realisations of the compensators of the considered stochastic processes as a function of the coefficients fitted in Section 3.1.1 and the cumulative baseline hazards fitted in Section 3.1.2. The code used to reconstruct the compensators, which is available in Section A.1 of the Appendix, is not optimised and requires a few minutes to run on a common laptop. Its improvement represents a possible extension of this thesis work.

Figures 3.6, 3.7, 3.8 and 3.9 show the compensators of the stochastic processes describing ACE-inhibitors purchases, BB agents purchases, AA agents purchases and HF rehospitalisations of 500 HF patients belonging to the dataset under analysis, fitted with models described in Section 3.1.1 and whose baseline cumulative hazard was smoothed according to the procedure described in Section 3.1.2. We can see that they are monotonically increasing and that they take value 0 at time equal to -0.5 as does the baseline cumulative hazards.

The large variability of the compensator across different patients reflects the variability of the realisations of their recurrent events. We observe that for the hospitalisations process the scale is smaller than for the purchases processes; this due to the fact that hospitalisations are less frequent than drugs purchases.

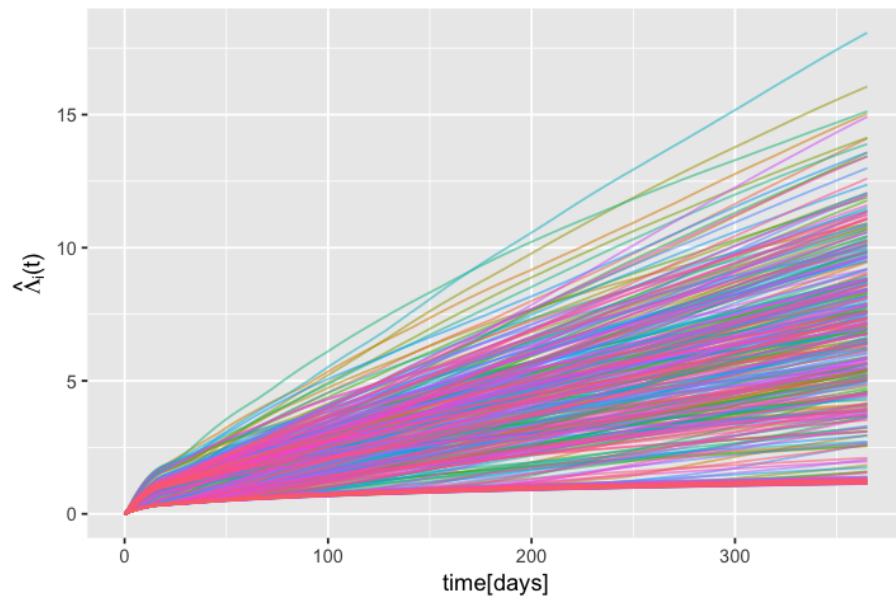


FIGURE 3.6: Compensators of the stochastic process describing the purchase of ACE inhibitors, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.

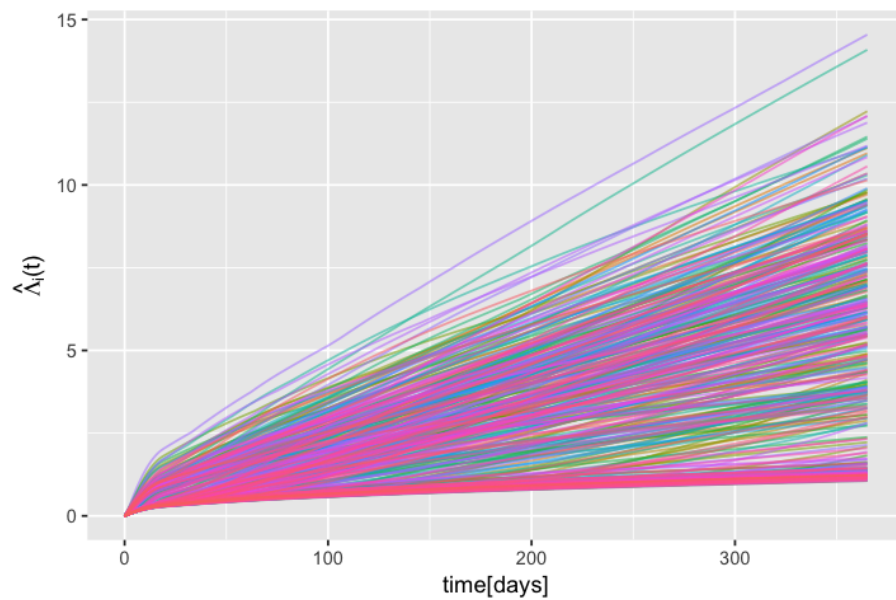


FIGURE 3.7: Compensators of the stochastic process describing the purchase of BB agents, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.

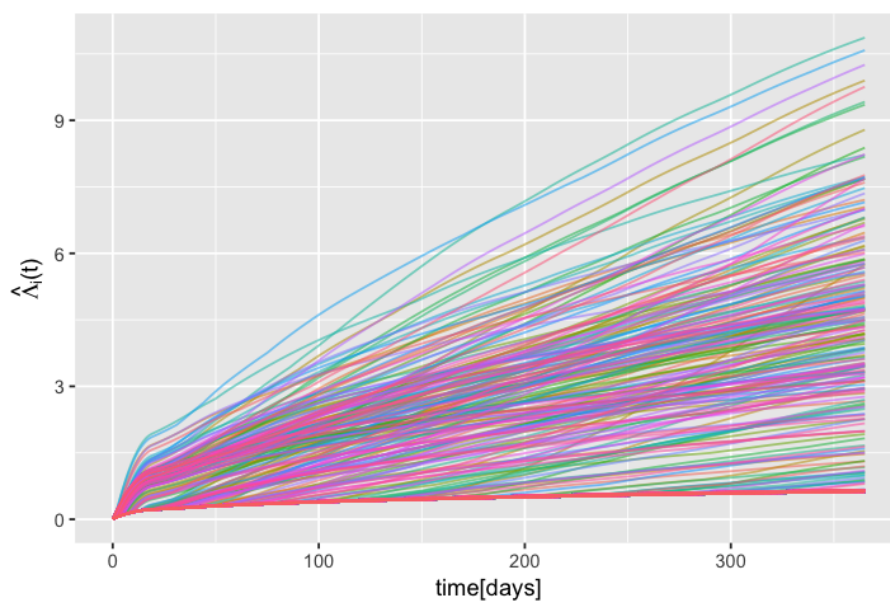


FIGURE 3.8: Compensators of the stochastic process describing the purchase of AA agents, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.

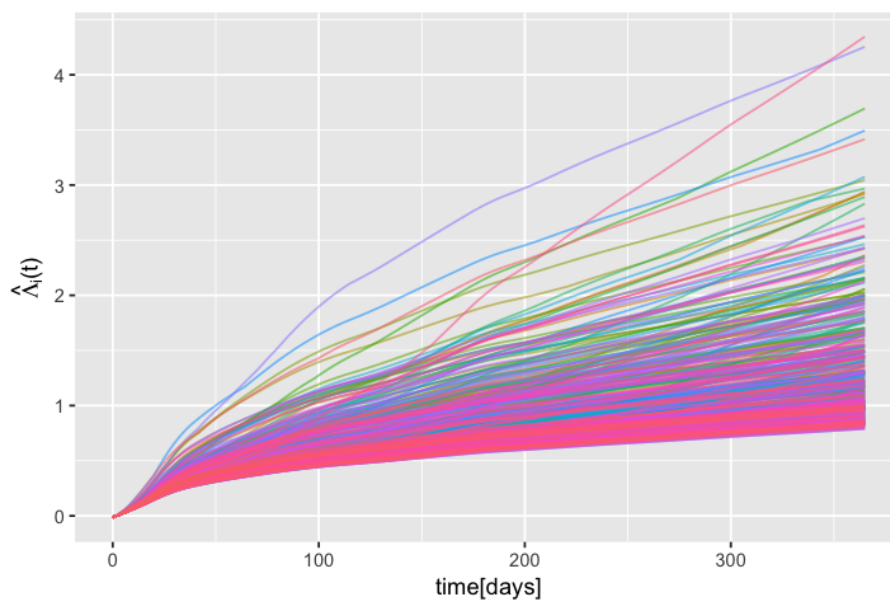


FIGURE 3.9: Compensators of the stochastic process describing the HF rehospitalisations, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.

Once we estimated the compensators $\hat{\Lambda}_i$ of the counting processes N_i for each of the four considered stochastic processes, we can reconstruct the residuals as:

$$\hat{M}_i = \hat{\Lambda}_i - N_i \quad (3.7)$$

We denote the residuals by \hat{M}_i because they should be estimates of realisations of the martingale M_i involved in the Doob-Meyer decomposition, described in Section 2.1.2.

We have to check if the fitted residuals $\hat{M}_i(t)$ may be effectively considered as realisations of martingales. To do so, for each process we considered (i.e. purchases of ACE inhibitors, purchases of BB agents, purchases of AA agents and rehospitalisations) we plotted the residuals evaluated in the whole observation year and we checked graphically that their mean $\bar{M}(t) = \frac{1}{N} \sum_{i=1}^N \hat{M}_i(t)$ is approximately 0.

Figures 3.10, 3.11, 3.12 and 3.13 show the fitted residuals $\hat{M}_i(t)$ for each process we considered for the sample of the 500 patients mentioned above. The black line corresponds to the temporal mean $\bar{M}(t)$, computed using all the 4,541 patients.

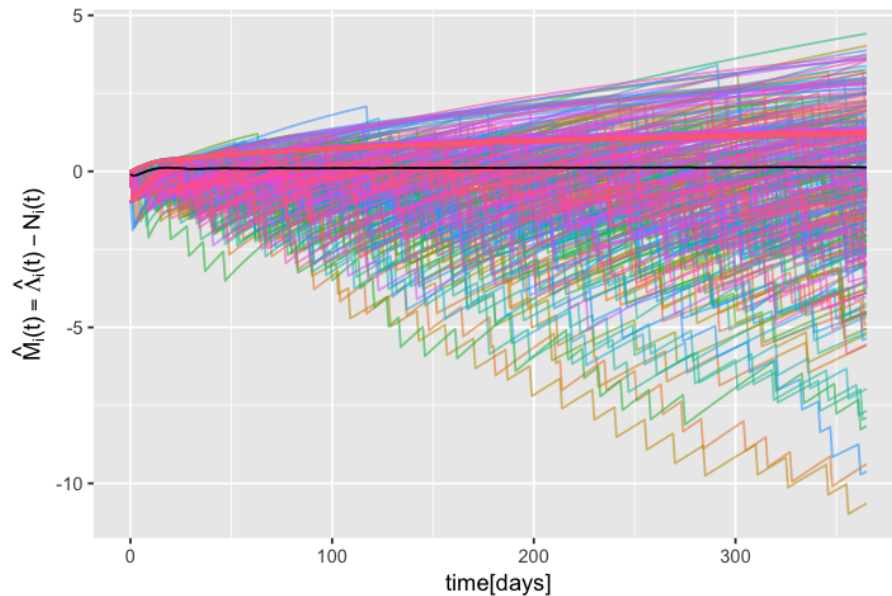


FIGURE 3.10: Residuals of the compensators of the stochastic process describing the purchase of ACE inhibitors, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.

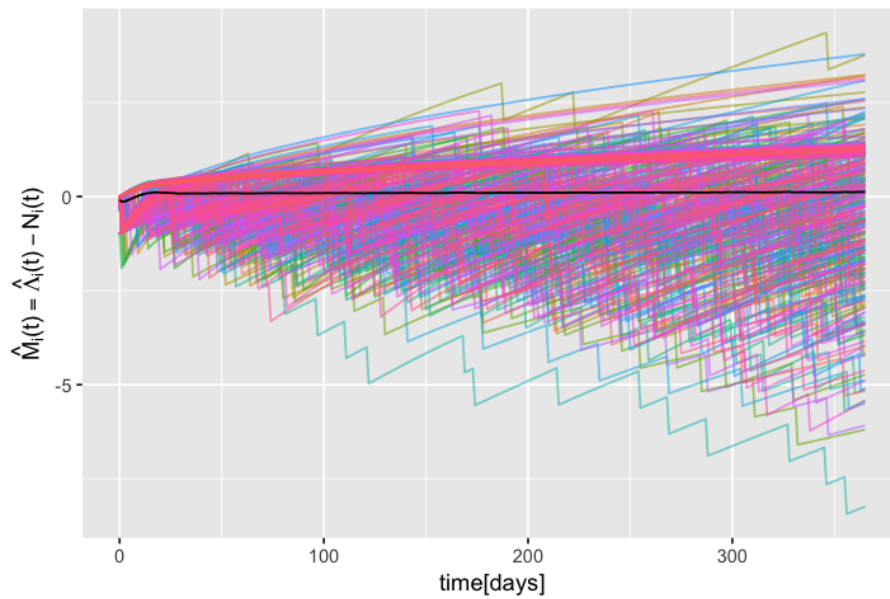


FIGURE 3.11: Residuals of the compensators of the stochastic process describing the purchase of BB agents, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.

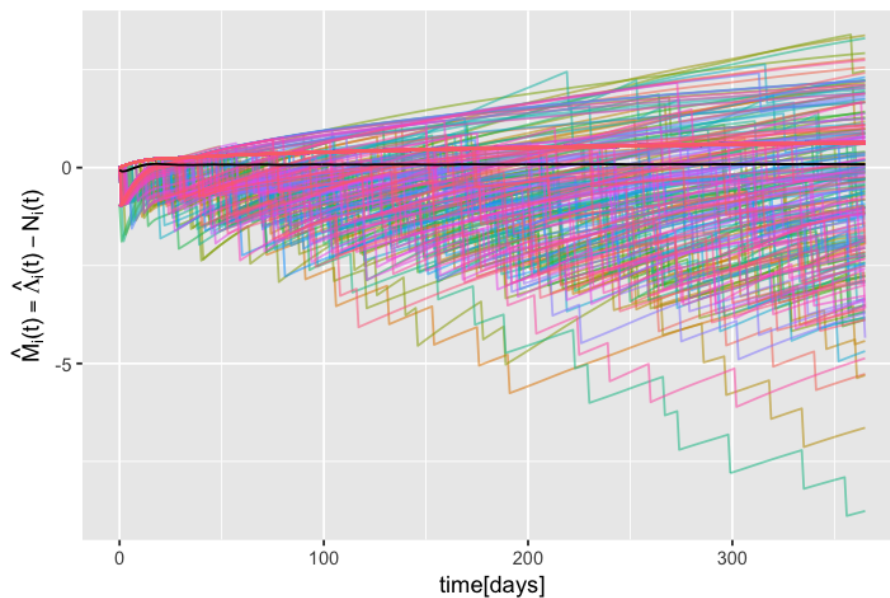


FIGURE 3.12: Residuals of the compensators of the stochastic process describing the purchase of AA agents, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.

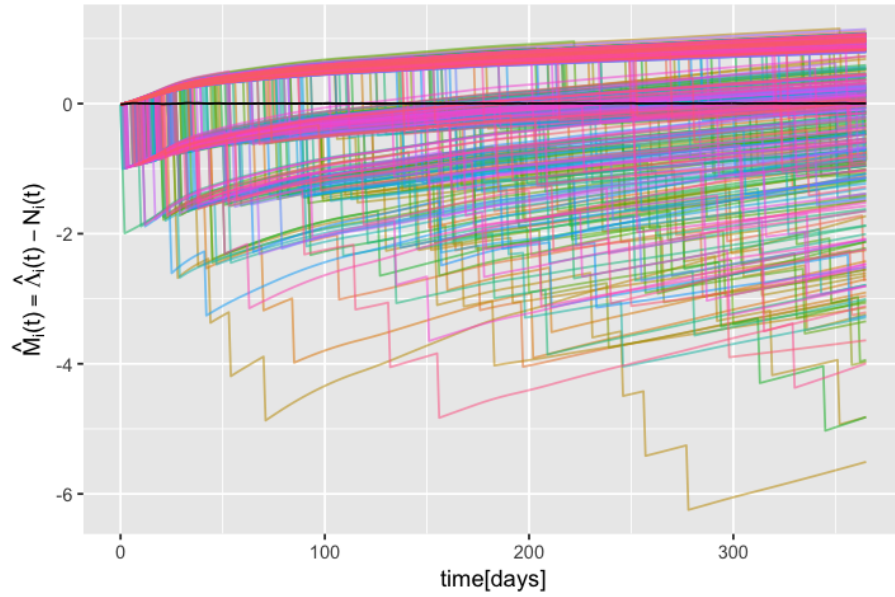


FIGURE 3.13: Residuals of the compensators of the stochastic process describing the HF rehospitalisations, fitted with the Cox model for recurrent events and smoothed according to the procedure described in Baraldo et al., 2013.

