Insights into Company Survival in Italy: A Comparative Study of Cox Proportional Hazards and DeepSurv Models

by

Panayiota Ismini Harken Alexiou

Supervisor: Caroni Chrysseis

Submitted to
School of Applied Mathematical and Physical Sciences
of the
National Technical University of Athens

in partial fulfillment of the requirements
for the degree of

MSc in Mathematical Modeling in Modern Technologies and Finance

Athens, Greece
October, 2023

Insights into Company Survival in Italy: A Comparative Study of Cox Proportional Hazards and DeepSurv Models

by

Panayiota Ismini Harken Alexiou

Supervisor: Caroni Chrysseis

Submitted to
School of Applied Mathematical and Physical Sciences
of the
National Technical University of Athens
in partial fulfillment of the requirements
for the degree of
MSc in Mathematical Modeling in Modern Technologies and Finance

Examination Board

C. Chrysseis	**V. Papanikolaou**	**K. Pavlopoulou**
Emeritus Professor	**Emeritus Professor**	**Professor**

Athens, Greece
October, 2023

ABSTRACT

This diploma thesis presents a comprehensive analysis of companies in Italy that have closed due to bankruptcy and other reasons. The study utilizes survival analysis techniques to examine the factors influencing the survival time of these companies. Two primary models, namely the Cox Proportional Hazards (CPH) model and the DeepSurv model, are employed to analyze the data.

The first part of the analysis focuses on the Cox Proportional Hazards model, a well-established approach in survival analysis. The CPH model is utilized to identify and understand the impact of various risk factors on the survival time of companies in the dataset.

In the second part, the study employs the DeepSurv model, which leverages deep neural network technology for survival analysis. Two variations of the DeepSurv model are explored: one incorporating hyperparameter optimization (HPO) and the other featuring a feature selection process. All models are evaluated using the C-Index, a widely used metric that measures the concordance between predicted and observed survival times.

The results indicate that the DeepSurv model with feature selection achieved the highest C-Index value among all the models tested. This finding suggests that the inclusion of feature selection significantly improved the model's predictive performance for survival analysis tasks. Furthermore, the DeepSurv model with HPO demonstrated a slightly better performance than the traditional Cox Proportional Hazards model, indicating the potential of deep learning techniques in capturing complex patterns within the data.

In conclusion, this thesis provides valuable insights into the factors influencing the closure of companies in Italy due to bankruptcy and other reasons. The comparison of the Cox Proportional Hazards model and the DeepSurv model highlights the advantages of utilizing deep learning methods for survival analysis. The findings underscore the importance of feature selection in enhancing the predictive capabilities of the DeepSurv model, making it a promising approach for future survival analysis studies.

Keywords: Cox Proportional Hazards, DeepSurv, Survival Analysis.

DECLARATION

I hereby certify that this report constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the report describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Panayiota Ismini Harken Alexiou

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Professor Chrysseis Caroni, my esteemed advisor, for her unwavering support, guidance, and valuable insights throughout the course of my diploma thesis. Her expertise and encouragement have been instrumental in shaping this research and enriching my understanding of the subject.

I am also deeply indebted to my family, my beloved mother and sister, for their constant love, encouragement, and understanding. Their unwavering belief in me has been a driving force behind my academic journey, and I am immensely grateful for their sacrifices and encouragement.

Furthermore, I would like to acknowledge the MSc in Mathematical Modeling in Modern Technologies and Finance program at the School of Applied Mathematical and Physical Sciences, National Technical University of Athens. This program has provided me with a wealth of knowledge and skills that have been invaluable in the completion of my diploma thesis.

I am humbled and privileged to have had the opportunity to pursue my Master's degree and the experience has been truly rewarding. I am grateful to all those who have supported me along the way.

Contents

List of Tables **7**

List of Figures **8**

1 Introduction **9**
1.1 Background and Motivation 9
1.1.1 Survival Analysis for Bankruptcy Prediction 10
1.1.2 Artificial Neural Network Approach to Survival Analysis 12

2 Mathematical background **14**
2.1 Survival Analysis 14
2.1.1 Cox Proportional Hazards Model 14
2.1.2 Estimation of Cox Model's Parameters 15
2.1.3 Equal Stopping Times 16
2.1.4 Hypothesis Testing 17
2.1.5 Extension of Cox's Model 17
2.1.6 Hypothesis Testing of the Proportional Hazards Assumption 19
2.1.7 Cox's Model Residuals 21
2.2 Deep Learning 23
2.2.1 Artificial Neural Networks 23
2.2.2 Multilayer Neural Networks 25
2.2.3 Training a Neural Network with Backpropagation 27
2.2.4 Hyperparameter options 29
2.2.5 The Problem of Overfitting 35
2.2.6 Regularization 36
2.2.7 Neural Network Architecture 37
2.2.8 Epochs and Batch Size 38
2.2.9 The Vanishing and Exploding Gradient Problems 39
2.2.10 Automatic hyperparameter tuning methods 39
2.2.11 Feature Selection 42
2.3 Overview of Deepsurv 44
2.4 Performance metrics 46
2.4.1 Concordance Index 46
2.4.2 Integrated Brier Score 46

3 Case Study and Analysis **48**
3.1 Dataset Description 48
3.2 Data Preprocessing 50
3.2.1 Normalization and Skewness Reduction 50
3.2.2 Multicollinearity Detection 51
3.3 Cox Proportional Hazards Model Application 54
3.4 DeepSurv Application 58
3.5 Comparison of 3 Models 64

4 Discussion **67**

Definitions and Abbreviations **69**

Appendix **75**

List of Tables

3.1 Cox Proportional Hazards Initial Model Results 56
3.2 Results of Proportional Hazards (PH) Assumption Tests 57
3.3 DeepSurv models with optimal combination of hyperparameters 59
3.4 Results of Feature Selection (Part 1) 62
3.5 Results of Feature Selection (Part 2) 63
3.6 Model Comparison Aggregating Results 64

List of Figures

2.1 The basic architecture of the perception (Aggarwal, 2018) 23
2.2 Multilayer neural networks (a) with no bias neurons (b) with bias neurons (Aggarwal, 2018) 26
2.3 Various activation functions (Aggarwal, 2018) 32
2.4 Effects of different learning rates. θ are the weights and $J(\theta)$ is the loss function (Jordan, 2023) 32
2.5 Grid and Random Layout (Lippert et al., 2022) 41

3.1 Training data before transformation 52
3.2 Training data after transformation 52
3.3 Pearson's correlations between numerical covariates 53
3.4 An illustration of nested resampling for hyperparameter optimization . . . 60
3.5 Performances of the 3 Models 65

Chapter 1

Introduction

1.1 Background and Motivation

Bankruptcy can be defined as the lack of resources to repay the obligations of a company as they come due (Boardman et al., 1981). It is a financial state where a company is unable to pay its debts and obligations. Bankruptcy is a critical issue for companies, with severe consequences for firm owners, partners, society, and the country's economy at large (Alaka et al., 2018). Moreover, the consequences of bankruptcy encompass substantial effects on a company's financial stability, reputation, and stakeholders' concerns. Accurate evaluations of bankruptcy likelihood are vital for banks and financial institutions due to their potential to diminish overall economic productivity. Also, analyzing bankruptcy enables companies and stakeholders to forecast the likelihood of a company's collapse and implement effective measures for viable recuperation. Therefore, there has been significant emphasis on the study of bankruptcy in academic research and practical applications, aiming to acquire the capacity to predict and prevent bankruptcy, which is crucial for ensuring a company's survival and advancement (Alaka et al., 2018).

In a review conducted by Adnan Aziz and Dar (2006) on bankruptcy prediction models, various classification models were listed, which can be categorized as either statistical or artificial intelligence (AI)-based , including discriminant analysis, logistic regression, case-based reasoning, neural networks, and rough sets. The authors highlighted that multiple discriminant analysis and logistic regression were the most commonly used models, with over 50% of the reviewed studies dedicated to them. Multiple discriminant analysis (MDA), which was initially introduced by Altman (1968) through the development of the Z model based on the recommendation of Beaver (1966). However, subsequent accounting and finance literature studies simply adopted MDA without considering its underlying assumptions, leading to inappropriate applications and limited generalizability of the developed models (Alaka et al., 2018). In some cases, researchers combined qualitative managerial variables with quantitative variables by using the A-score alongside the Z-score (MDA), despite logistic regression (LR) being more suitable for handling both types of variables separately (Alaka et al., 2018).

Among AI tools, Artificial Neural Networks (ANNs or NNs) are commonly employed for bankruptcy prediction (Alaka et al., 2018). Furthermore, Fletcher and Goss (1993) developed an ANN prediction model using relatively small sample size, despite the known need for large samples for optimal ANN performance (Alaka et al., 2018). Classification trees have been widely used in bankruptcy prediction, along with more recent applications of artificial neural networks (Wilson and Sharda, 1994, Charalambous et al., 2000, Perez, 2006). Shumway (2001) introduced a hazard model to incorporate a dynamic element into bankruptcy prediction. However, Zelenkov (2020) highlighted that survival analysis, which is commonly employed in the medical and technical sciences to analyze the timing of events, is relatively infrequently used in the context of predicting financial failure.

In this thesis, a comprehensive predictive analysis of bankruptcy in firms was undertaken, utilizing and comparing two distinct methods. The first method employed was the Cox proportional hazards model, a statistical technique widely used in survival analysis. This model allows for the examination of time-to-event data, making it suitable for bankruptcy prediction, where the event of interest is the occurrence of bankruptcy. The Cox proportional hazards model takes into account various predictor variables and estimates the hazard ratio, which represents the relative risk of bankruptcy.

In addition to the Cox proportional hazards model, a deep neural network model called DeepSurv was also utilized. Deepsurv is specifically designed for survival analysis, incorporating the advantages of neural networks in capturing complex relationships within the data. It enables the estimation of survival probabilities and the identification of significant risk factors contributing to bankruptcy.

1.1.1 Survival Analysis for Bankruptcy Prediction

The application of survival analysis, originally developed in biomedical sciences, extends to the prediction of bankruptcy. This statistical technique utilizes the survival time or hazard rate as the dependent variable, enabling the modeling of failure processes that may deviate from a steady-state path. An essential assumption in survival analysis is that both failed and non-failed firms are sampled from the same population. Non-failed firms represent cases where bankruptcy has not yet occurred. Consequently, survival analysis serves as another valuable technique employed in the prediction of bankruptcy.

Lane et al. (1986) applied survival analysis, specifically the Cox model, to predict bank failure. Their study found that the overall accuracy of the Cox model was similar to that of discriminant analysis but with a lower type I error. Keasey et al. (1998) also recommended survival analysis for analyzing company failure data.

Luoma (1991) conducted a study using survival analysis to predict business failure. They analyzed a sample of 36 failed companies (24 from industrial sectors and 12 from retailing firms), each paired with a non-failed company from the same industry and of

similar size. The results were compared to models developed using discriminant analysis and logistic regression. The correct classification rates were 61.8%, 70.6%, and 72.1% for survival analysis, discriminant analysis, and logistic regression, respectively. The authors attributed the lower accuracy of the survival analysis model to the different failure processes observed in the data.

Parker et al. (2002) conducted survival analysis on a sample of 176 distressed US firms that experienced a decrease in cash flows from positive to negative over successive fiscal years from 1988 to 1996. Their results indicated that corporate governance attributes influenced the likelihood of survival. They also found that replacing the CEO with an outsider was associated with a twofold increase in the likelihood of bankruptcy.

In a study by Partington and Kim (2008) a time-dependent Cox regression model with dynamic variables was employed to estimate survival probabilities and make dynamic financial distress predictions for Australian firms listed on the Australian Securities Exchange from 1989 to 2006. The results indicated that firms with higher book leverage, lower cash flow generating ability, and lower market leverage were more likely to experience financial distress. The baseline hazard was found to have a significant impact on the estimated hazard rate relative to the effects of covariates. Notably, the accuracy of financial distress predictions improved as the time horizon lengthened, reaching 81% accuracy in-sample for firms that had been in the risk set for 14 years.

Pereira (2014) utilized the Cox proportional hazards model to examine the characteristics of bank failure. The study emphasized the model's advantage in providing additional information by assessing the likelihood of a company's survival beyond a specific time period. However, the model's accuracy depended on data quality and assumed proportional risks, which may not always hold. The study concluded that the method showed promise for developing bankruptcy forecasting models, suggesting the incorporation of a larger sample size and qualitative variables.

Lee (2014) studied financial distress in companies listed on the Taiwan Stock Exchange between 2003 and 2009. The research employed survival analysis, specifically the Cox Proportional Hazard Model, to identify key indicators of business bankruptcy in Taiwan. The study found that a few selected financial ratios, including Profitability, Leverage, Efficiency, and Valuation ratios, were sufficient for anticipating potential business bankruptcy. The proposed prediction model achieved an overall classification accuracy of 87.93%.

Pierri and Caroni (2017) conducted a study examining the risk of failure among Small Business Enterprises (SBEs) in Umbria, Italy, during the economic crisis period of 2008-2013. They employed logistic regression and survival analysis techniques based on hazard models, utilizing a dataset consisting of 11,248 businesses. To assess the models' performance, ROC curves were employed to derive comprehensive performance measures. The inclusion of lagged covariates improved the performance of the models. Additionally, logistic regression models were constructed using the same dataset for comparative purposes. Notably, both the logistic regression and survival analysis models yielded similar outcomes in their predictions of business failure.

Gemar et al. (2019) examined the survival of resort hotels in Spain and investigated

the factors influencing their closure. Using Cox's semi-parametric proportional hazards regression, the study identified that hotel size, location, executive management, and the business cycle were significant factors affecting the closure of resort hotels. Interestingly, the type of hotel or its financial structure did not have a significant impact on survival rates. These findings contribute valuable insights to the understanding of resort hotel survival and highlight the importance of considering specific factors such as size, location, management, and the business cycle when assessing the risk of closure in this industry.

Ayadi et al. (2020) examined the survival prospects of reorganized firms in France. Their study identified factors that influenced the time to failure for these firms. The findings revealed that firm size, profitability, liquidity, industry profitability, and inflation rate positively affected survival, while leverage and variation in short-term interest rates had a negative impact. The study also noted similarities between the failure processes of reorganized firms and new firms.

Zelenkov (2020) conducted a comparative analysis of different survival analysis models, including the Cox Proportional Hazard Model (CPH), Aalen's Additive Regression (AAR), Weibull Accelerated Failure Time Models (WAF), Random Survival Forest (RSF), and Multi-task Neural Network (MNN). The study utilized a dataset of 2,457 Russian companies, out of which 280 went bankrupt within a year after reporting for 2014. The results showed that the machine learning model RSF achieved the highest concordance index, with a mean of 0.840 and a standard deviation of 0.029. The CPH model had a slightly lower concordance index, with a mean of 0.806 and the same standard deviation. The other models performed relatively well but were outperformed by RSF and CPH. In summary, the study demonstrated that the machine learning model RSF had superior predictive performance in bankruptcy prediction compared to other models, while the CPH model also showed good performance.

1.1.2 Artificial Neural Network Approach to Survival Analysis

Rapid advancements in data collection and analysis have stimulated researchers to explore the potential of deep learning in the field of survival analysis. Faraggi and Simon (1995) proposed replacing the linear predictor of the Cox proportional hazards (CPH) model with a single hidden-layer neural network. However, their neural network extension, applied to survival analysis from clinical data in low dimensions, did not yield reliable improvements in terms of the concordance index. In contrast, Katzman et al. (2018) introduced DeepSurv, which combines a deep neural network (DNN) with the CPH model to predict the risk of event occurrence for patients and provide personalized treatment recommendations. DeepSurv employs modern deep learning techniques such as weight decay, batch normalization, and dropout.

Another approach was proposed by Kvamme et al. (2019), who extended the Cox model by parametrizing the relative risk function with neural networks and modeling interactions between covariates and time. However, their method is primarily suitable for scenarios where the number of covariates is smaller than the number of patients.

Building on DeepSurv, Zhu et al. (2016) introduced DeepConSurv, which replaces the multi-layer perceptron (MLP) component with a convolutional neural network. They applied this method to pathological images of lung cancer and whole-slide histopathological images.

To overcome challenges such as memory constraints when dealing with survival data and images, Tarkhan et al. (2021). proposed a modified version of the Cox proportional hazards model that is amenable to stochastic gradient descent (SGD).

In the context of predicting Alzheimer's disease, Pölsterl et al. (2020) proposed a wide and deep neural network that serves as a clinical regression model for survival analysis.

Ranganath et al. (2016) introduced the Deep Survival Analysis (DSA) method, which utilizes a deep hierarchical Bayesian model. However, their approach does not handle censored events.

Giunchiglia et al. (2018) developed RNN-SURV, a recurrent neural network model capable of leveraging censored data to compute both the risk score and survival function for each patient. The network takes patient features and time step identifiers as input at each time step, creates embeddings, and outputs the survival function value for that time step. The risk scores are computed by linearly combining the values of the survival function.

Hao et al. (2019) proposed Cox-PASNet, a biologically interpretable pathway-based sparse deep neural network. This model integrates high-dimensional gene expression data and clinical data within a simple neural network architecture for survival analysis.

Ching et al. (2018) presented the Cox-nnet algorithm, which utilizes a two-layer neural network. The hidden layer performs dimension reduction of input covariates, while the output layer conducts Cox regression based on the activation levels of the hidden layer.

Ren et al. (2019) employed deep recurrent neural networks to capture sequential patterns of features over time in survival analysis.

To provide a flexible and general framework for deep survival analysis, Zhong et al. (2021) introduced the Deep Extended Hazard (DeepEH) model. This model encompasses the conventional Cox proportional hazards and accelerated failure time models as special cases, making it capable of subsuming popular models such as DeepSurv and DeepAFT.

Chapter 2

Mathematical background

2.1 Survival Analysis

Survival analysis is a statistical method used to analyze time-to-event data, where the event of interest could be death, failure or any other outcome. The survival function is denoted as

$$S(t) = P[T > t] = \int_t^\infty f(u)du$$

which represents the probability that the duration of time, until an event occurs, is greater than t. The probability density function of T is denoted as

$$f(t) = -\frac{d}{dt}S(t)$$

The fundamental concept in survival analysis is the hazard function, denoted as $h(t)$. The hazard function is defined as the probability of an event ending or stopping within a small time interval $(t, t + \delta t)$, given that the event has not ended or stopped up until time t (Caroni, 2009). Mathematically, it is expressed as:

$$h(t) = \lim_{\delta t \to 0} \frac{S(t) - S(t + \delta t)/S(t)}{\delta t} = \frac{f(t)}{S(t)}$$

2.1.1 Cox Proportional Hazards Model

Survival analysis presents a significant hurdle as it frequently involves the presence of censored data (represented by the symbol C), indicating that complete event time information for all subjects in the study is not available. To overcome this challenge, it is necessary to employ suitable statistical techniques that can accommodate such censored data. The semi-parametrical Cox proportional hazards model is a method, that allows the examination of the relationship between covariates and the hazard function. Cox proportional hazards model assumes that the hazard function for an individual is proportional to a baseline hazard function, denoted as $h_0(t)$, and a set of

covariates X, such that:

$$h(t; X) = h_0(t) \cdot \exp(\boldsymbol{\beta}_1 X_1 + \boldsymbol{\beta}_2 X_2 + \cdots + \boldsymbol{\beta}_p X_p)$$

where $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_p$ are the regression coefficients for the covariates $X_1, X_2, \ldots, X_p$, respectively.

Given the equation

$$H(t) = \int_0^t h(u)du$$

we can express the cumulative hazard function as

$$H(t; X) = H_0(t) \exp(\boldsymbol{\beta}_1 X_1 + \boldsymbol{\beta}_2 X_2 + \cdots + \boldsymbol{\beta}_p X_p)$$

Consequently, we can use this to derive the survival function $S(t)$ as

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

By substituting $H(t; X)$ into the previous equation, we can express the survival function as

$$S(t; x) = S_0(t)^{\exp(\boldsymbol{\beta}_1 X_1 + \boldsymbol{\beta}_2 X_2 + \cdots + \boldsymbol{\beta}_p X_p)}$$

(Caroni, 2009)

2.1.2 Estimation of Cox Model's Parameters

Suppose that k subjects end their lives at times

$$t_1 < t_2 < \cdots < t_k$$

where subject j has covariates x_j and experiences the event of interest at time t_j. Let $\mathbf{R}_j$ denote the total number of subjects at risk (i.e., not yet experiencing the event of interest) just before time t_j. The probability of subject j experiencing the event of interest given that any subject in the risk set experiences the event is given by

$$\frac{h(t_{(j)}; x_j)}{\sum_{i \in \mathbf{R}_j} h(t_{(j)}; x_i)}$$

which can be expressed as

$$\frac{\exp(\boldsymbol{\beta}^\top x_j)}{\sum_{i \in \mathbf{R}} \exp(\boldsymbol{\beta}^\top x_i)}$$

where $\boldsymbol{\beta}$ is a vector of regression coefficients.

We can estimate the maximum likelihood of the parameter vector $\boldsymbol{\beta}$, denoted as $\hat{\boldsymbol{\beta}}$, from

the partial likelihood function of the data given by

$$L(\boldsymbol{\beta}) = \prod_{j=1}^{k} \frac{\exp(\boldsymbol{\beta}^\top x_j)}{\sum_{i \in \mathbf{R}_j} \exp(\boldsymbol{\beta}^\top x_i)}$$

This function provides the likelihood of the observed data given the parameter values and allows us to estimate the values of $\boldsymbol{\beta}$ that maximize this likelihood.
The log-likelihood function is expressed as

$$l(\boldsymbol{\beta}) = \sum_{j=1}^{k} \boldsymbol{\beta}' \mathbf{x}_j - \sum_{j=1}^{k} \ln \left\{ \sum_{i \in R_j} \exp\{\boldsymbol{\beta}' \mathbf{x}_i\} \right\},$$

where $\boldsymbol{\beta}$ is a p-dimensional parameter vector, $\mathbf{x}_j$ is a vector of covariate values for the jth risk set, R_j is the set of indices of individuals in the jth risk set, and $\exp\{\boldsymbol{\beta}' \mathbf{x}_i\}$ is the hazard ratio for individual i.
The first derivatives of the likelihood function with respect to the rth component of $\boldsymbol{\beta}$ are given by

$$\frac{\partial l}{\partial \boldsymbol{\beta}_r} = \sum_{j=1}^{k} x_{jr} - \sum_{j=1}^{k} \frac{\sum_{i \in R_j} x_{ir} \exp\{\boldsymbol{\beta}' \mathbf{x}_i\}}{\sum_{i \in R_j} \exp\{\boldsymbol{\beta}' \mathbf{x}_i\}}.$$

The system of equations

$$\frac{\partial l}{\partial \boldsymbol{\beta}_r} = 0$$

for

$$r = 1, \ldots, p$$

is then solved to obtain the estimates $\hat{\boldsymbol{\beta}}$ of the parameters. Note that the linear predictor $\boldsymbol{\beta}' \mathbf{x}$ does not include any constant term because it will be part of the baseline hazard function $h_0(t)$ (Caroni, 2009).

2.1.3 Equal Stopping Times

If the stopping times are equal, the likelihood function needs to be adjusted accordingly. Firstly, let's examine the case where time is continuous. If there are multiple failures, denoted by $d_j > 1$, that coincide with time t_j, theoretically they would occur at different times if the measurements were more accurate. This implies that there is a specific ordering between the d_j failures. However, it is not possible to determine which of the possible d_j series has occurred. As a result, the function of partial likelihood should include all of them, making it much more complicated. For this reason, Breslow's simple estimation is commonly used. It replaces the term

$$\frac{e^{\boldsymbol{\beta}' \mathbf{x}_j}}{\sum_{i \in R_j} e^{\boldsymbol{\beta}' \mathbf{x}_i}}$$

in the partial likelihood when $d_j = 1$, with the term

$$\frac{e^{\boldsymbol{\beta}'\mathbf{z}_j}}{\{\sum_{i \in R_j} e^{\boldsymbol{\beta}'\mathbf{x}_i}\}^{d_j}}$$

where

$$\mathbf{z}_j = \sum_{k=1}^{d_j} \mathbf{x}_k$$

and $\mathbf{x}_k$ is the vector of covariates for the kth unit with failure at time $t_j, k = 1, ..., d_j$. This estimation is accurate when the quantity d_j/n_j is small.

When d_j/n_j is not small, Cox estimation is used, assuming that the data are measured in discrete instead of continuous scale. The method for these data is as follows: Assuming that at time t_j, we have d_j failures, the probability of observing a total of u units is given by

$$P(u) \propto e^{\beta' z_u}$$

As before, z_u is the sum of covariates x for the units in the total u. Then, the conditional probability of observing u^* with a stop is given by

$$P(u^*|d_j) = e^{\beta' z_j} / \sum e^{\beta' z_u}$$

where the denominator consists of the sum of vectors $\begin{pmatrix} \mathrm{n}_j \\ \mathrm{d}_j \end{pmatrix}$ and n_j is the number of units at risk right before time t_j.

In summary, Breslow's simple estimation is used when the number of failures is small compared to the number of units at risk. Cox estimation is used when the number of failures is not small, assuming the data are measured in discrete scale (Caroni, 2009).

2.1.4 Hypothesis Testing

Hypotheses of the form $\boldsymbol{\beta}_i = 0$ can be tested using the likelihood ratio test. This involves fitting the model with and without the covariate x_i. The model without x_i imposes the restriction that $\boldsymbol{\beta}_i = 0$. Let the maximum values of the logarithm of partial likelihood be denoted as $\hat{l}_1$ and $\hat{l}_0$ with and without x_i respectively. The test statistic $-2(\hat{l}_0 - \hat{l}_1)$ is then compared to the χ^2 distribution with one degree of freedom to determine the p-value for the test. This p-value can be used to assess the statistical significance of the hypothesis (Caroni, 2009).

