ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Università di Bologna

Dipartimento di Informatica - Scienza e Ingegneria (DISI)
Second Cycle Degree/Two Year Master in Software Engineering

Thesis For The Master Degree in Software Engineering

Neural Cox Model for Liver Transplant

Thesis Advisor
Prof. Michele Lombardi

Candidate
Ugo Marchesini
0937-0000977872

Academic Year MMXXIII-MMXXIV

Essentially, all models are wrong, but some are useful.

George E. P. Box

Abstract

Liver transplantation is a crucial for patients with end-stage liver disease and predicting post-transplant survival is a complex challenge. Deciding which patient receives a transplant or not is an important and hard medical decision that takes into account many factors. The aim of this thesis is to investigate a model capable of forecasting survival trends in patients subjected to liver transplant. In particular, we exploit a hybrid neural Cox proportional hazards model. To the best of our knowledge, this approach is novel and provides promising results. Indeed, this model is designed in collaboration with Sant'Orsola hospital of Bologna to be integrated within the decision process of surgeons. Our approach, which provides a data driven estimation of the the survivability function associated to patients who do not receive a transplantation, is compared with the more classical Cox survival model. More specifically, we compare our approach to two widely used libraries, namely, `scikit-survival` and `lifelines`. The results obtained on a synthetic dataset prove that the neural cox model effectively compares the classical model. We believe that the integration of deep learning with classical statical approaches can surpass the limitation of both approaches: on the one hand, classical cox model can be simplistic in describing complex relationships that neural networks can instead capture; while on the other hand, the use of pre existing survival framework allows to obtain a more transparent process within the realm of deep learning.

Keywords: survival analysis, neural networks, cox proportional hazard model, python.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Application Domain

Liver transplantation is a cornerstone treatment for end-stage liver disease and acute liver failure, providing a vital lifeline for patients facing otherwise insurmountable health challenges , *"whose condition can't be controlled with other treatments"*.[26]. This research aims to delve into the complexities of survival analysis in general but oriented to the context of liver transplantation, seeking to discover patterns, showing metrics for comparison with existing tools.

Survival analysis is a statistical approach to study the time to an event of interest, and is particularly suited to this investigation. In liver transplantation, the "event" typically refers to patient mortality, with the primary goal of understanding the duration of survival and identifying determinants that significantly influence this period. This analysis includes a range of methodologies, from the Kaplan-Meier estimator, which provides an estimate of the survival functions, or the Cox proportional hazard model, which can take into account the simultaneous influence of multiple variables. The Cox proportional hazard model is the one chosen for this research.[4]

The aim of this study is twofold, on the one hand to find a solution in the context of neural networks, since the use of a neural model together with the Cox model is scarcely present in the scientific literature, on the other hand to compare the results with software tools already used in the scientific community such as scikit-survival and lifelines.

In the following sections, we will provide a detailed overview of the methodology employed, present and discuss the results.

Chapter 2

Background

During the development of this thesis it was necessary to address several topics that revolve around software engineering. Some are necessary for the theoretical aspect such as the Cox Proportional Hazard Model and survival analysis, others from an applicative point of view such as neural networks or the gray-box model used for parameter identification, in this section they will be described with a historical note regarding neural networks.

2.1 Cox Proportional Hazard Model

The Cox Proportional Hazards model, often referred to as the Cox model, is a statistical technique used for survival analysis. Developed by Cox (1972), this model addresses the relationship between covariates, for example a blood value, its coefficients called β and baseline hazard with the hazard ratio and then survivability and survival time of subjects. Model take into account the risk of an event occurring at a particular time point, given that the subject has survived up to that time.
Key Components of the Cox Proportional Hazards Model are :

1. **Hazard Function**: The hazard function, $\lambda(t)$, represents the instantaneous rate at which events occur, given no prior event. The Cox model states that hazards can be separated into a baseline hazard function, $\lambda_0(t)$, which is common to all subjects and is dependent on the covariates X_i and coefficients variables β_i . Hazard is a non negative value.

2. **Predictor or Covariates Variables**: These are the variables that produce the hazard. They can be continuous (e.g., age, vital measure) or categorical (e.g., sex and treatment type) or binary. In the Cox model, the effects of these predictors were expressed exponentially to ensure that the hazard remained positive.

3. **Proportionality Assumption**: A key assumption of the Cox model is that the hazard ratios between individuals are constant over time. This implies that the effect of a predictors variables on the hazard is multiplicative, meaning it's proportional to the exponent of a sum of the product of coefficients and covariates, and does not change over time.

Model Specification

The Cox model has the form :

$$\lambda(t|X) = \lambda_0(t) \exp(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p)$$

where $\lambda(t|X)$ is the hazard function for a subject with a set of covariates variables $X = (X_1, X_2, ..., X_n)$. The term $\lambda_0(t)$ is the baseline hazard function and $\beta_1, \beta_2, ..., \beta_n$ are the coefficients of each covariates. [7]
Exponential part $\exp(\beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p)$ is called baseline scaling factor , in this thesis is also define as score risk.

Estimation and Interpretation

Estimating the coefficients β is typically performed using partial likelihood methods, which do not require the knowledge of $\lambda_0(t)$. This feature allows the analysis of survival data without the need to assume a particular parametric form for the baseline hazard function. [21]

The exponentiated coefficients, $\exp(\beta_i)$, are interpreted as hazard ratios. For example, if $\exp(\beta_1) = 2$, this indicates that the hazard for a subject with a one-unit increase in X_1 is twice that of a subject without such an increase, assuming all other variables are held constant. [5]

Applications and Limitations

The Cox Proportional Hazards model is widely used in various fields, including medicine, biology, and engineering, to study the time until an event occurs (e.g., death, relapse, and machine failure). Its flexibility in handling censored data, where the event of interest has not occurred for some subjects by the end of the study, adds to its robustness, some estimator manage censoring introducing boundaries to initial assumption of Cox model.

However, this model has several limitations. The proportionality assumption may not hold in all situations, and violations of this assumption can lead to biased estimates. Techniques, such as time-dependent covariates or stratification, can be used to address these issues.[29]

Conclusion

The Cox Proportional Hazards model is a powerful tool for survival analysis, offering insights into the relationships between covariates and all the features related to survivability like hazard and survival time. Its ability to handle various types of data and flexibility in not requiring a specific baseline hazard function make it a preferred choice for researchers. However, careful consideration of the assumptions and limitations is crucial for obtaining accurate and meaningful results.

2.2 Artificial Neural Network

In this paragraph, the characteristics of AI are briefly highlighted.

Introduction

Neural networks, inspired by the structure and function of the human brain, are the cornerstone of modern artificial intelligence (AI). These computational models has revolutionized fields from computer vision and natural language to robotics and gaming. Their ability to learn from data and make intelligent decisions has made them a necessity for advanced technological solutions. It could be deemed as an approximation function because they are trained to *imitate* the relationship between input and output

Neural Network Architecture

According to *Artificial Intelligence : a modern approach* : *"Basically, a neural network is composed of layers of nodes or neurons that process data and extract patterns. Neural networks are composed of nodes or units connected by directed links. A link from unit j to unit i serves to propagate the activation a_j from j to i. Each link also has a numeric weight W_ij associated with it, which determines the strength and sign of the connection."*[33]

The basic architecture includes:

1. **Input Layer**: The input layer receives the raw data. Each neuron in this layer represents a feature or attribute of the input data.

2. **Hidden Layers**: These intermediate layers perform feature extraction and complex operation , mathematical ones and more. Each neuron in the hidden layer receives input from the previous layer, processes it, and passes it on to the next layer. The height (number of neurons per layer) and width (number of hidden layers) can be varied, affecting the learning ability of the network. The ability to manage non linear operator lead to potentially approximate every possible function , some layer has an *activation* function that "fires" when a linear combination of its inputs exceeds some threshold.[33]

3. **Output layer**: The output layer produces the final result, which could be the classification, regression value, or any other expected output. In case of classification output it usually means a probability to behave to that class.

Connections between neurons are associated with weights that are adjusted through training to minimize errors in the network's predictions. This process is governed by tuning *hyperparameter* like learning rate, optimizer or loss function.

Types of Neural Networks

There are several types of neural networks, each designed for specific tasks;

1. **Feedforward neural networks** (FNNs): The simplest form where data is moved in one direction from input to output. There are no cycles or loops in the network. They are mainly used for pattern recognition and classification tasks.[10]

2. **Convolutional Neural Networks** (CNN): Specialized in processing grid-like data such as images. They use convolutional layers to detect local hierarchies and patterns and make them suitable for image and video recognition.[27][34]

3. **Recurrent Neural Networks** (RNN): Sequential prediction, such as time series, natural language or daily flood. RNNs have connection formation cycles so they can retain memory of previous inputs. *"While feedforward networks have different weights across each node, recurrent neural networks share the same weight parameter within each layer of the network"*.[17]

4. **Long Short-Term Memory** (LSTM) Networks: A type of RNN-like gradient vanishing problem that enables long-term learning capabilities. They are widely used in speech recognition.[14]

5. **Generative Adversarial Networks** (GAN): Consisting of two networks, *"a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G"*, essentially one compete with each other. GANs are powerful for generating realistic synthetic data, such as images and audio.[12]

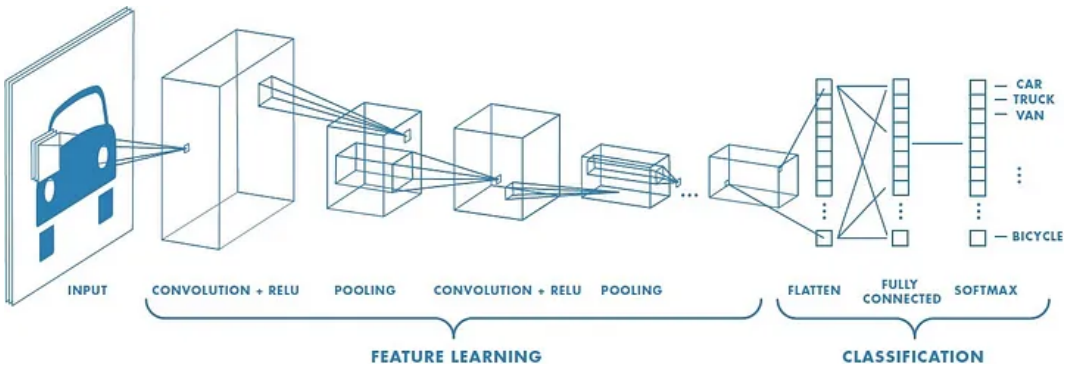

Figure 2.1: Architecture of Cnn [27]

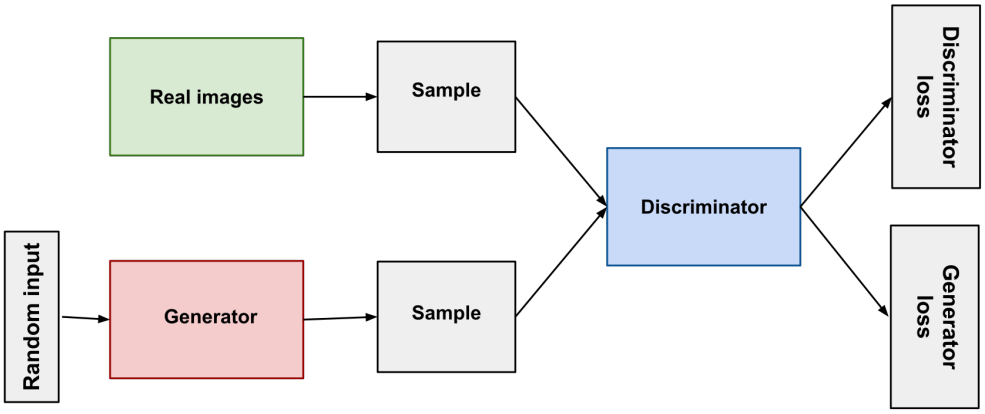

Figure 2.2: Architecture of Gan [13]

Applications of Neural Networks

Neural networks have a wide range of applications in various industries:

1. **Health**: They are used for diagnostic purposes, such as cancer detection in medical imaging, epidemic prediction, and personalization of treatment plans.
2. **Money**: Neural networks help in credit scoring, fraud detection, algorithmic trading, and risk management.
3. **Automotive**: They are crucial in the development of autonomous vehicles, enabling functions such as object detection, lane keeping, and road planning.
4. **Experience**: From recommendation systems in streaming services to real-world language translation and content generation, neural networks improve user experiences.
5. **Manufacturing**: They optimize supply chain management, predictive maintenance, product failure detection,and process quality control.
6. **Software Development** : They suggest api or whole function to software developer completing source code with code completion features.