We can see that the time varying mean $\bar{M}(t)$ is approximately a constant equal to zero for all the considered processes. We may conclude that we succeeded in fitting the compensators of the stochastic process following the model described in Baraldo et al., 2013.

We will now focus on dimensionality reduction of these curves in order to plug them into the Cox-type regression model.

3.2 Summarise the information content of complex data

The code that we developed to obtain the results presented in this Section is available in Section A.2 of the Appendix.

In order to summarise and plug into a Cox-type regression model the information emerging from the compensators obtained in Section 3.1.3, we applied FPCA on them. To do this, we centered data removing their mean, i.e. $\hat{L}_i(t) - \bar{L}(t)$ as suggested in applications of FPCA (Ramsay and Silverman, 2013). We found out that, for all the considered stochastic processes, two Functional Principal Components were enough to explain up to 99% of the

variability of the original data. The first Principal Component explains from 97.8 to 98.7% of the total variability, while the second Principal Component explains from 1.2 to 1.3% of the total variability.

Despite we might retain just one Principal Component for each process in view of the high amount of variance accounted for, we decided to include also a second component since, as we will see in the following, the second component brings important clinical meanings.

Let $m(t) = \bar{a}_{i=1}^N L_i(t)$ denote the average compensator for the considered process and let $PC_i(t)$ denote the i th Functional Principal Component. The centered compensators $\hat{L}_i(t) - \bar{L}(t)$ and the Principal Components $PC_i(t)$ correspond, respectively, to the terms $x_i(t)$ and $x(t)$ in Section 2.1.3. Figure 3.14, 3.15, 3.16 and 3.17 show for each process its first two Principal Components in the top panels and $m(t) \pm 10 \cdot PC_i(t)$ in the bottom panels.

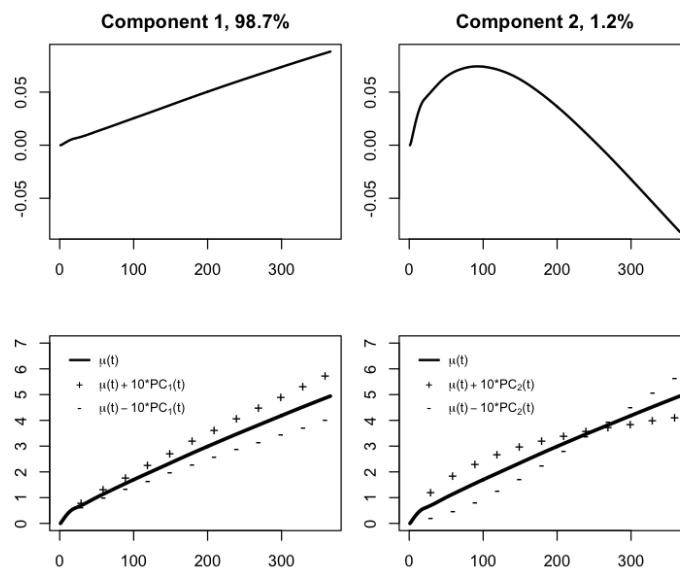


FIGURE 3.14: First two Functional Principal Components of the compensator of the stochastic process describing the purchase of ACE inhibitors. In the top panels, the two Principal Components. In the bottom panels, the average compensator ± 10 times the Principal Components.

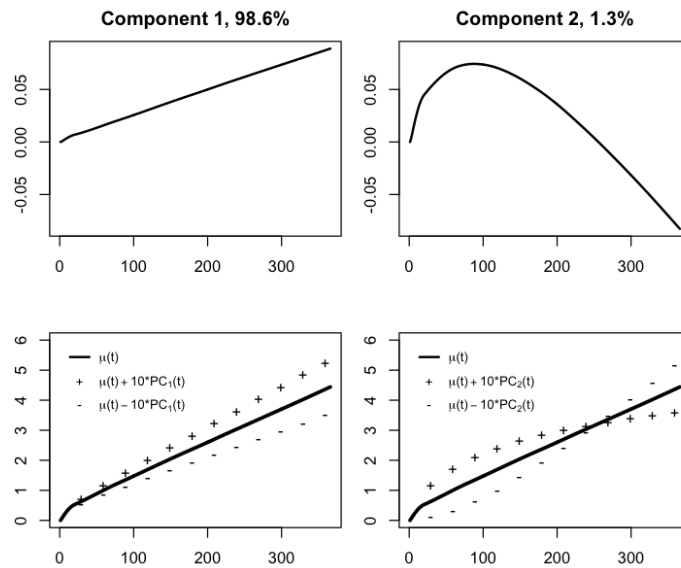


FIGURE 3.15: First two Functional Principal Components of the compensator of the stochastic process describing the purchase of BB agents. In the top panels, the two Principal Components. In the bottom panels, the average compensator $\mu(t)$ and $\mu(t) \pm 10 \times$ the Principal Components.

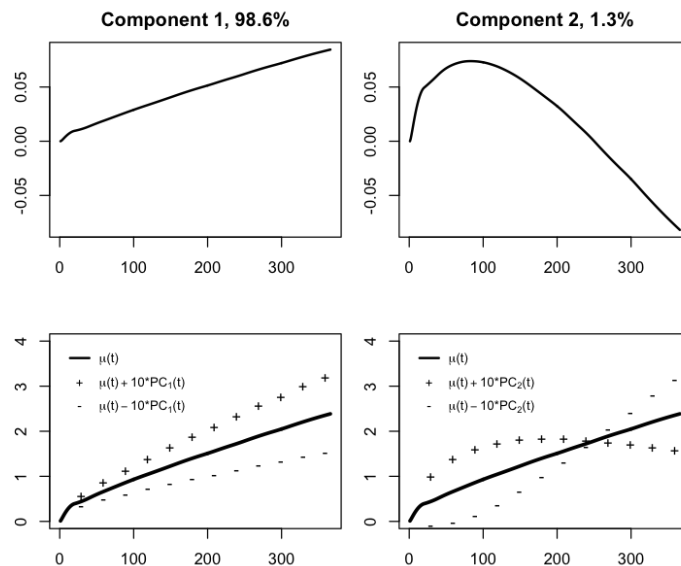


FIGURE 3.16: First two Functional Principal Components of the compensator of the stochastic process describing the purchase of AA agents. In the top panels, the two Principal Components. In the bottom panels, the average compensator $\mu(t)$ and $\mu(t) \pm 10 \times$ the Principal Components.

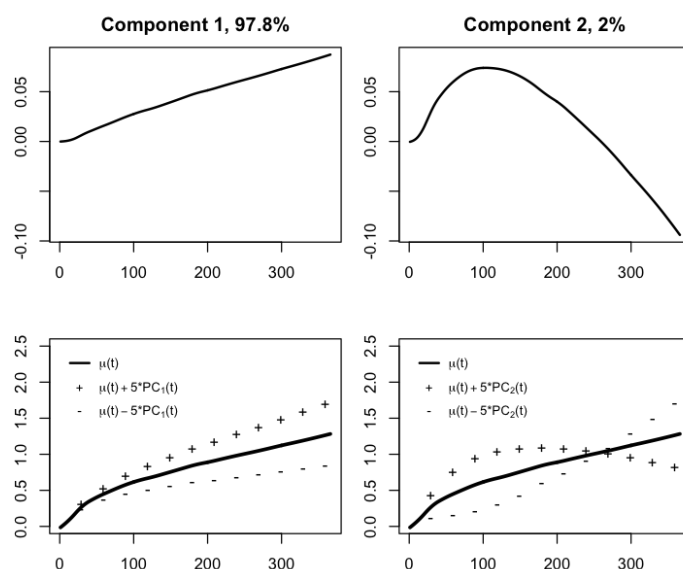


FIGURE 3.17: First two Functional Principal Components of the compensator of the stochastic process describing the purchase of HF hospitalisations. In the top panels, the two Principal Components. In the bottom panels, the average compensator 5 times the Principal Components.

We observe that Principal Components across the four processes have similar shapes. The first Principal Component distinguishes patients with different risks: a patient with a high score of the first Principal Component is likely to experience more events than a patient with a low score. The second Principal Component distinguishes patients with different time distribution of the events: a patient with a high score of the second Principal Component is likely to experience more events in the first months of the year and less events in the last months of the year than a patient with a lower score.

The computed FPCA scores allow us to "summarise" the information contained in the fitted compensators of the considered stochastic processes through the projection of the compensators on a finite dimensional orthonormal basis. We will use these scores in a Cox model to model survival time of patients, as shown in Section 2.1.4.

3.3 Predictive Survival Models

Up to this point we have worked on extracting and summarising the meaning and information content of patients' "clinical history" within the one year

observation period following the discharge from their first HF hospitalisation.

Now let us place ourselves at the end of the observation period. Our aim is to make inference and predict survival of our patient at the end of the follow-up. We want to use information carried by the compensators of clinical history, which we synthesised through FPCA.

Since we are using one year of clinical history for the observation period, we are implicitly conditioning the survival time to be greater than one year. We then set our "new time 0" at the end of the observation period.

The dataset used to make inference of the survival time during the follow-up of patients includes all the covariates described in Table 3.5.

Variable name	Variable description
<i>id</i>	patient identifier (<i>COD_REG</i>)
<i>sex</i>	sex indicator (0 if female, 1 if male)
<i>age_in</i>	age at the end of the follow-up
<i>time_event</i>	registered survival time
<i>status</i>	type of the event indicator (0 if censored, 1 otherwise)
<i>ACE_PC1</i>	score PC_1 for ACE inhibitors
<i>ACE_PC2</i>	score PC_2 for ACE inhibitors
<i>aldosteronics_PC1</i>	score PC_1 for AA agents
<i>aldosteronics_PC2</i>	score PC_2 for AA agents
<i>beta_PC1</i>	score PC_1 for BB agents
<i>beta_PC2</i>	score PC_2 for BB agents
<i>hospitalisation_PC1</i>	score PC_1 for hospitalisations
<i>hospitalisation_PC2</i>	score PC_2 for hospitalisations

TABLE 3.5: Description of the variables in the dataset used to predict survival time of individuals, enriched with the FPCA scores obtained in Section 3.2.

We refer to this dataset as "Short Format" (SF) dataset because, differently from the previous one, it has exactly one row for each patient.

We splitted the SF dataset in two parts, using 70% of the data (3,179 patients) for training survival models and keeping the remaining 30% (1,362 patients) for testing purposes.

In the next Sections we describe the results obtained by applying the functional Cox-type regression model described in Section 2.40 and the Machine

Learning models DeepHit and DRSA described respectively in Sections 2.2.3 and 2.2.5 onto data described in Table 3.5.

For the Cox-type regression model, the introduction of FPCA scores must be understood and interpreted as that of traditional covariates. There is not an equivalent theoretical justification to the usage of FPCA scores as predictors in the DeepHit and DRSA Machine Learning models. However, using FPCA scores as features is an effective technique to introduce information from functional data into a neural network, as shown in Rossi et al., 2007.

3.3.1 Functional Cox-type regression model

The code that we developed to obtain the results presented in this Section is available in Section A.3 of the Appendix.

We modelled the survival time of patients with a standard Cox-type regression model fitted on training data in the Short Format (SF) dataset (Table 3.5). This was possible using dedicated libraries within survival R package (Therneau and Lumley, 2014).

At first we considered many possible combinations of covariates, shown in Figure 3.18. Then, we applied 10 folds cross validation to each of the combinations and we chose the one with the highest Concordance Index (described in Section 2.3), corresponding to the following model.

$$l(t|\mathbf{x}) = l_0(t) \exp\{b_1 \text{age_in} + b_2 \text{beta_PC1} + b_3 \text{hospitalisation_PC1} + b_4 \text{hospitalisation_PC2}\} \quad (3.8)$$

We did not insert the *sex* gender in the models since we noticed it was not significant.

```

***** Cross validation Concordance Probability index *****
|-----|-----|
| C-INDEX | FORMULA |
|-----|-----|
| 66.8 % | age_in + sex" |
| 66.7 % | age_in + ACE_PC1" |
| 66.7 % | age_in + ACE_PC1 + ACE_PC2" |
| 66.6 % | age_in + aldosteronics_PC1" |
| 66.6 % | age_in + aldosteronics_PC1 + aldosteronics_PC2 <----- WORST MODEL" |
| 66.8 % | age_in + beta_PC1" |
| 66.7 % | age_in + beta_PC1 + beta_PC2" |
| 67.5 % | age_in + hospitalisation_PC1" |
| 68.6 % | age_in + hospitalisation_PC1 + hospitalisation_PC2" |
| 68.4 % | age_in + ACE_PC1 + ACE_PC2 + aldosteronics_PC1 + aldosteronics_PC2 + beta_PC1 + beta_PC2 + ... |
| 68.7 % | age_in + ACE_PC1 + beta_PC1 + hospitalisation_PC1 + hospitalisation_PC2" |
| 67.5 % | age_in + ACE_PC1 + beta_PC1 + hospitalisation_PC1" |
| 68.6 % | age_in + ACE_PC1 + aldosteronics_PC1 + beta_PC1 + hospitalisation_PC1 + hospitalisation_PC2" |
| 68.7 % | age_in + beta_PC1 + hospitalisation_PC1 + hospitalisation_PC2 <----- BEST MODEL" |

```

FIGURE 3.18: Cross validated C-indexes for 14 possible sets of covariates for the Cox-type regression model used to predict survival time of HF patients.

We then fitted the Cox-type regression model with the best choice of covariates (Equation 3.8) on the entire SF training data, and we used the fitted model (Table 3.6) to predict survival time of patients belonging to the test set. Using the Breslow estimator (Section 2.1.1) and exploiting Equation 2.18 we reconstructed the survival curves $S_i(t)$ of the patients. We recall that $S_i(t)$ denotes the probability that patient i survives longer than t .

Figure 3.19 shows the predicted survival curves for a sample of 500 patients in the test set using the Cox-type regression model with the coefficients shown in Table 3.6.

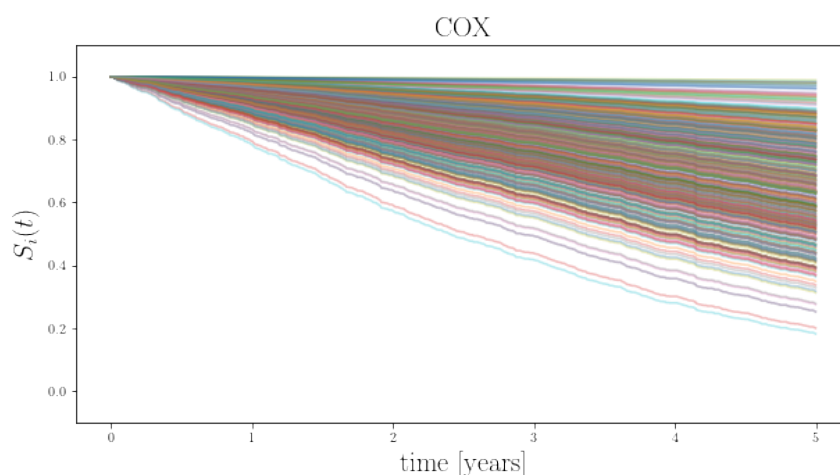


FIGURE 3.19: Survival curves obtained by fitting a Cox-type regression model for a sample of 500 HF patients in the test set.