2.1.5 Extension of Cox's Model

Stratified Cox Model

An important extension of Cox's basic model is the ability to perform stratified analysis, also known as the stratified Cox model. This model is usually applied when the hazard functions of two or more categories are not proportional to each other. This occurs when the risk of the event of interest varies across subgroups defined by some categorical variable, such as gender or age group. By stratifying the data, we can model the baseline hazard function separately for each subgroup while allowing the effect of the covariates to remain the same across subgroups. This approach allows for more flexibility in modeling the relationship between the covariates and the event of interest (Caroni, 2009). For every stratum m, the logarithm of the partial likelihood is given as:

$$l_m(\beta) = \sum_{j=1}^{k_m} \beta' x_{mj} - \sum_{j=1}^{k_m} \ln \left\{ \sum_{i \in R_{mj}} e^{\beta' x_{mi}} \right\}$$

where x_{mj}, R_{mj}, and k_m represent the data of stratum m in the function l_m. In total, for the strata $m = 1, \ldots, s$, we have:

$$l(\beta) = \sum_{m=1}^{s} l_m(\beta) = \sum_{m=1}^{s} \sum_{j=1}^{k_m} \beta' x_{mj} - \sum_{m=1}^{s} \sum_{j=1}^{k_m} \ln \left\{ \sum_{i \in R_{mj}} e^{\beta' x_{mi}} \right\}$$

Time-Dependent Variables

The Cox model assumes that the hazard rate is proportional to some set of covariates, which may include time-dependent variables. Time-dependent covariates can be either internal or external, depending on whether they are measured within the study or outside of it.

Internal time-dependent covariates refer to variables that change over time within the study, such as treatment status, disease progression, or changes in biomarker levels. These variables may affect the hazard rate differently at different times during the study, and thus including them in the Cox model can improve the accuracy of the analysis.

External time-dependent covariates, on the other hand, refer to variables that change over time outside of the study, such as weather patterns, economic indicators, or political events. These variables may also affect the hazard rate, and including them in the Cox model can help account for external factors that may influence the outcome of interest.

Incorporating time-dependent covariates in the Cox model requires careful modeling and interpretation, as the hazard rate is assumed to be proportional over time. Thus, changes in the covariates may affect the hazard rate differently at different time points. Furthermore, the Cox model assumes that the hazard ratio associated with a covariate is constant over time, which may not always hold true in practice.

Inclusion of time-dependent covariates in the Cox model is a useful technique, but it can be challenging when the covariates are only available at specific time points rather

than being continuously evaluated. In such cases, a common approach is to use the last available value of the covariate before the stop time.

To implement this approach, we replace every occurrence of the covariate x in the basic Cox model partial likelihood function with $x(t_{(j)})$, where $t_{(j)}$ is the stop time for the jth event. The resulting partial likelihood function can be expressed as:

$$l(\boldsymbol{\beta}) = \sum_{j=1}^{k} \boldsymbol{\beta}' x_j(t_{(j)}) - \sum_{j=1}^{k} ln \sum_{i \in Rj} e^{\boldsymbol{\beta}' x_i(tj)}$$

However, if the covariate values are only available at intervals (e.g., in months), then $x(t_{(j)})$ may not be available for every event. In such cases, we can use the last available value of the covariate before the stop time as a proxy for $x(t_{(j)})$. This approach assumes that the covariate value does not change significantly between the last available value and the stop time.

It is important to note that this approach may introduce bias if the covariate value changes rapidly over time or if there are missing values in the data. Therefore, researchers should carefully consider the nature of the covariate, the study design, and the available data when using this approach to include time-dependent covariates in the Cox model.

Despite these challenges, the use of Cox models with time-dependent covariates is an important extension that can help improve the accuracy and relevance of survival analyses. With careful consideration of the study design, the nature of the covariates, and the assumptions of the model, researchers can use this method to gain insights into the complex relationships between covariates and survival outcomes (Caroni, 2009).

2.1.6 Hypothesis Testing of the Proportional Hazards Assumption

In the Cox model, the proportional hazards assumption states that the hazard ratio between any two individuals is constant over time, i.e., the hazard ratio does not depend on time. Mathematically, this can be expressed as:

$$\frac{h_l(t)}{h_j(t)} = \frac{h_0(t)e^{\boldsymbol{\beta}' x_l}}{h_0(t)e^{\boldsymbol{\beta}' x_j}} = e^{\boldsymbol{\beta}'(x_l - x_j)}$$

where $h_l(t)$ and $h_j(t)$ are the hazard functions for individuals *l* and *j*, respectively, and $h_0(t)$ is the baseline hazard function at time t. The term $\boldsymbol{\beta}' x_l$ and $\boldsymbol{\beta}' x_j$ represent the linear predictors for individuals l and j, respectively, based on their covariate values.

The hazard ratio is the ratio of the hazard functions for two individuals, and it is assumed to be proportional to the exponential of the difference in their linear predictors, $\boldsymbol{\beta}'(x_l - x_j)$. This means that the hazard ratio between two individuals is independent of time, as long as their covariate values remain the same.

Hypothesis testing of the proportional hazards assumption involves testing whether the hazard ratio is constant over time or not.

One way to test the proportional hazards assumption in the Cox model is to create an

interaction variable between each covariate x_i and time. This can be done by defining a new variable $z = x_i t$, which expresses the interaction between x_i and time. Next, the Cox model is fitted with z included as one of the covariates. The null hypothesis $H_0 : \boldsymbol{\beta}_z = 0$ is then tested, where $\boldsymbol{\beta}_z$ is the coefficient of z in the Cox model. If we fail to reject H_0, it means that the effect of the covariate x_i on the hazard function is expressed by the coefficient $\boldsymbol{\beta}_i x_i$, which is independent of time and can be interpreted as a constant hazard ratio. This supports the proportional hazards assumption.

Another way is graphical testing using Breslow's estimator of the stratified Cox model. The estimation of the function

$$S(t; x) = \{S_0(t)\}^{e^{\boldsymbol{\beta}' x}}$$

requires the estimation of not only $\boldsymbol{\beta}$ but also $S_0(t)$. The function $S_0(t)$ is estimated using the non-parametric Breslow estimator given by

$$\hat{S}_0(t) = e^{-\hat{H}_0(t)}$$

where

$$\hat{H}_0(t) = \sum_{t(j) \leq t} \left(\frac{d_j}{\sum_{i \in R_j} e^{\hat{\boldsymbol{\beta}}' x_i}} \right)$$

is the Breslow estimator of the cumulative baseline hazard function. Here, d_j represents the number of events occurring at the j-th distinct failure time, and R_j denotes the set of all individuals who have experienced the failure at the j-th time. This estimation is performed using all significant covariates in a stratified Cox model. A graphical testing approach for the hypothesis of proportional hazard can be carried out by using the stratified estimator of hazard function instead of the Kaplan-Meier estimator. The methodology involves adjusting a stratified Cox's model to the strata, which are defined by the values of the covariates for which the hypothesis of proportional hazards is being tested.

- For each stratum, the basic function $S_0(t)$ is estimated, denoted by $\hat{S}_{0k}(t)$ for stratum k.
- Then, the vector of mean values of covariates in stratum k is denoted by $\bar{\boldsymbol{x}}_k$, and the estimated survival function for stratum k is calculated as $\hat{S}_k(t) = \hat{S}_{0k}(t) e^{\hat{\boldsymbol{\beta}}' \bar{\boldsymbol{x}}_k}$.
- Finally, if the assumption of proportional hazards holds for the covariate being studied, the lines of $\ln(-\ln(\hat{S}_k(t)))$ for $k = 1, \ldots, s$ should be parallel.

This graphical testing approach provides a visual assessment of the assumption of proportional hazards (Caroni, 2009).

2.1.7 Cox's Model Residuals

The Cox-Snell residuals in Cox's model are given by

$$-ln(\hat{S}(t_{(j)}; x_j)) = \hat{H}(t_{(j)}; x_j) = \hat{H}_0(t_{(j)})e^{\hat{\beta}' x_j}$$

where $\hat{H}_0(t)$ is a non-parametric estimator. However, these residuals are not always convenient to use. In contrast, the Cox-Snell residuals find significant applicability in parametric models. A specific type of residuals, applicable to the Cox model, relies on the derivation of the following variables. The probability of an event occurring for unit *j*, given that time t_j has elapsed, when the total number of units at risk just before this moment is denoted as R_j, is calculated as follows:

$$p_j = \frac{e^{\beta' x_j}}{\sum_{i \in R_j} e^{\beta' x_i}}$$

Considering that the unit from the total number R_j that will experience the event at time t_j is unknown, the values of covariates x for this unit can be treated as a random variable with an expected value:

$$E(x|R_j) = \sum_{k \in R_j} x_k p_k = \frac{\sum_{k \in R_j} x_k e^{\beta' x_k}}{\sum_{i \in R_j} e^{\beta' x_i}}$$

Now, we can easily define the residuals as:

$$r_j = x_j - E(x|R_j)$$

By replacing β with $\hat{\beta}$, we obtain the Schoenfeld residuals:

$$\hat{r}_j = x_j - \hat{E}(x|R_j)$$

In contrast to the residuals of the regression model $y_i - \hat{y}_i$, the Schoenfeld residuals ($\hat{r}_i$) are not derived from the values of the dependent variable (e.g., time t), but rather from the covariates x. It is important to note that the Schoenfeld residuals are computed at the event occurrence times, not at the censored observations used in the computation. These residuals are vectors because each uncensored observation has as many residuals as covariates.

Let $\hat{V}(\hat{r}_j)$ be the estimated variance matrix of $\hat{r}_j$. Then

$$\{\hat{V}(\hat{r}_j)\}^{-1}\hat{r}_j$$

represents the weighted Schoenfeld residuals, which are considered more powerful in detecting problems in Cox's model. The approximation

$$\{\hat{V}(\hat{r}_j)\}^{-1} \approx k\hat{V}(\hat{\beta})$$

where k is the population of non-censored observations and $\hat{V}(\hat{\boldsymbol{\beta}})$ is the estimated variance matrix of $\hat{\boldsymbol{\beta}}$, is a simplification. Finally, the scaled Schoenfeld residuals are given by

$$r_j^* = k\hat{V}(\hat{\boldsymbol{\beta}})\hat{r}_j$$

(Caroni, 2009).

2.2 Deep Learning

2.2.1 Artificial Neural Networks

Artificial neural networks are machine learning techniques inspired by the biological mechanism of learning in organisms. In the human nervous system, neurons are connected through synapses, and the strengths of these connections change in response to stimuli, enabling learning. Artificial neural networks simulate this process using computational units called neurons connected by weights.

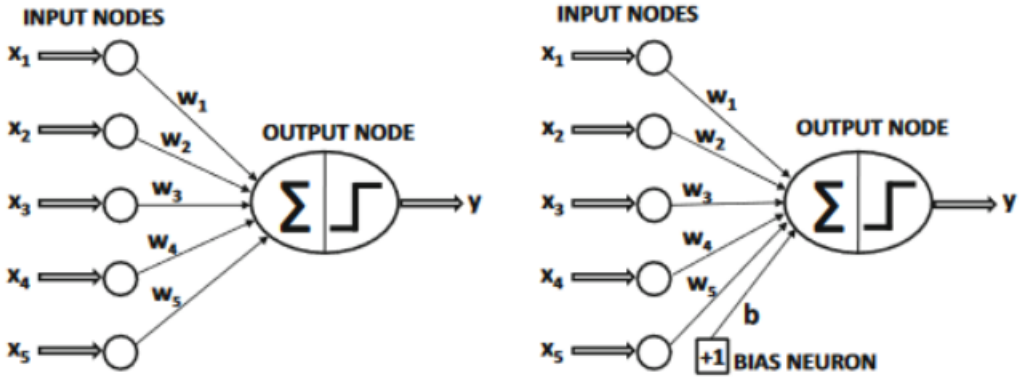
Training data, consisting of input-output pairs, is fed into the neural network to provide external stimuli. The network adjusts the weights between neurons based on the correctness of its predictions compared to the annotated output labels in the training data. This adjustment aims to reduce prediction errors and refine the computed function over time, enabling the network to generalize and accurately compute functions for unseen inputs.

While biological comparison is not a perfect representation, the principles of neuroscience have been useful in designing neural network architectures. Neural networks can be viewed as higher-level abstractions of classical machine learning models, with computational units inspired by traditional algorithms. By combining multiple units and training their weights jointly, neural networks can learn more complex functions than basic machine learning models (Aggarwal, 2018).

The Basic Architecture

Neural networks can be categorized into two main types: single-layer neural networks and multi-layer neural networks.

The perceptron is the simplest form of a neural network, consisting of a single input layer and an output node. It is visualized in Figure 2.1. In the context of training



(a) Perceptron without bias (b) Perceptron with bias

Figure 2.1: The basic architecture of the perception (Aggarwal, 2018)

instances represented as $(\overline{X}, y)$, where each $\overline{X} = [x_1, ...x_d]$ contains d feature variables and $y \in \{-1, +1\}$ represents the observed binary class variable, the perceptron aims to predict the class variable for unobserved cases. Historical data with observed class variables enables the perceptron to make predictions. The perceptron's architecture consists of a single input layer transmitting the features $\overline{X} = [x_1, ...x_d]$ to an output node. Computation occurs at the output node where the linear function

$$\hat{y} = \overline{W} \cdot \overline{X} = \sum_{i=1}^{d} w_i x_i \tag{2.1}$$

is calculated. The sign of this value determines the predicted class variable $\hat{y}$ using the sign function. Prediction errors are calculated as $E(X) = y - \hat{y}$, with possible values from the set $\{-2, 0, +2\}$. Nonzero errors necessitate weight updates in the direction opposite to the error gradient. Despite similarities to traditional machine learning models, the perceptron's interpretability as a computational unit allows for constructing more powerful models through the combination of multiple units. The perceptron's architecture, with its single input layer transmitting features to an output node, can be seen as a single-layer network. Different activation functions can simulate various machine learning models within the perceptron framework (Aggarwal, 2018).

In certain scenarios, the prediction process involves an invariant component known as the bias, which becomes crucial when dealing with mean-centered feature variables and imbalanced binary class distributions. To address this, an additional bias variable, denoted as b, is incorporated into the prediction equation. The prediction $\hat{y}$ is computed as

$$\hat{y} = sign\{\overline{W} \cdot \overline{X} + b\} = sign\{\sum_{j=1}^{d} w_j x_j + b\} \tag{2.2}$$

where $\overline{W}$ represents the weights and $\overline{X}$ denotes the feature variables. One way to introduce the bias is by using a bias neuron, which is a neuron that always transmits a value of 1 to the output node (Figure 2.1). The weight of the edge connecting the bias neuron to the output node corresponds to the bias variable. Alternatively, a feature engineering technique involves creating an additional feature with a constant value of 1, and the coefficient of this feature represents the bias. Usually, biases are not explicitly used in architectural representations for simplicity, as they can be incorporated using bias neurons. The training algorithms treat bias neurons like any other neuron with a fixed activation value of 1, ensuring that the subsequent analysis and predictions align with the assumption of Equation 2.2, which does not explicitly utilize biases (Aggarwal, 2018).

At the time of its proposal by Rosenblatt (Rosenblatt, 1958), the perceptron algorithm underwent heuristic optimization using hardware circuits without a formal notion of optimization in machine learning. Nevertheless, the goal was always to minimize prediction errors, even without a formal optimization formulation. The perceptron algorithm was designed heuristically to minimize misclassifications, and convergence proofs were

available for simplified settings. Hence, the goal of the perceptron algorithm can still be expressed in terms of least-squares minimization over a dataset D, consisting of feature-label pairs, as follows:

$$\min_{W} L = \sum_{(\overline{X},y)\in D} (y - \hat{y})^2 = \sum_{(\overline{X},y)\in D} (y - sign\{\overline{W} \cdot \overline{X}\})^2 \tag{2.3}$$

This objective function is known as a loss function and is commonly used in neural network learning algorithms. Unlike least-squares regression, which is defined for continuous-valued target variables, the perceptron's loss function exhibits step-like jumps and non-differentiability due to the sign function. Therefore, a smooth approximation of the gradient is used, denoted as ΔL_{smooth}, which allows for gradient descent optimization.

The perceptron's training algorithm operates by feeding each input data instance into the network individually and updating the weights based on the prediction error, $E(\overline{X}) = (y - \hat{y})$. The weights are updated using the equation:

$$\overline{W} \leftarrow \overline{W} + \alpha(y - \hat{y})\overline{X} \tag{2.4}$$

where α represents the learning rate. The perceptron algorithm iteratively adjusts the weights by cycling through the training examples in random order until convergence is reached. This approach can be seen as a stochastic gradient descent method, aiming to minimize the squared prediction error. Additionally, the use of mini-batch stochastic gradient descent, involving updates over randomly chosen subsets of training points, provides further advantages.

The perceptron model is a linear model, defining a hyperplane with the equation $\overline{W} \cdot \overline{X} = 0$. The model performs well on linearly separable datasets, where the value of $\overline{W} \cdot \overline{X}$ is positive on one side of the hyperplane and negative on the other. However, the perceptron algorithm tends to perform poorly on datasets that are not linearly separable, highlighting the limitations of this model and the need for more complex neural architectures.

While the original perceptron algorithm was a heuristic approach to minimizing classification errors, it was important to demonstrate its convergence to reasonable solutions in certain cases. It was shown that the perceptron algorithm converges to zero error on the training data when the data is linearly separable. However, convergence is not guaranteed for non-linearly separable data, and the perceptron may converge to suboptimal solutions compared to other learning algorithms (Aggarwal, 2018).

2.2.2 Multilayer Neural Networks

Multilayer neural networks consist of multiple computational layers, including input, hidden, and output layers. The hidden layers perform computations that are not directly visible to the user, while the output layer produces the final results. These networks are feed-forward, meaning that information flows from the input layer to the

output layer in a sequential manner. The connections between layers are typically fully connected, where all nodes in one layer are connected to all nodes in the next layer.

The architecture of a neural network is determined by the number of layers and the number and type of nodes in each layer. The choice of the loss function, which is optimized in the output layer, depends on the specific task. Common choices include softmax outputs with cross-entropy loss for discrete prediction tasks and linear outputs with squared loss for real-valued prediction tasks.

Bias neurons can be included in both the hidden layers and the output layer to introduce a shift in the activation of nodes. The presence or absence of bias neurons affects the structure of the neural network.

In a multilayer neural network with $p_1, p_2, ..., p_k$ units in each of the k layers, the output of each layer is represented by column vectors $h_1, h_2, ..., h_k$, respectively. The dimensionality of each layer refers to the number of units it contains. Figure 2.2 provides examples of multilayer neural networks, illustrating the presence or absence of bias neurons. In Figure 2.2 (a), a neural network with three layers is shown, including

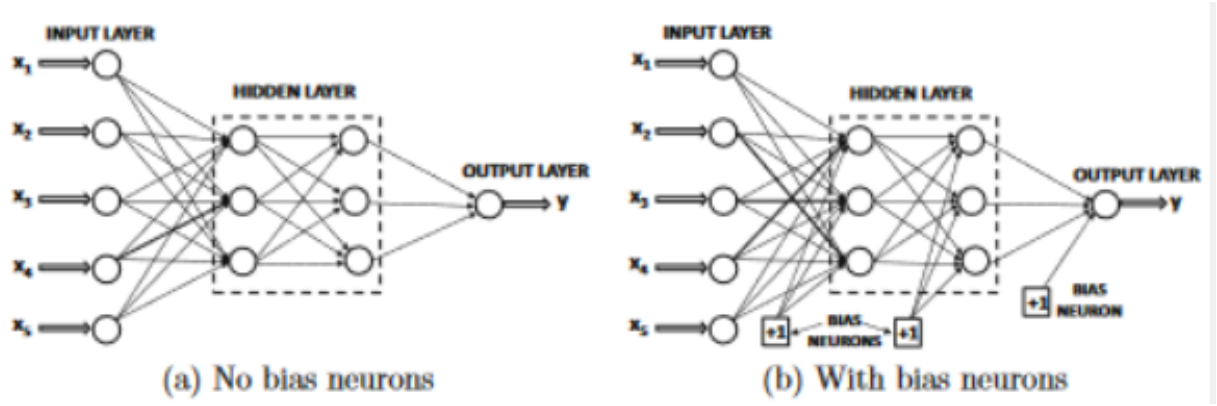


Figure 2.2: Multilayer neural networks (a) with no bias neurons (b) with bias neurons (Aggarwal, 2018)

an input layer, a hidden layer, and an output layer. On the other hand, Figure 2.2 (b) depicts a neural network with the same three layers but with bias neurons. These examples highlight the different configurations that can be used in multilayer networks to accommodate bias neurons and their impact on the overall structure of the network.

The weights of the connections in a multilayer neural network are represented by matrices. The matrix W_1 contains the weights between the input layer and the first hidden layer, with size $d \times p_1$. The weights between the rth hidden layer and the $(r+1)$th hidden layer are denoted by the $p_r \times p_{r+1}$ matrix W_r. If the output layer has o nodes, then the final matrix W_{k+1} is of size $p_k \times o$. The input vector x is transformed into the outputs using recursive equations. The activation functions, such as the sigmoid function, are applied element-wise to their vector arguments.

The equations are as follows:

$$\overline{h}_1 = \Phi(W_1^T \overline{x}) \quad \text{[Input to Hidden Layer]}$$
$$\overline{h}_{p+1} = \Phi(W_{p+1}^T \overline{h}_p) \forall p \in 1, ..., k-1 \quad \text{[Hidden to Hidden Layer]}$$
$$\overline{o} = \Phi(W_{k+1}^T h\overline{h}_k) \quad \text{[Hidden to Output Layer]}$$

In vector-based neural architectures, each layer is represented by a single vector unit, and the connections between the vector units are matrices. The assumption is that all units in a layer use the same activation function, applied element-wise to that layer. The architectural diagrams depict rectangular units for vector variables and circular units for scalar variables.

While the above equations and vector architectures are valid for layer-wise feed-forward networks, unconventional designs may have different rules, such as inputs incorporated in intermediate layers or connections between non-consecutive layers. The functions computed at a node can also be arbitrary computational functions.

Different variations of the classical architecture are possible, depending on the goals of the application. For example, an autoencoder is used for dimensionality reduction, where the number of inputs and outputs are equal. The hidden layers output the reduced representation of the data, but some loss in representation may occur.

In order to improve performance and avoid overfitting, connections in the network can be pruned or shared in a domain-specific way. Convolutional neural networks, designed for image data, are an example of weight pruning and sharing. Overfitting occurs when the number of free parameters (weight connections) is too large compared to the training data size. Increasing the number of nodes in the network tends to encourage overfitting, and methods like pretraining have been proposed to mitigate this issue.

Advanced training methods and network architectures are continuously researched and developed to improve the quality of learned solutions and address challenges such as overfitting.

2.2.3 Training a Neural Network with Backpropagation

The training process of a multi-layer neural network involves two main phases: the forward phase and the backward phase (Aggarwal, 2018).

1. Forward phase: In this phase, the inputs of a training instance are passed through the neural network, and computations propagate forward across the layers using the current set of weights. The final predicted output is compared to the target output of the training instance, and the derivative of the loss function with respect to the output is computed.

2. Backward phase: The goal of the backward phase is to learn the gradients of the loss function with respect to the weights by using the chain rule of differential calculus. These gradients are then used to update the weights. The backpropagation algorithm leverages dynamic programming to efficiently compute these gradients.

The backpropagation algorithm consists of recursively computing the gradients from the output layer to the input layer. Each layer's gradient is computed based on the gradients of the subsequent layers. The computation of the gradients involves the summation of local-gradient products over the different paths from a node to the output.

The generalized expression for computing the gradient of the loss function with respect to a weight connecting hidden unit h_r to h_{r+1}, denoted as $w_{(h_{r-1},h_r)}$, is as follows:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial o}\left[\frac{\partial o}{\partial h_k}\prod_{i=r}^{k-1}\frac{\partial h_{i+1}}{\partial h_i}\right]\frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}}, \quad \forall r \in 1...k \tag{2.5}$$

where L represents the loss function, o is the output, and $h_1, h_2, ..., h_k$ are the hidden units in the network.

To compute the gradients efficiently, the dynamic programming technique is used. The computation starts from the output layer and recursively propagates backward through the layers. The value of $\Delta(o, o)$ (the gradient of the loss function with respect to the output) is initialized as $(\frac{\partial L}{\partial o})$, and then the following recursion is applied:

$$\Delta(h_r, o) = \sum_{h:h_r \to h} \frac{\partial h}{\partial h_r}\Delta(h, o) \tag{2.6}$$

Here, $\Delta(h, o)$ represents the previously computed gradient for unit h. The computation of $\frac{\partial h}{\partial h_r}$ depends on the activation function Φ and the weight $w_{(h_r,h)}$, and it is given by:

$$\frac{\partial h}{\partial h_r} = \Phi'(a_h) \cdot w_{(h_r,h)} \tag{2.7}$$

where a_h is the value computed in hidden unit h just before applying the activation function Φ.

Finally, the update for the weight $w_{(h_{r-1},h_r)}$ in the backward direction is given by:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = h_{r-1} \cdot \Phi'(ah_r) \tag{2.8}$$

This process of recursively accumulating gradients and updating weights is performed in the backward phase until all layers have been processed. By efficiently computing the gradients using dynamic programming, the backpropagation algorithm enables effective training of multi-layer neural networks.

In the context of backpropagation, there are alternative ways to compute the dynamic programming recursion. One common approach uses the hidden layer variables as the "chain" variables, while another approach uses the pre-activation values of the variables. The pre-activation value of a hidden variable is obtained before applying the activation function.