Challenges and Future Directions

Despite their successes, neural networks face several challenges:

1. **Data requirements**: Training deep neural networks could requires large amounts of labeled data, which can be difficult and expensive. In some specific application like reinforced learning do not need to be trained with label or *true value*, so they are not supervised but some task could be computational challenging as well.
2. **Computational resources**: The training process is computationally intensive and often requires specialized hardware such as GPUs or TPUs. Some SAAS website like such *Google Colaboratory* that offers remote computation capabilities but could still not face some basic problem due to the limits imposed by service provider.

3. **Interpretability\Explainability**: Neural networks are often seen as "black boxes" due to their complex and opaque decision-making. Greater interpretability depends on trust and transparency. Explainability resides on how the model explain results , output and parameters , in human terms.[3]

4. **Overfitting**: *Over* trained neural network could not generalize the desired function , when a neural network is overtrained is like it remembers training dataset underperforming on new unseen data.

Some software framework like Tensorflow\Keras , PyTorch or ML from Microsoft offers powerful api to build complex neural network application.

2.3 Brief History of Neural Network

Historical hints [28]:

1. 1943 McCulloch and Pitts: defined the first binary threshold neuron model.
2. 1949 Hebb: from studies on the brain, he showed that learning is not a neuron property, but it is due to a modification of synapses.
3. 1962 Rosenblatt: he proposed a neuron model that can be learned by examples: the perceptron.
4. 1969 Minsky and Papert showed the limitations of the perceptron: diminished enthusiasm on neural networks.
5. 1982 Hopfield proposed a network model to create associative memories.
6. 1985 Rumelhart, Hinton and Williams: formalize supervised learning (Backpropagation)
7. 2006 Yoshua Bengio deep network

Introduction

Neural networks represent one of the most fascinating and revolutionary developments in the field of artificial intelligence (AI). Their evolution has been a journey that has crossed several disciplines, including neurobiology , mathematics and computer science.

The Beginnings: The Artificial Neuron Model

In 1943, Warren McCulloch and Walter Pitts published a paper describing a mathematical model of an artificial neuron. This model, known as the "McCulloch-Pitts neuron", used thresholds and logic functions to simulate the behavior of biological neurons. Although rudimentary, this work laid the foundation for future research on neural networks.

In 1949, six years after McCulloch and Pitts had shown how neural networks could compute, McGill University physiologist Donald O. Hebb suggested how they could learn. He proposed the idea that brain connections change as we learn different tasks, and that specific new neural structures account for knowledge. Hebb's ingenious proposal dealt with the conductivity of synapses, or connections between neurons. He postulated that the repeated activation of one neuron by another through a particular synapse increased its conductivity. This change would make further activations more likely and induce the formation of tightly connected paths of neurons in an otherwise loosely connected structure :
"When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A efficiency, as one of the cells firing B, is increased".[15]

Adaptation and Learning: The Perceptron

In 1958, Frank Rosenblatt developed the perceptron, a type of neural network capable of learning to distinguish between different categories of input. The perceptron used a supervised learning algorithm to adapt the weights of connections between neurons, improving its classification ability. However, the perceptron was limited and could not solve non-linearly separable problems, as demonstrated by the book "Perceptrons" by Marvin Minsky and Seymour Papert in 1969.
Minsky's early research experiences were in fish neurology and psychology: *"Minsky did experimental work in physical optics in the physics department He also grew interested in neurology and talked to a zoology professor, John Welsh, into letting him use a roomful of equipment, where he became an expert in the neurophysiology of crayfish (a small fresh-water lobster) Under the influence of electrodes Minsky had attached to individual nerves of the animal's claw, the crayfish picked up a pencil, waved it around, and released it when Minsky excited the fibers that inhibited the closing of the claws physics or dissecting crayfish, Minsky hung around the psychology laboratory, where he was able to sample a cross-section of psychology as it existed in the late 1940s. At one end of the lab was the behaviorist camp of B. F. Skinner and his followers, who then held sway over most psychological research in the United States."*[8]

The Rebirth: The Backpropagation Algorithm

Despite the initial limitations, interest in neural networks did not disappear. In the 1980s, a crucial breakthrough was the introduction of the error backpropagation algorithm, developed by Geoffrey Hinton, David Rumelhart, and Ronald Williams. This algorithm allowed training of multilayer neural networks , overcoming the limitations of the perceptron and paving the way for more complex and powerful networks. *"The backpropagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem"*[31] A model accurate description is found from the introduction of *Learning representations by back-propagating errors* : *"We describe a new learning procedure, back-propagation, for networks of neuron-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure."*[32]

The Adventure of Deep Neural Networks

With the advancement of computer technology and the availability of large amounts of data, deep neural networks (deep learning) have become practicable. These networks, made up of multiple layers of artificial neurons, are capable of learning complex representations of data. In 2012, a major

breakthrough occurred when a deep neural network called AlexNet won the ImageNet competition, demonstrating the superiority of deep learning techniques in many tasks of visual recognition : *"Our results show that a large, deep convolutional neural network is capable of achieving recordbreaking results on a highly challenging dataset using purely supervised learning"*.[22]

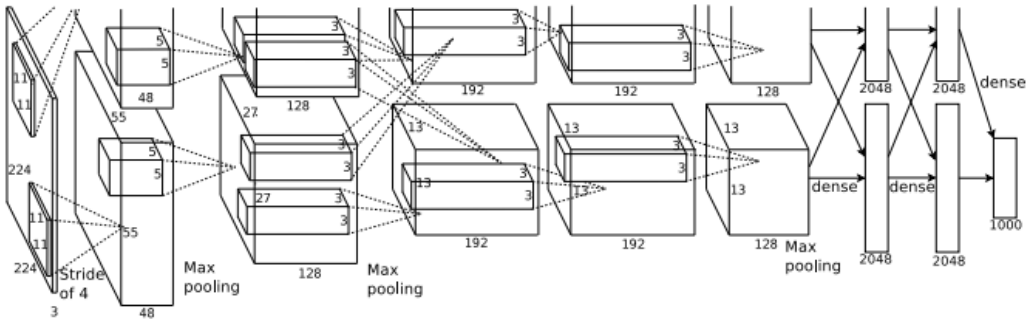

Figure 2.3: Architecture of AlexNet

2.4 Survival Analysis

2.4.1 Introduction

Survival analysis is a branch of statistics that deals with analyzing the time until an event of interest occurs. This event could be death, the failure of a machine, the relapse of a disease, or any other event that can be precisely defined and timed. Originating primarily in the field of medical research, survival analysis has found applications across various disciplines, including engineering, economics, and social sciences.[20] In engineering , and specifically in diagnosis and control, is called reliability that a part of dependability.

2.4.2 Fundamental Concepts

Survival analysis revolves around three main components[20]:

- **Survival Time**: The duration between the starting point (e.g., diagnosis of a disease) and the event of interest (e.g., death). *When doing a survival analysis, we usually refer to the time variable as survival time. We also typically refer to the event as a failure.*[20]

- **Censoring**: This occurs when the exact survival time is unknown for some subjects. This could be due to the study ending before the event occurs, the subject being lost to follow-up, or the subject withdrawing from the study. There are three type of censoring : left-censored , right-censored , interval-censored.

  Left-centered : data can occur when a person's true survival time is less than or equal to that person's observed survival time. For example the observation of the patient start after the beginning of clinical study , or in the case of HIV first (positive) event of the patient like a clinical exam is performed after the beginning of the study. It's not possible to set at what time first event occurs with the effect that survival time last longer then the period that the person has been followed.

  Right-centered : true survival time is equal to or greater than observed survival time. For example when a patients exits observation.

  Interval-censored : true survival time is within a known time interval. For example the outcome of a HIV test of a patient is within two test , first negative and second positive , it's unknown the exact time of contagion.

  There are generally three reasons why censoring may occur:
  (1) a person does not experience the event before the study ends.
  (2) a person is lost to follow-up during the study period.
  (3) a person withdraws from the study because of death (if death is not the event of interest) or some other reason (e.g., adverse drug reaction or other competing risk).

- **Survival Function**: Denoted as $S(t)$, it represents the probability that the time to event is longer than a specified time t. Mathematically, $S(t) = P(T > t)$, where T is the random variable representing the time to event.

Methods in Survival Analysis

Several statistical methods are used in survival analysis, each serving a specific purpose:

1. **Kaplan-Meier Estimator**: A non-parametric statistic used to estimate the survival function from observed survival times. It provides a step function representing the probability of survival over time and is useful for comparing survival curves between groups.

2. **Cox Proportional Hazards Model**: A semi-parametric model that assesses the effect of explanatory variables on the hazard rate, which is the rate at which the event of interest occurs. The Cox model assumes that the hazard ratios between groups are constant over time. It's a semi-parametric model because although the number of parameters is known, the distribution of the baseline hazard is not known.

3. **Parametric Models**: These models assume a specific statistical distribution for the survival times (e.g., exponential, Weibull, log-normal). They can provide more precise estimates when the assumed distribution fits the data well.

4. **Competing Risks Models**: Used when subjects may experience one of several different types of events, and the occurrence of one event prevents the occurrence of another (e.g., different causes of death). *"A competing risk is an event whose occurrence precludes the occurrence of the primary event of interest. For instance, in a study in which the primary outcome was time to death attributable to a cardiovascular cause, death attributable to a non cardiovascular cause serves as a competing event."*[2]

Applications of Survival Analysis

Survival analysis has wide-ranging applications:

- **Medical Research**: The primary application is in clinical trials and epidemiological studies to evaluate the effectiveness of treatments, compare survival rates among patient groups, and identify risk factors for diseases.[35]

- **Reliability Engineering**: Used to predict the lifespan of products and systems, device failure, assess warranty claims, and plan maintenance schedules. For example, engineers might use survival analysis to determine the probability of a machine part failing within a certain period.[35]

  Take in account differencies between the definition of reliability and avaliability in dependability analysis reliability has some property[1] :

  1. *Ability of entity E to perform a desired function under given operational conditions and for a given time interval*
  2. *Assume an entity/item E s.t. it can fail, but it cannot be repaired*
  3. *Reliability is measured as the probability that E is able to perform the required function in time interval [0,t]*

In dependability analysis avaliability has some property:

1. *Assume E can fail and can be repaired*
2. *Availability is the ability of an entity E to be able to perform at time t the required function under given operational conditions (more precisely is the probability of being in t into the desired operational state).*
3. *It can be measured as probability A(t)=Pr(E is not down at time t), assuming is working at 0*

- **Economics and Social Sciences**: Applied to study job tenure, duration of unemployment, time to divorce, and other life events. For example, economists might use survival analysis to understand how long individuals stay unemployed and what factors influence the duration. [9]
- **Marketing**: Helps in understanding customer churn, product life cycles, and the duration of customer relationships. Businesses can use these insights to develop strategies for customer retention, product development ,ranking , customer ratings, and contents size of description of a mobile app. [25] [18][35]

Challenges and Future Directions

Despite its usefulness, survival analysis faces several challenges:

1. **Censoring**: Handling censored data appropriately is crucial for accurate analysis. Improper handling can lead to biased results.
2. **Complexity of Models**: Advanced models, such as those involving time-varying covariates or competing risks, can be complex and computationally intensive.
3. **Assumptions of Models**: Many survival analysis models rely on assumptions (e.g., proportional hazards in the Cox model) that may not always hold true. Violations of these assumptions can lead to incorrect conclusions.
4. **Data Requirements**: Large and well-structured datasets are often required to achieve reliable results. In many fields, collecting such data can be challenging.

Future research and advancements in survival analysis aim to address these challenges through:

1. **Improved Computational Methods**: Enhancements in computational power and algorithms will allow for more complex and accurate models.
2. **Machine Learning Integration**: Combining survival analysis with machine learning techniques can offer more flexible and powerful predictive models, especially for high-dimensional data.
3. **Robustness to Assumptions**: Developing methods that are less sensitive to model assumptions or that can dynamically adjust to violations of these assumptions will improve the reliability of survival analysis.

2.4.3 Computation

Some definition

Letter **T** is used for the random variable for a time event, in this thesis means time of death , must me non negative.

Pr is a probability of an event ,

Survival function

$$S(t) = Pr(T > t) \tag{2.1}$$

is the *Survival function* , means the probability the patient survive after time t , or the probability of time of death is later of time t.

Lifetime distribution function

$$F(t) = Pr(T \leq t) = 1 - S(t) \tag{2.2}$$

The lifetime distribution function, conventionally denoted F, is defined as the complement of the survival function which represents the probability that the event of interest occurs earlier than t.[35][36]

$$f(t) = F'(t) = \frac{d}{dt}F(t) \tag{2.3}$$

The function f(t) is the event density function or a probability density function of S(t) (PDF) , it is the rate of death or failure events per unit time.

$$S(t) = Pr(T > t) = \int_t^\infty f(u)\,du = 1 - F(t) \tag{2.4}$$

The Survival function can be expressed as probability or a density function.