In order to evaluate the performance of the fitted Cox-type regression model, we computed the Concordance Index on the test set using the terms $\hat{b}^T \mathbf{x}_i$ as measures of risks. We obtained a Concordance Index on the test set equal to 67.56%.

Table 3.6 shows the estimated Hazard Ratios (HR) for the best choice of covariates (Equation 3.8) and the relative CIs.

Variable name	HR ($e^{\hat{b}_k}$)	CI (2.5%)	CI (97.5%)
<i>age_in</i>	1.065	1.0558	1.0744
<i>beta_PC1</i>	0.997	0.9948	0.9992
<i>hospitalisation_PC1</i>	1.020	1.0080	1.0323
<i>hospitalisation_PC2</i>	0.756	0.7039	0.8119

TABLE 3.6: Fitted coefficients of the Cox-type regression model describing the survival time of HF patients.

Looking at the values of the coefficients, we can observe that old patients correctly have a higher risk of dying than younger ones. In particular every year of aging corresponds to a 6.5% increase in the hazard function. The HR relative to *beta_PC1*, being lower than 1 and significant, indicates that BB agents lead to longer life expectancy (0.3% lower hazard function per unit increase of *beta_PC1*). The HR relative to *hospitalisation_PC1*, being greater than 1, indicates that patients that have already experienced many hospitalisations in the past have a higher risk of dying than patients that were hospitalised only a few times.

We found that the *sex* variable is not significant. However *sex* has a 30% positive correlation with the variable *hospitalisation_PC1*, coherently with the fact that male patients are more likely to be hospitalised than female patients (according to the hospitalisation model described in Section 3.1). Furthermore we noticed that removing the variable *hospitalisation_PC1*, the variable *sex* becomes significant, meaning that its effect is not negligible, but is included in the effect of *hospitalisation_PC1*.

Interestingly, the HR relative to the second principal component of the hospitalisations is lower than 1. To interpret this, let us consider two patients *i* and *j*. If, at the beginning of the follow-up year, *i* experienced more hospitalisations than *j* and, towards the end of the year, *j* experienced more hospitalisations than *i*, then *j* has a higher risk of dying than *i*.

Observe that this is probably due to the fact that we are conditioning the patients to survive at least for one year (see Figure 1.1). Patients who had many hospitalisations at the beginning of the year and few hospitalisations in the end of the year probably correspond to the ones who have already experienced a critical phase of the disease and survived from it.

3.3.2 Machine Learning models

In this Section, we illustrate the results obtained by modelling the survival time of patients with the DeepHit and the DRSA Deep Learning models described in Section 2.2.

We recall that these model do not assume any particular form for the underlying stochastic process, thus allowing us to discard the proportionality assumption made by the Cox-type regression model.

For both models we used as predictors all the features of the dataset (Table 3.5) except for the Principal Components of the stochastic process describing the AA agents purchase.

DeepHit

The code that we developed to obtain the results presented in this Section is available in Section A.4 of the Appendix.

We fitted a DeepHit model as in Section 2.2.3 to predict survival time of patients. We coded the model in PYTHON, using the PyTorch framework (Paszke et al., 2017).

We recall that the DeepHit model takes in input the vector of predictors of one patient and returns the estimated probability distribution of its survival time over a discrete time grid, as follows.

$$\mathbf{y} = f_q(\mathbf{x}) \tag{3.9}$$

In Equation 3.9 f_q represents the DeepHit model, \mathbf{x} the vector of patient's predictors and \mathbf{y} the probability distribution of the patient's survival time over the discrete time grid we set.

Since the number of time steps in the grid corresponds to the number of Artificial Neurons in the output layer of the FNN (Section 2.2.2), the chosen time discretisation has an impact in the number of parameters of the model to be fitted. In order to limit this number, we discretised the time into intervals of two months, setting the maximum survival time to 10 years. While

the algorithm did not seem to be too sensitive to the choice of the time interval, we noticed that the performances of the algorithm get worse in terms of estimated concordance index when increasing the maximum survival time. For instance: we obtained a 10% lower concordance index on the test set by setting at 20 years the maximum survival time. This is probably due to the fact that we do not have observations to make such long-term predictions since in the SF dataset the maximum observed survival time is 7 years. Nevertheless it would be wrong to truncate the time grid at 7 years, since the last time step represent the maximum possible survival time.

To set the parameters a and s of equation 2.59 we evaluated the performance on a validation set corresponding to 15% of the training data, using the concordance index as measure of goodness of fit. We did not perform cross-validation due to high computational cost (several days of computation on a common laptop without GPU support).

In Table 3.7 we reported a list of the values set for the hyperparameters. We trained the network using the *Adam* optimizer (Kingma and Ba, 2014) using a learning rate of 10^{-4} and a 60% *dropout* as done by the authors of the DeepHit model (C. Lee et al., 2018). Through the validation process we selected a value of a equal to 1.5 and a value of s equal to 10^4 .

We set the remaining hyperparameters to values commonly accepted in literature for similar FNNs. We set the maximum number of epochs equal to 5,000, the *batch* size to 25 and the number of *early-stopping* epochs (i.e. the maximum number of epochs in which the loss validation function may not decrease) equal to 150.

Hyperparameters	Values
<i>training epochs</i>	5000
<i>batch size</i>	25
<i>optimizer</i>	Adam
<i>learning rate</i>	10^{-4}
<i>early stopping epochs</i>	150
<i>dropout</i>	60%
a	1.5
s	10^4

TABLE 3.7: Hyper-parameters choice for the training of the DeepHit model used to predict survival time of HF patients.

In Figure 3.20 we show the losses obtained during the training phase. The training and the validation losses decrease for a large number of epochs and the network does not overfit the training set, since the validation loss is substantially decreasing (thanks to early stopping). Note that the validation loss has a smoother trend than the training loss. This is due to the fact that predictions on the training set are made using dropout, i.e setting to 0 each Artificial Neuron in the hidden layers with 60% of probability, while predictions on validation set are made in absence of dropout.

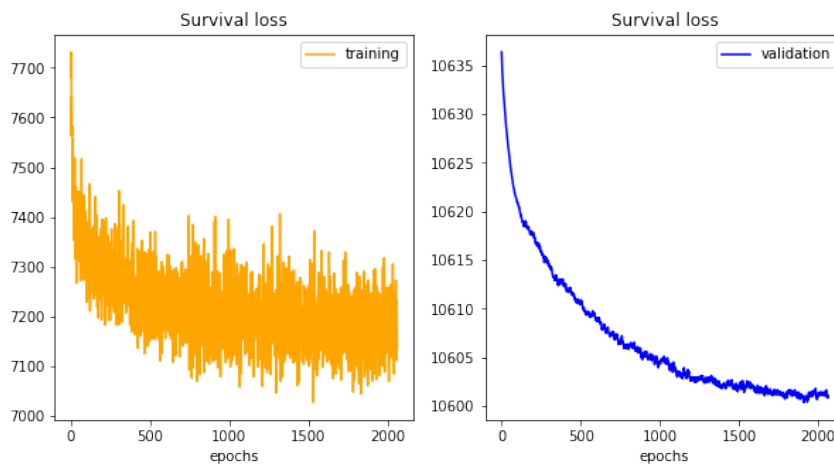


FIGURE 3.20: Loss function of the DeepHit model in the training phase, for each epoch. On the left, the loss function evaluated on the training set (dropout is applied). On the right, the loss function evaluated on the test set.

Finally, we predicted distributions of survival time for patients in the test set and we reconstructed their survival curves (Equation 2.61). Figure 3.21 shows the values of the predicted survival curves in the first 5 years of predictions for a sample of 500 patients.

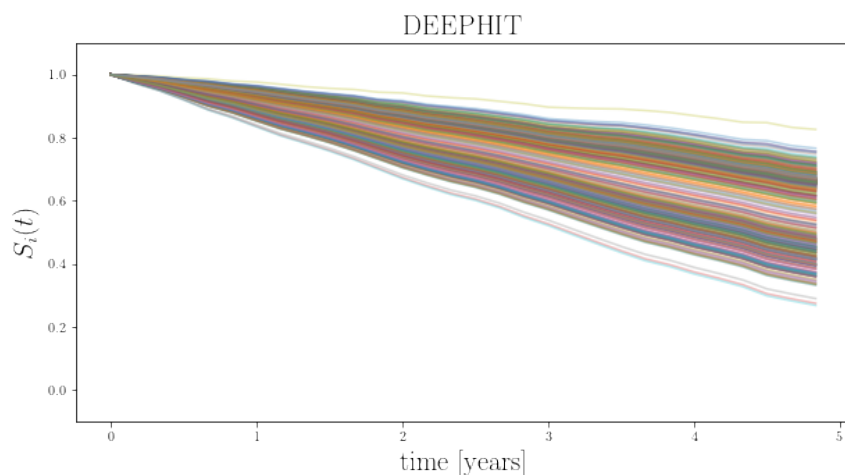


FIGURE 3.21: Predicted survival curves obtained by fitting a Deephit model for a sample of 500 HF patients in the test set.

In order to measure performance, we estimated the concordance probability index using the negative predicted mean survival time $\mathbb{E}[T_i]$ as measure of risk. We obtained an estimate equal to 68.07%.

DRSA

The code that we developed to obtain the results presented in this Section is available in Section A.5 of the Appendix.

We fitted a DRSA model as in Section 2.2.5 to the SF dataset presented at the beginning of Section 3.3 to model survival time of HF patients. We coded also this model in PYTHON, using the PyTorch framework (Paszke et al., 2017).

We remind that the DRSA model assumes a discrete time space for the survival time of patients. Predictions have to be made sequentially in time. The model make predictions sequentially in time, updating the hidden and cell states at each prediction (see Sections 2.2.4 and 2.2.5). At each time step, it receives in input the vector of predictors and the time and it returns the discrete hazard evaluated in that time step. It can be represented as follows:

$$l(t) = f_q(\mathbf{x}, t | \mathbf{h}(t-1), \mathbf{c}(t-1)) \quad (3.10)$$

where f_q represents the DRSA model, \mathbf{x} the vector of predictors for one patient, $l(t)$ the predicted discrete hazard at time t and $\mathbf{h}(t-1)$ and $\mathbf{c}(t-1)$ the hidden and cell state updated in the previous computation.

It is a good practice to never exceed 500 time steps when training LSTM networks (Brownlee, 2017). In order to limit the size of the sequence to be predicted by the model, we discretised the time into intervals of one month, setting a reasonable maximum survival time at 30 years (the median age of patients is about 75 years). It may seem risky to predict such long-term survival times by having observations of up to 7 years. However, as for the DeepHit model, it would be wrong to truncate the time grid at 7 years, since the last time step represents the maximum possible survival time. Furthermore, we observed an improvement in performance in terms of estimated concordance index when the maximum survival time is increased up to 30 years.

We reported in Table 3.8 a list of the values set for the hyperparameters. We trained the network using the *Adam* optimizer (Ren et al., 2018) as done by the authors of the DRSA model (C. Lee et al., 2018). We set a value of a (in Equation 2.71) to 0.75. For the remaining hyperparameters we chose values commonly accepted in literature for similar networks. We set the learning rate to 5^{-5} , the *dropout* to 60%, the maximum number of epochs to 500, the *batch* size to 25 and the number of *early-stopping* epochs (i.e. the maximum number of epochs in which the loss validation function may not decrease) to 20. Finally, we set the dimension of the hidden state equal to 10 times the number of input features.

Hyperparameters	Values
<i>training epochs</i>	500
<i>batch size</i>	25
<i>optimizer</i>	Adam
<i>learning rate</i>	5^{-5}
<i>early stopping epochs</i>	20
<i>dropout</i>	60%
<i>dimensions hidden state</i>	10 $\#fcovariatesg$
a	0.75

TABLE 3.8: Hyper-parameters choice for the training of the DRSA model used to predict survival time of HF patients.

The losses obtained during the training phase, which we reported in Figure 3.22, show an acceptable behaviour. Indeed, training and validation losses have a similar trend, they decrease for a considerable amount of epochs

and the network does not overfit the training set, since the validation loss is substantially decreasing (thanks to early stopping).

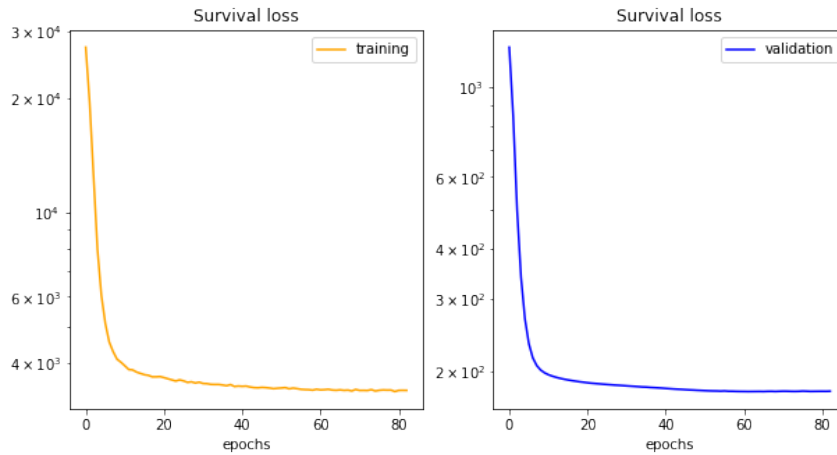


FIGURE 3.22: Loss function of the DeepHit model in the training phase, for each epoch. On the left, the loss function evaluated on the training set (dropout is applied). On the right, the loss function evaluated on the test set.

Finally, we did predictions on the test set. Figure 3.23 shows the predicted discrete hazard functions for a sample of 500 patients in the test set.

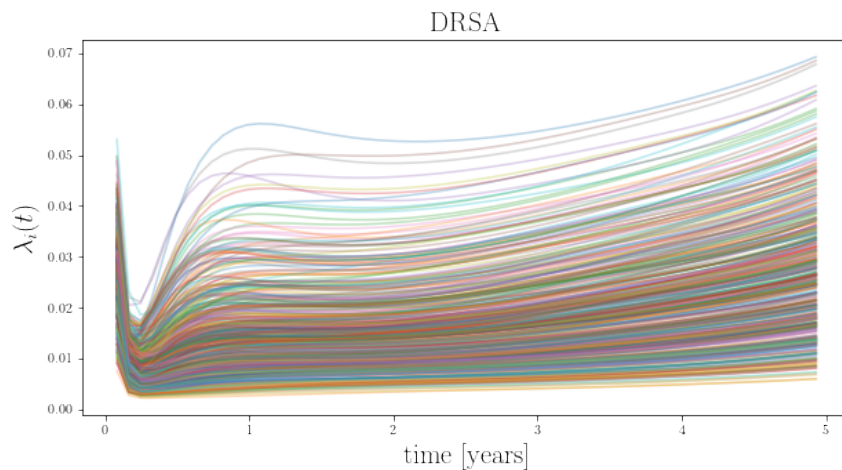


FIGURE 3.23: Discrete hazard functions obtained by fitting a DRSA model for a sample of 500 HF patients in the test set.

We can see a peak that rapidly decreases in the first months. This peak is probably model-related and does not bring biological meaning. The modification of the DRSA model to avoid obtaining this anomaly represents a

possible future development of this thesis.

We detected 182 patients whose hazard after 1 year is greater than after 2 years, and we found that these patients have very low values of first Principal Components for ACE inhibitors (-20 on average) and BB agents (-15 on average). These patients therefore received no or few of these drugs in their observation period. Their lack of cures probably caused a critical phase in their disease within the first year of follow-up.

Using Equation 2.75 we reconstructed the survival curves from the predicted hazards. Figure 3.24 shows the predicted survival curves for the same sample of 500 patients as the one previously used.

FIGURE 3.24: Predicted survival curves obtained by fitting a DRSA model for a sample of 500 HF patients in the test set.