Using the pre-activation values, the chain rule for computing the derivative with respect to the weights can be expressed as:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial o} \cdot \Phi'(a_o) \cdot \left(\sum_{[h_r,h_{r+1},\ldots,h_k,o] \in P} \frac{\partial a_o}{\partial a_{hk}} \prod_{i=r}^{k-1} \frac{\partial a_{hi+1}}{\partial a_{hi}} \right) \frac{\partial a_{h_r}}{\partial w_{(h_{r-1},h_r)}} \tag{2.9}$$

Here, we introduce the notation $\delta(h_r, o) = \frac{\partial L}{\partial a_{hr}}$ instead of $\Delta(h_r, o) = \frac{\partial L}{\partial h_r}$ for setting up the recursive equation. The value of $\delta(o, o) = \frac{\partial L}{\partial a_o}$ is initialized as:

$$\delta(o, o) = \Phi'(a_o) \cdot \frac{\partial L}{\partial o} \tag{2.10}$$

Using the multivariable chain rule, we can set up a similar recursion:

$$\delta(h_r, o) = \Phi'(a_{h_r}) \sum_{h:h_r \Rightarrow h} \cdot w_{(h_r,h)} \cdot \delta(h, o) \tag{2.11}$$

The partial derivative of the loss with respect to the weight is then computed using $\delta(h_r, o)$ as:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \delta(h_r, o) \cdot h_{r-1} \tag{2.12}$$

To train the neural network, the process of updating the nodes is repeated until convergence by cycling through the training data in epochs. It may take thousands of epochs for a neural network to learn the weights at the different units (Aggarwal, 2018).

2.2.4 Hyperparameter options

Optimizer

Optimizers play a role in the backpropagation step of a neural network. They determine how the weights and biases are updated. Commonly used optimizers are explained below.

1. Regular Gradient Descent (RGD): RGD calculates the gradient based on the sum of the errors of every training sample in an epoch, which means it processes the entire training dataset at once. However, RGD can be computationally expensive as it needs to store all the losses before updating the weights (Aggarwal, 2018).

2. Stochastic Gradient Descent (SGD): SGD calculates the gradient based on the error of a single training sample. It updates the weights after processing each training sample. However, SGD can fluctuate a lot, leading to overshooting minima due to the high variance caused by using a single sample at a time (Aggarwal, 2018).

3. Mini-Batch Gradient Descent (MBGD): MBGD strikes a balance between RGD and SGD. It calculates the gradient based on the sum of the errors of a small batch of training samples. MBGD offers a compromise between computational efficiency and reducing variance compared to SGD (Aggarwal, 2018).

4. Adam (Adaptive Moment Estimation): Adam estimates the first and second moments of the gradient of the loss for each weight in the neural network. It uses exponentially moving averages to estimate these moments based on the gradients evaluated on the current batch. Adam adapts the learning rate for each weight individually, resulting in faster convergence. It is able to find local minima that are close to the global minimum and performs well in areas with both small and large gradients.

The weight update equation for Adam is:

$$w_{(n+1)} = w_{(n)} - \eta \frac{\hat{m}_{(n)}}{\sqrt{\hat{v}_{(n)}} + \epsilon}$$

where $w_{(n)}$ represents the weight at iteration n, η is the learning rate, $\hat{m}_{(n)}$ and $\hat{v}_{(n)}$ are the bias-corrected estimates of the first and second moments respectively, and ϵ is a small constant to avoid division by zero.

Adam has become a popular optimizer due to its fast convergence and robust performance in various scenarios. The default hyperparameter values, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, are commonly used and often provide satisfactory results.

It's important to note that there are other optimizers available as well, such as RMSprop, Adagrad, and Adadelta, each with its own advantages and characteristics. The choice of optimizer depends on the specific problem and the characteristics of the data (Kingma and Ba, 2014)

Choice of Activation Functions

The choice of activation function is a critical aspect of neural network design. In the case of the perceptron, the sign activation function is chosen since it predicts a binary class label. However, different situations may require the prediction of different target variables. For instance, if the target variable is a real value, it is appropriate to use the identity activation function, resulting in the least-squares regression algorithm. When predicting the probability of a binary class, it is common to use the sigmoid function as the activation function for the output node. This allows the predicted value, $\hat{y}$, to represent the probability that the observed value, y, of the dependent variable is 1. The loss function is determined by the negative logarithm of $|y/2 - 0.5 + \hat{y}|$, assuming y is encoded as $\{-1, 1\}$. This loss function can be seen as representative of the negative log-likelihood of the training data.

Nonlinear activation functions become crucial when moving from single-layer perceptrons to multi-layer architectures. Different types of nonlinear functions, such as sign, sigmoid, or hyperbolic tangents, may be used in various layers. We denote the activation function as Φ:

$$\hat{y} = \Phi(\overline{W} \cdot \overline{X}) \tag{2.13}$$

Thus, a neuron computes two functions within the node. The value computed before applying the activation function $\Phi(\cdot)$ is referred to as the pre-activation value, while

the value computed after applying the activation function is the post-activation value. The output of a neuron is always the post-activation value, although the pre-activation variables are often used in various analyses, such as the computations of the back-propagation algorithm.

The most basic activation function $\Phi(\cdot)$ is the identity or linear activation:

$$\Phi(v) = v \tag{2.14}$$

The linear activation function is often used in the output node when the target is a real value, or even for discrete outputs when a smoothed surrogate loss function needs to be set up.

Early in the development of neural networks, classical activation functions such as the sign, sigmoid, and hyperbolic tangent functions were commonly used:

$$\begin{aligned} \Phi(v) &= \text{sign}(v) \quad \text{(sign function)} \\ \Phi(v) &= \frac{1}{1+e^{-v}} \quad \text{(sigmoid function)} \\ \Phi(v) &= \frac{e^{2v}-1}{e^{2v}+1} \quad \text{(tanh function)} \end{aligned} \tag{2.15}$$

While the sign activation can map to binary outputs at prediction time, its non-differentiability prevents its use in creating the loss function during training. The sigmoid activation outputs values in the range (0, 1), making it useful for interpreting computations as probabilities and constructing loss functions based on maximum likelihood models. The hyperbolic tangent function (tanh) is similar in shape to the sigmoid function but re-scaled and translated to the range $[-1, 1]$. The tanh function is preferred when both positive and negative outputs are desired, and its mean-centering and larger gradient make it easier to train compared to the sigmoid function.

In recent years, piecewise linear activation functions such as the Rectified Linear Unit (ReLU) and hard tanh have gained popularity:

$$\begin{aligned} \Phi(v) &= \max\{v, 0\} \quad \text{(ReLU)} \\ \Phi(v) &= \max\{\min[v, 1], -1\} \quad \text{(hard tanh)} \end{aligned} \tag{2.16}$$

ReLU and hard tanh have replaced sigmoid and soft tanh in modern neural networks due to their ease of training in multilayer networks.

These activation functions are visualized in Figure 2.3. It's important to note that all shown activation functions are monotonic, and many of them saturate at large absolute values, meaning further increasing the argument does not significantly change the activation output. Nonlinear activation functions play a fundamental role in enhancing the modeling power of a network, allowing for more powerful compositions of different types of functions (Aggarwal, 2018).

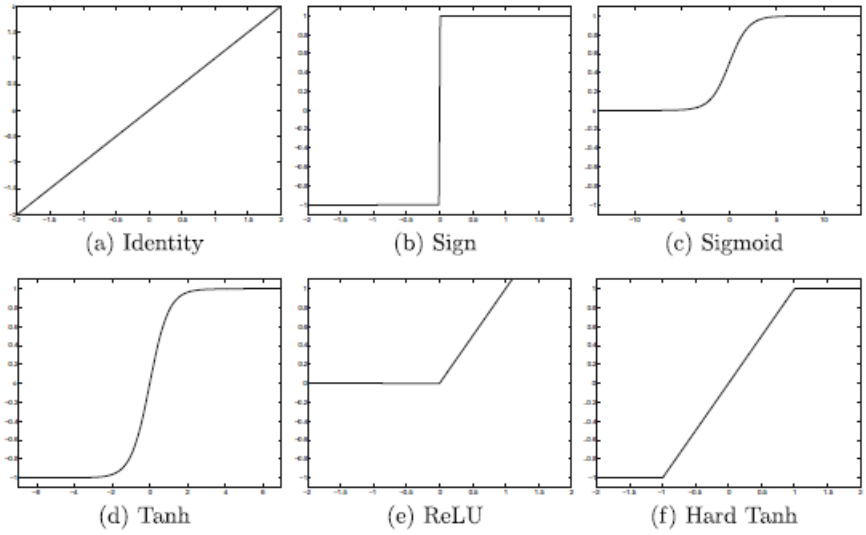


Figure 2.3: Various activation functions (Aggarwal, 2018)

Learning rate

The learning rate is a critical hyperparameter in neural network training through backpropagation. It determines the magnitude of weight updates at each iteration. Choosing an appropriate learning rate is crucial for achieving optimal results This is illustrated in Figure (2.4).

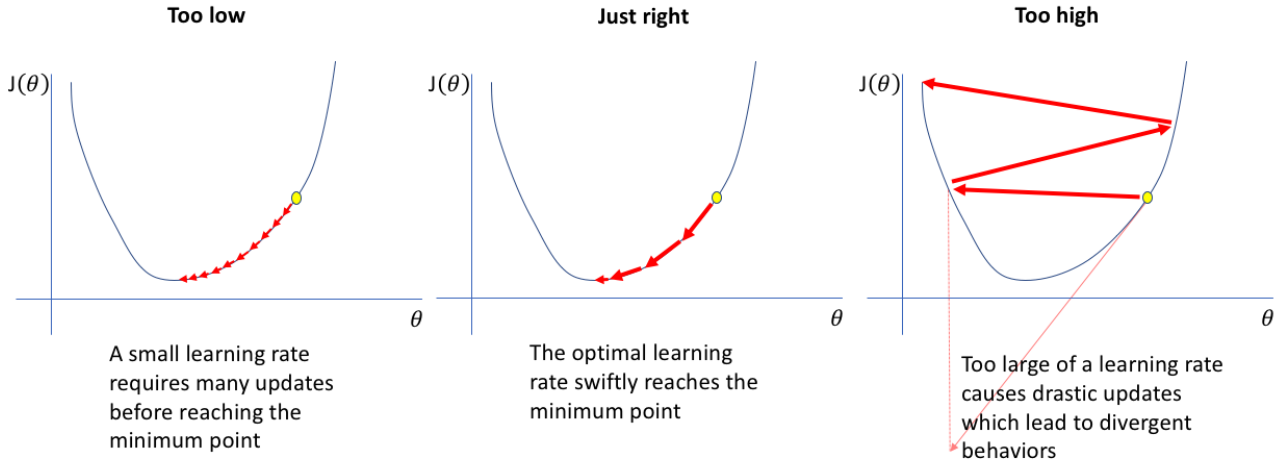


Figure 2.4: Effects of different learning rates. θ are the weights and $J(\theta)$ is the loss function (Jordan, 2023)

If the learning rate is too low, the training process becomes slow because the network takes a long time to converge to the minimum of the loss function. On the other hand, if the learning rate is too high, the network can oscillate around the minimum or even

diverge, preventing convergence (Aggarwal, 2018).

To strike a balance, it is ideal to use a larger learning rate in the initial stages of training when the weights are far from the optimal solution. This allows the network to quickly converge to a vicinity of a good local or global minimum. As training progresses and the network approaches the minimum, a smaller learning rate is preferred to make smaller adjustments to the weights for fine-tuning and achieving the best possible result (Aggarwal, 2018).

One common approach to address this is learning rate decay. Learning rate decay involves reducing the learning rate as the training progresses. Different decay strategies can be used, such as a step function, exponential decay, or other user-defined functions. By gradually reducing the learning rate, the network can make smaller and more precise weight adjustments as it gets closer to the minimum, improving convergence and performance (Aggarwal, 2018).

It's worth noting that finding the optimal learning rate and decay strategy can be problem-specific and may require experimentation. Techniques like learning rate schedules, adaptive learning rates (e.g., in Adam optimizer), or using learning rate annealing can also be employed to dynamically adjust the learning rate during training based on the network's progress.

Choosing an appropriate learning rate and incorporating learning rate decay techniques contribute to achieving better convergence and more accurate neural network models.

Choice and Number of Output Nodes

The choice and number of output nodes in a neural network are influenced by the activation function, which depends on the specific application. For k-way classification tasks, *k* output values can be utilized with a softmax activation function. Each output node has an activation function defined as follows:

$$\Phi(v)_i = \frac{e^{v_i}}{\sum_{j=1}^{k} e^{v_j}} \quad \text{for} \quad i \in \{1, \ldots, k\}$$

These *k* values represent the outputs of *k* nodes, with $v_1, ..., v_k$ as the inputs. The softmax function converts the outputs of the final hidden layer into probabilities, where each output corresponds to the probability of a specific class.

In certain cases, linear (identity) activations are used in the final hidden layer before the softmax layer. The softmax layer does not have associated weights since its purpose is to convert real-valued outputs into probabilities. The softmax function, combined with a single hidden layer of linear activations, implements a model known as multinomial logistic regression.

Multiple output nodes are also employed in architectures such as autoencoders, where the goal is to fully reconstruct each input data point. Autoencoders can be utilized for matrix factorization methods like singular value decomposition (Aggarwal, 2018).

Choise of Hidden and Output Layers

The activation functions in the hidden layer of a neural network serve the purpose of introducing nonlinearity, enabling the network to approximate nonlinear relationships. The derivative of the activation function is essential in the backpropagation algorithm for computing gradients. Therefore, it is advantageous to choose activation functions with easily computable derivatives to ensure computational efficiency. There are various activation functions available for selection, but commonly used ones in hidden layers are the Sigmoid and ReLU functions [15]. It is possible to have different activation functions for each layer or even each neuron, but for simplicity, this report employs a single activation function per layer (Aggarwal, 2018).

In the output layer of a neural network, the choice of activation function is crucial as it determines the format of the network's output. For classification problems, it is typical to use an activation function that produces values ranging from 0 to 1, such as the Sigmoid function. This facilitates the computation of the likelihood or probability of a certain class prediction. In contrast, for regression problems where the goal is to predict continuous values, a linear activation function is commonly employed. The linear activation function allows for unbounded output values, which is advantageous for extrapolating beyond the bounds of the training data, especially when dealing with normalized data (Aggarwal, 2018).

Determining the number of layers in a neural network is a crucial design decision. While there is no one-size-fits-all answer, it depends on the complexity of the problem and the availability of data. In general, adding more layers to the network allows for increased model capacity and the ability to learn more complex representations. However, this comes at the cost of increased computational complexity and potential overfitting if the network becomes too large relative to the available data. It is essential to strike a balance by considering the complexity of the problem, the size of the dataset, and regularly monitoring the model's performance through validation and test sets (Aggarwal, 2018).

Choice of Loss Function

The choice of the loss function is crucial as it determines how the outputs are defined and how the model responds to different applications. For instance, in least-squares regression with numeric outputs, a simple squared loss is used, given by $(y - \hat{y})^2$ for a single training instance with target y and prediction $\hat{y}$. However, other types of loss functions can be employed depending on the scenario. For instance, when dealing with binary classification where $y \in \{-1, +1\}$ and real-valued prediction $\hat{y}$ (using the identity activation function), the hinge loss can be utilized:

$$L = \max\{0, 1 - y \cdot \hat{y}\}$$

The hinge loss is commonly employed in support vector machines (SVMs), which are a type of learning method.

For probabilistic predictions, two types of loss functions are used depending on the binary or multiway nature of the prediction:

1. Binary targets (logistic regression): The loss function for a single instance with a real-valued prediction $\hat{y}$ and observed value y (with identity activation) is defined as:

$$L = \log(1 + \exp(-y \cdot \hat{y}))$$

2. Categorical targets: If $\hat{y}_1, \ldots, \hat{y}_k$ are the probabilities of k classes (using the softmax activation) and the rth class is the ground-truth class [1], the loss function for a single instance is defined as:

$$L = -\log(\hat{y}_r)$$

The choice of output nodes, activation functions, and loss functions depends on the specific application and are interdependent. The perceptron criterion is rarely used as the loss function in practice. Softmax activation with cross-entropy loss is commonly used for discrete-valued outputs, while linear activation with squared loss is common for real-valued outputs. In general, cross-entropy loss is easier to optimize compared to squared loss (Aggarwal, 2018).

2.2.5 The Problem of Overfitting

The problem of overfitting occurs when a model fits the training data well but performs poorly on unseen test data. This issue is particularly prominent when the model is complex and the data set is small. To illustrate this, let's consider a simple single-layer neural network with five attributes and an identity activation function. The network aims to learn the following function:

$$\hat{y} = \sum_{i=1}^{5} w_i \cdot x_i$$

Suppose we have a situation where the target value is always twice the value of the first attribute, while the other attributes are unrelated. However, we only have four training instances, which is one less than the number of features. The training instances are as follows:

[1] The "ground-truth class" refers to the true or actual class label of a given sample or instance in a classification problem. In supervised learning, when training a model, each sample in the training dataset is associated with a known class label. The ground-truth class represents the correct class label assigned to a particular instance in the dataset. During the training process, the model aims to predict the class label for each instance. The loss function compares the predicted probabilities with the ground-truth class label to measure the discrepancy or error. By minimizing this error, the model learns to make more accurate predictions and generalize to unseen data.

x_1	x_2	x_3	x_4	x_5	y
1	1	0	0	0	2
2	0	1	0	0	4
3	0	0	1	0	6
4	0	0	0	1	8

The correct parameter vector in this case is $W = [2, 0, 0, 0, 0]$ based on the known relationship between the first feature and the target. This parameter vector produces zero error on the training data. However, due to the limited number of training points, there are infinitely many parameter sets that also yield zero error on the training data. For example, the parameter set $[0, 2, 4, 6, 8]$ also gives zero error. However, this alternative solution is unlikely to generalize well to unseen test data where the relationship between the features and the target is different.

The problem of overfitting arises because the model is encoding random nuances from the limited training data, leading to poor generalization. Increasing the number of training instances improves the model's ability to generalize, while increasing the complexity of the model reduces its generalization power. A rule of thumb is to have a training data set that is at least 2 to 3 times larger than the number of parameters in the neural network. Models with a higher number of parameters have higher capacity but require more data to generalize effectively to unseen test data.

Neural networks can theoretically approximate any function, but the availability of data plays a crucial role in achieving good performance. The design of neural networks must carefully address the issue of overfitting, even when a large amount of data is available. Following, various design methods to mitigate the impact of overfitting in neural networks are provided (Aggarwal, 2018).

2.2.6 Regularization

To reduce overfitting, the number of non-zero parameters in the model can be constrained. This is achieved by adding a penalty term to the loss function. The penalty term is typically the squared value of each parameter multiplied by a regularization parameter λ. An example of regularized version of Equation 2.4 is shown below:

$$\overline{W} \leftarrow \overline{W}(1 - \alpha\lambda) + \alpha \sum_{\overline{X} \in S} E(\overline{X})\overline{X}$$

where $E[\overline{X}]$ represents the current error $(y - \hat{y})$ between observed and predicted values of training instance $\overline{X}$ (Aggarwal, 2018).

Ensemble Methods

Ensemble methods, such as bagging, can be used to increase the generalization power of the model. These methods combine multiple models to make predictions.

Dropout and Dropconnect are specific ensemble methods used in neural networks. Dropout is a regularization technique where, during training, a random subset of neurons is temporarily "dropped out" or ignored. This means that these neurons do not contribute to the forward pass or backward pass of the network for that particular training iteration. By randomly dropping out neurons, the network becomes more robust and prevents co-adaptation among neurons, forcing them to learn more general features. Dropout has been shown to effectively reduce overfitting and improve the generalization performance of neural networks. Dropconnect is similar to Dropout, but instead of dropping out entire neurons, it randomly sets a fraction of the weights in the network to zero during training. This means that each connection between neurons has a probability of being dropped or disconnected. Dropconnect provides a form of regularization that encourages the network to learn redundant representations and prevents individual connections from dominating the learning process. Like Dropout, Dropconnect has been found to enhance the generalization capabilities of neural networks (Aggarwal, 2018).

2.2.7 Neural Network Architecture

Designing the architecture of a neural network is an essential aspect that can help mitigate overfitting and improve performance. The choice of architecture can be tailored to the specific characteristics of the data domain, such as text or image data, where relationships between nearby elements are significant. Specialized architectures, like convolutional neural networks (CNNs), have proven effective for tasks involving spatial or sequential data by incorporating parameter sharing mechanisms to reduce the overall number of parameters.

The structure of a neural network, including the number of hidden layers and neurons in each layer, significantly influences the network's ability to learn the desired mapping or function. Although there are no strict rules for determining these parameters, some general guidelines have been suggested. For example, [4] provides several rules of thumb for determining the number of neurons in the first hidden layer. However, it is important to note that these rules may sometimes contradict each other, underscoring the challenge of providing definitive guidelines:

The number of hidden neurons can fall between the size of the input layer and the size of the output layer. The number of hidden neurons can be calculated as 2/3 the size of the input layer, plus the size of the output layer. The number of hidden neurons can be less than twice the size of the input layer. While these rules can serve as initial starting points, designing an optimal neural network structure often requires an iterative process of trial and error. It is advisable to begin with a simpler network and gradually increase its complexity if necessary to achieve the desired level of accuracy or inference capability (Aggarwal, 2018).

It is crucial to avoid unnecessarily large networks as they can lead to overfitting. Overfitting occurs when a network becomes large enough to memorize the training data, including the noise, rather than learning the underlying features. Overfitting

is typically identified when the network performs well on the training data but shows relatively poor performance on the validation data (Aggarwal, 2018).

Conversely, underfitting occurs when the network lacks sufficient trainable parameters, such as weights and biases, to capture the complexity of the data's features. Underfitting results in poor performance on both the training and validation sets.

In summary, designing an appropriate neural network architecture involves understanding the characteristics of the data domain, selecting specialized architectures if necessary, and iteratively refining the structure to achieve the desired performance while avoiding overfitting or underfitting (Aggarwal, 2018).

2.2.8 Epochs and Batch Size

The choice of epochs and batch size plays a crucial role in managing the data during neural network training. The available data is typically divided into three sets: the training set, the validation set, and the test set. The training set is used to adjust the weights and biases of the network. To enhance training efficiency, the training data is further divided into smaller batches. For most optimizers, such as Adam and Mini-batch Gradient Descent, the weights and biases are updated after each batch has passed through the network. In contrast, Stochastic Gradient Descent or Regular Gradient Descent update the weights and biases after processing a single training sample or all training samples, respectively. In these cases, the batch size does not have a significant impact (Aggarwal, 2018).

Once all batches, representing all training samples in the training set, have been processed (an event known as an epoch), the network's performance is evaluated using the validation set. The samples in the validation set are fed into the network, and the resulting loss is calculated. Although this loss does not contribute to weight updates, it is a crucial step as it provides insights into the network's performance on unseen data and its ability to generalize. After calculating the validation loss, the network proceeds to the next epoch, repeating the process of iterating through the training samples to update the weights and biases. The network's performance is then re-evaluated on the validation set. This process is repeated for a predefined number of epochs.

During hyperparameter tuning, the values or options of the hyperparameters are adjusted based on the network's performance on the validation set. However, it is important to note that continuously tuning the hyperparameters based on the validation set may introduce bias towards that specific data. To ensure an unbiased assessment of the network's performance, the final network is tested on a separate test set. This test set consists of data that has not been seen by the network before, and the resulting loss from this evaluation provides a fair and unbiased measure of the network's overall performance.

Early Stopping

To prevent overfitting, which occurs when the training loss continues to decrease while the validation loss starts to increase, early stopping can be employed. Early stopping involves monitoring the validation loss during training. If the validation loss does not decrease for a predefined number of epochs, training is halted. This technique saves training time by avoiding unnecessary iterations and improves the network's performance by preventing overfitting to the training set. Early stopping ensures that the network has better generalization capabilities compared to a network that continues to train and potentially overfits on the training data (Aggarwal, 2018).

2.2.9 The Vanishing and Exploding Gradient Problems

The deep neural network faces challenges related to vanishing and exploding gradient problems. The term "deep" of neural networks refers to networks with a significant number of hidden layers. A deep neural network consists of multiple layers of interconnected nodes, allowing it to learn and represent complex patterns in the data. When the network has a large number of layers, the updates during the backpropagation process can become unstable. This instability manifests as either excessively small updates (vanishing gradient) or excessively large updates (exploding gradient) in certain neural network architectures. The underlying cause is the chain-like product computation in the backpropagation process, which can exponentially decrease or increase along the path. The vanishing gradient problem occurs when the product of weight and activation function derivatives progressively diminishes, while the exploding gradient problem arises when the product increases uncontrollably. Even if the local derivatives have an expected value of 1, the overall derivative can still exhibit instability based on their distribution. These challenges are inherent to deep networks and contribute to the instability of their training process (Aggarwal, 2018).

Numerous solutions have been proposed to mitigate the vanishing and exploding gradient problems. One approach involves using activation functions that are less prone to vanishing gradients, such as the Rectified Linear Unit (ReLU), which has a derivative of 1 for positive values. In addition to the choice of activation function, various gradient-descent techniques are employed to improve convergence behavior. These techniques include adaptive learning rates and conjugate gradient methods, which can be beneficial in many cases (Aggarwal, 2018).

2.2.10 Automatic hyperparameter tuning methods

Tuning the hyperparameters of a neural network is a challenging task, as there are no strict rules or predefined values for setting these hyperparameters. The behavior of hyperparameters can vary depending on their interactions with other hyperparameters, making it difficult to find the optimal combination. Manual hyperparameter tuning typically involves a trial and error approach, where engineers rely on their expertise and experience, along with some luck, to find suitable values.

However, the automation of hyperparameter tuning has gained significant interest. Several automatic hyperparameter tuning methods have been developed to alleviate the burden of manual tuning. Two commonly used methods are grid search and random search.

Grid Search

Grid search is a hyperparameter tuning method that involves systematically searching over a grid of hyperparameter values. The user defines a discrete grid for each hyperparameter, and all possible combinations of hyperparameters are tested.

The main advantage of grid search is that it guarantees finding the best combination of hyperparameter values within the specified grid, leading to the lowest validation loss. By exhaustively exploring all combinations, grid search ensures that no potential optimal configuration is overlooked.