Hazard function

The hazard function h , is also denoted as λ

$$h(t) = \lim_{dt \to 0} \frac{Pr(t \leq T < t + dt | T \geq t)}{dt} = \lim_{dt \to 0} \frac{Pr(t \leq T < t + dt)}{dt \cdot S(t)} = \frac{f(t)}{S(t)} = -\frac{S'(t)}{S(t)} \tag{2.5}$$

Previuos expression states relationship between h , f, S. *"The hazard function does not indicate the chance or probability of the event of interest, but instead it is the rate of event at time t given that no event occurred before time t".*(Pag 5)[35]
Any hazard function must have these contrains :

1. hazard function must be not negative

$$\forall t > 0, h(t) \geq 0$$

2. sum of hazard must converge to infinite for t to infinite

$$\int_0^\infty h(t)\,dt = \infty$$

Cumulative hazard function

The cumulative hazard function is the integral , or the sum in case of discrete time , of the hazard rate λ or h function from 0 to t.

$$\Lambda(t) = -\log S(t) \tag{2.6}$$

$$S(t) = \exp(-\Lambda(t)) \tag{2.7}$$

$$\frac{d}{dt}\Lambda(t) = -\frac{S'(t)}{S(t)} = \lambda(t) \tag{2.8}$$

$$\Lambda(t) = \int_0^t \lambda(u)\,du \tag{2.9}$$

The interpretation of cumulative hazard and hazard rate could be the following : *"Hazard are rates, and in that they are not unlike RPM revolution per minute of an automobile engine.Cumulative hazards are the integral from zero to t of the hazard rates. Because an integral is really just a sum, a cumulative hazard is like the total number of revolutions an automobile's engine makes over a given period"*(Pag 15) [6]

Future lifetime

The expected future lifetime of a patient at time t giver is alive at time t_0 :
starting with the probability of an event before t + t_0

$$Pr(T \leq t_0 + t | T \geq t_0) = \frac{Pr(t_0 < T \leq t_0 + t)}{Pr(T > t_0)} = \frac{F(t_0 + t) - F(t_0)}{S(t_0)} \tag{2.10}$$

then the probability density

$$\frac{d}{dt}\frac{F(t_0 + t) - F(t_0)}{S(t_0)} = \frac{f(t_0 + t)}{S(t_0)} \tag{2.11}$$

the following equation , solved by integral by parts, compute the expected future lifetime as

$$\frac{1}{S(t_0)}\int_0^\infty t \cdot f(t_0 + t)dt = \frac{1}{S(t_0)}\int_{t_0}^\infty S(t)dt \tag{2.12}$$

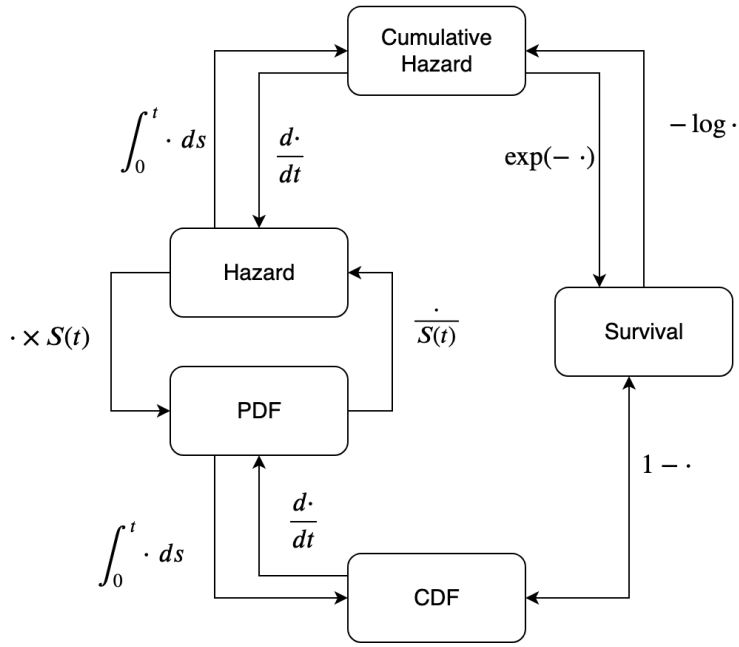

Figure 2.4: Map of mathematical entity used in survival analysis [37]

Survivability $S(t_0)$ is conventionally 1 because it's the time of the diagnosis of the patient. This simplify the function that states that expected future lifetime is the integral of S(t) from t_0 to ∞ or last observed event.

2.5 Gray Box Model

Modelling approach could be divided in :

1. **White Box** : we have theoretical knowledge of the system and the value of configuration parameters.Complete explainability of the system. Usable for prediction without dataset.
2. **Gray Box** : we have theoretical knowledge of the system but the value of configuration parameters. Parameter must be detected , for example in ANN could be trained with dataset.
3. **Black Box** : we don't have neither theoretical knowledge of the system nor the value of configuration parameters. No assumption on the inner mechanism of the system. Training time could differ according to the used technology , for example a deeper DNN takes longer. Data dataset must be avaliable and no explainability of the value of the parameter. Must be trained and then use for prediction.

In this thesis will be use a gray box modelling approach (serial approch Type I [38]) in order to detect parameter of Cox model (coefficients β , and baseline hazard) during the training phase and predict survivability , hazard and future lifetime during prediction phase. Motivation of using Grey-box model for parameter identification is also supported by experience : *"Grey-box modeling is an advantageous tool for system identification when obtained input/output experimental data are insufficiently excited. The lack of information in the data can be often replaced with some additional knowledge about the modeled system, which constricts the class of models under consideration."*[30]

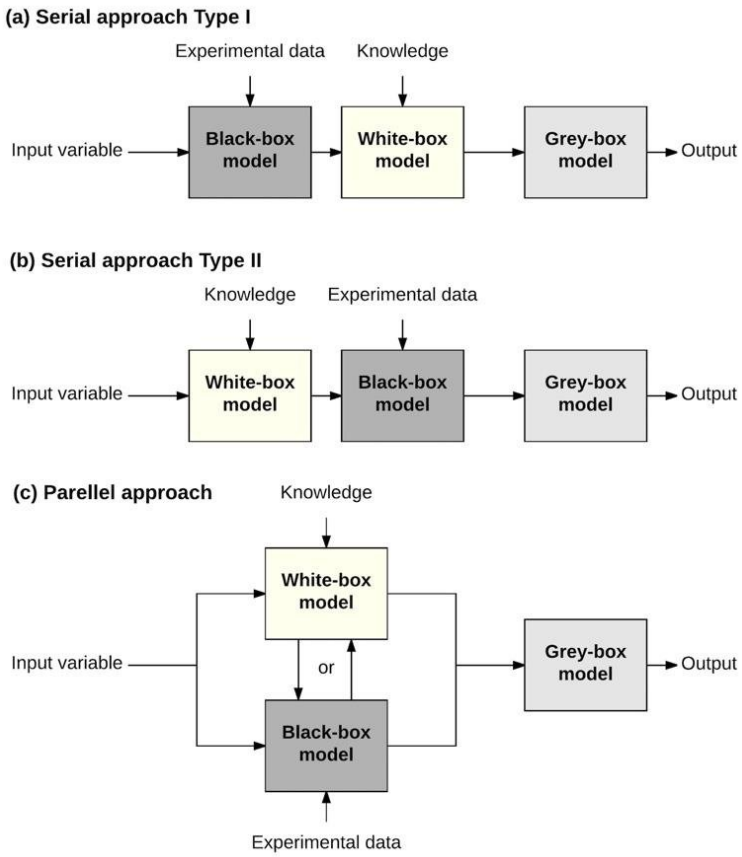

Figure 2.5: Basic grey-box modeling approaches [38]

2.6 Related Works

In this section is presented a summary of the content of a list of paper involved not deep neural network and Cox model.

A scalable discrete-time survival model for neural networks : this paper addresses the problem of survival analysis using a shallow neural model Nnet-survival, a discrete-time survival model that is theoretically justified, naturally deals with non-proportional hazards that do not represent the Cox model, with a good result for what concerns survivability.[11]

Continuous and discrete-time survival prediction with neural networks : it uses log-likelihood but does not describe a particular neural network. Define a Constant Density Interpolation (CDI) where $S(t)$ is constant within two t interval, and Constant Hazard Interpolation (CHI) where hazard is constant between 2 t interval, this method is used in various publications but its use is not considered in this thesis.[23]

Time-to-event prediction with neural networks and Cox regression : in this paper the *"proposed loss function is verified to be a good approximation for the Cox partial log-likelihood"*, extend to non proportional hazard assumption of Cox model and use SurvivalNet. [24]

Deep Neural Networks for Survival Analysis Based on a Multi-Task Framework : *"introduce a new method to calculate survival functions using the Multi-Task Logistic Regression (MTLR) model as its base and a deep learning architecture as its core"* but no neural network architecture is provide.

Five Years Survival of Patients After Liver Transplantation and Its Effective focus on Cox model

using ANN without define a specific architecture.[19]

Neural Network-Based Piecewise Survival Models focus on : the piecewise constant density model, the piecewise linear density model, the constant hazard model, and the linear hazard mode. All methods based on shaping survivability as a piecewise with different contrains, also in this case no neural achitecture model is provided.[16]

Chapter 3

Metodology

The main objective of the research is the comparison between existing software tools and a potential new approach to calculate the survival of patients undergoing liver transplantation. In this phase, generated mock data will be used. The tools used, apart from the already mentioned scikit-survival and lifelines, are Visual Studio Code as IDE, Python as programming language and Tensorflow Keras as framework for managing neural networks. A first version of the application that will have to be developed has been provided, which already contains a library for generating mock data and a method for the baseline hazard. Two neural models will be developed that have the same architecture of the input and hidden layers, differing only in the output layer, each will be identified uniquely. The developed model is identified from this point as rm (regressive model).

3.1 Reference Library

Software tools used as reference are **scikit-survival** and **lifelines** which provide an array of function related to survial analisys. The main characteristics that will be compared are those related to survivability, hazard and the parameters of the Cox model. Both library expose training capabilities from a dataset in a form of time-to-event format that containt the event time and 2 flag for deceased and censored. Occasionally scikit-survival is referred as sksurv.

3.2 Architecture

This section expose the achitecture of the application and the role of it's main module

The modules described are to be understood in general as an abstraction of the data generation pipeline and output benchmarks such as plots or tables, but despite this some "main" files of the application work as described in the architecture.

1. **Data Generation** : is the module dedicated to the generation of mock data, the algorithm used is the Discrete-time proportional hazards, the module accepts as parameters the seed and the baseline hazard method, the censoring ratio, the number of beta coefficients , produces as output a Pandas dataset , saves the dataset to a file

2. **Data Build Model** : the module is dedicated to the creation of neural models with the Tensorflow Keras framework, the creation parameters such as the number of input and output

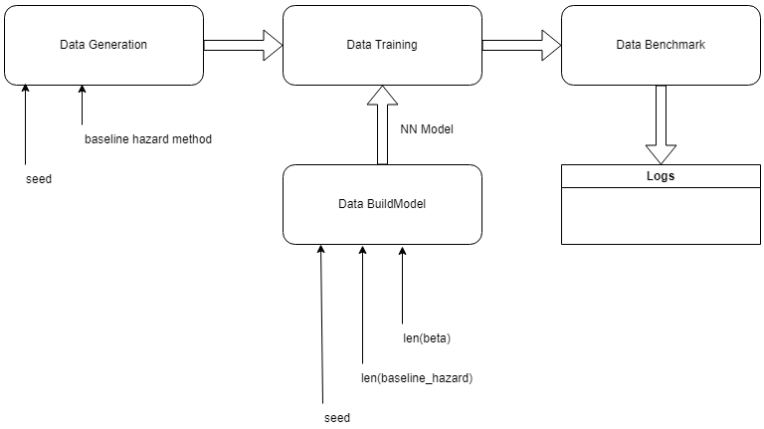

Figure 3.1: Architecture of application
[38]

are passed from a global configuration file. It is possible to configure the initial values of the network weights through a seed.

3. **DataTraining** : the module is dedicated to training models, both SkSurv and Lifelines and rm , and is largely configurable. Hyperparameters are configurable through a global configuration file.