In order to measure performance, we estimated the concordance probability index using the negative predicted mean survival time $\mathbb{E}[T_i]$ as measure of risk. We obtained an estimate equal to 68.58%.

3.4 Performance Comparison

In this Section we compare the models obtained in Sections 3.3.1 (Cox-type regression) and 3.3.2 (DeepHit and DRSA) in terms of Concordance Indexes (described in Section 2.3) and by graphically comparing their predictions.

In Table 3.9 we reported the values of the sample Concordance Indexes for these models.

Model	Sample Concordance Index
Cox-type regression	67.56%
DeepHit	68.07%
DRSA	68.58%

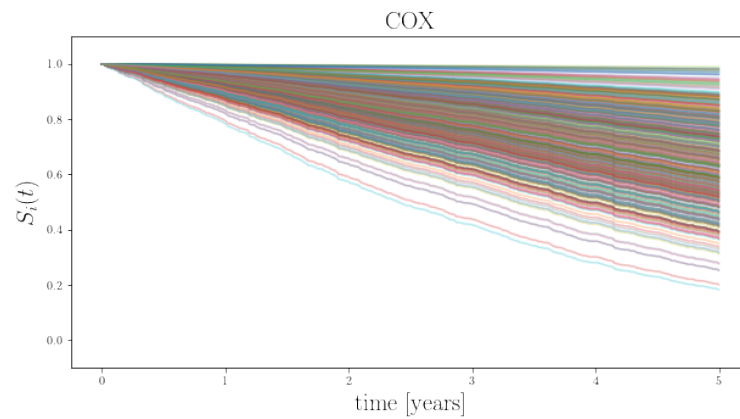
TABLE 3.9: Sample Concordance Indexes for the 1,362 patients in the test set obtained with Cox-type regression, Deephit and DRSA models.

We observe that the DRSA model, which claims to be the state-of-the-art model, obtained the highest estimates for the Concordance Index among the three models, in particular outperforming the Cox-type regression model. To assess if the true Concordance Index of the DRSA model is significantly greater than the Cox-type regression model Concordance Index, we exploited the test developed in Kang, Chen, et al., 2015, which is based on the asymptotic normality of the sample Concordance Index. Exploiting the COMPAREC R package (Kang and Chen, 2015) we performed the following test:

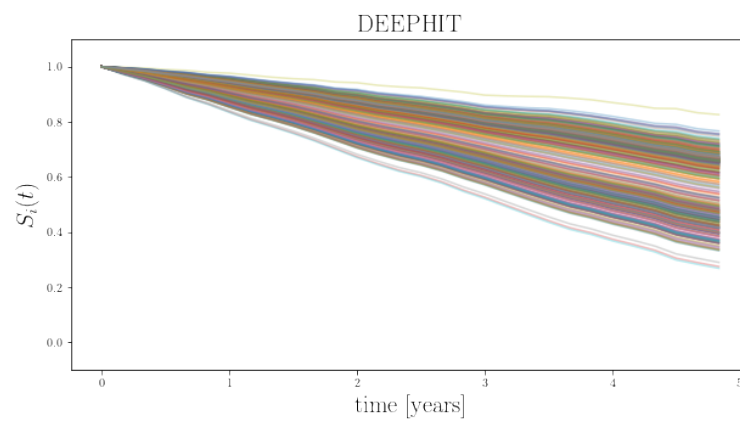
$$\begin{aligned} H_0 : C^{(DRSA)} &= C^{(Cox)} \\ H_1 : C^{(DRSA)} &> C^{(Cox)} \end{aligned} \quad (3.11)$$

Since we obtained a p-value of 0.13%, we can claim that the DRSA model outperformed significantly the Cox-type regression model.

To make predictions comparable, in Sections 3.3.1 and 3.3.2 we converted the output of all the models into survival curves. In Figures 3.19, 3.21 and 3.24 we have already shown these survival curves for a sample of 500 patients out of 1,362 in the test set, considering a time horizon of 5 years for predictions. Figure 3.25 shows the same Figures (3.19, 3.24 and 3.21) together, in order to better visualise the differences among predictions made with different models.



(A) Cox



(B) DeepHit

(C) DRSA

FIGURE 3.25: Survival curves obtained by fitting three different models for a sample of 500 HF patients in the test set. Panel (A): predictions obtained with the Cox-type regression model. Panel (B): predictions obtained with the DeepHit model. Panel (C): predictions obtained with the DRSA model.

The survival curves predicted by the DRSA model show a variability that is higher than for the Cox-type regression model. This may be due to the

proportionality assumption made by the Cox-type regression model, which forces the survival curves to be equal to a certain function to the power of a constant (Equation 2.13 in Section 2.1.1).

The variability of survival curves predicted by the DeepHit model is lower than for the Cox-type regression and the DRSA models. This may be due to the fact that the DeepHit model needs a larger amount of data for training, since it regards the event probability estimation as a pointwise prediction problem.

Finally, we observe that the Cox-type regression model is the only one which predicted survival curves close to 1 after 5 years. Since survival time of individuals with extremely high predicted survival probabilities are mainly censored before 5 years, we cannot assess if this is a good property.

In order to further compare the performances of our models in a personalised prediction setting given the results of the previous paragraphs, we divided patients into four groups based on sex (males, females) and age (under 75, over 75), and we examined the predicted survival functions for each group.

The considered groups resulted then in the following ones: females over 75, females under 75, males over 75 and males under 75.

Figures 3.26, 3.27, 3.28 and 3.29 show the predicted survival curves for the sample of 500 patients previously used, obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models, where the selected group is highlighted in red.

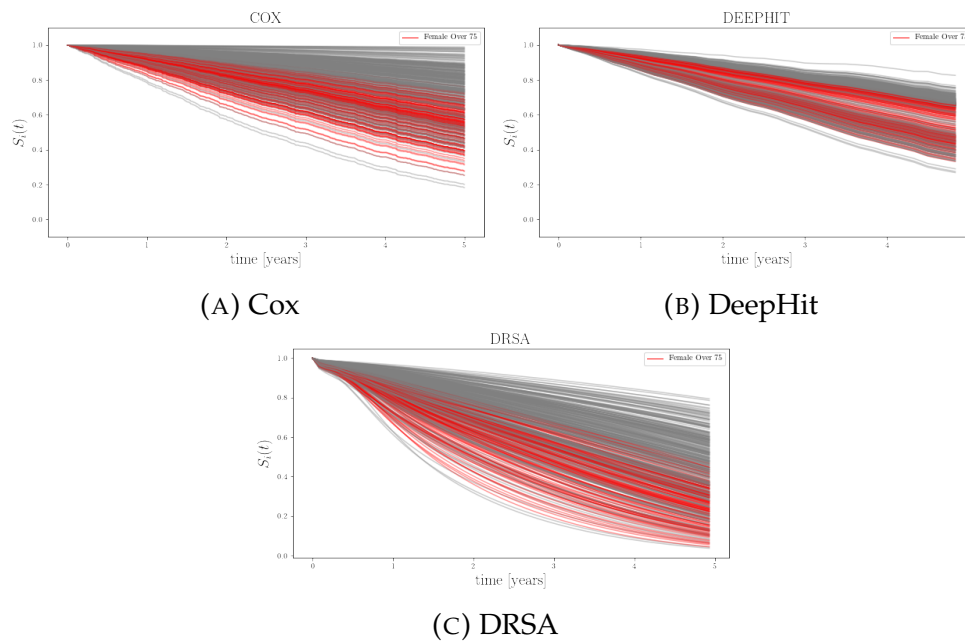


FIGURE 3.26: Predicted survival curves for a sample of 500 HF obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models. The red curves correspond to the group of Females Over 75.

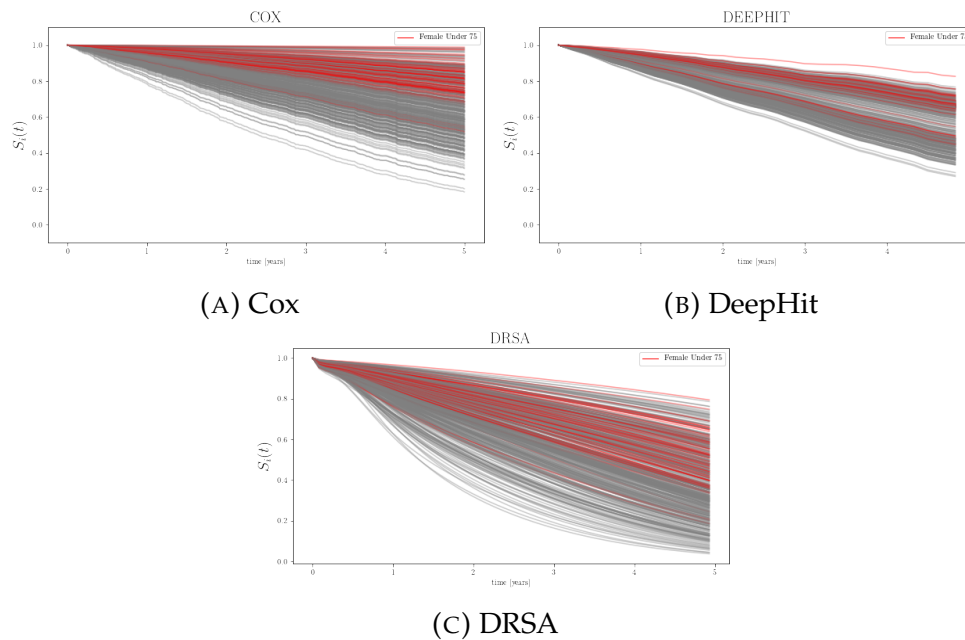


FIGURE 3.27: Predicted survival curves for a sample of 500 HF obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models. The red curves correspond to the group of Females Under 75.

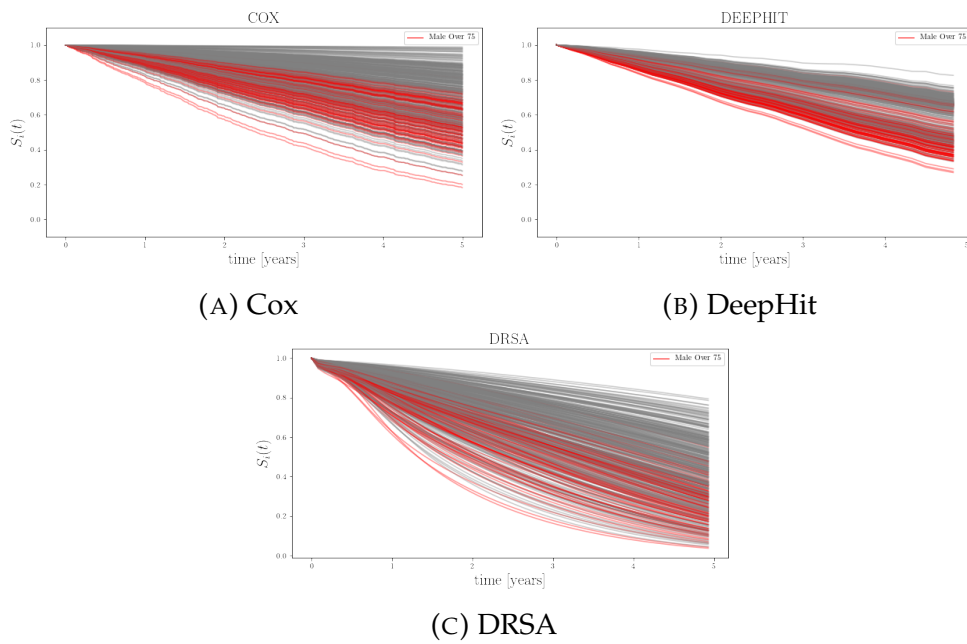


FIGURE 3.28: Predicted survival curves for a sample of 500 HF obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models. The red curves correspond to the group of Males Over 75.

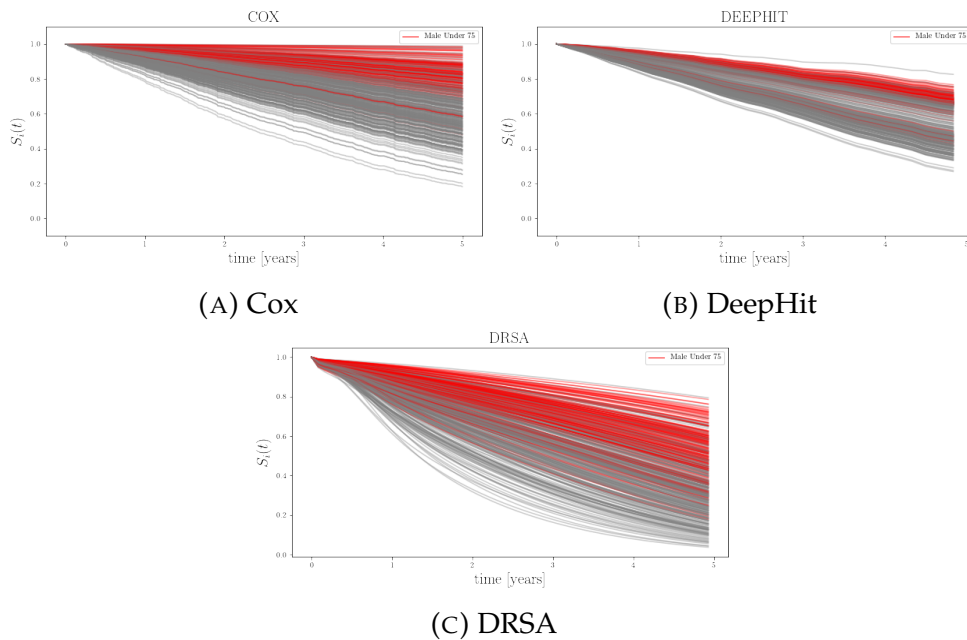


FIGURE 3.29: Predicted survival curves for a sample of 500 HF obtained from the Cox-type regression (A, upper left panel), DeepHit (B, upper right panel) and DRSA (C, bottom panel) models. The red curves correspond to the group of Males Under 75.

Regardless of the different trends, predictions obtained with the three different models seems to be coherent. For all the models we observe a clear and reasonable distinction among survival curves predicted for Under 75 and Over 75 patients, and we do not detect anomalous trends among specific groups.

Conclusions

Within this work, we proposed an effective methodology to extract and summarise information from functional data intended as trajectories of suitable counting processes, plugging them into a Cox-type regression model.

Our technique consists in fitting the compensators (i.e., the predictable part) of stochastic processes describing a dynamic that may influence the survival, summarising them through FPCA, then subsequently including the scores of significant components into a Cox-type regression model. To fit the compensators we used a Cox model for recurrent events with a suitable choice of covariates selected through cross-validation. We then reconstructed the smoothed realisations of the compensators of the process and performed a dimensionality reduction through FPCA.

This methodology allows for a handleable and relatively simple interpretation of the results, and represents one of the first attempts in literature of including such type of time-dependent covariates into a Cox-type regression model.

Two competitors of the Cox-type regression model within a Machine Learning based set of models were also considered and compared with this first method in terms of predictions of survival and/or hazard estimation: DeepHit (a Feed-Forward Neural Network which outputs the probability distribution of survival time over a discrete time grid) and DRSA (a Long-Short Term Memory Neural Network which outputs the discrete hazard at each time step).

We applied these models to a real dataset of patients affected by Heart Failure, whose survival over the years was retrieved through the administrative database of Lombardy Region (Italy), together with their drugs purchases and rehospitalisations over time.

This work extends the methodology proposed in Baraldo et al., [2013](#) relative to the part of the fitting of the compensators of stochastic processes and their summary through FPCA. In particular, we allowed the counting processes to depend on their marks and we proposed a cross-validation framework to select an appropriate set of variables as covariates.

For what concerns survival predictive models, we discovered that the DRSA significantly outperformed the Cox-type regression model in terms of Concordance Index. This is probably due to the flexibility of the DRSA and its no need to assume proportionality or a parametric or semi-parametric form for the hazard.