However, a significant drawback of grid search is its computational expense. The number of neural networks that need to be trained grows exponentially with the number of hyperparameters and their values. For example, if there are five hyperparameters, each with five values to be tested, grid search would require training $5^5 = 3125$ neural networks. If an additional hyperparameter with three options is introduced, the number of networks to be trained increases to $3125 * 3 = 9375$. This exponential growth in the number of networks to be trained is often referred to as the "curse of dimensionality."

As a result, the computational cost of grid search can become prohibitively high, especially when dealing with a large number of hyperparameters or a large range of values for each hyperparameter. Therefore, while grid search guarantees finding the optimum within the defined grid, its computational complexity may limit its practicality in scenarios with many hyperparameters.

It is worth noting that the curse of dimensionality can be mitigated by carefully selecting a smaller and more informative grid, guided by prior knowledge or domain expertise. This can help reduce the number of network trainings required while still exploring a meaningful portion of the hyperparameter space.

Random Search

Random search is a hyperparameter tuning method in which the values or options for the hyperparameters are randomly chosen. These values can be selected from either a discrete or continuous domain, depending on the nature of the hyperparameter. The selection of hyperparameter values is based on probability. In the case of a discrete domain, each option within the domain has an equal chance of being chosen. For continuous domains, the user can specify a distribution, such as a uniform or normal distribution, from which the hyperparameter values are sampled.

At first glance, random search might seem less effective compared to grid search. However, there are two key factors that make random search a valuable approach.

Firstly, when the possible hyperparameter values/options in random search match the discrete values used in grid search, it has been shown that training around 60

networks using random search is typically sufficient to obtain a neural network that performs within the top 5% of the best-performing networks from the grid search, with a 95% probability. This can be explained by considering the probability of randomly sampling a combination of hyperparameter values that falls within the top 5% best-performing combinations in the grid search. The probability of not hitting this top 5% region is $(1 - 0.05)$, and if n points are randomly sampled, the probability of hitting the top 5% is $1 - (1 - 0.05)^n$. To achieve at least a 0.95 probability of success, the inequality $1 - (1 - 0.05)^n > 0.95$ should hold. Solving this inequality, we find that n should be greater than or equal to 60. This approach is useful if the neural networks, parameterized with the top-performing combinations from the grid search, still exhibit satisfactory performance. This is often the case when the region of near-optimal hyperparameter values is relatively large or when there are many grid points within that region.

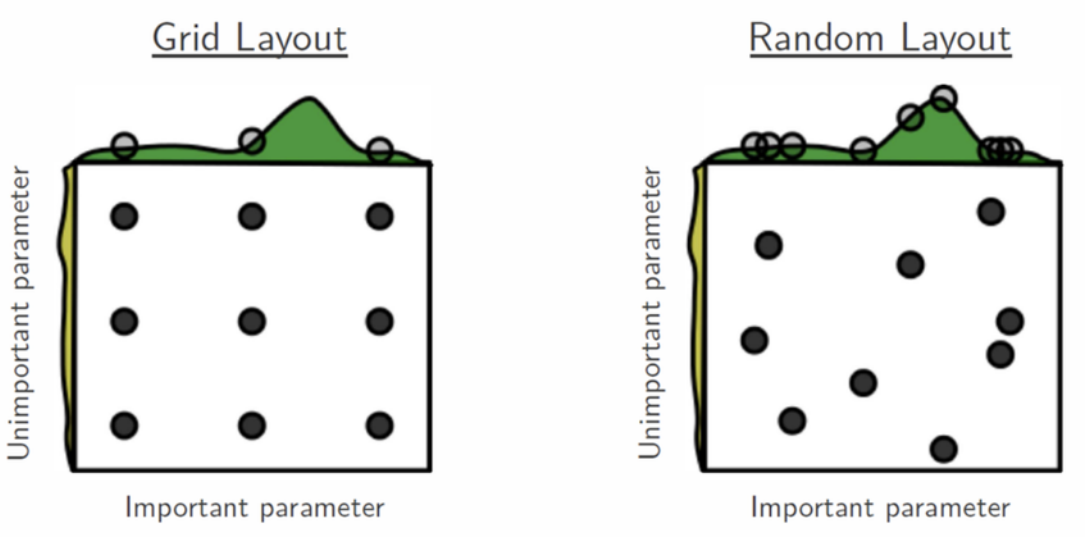


Figure 2.5: Grid and Random Layout (Lippert et al., 2022)

Secondly, random search allows for exploring a larger set of hyperparameter values/options when compared to grid search, particularly when the hyperparameters are defined in continuous ranges instead of discrete values. This is achieved at a similar computational cost, as not all possible values need to be exhaustively combined with each other. This advantage is illustrated in Figure 2.5, where two hyperparameters that do not significantly influence each other are tested together. Random search explores a wider range of values for the important hyperparameter, and importantly, it identifies values much closer to the maximum performance (indicated by the green peak) that were missed by grid search. This capability of random search to outperform grid search highlights its potential for finding better hyperparameter configurations.

In summary, random search is a valuable alternative to grid search in hyperparameter tuning. Despite its random nature, it has been shown to achieve competitive

results with a smaller number of network trainings when the hyperparameter values/options are matched to those in grid search. Additionally, random search's ability to explore continuous ranges of hyperparameter values can lead to the discovery of better-performing configurations that might be missed by grid search.

2.2.11 Feature Selection

Feature selection is a crucial aspect of machine learning that focuses on reducing the dimensionality of data to improve the performance of learning algorithms. The goal is to identify and retain only the relevant features while eliminating irrelevant and redundant ones. By doing so, feature selection enhances predictive accuracy, model interpretability, learning efficiency, and facilitates effective data collection (Liu, 2011).

Effective learning relies on two key factors: the number of features (M) and the number of instances (N). Increasing the number of instances provides more constraints for guiding the search for a correct hypothesis, resulting in a more reliable model. On the other hand, reducing the number of features decreases the hypothesis space, effectively increasing the effective number of instances and aiding in the creation of a compact model (Liu, 2011).

Complex models that include irrelevant features lead to overfitting, where the learned hypothesis performs poorly on unseen data. This issue is known as the curse of dimensionality, where the addition of dimensions to a high-dimensional space exponentially increases the required data size, impairing learning performance. Dimensionality reduction techniques, such as feature selection, can mitigate the curse of dimensionality by shrinking the hypothesis space (Liu, 2011).

The structure of a feature selection system consists of four components: input, search, evaluation, and output. The output can be a ranked list or a subset of features. Feature selection systems can operate with different types of learning algorithms: supervised, unsupervised, and semi-supervised. Supervised feature selection utilizes class labels to determine feature quality, often by measuring correlations or differentiation capabilities. Unsupervised feature selection focuses on promoting data separability without relying on class labels. Semi-supervised feature selection leverages a small number of labeled instances and unlabeled data to constrain the feature selection problem further (Liu, 2011).

The search for relevant features can be accomplished through feature ranking or subset selection. Feature ranking involves ranking features based on their intrinsic properties, allowing for the selection of the top-ranked features. Subset selection involves selecting a subset of features without significant differences among them. Various search strategies, such as forward selection, backward elimination, and random selection, can be employed (Liu, 2011).

There are three classic models of feature selection: filter, wrapper, and embedded.

Filter models use measures based on intrinsic data properties, such as mutual information or data consistency. Wrapper models rely on the performance of induction algorithms to guide the feature selection process, resulting in features that are more compatible with downstream analyses. Although wrapper methods can be computationally expensive, they often yield superior results compared to filter-based techniques. Embedded methods combine the benefits of wrapper and filter methods by incorporating feature interactions while maintaining reasonable computational costs. Popular examples of embedded methods include tree-based methods and shrinkage-based methods like LASSO (Liu, 2011).

In summary, feature selection enhances the learning process, mitigates the curse of dimensionality, and facilitates the creation of compact and effective models.

2.3 Overview of Deepsurv

The limitations of the Cox proportional hazards (CPH) model in capturing complex relationships among covariates in nonlinear risk surfaces and the underperformance of single hidden layer neural networks compared to CPH have motivated the utilization of deep learning techniques to model the intricate interactions between covariates and the risk of failure. In response to the drawbacks of the CPH model and the Faraggi-Simon (Faraggi and Simon, 1995) network, Deepsurv has been developed as a non-linear extension of the Cox proportional hazards model. By employing deep architecture, Deepsurv, which is a multi-layer perceptron, aims to model the complex associations among covariates without relying on any prior assumptions about the underlying risk surface.

In survival analysis, the survival data typically consists of covariate data (X), an event indicator (e), and event time (t). Deepsurv is a feed-forward neural network that takes an n-dimensional array of covariates as input. The network can be configured with any number of hidden layers, and the input data is propagated through these layers. Each hidden layer is equipped with a non-linear activation function and can include additional layers such as dropout layers and batch normalization layers. The output layer of the network consists of a single node with a linear activation function, and its output corresponds to the estimated log partial hazard ($\hat{h}(x)$).

The Deepsurv model employs a fully connected deep neural network architecture, utilizing the average negative log-likelihood as the loss function. The loss function is derived from the Cox partial likelihood and the Faraggi-Simon network. In the general training of the CPH model, the weights (θ) are optimized to maximize the Cox partial likelihood, denoted by $L_c(\theta)$:

$$L_c(\theta) = \prod_{i:E_i=1} \frac{\exp(\hat{h}_\theta(x_i))}{\sum_{j \in R(T_i)} \exp(\hat{h}_\theta(x_j))} \tag{2.17}$$

The Faraggi-Simon network, as a preliminary attempt at a nonlinear extension of the CPH model, optimizes the network weights using a modified version of $L_c(\theta)$, where the output of the network $\hat{h}(x)$ replaces $\hat{h}_\theta(x)$. This optimization is coupled with $l2$ regularization, resulting in the "penalized likelihood function" of the Faraggi-Simon network:

$$L_c(\theta) + \lambda \sum_{k=1}^{p} \theta_k^2 \tag{2.18}$$

Here, $L_c(\theta)$ represents the modified Cox partial likelihood parameterized by the weights of the Faraggi-Simon network, λ is the $l2$ regularization parameter, and p is the number of predictor variables (X: $n \times p$ data matrix).

Similar to the Faraggi-Simon network, Deepsurv employs a penalized negative average log-likelihood function given by the equation:

$$l(\theta) := -\frac{1}{N} \sum_{i:E_i=1} \left(\hat{h}_\theta(x_i) - \log \left(\sum_{j \in R(T_i)} e^{\hat{h}_\theta(x_j)} \right) \right) + \lambda \theta_k^2 \tag{2.19}$$

In this equation, $E_i = 1$ represents the set of subjects with observed events ($e = 1$), N_E is the total number of subjects with observed events, and $\hat{h}(x)$ denotes the output of the network, which estimates the log partial hazard (Katzman et al., 2018).

One of the advantages of Deepsurv lies in its utilization of modern techniques in deep learning, such as dropout (Srivastava et al., 2014), weight decay regularization, batch normalization (Yuan et al., 2019), learning rate scheduling, gradient clipping, and advanced optimizers such as the Adaptive Moment Estimation (Adam) optimization algorithm (Kingma and Ba, 2014). Additionally, DeepSurv can benefit from the use of gradient descent methods such as Nesterov momentum, which is a variant of the basic momentum method.

While maintaining the constant hazard ratio assumption from the CPH model, Deepsurv also assumes that the training data does not contain tied events, and appropriate measures should be taken when dealing with such events. Methods like the Breslow approximation or the Efron approximation are commonly employed to handle ties. Nevertheless, Deepsurv demonstrates the capability to learn the complex relationship among covariates. The predictive performance of the model is validated using the concordance index (c-index).

2.4 Performance metrics

2.4.1 Concordance Index

The concordance index, also known as the c-index is derived from the Wilcoxon-Mann-Whitney two-sample rank test, is computed by taking all possible pairs of subjects such that one subject responded and the other did not. The index is the proportion of such pairs with the responder having a higher predicted probability of response than the nonresponder (Harrell, 2016). In short, the C-index estimates the probability that, for a random pair of individuals, the predicted survival times of the two individuals have the same ordering as their true survival times. Bamber (1975) and Hanley and McNeil (1982) have shown that c is identical to a widely used measure of diagnostic discrimination, the area under a "receiver operating characteristic" (ROC) curve, is a widely used evaluation metric in survival analysis and medical research. It measures the predictive accuracy of a model in terms of its ability to rank the relative risks or survival probabilities of different individuals (Harrell, 2016). The concordance index, denoted as c, is calculated as follows:

$$c = \frac{\text{Number of concordant pairs} + 0.5 \times \text{Number of tied pairs}}{\text{Total number of pairs}}$$

Here, the number of concordant pairs represents the pairs of individuals in which the predicted outcome is consistent with the observed outcome. The number of tied pairs accounts for situations where two or more individuals have the same predicted outcome. The total number of pairs refers to all possible pairs of individuals in the dataset. The C-index is particularly valuable in assessing proportional hazards models because it solely relies on the order of predictions. This is advantageous because the order of proportional hazards models remains constant over time, allowing us to utilize the relative risk function instead of a metric for predicting survival time. The concordance index ranges from 0 to 1, with a value of 1 indicating perfect predictive accuracy and a value of 0.5 indicating random predictions. The concordance index provides a quantitative measure of how well a model distinguishes between individuals with different risks or survival times.

2.4.2 Integrated Brier Score

The Integrated Brier Score (IBS) or Integrated Graf Score is a commonly used metric in survival analysis to assess the predictive accuracy of survival models. It is a generalization of the Brier Score, which is widely used in probability forecasting, to the context of survival analysis. The IBS measures the average squared difference between predicted and observed event probabilities over a specified time horizon.

In survival analysis, the predicted probabilities refer to the predicted survival probabilities or the hazard probabilities from a given survival model. The observed probabilities are the actual event occurrences, typically represented as binary indicators (0

for censored observations and 1 for event occurrences) within the specified time frame.

The IBS is particularly useful for comparing the predictive performance of different survival models. A lower IBS value indicates better predictive accuracy, as it signifies that the model's predicted probabilities are closer to the observed event occurrences.

The Integrated Brier Score (IBS) is mathematically represented as follows:

$$IBS = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=t_i}^{T} \left[\hat{S}_i(t) - O_i(t) \right]^2$$

where:
- N is the total number of subjects or observations in the dataset,
- t_i is the observed event time for the i-th subject,
- T is the maximum time horizon for evaluation,
- $\hat{S}_i(t)$ is the predicted survival probability of the i-th subject at time t,
- $O_i(t)$ is the observed event indicator for the i-th subject at time t, where $O_i(t) = 1$ if an event occurred for the i-th subject at or before time t, and $O_i(t) = 0$ otherwise.

The IBS calculates the squared differences between the predicted survival probabilities and the observed event indicators for all subjects over the specified time horizon and then averages these differences to obtain the final IBS value.

The IBS is an important tool for model evaluation in survival analysis, as it provides a comprehensive assessment of a model's predictive accuracy across different time points. It enables researchers to make informed decisions when selecting the most suitable survival model for their specific application.

Chapter 3

Case Study and Analysis

The data preprocessing for the presented results was conducted using Python version 3.8. Subsequently, the survival analysis was performed using R version 4.3.1., mainly using the packages mlr3 (Lang et al., 2019), mlr3proba (Sonabend et al., 2021) and survival models and plots were created using ggplot2. Additional details and the code are available in the Appendix.

3.1 Dataset Description

The dataset used in this analysis comprises data from 6,897 firms in Italy, followed up for a period of 19 years. Among these firms, 3,462 became inactive due to bankruptcy or other legal procedures. The dataset consists of 5 features representing the firms' basic information and 12 features representing their financial information. The basic information consists of the following features:

- **Time**: This variable represents the time in years until closure. A value of 19 indicates that the firm is still active at the end of the study.
- **Age**: This variable indicates the duration of time the firm has been in operation at the beginning of the study.
- **Status**: This binary variable denotes the status of the firm. A value of 0 indicates an active firm at the end of the study, while a value of 1 indicates an inactive firm.
- **Region**: This variable represents the region in Italy where the firm is located, categorized into five regions: Northwest, South, Islands, Northeast, and Central.
- **Legal_form**: This variable represents the legal form of the firm, categorized as Partnership, Private Limited Company, or Public Limited Company.

The financial information consists of 12 financial ratios known or expected to be related to default events. These ratios are as follows:

- **ln(EBITDA)**: The natural logarithm of Earnings Before Interest, Taxes, Depreciation, and Amortization (EBITDA), which measures a company's operating performance.
- **Operating Revenue/Inventories**: The ratio of operating revenue to inventories, indicating the efficiency of inventory management and the relationship between sales revenue and inventory levels.
- **(Creditors/Operating revenue) * 360**: The product of the ratio of creditors to operating revenue and 360, approximating the number of days of operating revenue owed to creditors.
- **Current Assets/Current Liabilities**: The ratio of current assets to current liabilities, assessing the liquidity and short-term financial strength of a company.
- **Debtors/Operating Revenue**: The ratio of debtors (accounts receivable) to operating revenue, measuring the efficiency of credit management and the relationship between sales revenue and outstanding receivables.
- **Shareholders Funds/Total Assets**: The ratio of shareholders' funds (equity) to total assets, indicating the proportion of assets financed by shareholders' investments.
- **Non-Current Liabilities/Total Assets**: The ratio of non-current liabilities (long-term debt) to total assets, reflecting the portion of assets financed by long-term debt.
- **(Long Term Debt + Loans)/Total Assets**: The ratio of long-term debt and loans to total assets, indicating the proportion of assets financed by long-term debt and loans.
- **Profit (Loss) for period/Shareholders Funds**: The ratio of profit or loss for a specific period to shareholders' funds (equity), measuring the return on shareholders' investments.
- **EBIT/Shareholders Funds**: The ratio of Earnings Before Interest and Taxes (EBIT) to shareholders' funds (equity), assessing the profitability generated from shareholders' investments.
- **Profit (Loss) for period/Operating Revenue**: The ratio of profit or loss for a specific period to operating revenue, measuring the profitability relative to revenue.
- **Operating Revenue/Total Assets**: The ratio of operating revenue to total assets, assessing asset utilization efficiency and the relationship between revenue generation and the asset base.
- **Debtors/Current Assets**: The ratio of debtors (accounts receivable) to current assets, measuring the proportion of current assets tied up in outstanding receivables.

3.2 Data Preprocessing

Data preprocessing is a technique aimed at transforming raw data into a usable format (Famili et al., 1997). Real-world financial data often suffer from issues such as incompleteness, inconsistencies, and errors (Magdon-Ismail et al., 1998). Furthermore, the accuracy of machine learning algorithms can be influenced by the underlying data distribution. To address these challenges and enhance the accuracy of machine learning models, various preprocessing techniques are applied to make the data meaningful.

3.2.1 Normalization and Skewness Reduction

Addressing skewness is critical in machine learning for several reasons. Firstly, skewness significantly affects performance metrics, leading to distorted results and potentially misleading performance estimates. Secondly, skewness introduces biases in evaluating classifier performance, making it challenging to assess the true effectiveness of the model. Lastly, comparing classifiers across datasets is confounded by skewness, as metric values can vary solely due to differences in skewness rather than actual differences in performance. Therefore, mitigating skewness is crucial to ensure reliable and accurate machine learning analyses and comparisons (Jeni et al., 2013).

The dataset was initially split into a training set and a test set. The training set accounted for 67% of the data, resulting in a total of $4,111$ observations. Conversely, the test set constituted 33% of the dataset, encompassing a total of $2,026$ observations.

In order to reduce skewness in the train and test data, two common transformations, namely Box-Cox transformation (Box and Cox, 1964) and Yeo-Johnson Power Transformation (Cook and Weisberg, 1999; Yeo, 2000), are employed. These techniques are applied to address the skewness in different types of variables.

For positive-skewed variables with positive values, a logarithmic transformation is applied. This helps to normalize the distribution and reduce the skewness of the data. For negative-skewed variables with positive values, the Box-Cox transformation is implemented. This transformation adjusts the variable's values to achieve a more symmetric distribution and reduce skewness. In the case of negative-skewed variables with negative values and positive-skewed variables with negative values, the Yeo-Johnson transformation is utilized. This transformation accommodates both positive and negative values and ensures a more balanced distribution, effectively reducing skewness.

By employing these transformations, the data is modified to achieve reduced skewness, enabling a more accurate analysis and interpretation of the results. Figures 3.1 and 3.2 illustrate the variables before and after the transformation, respectively, applied to the training data.

3.2.2 Multicollinearity Detection

When a variable in a data frame contains significant information about another independent feature, it is considered to exhibit multicollinearity (Vatcheva and Lee, 2016). Addressing multicollinearity is essential to ensure reliable estimation of coefficients in a regression model (Vatcheva and Lee, 2016). In this study, the correlations between numerical features were evaluated using Pearson's pairwise method, and the results were visualized in Figure 3.3. From the correlation matrix, several pairs of highly correlated variables can be observed. Firstly, the variables *Shareholders Funds/ Total Assets* and *Current Assets/Current Liabilities* exhibit a strong positive correlation of 0.657. This suggests that there is a significant relationship between the ratio of shareholders' funds to total assets and the ratio of current assets to current liabilities. As the ratio of current assets to current liabilities increases, the ratio of shareholders' funds to total assets tends to increase. Additionally, the variables *Profit (Loss) for period / Operating Revenue* and *ln(EBITDA)* show a strong positive correlation of 0.633. This indicates that there is a notable association between the profitability of a company in terms of profit or loss for a given period and the logarithm of EBITDA. Higher profitability is associated with higher EBITDA values. Moreover, the variables *Profit (Loss) for period / Operating Revenue* and *Profit (Loss) for period / Shareholders Funds* also exhibit a strong positive correlation of 0.633. This implies that there is a significant relationship between the profitability of a company in terms of profit or loss for a given period and the ratio of profit or loss to shareholders' funds. Furthermore, the variables *Debtors/Current Assets* and *Debtors/Operating Revenue* show a strong positive correlation of 0.735. This suggests a strong relationship between the ratio of debtors to current assets and the ratio of debtors to operating revenue. As the ratio of debtors to current assets increases, the ratio of debtors to operating revenue also tends to increase. Finally, the variables *Status* and *Time* exhibit a strong negative correlation of -0.792. This indicates a significant inverse relationship between the status of a company and the time at which the data was collected. As time progresses, the status of the company tends to decline.

These findings provide valuable insights into the relationships between the correlated variables. However, a strong association between variables does not necessarily imply redundancy or the need to remove variables from the dataset.