4. **Data Benchmark** : the module is dedicated to processing parameters

3.3 Creating The Model

The choice of the model is the result of attempts with deep networks. The research must satisfy the requirement of building a predictive model according to the Cox model, the firsts attempts went in the direction of building deep networks that allows to predict survivability using T as only output label, but this path couldn't identify, by architecture, the *beta* coefficients and the baseline hazard, the first attempts also took into account the censoring event. Despite the exploration of hyperparameters, the increase of states, the activation functions or the loss function, it was not possible to achieve a significant result. The choice at this point fell on a shallow network that implements the Cox function, with the T, survivability and optionally hazard as output.

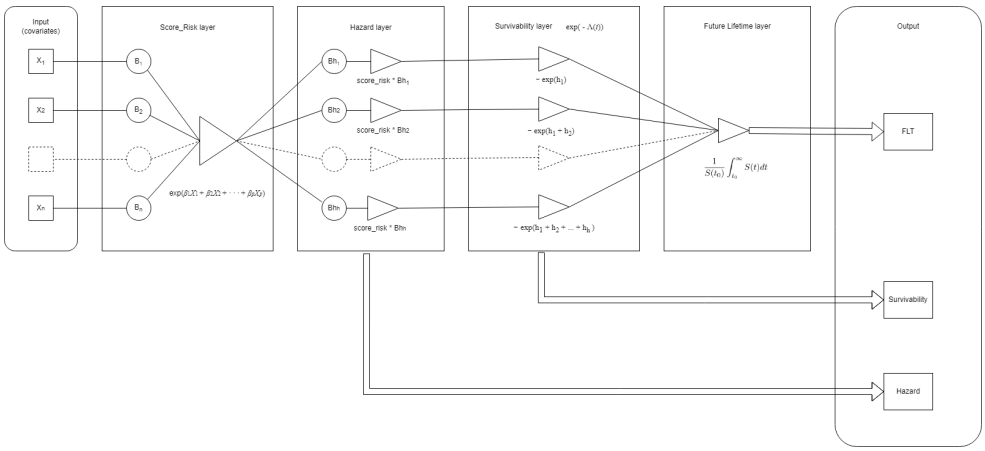

Figure 3.2: Neural network Cox layers

There are two versions of the neural model, one with hazard, survivability and FLT (expected future lifetime) called *hsf* and one with survivability and FLT. Both are used in the experimentation phase. It has to consider Survivability layer as a cumulative sum of previous layer , in python is computed by np.cumsum function, image is a semplification. The model proposed do not need bias parameter.

3.4 Parameter Identification

Model has two main goal : identify parameter for cox model and predict key features of survivability analysis. It's clear that accuracy of the first affect performance of the second. More model parameters are similar to the generated ones more accurate will be the output of the network. For the simplicity of the model the expected result is to perform better then other tools.

3.5 Design of the output

In order to train the neural network, it is necessary to build an output label function or ground truth with which the network can compare to minimize the loss with the value predicted by the network itself. Two heuristic functions have been identified, in this section their characteristics, constraints and motivations will be explained. The calculations are intended for a continuous-time context.

Heuristic E1

$$FLT_e = T \tag{3.1}$$

$$S_e(0) = 1 \tag{3.2}$$

$$S_e(T) = \frac{T}{len(H)} \tag{3.3}$$

$$S_e(t >= len(H) + 1) = 0 \tag{3.4}$$

Where $S(t)$ is survivability from theory and $S_e(t)$ is survivability computed from heuristic.

H (horizon) is the array of times and len(H) the lenth.

Patients who survive after horizon are dropped from observation and are therefore "almost" censored, it is assumed that they die on t after horizon for a conservative approach and computation semplification, therefore at H + 1 all patients are dead, the hypothesis is that H is too small. It is necessary to make a consideration about the calculation of FLT_e starting from the survivability curve S_e created with the heuristic E1 :

$$\int_H^{H+1} S_e(t)dt > 0 \tag{3.5}$$

$$\Delta FLT_e(t) = FLT_e(t) - FLT \tag{3.6}$$

$$\forall t > 0, \Delta FLT_e(t) = \frac{t}{2} > 0 \tag{3.7}$$

A heuristic that imposed $S_e(H) == 0$ would cancel the contribution of $\Delta FLT_e(t)$, but would impose as a constraint that no patient survives to time H, this is considered a violation of the generated data. During the experimental phase it will emerge that the FLT prediction is conservative despite the label being optimistic.

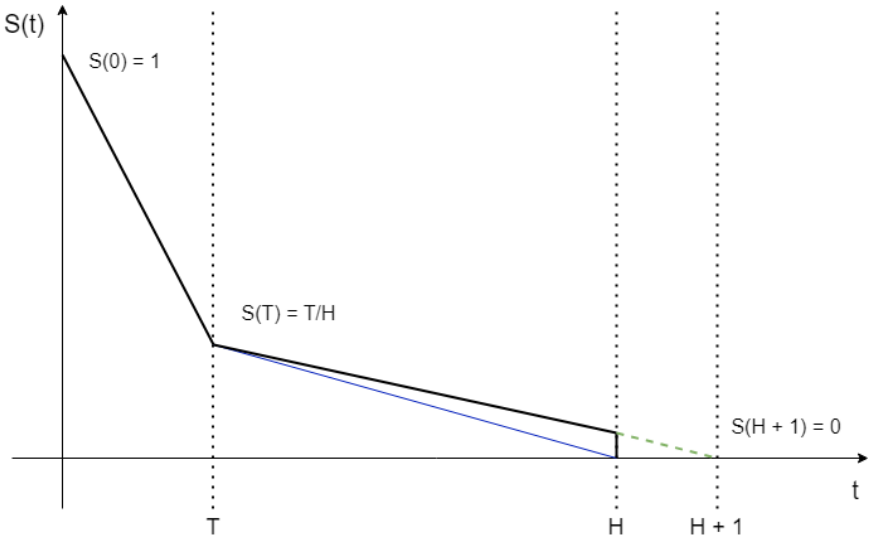

Figure 3.3: Example not in scale of survivability curve for Heuristic E1

Heuristic E2

$$FLT_e = T \tag{3.8}$$

$$S_e(0) = 1 \tag{3.9}$$

$$S_e(T) = \frac{1}{2} \tag{3.10}$$

$$S_e(t >= 2 * T) = 0 \tag{3.11}$$

In this case we consider the survivability of the patient $\frac{1}{2}$ at T, this is motivated by considering that at T the patient is at his MTTF (mean time to failure) so he has a 50% probability of a death event, in order to guarantee the conservation of the first constraint that comes from the theory then the $S(t) = 0$ for $t = 2 * T$, essentially the function is a right-angled triangle with height $S(0) = 1$ and base 2 * T. In this case there is no $\Delta FLT_e(t)$. It is necessary to make a consideration regarding the calculation of the FTL starting from the survivability, given that $FLT_s = \int_{t_0}^{\infty} S(t)dt$, the computation in the model is performed at discrete-time or it is a summation of array values that covers up to H + 1 , the survivability of the E2 that exceeds H + 1 is not taken into consideration.

$$FLT_s(T) = \begin{cases} T <= \frac{H}{2} \Longrightarrow \int_{t_0}^{\infty} S(t)dt \\ \\ T > \frac{H}{2} \Longrightarrow \int_{t_0}^{H} S(t)dt \end{cases} \tag{3.12}$$

This means that for $T > \frac{H}{2}$ the FLT_s will be conservative.

The equation 3.12 has as a consequence the rewriting of the equation 3.11 which concerns the survivability at the last event.

$$\begin{cases} T <= \frac{H}{2} \Longrightarrow S_e(t >= 2 * T) = 0 \\ \\ T > \frac{H}{2} \Longrightarrow S_e(t >= len(H) + 1) = 0 \end{cases} \tag{3.13}$$

During the experimentation phase, it will be evaluated how this feature impacts the performance of the metrics.

There are two versions of the neural model: *sf* and *hsf* , for each it is necessary that the instantiation of the model in pair with its output.

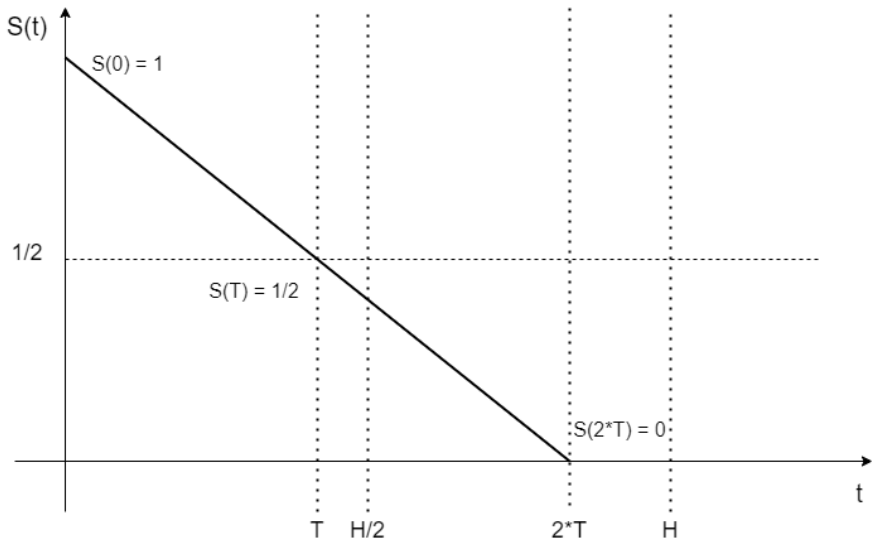

Figure 3.4: Example not in scale of survivability curve for Heuristic E2 in case $T <= \frac{H}{2}$

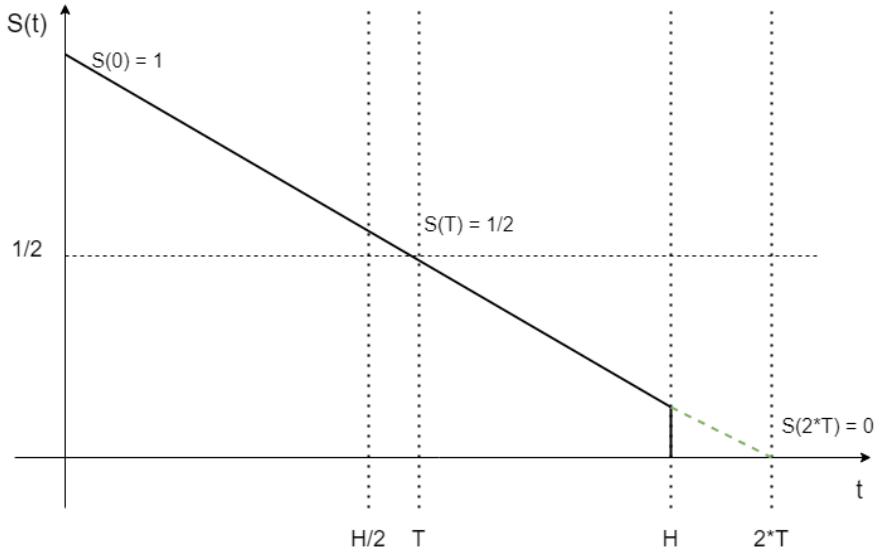

Figure 3.5: Example not in scale of survivability curve for Heuristic E2 in case $T > \frac{H}{2}$