Our work may result in several possible developments. We are currently developing an R package to formalise and make available our methodology to fit and smooth the compensators and to perform the dimensionality reduction over them through FPCA, starting from a dataset of observed recurrent events.

From a computational point of view, the reconstruction of the compensators may be sped up by coding the relative functions in a compiled language such as C++, or possibly through vectorisation.

The proposed predictive models could be enriched, for instance, by considering comorbidities of patients as covariates or other relevant clinical information, and enlarging the cohort of patients. This would result in benefit also for the deep learning models.

Concerning the latter ones, further investigation on their performance should be considered, in particular for inducing dependence among predictions of different instances.

Appendix A

Code

In this Appendix we show the main parts of the code used to develop this thesis work. The full code is available at <https://github.com/davide-burba/thesis> .

A.1 Fit compensators of stochastic processes

The following functions are used to compute the cumulative hazard in a grid of points.

```

1
2 library(splines)
3
4
5 #' Evaluate cumulative Hazard in a grid of points; returns a dataframe in long format
6 compute_cumulative_hazard = function(model,
7                                     sel_df,
8                                     smoothed_baseline,
9                                     times,
10                                    verbose = FALSE){
11   # Evaluate Lambda on a grid (NB: Lambda0_fun(t) == Lambda0s_value[t+1])
12   Lambda0s_value = .Lambda0_fun(times, smoothed_baseline)
13   # Compute constant at times coefficients
14   patient_coefficients = .compute_coefficients_ck(sel_df, model, verbose)
15   # Compute daily deltas of cumulative Hazard and sum it up
16   cumulative_hazard = .compute_cumulative_hazard(patient_coefficients, Lambda0s_value, verbose)
17   return(cumulative_hazard)
18 }
19
20
21 #' Compute coefficient  $ck = \exp\{\beta x_i(t_k)\}$  at each interval for each patient
22 .compute_coefficients_ck = function(sel_df, model, verbose = FALSE){
23   if(verbose){print('Computing coefficients ck')}
24   # take coefficients beta
25   name_coefficients = names(model$coefficients)
26   coefficients = model$coefficients
27
28   patient_coefficients = NULL
29   patient_ids = unique(sel_df$id)
30   pb <- txtProgressBar(min = 0, max = length(patient_ids), style = 3)
31   # compute coefficients for each individual
32   for(i in 1:length(patient_ids)){
33     setTxtProgressBar(pb, i)
34     patient_id = patient_ids[i]
35     patient_df = sel_df[id == patient_id,]
36     ck = c()
37     # compute coefficients for each time interval (start, stop]
38     for(k in patient_df$time){
39       beta_times_xik = sum(patient_df[, ..name_coefficients][k+1] * coefficients)
40       ck = c(ck, exp(beta_times_xik))
41     }

```

```

42   patient_coefficients = rbind(patient_coefficients, data.frame(id = patient_id,
43                                                                 k = patient_df$Nm,
44                                                                 tk = patient_df$start,
45                                                                 ck))
46   }
47   cat("\n")
48   return(data.table(patient_coefficients))
49 }
50
51
52 #' Compute daily deltas of cumulative Hazard and it sums it up; returns a dataframe in long format
53 .compute_cumulative_hazard = function(patient_coefficients, Lambda0s_value, verbose = FALSE){
54   if(verbose){print('Computing cumulative Hazard on the grid')}
55   cumulative_hazard = NULL
56   patient_ids = unique(patient_coefficients$id)
57   pb <- txtProgressBar(min = 0, max = length(patient_ids), style = 3)
58   for(i in 1:length(patient_ids)){
59     setTxtProgressBar(pb, i)
60     patient_id = patient_ids[i]
61     # treat apart interval [ 0.5,0)
62     ck = patient_coefficients[id == patient_id & tk < 0]$ck
63     deltas = c(ck Lambda0s_value[1])
64     # compute deltas for each day
65     for(t in 1:max(times)){
66       tmp = patient_coefficients[id == patient_id & tk < t]$ck
67       ck = tmp[length(tmp)]
68       # deltas: Lambda0s(t) Lambda0s(t-1)
69       deltas = c(deltas, ck (Lambda0s_value[t+1] Lambda0s_value[t]))
70     }
71     cumulative_hazard = rbind(cumulative_hazard, data.frame(id = patient_id,
72                                                           time = times,
73                                                           cumhaz = cumsum(deltas)))
74   }
75   cat("\n")
76   return(cumulative_hazard)
77 }
78
79
80 # Evaluate smoothed baseline cumulative hazard
81 .Lambda0_fun <- function(t, smoothed_baseline){
82   return(.basis(t, smoothed_baseline$knots) %% smoothed_baseline$coef)
83 }
84
85
86 # Evaluate a b spline basis on a vector of points x
87 .basis <- function(x, knots, deg=2){
88   return(bs(x,
89            knots=knots[2:(length(knots) - 1)],
90            degree=deg,
91            Boundary.knots=c(knots[1], knots[length(knots)]),
92            intercept=TRUE))
93 }

```

The following script shows the steps used to fit the compensators for the beta-blocking process. The code for other processes is analogous. This script was originally structured as a jupyter notebook (Kluyver et al., 2016).

```

1  require(data.table)
2  require(ggplot2)
3  require(survival)
4  library(cobs) # Package for constrained L1 B splines
5  require(latex2exp)
6
7  options(repr.plot.width=6, repr.plot.height=4)
8
9
10 path = '/Users/davide/Documents/universita/tesi/src/fit_compensators/'
11 setwd(path)
12 source('compute_cumulative_hazards.R')
13 load('../data/preprocessed_data_beta.RData')
14
15 sel_df = beta_df
16
17 # preprocessed data

```

```

18 head(sel_df, n = 8)
19
20 set.seed(123)
21 #Randomly shuffle the patients
22 patients = sample(unique(sel_df$id))
23 #Create 10 equally size folds
24 folds <- cut(seq(1, length(patients)), breaks=10, labels=FALSE)
25
26 #Perform 10 fold cross validation
27 scores = NULL
28 pb <- txtProgressBar(min = 0, max = 10, style = 3)
29 for(i in 1:10){
30   setTxtProgressBar(pb, i)
31   #Segment patients by fold
32   validIndexes <- which(folds==i, arr.ind=TRUE)
33   valid_patients <- patients[validIndexes]
34   train_patients <- patients[ -validIndexes]
35   # Split train, valid
36   train = sel_df[id %in% train_patients]
37   valid = sel_df[id %in% valid_patients]
38
39   # fit
40   models = list(
41     coxph(Surv(start, stop, status)~ age_in + Nm + cluster(id), data = train),
42     coxph(Surv(start, stop, status)~ age_in + y + cluster(id), data = train),
43     coxph(Surv(start, stop, status)~ age_in + Nm + y + cluster(id), data = train),
44     coxph(Surv(start, stop, status)~ age_in + Nm:y + cluster(id), data = train),
45     coxph(Surv(start, stop, status)~ age_in + Nm + Nm:y + cluster(id), data = train),
46     coxph(Surv(start, stop, status)~ age_in + y + Nm:y + cluster(id), data = train),
47     coxph(Surv(start, stop, status)~ age_in + Nm + y + Nm:y + cluster(id), data = train),
48
49     # log(Nm+1)
50     coxph(Surv(start, stop, status)~ age_in + log(Nm+1) + cluster(id), data = train),
51     coxph(Surv(start, stop, status)~ age_in + log(Nm+1) + y + cluster(id), data = train),
52     coxph(Surv(start, stop, status)~ age_in + log(Nm+1):y + cluster(id), data = train),
53     coxph(Surv(start, stop, status)~ age_in + log(Nm+1) + log(Nm+1):y + cluster(id), data = train),
54     coxph(Surv(start, stop, status)~ age_in + y + log(Nm+1):y + cluster(id), data = train),
55     coxph(Surv(start, stop, status)~ age_in + log(Nm+1) + y + log(Nm+1):y + cluster(id), data = train),
56
57     # log(y+1)
58     coxph(Surv(start, stop, status)~ age_in + log(y+1) + cluster(id), data = train),
59     coxph(Surv(start, stop, status)~ age_in + Nm + log(y+1) + cluster(id), data = train),
60     coxph(Surv(start, stop, status)~ age_in + Nm:log(y+1) + cluster(id), data = train),
61     coxph(Surv(start, stop, status)~ age_in + Nm + Nm:log(y+1) + cluster(id), data = train),
62     coxph(Surv(start, stop, status)~ age_in + y + log(y+1) + Nm:log(y+1) + cluster(id), data = train),
63     coxph(Surv(start, stop, status)~ age_in + Nm + log(y+1) + Nm:log(y+1) + cluster(id), data = train),
64
65     # log(Nm+1), log(y+1)
66     coxph(Surv(start, stop, status)~ age_in + log(Nm+1) + cluster(id), data = train),
67     coxph(Surv(start, stop, status)~ age_in + log(y+1) + cluster(id), data = train),
68     coxph(Surv(start, stop, status)~ age_in + log(Nm+1) + log(y+1) + cluster(id), data = train),
69     coxph(Surv(start, stop, status)~ age_in + log(Nm+1):log(y+1) + cluster(id), data = train),
70     coxph(Surv(start, stop, status)~ age_in + log(Nm+1) + log(Nm+1):log(y+1) + cluster(id), data = train),
71     coxph(Surv(start, stop, status)~ age_in + log(y+1) + log(Nm+1):log(y+1) + cluster(id), data = train),
72     coxph(Surv(start, stop, status)~ age_in + log(Nm+1) + log(y+1) + log(Nm+1):log(y+1) + cluster(id), data =
       train)
73   )
74   # predict, evaluate
75   fold_scores = c()
76   for(model in models){
77     # evaluate martingale residuals at validation events
78     valid_model = coxph(model$formula, data = valid, init = model$coefficients, iter.max = 0)
79     res = residuals(valid_model, type = "martingale")
80     score = mean(abs(res)) # mean absolute residual
81     fold_scores = c(fold_scores, score)
82   }
83   scores = rbind(scores, fold_scores)
84 }
85 rownames(scores) = 1:10
86 colnames(scores) = names(models)
87
88 # summarize CV scores
89 mean_scores = colMeans(scores)
90 best_score = min(mean_scores)
91 worst_score = max(mean_scores)
92
93 # print cv scores

```

```

94 print('                                Cross validation mean absolute Martingale residual                                ')
95 cat('\n')
96 print('|                                |                                |                                |')
97 print('| MEAN ABS. RESIDUAL |                                FORMULA                                |')
98 print('|                                |                                |                                |')
99 for(i in 1:length(models)){
100   score = mean_scores[i]
101   if (score != best_score & score != worst_score){
102     print(paste('|',round(mean_scores[i],3),'|',models[[i]]$formula[3]))
103   } else if (score == best_score){
104     print(paste('|',round(mean_scores[i],3),'|',models[[i]]$formula[3], '
105     < BEST MODEL'))
106   } else {
107     print(paste('|',round(mean_scores[i],3),'|',models[[i]]$formula[3], '
108     < WORST MODEL'))
109   }
110 }
111 # in order to use some function in the following cells , we build features corresponding to the transformed
112 # variables
113 sel_df[, 'logp1_Nm' := log(Nm+1)]
114 sel_df[, 'logp1_y' := log(y+1)]
115 sel_df[, 'logp1_Nm_times_logp1_y' := log(Nm+1) * log(y+1)]
116 model = coxph(Surv(start, stop, status)~ age_in + logp1_Nm + logp1_y + logp1_Nm_times_logp1_y + cluster(id), data =
117   sel_df)
118 # get baseline
119 bh = basehaz(model, centered = FALSE)
120 t <- bh$time
121 Lambda0 <- bh$hazard
122 # Smooth version of Lambda0
123 Lambda0S <- cobs(t, Lambda0, constraint=c("increase"), pointwise=matrix(c(0, 0.5, 0), nrow=1), nknots=20, lambda=0,
124   toler.kn=0)
125 # Comparison between basic and smooth estimate
126 plot(Lambda0S$x, Lambda0S$y, type="l", main="", ylab="Baseline cumulative hazard", xlab="time [days]", col = '
127   red')
128 points(t, Lambda0, type="s", lty=2, col = 'blue')
129 legend(280, 0.9, legend=c(TeX("$\\tilde{\\Lambda}_0$"), TeX("$\\hat{\\Lambda}_0$")), col=c("red", "blue"), lty=1:2,
130   cex=1, y.intersp=3, box.lty=0)
131 # We evaluate cumulative hazards in a grid of days
132 times <- seq(0, 365, by=1)
133 cumulative_hazard = compute_cumulative_hazard(model, sel_df, Lambda0S, times, verbose = TRUE)
134 # select a sample to plot
135 sample_patients = sample(unique(sel_df$id), 500)
136 sample = cumulative_hazard[cumulative_hazard$id %in% sample_patients,]
137 sample_N = sel_df[id %in% sample_patients,]
138 bh['id'] = 0
139 # plot
140 ggplot(sample, aes(x= time, y=cumhaz, group = factor(id), color=factor(id), alpha = 0.1)) +
141   geom_line() +
142   xlab('time[days]') +
143   ylab(TeX("$\\hat{\\Lambda}_i(t)$")) +
144   theme(legend.position="none", plot.title = element_text(hjust = 0.5))
145 # plot also realization
146 ggplot(sample, aes(x= time, y=cumhaz, group = factor(id), color=factor(id), alpha = 0.1)) +
147   geom_line() +
148   geom_step(data = sample_N, aes(x= start, y=Nm, group = factor(id), color=factor(id))) +
149   ggtitle(paste(expression(Lambda[i](t)), ' and N_i(t)')) +
150   theme(legend.position="none", plot.title = element_text(hjust = 0.5))
151 # save
152 save(cumulative_hazard, file = '../data/cumulative_hazards_beta.RData')
153 load('../data/cumulative_hazards_beta.RData')
154 # compute dataset of daily realizations (only once)

```

```

164 #source('compute_daily_realizations.R')
165 #daily_realizations_beta = compute_daily_realizations(sel_df)
166 #save(daily_realizations_beta, file = '../data/daily_realizations_beta.RData')
167
168 # load dataset of daily realizations
169 load('../data/daily_realizations_beta.RData')
170 daily_realizations = daily_realizations_beta
171
172 # compute residuals
173 residuals = cbind(cumulative_hazard, Nt = daily_realizations$Nt)
174 residuals['residuals'] = residuals$cumhaz - residuals$Nt
175 residuals = data.table(residuals)
176
177 # compute mean residuals
178 mean_residuals = residuals[,list('residuals' = mean(residuals)), by = 'time']
179
180 # select sample of patients to plot
181 sample_patients = sample(unique(sel_df$id),500)
182 sample = residuals[id %in% sample_patients,]
183
184 # plot
185 ggplot(data = mean_residuals, aes(x= time, y=residuals)) +
186   geom_line() +
187   geom_line(data = sample, aes(x= time, y=residuals, group = factor(id), color=factor(id), alpha = 0.1)) +
188   geom_line(data = mean_residuals, aes(x= time, y=residuals)) +
189   xlab('time[days]') +
190   ylab(TeX("$\\hat{W}_i(t) = \\hat{\\Lambda}_i(t) - N_i(t)")) +
191   theme(legend.position="none", plot.title = element_text(hjust = 0.5))