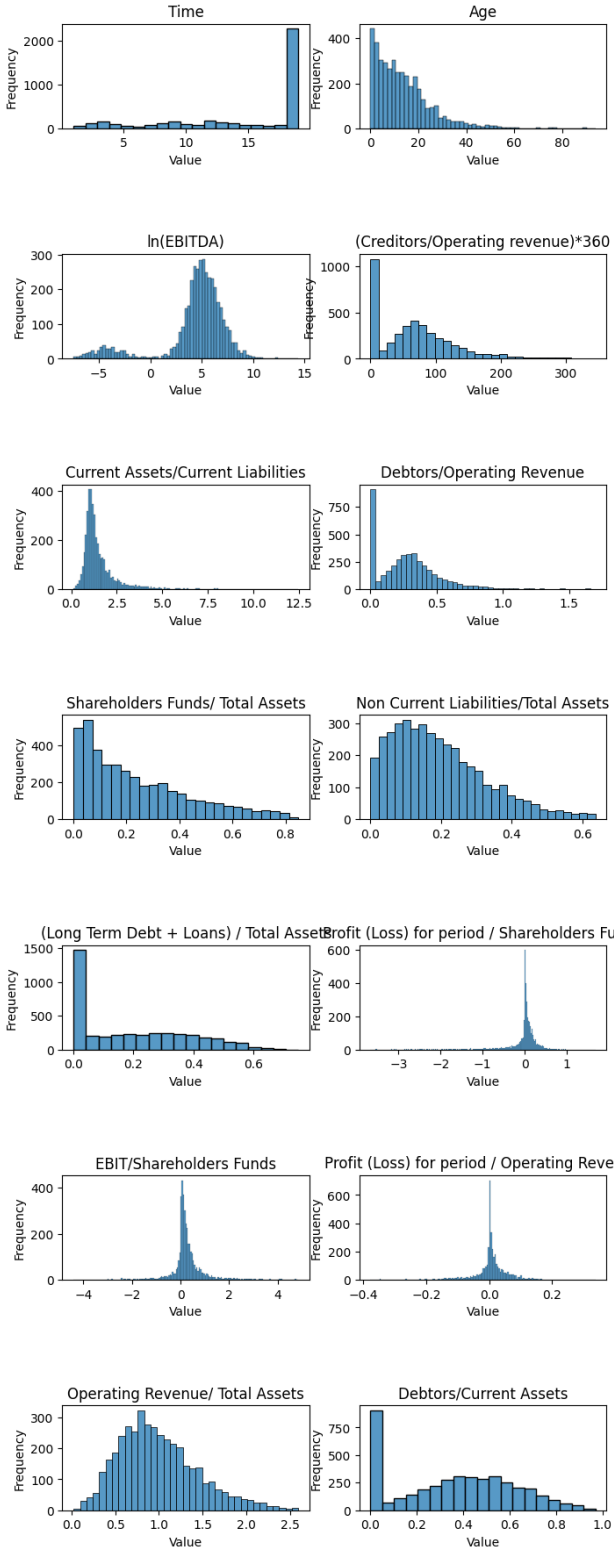


Figure 3.1: Training data before transformation

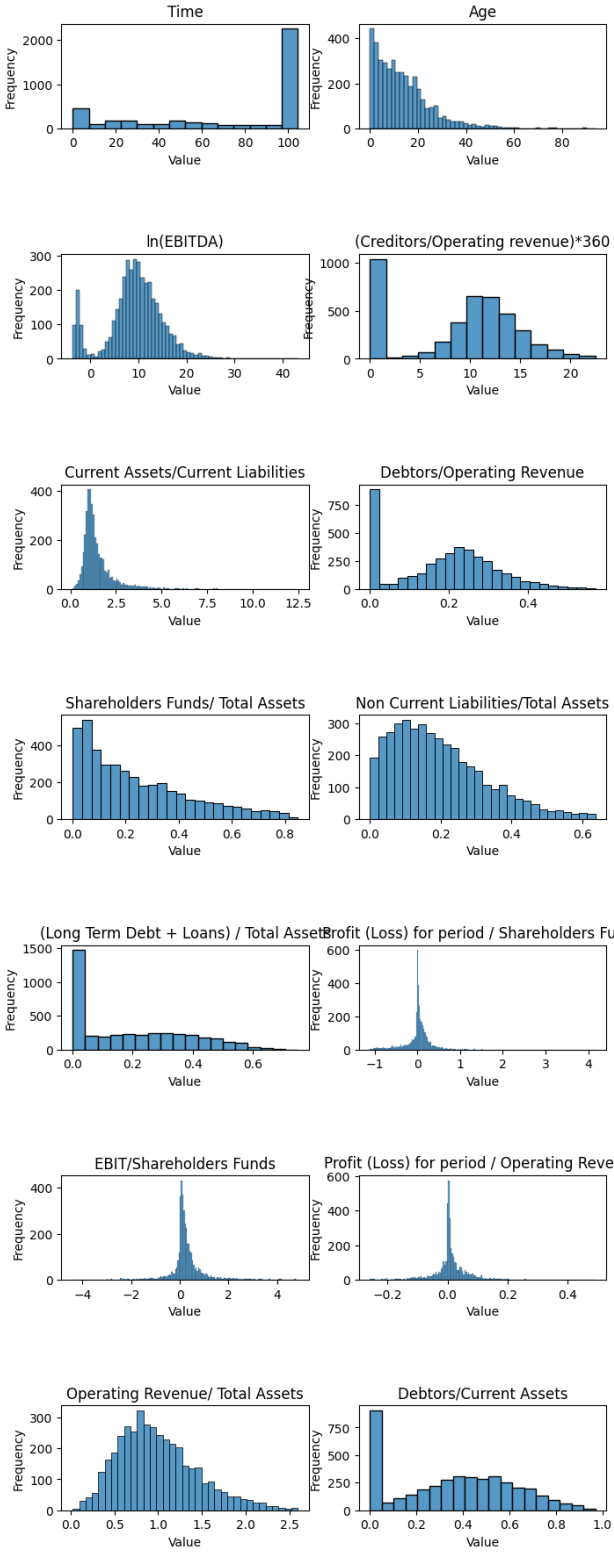


Figure 3.2: Training data after transformation

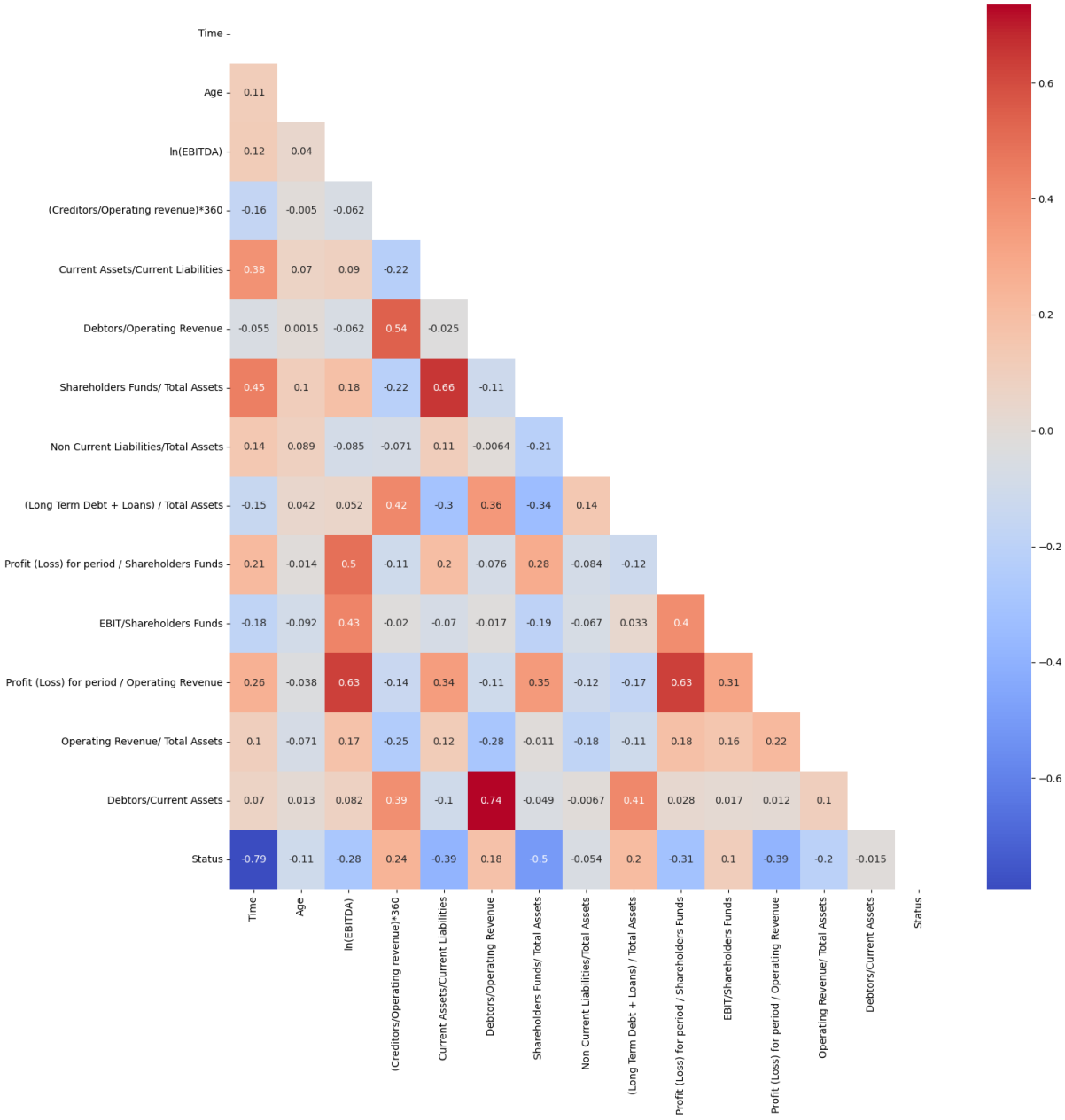


Figure 3.3: Pearson's correlations between numerical covariates

3.3 Cox Proportional Hazards Model Application

The Cox Proportional Hazards Initial Model Results are presented in Table 3.1. This table provides information about the coefficients, hazards ratios, standard errors, z-scores, and p-values for each variable included in the model.

The first variable in the model is "Age". The p-value for this variable is small (0.0041) which indicates strong evidence of a significant association between Age and the hazard. Although the hazard ratio is 0.9963 with a 95% confidence interval of 0.9920 to 0.99853 which indicates that holding the other covariates constant an additional year of Age is associated with the risk of "death".

Similarly, the second variable "ln(EBITDA)," which represents the natural logarithm of the Earnings Before Interest, Taxes, Depreciation, and Amortization has a hazard ratio of 0.9487. This indicates that a one-unit increase in the logarithm of EBITDA is associated with a 5.2% decrease in the hazard, holding other variables constant. The p-value for this variable is extremely small ($< 2e-16$), suggesting strong evidence of a significant association between ln(EBITDA) and the hazard.

In addition, the variable "Current.Assets/Current.Liabilities." has a coefficient of 0.02361, implying that a one-unit increase in the ratio of current assets to current liabilities leads to a 2.40% increase in the hazard, assuming other variables are held constant. The p-value is very small ($6.74e-14$), indicating a significant association between this variable and the hazard.

Furthermore, "Shareholders.Funds/Total.Assets" has a coefficient of -4.602. The hazard ratio is 0.0100, indicating that a one-unit increase in the ratio of shareholders' funds to total assets is associated with a 99.00% decrease in the hazard. The p-value is extremely small ($< 2e-16$), indicating a highly significant association between this variable and the hazard.

Also, the variable "Non.Current.Liabilities/Total.Assets" has a coefficient of -2.584, and the hazard ratio is 0.0755. This suggests that a one-unit increase in the ratio of non-current liabilities to total assets is associated with a 92.45% decrease in the hazard. The p-value is extremely small ($< 2e-16$), indicating a highly significant association.

Similarly, the variable "(Long.Term.Debt + Loans)/Total.Assets" has a coefficient of 0.6326, indicating that a higher ratio of long-term debt and loans to total assets is associated with a 88.20% increase in the hazard. The p-value is small ($5.11e-10$), indicating a significant association.

Furthermore, the variable "Profit.Loss.for.period/Shareholders.Funds" has a coefficient of -0.07502, implying that a one-unit increase in the ratio of profit/loss for the period to shareholders' funds is associated with a 7.23% decrease in the hazard. The p-value is highly significant ($1.28e-08$).

Additionally, "EBIT/Shareholders.Funds" has a coefficient of 0.1089. This suggests that a higher ratio of earnings before interest and taxes to shareholders' funds is associated with an 11.50% increase in the hazard. The p-value is extremely small ($< 2e-16$), indicating a highly significant association.

Moreover, "Operating.Revenue/Total.Assets" has a coefficient of -0.3314. This implies that a higher ratio of operating revenue to total assets is associated with a 28.21% decrease in the hazard. The hazard ratio is 0.7179, and the p-value is extremely small ($2.29e-15$), indicating a highly significant association.

Also, the variable "Debtors/Current.Assets" has a coefficient of -0.5208, suggesting that a higher ratio of debtors to current assets is associated with a 40.60% decrease in the hazard. The hazard ratio is 0.5940, and the p-value is highly significant ($5.25e-10$).

Next, we have the variables representing different regions. "Region2" has a coefficient of 0.2796, indicating that being in Region2 is associated with a 32.30% increase in the hazard. The hazard ratio is 1.3230, and the p-value is highly significant ($6.62e-07$). Similarly, "Region3" has a coefficient of 0.2339, suggesting a 26.40% increase in the hazard compared to the baseline region. The hazard ratio is 1.2640, and the p-value is significant (0.00981). On the other hand, "Region4" has a coefficient of -0.08243, implying a 7.91% decrease in the hazard compared to the baseline region. The hazard ratio is 0.9209, and the p-value suggests a marginal association (0.05512). On the contrary, when examining "Region5", we observe a p-value of 0.38883, indicating a lack of significant association. The hazard ratio stands at 0.9583, and the corresponding 95% confidence interval ranges from 0.86 to 1.05. Given that the confidence interval encompasses the value 1, these findings suggest that "Region5" does not exhibit substantial differences from the baseline region, "Region1".

Lastly, we have the variables representing different legal forms. "Legal_form2" has a coefficient of 0.5717, indicating that having Legal_form2 is associated with a 77.10% increase in the hazard compared to the baseline legal form. The hazard ratio is 1.7710, and the p-value is highly significant ($5.77e-06$). Similarly, "Legal_form3" has a coefficient of 0.6878, suggesting a 98.90% increase in the hazard compared to the baseline legal form. The hazard ratio is 1.9890, and the p-value is highly significant ($4.13e-07$).

However, the variable "Debtors/Operating.Revenue" has a hazard of 1.0710 with a 95% confidence interval of 0.99 to 1.15 which indicates that the variable makes a smaller contribution to the difference in the hazard ratio. Additionally, the p-value for this variable (0.06143) suggests that the association is not statistically significant at conventional levels (p > 0.05).

Similarly, the variable, "(Creditors.Operating.revenue/360)," has a coefficient of 3.736e-05, which is close to zero. The hazard ratio is 1.0000, indicating no substantial effect on the hazard. The p-value for this variable is 0.82848, suggesting that it is not significantly associated with the hazard.

Lastly, the variable "Profit.Loss.for.period/Operating.Revenue" has a coefficient of -0.2169, indicating that a higher ratio of profit/loss for the period to operating revenue is associated with a 19.50% decrease in the hazard. However, the p-value (0.10631) suggests that the association is not statistically significant at conventional levels.

The table presents additional statistics to evaluate the overall model performance. The Concordance value of 0.772 indicates a moderate level of predictive accuracy, with higher values indicating better predictive performance. The Likelihood ratio test, Wald test, and Score (logrank) test all suggest that the overall model is highly significant

Table 3.1: Cox Proportional Hazards Initial Model Results

Variable	Coef	Exp(coef)	SE(coef)	z	Pr(> |z|)	lower 0.95	upper 0.95
Age	-0.004711	0.9953	0.001654	-2.847	0.00441	0.992078	0.99853
ln(EBITDA)	-0.05266	0.9487	0.005715	-9.213	$< 2e-16$	0.938137	0.95939
(Creditors.Operating.revenue/360)	3.736e-05	1.0000	0.0001725	0.217	0.82848	0.999699	1.00038
Current.Assets/Current.Liabilities	0.02361	1.0240	0.003152	7.493	$6.74e-14$	1.017590	1.03024
Debtors/Operating.Revenue	0.06890	1.0710	0.03684	1.870	0.06143	0.996706	1.15155
Shareholders.Funds/Total.Assets	-4.602	0.0100	0.1293	-35.588	$< 2e-16$	0.007788	0.01293
Non.Current.Liabilities/Total.Assets	-2.584	0.0755	0.1255	-20.592	$< 2e-16$	0.059019	0.09652
(Long.Term.Debt + Loans)/Total.Assets	0.6326	1.8820	0.1018	6.216	$5.11e-10$	1.542028	2.29797
Profit.Loss.for.period/Shareholders.Funds	-0.07502	0.9277	0.01319	-5.689	$1.28e-08$	0.904050	0.95201
EBIT/Shareholders.Funds	0.1089	1.1150	0.008713	12.494	$< 2e-16$	1.096129	1.13421
Profit.Loss.for.period/Operating.Revenue	-0.2169	0.8050	0.1343	-1.615	0.10631	0.618736	1.04742
Operating.Revenue/Total.Assets	-0.3314	0.7179	0.04182	-7.924	$2.29e-15$	0.661427	0.77925
Debtors/Current.Assets	-0.5208	0.5940	0.08385	-6.211	$5.25e-10$	0.504020	0.70015
Region2	0.2796	1.3230	0.05624	4.972	$6.62e-07$	1.184593	1.47675
Region3	0.2339	1.2640	0.09058	2.583	0.00981	1.058010	1.50902
Region4	-0.08243	0.9209	0.04298	-1.918	0.05512	0.846477	1.00181
Region5	-0.04257	0.9583	0.04940	-0.862	0.38883	0.869897	1.05575
Legal_form2	0.5717	1.7710	0.1261	4.535	$5.77e-06$	1.383446	2.26769
Legal_form3	0.6878	1.9890	0.1359	5.063	$4.13e-07$	1.524344	2.59637
Concordance	0.772 (se = 0.004)						
Likelihood ratio test	3214 on 19 df, $p < 2e-16$						
Wald test	3019 on 19 df, $p < 2e-16$						
Score (logrank) test	3625 on 19 df, $p < 2e-16$						

($p < 2e-16$), indicating that the combined effect of all the variables is significantly associated with the hazard.

In summary, the Cox proportional hazards model results indicate significant associations between several variables and the hazard of the event. Age, ln(EBITDA), Current.Assets/Current.Liabilities, Shareholders.Funds/Total.Assets, (Long.Term.Debt + Loans)/Total.Assets, Non.Current.Liabilities/Total.Assets, EBIT/Shareholders.Funds, Profit.Loss.for.period/Shareholders.Funds, Operating.Revenue/Total.Assets, Debtors/Current.Assets, Region2, Region3, Legal_form2, and Legal_form3 all show significant relationships with the hazard. These findings provide valuable insights into the factors influencing the risk of the business' termination.

Afterwards, we conducted an assessment of the proportional hazards (PH) assumption as an essential step in the survival analysis. The PH assumption posits that the hazard ratios or coefficients of covariates remain constant over time, which is a fundamental assumption in Cox proportional hazards regression. To evaluate the validity of this assumption, we performed a series of statistical tests on the included predictor variables. The results, summarized in Table 3.2 below, provide valuable insights into whether the PH assumption holds for each covariate. It is noteworthy that several covariates, *ln(EBITDA), (Creditors.Operating.revenue)/360, Debtors/Operating.Revenue, Non.Current.Liabilities/Total.Assets, (Long.Term.Deb+Loans)/Total.Assets, Profit Loss for period/Shareholders Funds, EBIT/Shareholders Funds, Profit Loss for period/Operating Revenue, Operating Revenue/Total Assets, Debtors/Current Assets, Region, Legal_form*, and *GLOBAL* all exhibit significant deviations from the PH assumption (p-values < 0.05). Consequently, these variables may require special consideration in our survival analysis to account for the time-dependent effects properly. For the complete

set of results, please refer to Table 3.2.

Table 3.2: Results of Proportional Hazards (PH) Assumption Tests

Variable	χ^2	df	p-value
Age	0.0526	1	0.8186
ln.EBITDA	262.6087	1	$< 2 \times 10^{-16}$
X.Creditors.Operating.revenue..360	98.8796	1	$< 2 \times 10^{-16}$
Current.Assets.Current.Liabilities	1.1110	1	0.2919
Debtors.Operating.Revenue	100.5785	1	$< 2 \times 10^{-16}$
Shareholders.Funds..Total.Assets	1.8080	1	0.1788
Non.Current.Liabilities.Total.Assets	89.2466	1	$< 2 \times 10^{-16}$
X.Long.Term.Debt...Loans....Total.Assets	34.7467	1	3.8×10^{-09}
Profit..Loss..for.period...Shareholders.Funds	102.3749	1	$< 2 \times 10^{-16}$
EBIT.Shareholders.Funds	31.3627	1	2.1×10^{-08}
Profit..Loss..for.period...Operating.Revenue	83.8785	1	$< 2 \times 10^{-16}$
Operating.Revenue..Total.Assets	283.2052	1	$< 2 \times 10^{-16}$
Debtors.Current.Assets	90.8881	1	$< 2 \times 10^{-16}$
Region	15.0765	4	0.0045
Legal_form	66.0826	2	4.5×10^{-15}
GLOBAL	856.7996	19	$< 2 \times 10^{-16}$

In Table 3.2, χ^2 represents the chi-squared statistic, df denotes degrees of freedom, and p-value indicates the level of significance for each covariate's violation of the PH assumption. These findings could lead to selecting appropriate techniques, such as stratification, to account for the non-proportional hazards and ensure the robustness of our survival analysis.

3.4 DeepSurv Application

Two models of DeepSurv were employed in the analysis. In the first model, hyperparameter optimization was conducted to enhance the model's performance. The AutoTuner object was utilized to search through a predefined search space, aiming to identify the optimal hyperparameter configurations. The concordance index (C-index) was selected as the performance metric for optimization.

In the second model of DeepSurv, feature selection was employed to identify the most informative predictors for survival analysis. Sequential feature selection was utilized, using the "surv.cindex" performance metric to determine the subset of features that exhibited the best predictive performance.

Hyperarameter optimization of DeepSurv

In this section, the hyperparameter optimization process used to fine-tune DeepSurv model's performance is described. The goal is to optimize the model's hyperparameters to achieve the highest survival concordance index (C-index) on the given dataset. The optimization process employs a grid search with a resolution of 20 and a batch size of 50, combined with cross-validation and a termination criterion. The hyperparameter optimization process aims to find the best combination of hyperparameters that maximizes the survival C-index. The hyperparameters considered for optimization are as follows:

- **Decay:** A numeric hyperparameter controlling weight decay regularization in the neural network.
- **Learning Rate:** A numeric hyperparameter representing the step size for adjusting model weights during training.
- **Dropout:** A numeric hyperparameter controlling the dropout rate, a regularization technique to prevent overfitting.
- **Nodes:** A numeric hyperparameter indicating the number of nodes (neurons) in each hidden layer of the neural network.
- **Layers:** A numeric hyperparameter representing the total number of hidden layers in the neural network.
- **Optimizer:** A categorical hyperparameter with two options (optimizer factors) - SGD and Adam.
- **Activation Function:** A categorical hyperparameter with two options (activation function factors) - ReLU and Selu.

The hyperparameter optimization process proceeds as follows:

1. **Grid Search:** - For each numeric hyperparameter (Decay, Learning Rate, Dropout, Nodes, Layers), a predefined range is specified, and the grid search with a resolution of 20 samples twenty equally spaced values within each range. - For the factor hyperparameters (Optimizer, Activation Function), all possible combinations of the two options are considered (SGD vs. Adam and ReLU vs. Selu).

2. **Cross-Validation:** - The grid search is combined with a cross-validation resampling strategy, where the data is split into 5 folds. - The model is trained and evaluated 5 times (one for each fold) using different subsets of the data for training and validation.
3. **Tuning and Termination:** - The performance measure, survival C-index, is calculated for each combination of hyperparameters across the 5 cross-validation runs. - The termination criterion of stagnation batch is applied. The optimization process keeps track of the last 7 consecutive batches of hyperparameter combinations and checks whether there is a significant improvement in the C-index, i.e., if the improvement is above the specified threshold of 1e-4. If there is no significant improvement, the tuning process stops.
4. **Outer Resampling:** - The entire hyperparameter optimization process is repeated 10 times using a 10-fold cross-validation resampling strategy. - After each iteration of the outer resampling, the best hyperparameters with the highest C-index are recorded.
5. **Final Model Selection:** - For each iteration of the outer resampling, we obtain one model from inner resampling (the best one), each with its optimal set of hyperparameters and this model will be evaluated on the test set for this outer fold.
6. **Aggregation:** - Take the sample mean of the ten C-index values for an unbiased performance estimate.

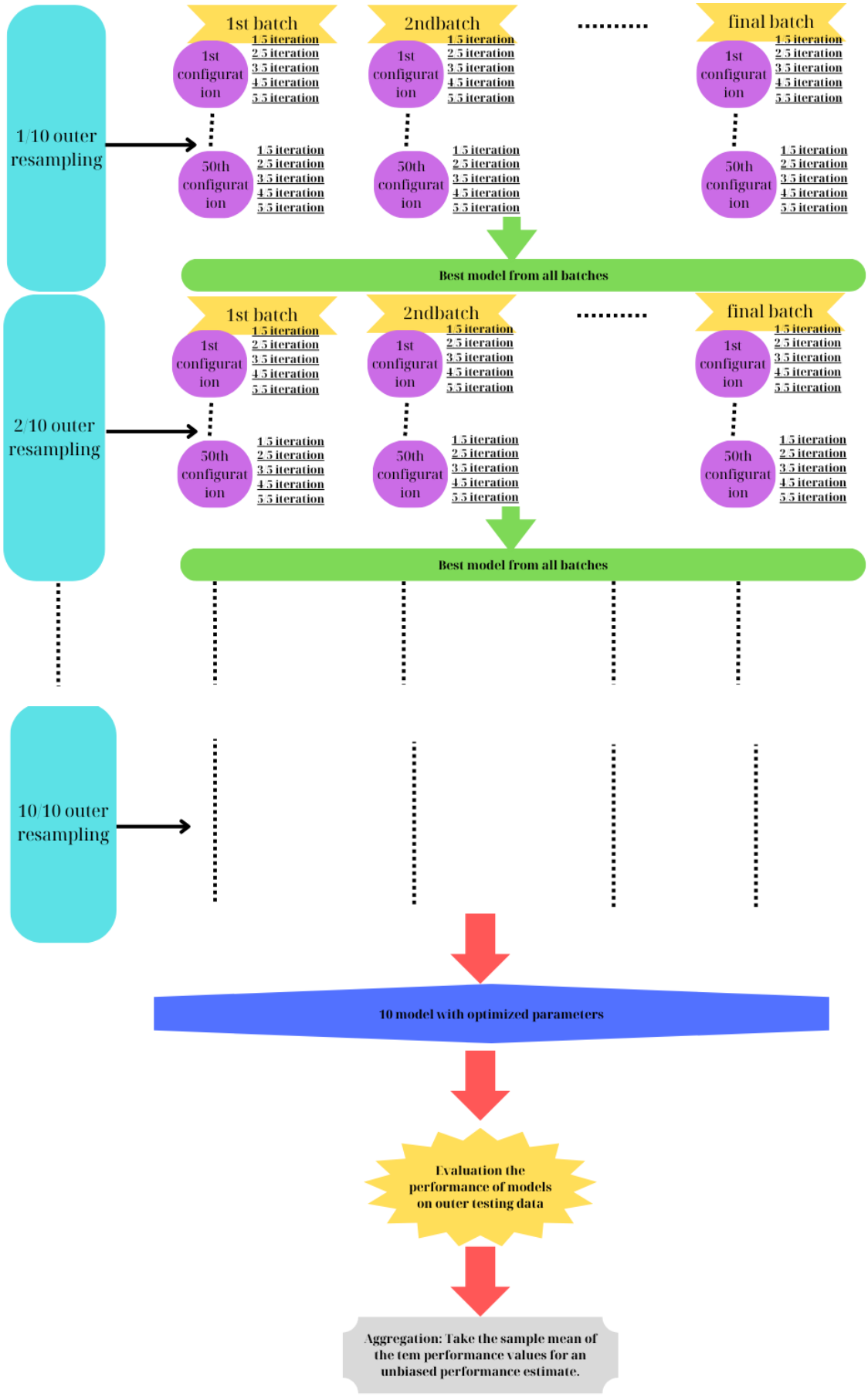
In summary, the hyperparameter optimization process which is presented in Figure 3.4, utilizes a grid search, cross-validation, and a termination criterion to efficiently explore the hyperparameter space and find the optimal configuration for the DeepSurv model. The outer resampling with 10-fold cross-validation ensures the robustness of the model's performance evaluation, and the final selected model represents the best-performing configuration on the given dataset.

Table 3.3 presents the 10 models that achieved the highest C-index scores with their optimal set of hyperparameters and the evaluation of the performances of models on test data. It is clear that there is no significant evidence of a consistent trend in the C-index with respect to these hyperparameters. The C-index values appear to vary without displaying a discernible pattern as the hyperparameter values change.