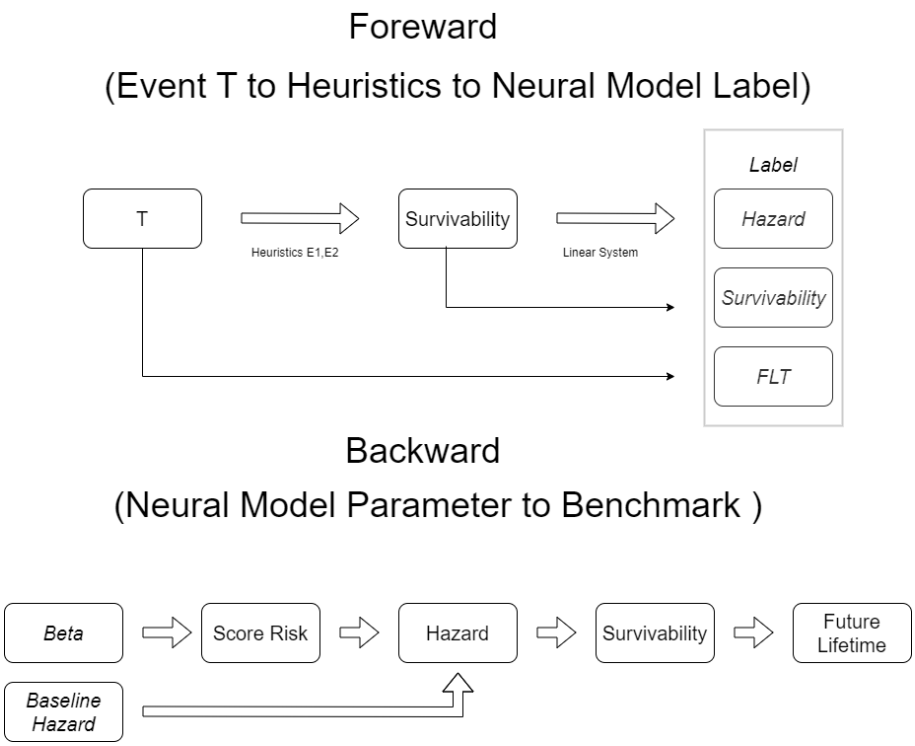

Figure 3.6: Pipeline for neural model label creation and benchmarks

Hazard

Considerations on the calculation of the hazard are necessary. The model in the *hsf* version requires the calculation of the hazard starting from the survivability which is the heuristic function. In the Cox model, the calculation of the hazard starts from the covariates X_i , the coefficients $beta_i$ and the baseline hazard , while the calculation of the survivability is obtained once the hazard has been obtained. The heuristic function instead defines a survivability function and to obtain its corresponding hazard it is necessary to proceed "backwards". Starting from the definition of survivability as a function of the hazard

$$S(t) = -\exp(\Lambda(t))$$

and making it explicit for various t

$$\begin{cases} S(0) = -\exp(h(0)) \\ S(1) = -\exp(h(0) + h(1)) \\ \quad \dots \\ S(n) = -\exp(h(0) + h(1) + \dots + h(n)) \end{cases} \tag{3.14}$$

it is possible to obtain a system of linear equations of len(H) equations in len(H) unknowns

$$\begin{cases} \ln S(0) = -(h(0)) \\ \ln S(1) = -(h(0) + h(1)) \\ \quad \dots \\ \ln S(n) = -(h(0) + h(1) + ... + h(n)) \end{cases} \tag{3.15}$$

or in more explit form better usable by python numpy library

$$\begin{cases} \ln S(0) = -h(0) \\ \ln S(1) = -h(0) - h(1) \\ \quad \dots \\ \ln S(n) = -h(0) - h(1) - \dots - h(n) \end{cases} \tag{3.16}$$

whose solution are the values of the hazard for each t. It is possible to consider the fact that in hazard is invariant meaning that an infinite number of combinations of covariates X_i, from the coefficients $beta_i$ and from the baseline hazard have as output according to the Cox model the same value of hazard, therefore it is already possible at this point to identify that it will not be possible to correctly identify the network parameters through its training.

Chapter 4

Experiment

4.1 Data Generation

The Data Generation module requires a subsection for further information.

4.1.1 Parameter input

In this section, the steps for generating the data will be briefly described.

The data generation algorithm takes as input this parameter :

1. number of *Beta* coefficients
2. number of patients
3. split ratio train/test
4. censoring ratio
5. baseline hazard method
6. baseline hazard length (or number of observed tte)
7. random data generation seed

A library provide 4 baseline hazard method : orig, weibull, lognormal, nbinomial.

The module has in outputs:

1. the covariates X_i for each patient, uniform random distribution between 0 and 1 excluded
2. the label for training the network: survivability, hazard and futurelifetime

The data structure *(*horizon) is defined as follows

horizon = [number of observed tte + 1 , min follow-up , max follow-up , step follow-up]

this data structure was used in the preliminary version but is no longer used in the latest , but for backwards compatibility with the previous data generation it has remained.

The scikit-survival software library has some constraints for training its model

1. Constraint 1 : scikit-survival requires that test survival event lie within the range of train survival event
2. Constraint 2 : horizons.max() <= test.max() or horizons.min() < test.min(), TTE of test must encompass min end max time of horizon
3. Constraint 3 : every split ,segment in *split_ data_ from_ dataframe_ multisegment python function*, must have at least 2 event, at least 1 for train and 1 for test

The split function must satisfy a further constraint : every interval defined in horizon is defined as a segment or split and in each segment the train test rate is satisfied, this implies that in particular that in the last segment, the one after the last follow up must be at least one test sample, it must not be the last one.

4.1.2 Code

To produce the dataset, it is necessary to produce the input parameters sequentially. In this section, some key lines of code are dedicated to data generation.

Coefficients β

```
covars_coeff
    =
np.random.uniform(low=config.beta_low, high=config.beta_high, size=self.args.n_covariates)
```

Covariates X_i

```
covars = np.random.rand(self.args.n_patients, self.args.n_covariates)
```

Baseline hazard

```
bhm = BaselineHazardMethods()
self.baseline_hazard
=
[bhm.get_method(self.baseline_hazard_method_name)(_) for _ in range(config.end_horizon)]
```

Function for generate dataset E1

```
simulation_df
=
generator_.query_E1_v1( h=bhm.get_method(self.baseline_hazard_method_name),
                        x=covars,
                        beta=covars_coeff)
```

Function for generate dataset E2

```
simulation_df
=
generator_.query_E2_v2( h=bhm.get_method(self.baseline_hazard_method_name),
                        x=covars,
                        beta=covars_coeff)
```

Structure for dataset

```
H = self.discrete_time_proportional_hazards(self.horizon,h,x,beta)

n_covars = x.shape[1]

cov_name = [f"x_{i}" for i in range(n_covars)]

data = DataFrame(
    columns=["horizon"] + cov_name + ["h_t",
                                      "deceased",
                                      "censored",
                                      "TTE",
                                      "hazard",
                                      "survivability",
                                      "futurelifetime"],
    dtype=object
)
```

Where "TTE","hazard","survivability" and "futurelifetime" are generated as described in methodology. Futurelifetime is normalize between 0 and 1.

Complete loop for data generation

```
#linear equation
A = self.get_matrix_A(config.baseline_hazard_size + 1)

random.seed(global_prj.seed)
# for each patient compute flags
for i_patient in range(len(x)):

    tte_value = -1
    j_time = 1
    is_censored = 0
    is_deceased = 0
    survivability = np.zeros(config.baseline_hazard_size + 1)
    survivability[j_time - 1] = 1
```

```
future_lifetime = 0
while  j_time < self.horizon:

    chance = random.random()
    if chance <=  self.p_censoring:

        tte_value = j_time
        is_censored = 1
        break
    elif    (self.p_censoring < chance)
            and
            (chance <=  self.p_censoring + H[j_time, i_patient]):
            # corresponds to chance > 1 - P(survive)
        tte_value = j_time
        is_deceased = 1
        future_lifetime = tte_value / (config.baseline_hazard_size)
        break
    else:
        pass
    j_time += 1
if tte_value >= 0 :
    pass
else:
    tte_value = config.baseline_hazard_size
    if is_deceased != 1:  # needed when patient dies at last observation
        # survived
        is_censored = 1 # last time horizon
        future_lifetime = tte_value / (config.baseline_hazard_size) # 1 by definition

survivability = self.get_survivability_E1(tte_value,config.baseline_hazard_size)
B = np.log(survivability)
hazard = np.linalg.solve(A, B)

row = np.zeros( 1 + n_covars + 4 + 3, dtype=object)

i_column:int = 0

row[i_column] =  config.baseline_hazard_size
i_column += 1

for i, v in enumerate(x[i_patient]):
    row[i_column + i] = v
```

```
    row[i_column + n_covars + 0] = H[-1, i_patient]
    row[i_column + n_covars + 1] = int(is_deceased)
    row[i_column + n_covars + 2] = int(is_censored)
    row[i_column + n_covars + 3] = int(tte_value)

    row[i_column + n_covars + 4] = hazard
    row[i_column + n_covars + 5] = survivability
    row[i_column + n_covars + 6] = future_lifetime

    data.loc[i_patient] = row
```

Previous line expose how data is generated , how censoring and deceased event are choosen and how are stored in a data structure that will be saved as json file.

4.1.3 Parameter Value

Data generation has this parameter value for all the duration of experiment :

1. number of *Beta* coefficients = 7
2. number of patient s= 10000
3. split ratio train/test = 0.2
4. censoring ratio = 0.01
5. baseline hazard method = [orig, weibull, lognormal, nbinomial]
6. baseline hazard length (or number of observed tte) = 60
7. random data generation seed = [40128,40129,40131,40134]
8. random model seed = [42,2001,1984,10191,100,221,451,2024]

4.2 Baseline distribution

The 4 baseline hazards used have constant values throughout the experiment.

1. orig : log_correction = 1.002, risk_correction = 3, decease_rate = 0.001
2. weibull : weibull_lambda = 60, weibull_kappa = 5
3. lognormal : lognormal_mu = 2, lognormal_sigma = 0.9
4. nbinomial : negative_binomial_n = 200, negative_binomial_p = 0.9

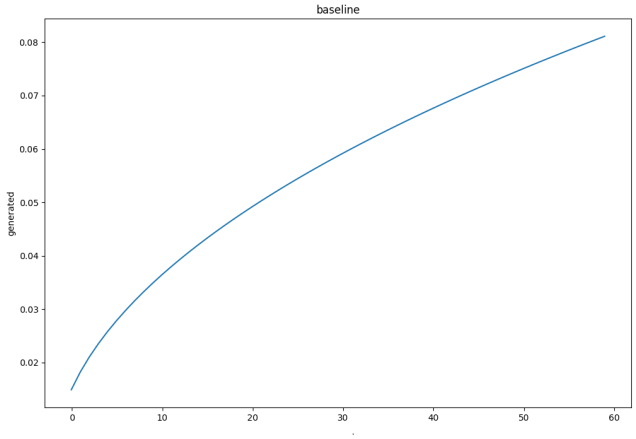

Figure 4.1: Baseline for orig method

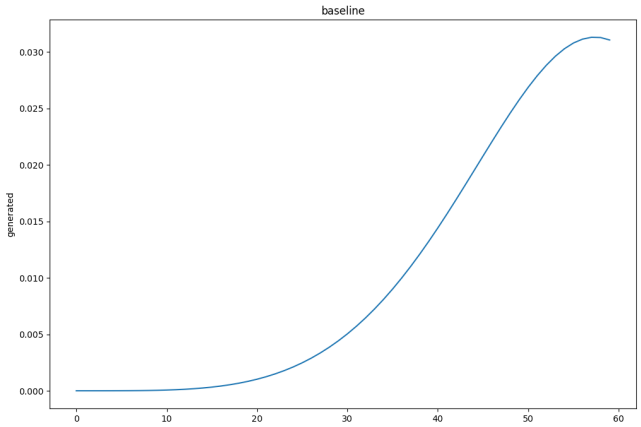

Figure 4.2: Baseline for weibull method

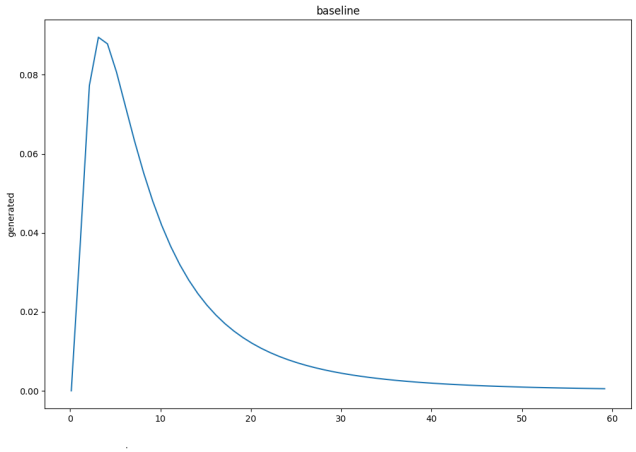

Figure 4.3: Baseline for lognormal method

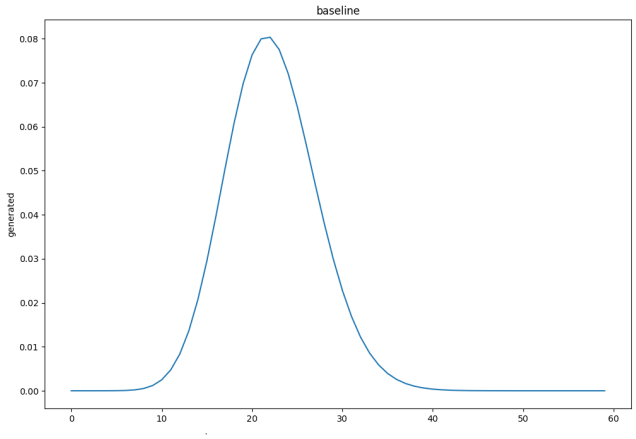

Figure 4.4: Baseline for nbinom method

4.3 Pipeline

The generation of results is done through the execution of scripts. The scripts have input parameters defined in configuration files such as *config.py* or directly in the main of the executed file.