```

A.2 FPCA

The following script shows the steps used to summarise the compensators for the beta-blocking process through FPCA. The code for other processes is analogous. This script was originally structured as a jupyter notebook (Kluyver et al., 2016).

```

1
2 require(data.table)
3 require(ggplot2)
4 require(dplyr)
5 require(latex2exp)
6
7 options(repr.plot.width=6, repr.plot.height=5)
8
9 path = '/Users/davide/Documents/universita/tesi/src/fpca'
10 setwd(path)
11
12 # get cumulative Hazard and its mean
13 load('../data/cumulative_hazards_beta.RData')
14 cumulative_hazard = data.table(cumulative_hazard)
15 mean_hazard = cumulative_hazard[,list('cumhaz' = mean(cumhaz)), by = 'time']
16
17 # reformat cumulative Hazard in a matrix format
18 cumulative_hazard_matrix = list()
19 patient_ids = unique(cumulative_hazard$id)
20 pb <- txtProgressBar(min = 0, max = length(patient_ids), style = 3)
21 for(i in 1:length(patient_ids)){
22   setTxtProgressBar(pb, i)
23   patient_id = patient_ids[i]
24   cumulative_hazard_matrix[[patient_id]] = cumulative_hazard[id == patient_id, 'cumhaz']
25 }
26 cumulative_hazard_matrix = bind_cols(cumulative_hazard_matrix)
27 colnames(cumulative_hazard_matrix) = as.character(patient_ids)
28
29 # number of rows: time
30 # number of columns: number of patients
31 dim(cumulative_hazard_matrix)
32

```

```

33 nc <- dim(cumulative_hazard_matrix)[2]
34 m <- dim(cumulative_hazard_matrix)[1]
35 h <- 1
36
37 # Functional data considered in FPCA.
38 matplot(cumulative_hazard_matrix, type="l", xlab="time (days)", ylab="cumulative hazard", main="Reconstructed hazard
  functions")
39 mu <- apply(cumulative_hazard_matrix, MARGIN=1, "mean")
40 points(mean_hazard$cumhaz, type="l", lwd=3)
41
42 # Eigenvalues and eigenfunctions
43 pca <- prcomp(t(cumulative_hazard_matrix))
44 efuncs <- pca$rotation sqrt(1/h)
45 evalues <- pca$sdev^2 h
46
47 # Principal components diagnostics
48
49 # Eigenvalues
50 sum.evalues <- sum(evalues)
51 percent.var <- evalues/sum.evalues
52 # Choose the minimum K1 such that the sum of eigenvalues (variances) is at least 99% of total
53 K <- min(which(cumsum(percent.var) >= 0.99))
54 # Summary table
55 evtbale <- rbind(evalues[1:K], 100 evalues[1:K]/sum.evalues, 100 cumsum(evalues[1:K])/sum.evalues)
56 rownames(evtbale) <- c("eigenvalue", "% variance", "cumulated % variance")
57 colnames(evtbale) <- paste("component", 1:K)
58 evtbale
59
60 # Scree plots:
61 par(mfrow=c(1,2))
62 plot(evalues[1:K], xlab="", ylab="", main="eigenvalues", type="b")
63 plot(evtbale[3,], xlab="", ylab="", main="% of variance", type="b", ylim=c(0,100))
64 abline(99,0, lty=2)
65
66 # Eigenfunctions
67 # Sign chosen to have positive integral, for ease of interpretation
68 for(i in 1:K) {
69   tempfun <- efuncs[,i]
70   tempsign <- sum(tempfun)
71   efuncs[,i] <- ifelse(tempsign < 0, 1,1) tempfun
72 }
73
74 time = 0:366
75 # Plot of first k eigenfunctions
76 k=K
77 par(mfcol=c(2,k), mar=c(2.5,2,2.5,1))
78 ylim <- range(efuncs[,1:k])
79 ylim2 <- c(0,6)
80 for(i in 1:k) {
81   v1 <- efuncs[,i]
82   plot(v1, type="l", ylab="", xlab="time (days)", ylim=ylim, lwd=2, main=paste("Component ", i, ", ", 100 round(
  percent.var[i,3], "%", sep=""))
83   text(2, 0.95, "time (days)")
84   #v2 <- v1 sqrt(evalues[i])
85   v2 <- v1 10
86   wided <- ((1:m)%%30)==0
87   plot(mu, type="l", ylim=ylim2, lwd=3)
88   points(time[wided], mean_hazard$cumhaz[wided] + v2[wided], lwd=3, pch="+")
89   points(time[wided], mean_hazard$cumhaz[wided] v2[wided], lwd=3, pch=" ")
90
91   legend(10,6, # where
92     legend=c(TeX('$\mu(t)$'), TeX(paste('$\mu(t) + ', '10 PC_', i, '(t)$', sep = '')), TeX(paste('$\mu(t) ',
  '10 PC_', i, '(t)$', sep = ''))), # text
93     lty=c(1,NA,NA), lwd = c(2,1,1), pch = c(NA, '+', ' '), # symbols
94     cex=0.8, y.intersp=2.5, box.lty=0 # other options
95   )
96 }
97
98 # Scores on the first K principal components
99 score <- NULL
100 for(i in 1:K){
101   score <- cbind(score, (as.matrix(t(cumulative_hazard_matrix mean_hazard$cumhaz)) %% cbind(efuncs[,i]) h)
102 }
103
104 options(repr.plot.width=10, repr.plot.height=5)
105
106 # Projection on first 2 P.C.

```

```

107 score1 <- score[,1]
108 score2 <- score[,2]
109 proj1 <- apply(X=cbind(score1),FUN=``,MARGIN=1,y=efuncs[,1]) + apply(X=cbind(score2),FUN=``,MARGIN=1,y=efuncs
    [,2])+mu
110 layout(cbind(1,2))
111 ylim=range(c(range(cumulative_hazard_matrix),range(proj1)))
112 matplot(proj1, type="l",xlab="time (days)", ylab="cumulative hazard", main=paste("projection of cumulative
    hazard on first 2 p.c.", sep=" "), ylim=ylim)
113 matplot(cumulative_hazard_matrix, type="l", xlab="time (days)", ylab="cumulative hazard", main="original dataset
    ", ylim=ylim)
114
115 # reformat score fpca
116 patients = rownames(score)
117 rownames(score) = NULL
118 colnames(score) = c('PC1','PC2')
119 score = data.table('id'=patients ,score)
120
121 head(score)
122
123 score_beta = score
124 save(score_beta, file = '../data/score_fpca_beta.RData')

```

A.3 Functional Cox-type regression model

The following script illustrates the steps we ensued to select the features, fit the model and predict survival times of HF patients with the Cox-type regression model. This script was originally structured as a jupyter notebook (Kluyver et al., 2016).

```

1
2 require(data.table)
3 require(ggplot2)
4 require(survival)
5 library(corrplot)
6
7 options(repr.plot.width=4, repr.plot.height=4)
8
9 path = '/Users/davide/Documents/universita/tesi/src/survival_process/'
10 setwd(path)
11
12 load('../data/main_process_preprocessed_data.RData')
13
14 head(new_df)
15
16 dim(new_df)
17
18 cols = c('age_in', 'ACE_PC1', 'ACE_PC2', 'aldosteronics_PC1', 'aldosteronics_PC2', 'beta_PC1', 'beta_PC2',
    'hospitalisation_PC1', 'hospitalisation_PC2')
19 corrplot(cor(cbind(sexM = ifelse(new_df$sex == 'M',1,0),new_df[,...cols])),type = "upper")
20
21 set.seed(143)
22 #Randomly shuffle the observations
23 new_df = new_df[sample(1:dim(new_df)[1]),]
24 #Create 10 equally size folds
25 folds <- cut(seq(1,dim(new_df)[1]),breaks=10,labels=FALSE)
26
27 #Perform 10 fold cross validation
28 scores = NULL
29 pb <- txtProgressBar(min = 0, max = 10, style = 3)
30 for(i in 1:10){
31   setTxtProgressBar(pb, i)
32   #Segment data by fold
33   validIndexes <- which(folds==i, arr.ind=TRUE)
34   valid = new_df[validIndexes,]
35   train = new_df[ -validIndexes,]
36
37   # fit
38   models = list(

```

```

39 coxph(Surv(time_event, status)~ age_in + sex , data = train),
40 # single process effect, 1 or 2 Principal components
41 coxph(Surv(time_event, status)~ age_in + ACE_PC1, data = train),
42 coxph(Surv(time_event, status)~ age_in + ACE_PC1 + ACE_PC2, data = train),
43 coxph(Surv(time_event, status)~ age_in + aldosteronics_PC1, data = train),
44 coxph(Surv(time_event, status)~ age_in + aldosteronics_PC1 + aldosteronics_PC2, data = train),
45 coxph(Surv(time_event, status)~ age_in + beta_PC1, data = train),
46 coxph(Surv(time_event, status)~ age_in + beta_PC1 + beta_PC2, data = train),
47 coxph(Surv(time_event, status)~ age_in + hospitalisation_PC1, data = train),
48 coxph(Surv(time_event, status)~ age_in + hospitalisation_PC1 + hospitalisation_PC2, data = train),
49 # all together
50 coxph(Surv(time_event, status)~ age_in + ACE_PC1 + ACE_PC2 + aldosteronics_PC1 + aldosteronics_PC2
51 + beta_PC1 + beta_PC2 + hospitalisation_PC1 + hospitalisation_PC2, data = train),
52 coxph(Surv(time_event, status)~ age_in + ACE_PC1 + beta_PC1 +
53 hospitalisation_PC1 + hospitalisation_PC2, data = train),
54 coxph(Surv(time_event, status)~ age_in + ACE_PC1 + beta_PC1 +
55 hospitalisation_PC1, data = train),
56 coxph(Surv(time_event, status)~ age_in + ACE_PC1 + aldosteronics_PC1 + beta_PC1 +
57 hospitalisation_PC1 + hospitalisation_PC2, data = train),
58 coxph(Surv(time_event, status)~ age_in + beta_PC1 +
59 hospitalisation_PC1 + hospitalisation_PC2, data = train)
60
61 )
62 # predict, evaluate
63 fold_scores = c()
64 for(model in models){
65   prediction = predict(model, newdata = valid)
66   # evaluate concordance probability
67   score = survConcordance(Surv(time_event, status) ~ prediction, data = valid)$concordance[[1]]
68   fold_scores = c(fold_scores, score)
69 }
70 scores = rbind(scores, fold_scores)
71 }
72 rownames(scores) = 1:10
73 colnames(scores) = names(models)
74
75 # summarize CV scores
76 mean_scores = colMeans(scores)
77 best_score = max(mean_scores)
78 worst_score = min(mean_scores)
79
80 # print cv scores
81 print('          Cross validation Concordance Probability index          ')
82 cat('\n')
83 print('          |          |')
84 print('          C INDEX |          FORMULA          |')
85 print('          |          |')
86 for(i in 1:length(models)){
87   score = mean_scores[i]
88   if(score != best_score & score != worst_score){
89     print(paste('          ', round(mean_scores[i] 100,1), '%          | ', models[[i]]$formula[3]))
90   } else if (score == best_score){
91     print(paste('          ', round(mean_scores[i] 100,1), '%          | ', models[[i]]$formula[3], ' <
92           BEST MODEL'))
93   } else {
94     print(paste('          ', round(mean_scores[i] 100,1), '%          | ', models[[i]]$formula[3], ' <
95           WORST MODEL'))
96   }
97 }
98 best_model = coxph(Surv(time_event, status)~ age_in + beta_PC1 + hospitalisation_PC1 + hospitalisation_PC2, data
99 = new_df)
100 confint(best_model)
101 load('.../data/main_process_preprocessed_data_test.RData')
102
103 prediction = predict(best_model, newdata = test_df)
104 test_score = survConcordance(Surv(time_event, status) ~ prediction, data = test_df)$concordance[[1]]
105
106 print(paste('Concordance Index on test set:', round(test_score 100,1), '%'))
107
108 write.csv(prediction, file = '.../data/predictions_cox.csv', row.names = FALSE)
109
110 bh = basehaz(best_model, centered = FALSE)
111
112 options(repr.plot.width=6, repr.plot.height=4)

```

```

113 plot(bh$time/365,bh$hazard, type = 'l',xlab = 'time [years]',ylab = 'Hazard', main = 'Baseline Cumulative Hazard
114 ')
115 surv_curves = survfit(best_model,test_df,se.fit = FALSE)$surv
116 write.csv(surv_curves, file = '../data/full_predictions_cox.csv',row.names = FALSE)
117
118 # plot a sample
119 sample_ids = sample(1:dim(test_df)[1],100)
120 surv_sample = survfit(best_model,test_df[sample_ids,],se.fit = FALSE)
121 plot(surv_sample)

```

A.4 DeepHit

The following classes are used for the DeepHit model.

```

1
2 import torch
3 import torch.nn as nn
4 import torch.nn.functional as F
5 from torch.utils.data import Dataset
6 import numpy as np
7
8 class DeepHit(nn.Module):
9     """DeepHit network architecture
10     """
11     def __init__(self,n_features,n_times):
12         super(DeepHit, self).__init__()
13         self.fc1 = nn.Linear(n_features, 3*n_features)
14         self.fc2 = nn.Linear(3*n_features, 5*n_features)
15         self.fc3 = nn.Linear(5*n_features, 3*n_features)
16         self.output = nn.Linear(3*n_features, n_times)
17         self.dropout = nn.Dropout(p=0.6)
18
19     def forward(self, x):
20         x = self.dropout(F.relu(self.fc1(x)))
21         x = self.dropout(F.relu(self.fc2(x)))
22         x = self.dropout(F.relu(self.fc3(x)))
23
24         x = self.dropout(self.output(x))
25         x = F.softmax(x,dim=1)
26         return x
27
28
29 class Surv_Loss(torch.nn.Module):
30     """Survival loss function
31     """
32     def __init__(self,sigma = 1e2, alpha = 1):
33         super(Surv_Loss, self).__init__()
34         self.sigma = sigma
35         self.alpha = alpha
36
37     def forward(self,y_pred,y,status):
38         L1,L2 = 0,0
39         F1 = y_pred.cumsum(1)
40
41         # compute L1
42         #for uncensored: log P(T=t|x)
43         L1 = torch.log(y_pred.gather(1, y.view(-1,1))).reshape(-1).dot(status)
44         #for censored: log \sum P(T>t|x)
45         L1 = torch.log(1 - F1.gather(1, y.view(-1,1))).reshape(-1).dot(1 - status)
46
47         # compute L2
48         for i in range(len(y)):
49             if status[i] ==1:
50                 mask = (y[i] < y)
51                 if sum(mask)>0:
52                     diff = F1[i,y[i]] - F1[mask,y[i]]
53                     L2 += self.alpha torch.exp(-diff/self.sigma).sum()
54
55         loss = L1 + L2
56         return loss

```

```

57
58
59 class ColumnarDataset(Dataset):
60     '''Class to manage tabular dataset in pytorch framework
61     '''
62     def __init__(self, X, y, status):
63         self.dfconts = X
64         self.conts = np.stack([c.values for n, c in self.dfconts.items()], axis=1).astype(np.float32)
65         self.y = y.values
66         self.status = status.values.astype(np.float32)
67
68     def __len__(self): return len(self.y)
69
70     def __getitem__(self, idx):
71         return [self.conts[idx], self.y[idx], self.status[idx]]

```

The following script shows the steps we ensued to train and build predictions for the DeepHit model. This script was originally structured as a jupyter notebook (Kluyver et al., 2016).

```

1  #!/usr/bin/python
2  # coding: utf 8
3
4  ## DeepHit
5  # This notebook implements the DeepHit survival model for Heart Failure patients (with functional covariates).
6
7  import pandas as pd
8  import numpy as np
9  import matplotlib.pyplot as plt
10 from sklearn.model_selection import train_test_split
11 from sklearn.preprocessing import scale
12 import torch
13 import torch.nn as nn
14 import torch.nn.functional as F
15 import torch.optim as optim
16 from torch.utils.data import Dataset, DataLoader
17 from lifelines.utils import concordance_index
18 from tqdm import tqdm
19
20 from utils.deep_hit_utils import
21 from utils.common import
22
23
24 # load data
25 df = pd.read_csv('../data/main_process_preprocessed_data.csv')
26 test = pd.read_csv('../data/main_process_preprocessed_data_test.csv')
27
28 # use GPU if available
29 device = torch.device(('cuda:0' if torch.cuda.is_available() else 'cpu'
30                        ))
31
32 ### Prepare data
33 # set time windows parameters
34 T_max = 365 * 10
35 n_times = int(T_max / 60)
36 # discretise time
37 df['discretised_time_event'] = [int(v) for v in df.time_event / T_max
38                                n_times]
39
40 print ('(Arbitrary) maximum survival time:', np.round(T_max / 365, 1),
41        'years')
42 print ('Number of time steps:', n_times)
43 print ('Size time step:', np.round(T_max / n_times / 30, 1), 'months')
44
45 # create dummy sex
46 df['sexM'] = [(1 if v == 'M' else 0) for v in df.sex]
47 test['sexM'] = [(1 if v == 'M' else 0) for v in test.sex]
48
49 # set features
50 features = [
51     'sexM',
52     'age_in',
53     'ACE_PC1',
54     'ACE_PC2',