Table 3.3: DeepSurv models with optimal combination of hyperparameters

Weight Decay	Learning Rate	Optimizer	Dropout	Nodes/Layer	Layers	Activation	C-Index of Train Data	C-Index of Test Data
0.4736842	0.4210526	sgd	0.5789474	5	10	selu	0.7917773	0.7665178
0.4736842	0.4210526	adam	0.5789474	1	3	relu	0.7919148	0.7755227
0.4736842	0.7368421	sgd	0.8421053	3	14	relu	0.7871779	0.8047545
0.2631579	0.3157895	sgd	0.9473684	4	31	selu	0.7885051	0.7932329
0.2368421	0.8421053	adam	0	8	44	selu	0.7882624	0.7913179
0.1315789	0.3157895	sgd	0.5263158	7	42	relu	0.7862546	0.7867108
0.02631579	0.6842105	adam	0.8947368	10	1	selu	0.7885757	0.7685149
0.07894737	0.1578947	sgd	0.05263158	4	26	selu	0.7889386	0.8034955
0.5	0.5789474	adam	0.7894737	11	1	relu	0.7897789	0.7695737
0.2368421	0.9473684	sgd	0.8947368	3	10	selu	0.7881934	0.7810905

Figure 3.4: An illustration of nested resampling for hyperparameter optimization



Feature Selection using DeepSurv

The feature selection process is employed to identify the most informative features that contribute significantly to the predictive performance of the DeepSurv model. In this study,a sequential feature selection technique is employed, characterized by its iterative nature. This method aligns with the wrapper feature selection approach, given that it relies on a learner (induction algorithm) to steer the process of feature selection. Furthermore, it's worth noting that this feature selection method operates in a supervised manner. The process starts with an empty feature set and incrementally adds features based on their impact on the performance measure, specifically the C-index, which evaluates the predictive accuracy of the survival model. The goal is to identify the optimal subset of features that leads to the highest improvement in the C-index.

To evaluate the performance of the feature selection process and to mitigate the risk of overfitting, a nested resampling cross-validation scheme is implemented. The entire dataset is divided into ten outer folds, each representing a distinct data partition. For each outer fold, a two-level cross-validation process is performed. Within each outer fold, the data is further divided into five inner folds. The feature selection method is then applied on each inner fold separately to select the most informative features.

The performance of the feature selection process is evaluated using the C-index calculated on the inner validation set. The process continues iteratively until a termination criterion is met. In this study, the termination criterion is set to "stagnation_batch," which halts the process if no significant improvement in the C-index is observed for seven consecutive iterations. This prevents unnecessary feature additions that may not contribute to better predictive performance.

The combination of the sequential feature selection method and nested resampling cross-validation ensures a systematic and rigorous approach to identify relevant features and assess their impact on the DeepSurv model's performance. It helps to avoid overfitting and enhances the generalizability of the selected features, leading to a more robust and accurate survival prediction model. Tables 3.4 and 3.5 provide the performance metrics for the DeepSurv model when trained on selected features. The C-Index and C-Index of Test Data are used as evaluation metrics. The results of feature selection provide valuable insights into the most important variables for predicting survival times using the DeepSurv model. We can observe that the following features consistently appear in the selected sets:

Debtors/Operating Revenue: This feature appears in both parts, indicating its high importance in predicting survival times. It suggests that the ratio of debtors to operating revenue has a significant impact on the risk of events being observed.

EBIT/Shareholders Funds: Another feature present in both parts, indicating its relevance. The ratio of earnings before interest and taxes to shareholders' funds appears to be a critical predictor of survival times.

Non Current Liabilities/Total Assets: This feature is consistently selected, suggesting that the proportion of non-current liabilities to total assets is a strong indicator of

Table 3.4: Results of Feature Selection (Part 1)

Features	**C-Index**	**C-Index of Test Data**
Debtors/Operating Revenue EBIT/Shareholders Funds Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Profit Loss for period/Shareholders Funds Region 2 Shareholders Funds/Total Assets (Long Term Debt + Loans)/Total Assets ln(EBITDA)	0.7959486	0.7729620
Debtors/Current Assets Debtors/Operating Revenue EBIT/Shareholders Funds Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Profit Loss for period/Shareholders Funds Region 2 Shareholders Funds/Total Assets ln(EBITDA)	0.7928629	0.7784615
Debtors/Current Assets Debtors/Operating Revenue EBIT/Shareholders Funds Legal_Form 3 Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Region 4 Shareholders Funds/Total Assets (Long Term Debt + Loans)/Total Assets ln(EBITDA)	0.7886929	0.7862141
EBIT/Shareholders Funds Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Profit Loss for period/Shareholders Funds Shareholders Funds/Total Assets ln(EBITDA)	0.7906242	0.8002924
EBIT/Shareholders Funds Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Profit Loss for period/Shareholders Funds Shareholders Funds/Total Assets ln(EBITDA)	0.7915772	0.7996605

Table 3.5: Results of Feature Selection (Part 2)

Features	**C-Index**	**C-Index of Test Data**
Debtors/Current Assets EBIT/Shareholders Funds Legal_Form 3 Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Profit Loss for period/Shareholders Funds Region 2 Shareholders Funds/Total Assets ln(EBITDA)	0.7935748	0.7923255
EBIT/Shareholders Funds Legal_Form 3 Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Shareholders Funds/Total Assets ln(EBITDA)	0.7918338	0.7888424
Debtors/Current Assets Debtors/Operating Revenue EBIT/Shareholders Funds Legal_Form 2 Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Profit Loss for period/Shareholders Funds Region 2 Region 3 Shareholders Funds/Total Assets (Long Term Debt + Loans)/Total Assets ln(EBITDA)	0.7918511	0.7940536
Debtors/Operating Revenue EBIT/Shareholders Funds Legal_Form 3 Non Current Liabilities/Total Assets Operating Revenue/Total Assets Profit Loss for period/Operating Revenue Profit Loss for period/Shareholders Funds Region 2 Region 3 Shareholders Funds/Total Assets (Creditors/Operating Revenue) * 360 (Long Term Debt + Loans)/Total Assets ln(EBITDA)	0.7926354	0.8018290

risk in the context of survival analysis.

ln(EBITDA): The natural logarithm of EBITDA appears in both parts, emphasizing its importance as a predictor. This implies that the logarithm of earnings before interest, taxes, depreciation, and amortization has a significant influence on the model's performance.

Profit Loss for period/Operating Revenue and Profit Loss for period/Shareholders Funds: Both of these profitability-related features consistently appear, indicating their relevance in predicting event risks.

3.5 Comparison of 3 Models

To compare the performance of three models, the Survival Cox Hazard model was also evaluated using cross-validation with 10 folds, along with the DeepSurv model in two variations: one with feature selection and another using Hyperparameter Optimization (HPO). The metric used to assess model performance was the C-Index. The results presented in Figure 3.5 indicated that the DeepSurv model with feature selection consistently achieved the highest C-Index values in the majority of the cross-validation folds. This outcome suggests that the inclusion of feature selection significantly contributed to enhancing the model's predictive performance, as it likely helped in identifying the most relevant and informative features for survival analysis.

Conversely, the Cox Proportional Hazards model generally yielded lower C-Index values when compared to the DeepSurv models. This observation implies that the DeepSurv models, leveraging deep learning techniques, possess the ability to capture more intricate patterns within the data. Consequently, they outperformed the traditional Cox Proportional Hazards model in the context of survival analysis tasks, indicating their potential to better handle complex relationships and dependencies present in survival data.

Table 3.6: Model Comparison Aggregating Results

#	Model	Resampling Method	Iters	C-Index	Integrated Graf Score
1	DeepSurv (HPO)	CV	10	0.7841	0.1238
2	Proportional Cox Hazard	CV	10	0.7708	0.1319
3	DeepSurv (Feature Selection)	CV	10	0.7896	0.1323

Table 3.6 presents the performance evaluation of the three models using two metrics: C-Index and Integrated Graf Score. The models were subjected to cross-validation

with 10 folds (CV).

DeepSurv (HPO): This model, employing Hyperparameter Optimization (HPO), achieved a mean C-Index of approximately 0.7841 and a mean Integrated Graf Score of around 0.1238. The higher C-Index suggests that the model demonstrated reasonably good concordance between predicted and observed survival times.

Proportional Cox Hazard: This model, based on the traditional Cox Proportional Hazards approach, obtained a mean C-Index of approximately 0.7708 and a mean Integrated Graf Score of about 0.1319. Although its C-Index is slightly lower than the DeepSurv model with HPO, it still indicates a reasonable performance in survival analysis tasks.

DeepSurv (Feature Selection): The DeepSurv model with feature selection exhibited the best performance among all the models, achieving a mean C-Index of approximately 0.7896 and a mean Integrated Graf Score of around 0.1323. These results suggest that the inclusion of feature selection improved the model's ability to predict survival times, making it the most effective model in this comparison.

In conclusion, the DeepSurv model with feature selection outperformed the other models, including the DeepSurv model using HPO and the traditional Proportional Cox Hazard model, based on the evaluation metrics (C-Index and Integrated Graf Score).

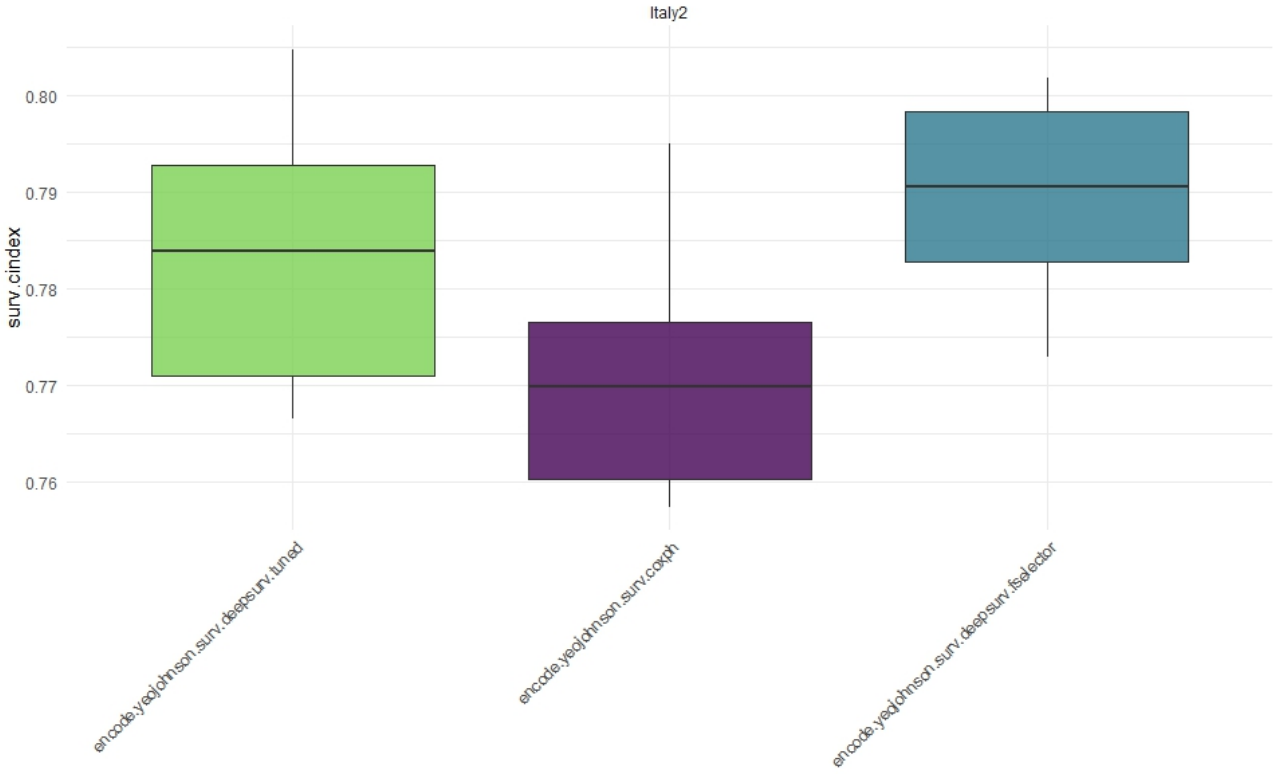


Figure 3.5: Performances of the 3 Models

In a comparative analysis between the Cox Proportional Hazards model and the

DeepSurv model with feature selection, both models revealed significant associations between various variables and the hazard of the event, providing valuable insights into the factors influencing the risk of termination for businesses in the dataset.

The Cox Proportional Hazards model identified several variables that showed significant relationships with the hazard, including Age, ln(EBITDA), Current.Assets/Current.Liabilities, Shareholders.Funds/Total.Assets, (Long.Term.Debt + Loans)/Total.Assets, Non.Current.Liabilities/Total.Assets, EBIT/Shareholders.Funds, Profit.Loss.for.period/Shareholders.Funds, Operating.Revenue/Total.Assets, Debtors/Current.Assets, Region2, Region3, Legal_form2, and Legal_form3.

On the other hand, the DeepSurv model with feature selection provided valuable insights into the most important variables for predicting survival times. Notably, Debtors/Operating Revenue, EBIT/Shareholders Funds, Non Current Liabilities/Total Assets, and ln(EBITDA) consistently appeared in the selected sets, highlighting their high importance as predictors of survival times. Additionally, Profit Loss for period/Operating Revenue and Profit Loss for period/Shareholders Funds, both related to profitability, were consistently selected, further indicating their relevance in predicting event risks.

It is noteworthy that the DeepSurv model with feature selection found a smaller set of variables compared to the Cox Proportional Hazards model. However, the selected variables were still crucial in capturing the complex patterns related to survival analysis. This suggests that the feature selection process was effective in identifying the most relevant and informative features, enhancing the predictive performance of the DeepSurv model with fewer variables.

Overall, both models yielded valuable insights into the survival patterns of companies in Italy. The Cox Proportional Hazards model revealed associations between several variables and the hazard of events, while the DeepSurv model with feature selection identified a smaller yet highly important set of predictors. The findings from both models contribute to a better understanding of the factors influencing business termination risks and highlight the potential of the DeepSurv model with feature selection for predictive modeling in survival analysis.

Chapter 4

Discussion

The aim of this thesis was to investigate and compare the predictive abilities of the standard Cox Proportional Hazards (PH) model with the artificial neural network DeepSurv. The goal was to estimate survival for companies in Italy and to investigate which features play a significant role in business failure. To achieve this, three models were created: a standard Cox Proportional Hazard model, a DeepSurv model with hyperparameter optimization, and a DeepSurv model with feature selection. Overall, all three models exhibited similar predictive performance, with the DeepSurv models outperforming the Cox PH model in terms of the Survival Concordance Index.

Previous studies by Deng et al. (2023),Tong and Zhao (2022), and Hathaway et al. (2021) have demonstrated that DeepSurv outperforms the Cox PH model in datasets related to the medical field. The difficulties faced during the tuning process of the neural network models highlight the need for experience in adjusting the tuning parameters properly. For this reason, a hyperparameter optimization of the DeepSurv model was carried out. The section concerning hyperparameter optimization of the DeepSurv model gives useful insights into the process of improving the model's performance.

The process of hyperparameter optimization followed a methodical approach, combining grid search, cross-validation, and a stopping criterion. This helped explore the hyperparameter options effectively and find the best combination that gives the highest survival concordance index (C-index).

The choice of hyperparameters for optimization, including decay, learning rate, dropout, nodes per layer, layers, optimizer, and activation function, was done thoughtfully. These settings influence how the neural network works within the DeepSurv model.

The use of grid search and cross-validation made sure that the optimization process was strong and that the chosen hyperparameters work well for new data that the model has not seen before. This leads to a better performance when predicting outcomes.

Even though a lot of effort was put into this process, it was hard to find a clear pattern between factors like weight decay, learning rate, optimizer, dropout rate, number of nodes, layers, and activation functions, and how well the c-index did. This shows that how these factors work together is complex, and some combinations might give

surprisingly good results.

Still, the process of hyperparameter optimization shows how important it is to carefully explore the options. Even though it is not always straightforward, this process is crucial for getting the best performance from the model and emphasizes its role in achieving a better model performance.

The section on feature selection using DeepSurv focuses on identifying the most informative predictors for survival analysis. The study adopted a sequential feature selection approach, aiming to construct a subset of features that maximizes the predictive accuracy of the model. The feature selection process was integrated with nested resampling, ensuring unbiased evaluation and mitigating the risk of overfitting.

The selected features provide valuable insights into the factors that contribute significantly to predicting survival times. The consistently selected features, such as Debtors/Operating Revenue, EBIT/Shareholders Funds, Non Current Liabilities/Total Assets, and ln(EBITDA), underline their importance in assessing business termination risks. These findings align with financial indicators that are often associated with the financial health and stability of companies.

The comparison of the three models (DeepSurv with HPO, Proportional Cox Hazard, and DeepSurv with feature selection) further highlights the superiority of the DeepSurv model with feature selection. This model consistently achieved the highest C-index values, indicating its ability to effectively capture survival patterns and predict event risks. The results also demonstrated the advantage of utilizing deep learning techniques, as the DeepSurv models outperformed the traditional Cox Proportional Hazards model in terms of predictive performance.

However, one limitation of using deep learning for survival analysis is the difficulty in interpreting the results. Neural networks employ intricate relationships between variables, which enhances their predictive accuracy but comes at the expense of being easy to understand. This is an aspect where deep learning falls short compared to more traditional methods. The ability to interpret the inner workings of artificial intelligence models is a current area of research that goes beyond the scope of this study.

As a result, we suggest a complementary approach where traditional techniques are combined with deep learning rather than completely replacing them. If future advancements in research enable a better understanding of the "black box" nature of neural networks, they might eventually replace existing models like the Cox Proportional Hazards model.

Definitions and Abbreviations

AAR Aalen's Additive Regression.
AI Artificial Intelligence.
ANN Artificial Neural Network.
CPH Cox Proportional Hazard Model.
HPO Hyperparameter Optimization.
IBS Integrated Brier Score.
LR Logistic Regression.
MDA Multiple discriminant analysis.
MNN Multi-task Neural Network.
NN Neural Network.
ROC Receiver Operating Characteristic.
RSF Random Survival Forest.
SBE Small Business Enterprises.
WAF Weibull Accelerated Failure Time Models.

Bibliography

Adnan Aziz, M., & Dar, H. A. (2006). Predicting corporate bankruptcy: Where we stand? *Corporate Governance: The International Journal of Business in Society*, *6*, 18–33. https://doi.org/10.1108/14720700610649436

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning : A Textbook*. Springer International Publishing.

Alaka, H. A., Oyedele, L. O., Owolabi, H. A., Kumar, V., Ajayi, S. O., Akinade, O. O., & Bilal, M. (2018). Systematic review of bankruptcy prediction models: Towards a framework for tool selection. *Expert Systems with Applications*, *94*, 164–184. https://doi.org/10.1016/j.eswa.2017.10.040

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, *23*, 589–609. https://doi.org/10.1111/j.1540-6261.1968.tb00843.x

Ayadi, R., Abid, I., & Guesmi, K. (2020). Survival of reorganized firms in France. *Finance Research Letters*, 101434. https://doi.org/10.1016/j.frl.2020.101434

Bamber, D. (1975). The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology*, *12*, 387–415. https://doi.org/10.1016/0022-2496(75)90001-2

Beaver, W. H. (1966). Financial Ratios as Predictors of Failure. *Journal of Accounting Research*, *4*, 71–111. https://doi.org/10.2307/2490171

Boardman, C. M., Bartley, J. W., & Ratliff, R. L. (1981). Small Business Growth Characteristics. *American Journal of Small Business*, *5*, 33–43. https://doi.org/10.1177/104225878100500307

Box, G. E. P., & Cox, D. R. (1964). An Analysis of Transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, *26*, 211–243. https://doi.org/10.1111/j.2517-6161.1964.tb00553.x

Caroni, C. (2009). *Models for Reliability and Survival Analysis*. Symeon Publications.

Charalambous, C., Charitou, A., & Kaourou, F. (2000). Comparative Analysis of Artificial Neural Network Models: Application in Bankruptcy Prediction. *Annals of Operations Research*, *99*, 403–425. https://doi.org/10.1023/a:1019292321322

Ching, T., Zhu, X., & Garmire, L. X. (2018). Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data (F. Markowetz, Ed.). *PLOS Computational Biology*, *14*, e1006076. https://doi.org/10.1371/journal.pcbi.1006076

Cook, R. D., & Weisberg, S. (1999). *Applied regression including computing and graphics*. Wiley.

Deng, Y., Liu, L., Jiang, H., Peng, Y., Wei, Y., Zhou, Z., Zhong, Y., Zhao, Y., Yang, X., Yu, J., Lu, Z., Kho, A. N., Ning, H., Allen, N. B., Wilkins, J. T., Liu, K., Lloyd-Jones, D. M., & Zhao, L. (2023). Comparison of state-of-the-art neural network survival models with the pooled cohort equations for cardiovascular disease risk prediction. *BMC Medical Research Methodology*, *23*. https://doi.org/10.1186/s12874-022-01829-w

Famili, A., Shen, W., Weber, R., & Simoudis, E. (1997). Data preprocessing and intelligent data analysis. *Intelligent Data Analysis*, *1*, 3–23. https://doi.org/10.1016/s1088-467x(98)00007-9

Faraggi, D., & Simon, R. (1995). A neural network model for survival data. *Statistics in Medicine*, *14*, 73–82. https://doi.org/10.1002/sim.4780140108

Fletcher, D., & Goss, E. (1993). Forecasting with neural networks. *Information Management*, *24*, 159–167. https://doi.org/10.1016/0378-7206(93)90064-z

Gemar, G., Soler, I. P., & Guzman-Parra, V. F. (2019). Predicting bankruptcy in resort hotels: A survival analysis. *International Journal of Contemporary Hospitality Management*, *31*, 1546–1566. https://doi.org/10.1108/ijchm-10-2017-0640

Giunchiglia, E., Nemchenko, A., & van der Schaar, M. (2018). RNN-SURV: A deep recurrent model for survival analysis. *Artificial Neural Networks and Machine Learning - ICANN 2018*, 23–32. https://doi.org/10.1007/978-3-030-01424-7_3

Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*, 29–36. https://doi.org/10.1148/radiology.143.1.7063747

Hao, J., Kim, Y., Mallavarapu, T., Oh, J. H., & Kang, M. (2019). Interpretable deep neural network for cancer survival analysis by integrating genomic and clinical data. *BMC Medical Genomics*, *12*. https://doi.org/10.1186/s12920-019-0624-2

Harrell, F. (2016). *Regression modeling strategies: With applications to linear models, logistic and ordinal regression, and survival analysis*. Springer.

Hathaway, Q. A., Yanamala, N., Budoff, M. J., Sengupta, P. P., & Zeb, I. (2021). Deep neural survival networks for cardiovascular risk prediction: The multi-ethnic study of atherosclerosis (mesa). *Computers in Biology and Medicine*, *139*, 104983. https://doi.org/10.1016/j.compbiomed.2021.104983

Jeni, L. A., Cohn, J. F., & De La Torre, F. (2013). Facing Imbalanced Data–Recommendations for the Use of Performance Metrics. *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*. https://doi.org/10.1109/acii.2013.47

Jordan, J. (2023). Setting the learning rate of your neural network. *Jeremyjordan.me*. Retrieved July 12, 2023, from https://www.jeremyjordan.me/content/images/2018/02/Screen-Shot-2018-02-24-at-11.47.09-AM.png

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). Deepsurv: Personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology*, *18*. https://doi.org/10.1186/s12874-018-0482-1

Keasey, K., Thompson, S., & Wright, M. (1998). Corporate Governance, Economic, Management, and Financial Issues. oxford University Press, 1997. pp. 308 Hardback. *Managerial Auditing Journal*, *13*, 390–391. https://doi.org/10.1108/maj.1998.13.6.390.2

Kingma, D., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *Computer Science*. https://doi.org/10.48550/arXiv.1412.6980

Kvamme, H., Borgan, Ø., & Scheel, I. (2019). Time-to-Event Prediction with Neural Networks and Cox Regression. *Journal of Machine Learning Research*, *20*, 1–30.

Lane, W. R., Looney, S. W., & Wansley, J. W. (1986). An application of the Cox proportional hazards model to bank failure. *Journal of Banking Finance*, *10*, 511–531. https://doi.org/10.1016/s0378-4266(86)80003-6

Lang, M., Binder, M., Richter, J., Schratz, P., Pfisterer, F., Coors, S., Au, Q., Casalicchio, G., Kotthoff, L., & Bischl, B. (2019). Mlr3: A modern object-oriented machine learning framework in R. *Journal of Open Source Software*, *4*, 1903. https://doi.org/10.21105/joss.01903

Lee, M.-C. (2014). Business Bankruptcy Prediction Based on Survival Analysis Approach. *International Journal of Computer Science and Information Technology*, *6*, 103–119. https://doi.org/10.5121/ijcsit.2014.6207

Lippert, C., Shahriar, M. H., Sangtani, S., Arous, B., Aziz, M. A., Shaikh, A., & Chauhan, P. (2022). Sign language gesture recognition using Bayesian optimization and transfer learning. https://doi.org/10.13140/RG.2.2.32482.40643

Liu, H. (2011). Feature selection. In C. Sammut & G. I. Webb (Eds.). Springer. https://doi.org/10.1007/978-0-387-30164-8âĆČ06

Luoma, M. (1991). Survival analysis as a tool for company failure prediction. *Omega*, *19*, 673–678. https://doi.org/10.1016/0305-0483(91)90015-l

Magdon-Ismail, M., Nicholson, A., & Abu-Mostafa, Y. (1998). Financial markets: Very noisy information processing. *Proceedings of the IEEE*, *86*, 2184–2195. https://doi.org/10.1109/5.726786

Parker, S., Peters, G. F., & Turetsky, H. F. (2002). Corporate governance and corporate failure: A survival analysis. *Corporate Governance: The international journal of business in society*, *2*, 4–12. https://doi.org/10.1108/14720700210430298

Partington, G., & Kim, M. H. (2008). Modeling Bankruptcy Prediction Using Cox Regression Model with Time-Varying Covariates. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.1101876

Pereira, J. (2014). Survival Analysis Employed in Predicting Corporate Failure: A Forecasting Model Proposal. *International Business Research*, *7*. https://doi.org/10.5539/ibr.v7n5p9

Perez, M. (2006). Artificial neural networks and bankruptcy forecasting: A state of the art. *Neural Computing and Applications*, *15*, 154–163. https://doi.org/10.1007/s00521-005-0022-x

Pierri, F., & Caroni, C. (2017). Bankruptcy prediction by survival models based on current and lagged values of time-varying financial data. *Communications in Statistics: Case Studies, Data Analysis and Applications*, *3*, 62–70. https://doi.org/10.1080/23737484.2018.1431816

Pölsterl, S., Sarasua, I., Gutiérrez-Becker, B., & Wachinger, C. (2020). A wide and deep neural network for survival analysis from anatomical shape and tabular clinical data. *PKDD/ECML Workshops*, 453–464. https://doi.org/10.1007/978-3-030-43823-4_37

Ranganath, R., Perotte, A. J., Elhadad, N., & Blei, D. M. (2016). Deep Survival Analysis. *Machine Learning in Health Care*, 101–114.