Generate Data

First the file *generate_data_Ex.py* is followed where x is the version of the heuristic. *generate_data_Ex.py* allows the generation of mock data, both those of typical inputs of a time-to-event problem such as the time of the event (T) whether it is a death or censoring, and the output to be provided to the network as supervised labels such as the hazard values, survivability for each expected time and the T. The files are named data.<bm>.json , baseline_hazard.<bm>.npy and beta.<bm>.npy where <bm> are the methods for generating the baseline hazard.

Generate X Y

generate_x_y_Ex.py allows the creation of the train and test files of x and y for the neural network in its *hsf* or *sf* version starting from the data.<bm>.json , baseline_hazard.<bm>.npy and beta.<bm>.npy.

Generate benchmark

In this phase, all the models are trained, both the sksurv and lifelines ones and the rm models. In input, there are the files from the previous phase and in output, the neural models and the benchmark files with the *beta* and *baseline_hazard* parameters extracted from all the models and collected in files of the type <bm>.bmk.csv. The script used has a name of the form *generate_benchmark_Ex.py*.

Generate Plot

This phase is composed of a set of scripts with the aim of generating drawings, tables starting from the benchmark files generated in the previous phase.

4.4 Robusteness of the seeds

The initialization of the parameters of the neural network can disturb the training result. In this paragraph the results of a training with various *seed* will be illustrated:

$$42, 2001, 1984, 10191, 100, 221, 451, 2024$$

During the exposure of the results in generate_plot_Ex_F3.py the metrics that will demonstrate the substantial robustness of the network to the initialization of the weights will be visible.

4.5 Results

In this section the results of the experiments will be shown, divided by script.

4.5.1 F1 - Expected Future Lifeline Error

The *generate_plot_Ex_F1.py* script outputs box plots of FLT errors on the sksurv-kit , lifelines and rm models. According to seaborn documentation: *"A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be "outliers" using a method that is a function of the inter-quartile range."* The error calculation is carried out on all patients in a dataset, in particular on the dataset.1. As regards the rm model, all the seeds of the models (8) are taken into consideration.

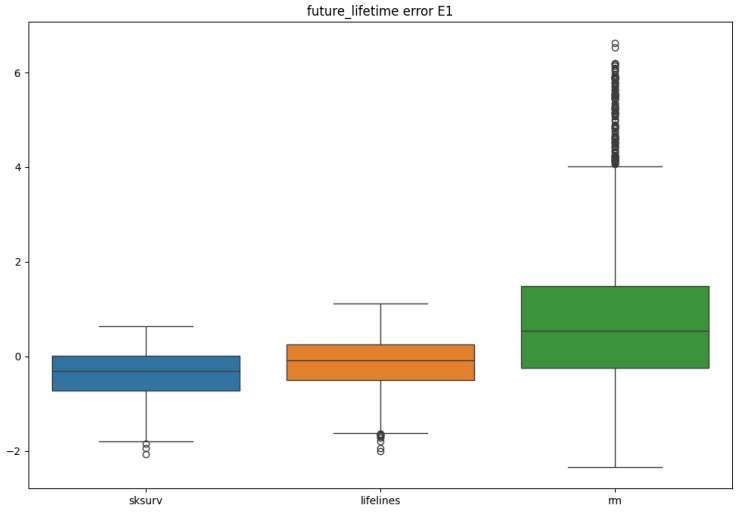

Figure 4.5: Error of future lifetime for orig baseline E1

-	sksurv	lifelines	rm
count	2648	2648	2648
mean	-0.373303	-0.148108	0.837364
std	0.488250	0.553287	1.567829
min	-2.070085	-2.008368	-2.340332
25%	-0.715417	-0.495635	-0.238274
50%	-0.301691	-0.088969	0.547827
75%	0.019571	0.258855	1.480802
max	0.637465	1.113489	6.625526

Table 4.1: Dataframe.describe() for orig baseline E1

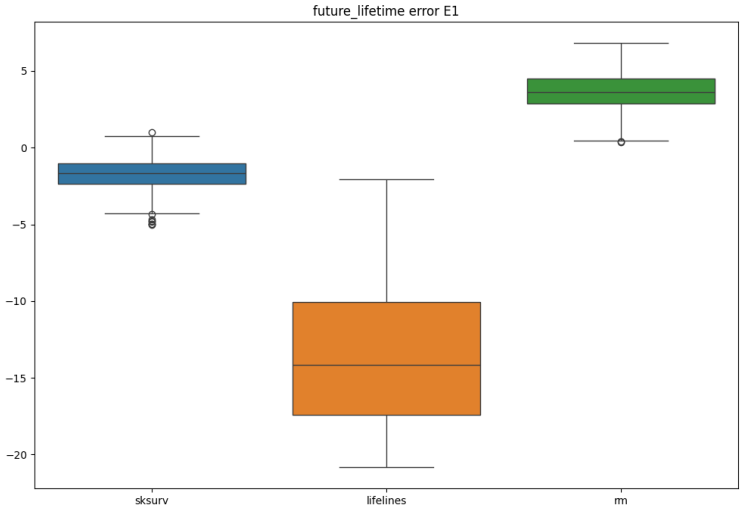

Figure 4.6: Error of future lifetime for lognormal baseline E1

-	sksurv	lifelines	rm
count	1508	1508	1508
mean	-1.708762	-13.512008	3.666658
std	0.974322	4.390788	1.196521
min	-5.041092	-20.819822	0.366877
25%	-2.340052	-17.389882	2.871436
50%	-1.668061	-14.150164	3.633177
75%	-1.032679	-10.044308	4.495265
max	0.983616	-2.058698	6.822094

Table 4.2: Dataframe.describe() for lognormal baseline E1

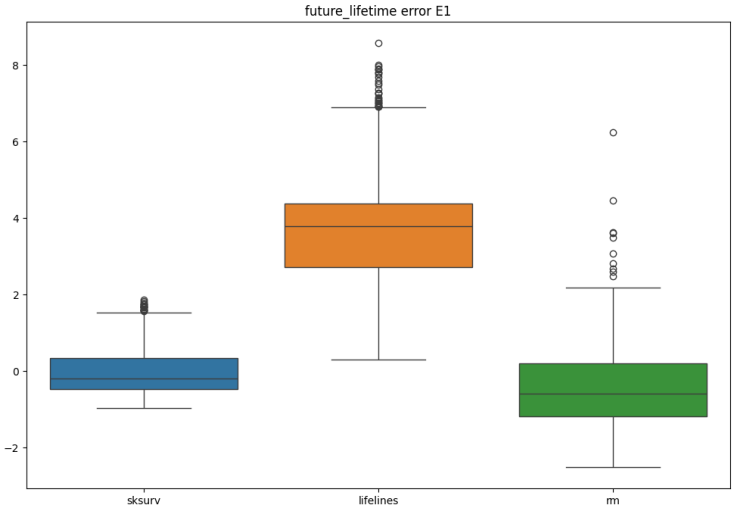

Figure 4.7: Error of future lifetime for weibull baseline E1

-	sksurv	lifelines	rm
count	1494	1494	1494
mean	-0.020129	3.685576	-0.469571
std	0.590456	1.440856	0.936752
min	-0.961195	0.297831	-2.505561
25%	-0.468129	2.717292	-1.188734
50%	-0.186019	3.783859	-0.576676
75%	0.342947	4.390283	0.214644
max	1.877410	8.580904	6.236416

Table 4.3: Dataframe.describe() for weibull baseline E1

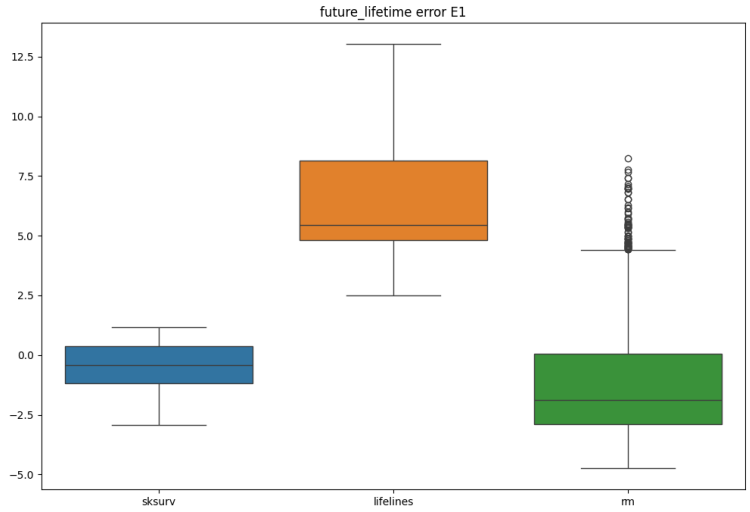

Figure 4.8: Error of future lifetime for nbinom baseline E1

-	sksurv	lifelines	rm
count	2806	2806	2806
mean	-0.452776	6.384822	-1.176806
std	0.916857	2.636889	2.268408
min	-2.934620	2.483639	-4.736751
25%	-1.168143	4.818322	-2.884139
50%	-0.424757	5.450116	-1.876407
75%	0.385262	8.129015	0.038940
max	1.161252	13.030403	8.251937

Table 4.4: Dataframe.describe() for nbinom baseline E1

From these representations, which put different models in competition, it becomes evident that the rm model can be defined as qualitatively competitive with other models created by widely used frameworks. It can also be noted that there is a dependence on the baseline distribution.
The hypothesis that can be made is that since the E1 heuristic approximates an exponential, that is, a function that is in the form:

$$x^{\lambda t}$$

distributions that have a similarity to an exponential distribution have a smaller error.

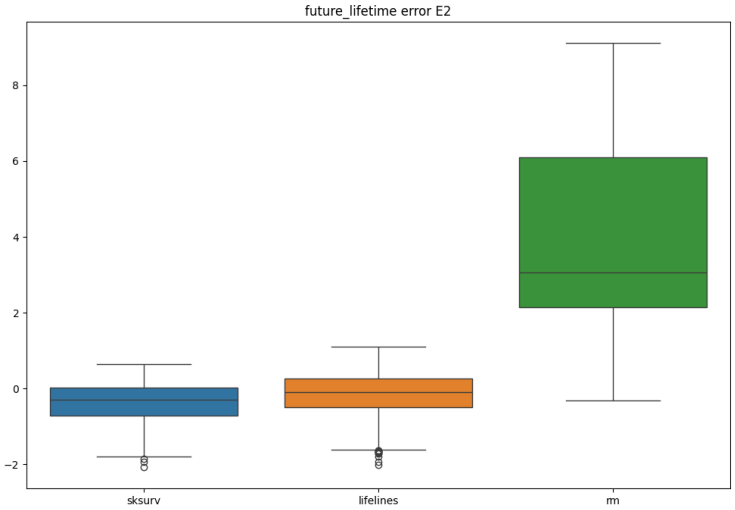

Figure 4.9: Error of future lifetime for orig baseline E2

-	sksurv	lifelines	rm
count	2648	2648	2648
mean	-0.373303	-0.148108	3.908155
std	0.488250	0.553287	2.256224
min	-2.070085	-2.008368	-0.322304
25%	-0.715417	-0.495635	2.146486
50%	-0.301691	-0.088969	3.068770
75%	0.019571	0.258855	6.101203
max	0.637465	1.113489	9.113644

Table 4.5: Dataframe.describe() for orig baseline E2

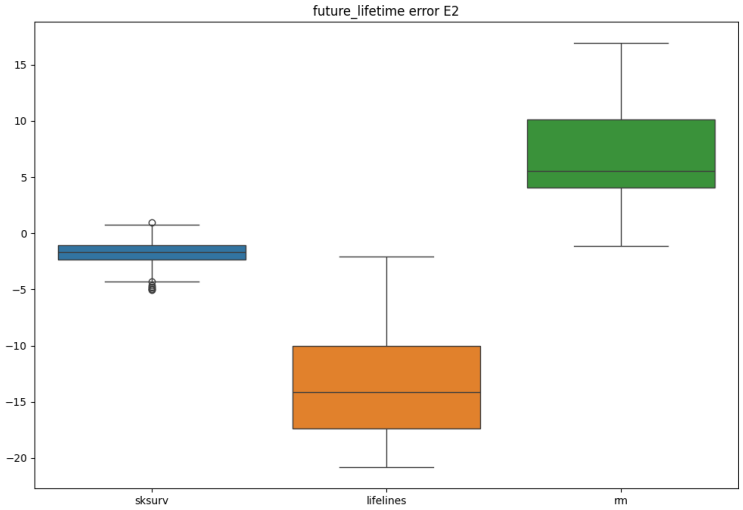

Figure 4.10: Error of future lifetime for lognormal baseline E2

-	sksurv	lifelines	rm
count	1508	1508	1508
mean	-1.708762	-13.512008	6.826028
std	0.974322	4.390788	3.922383
min	-5.041092	-20.819822	-1.138089
25%	-2.340052	-17.389882	4.032566
50%	-1.668061	-14.150164	5.513701
75%	-1.032679	-10.044308	10.107318
max	0.983616	-2.058698	16.937363