```

```

55     'beta_PC1',
56     'beta_PC2',
57     'hospitalisation_PC1',
58     'hospitalisation_PC2',
59     ]
60
61 (X, X_test) = (df[features], test[features])
62 y = df.discretised_time_event
63 true_times = df.time_event
64 status = df.status
65
66 # scale
67 mean = X.mean()
68 std = X.std()
69 X = mean
70 X /= std
71 X_test = mean
72 X_test /= std
73
74 # split train in train/validation (for early stopping)
75 (
76     X_train ,
77     X_valid ,
78     y_train ,
79     y_valid ,
80     status_train ,
81     status_valid ,
82     ) = train_test_split(X, y, status, test_size=0.15, random_state=47)
83
84 # transform validation sets in tensors
85 X_valid = torch.from_numpy(X_valid.values.astype('float32'))
86 y_valid = torch.from_numpy(y_valid.values)
87 status_valid = torch.from_numpy(status_valid.values.astype('float32'))
88 (X_valid, y_valid, status_valid) = (X_valid.to(device),
89                                     y_valid.to(device),
90                                     status_valid.to(device))
91
92 # transform test set in tensors
93 X_test = torch.from_numpy(X_test.values.astype('float32'))
94 X_test = X_test.to(device)
95
96 X_train.head()
97
98 ### Prepare for training
99 # set early stopping parameter
100 max_epochs_no_improvement = 150
101
102 # build a data loader
103 trainds = ColumnarDataset(X_train, y_train, status_train)
104 params = {'batch_size': 25, 'shuffle': True, 'num_workers': 4}
105
106 train_dl = DataLoader(trainds, params)
107
108 ### Tuning Hyperparameters sigma and alpha
109 sigmas = [1e2, 1e3, 1e4]
110 alphas = [1.5, 2, 2.5, 3, 3.5]
111
112 hyp_scores = []
113 # prepare list of hyper parameters tuples
114 hyper_params = []
115 for sigma in sigmas:
116     for alpha in alphas:
117         hyper_params.append((sigma, alpha))
118
119 # loop over hyper parameters tuples
120 for (sigma, alpha) in tqdm(hyper_params):
121     # set loss hyperparams:
122     criterion = Surv_Loss(sigma, alpha)
123     # build the network
124     net = DeepHit(len(features), n_times).to(device)
125     # initialize weights
126     nn.init.xavier_normal_(net.fc1.weight)
127     nn.init.xavier_normal_(net.fc2.weight)
128     nn.init.xavier_normal_(net.fc3.weight)
129     # set optimizer
130     optimizer = optim.Adam(net.parameters(), lr=1e-4)
131     # train

```

```

132 (train_losses, valid_losses) = ([], [])
133 best_net = DeepHit(len(features), n_times)
134 best_loss = 1e10
135 epochs_no_improvement = 0
136 for epoch in range(5000): # loop over the dataset multiple times
137     running_loss = 0.0
138
139     net.train()
140     for (i, data) in enumerate(train_dl, 0):
141         (inputs, labels, status) = data
142         # make it use GPU, if you have it
143         (inputs, labels, status) = (inputs.to(device),
144                                     labels.to(device), status.to(device))
145         # zero the parameter gradients
146         optimizer.zero_grad()
147         # forward + backward + optimize
148         outputs = net(inputs)
149         loss = criterion(outputs, labels, status)
150         loss.backward()
151         optimizer.step()
152         # training loss
153         running_loss += loss.item()
154         # Evaluate on validation set
155
156     net.eval()
157     valid_pred = net(X_valid)
158     valid_loss = criterion(valid_pred, y_valid, status_valid).item()
159     # store loss
160     train_losses.append(running_loss)
161     valid_losses.append(valid_loss)
162     # early stopping
163
164     if valid_loss < best_loss:
165         best_loss = valid_loss
166         # save weights
167         best_net.load_state_dict(net.state_dict())
168         epochs_no_improvement = 0
169     else:
170         epochs_no_improvement += 1
171
172     if epochs_no_improvement > max_epochs_no_improvement:
173         break
174
175     best_net.eval()
176     valid_pred = best_net(X_valid)
177     expected_survival_times_valid = \
178         compute_expected_survival_time(valid_pred, n_times, T_max)
179     C_valid = concordance_index(y_valid, expected_survival_times_valid,
180                               status_valid)
181     hyp_scores.append((sigma, alpha, C_valid))
182
183 best_C_overall = max([v[2] for v in hyp_scores])
184 for v in hyp_scores:
185     if v[2] == best_C_overall:
186         print (
187             'sigma:',
188             v[0],
189             ' alpha:',
190             v[1],
191             ' C index:',
192             v[2],
193             ' < best hyper parameters choice',
194         )
195         best_sigma = v[0]
196         best_alpha = v[1]
197     else:
198         print (
199             'sigma:',
200             v[0],
201             ' alpha:',
202             v[1],
203             ' C index:',
204             v[2],
205         )
206
207 ### Train
208 # set criterion: best hyper params choice

```

```

209 criterion = Surv_Loss(best_sigma, best_alpha)
210 # build the network
211 net = DeepHit(len(features), n_times).to(device)
212 # initialize weights
213 nn.init.xavier_normal_(net.fc1.weight)
214 nn.init.xavier_normal_(net.fc2.weight)
215 nn.init.xavier_normal_(net.fc3.weight)
216 # set optimizer
217 optimizer = optim.Adam(net.parameters(), lr=1e-4)
218
219 (train_losses, valid_losses) = ([], [])
220 best_net = DeepHit(len(features), n_times)
221 best_loss = 1e10
222 epochs_no_improvement = 0
223
224 # train
225 for epoch in range(5000): # loop over the dataset multiple times
226     running_loss = 0.0
227     net.train()
228     for (i, data) in enumerate(train_dl, 0):
229         (inputs, labels, status) = data
230         # make it use GPU, if you have it
231         (inputs, labels, status) = (inputs.to(device),
232                                   labels.to(device),
233                                   status.to(device))
234
235         # zero the parameter gradients
236         optimizer.zero_grad()
237         # forward + backward + optimize
238         outputs = net(inputs)
239         loss = criterion(outputs, labels, status)
240         loss.backward()
241         optimizer.step()
242         # training loss
243         running_loss += loss.item()
244     # Evaluate on validation set
245     net.eval()
246     valid_pred = net(X_valid)
247     valid_loss = criterion(valid_pred, y_valid, status_valid).item()
248     # store loss
249     train_losses.append(running_loss)
250     valid_losses.append(valid_loss)
251     # early stopping
252     if valid_loss < best_loss:
253         best_loss = valid_loss
254         # save weights
255         best_net.load_state_dict(net.state_dict())
256         epochs_no_improvement = 0
257     else:
258         epochs_no_improvement += 1
259     if epochs_no_improvement > max_epochs_no_improvement:
260         break
261     if epoch % 100 == 0:
262         print (
263             'Epoch: ',
264             epoch,
265             ' Training Loss: ',
266             np.round(running_loss, 2),
267             ' Current Validation Loss: ',
268             np.round(valid_loss, 2),
269             ' Best Validation Loss: ',
270             np.round(best_loss, 2),
271             )
272 print 'Finished Training'
273
274 plt.figure(figsize=(10, 5))
275
276 plt.subplot(1, 2, 1)
277 plt.plot(train_losses, color='orange', label='training')
278 plt.legend()
279 plt.title('Survival loss')
280 plt.xlabel('epochs')
281
282 plt.subplot(1, 2, 2)
283 plt.plot(valid_losses, color='blue', label='validation')
284 plt.legend()
285 plt.title('Survival loss')

```

```

286 plt.xlabel('epochs')
287
288 plt.show()
289
290 ## Evaluate Concordance Index on test set
291 # Note: we convert predictions to original time scale and we compare with true
292 #       time of events; this is done in order to make it comparable with other models
293 # save model
294 torch.save(best_net.state_dict(), '../data/deepHit_weights.pt')
295 # load model
296 best_net = DeepHit(len(features), n_times).to(device)
297 best_net.load_state_dict(torch.load('../data/DeepHit_weights.pt'))
298 best_net.eval()
299 prediction_test_set = best_net(X_test)
300 expected_survival_times = \
301     compute_expected_survival_time(prediction_test_set, n_times, T_max)
302
303 C = concordance_index(test.time_event, expected_survival_times,
304                      test.status)
305 print('Concordance Index on test set:', np.round(C * 100, 2), '%')
306
307 # save predictions
308 pd.DataFrame(expected_survival_times).to_csv('../data/predictions_deepHit.csv'
309      , index=False)
310 pd.DataFrame(prediction_test_set.detach().numpy()).to_csv('../data/full_predictions_deephit.csv'
311      , index=False)
312
313 ## plot a sample of predictions
314 predictions_deephit = pd.DataFrame(prediction_test_set.detach().numpy())
315 predictions_deephit.index = test.id
316 sample_ids = np.random.choice(test.id, size=500, replace=False)
317 year_max_deephit = 10
318 deephit_time = np.arange(1, len(predictions_deephit.columns) + 1) \
319     / len(predictions_deephit.columns) * year_max_deephit
320 plt.figure(figsize=(10, 7))
321 for i in sample_ids:
322     plt.plot(deephit_time, predictions_deephit.loc[i, :])
323 plt.xlabel('time [years]')
324 plt.title('DeepHit, Density survival time test set', size=14)
325 plt.show()

```

A.5 DRSA

The following classes are used for the DRSA model.

```

1 import torch
2 import torch.nn as nn
3 import torch.nn.functional as F
4 from torch.utils.data import Dataset
5 import numpy as np
6
7
8 class SequenceDataset(Dataset):
9     '''Class to manage 3D dataset in pytorch framework for RNN (dimensions: time,id,features)
10     ...
11     def __init__(self, obs, y, status):
12         self.conts = obs.astype(np.float32)
13         self.y = y.values
14         self.status = status.values.astype(np.float32)
15
16     def __len__(self): return len(self.y)
17
18     def __getitem__(self, idx):
19         return [self.conts[:, idx, :], self.y[idx], self.status[idx]]
20
21
22 class DRSA(nn.Module):
23     '''DRSA network architecture
24     ...
25     def __init__(self, n_features, dropout = 0.6):
26         super(DRSA, self).__init__()

```

```

27     self.n_features = n_features
28     self.hidden_dim = 10 * n_features
29     self.lstm = nn.LSTM(n_features, self.hidden_dim)
30     self.output = nn.Linear(self.hidden_dim, 1)
31     self.dropout = nn.Dropout(p=dropout)
32
33     # custom learnable initial values for h0 and c0
34     self.initialise_h0 = nn.Linear(self.n_features, 1, self.hidden_dim)
35     self.initialise_c0 = nn.Linear(self.n_features, 1, self.hidden_dim)
36
37     def forward(self, x):
38         # compute initial values hidden and cell vectors
39         h0 = torch.relu(self.initialise_h0(x[0, :, : self.n_features, 1])).reshape(1, 1, self.hidden_dim)
40         c0 = torch.relu(self.initialise_c0(x[0, :, : self.n_features, 1])).reshape(1, 1, self.hidden_dim)
41         h0 = self.dropout(h0)
42         c0 = self.dropout(c0)
43         # run model recurrently for the whole sequences
44         x = self.dropout(self.lstm(x, (h0, c0))[0])
45         x = self.output(x)
46         x = torch.sigmoid(x)
47         return x
48
49
50 class DRSA_Loss(torch.nn.Module):
51     """DRSA loss function
52     """
53     def __init__(self, alpha = 0.25):
54         super(DRSA_Loss, self).__init__()
55         self.alpha = alpha
56
57     def forward(self, y_pred, y, status):
58         mask = (status == 1)
59
60         # Compute Lz
61         # compute logs and sum of (1 - log)
62         logs = torch.log(y_pred[mask].gather(1, y[mask].view(-1, 1)))
63         # little trick for events happening at time 0, to avoid looking at index -1
64         tmp = y[mask & (y != 0)] - 1
65         sum_one_minus_log = torch.log(1 - y_pred[mask & (y != 0)]).cumsum(1).gather(1, tmp.view(-1, 1))
66         Lz = (logs.sum() + sum_one_minus_log.sum())
67
68         # Compute L_uncensored
69         L_uncensored = torch.log(1 - (1 - y_pred[mask]).cumprod(1).gather(1, y[mask].view(-1, 1))).sum()
70
71         # Compute L_censored
72         L_censored = torch.log(1 - y_pred[1 - mask]).cumsum(1).gather(1, y[1 - mask].view(-1, 1)).sum()
73
74         Lc = L_uncensored + L_censored
75
76         loss = self.alpha * Lz + (1 - self.alpha) * Lc
77
78         return loss
79
80
81
82 def reformat_dataset(x_in, times):
83     """ Adds time feature, returns array with 3 dimensions: time, id, features
84     """
85     x_out = []
86     for i in x_in.index:
87         tmp = np.repeat(x_in.loc[i, :].values.reshape(1, 1, -1), len(times), axis = 0)
88         observation_i = np.concatenate([tmp, times], axis = 2)
89         x_out.append(observation_i)
90     x_out = np.concatenate(x_out, axis = 1)
91     return x_out
92
93
94 def get_survival_density(predictions):
95     """ compute survival time probabilities from hazards """
96     survival_density = [predictions[:, 0].reshape(-1, 1)]
97     tmp = (1 - predictions).cumprod(1)
98     for j in np.arange(1, predictions.shape[1]):
99         if j > 0:
100             survival_density.append((predictions[:, j] * tmp[:, :j - 1]).reshape(-1, 1))
101     survival_density = torch.cat(survival_density, dim = 1)
102     return survival_density

```

The following script shows the steps ensued to train and build predictions for the DRSA model. This script was originally structured as a jupyter notebook (Kluyver et al., 2016).

```

1  #!/usr/bin/python
2  #   coding: utf 8
3
4  ## DRSA
5  # This notebook implements the DRSA survival model for Heart Failure patients (with functional covariates).
6
7  import pandas as pd
8  import numpy as np
9  import matplotlib.pyplot as plt
10 from sklearn.model_selection import train_test_split
11 from sklearn.preprocessing import scale
12 import torch
13 import torch.nn as nn
14 import torch.nn.functional as F
15 import torch.optim as optim
16 from torch.utils.data import Dataset, DataLoader
17 from lifelines.utils import concordance_index
18 from tqdm import tqdm
19
20 from utils.drsa_utils import
21 from utils.common import
22
23
24 # load data
25 df = pd.read_csv('../data/main_process_preprocessed_data.csv')
26 test = pd.read_csv('../data/main_process_preprocessed_data_test.csv')
27
28 # use GPU if available
29 device = torch.device(('cuda:0' if torch.cuda.is_available() else 'cpu'
30                        ))
31
32 ### Prepare data
33 # set time windows parameters
34 T_max = 365 * 30
35 n_times = int(T_max / 30)
36
37 # discretise time
38 df['discretised_time_event'] = [int(v) for v in df.time_event / T_max
39                                n_times]
40 times = np.arange(n_times).reshape(1, 1, 1)
41
42 print ('(Arbitrary) maximum survival time:', np.round(T_max / 365, 1),
43        'years')
44 print ('Number of time steps:', n_times)
45 print ('Size time step:', np.round(T_max / n_times / 30, 1), 'months')
46
47 # create dummy sex
48 df['sexM'] = [(1 if v == 'M' else 0) for v in df.sex]
49 test['sexM'] = [(1 if v == 'M' else 0) for v in test.sex]
50
51 # set features
52 features = [
53     'sexM',
54     'age_in',
55     'ACE_PC1',
56     'ACE_PC2',
57     'beta_PC1',
58     'beta_PC2',
59     'hospitalisation_PC1',
60     'hospitalisation_PC2',
61 ]
62
63 (X, X_test) = (df[features], test[features])
64 y = df.discretised_time_event
65 true_times = df.time_event
66 status = df.status
67
68 # scale
69 mean = X.mean()
70 std = X.std()
71 X = mean