Ren, K., Qin, J., Zheng, L., Yang, Z., Zhang, W., Qiu, L., & Yu, Y. (2019). Deep Recurrent Survival Analysis. *AAAI Conference on Artificial Intelligence*, *33*, 4798–4805. https://doi.org/10.1609/aaai.v33i01.33014798

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–408. https://doi.org/10.1037/h0042519

Shumway, T. (2001). Forecasting Bankruptcy More Accurately: A Simple Hazard Model. *The Journal of Business*, *74*, 101–124. https://doi.org/10.1086/209665

Sonabend, R., Király, F. J., Bender, A., Bischl, B., & Lang, M. (2021). Mlr3proba: An R package for machine learning in survival analysis (J. Wren, Ed.). *Bioinformatics*, *37*, 2789–2791. https://doi.org/10.1093/bioinformatics/btab039

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*, 1929–1958.

Tarkhan, A., Simon, N., Bengtsson, T., Nguyen, K., & Dai, J. (2021). *Survival prediction using deep learning* (R. Greiner, N. Kumar, T. A. Gerds, &, Eds.; Vol. 146). PMLR. https://proceedings.mlr.press/v146/tarkhan21a.html

Tong, J., & Zhao, X. (2022). Deep survival algorithm based on nuclear norm. *Journal of Statistical Computation and Simulation*, *92*, 1964–1976. https://doi.org/10.1080/00949655.2021.2015770

Vatcheva, K., & Lee, M. (2016). Multicollinearity in Regression Analyses Conducted in Epidemiologic Studies. *Epidemiology: Open Access*, *06*. https://doi.org/10.4172/2161-1165.1000227

Wilson, R. L., & Sharda, R. (1994). Bankruptcy prediction using neural networks. *Decision Support Systems*, *11*, 545–557. https://doi.org/10.1016/0167-9236(94)90024-8

Yeo, I.-K. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*, 954–959. https://doi.org/10.1093/biomet/87.4.954

Yuan, X., Feng, Z., Norton, M., & Li, X. (2019). Generalized Batch Normalization: Towards Accelerating Deep Neural Networks. *Proceedings of the AAAI Conference*

on Artificial Intelligence, *33*, 1682–1689. https://doi.org/10.1609/aaai.v33i01.33011682

Zelenkov, Y. (2020). Bankruptcy Prediction Using Survival Analysis Technique. https://doi.org/10.1109/cbi49978.2020.10071

Zhong, Q., Mueller, J. W., & Wang, J.-L. (2021). *Deep extended hazard models for survival analysis* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan, Eds.; Vol. 34). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2021/file/7f6caf1f0ba788cd7953d817724c2b6e-Paper.pdf

Zhu, X., Yao, J., & Huang, J. (2016). Deep convolutional neural network for survival analysis with pathological images. *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 544–547. https://doi.org/10.1109/bibm.2016.7822579

Appendix

R Code

```
library(reticulate)


library(mlr3extralearners)
library(mlr3)
library(ggplot2)
library(mlr3benchmark)
library(mlr3pipelines)
library(mlr3proba)
library(mlr3tuning)
library(survivalmodels)
library(paradox)
library(reticulate)
library(mlr3viz)
library(mlr3fselect)
library(mlr3filters)
library(mlr3mbo)
library(ggplot2)

set_seed(1234)

data= read.csv("Table8.csv")
data$Region= as.factor(data$Region)
data$Legal_form= as.factor(data$Legal_form)
data$Status= as.numeric(data$Status)
```

```
library(caret)

library(MASS)
args(stepAIC)
library(survival)
data1= data[,-c(15, 16,17)]
transdata <- as.data.frame(lapply(data1, function(x)
               bestNormalize::yeojohnson(x)$x))

transdata= cbind(transdata, data[,c(15,16,17)])


fit = coxph(Surv(Time,Status) ~ ., data=transdata)
summary(fit)
predicted_probs= survfit(fit, data=transdata)
summary(predicted_probs)
# Plot the predicted survival curves
plot(predicted_probs)


# Get summary statistics
summary(predicted_probs)


fitb=stepAIC(fit, direction = "backward")
summary(fitb)


task_surv <- TaskSurv$new("Italy2", data, time = "Time", event = "Status")

# Print the task object
print(task_surv)

summary(as.data.table(task_surv))

create_pipeops <- function(learner) {
  po("encode", method="treatment")
  %>>% po("yeojohnson")%>>% po("learner", learner)
}
```

```
learners <- lrns( "surv.deepsurv",
            frac = 0.3, early_stopping = TRUE, epochs = 100)


search_space <- ps(
  ## p_dbl for numeric valued parameters
  weight_decay = p_dbl(lower = 0, upper = 0.5),
  learning_rate = p_dbl(lower = 0, upper = 1),
  optimizer= p_fct(c("adam","sgd")),
  dropout = p_dbl(lower = 0, upper = 1),
  ## p_int for integer valued parameters
  nodes = p_int(lower = 1, upper =14 ),
  k = p_int(lower = 1, upper = 44),
  activation= p_fct(c("relu","selu"))
)

search_space$trafo <- function(x, param_set) {
  x$num_nodes = rep(x$nodes, x$k)
  x$nodes = x$k = NULL
  return(x)
}

future::tweak("multisession", workers = 8)

create_autotuner <- function(learner) {
  AutoTuner$new(
    learner = learner,
    search_space = search_space,
    resampling = rsmp("cv", folds=5),
    measure = msr("surv.cindex"),
    terminator = trm("stagnation_batch",n=7,threshold = 1e-4),
    tuner = tnr("grid_search", resolution=20, batch_size=50)
  )
}
create_fselector= function(learner){
  auto_fselector(
    fselector = fs("sequential"),
    learner = learner,
    resampling = rsmp("cv", folds=5),
    measure = msr("surv.cindex"),
    terminator = trm("stagnation_batch",n=7,threshold = 1e-4)
)
}
```

```
dpsurvat= lapply(learners, create_autotuner)
dpsurvfs= lapply(learners, create_fselector)
dpsurvat= lapply(dpsurvat, create_pipeops)
dpsurvfs=lapply(dpsurvfs, create_pipeops)
# apply our function


resampling <- rsmp("cv", folds = 10)

## add CPH

coxph=create_pipeops(lrn("surv.coxph"))

learners <- c(dpsurvat, coxph, dpsurvfs)


design <- benchmark_grid(task_surv, learners, resampling)
design
bm <- benchmark(design)
msrs <- msrs(c("surv.cindex", "surv.graf"))
aggr=bm$aggregate(msrs)
autoplot(bm)
```

Python Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn_pandas import DataFrameMapper
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
import seaborn as sns
from scipy import stats
```

```
data= pd.read_csv(r'C:/Users/Panayiota/Desktop/Italy_2.csv')
data.head(2)

data.columns

Index(['ID', 'Time', 'Age', 'ln(EBITDA)', 'Operating Revenue/Inventories',
       'ind007', 'ind011', 'ind020', '(Creditors/Operating revenue)*360',
       'Current Assets/Current Liabilities',
       'Debtors/Operating Revenue',
       'Shareholders Funds/ Total Assets', 'ind044', 'ind050',
       'ind052', 'ind055', 'ind056',
       'Non Current Liabilities/Total Assets',
       '(Long Term Debt + Loans) / Total Assets', 'ind063', 'ind065',
       'ind072',
       'ind079', 'ind080',
       'Profit (Loss) for period / Shareholders Funds',
       'EBIT/Shareholders Funds',
       'Profit (Loss) for period / Operating Revenue',
       'ind087', 'ind088',
       'ind089', 'ind090', 'Operating Revenue/ Total Assets',
       'ind093',
       'ind094', 'ind104', 'ind105', 'ind116', 'ind117',
       'Debtors/Current Assets', 'ind132', 'Status',
       'Region', 'Legal_form'],
      dtype='object')

data= data[[ 'Time', 'Age', 'ln(EBITDA)',
'(Creditors/Operating revenue)*360',
            'Current Assets/Current Liabilities',
            'Debtors/Operating Revenue',
            'Shareholders Funds/ Total Assets',
            'Non Current Liabilities/Total Assets',
            '(Long Term Debt + Loans) / Total Assets',
            'Profit (Loss) for period / Shareholders Funds',
            'EBIT/Shareholders Funds',
            'Profit (Loss) for period / Operating Revenue',
            'Operating Revenue/ Total Assets',
            'Debtors/Current Assets', 'Status', 'Region', 'Legal_form']]

data.head(2)

data.dtypes

Time                                                float64
```

```
Age                                                 float64
ln(EBITDA)                                          float64
(Creditors/Operating revenue)*360                   float64
Current Assets/Current Liabilities                  float64
Debtors/Operating Revenue                           float64
Shareholders Funds/ Total Assets                    float64
Non Current Liabilities/Total Assets                float64
(Long Term Debt + Loans) / Total Assets             float64
Profit (Loss) for period / Shareholders Funds       float64
EBIT/Shareholders Funds                             float64
Profit (Loss) for period / Operating Revenue        float64
Operating Revenue/ Total Assets                     float64
Debtors/Current Assets                              float64
Status                                                int64
Region                                                int64
Legal_form                                            int64
dtype: object

count_state_1 = len(data[data['Status'] == 1])
count_state_1

data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6897 entries, 0 to 6896
Data columns (total 17 columns):
 #   Column                                      Non-Null Count
Dtype
---  ------                                      --------------
-----
 0   Time                                        6897 non-null
float64
 1   Age                                         6897 non-null
float64
 2   ln(EBITDA)                                  6897 non-null
float64
 3   (Creditors/Operating revenue)*360           6897 non-null
float64
 4   Current Assets/Current Liabilities          6897 non-null
float64
 5   Debtors/Operating Revenue                   6897 non-null
float64
 6   Shareholders Funds/ Total Assets            6897 non-null
float64
```

```
 7   Non Current Liabilities/Total Assets           6897 non-null
float64
 8   (Long Term Debt + Loans) / Total Assets        6897 non-null
float64
 9   Profit (Loss) for period / Shareholders Funds  6897 non-null
float64
 10  EBIT/Shareholders Funds                        6897 non-null
float64
 11  Profit (Loss) for period / Operating Revenue   6897 non-null
float64
 12  Operating Revenue/ Total Assets                6897 non-null
float64
 13  Debtors/Current Assets                         6897 non-null
float64
 14  Status                                         6897 non-null
int64
 15  Region                                         6897 non-null
int64
 16  Legal_form                                     6897 non-null
int64
dtypes: float64(14), int64(3)
memory usage: 916.1 KB

data=pd.get_dummies(data, columns = ['Region', 'Legal_form'])

data= data.drop(['Region_1','Legal_form_1'], axis=1)
data.head(2)

data['(Long␣Term␣Debt␣+␣Loans)␣/␣Total␣Assets'].hist()

def plot_distributions(train):
    for column in train.columns[0:14]:
        sns.displot(train[column], kde=False)

plot_distributions(data)

def plot_outliers_boxplot(data):
    sns.boxplot(x=data)
    plt.show()

def detect_and_plot_outliers(data, method='zscore', threshold=3):
    if method == 'zscore':
        z_scores = stats.zscore(data)
        outliers = np.where(np.abs(z_scores) > threshold)[0]
```

```
    elif method == 'iqr':
        q1 = np.percentile(data, 25)
        q3 = np.percentile(data, 75)
        iqr = q3 - q1
        lower_bound = q1 - threshold * iqr
        upper_bound = q3 + threshold * iqr
        outliers = np.where((data < lower_bound) | (data > upper_bound))[0]
    elif method == 'mad':
        median = np.median(data)
        mad = np.median(np.abs(data - median))
        modified_z_scores = np.abs(0.6745 * (data - median) / mad)
        outliers = np.where(modified_z_scores > threshold)[0]
    else:
        raise ValueError("Invalid method specified.
        Available methods: 'zscore', 'iqr', 'mad'")

    plot_outliers_boxplot(data)

    return outliers


def find_outliers(data):
    all_outliers = []
    total_outliers = 0
    outliers_with_index = {}

    for column in data.columns[2:14]:
        col_outliers = detect_and_plot_outliers(data[column],
        method='zscore', threshold=3)

        total_outliers += len(col_outliers)
        all_outliers.extend(col_outliers)

        for outlier in col_outliers:
            if outlier in outliers_with_index:
                outliers_with_index[outlier].append(column)
            else:
                outliers_with_index[outlier] = [column]

    return all_outliers, outliers_with_index, total_outliers
```

```
all_outliers, outliers_with_index, total_outliers = find_outliers(data)

print("Total Outliers:", total_outliers)
print("Outliers with Index:", all_outliers)

print("\nOutliers with Same Index by Column:")
for index, columns in outliers_with_index.items():
    print("Index:", index, "Outliers in Columns:", columns)

def remove_outliers(data, outliers):
    data_no_outliers = data.drop(outliers)
    return data_no_outliers

data_no_outliers = remove_outliers(data, all_outliers)

data_no_outliers.shape

(6137, 21)

def plot_distributions(train):
    for column in train.columns[0:14]:
        sns.displot(train[column], kde=False)

plot_distributions(data_no_outliers)

import matplotlib.pyplot as plt
import seaborn as sns

def plot_distributions(train):
    num_columns = 14
    num_plots_per_line = 2
    num_lines = num_columns // num_plots_per_line

    fig, axs = plt.subplots(num_lines, num_plots_per_line,
    figsize=(8, 22))
    fig.subplots_adjust(hspace=1.2)

    for i, column in enumerate(train.columns[0:num_columns]):
        row = i // num_plots_per_line
        col = i % num_plots_per_line
        ax = axs[row, col]
        sns.histplot(train[column], ax=ax, kde=False)
        ax.set_title(column)
        ax.set_xlabel('Value')
```

```
        ax.set_ylabel('Frequency')

    # Remove any unused subplots
    if num_columns % num_plots_per_line != 0:
        for i in range(num_columns % num_plots_per_line,
        num_plots_per_line):
            fig.delaxes(axs[num_lines - 1, i])

    plt.show()

# Example usage:
plot_distributions(train)

def skew(data):
    for column in data.columns[0:14]:
        print(column)
        print(data[column].skew())

skew(train)

def trans(data):
    transformed_data = data.copy()
    for column in data.columns[0:14]:
        if data[column].skew() > 1:
         if (data[column] <= 0).any():
             transformed_data[column] = stats.yeojohnson(data[column])[0]
            else:
                # Apply log transform to column
                data[column] = np.log(data[column])
        elif data[column].skew() < -1:
            if (data[column] <= 0).any():
                if (data[column] < 0).any():
                 transformed_data[column]=stats.yeojohnson(data[column])[0]
                    # Alternatively, you can handle non-positive values
                    separately:
                    # transformed_data.loc[data[column] <= 0, column] = 0
                    # or some other value
            else:
                data_transformed, lambda_ = stats.boxcox(data[column])
                transformed_data.loc[data[column] > 0, column] =
                data_transformed
    return transformed_data
```

```
# Apply the transformation function
transformed_test=trans(test)
transformed_train= trans(train)
skew(transformed_train)
plot_distributions(transformed_train)

(transformed_train['EBIT/Shareholders Funds']).isna().sum(axis=0)

train_without_dummy= train.drop(['Region_2','Region_3','Region_4',
                    'Region_5','Legal_form_2','Legal_form_3'],axis=1)
corr_matrix = (train_without_dummy.corr())

mask = np.triu(np.ones_like(train_without_dummy.corr()))
plt.figure(figsize = (16,16))
sns.heatmap(corr_matrix, cmap='coolwarm',annot=True, mask= mask )
# show plot

plt.show()

threshold=0.6
# check for high correlation coefficients
high_corr = []
values = [] # list to store pairs of highly correlated variables
for i in range(len(corr_matrix.columns)):
    for j in range(i):
        if abs(corr_matrix.iloc[i, j]) > threshold:
            pair = (corr_matrix.columns[i], corr_matrix.columns[j])
            correlation_value = corr_matrix.iloc[i, j]
            high_corr.append(pair)
            values.append(correlation_value)

print('Pairs of highly correlated variables:', high_corr)
print('Correlation values:', values)