Table 4.6: Dataframe.describe() for lognormal baseline E2

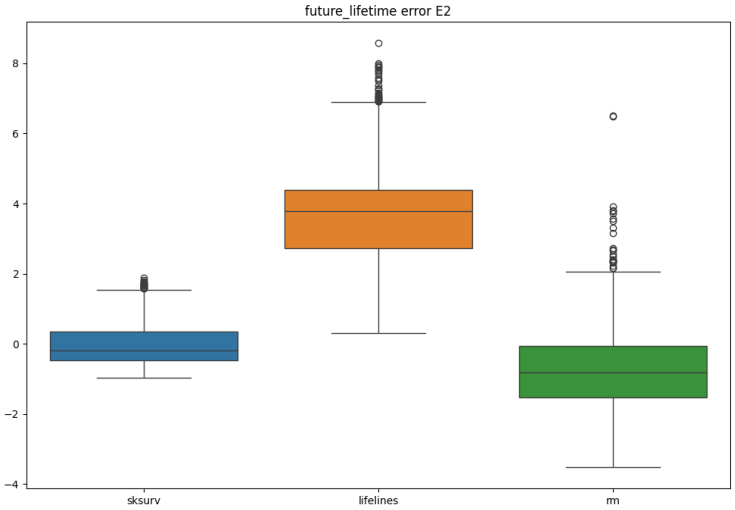

Figure 4.11: Error of future lifetime for weibull baseline E2

-	sksurv	lifelines	rm
count	1494	1494	1494
mean	-0.020129	3.685576	-0.767771
std	0.590456	1.440856	1.085870
min	-0.961195	0.297831	-3.518409
25%	-0.468129	2.717292	-1.539793
50%	-0.186019	3.783859	-0.808041
75%	0.342947	4.390283	-0.057648
max	1.877410	8.580904	6.502362

Table 4.7: Dataframe.describe() for weibull baseline E2

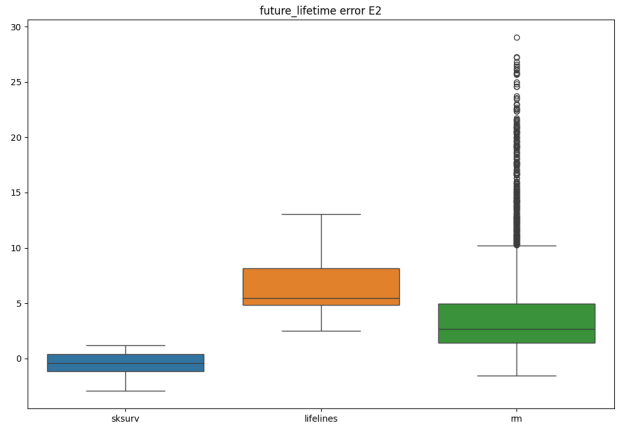

Figure 4.12: Error of future lifetime for nbinom baseline E2

-	sksurv	lifelines	rm
count	2806	2806	2806
mean	-0.452776	6.384822	4.213793
std	0.916857	2.636889	4.695931
min	-2.934620	2.483639	-1.566870
25%	-1.168143	4.818322	1.429265
50%	-0.424757	5.450116	2.674796
75%	0.385262	8.129015	4.951724
max	1.161252	13.030403	29.049124

Table 4.8: Dataframe.describe() for nbinom baseline E2

The E2 heuristic performs worse than E1, this confirms the hypothesis that an output label distribution similar to an exponential is more similar to the generated distribution.
The last consideration that can be made is that the model with E1 heuristic performs for some distribution as well or better than the lifelines model but still worse than sksurv-kit.

4.5.2 F2 - Survivability Samples

In this paragraph the survivability of the three models are put in competition for the entire observation time. Three generated patients were taken and the survival curves are shown only for the lognormal baseline.

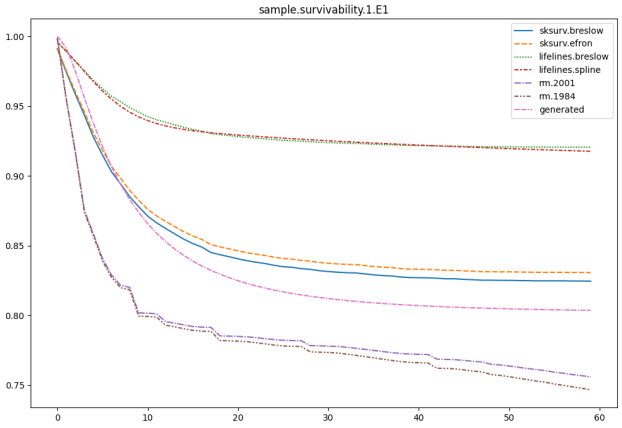

Figure 4.13: Survivability curve for sample patient 1 with lognormal baseline E1

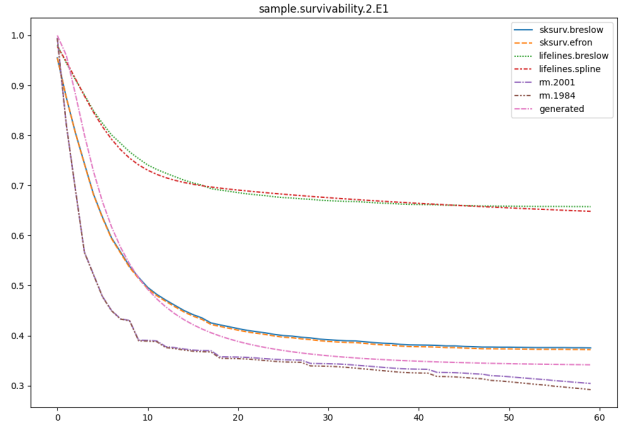

Figure 4.14: Survivability curve for sample patient 2 with lognormal baseline E1

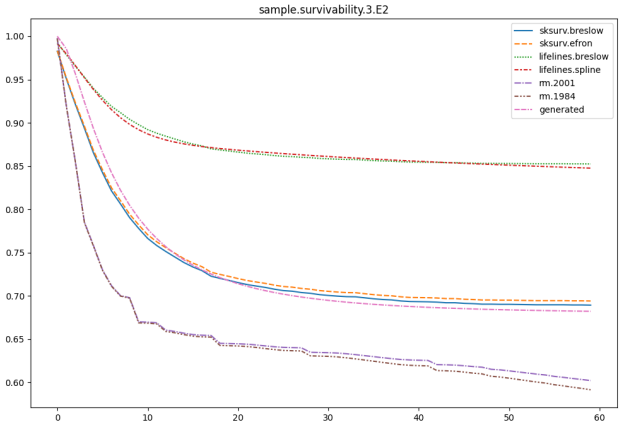

Figure 4.15: Survivability curve for sample patient 3 with lognormal baseline E1

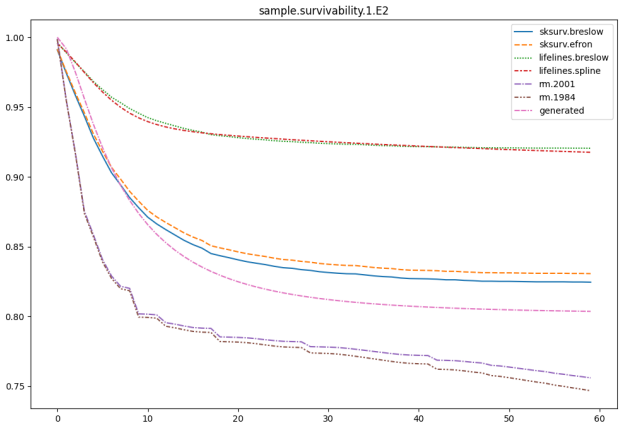

Figure 4.16: Survivability curve for sample patient 1 with lognormal baseline E2

The E1 heuristic for these samples is better than the one for E2 , the most interesting aspects are that the rm model performs better than lifelines both because it is more precise and because it is conservative compared to lifelines which is optimistic since F1, and in general for the prediction of survivability or FLT it is preferable to have a conservative result.

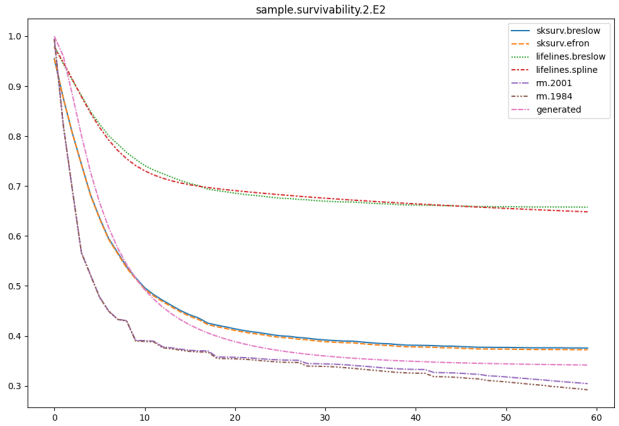

Figure 4.17: Survivability curve for sample patient 2 with lognormal baseline E2

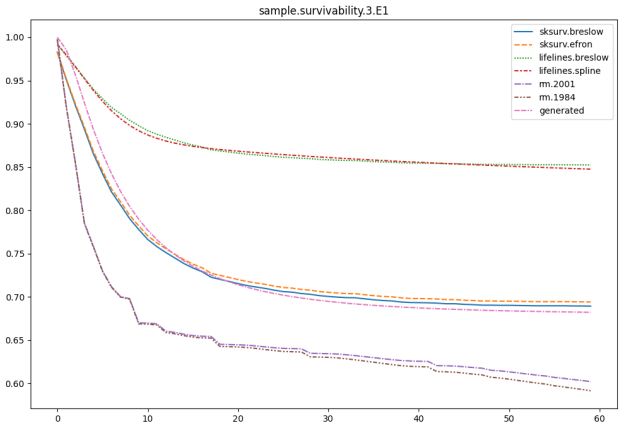

Figure 4.18: Survivability curve for sample patient 3 with lognormal baseline E2

4.5.3 F3 - Survivability Error

In this section, the metrics for the survivability error of each model are presented only for the lognormal baseline hazard generation method.

metrics	sksurv.breslow	sksurv.efron	lifelines.breslow	lifelines.spline
mean	-0.01244	-0.01149	-0.21346	-0.21290
var	0.00086	0.00092	0.00889	0.00901
mse	0.00102	0.00105	0.05445	0.05434
mape	6.25	5.66	79.40	79.590

Table 4.9: Survivability error of sksurv and lifelines

E1

	sf	hsf
count	361920	361920
mean	0.05653	0.05956
std	0.05110	0.06412
min	-0.04939	-0.05802
25%	0.02483	0.01976
50%	0.04718	0.04557
75%	0.07354	0.07721
max	0.41009	0.45686

Table 4.10: Dataframe.describe() for survivability error of model lognormal baseline E1

metrics	sksurv.breslow	sksurv.efron	lifelines.breslow	lifelines.spline	rm.sf.42	rm.hsf.42
mse	9.30E-05	9.38E-05	0.00030	0.00029	0.00107	0.00245
rmse	0.00964	0.00968	0.01757	0.017206	0.03279	0.04956
mape	3.29E+17	3.32E+17	1.42E+17	1.60E+17	4.02E+16	2.64E+14

Table 4.11: Metrics for baseline hazard E1

metrics	sksurv.breslow	sksurv.efron	lifelines.breslow	lifelines.spline	sf.42	hsf.42
mse	0.00502	0.00764	0.00493	0.00471	0.016555	0.02022
rmse	0.07086	0.08742	0.07027	0.06869	0.12866	0.14221
mape	12.21	14.63	12.24	12.02	14.65	15.57

Table 4.12: Metrics for *Beta* E1

E2

-	sf	hsf
count	45240	45240
mean	0.068420	0.148443
std	0.048863	0.092878
min	-0.023171	-0.064375
25%	0.037589	0.075524
50%	0.058866	0.156753
75%	0.085200	0.209108
max	0.405986	0.557202

Table 4.13: Dataframe.describe() for survivability error of model lognormal baseline E2

metrics	sksurv.breslow	sksurv.efron	lifelines.breslow	lifelines.spline	rm.sf.42	rm.hsf.42
mse	9.30E-05	9.38E-05	0.00030	0.00029	0.00111	0.00878
rmse	0.00964	0.00968	0.01757	0.01720	0.03338	0.09370
mape	3.29E+17	3.32E+17	1.42E+17	1.60E+17	7.94E+16	126.79