```

```

72 X /= std
73 X_test = mean
74 X_test /= std
75
76 # split train in train/validation (for early stopping)
77 (
78     X_train ,
79     X_valid ,
80     y_train ,
81     y_valid ,
82     status_train ,
83     status_valid ,
84     ) = train_test_split(X, y, status, test_size=0.05, random_state=47)
85
86 (X_train.shape, X_valid.shape, X_test.shape)
87
88 ### Reformat datasets for lstm layers (3 dimensions: time,id,features)
89 x_training = reformat_dataset(X_train, times)
90 x_validation = reformat_dataset(X_valid, times)
91 x_test = reformat_dataset(X_test, times)
92 # dimensions: time,id,features (added time)
93 (x_training.shape, x_validation.shape, x_test.shape)
94
95 # transform validation sets in torch tensors
96 X_valid = torch.from_numpy(x_validation.astype('float32'))
97 y_valid = torch.from_numpy(y_valid.values)
98 status_valid = torch.from_numpy(status_valid.values.astype('float32'))
99 (X_valid, y_valid, status_valid) = (X_valid.to(device),
100                                     y_valid.to(device),
101                                     status_valid.to(device))
102
103 # transform test set in tensors
104 X_test = torch.from_numpy(x_test.astype('float32'))
105 X_test = X_test.to(device)
106
107 ### Prepare for training
108 # set early stopping parameter
109 max_epochs_no_improvement = 20
110 # build a training data loader
111 trainds = SequenceDataset(x_training, y_train, status_train)
112 params = {'batch_size': 25, 'shuffle': True, 'num_workers': 4}
113 train_dl = DataLoader(trainds, params)
114
115 ### Train
116 # set criterion
117 criterion = DRSA_Loss(0.75)
118 # build the network
119 net = DRSA(len(features) + 1).to(device)
120 # set optimizer
121 optimizer = optim.Adam(net.parameters(), lr=5e-5)
122
123 (train_losses, valid_losses) = ([], [])
124 best_net = DRSA(len(features) + 1).to(device)
125 best_loss = 1e10
126 epochs_no_improvement = 0
127
128 # train
129 for epoch in range(500): # loop over the dataset multiple times
130     running_loss = 0.0
131
132     net.train()
133     for (i, data) in enumerate(train_dl, 0):
134         (inputs, labels, status) = data
135         # dimensions: time, id, features
136         inputs = inputs.transpose(0, 1)
137         # make it use GPU, if you have it
138         (inputs, labels, status) = (inputs.to(device),
139                                     labels.to(device),
140                                     status.to(device))
141
142         # zero the parameter gradients
143         optimizer.zero_grad()
144         # forward + backward + optimize (rows: id; columns: time)
145         outputs = net(inputs).reshape(n_times, 1).transpose(0, 1)
146         loss = criterion(outputs, labels, status)
147         loss.backward()
148         optimizer.step()
149         # training loss

```

```

149     running_loss += loss.item()
150
151     # Evaluate on validation set
152     net.eval()
153     valid_pred = net(X_valid).reshape(n_times, 1).transpose(0, 1)
154     valid_loss = criterion(valid_pred, y_valid, status_valid).item()
155
156     # store losses
157     train_losses.append(running_loss)
158     valid_losses.append(valid_loss)
159
160     # save best weights
161     if valid_loss < best_loss:
162         best_loss = valid_loss
163         best_net.load_state_dict(net.state_dict())
164         epochs_no_improvement = 0
165     else:
166         epochs_no_improvement += 1
167
168     # early stopping
169     if epochs_no_improvement > max_epochs_no_improvement:
170         break
171     if epoch % 25 == 0:
172         print (
173             'Epoch: ',
174             epoch,
175             ' Training Loss: ',
176             np.round(running_loss, 2),
177             ' Current Validation Loss: ',
178             np.round(valid_loss, 2),
179             ' Best Validation Loss: ',
180             np.round(best_loss, 2),
181             )
182
183     print 'Finished Training'
184
185     plt.figure(figsize=(10, 5))
186
187     plt.subplot(1, 2, 1)
188     plt.plot(train_losses, color='orange', label='training')
189     plt.legend()
190     plt.title('Survival loss')
191     plt.xlabel('epochs')
192
193     plt.subplot(1, 2, 2)
194     plt.plot(valid_losses, color='blue', label='validation')
195     plt.legend()
196     plt.title('Survival loss')
197     plt.xlabel('epochs')
198
199     plt.show()
200
201     ## Evaluate Concordance Index on test set
202     # Note: we convert predictions to original time scale and we compare them with true
203     # time of events; this is done in order to make it comparable with other models
204
205     best_net.eval()
206     # predict hazards
207     predictions_test_set = best_net(X_test).reshape(n_times,
208             1).transpose(0, 1)
209
210     # compute survival time probabilities from hazards
211     survival_density = get_survival_density(predictions_test_set)
212
213     # compute expected survival times in the original scale
214     expected_survival_times = \
215         compute_expected_survival_time(survival_density, n_times, T_max)
216
217     # evaluate C index
218     C = concordance_index(test.time_event, expected_survival_times,
219             test.status)
220     print ('Concordance Index on test set:', np.round(C * 100, 2), '%')
221
222     # save model
223     torch.save(best_net.state_dict(), '../data/DRSA_weights.pt')
224
225     # save predictions

```

```
226 pd.DataFrame(expected_survival_times).to_csv('../data/predictions_DRSA.csv'
227         , index=False)
228
229 # check that probability of surviving more than predicted period is negligible:
230 survival_density.sum(1).min()
231
232 plt.hist(expected_survival_times / 365)
233 plt.title('Expected survival times')
234 plt.show()
235
236 ### Plot predicted Hazards
237 best_net = DRSA(len(features) + 1).to(device)
238 best_net.load_state_dict(torch.load('../data/DRSA_weights.pt'))
239 best_net.eval()
240 predictions_test_set = best_net(X_test).reshape(n_times,
241         1).transpose(0, 1)
242 tmp = predictions_test_set.detach().numpy()
243 entire_predictions = pd.DataFrame(tmp)
244 entire_predictions.index = test.id
245 entire_predictions.to_csv('../data/full_predictions_drsa.csv',
246         index=True)
247 sample = np.random.choice(range(len(tmp)), size=500, replace=False)
248 original_times = times.reshape( 1, T_max / n_times)
249 plt.figure(figsize=(10, 7))
250 for i in sample:
251     plt.plot(original_times / 365, tmp[i, :])
252     plt.xlabel('time [years]')
253     plt.title('Hazards test set', size=14)
254     plt.show()
255
256 plt.figure(figsize=(10, 5))
257 plt.plot(original_times / 365, tmp.mean(axis=0))
258 plt.xlabel('time [years]')
259 plt.title('Hazard average prediction')
260 plt.show()
```


Bibliography

- Amorim, L. D. A. F. and J. Cai (2014). "Modelling recurrent events: a tutorial for analysis in epidemiology". In: *International Journal of Epidemiology* 44.1, pp. 324–333.
- Andersen, P. K., O. Borgan, et al. (2012). *Statistical Models Based on Counting Processes*. Springer Series in Statistics. Springer New York.
- Andersen, P. K. and R. D. Gill (1982). "Cox's Regression Model for Counting Processes: A Large Sample Study". In: *The Annals of Statistics* 10.4, pp. 1100–1120.
- Andersen, P. K. and N. Keiding (2002). "Multi-state models for event history analysis". In: *Statistical Methods in Medical Research* 16.2, pp. 91–115.
- Bahdanau, D., K. Cho, and Y. Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: arXiv preprint arXiv:1409.0473
- Baraldo, S. et al. (2013). "Outcome Prediction for Heart Failure Telemonitoring Via Generalized Linear Models with Functional Covariates". In: *Scandinavian Journal of Statistics* 40.3, pp. 403–416.
- Berry, C., D. R. Murdoch, and J. J. V. McMurray (2001). "Economics of chronic heart failure". In: *European Journal of Heart Failure* 3, pp. 283–291.
- Brandon, J. G., S. Seals, and I. Aban (2014). "Survival analysis and regression models". In: *Journal of Nuclear Cardiology* 1.4, pp. 686–694.
- Breslow, N. E. (1975). "Analysis of survival data under the proportional hazards model". In: *International Statistical Review/Revue Internationale de Statistique*, pp. 45–57.
- Brownlee, J. (2017). How to Handle Very Long Sequences with Long Short-Term Memory Recurrent Neural Networks [Online; accessed 16-Jun-2019] URL: <https://machinelearningmastery.com/handle-long-sequences-long-short-term-memory-recurrent-neural-networks/> .
- Cox, D. et al. (1972). "Regression models and life tables (with discussion)". In: *Journal of the Royal Statistical Society* 34.2, pp. 187–220.
- Elixhauser, A., C. Steiner, and L. Palmer (2008). "Clinical classifications software (CCS)". In: *Book Clinical Classifications Software (CCS)* (Editor ed eds)

- Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning Adaptive Computation and Machine Learning series*. MIT Press.
- Harte, D. et al. (2010). "PtProcess: An R package for modelling marked point processes indexed by time". In: *Journal of Statistical Software* 35.8, pp. 1–32.
- He, X. and P. Ng (1999). "COBS: Qualitatively constrained smoothing via linear programming". In: *Computational Statistics* 14.3, pp. 315–338.
- Hecht-Nielsen, R. (1992). "Theory of the backpropagation neural network". In: *Neural networks for perception* Elsevier, pp. 65–93.
- Ho, K. K. L. et al. (1993). "The epidemiology of heart failure: The Framingham Study". In: *Journal of the American College of Cardiology* 22.4 Supplement 1, A6–A13.
- Kang, L. and W. Chen (2015). "Package `compareC`". In: *Survival analysis Published on CRAN*
- Kang, L., W. Chen, et al. (2015). "Comparing two correlated C indices with right-censored survival outcome: a one-shot nonparametric approach". In: *Statistics in medicine* 34.4, pp. 685–703.
- Katzman, J. L. et al. (2018). "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network". In: *BMC medical research methodology* 18.1, p. 24.
- Kelly, P. J. and L. L-Y. Lim (2000). "Survival analysis for recurrent event data: an application to childhood infectious diseases". In: *Statistics in Medicine* 19.1, pp. 13–33.
- Kingma, D. P. and J. Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*
- Kleinbaum, D. G. and M. Klein (2013). *Survival Analysis: A Self-Learning Text Statistics for Biology and Health*. Springer New York.
- Kluyver, T. et al. (2016). "Jupyter Notebooks—a publishing format for reproducible computational workflows." In: *ELPUB*, pp. 87–90.
- Koturwar, S. and S. Merchant (2017). "Weight Initialization of Deep Neural Networks(DNNs) using Data Statistics". In: *CoRRabs/1710.10570*.
- LeCun, Y. A. et al. (2012). "Efficient backprop". In: *Neural networks: Tricks of the trade* Springer, pp. 9–48.
- Lee, C. et al. (2018). "DeepHit: A Deep Learning Approach to Survival Analysis With Competing Risks". In: *AAAI*.
- Lee, E. T. and J. Wang (2003) *Statistical Methods for Survival Data Analysis* Wiley Series in Probability and Statistics. Wiley.

- Lin, D. Y. (2007). "On the Breslow estimator". In: *Lifetime data analysis* 3.4, pp. 471–480.
- Maggioni, A. P. and F. Spandonaro (2014). "Lo scompenso cardiaco acuto in Italia". In: *Giornale italiano di cardiologia* 5.2 (Suppl. 2), 3S–4S.
- Mancini, L. and A. M. Paganoni (2019). "Marked point process models for the admissions of heart failure patients". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 2.2, pp. 125–135.
- McMurray, J. et al. (2005). "Practical recommendations for the use of ACE inhibitors, beta-blockers, aldosterone antagonists and angiotensin receptor blockers in heart failure: Putting guidelines into practice". In: *European Journal of Heart Failure* 7.5, pp. 710–721.
- Mihajlovic, I. (2018). *Introduction To Artificial Intelligence — Neural Networks* [Online; accessed 7-Jun-2019] URL: <https://mc.ai/introduction-to-artificial-intelligence%E2%80%8A-%E2%80%8Aneural-networks/>.
- Moolgavkar, S. H. et al. (2018). "An Assessment of the Cox Proportional Hazards Regression Model for Epidemiologic Studies". In: *Risk Analysis* 38.4, pp. 777–794.
- Mosterd, A. and A. W. Hoes (2007). "Clinical epidemiology of heart failure". In: *Heart* 93.9, pp. 1137–1146.
- Paszke, A. et al. (2017). "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration". In: *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*.
- Peña, E. A. and M. Hollander (2004). "Models for Recurrent Events in Reliability and Survival Analysis". In: *Mathematical Reliability: An Expository Perspective* Ed. by Re k Soyer, Thomas A. Mazzuchi, and Nozer D. Singpurwalla. Boston, MA: Springer US, pp. 105–123.
- Pencina, M. J. and R. B. D'Agostino (2004). "Overall C as a measure of discrimination in survival analysis: model specific population value and confidence interval estimation". In: *Statistics in medicine* 23.13, pp. 2109–2123.
- Prentice, R. L., B. J. Williams, and A. V. Peterson (1981). "On the regression analysis of multivariate failure time data". In: *Biometrika* 68.2, pp. 373–379.
- Ramachandran, P., B. Zoph, and Q. V. Le (2017). "Searching for activation functions". In: *arXiv preprint arXiv:1710.05941*.
- Ramsay, J. and B. W. Silverman (2013) *Functional Data Analysis* Springer Series in Statistics. Springer New York.
- Regione Lombardia (2012). "HFData project: Utilization of Regional Health Source Databases for Evaluating Epidemiology, short- and medium-term

- outcome and process indicators in patients hospitalized for heart failure". In: *Progetto di Ricerca Finalizzata di Regione Lombardia - HFData-RF-2009-1483329*.
- Ren, K. et al. (2018). "Deep Recurrent Survival Analysis". In: *CoRR abs/1809.02403*.
- Rossi, F. et al. (2007). "Representation of Functional Data in Neural Networks". In: *CoRR abs/0709.3641*.
- UC-R-Programming (2018). *File:deep_nn.png*. [Online; accessed 8-May-2019]. URL: http://uc-r.github.io/public/images/analitics/deep_learning/deep_nn.png.
- S. Kennedy, B. (2001). "Repeated Hospitalizations and Self-rated Health among the Elderly: A Multivariate Failure Time Analysis". In: *American Journal of Epidemiology* 153, pp. 232–241.
- Spreafico, M. (2017). "Statistical modelling of adherence to drug prescription and its effects on survival in Heart Failure patients". MA thesis. Italy: Politecnico di Milano.
- Srivastava, N. et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.
- Therneau, T. M., P. M. Grambsch, and T. R. Fleming (1990). "Martingale-based residuals for survival models". In: *Biometrika* 77.1, pp. 147–160.
- Therneau, T. M. and T. Lumley (2014). "Package 'survival'". In: *Survival analysis Published on CRAN*.
- Wikimedia-Commons (2019). *File:The LSTM cell.png* — *Wikimedia Commons, the free media repository*. [Online; accessed 8-May-2019]. URL: https://commons.wikimedia.org/w/index.php?title=File:The_LSTM_cell.png&oldid=344169785.
- Yao, Y., L. Rosasco, and A. Caponnetto (2007). "On early stopping in gradient descent learning". In: *Constructive Approximation* 26.2, pp. 289–315.