Table 4.14: Metrics for baseline hazard E2

metrics	sksurv.breslow	sksurv.efron	lifelines.breslow	lifelines.spline	sf.42	hsf.42
mse	0.00502	0.00764	0.00493	0.00471	0.01244	0.15335
rmse	0.07086	0.08742	0.07027	0.06869	0.11156	0.39161
mape	12.21	14.63	12.24	12.02	13.38	45.34

Table 4.15: Metrics for *Beta* E2

Consideration

Using operator $>$ as "performs better" (mean lower) it's possible make some considerations that can be done are that:

1. the rm model is robust to the initialization seed of the weight parameters of the neural networks (see Appendix)
2. sksurv is the model, with both methods, that performs better overall
3. E1 performs better than E2 on future lifetime (mean , var)
4. model performs better than lifelines on future lifetime , in 6 out of 8 case mean is less , and when mean is higher it's conservative
5. E1 performs better than E2 on survivability (mse and mape)
6. baseline hazard parameter identification (mape): E2 $>$ E1 for both version
7. beta parameter identification (mape): E2 sf $>$ E1 sf $>$ E1 hsf $>$ E2 hsf
8. baseline hazard parameter identification for both heuristics (mape) : hsf version $>$ sf versione $>$ lifeline $>$ sksurv
9. baseline hazard parameter identification for both heuristics (mse) : sksurv $>$ lifeline $>$ hsf version $>$ sf version
10. beta parameter identification for both heuristics (mse) : sksurv or lifeline $>$ sf version or hsf version
11. beta parameter identification for both heuristics (mape) : sksurv $\sim$ lifeline $\sim$ sf version

Chapter 5

Conclusion

The "competition" between models led to the conclusion that although sksurv is the most accurate, a shallow neural model with only 2 hidden layers is competitive with the lifelines prediction model.

The aim of obtaining the values of β_i and baseline hazard through a gray-box model used as a parameter identification tool did not give the expected results for the constraint of the hazard as an invariant in the computation of the survivability.

The complete explainability of the model, the simplicity that leads to a low execution time and low memory occupation, a good "conservative" accuracy of the future lifetime and a survivability that has a distribution similar to the one generated are the positive results of the research.

Needs to satisfy constrains of sksurv and lifelines in order to compare to a model do not allow to explore cases like all patients died before horizon.

Future implementations can include an identification of the baseline hazard and its parameters because if it is true that the Cox model does not make assumptions on the distribution of the baseline it is still possible to identify it during the creation phase of the output labels of the model without invalidating the initial assumptions.

Chapter 6

Appendix

6.1 Model metrics for single seed

metrics	42	2001	1984	10191	100	221	451	2024
mean	0.06042	0.05457	0.05964	0.04753	0.06832	0.04802	0.06533	0.04840
var	0.00244	0.00258	0.00244	0.00273	0.00232	0.00277	0.00241	0.00269
mse	0.00609	0.00555	0.00600	0.00499	0.00699	0.00508	0.00668	0.00503
mape	12.03	10.89	11.84	9.95	13.84	10.05	13.25	10.03

Table 6.1: Survivability error for sf version of rm model orig baseline E1

metrics	42	2001	1984	10191	100	221	451	2024
mean	0.06349	0.05810	0.06235	0.05019	0.07092	0.05125	0.06877	0.05139
var	0.00390	0.00409	0.00391	0.00434	0.00372	0.00434	0.00379	0.00429
mse	0.00793	0.00747	0.00780	0.00686	0.00875	0.00697	0.00852	0.00694
mape	12.67	11.90	12.44	11.38	13.98	11.44	13.61	11.53

Table 6.2: Survivability error for hsf version of rm model nbinom baseline E1

metrics	42	2001	1984	10191	100	221	451	2024
mean	0.08050	0.07468	0.07926	0.06640	0.08689	0.06998	0.08796	0.06841
var	0.00217	0.00230	0.00218	0.00245	0.00209	0.00243	0.00215	0.00238
mse	0.00865	0.00788	0.00846	0.00686	0.00964	0.00733	0.00989	0.00706
mape	17.410	15.950	17.130	13.760	19.180	14.770	19.670	14.250

Table 6.3: Survivability error for sf version of rm model lognormal baseline E2

metrics	42	2001	1984	10191	100	221	451	2024
mean	0.15014	0.15139	0.15033	0.14946	0.15115	0.15146	0.15406	0.14844
var	0.00862	0.00880	0.00871	0.00870	0.00875	0.00870	0.00877	0.00862
mse	0.03116	0.03172	0.03131	0.03104	0.03159	0.03165	0.03251	0.03066
mape	41.550	41.850	41.640	41.430	41.780	41.810	42.340	41.230

Table 6.4: Survivability error for hsf version of rm model weibull baseline E2

6.2 Survivability error E1

	sf	hsf
count	635520	635520
mean	0.01475	0.01390
std	0.10270	0.11931
min	-0.40409	-0.40544
25%	-0.06824	-0.08549
50%	-0.01073	-0.01847
75%	0.09774	0.11645
max	0.27439	0.29618

Table 6.5: Dataframe.describe() for survivability error of model orig baseline E1

	sf	hsf
count	673440	673440
mean	-0.00170	-0.00322
std	0.18498	0.19012
min	-0.29744	-0.29150
25%	-0.13819	-0.15096
50%	-0.05903	-0.05799
75%	0.10327	0.11445
max	0.74381	0.68320

Table 6.6: Dataframe.describe() for survivability error of model nbinom baseline E1

	sf	hsf
count	361920	361920
mean	0.05653	0.05956
std	0.05110	0.06412
min	-0.04939	-0.05802
25%	0.02483	0.01976
50%	0.04718	0.04557
75%	0.07354	0.07721
max	0.41009	0.45686

Table 6.7: Dataframe.describe() for survivability error of model lognormal baseline E1

	sf	hsf
count	358560	358560
mean	-0.00923	-0.00876
std	0.12765	0.13106
min	-0.37842	-0.39286
25%	-0.08593	-0.08023
50%	0.02339	0.02694
75%	0.07189	0.07576
max	0.46546	0.41874

Table 6.8: Dataframe.describe() for survivability error of model weibull baseline E1

6.3 Survivability error E2

	sf	hsf
count	635520	635520
mean	0.03843	0.09065
std	0.09722	0.13597
min	-0.40398	-0.40544
25%	-0.03510	-0.00290
50%	0.00119	0.04122
75%	0.11532	0.18223
max	0.30428	0.55824

Table 6.9: Dataframe.describe() for survivability error of model orig baseline E2

	sf	hsf
count	673440	673440
mean	0.04461	0.11854
std	0.15460	0.16052
min	-0.20949	-0.27342
25%	-0.05127	0.00340
50%	-0.01267	0.07299
75%	0.10631	0.23223
max	0.80464	0.78152

Table 6.10: Dataframe.describe() for survivability error of model nbinom baseline E2

	sf	hsf
count	361920	361920
mean	0.07677	0.15081
std	0.04830	0.09336
min	-0.02452	-0.06437
25%	0.04672	0.07763
50%	0.06881	0.15961
75%	0.09514	0.21192
max	0.42522	0.56384

Table 6.11: Dataframe.describe() for survivability error of model lognormal baseline E2

	sf	hsf
count	358560	358560
mean	-0.02623	-0.02651
std	0.15885	0.15384
min	-0.49780	-0.47972
25%	-0.10701	-0.11204
50%	0.02560	0.02373
75%	0.07447	0.07207
max	0.46384	0.46404

Table 6.12: Dataframe.describe() for survivability error of model weibull baseline E2

Bibliography

[1] From lesson material (slide) of Diagnosis and Control course of Prof. Andrea Tilli, Unibo.

[2] Austin, P. C., Lee, D. S., and Fine, J. P. Introduction to the analysis of survival data in the presence of competing risks. *Circulation 133*, 6 (2016), 601–609.

[3] AWS. Interpretability versus interpretability.

[4] bmj. Suvival analisis.

[5] Brembilla, A., Olland, A., Puyraveau, M., Massard, G., Mauny, F., and Falcoz, P.-E. Use of the cox regression analysis in thoracic surgical research. *Journal of thoracic disease 10*, 6 (2018), 3891.

[6] Cleves, M. *An introduction to survival analysis using Stata*. Stata press, 2008.

[7] Cox, D. R. Regression models and life-tables. *Journal of the Royal Statistical Society 34*, 2 (January 1972), 187–202.

[8] Crevier, D. *AI: The Tumultuous Search for Artificial Intelligence 1993 New York*. NY Basic Books, 1993.

[9] Danacica, D.-E., and Babucea, A.-G. Using survival analysis in economics. *survival 11* (2010), 15.

[10] deepai. Feedforward neural network.

[11] Gensheimer, M. F., and Narasimhan, B. A scalable discrete-time survival model for neural networks. *PeerJ 7* (2019), e6257.

[12] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM 63*, 11 (2020), 139–144.

[13] google. Generative neural network.

[14] Graves, A., and Schmidhuber, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks 18*, 5-6 (2005), 602–610.

[15] Hebb, D. O. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

[16] Holmer, O., Frisk, E., and Krysander, M. Neural network-based piecewise survival models. *arXiv preprint arXiv:2403.18664* (2024).

[17] ibm. Recurrent neural network.

[18] Jung, E.-Y., Baek, C., and Lee, J.-D. Product survival analysis for the app store. *Marketing Letters 23* (2012), 929–941.

[19] Khosravi, B., Pourahmad, S., Bahreini, A., Nikeghbalian, S., and Mehrdad, G. Five years survival of patients after liver transplantation and its effective factors by neural network and cox poroportional hazard regression models. *Hepatitis monthly 15*, 9 (2015).

[20] Kleinbaum, D. G., and Klein, M. *Survival analysis a self-learning text*. Springer, 1996.

[21] Konstantinou, M., Biedermann, S., and Kimber, A. C. Optimal designs for full and partial likelihood information—with application to survival models. *Journal of Statistical Planning and Inference 165* (2015), 27–37.

[22] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems 25* (2012).

[23] Kvamme, H., and Borgan, Ø. Continuous and discrete-time survival prediction with neural networks. *arXiv preprint arXiv:1910.06724* (2019).

[24] Kvamme, H., Borgan, Ø., and Scheel, I. Time-to-event prediction with neural networks and cox regression. *Journal of machine learning research 20*, 129 (2019), 1–30.

[25] Li, S. Survival analysis. *Marketing Research 7*, 4 (1995), 16.

[26] mayoclinic. Liver transplant.

[27] medium. Convolutional neural network.

[28] From lesson material (slide) of Intelligent Systems course of Prof. Michela Milano, Unibo.

[29] Nguyen, V. Q., and Gillen, D. L. Censoring-robust estimation in observational survival studies: Assessing the relative effectiveness of vascular access type on patency among end-stage renal disease patients. *Statistics in biosciences 9* (2017), 406–430.

[30] Rehor, J., and Havlena, V. Grey-box model identification–control relevant approach. *IFAC Proceedings Volumes 43*, 10 (2010), 117–122.

[31] Rojas, R., and Rojas, R. The backpropagation algorithm. *Neural networks: a systematic introduction* (1996), 149–182.

[32] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature 323*, 6088 (1986), 533–536.

[33] Stuart J. Russell, P. N. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2020.

[34] towardsdatascience. Convolutional neural network.

[35] Wang, P., Li, Y., and Reddy, C. K. Machine learning for survival analysis: A survey. *ACM Computing Surveys (CSUR) 51*, 6 (2019), 1–36.

[36] Wikipedia. Survival analysis.

[37] Wikipedia. Survival analysis map.

[38] Yang, Z., Eddy, D., Krishnamurty, S., Grosse, I., Denno, P., Lu, Y., and Witherell, P. Investigating grey-box modeling for predictive analytics in smart manufacturing. In *International design engineering technical conferences and computers and information in engineering conference* (2017), vol. 58134, American Society of Mechanical Engineers, p. V02BT03A024.

Acknowledgements

I would like to thank

- all the teachers of the courses I attended for the passion they transmitted to me and in particular to Dr Federico Balbo and Prof Michele Lombardi for providing guidance and feedback throughout this project.
- all the friends who encouraged me.

Dedication

Dedico questa tesi a mia moglie e mio figlio per l'amore e la pazienza che mi hanno dedicato durante questi studi , un'attenzione che è stata fondamentale per completare questo percorso impegnativo, e al mio amico Nicola Santini che ci ha lasciato troppo presto.