corr_matrix
```

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
Δ.Π.Μ.Σ. ΜΑΘΗΜΑΤΙΚΗ ΠΡΟΤΥΠΟΠΟΙΗΣΗ ΣΕ ΣΥΓΧΡΟΝΕΣ
ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΤΗ ΧΡΗΜΑΤΟΟΙΚΟΝΟΜΙΚΗ

Στοιχεία για την Επιβίωση Εταιρειών στην Ιταλία: Μια Συγκριτική Μελέτη των Μοντέλων Cox Proportional Hazards και DeepSurv

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Χάρκεν Αλεξίου Παναγιώτα Ισμήνη
Αριθμός μητρώου: **09321027**

Επιβλέπουσα
Χρυσηίς Καρώνη

Αθηνα
Ιούλιος, 2023

Περιεχόμενα

Περίληψη **2**

Ευχαριστίες **3**

1 Εισαγωγή **4**

2 Μαθηματικό Υπόβαθρο **5**

3 Ανάλυση Δεδομένων **7**

4 Συζήτηση **13**

Περίληψη

Η διπλωματική αυτή παρουσιάζει μια ολοκληρωμένη ανάλυση εταιρειών στην Ιταλία που έχουν κλείσει λόγω πτώχευσης και άλλων αιτίων. Η μελέτη χρησιμοποιεί τεχνικές ανάλυσης επιβίωσης για να εξετάσει τους παράγοντες που επηρεάζουν τον χρόνο επιβίωσης αυτών των εταιρειών. Δύο κύρια μοντέλα, το Cox Proportional Hazards (CPH) και το μοντέλο DeepSurv, χρησιμοποιούνται για την ανάλυση των δεδομένων.

Στο πρώτο μέρος της ανάλυσης επικεντρώνεται στο μοντέλο Cox Proportional Hazard,μια καθιερωμένη προσέγγιση στην ανάλυση επιβίωσης. Το μοντέλο CPH χρησιμοποιείται για να αναγνωρίσει και να κατανοήσει την επίδραση διάφορων παραγόντων κινδύνου στον χρόνο επιβίωσης των εταιρειών στο σύνολο δεδομένων.

Στο δεύτερο μέρος, η μελέτη χρησιμοποιεί το μοντέλο DeepSurv, το οποίο βασίζεται στην τεχνολογία των βαθιών νευρωνικών δικτύων για την ανάλυση επιβίωσης. Δύο παραλλαγές του μοντέλου DeepSurv εξετάζονται: μία που ενσωματώνει βελτιστοποίηση υπερπαραμέτρων (HPO) και μία που περιλαμβάνει διαδικασία επιλογής χαρακτηριστικών. Όλα τα μοντέλα αξιολογούνται χρησιμοποιώντας τον δείκτη C-Index, ένα διαδεδομένο μέτρο που μετρά τη συμφωνία μεταξύ προβλεπόμενων και παρατηρούμενων χρόνων επιβίωσης.

Τα αποτελέσματα δείχνουν ότι το μοντέλο DeepSurv με επιλογή χαρακτηριστικών επέτυχε τη υψηλότερη τιμή C-Index σε σχέση με όλα τα δοκιμασθέντα μοντέλα. Αυτό το εύρημα υποδηλώνει ότι η ενσωμάτωση της επιλογής χαρακτηριστικών συνέβαλε σημαντικά στη βελτίωση της προγνωστικής ικανότητας του μοντέλου για εργασίες ανάλυσης επιβίωσης. Επιπλέον, το μοντέλο DeepSurv με HPO επέδειξε ελαφρώς καλύτερη απόδοση από το παραδοσιακό μοντέλο Cox Proportional Hazard, προτείνοντας την δυνατότητα των τεχνικών βαθιάς μάθησης να ανακαλύπτουν περίπλοκα πρότυπα στα δεδομένα.

Συνολικά, αυτή η διπλωματική εργασία παρέχει πολύτιμες πληροφορίες σχετικά με τους παράγοντες που επηρεάζουν το κλείσιμο εταιρειών στην Ιταλία λόγω πτώχευσης και άλλων αιτίων. Η σύγκριση του μοντέλου Cox Proportional Hazard και του μοντέλου DeepSurv τονίζει τα πλεονεκτήματα της χρήσης των μεθόδων βαθιάς μάθησης για την ανάλυση επιβίωσης. Τα ευρήματα υπογραμμίζουν τη σημασία της επιλογής χαρακτηριστικών για τη βελτίωση των προγνωστικών δυνατοτήτων του μοντέλου DeepSurv, καθιστώντας το μια ελπιδοφόρα προσέγγιση για μελλοντικές μελέτες ανάλυσης επιβίωσης.

Λέξεις-κλειδιά: Cox Proportional Hazards, DeepSurv, Ανάλυση Επιβίωσης.

Ευχαριστίες

Θα ήθελα να εκφράσω την ειλικρινή μου ευγνωμοσύνη προς την καθηγήτρια μου, την αξιότιμη καθηγήτρια Χρυσηίς Καρώνη, για την ακλόνητη υποστήριξή της, την καθοδήγηση και τις πολύτιμες συμβουλές της κατά τη διάρκεια της διπλωματικής μου εργασίας. Η εμπειρία και ο ενθουσιασμός της ήταν καθοριστικοί για τον εμπλουτισμό και της κατανόησης μου στο θέμα.

Επίσης, ευχαριστώ βαθύτατα στην οικογένειά μου, την αγαπημένη μου μητέρα και αδελφή, για την συνεχή αγάπη, ενθάρρυνση και την κατανόησή τους. Η ακλόνητη πίστη τους σε εμένα ήταν μια κινητήρια δύναμη πίσω από τον ακαδημαϊκό μου προσανατολισμό, και είμαι απερίγραπτα ευγνώμων για τις θυσίες και την ενθάρρυνσή τους.

Τέλος, νιώθω ευγνώμων για την ευκαιρία που μου δόθηκε να παρακολουθήσω το πρόγραμμα Μεταπτυχιακών Σπουδών στη Μαθηματική Προτυποποίηση στις Σύγχρονες Τεχνολογίες και στα Χρηματοοικονομικά της Σχολής Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών του Εθνικού Μετσόβιου Πολυτεχνείου της Αθήνας, το οποίο με προμήθευσε με έναν πλούτο γνώσεων και δεξιοτήτων που ήταν ανεκτίμητες για την ολοκλήρωση της διπλωματικής μου εργασίας. Ευχαριστώ εκ βάθους καρδίας όλους όσους με υποστήριξαν σε αυτό το ταξίδι μου.

1 Εισαγωγή

Η χρεοκοπία αναφέρεται στην οικονομική κατάσταση όπου μια εταιρεία είναι ανίκανη να καταβάλει τις οφειλές της και έχει σοβαρές συνέπειες για τους ενδιαφερόμενους φορείς και την οικονομία. Η ακριβής πρόβλεψη της χρεοκοπίας είναι κρίσιμη για την αντιμετώπιση των οικονομικών κινδύνων. Αυτό έχει οδηγήσει σε εκτεταμένη έρευνα στο ακαδημαϊκό και πρακτικό πεδίο, με σκοπό την πρόβλεψη και πρόληψη της χρεοκοπίας, εξασφαλίζοντας την επιβίωση και την πρόοδο μιας εταιρείας.

Διάφορα μοντέλα κατηγοριοποιητικής ανάλυσης έχουν χρησιμοποιηθεί για την πρόβλεψη της χρεοκοπίας. Η πολλαπλή διακριτική ανάλυση (MDA) και η λογιστική παλινδρόμηση είναι συνήθεις επιλογές, αλλά οι υποθέσεις και τα περιορισμένα αποτελέσματά τους έχουν προκαλέσει ανησυχίες. Τα τεχνητά νευρωνικά δίκτυα (ANNs) έχουν επίσης αποκτήσει δημοτικότητα για την πρόβλεψη της χρεοκοπίας.

Η ανάλυση επιβίωσης, αρχικά αναπτυγμένη για εφαρμογές στην ιατρική, έχει εφαρμοστεί στην πρόβλεψη της χρεοκοπίας. Το μοντέλο Cox αναλογικών κινδύνων, ένα στατιστικό εργαλείο για χρονοσειρές δεδομένων, έχει χρησιμοποιηθεί για την εκτίμηση του σχετικού κινδύνου της χρεοκοπίας.

Επιπλέον, βαθιά νευρωνικά δίκτυα όπως το DeepSurv έχουν χρησιμοποιηθεί για την πρόβλεψη της χρεοκοπίας. Το DeepSurv συνδυάζει τη δύναμη των νευρωνικών δικτύων με την ανάλυση επιβίωσης, επιτρέποντας την εκτίμηση των πιθανοτήτων επιβίωσης και την αναγνώριση των σημαντικών παραγόντων κινδύνου της χρεοκοπίας.

Σε αυτή την εργασία, διεξάγουμε μια περιεκτική ανάλυση της πρόβλεψης της χρεοκοπίας, χρησιμοποιώντας δύο μέθοδους: το μοντέλο αναλογικών κινδύνων Cox και το DeepSurv , για να προσφέρουμε αξιόλογες εισηγήσεις για την πρόβλεψη της χρεοκοπίας, λαμβάνοντας υπόψη τόσο τα παραδοσιακά στατιστικά μοντέλα όσο και τις σύγχρονες μεθόδους βασισμένες σε νευρωνικά δίκτυα.

2 Μαθηματικό Υπόβαθρο

Ανάλυση Επιβίωσης

Η ανάλυση επιβίωσης είναι μια στατιστική μέθοδος που χρησιμοποιείται για την ανάλυση δεδομένων χρόνου-προς-συμβάν, όπου το συμβάν ενδιαφέροντος μπορεί να είναι ο θάνατος, η αποτυχία ή οποιοσδήποτε άλλος τύπος αποτέλεσματος. Η συνάρτηση επιβίωσης συμβολίζεται ως

$$S(t) = P[T > t] = \int_t^\infty f(u)du$$

η οποία αντιπροσωπεύει την πιθανότητα ότι η διάρκεια του χρόνου μέχρι να συμβεί ένα συμβάν είναι μεγαλύτερη από το t. Η πυκνότητα πιθανότητας της T συμβολίζεται ως

$$f(t) = -\frac{d}{dt}S(t)$$

Η βασική έννοια στην ανάλυση επιβίωσης είναι η συνάρτηση κινδύνου, που συμβολίζεται ως $h(t)$. Η συνάρτηση κινδύνου ορίζεται ως η πιθανότητα ενός συμβάντος να τελειώσει εντός ενός μικρού χρονικού διαστήματος $(t, t + \delta t)$, δεδομένου ότι το συμβάν δεν έχει τελειώσει ή διακοπεί μέχρι το χρόνο t. Μαθηματικά, εκφράζεται ως:

$$h(t) = \lim_{\delta t \to 0} \frac{S(t) - S(t + \delta t)/S(t)}{\delta t} = \frac{f(t)}{S(t)}$$

Μοντέλο Cox Αναλογικών Κινδύνων

Η ανάλυση επιβίωσης παρουσιάζει μια σημαντική πρόκληση καθώς συχνά περιλαμβάνει την ύπαρξη δεδομένων που έχουν αποκοπεί (συμβολίζονται με το σύμβολο C), οπότε δεν είναι διαθέσιμοι οι χρόνοι των συμβάντων όλων των αντικειμένων της έρευνας. Για να αντιμετωπιστεί αυτή η πρόκληση, είναι απαραίτητο να χρησιμοποιηθούν κατάλληλες στατιστικές τεχνικές που μπορούν να χειριστούν αυτά τα αποκομμένα δεδομένα. Το ημι-παραμετρικό μοντέλο αναλογικών κινδύνων Cox είναι μια μέθοδος που επιτρέπει την εξέταση της σχέσης μεταξύ των συντελεστών και της συνάρτησης κινδύνου. Υποθέτει ότι η συνάρτηση κινδύνου για ένα αντικείμενο είναι ανάλογη με τη βασική συνάρτηση κινδύνου, που συμβολίζεται ως $h_0(t)$, και ένα σύνολο συντελεστών X, ως εξής:

$$h(t; X) = h_0(t) \cdot \exp(\boldsymbol{\beta}_1 X_1 + \boldsymbol{\beta}_2 X_2 + \cdots + \boldsymbol{\beta}_p X_p)$$

όπου $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_p$ είναι οι παράμετροι παλινδρόμησης για τις μεταβλητές $X_1, X_2, \ldots, X_p$, αντίστοιχα.

Δεδομένης της εξίσωσης $H(t) = \int_0^t h(u)du$, μπορούμε να εκφράσουμε τη συσσωρευτική συνάρτηση κινδύνου ως

$$H(t; X) = H_0(t) \exp(\boldsymbol{\beta}_1 X_1 + \boldsymbol{\beta}_2 X_2 + \cdots + \boldsymbol{\beta}_p X_p)$$

Συνεπώς, μπορούμε να χρησιμοποιήσουμε αυτό για να παραγάγουμε τη συνάρτηση επιβίωσης $S(t)$ ως

$$S(t; X) = S_0(t)^{\exp(\boldsymbol{\beta}_1 X_1 + \boldsymbol{\beta}_2 X_2 + \cdots + \boldsymbol{\beta}_p X_p)}$$

Επισκόπηση του Deepsurv

Το DeepSurv αποτελεί ένα δίκτυο βαθιάς μάθησης που έχει ως βάση τον υπολογισμό της συνάρτησης κινδύνου και αποτελεί εξέλιξη του μοντέλου αναλογικού κινδύνου του Cox. Η ανάπτυξη των νευρωνικών δικτύων

σε συνδυασμό με το μοντέλο τουCox τις μεγάλες βάσεις δεδομένων με μη γραμμικές σχέσεις μεταξύ των συμμεταβλητών οδήγησαν στην δημιουργία του αλγορίθμου αυτού. Η δομή του έχει ως στόχο τον υπολογισμό της επίδρασης του κάθε ατόμου στην συνάρτηση κινδύνου του σε σχέση με τα αντίστοιχα βάρη του δικτύου και τα βήματα που ακολουθούμε είναι τα παρακάτω. Αρχικά, τα δεδομένα εισόδου X αποτελούνται από τις συμμεταβλητές που έχουν παρατηρηθεί. Ενώ, ο κορμός αποτελείται από τα κρυφά στρώματα, πλήρως συνδεδεμένα στρώματα κόμβων, ακολουθούμενα από ένα επίπεδο εγκατάλειψης (dropout) με στόχο την αποφυγή της υπερπροσαρμογής και τέλος το επίπεδο εξόδου έχει μόνο έναν κόμβο με μια γραμμική λειτουργία ενεργοποίησης που δίνει την έξοδο ($\hat{h}(x)$)(Λογαριθμικές εκτιμήσεις κινδύνου. Η προσδιοριστική απόδοση του μοντέλου επικυρώνεται χρησιμοποιώντας το δείκτη συμφωνίας (concordance index - c-index).

3 Ανάλυση Δεδομένων

Η προ-επεξεργασία και η ανάλυση των δεδομένων πραγματοποιήθηκαν χρησιμοποιώντας τη γλώσσα προγραμματισμού Πψτηον, έκδοση 3.8, και τη γλώσσα P, έκδοση 4.3.1. με τα πακέτα μλρ3, μλρ3προβα και γγπλοτ2. Περισσότερες λεπτομέρειες και κώδικας είναι διαθέσιμα στο Παράρτημα.

Το σύνολο δεδομένων περιλαμβάνει πληροφορίες από 6.897 επιχειρήσεις στην Ιταλία, παρακολουθημένες για μια περίοδο 19 ετών. Από αυτές τις επιχειρήσεις, 3.462 κατέληξαν σε ανενεργό καθεστώς λόγω πτώχευσης ή άλλων νομικών διαδικασιών. Το σύνολο δεδομένων αποτελείται από 5 χαρακτηριστικά που αντιπροσωπεύουν τις βασικές πληροφορίες των επιχειρήσεων και 12 χαρακτηριστικά που αντιπροσωπεύουν τις οικονομικές τους πληροφορίες.

Οι βασικές πληροφορίες περιλαμβάνουν τα εξής χαρακτηριστικά:

- Χρόνος (Time) : Αυτό το χαρακτηριστικό αναπαριστά τον χρόνο σε έτη μέχρι το κλείσιμο. Η τιμή 19 υποδηλώνει ότι η επιχείρηση είναι ακόμα ενεργή στο τέλος της μελέτης.
- Ηλικία (Age): Αυτό το χαρακτηριστικό υποδεικνύει τη διάρκεια του χρόνου που η επιχείρηση λειτουργεί στην αρχή της μελέτης.
- Κατάσταση (Status): Αυτή η δυαδική μεταβλητή υποδεικνύει την κατάσταση της επιχείρησης. Η τιμή 0 υποδηλώνει μια ενεργή επιχείρηση στο τέλος της μελέτης, ενώ η τιμή 1 υποδηλώνει μια ανενεργή επιχείρηση.
- Περιοχή (Region): Αυτό το χαρακτηριστικό αναπαριστά την περιοχή στην Ιταλία όπου βρίσκεται η επιχείρηση, κατηγοριοποιημένη σε πέντε περιοχές: Βορειοδυτική, Νότια, Νησιά, Βορειοανατολική και Κεντρική.
- Νομική Μορφή (Legal_form): Αυτό το χαρακτηριστικό αναπαριστά τη νομική μορφή της επιχείρησης, κατηγοριοποιημένη ως Εταιρία Κοινού Κεφαλαίου, Ιδιωτική Εταιρία Περιορισμένης Ευθύνης ή Δημόσια Εταιρία.

Οικονομικές πληροφορίες περιλαμβάνουν 12 οικονομικούς παράγοντες που αναμένεται να σχετίζονται με τα γεγονότα πτώχευσης. Αυτές οι μεταβλητές είναι οι εξής:

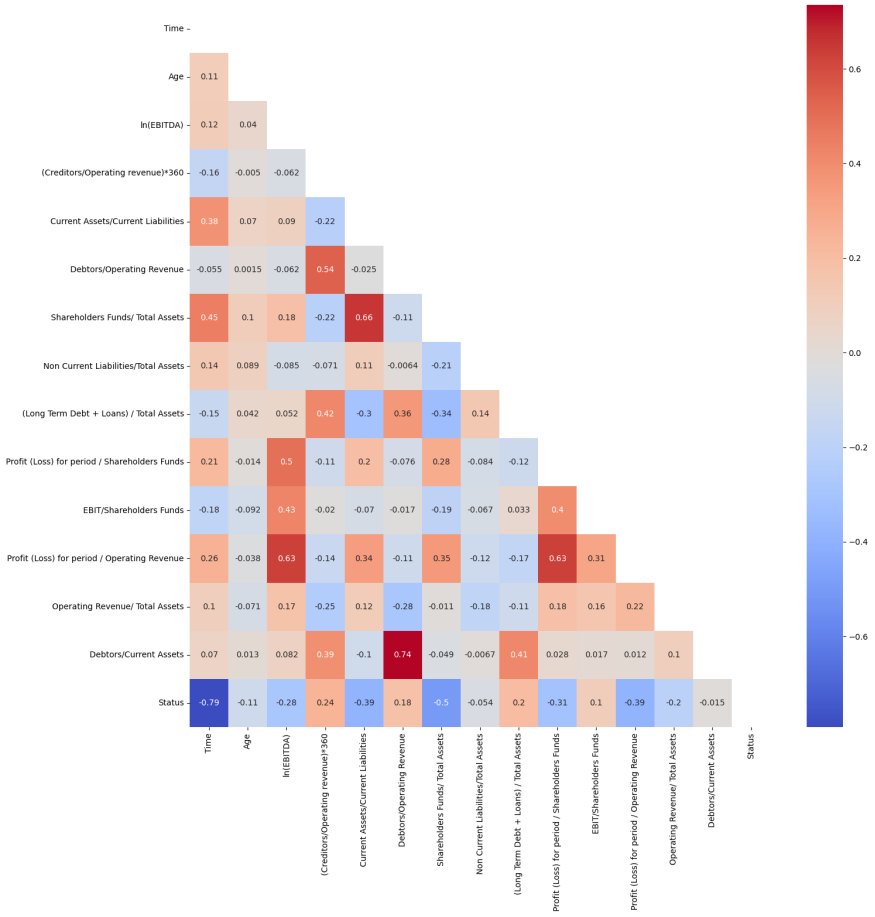
- ln(EBITDA)
- Operating Revenue/Inventories
- (Creditors/Operating revenue) * 360
- Current Assets/Current Liabilities
- Debtors/Operating Revenue
- Shareholders Funds/Total Assets
- Non-Current Liabilities/Total Assets
- (Long Term Debt + Loans)/Total Assets
- Profit (Loss) for period/Shareholders Funds.
- EBIT/Shareholders Funds
- Profit (Loss) for period/Operating Revenue
- Operating Revenue/Total Assets
- Debtors/Current Assets

Προεπεξεργασία δεδομένων είναι η διαδικασία μετατροπής ακατέργαστων δεδομένων σε ένα χρήσιμο μορφοποιημένο σύνολο δεδομένων. Στην περίπτωση των χρηματοοικονομικών δεδομένων του πραγματικού κόσμου, συχνά υπάρχουν προβλήματα όπως ατέλεια, αντιφάσεις και λάθη. Επίσης, η ακρίβεια των αλγορίθμων μηχανικής μάθησης μπορεί να επηρεαστεί από την κατανομή των δεδομένων. Για να αντιμετωπιστούν αυτές οι προκλήσεις και να ενισχυθεί η ακρίβεια των μοντέλων μηχανικής μάθησης, εφαρμόζονται διάφορες τεχνικές προεπεξεργασίας για να καταστήσουν τα δεδομένα σημαντικά.

Οι τεχνικές προεπεξεργασίας που χρησιμοποιούνται περιλαμβάνουν την κανονικοποίηση και τη μείωση της ανομοιογένειας. Για να διορθωθεί η ανομοιογένεια στα δεδομένα, εφαρμόζονται δύο κοινές μετασχηματίσεις, δηλαδή η μετασχηματισμός Box-Cox και ο Yeo-Johnson Power Transformation. Αυτές οι τεχνικές εφαρμόζονται για να διορθωθεί η ανομοιογένεια σε διάφορους τύπους μεταβλητών.

Με τη χρήση αυτών των μετασχηματισμών, τα δεδομένα τροποποιούνται για να επιτευχθεί μειωμένη ανομοιογένεια, επιτρέποντας μια πιο ακριβή ανάλυση και ερμηνεία των αποτελεσμάτων.

Όταν μια μεταβλητή σε ένα πλαίσιο δεδομένων περιέχει σημαντικές πληροφορίες σχετικά με μια άλλη ανεξάρτητη μεταβλητή, θεωρείται ότι εμφανίζει πολυσυσχέτιση. Η αντιμετώπιση της πολυσυσχέτισης είναι απαραίτητη για τη διασφάλιση της αξιόπιστης εκτίμησης των συντελεστών σε ένα μοντέλο παλινδρόμησης. Σε αυτήν τη μελέτη, οι συσχετίσεις μεταξύ αριθμητικών χαρακτηριστικών αξιολογήθηκαν χρησιμοποιώντας τη μέθοδο Pearson.



Σχήμα 1: Συσχετίσεις Pearson μεταξύ αριθμητικών μεταβλητών

Από τον πίνακα συσχέτισης, μπορούν να παρατηρηθούν αρκετά ζεύγη υψηλά συσχετισμένων μεταβλητών. Ωστόσο, η ισχυρή συσχέτιση μεταξύ των μεταβλητών δεν σημαίνει απαραίτητα περιττότητα ή την ανάγκη αφα-

ίρεσης των μεταβλητών από το σύνολο δεδομένων.

Τα αποτελέσματα του Αρχικού Μοντέλου Κινδύνων Αναλογίας Cox παρουσιάζονται παρακάτω.

Η πρώτη μεταβλητή στο μοντέλο είναι Age. Η τιμή p για αυτήν τη μεταβλητή είναι μικρή (0.0041), πράγμα που υποδηλώνει ισχυρά ενδείξεις για σημαντική συσχέτιση μεταξύ της ηλικίας (Age) και του κινδύνου. Παράλληλα, ο λόγος κινδύνου είναι 0.9963 με ένα 95% διάστημα εμπιστοσύνης από 0.9920 έως 0.99853, το οποίο υποδηλώνει ότι κρατώντας σταθερές τις άλλες μεταβλητές, μια επιπλέον χρονιά ηλικίας συσχετίζεται με τον κίνδυνο ¨θανάτου¨.

Επίσης, η δεύτερη μεταβλητή, ln(EBITDA) , που αντιπροσωπεύει τον φυσικό λογάριθμο του Κέρδους πριν από το Επιτόκιο, τους Φόρους, την Κατανάλωση, και την Αποσβέστωση, έχει έναν λόγο κινδύνου 0.9487. Αυτό υποδηλώνει ότι μια μονάδα αύξησης στον φυσικό λογάριθμο του EBITDA συσχετίζεται με μείωση 5.2% στον κίνδυνο, κρατώντας σταθερές τις άλλες μεταβλητές. Η τιμή p για αυτήν τη μεταβλητή είναι πολύ μικρή ($< 2e-16$), προτείνοντας ισχυρές ενδείξεις για σημαντική συσχέτιση μεταξύ της ln(EBITDA) και του κινδύνου.

Επιπλέον, η μεταβλητή Current.Assets/Current.Liabilities έχει ένα συντελεστή 0.02361, υποδηλώνοντας ότι μια μονάδα αύξησης στον λόγο των τρεχουσών ενεργητικών προς τις τρέχουσες υποχρεώσεις οδηγεί σε αύξηση 2.40% στον κίνδυνο, εφόσον οι άλλες μεταβλητές παραμένουν σταθερές. Η τιμή p είναι πολύ μικρή ($6.74e-14$), υποδηλώνοντας σημαντική συσχέτιση μεταξύ αυτής της μεταβλητής και του κινδύνου.

Επιπλέον, η μεταβλητή Shareholders.Funds/Total.Assets έχει έναν συντελεστή -4.602. Ο λόγος κινδύνου είναι 0.0100, υποδηλώνοντας ότι μια μονάδα αύξησης στον λόγο των κεφαλαίων των μετόχων προς τα συνολικά ενεργητικά συσχετίζεται με μείωση 99.00% στον κίνδυνο. Η τιμή p είναι πολύ μικρή ($< 2e-16$), υποδηλώνοντας υψηλή σημαντικότητα αυτής της μεταβλητής στον κίνδυνο.

Επίσης, η μεταβλητή Non.Current.Liabilities/Total.Assets έχει έναν συντελεστή -2.584 και έναν λόγο κινδύνου 0.0755. Αυτό υποδηλώνει ότι μια μονάδα αύξησης στον λόγο των μη τρεχουσών υποχρεώσεων προς τα συνολικά ενεργητικά συσχετίζεται με μείωση 92.45% στον κίνδυνο. Η τιμή p είναι πολύ μικρή ($< 2e-16$), υποδηλώνοντας υψηλή σημαντικότητα.

Επίσης, η μεταβλητή (Long.Term.Debt + Loans)/Total.Assets έχει έναν συντελεστή 0.6326, υποδηλώνοντας ότι ένας υψηλότερος λόγος μακροπρόθεσμων χρεών και δανείων προς τα συνολικά ενεργητικά συσχετίζεται με αύξηση 88.20% στον κίνδυνο. Η τιμή p είναι μικρή ($5.11e-10$), υποδηλώνοντας σημαντική συσχέτιση.

Επιπλέον, η μεταβλητή Profit.Loss.for.period/Shareholders.Funds έχει έναν συντελεστή -0.07502, υποδηλώνοντας ότι μια μονάδα αύξησης στον λόγο κέρδους/ζημίας για την περίοδο προς τα κεφάλαια των μετόχων συσχετίζεται με μείωση 7.23% στον κίνδυνο. Η τιμή p είναι υψηλά σημαντική ($1.28e-08$).

Επιπλέον, η μεταβλητή EBIT/Shareholders.Funds έχει έναν συντελεστή 0.1089. Αυτό υποδηλώνει ότι ένας υψηλότερος λόγος κέρδους πριν από το επιτόκιο και τους φόρους προς τα κεφάλαια των μετόχων συσχετίζεται με αύξηση 11.50% στον κίνδυνο. Η τιμή p είναι πολύ μικρή ($< 2e-16$), υποδηλώνοντας υψηλή σημαντικότητα.

Επίσης, η μεταβλητή Operating.Revenue/Total.Assets έχει έναν συντελεστή -0.3314. Αυτό υποδηλώνει ότι ένας υψηλότερος λόγος λειτουργικών εσόδων προς τα συνολικά ενεργητικά συσχετίζεται με μείωση 28.21% στον κίνδυνο. Ο λόγος κινδύνου είναι 0.7179, και η τιμή p είναι πολύ μικρή ($2.29e-15$), υποδηλώνοντας υψηλή σημαντικότητα.

Επιπλέον, η μεταβλητή Debtors/Current.Assets έχει έναν συντελεστή -0.5208, υποδηλώνοντας ότι ένας υψηλότερος λόγος οφειλετών προς τα τρέχοντα ενεργητικά συσχετίζεται με μείωση 40.60% στον κίνδυνο. Ο λόγος κινδύνου είναι 0.5940, και η τιμή p είναι υψηλά σημαντική ($5.25e-10$).

Στη συνέχεια, έχουμε τις μεταβλητές που αντιπροσωπεύουν διάφορες περιοχές. Region2 έχει έναν συντελεστή 0.2796, υποδηλώνοντας ότι η παρουσία στην Region2 συσχετίζεται με αύξηση 32.30% στον κίνδυνο. Ο λόγος κινδύνου είναι 1.3230, και η τιμή p είναι υψηλά σημαντική ($6.62e-07$). Παρόμοια, η Region3 έχει έναν συντελεστή 0.2339, υποδηλώνοντας αύξηση 26.40% στον κίνδυνο σε σύγκριση με τη βασική περιοχή. Ο λόγος κινδύνου είναι 1.2640, και η τιμή p είναι σημαντική (0.00981). Αντίθετα, στην περίπτωση της Region4 , ο συντελεστής είναι -0.08243, υποδηλώνοντας μείωση 7.91% στον κίνδυνο σε σύγκριση με τη βασική περιοχή. Ο λόγος κινδύνου είναι 0.9209, και η τιμή p υποδηλώνει μια περιθωριακή σχέση (0.05512). Αντίθετα, όταν εξετάζουμε την Region5 , παρατηρούμε μια τιμή p της 0.38883, υποδηλώνοντας έλλειψη σημαντικής συσχέτισης. Ο λόγος κινδύνου είναι 0.9583, και το αντίστοιχο 95% διάστημα εμπιστοσύνης κυμαίνεται από 0.86 έως 1.05. Δεδομένου ότι το διάστημα εμπιστοσύνης περιλαμβάνει την τιμή 1, αυτά τα ευρήματα υποδηλώνουν ότι

η Region5 δεν εμφανίζει σημαντικές διαφορές από τη βασική περιοχή, Region1 .

Τέλος, έχουμε τις μεταβλητές που αντιπροσωπεύουν διαφορετικές νομικές μορφές. Η Legal_form2 έχει έναν συντελεστή 0.5717, υποδηλώνοντας ότι η ύπαρξη της Legal_form2 συσχετίζεται με αύξηση 77.10% στον κίνδυνο σε σύγκριση με τη βασική νομική μορφή. Ο λόγος κινδύνου είναι 1.7710, και η τιμή p είναι σημαντική (0.00596). Παρόμοια, η Legal_form3 έχει έναν συντελεστή 0.2256, υποδηλώνοντας αύξηση 25.70% στον κίνδυνο. Ο λόγος κινδύνου είναι 1.2570, και η τιμή p είναι σημαντική (0.0347). Όσο για τη Legal_form4 , η τιμή p είναι 0.42358, υποδηλώνοντας έλλειψη σημαντικής συσχέτισης. Ο λόγος κινδύνου είναι 0.8626, και το αντίστοιχο 95% διάστημα εμπιστοσύνης κυμαίνεται από 0.75 έως 0.98, προτείνοντας ότι η Legal_form4 δεν εμφανίζει σημαντικές διαφορές από τη βασική νομική μορφή, Legal_form1 .

Αυτά τα αποτελέσματα προσφέρουν σημαντικές ενδείξεις για το πώς διάφοροι παράγοντες συσχετίζονται με τον κίνδυνο αναλογίας στον τομέα της οικονομικής ανάλυσης. Αυτό μπορεί να βοηθήσει τους επενδυτές, τους τραπεζίτες και τους αναλυτές να κατανοήσουν καλύτερα τους παράγοντες που επηρεάζουν τον κίνδυνο στον τομέα της χρηματοοικονομικής απόδοσης επιχειρήσεων.

Δύο μοντέλα DeepSurv χρησιμοποιήθηκαν στην ανάλυση. Στο πρώτο μοντέλο, διεξήχθη βελτιστοποίηση υπερπαραμέτρων για τη βελτίωση της απόδοσης του μοντέλου. Χρησιμοποιήθηκε το αντικείμενο AutoTuner για να αναζητήσει μέσα από έναν προκαθορισμένο χώρο αναζήτησης, με στόχο την εύρεση των βέλτιστων υπερπαραμέτρων. Το δείκτης συμφωνίας (Concordance index - C-index) επιλέχθηκε ως η μετρική απόδοσης για τη βελτιστοποίηση.

Στο δεύτερο μοντέλο DeepSurv , χρησιμοποιήθηκε η επιλογή χαρακτηριστικών για την εύρεση των πιο ενδιαφέροντων προβλέπτων για την ανάλυση επιβίωσης. Χρησιμοποιήθηκε η feature selection , χρησιμοποιώντας τη μετρική απόδοσης "surv.cindex"για να προσδιοριστεί το υποσύνολο των χαρακτηριστικών που εμφάνισαν την καλύτερη προβλεπτική απόδοση.

Η διαδικασία βελτιστοποίησης υπερπαραμέτρων επιδιώκει να εντοπίσει τον καλύτερο συνδυασμό υπερπαραμέτρων που μεγιστοποιεί τον δείκτη συμφωνίας επιβίωσης (Survival C-index) στον δοθέντα σύνολο δεδομένων. Η διαδικασία βελτιστοποίησης χρησιμοποιεί έναν πλέγμα αναζήτησης με ανάλυση 20 και μέγεθος πακέτου 50, σε συνδυασμό με διασταυρούμενη επικύρωση και κριτήριο τερματισμού.

Η διαδικασία βελτιστοποίησης υπερπαραμέτρων στοχεύει στον εντοπισμό του καλύτερου συνδυασμού υπερπαραμέτρων που μεγιστοποιεί τον δείκτη συμφωνίας επιβίωσης.

Συνολικά, η διαδικασία βελτιστοποίησης υπερπαραμέτρων χρησιμοποιεί ένα πλέγμα αναζήτησης, διασταυρούμενη επικύρωση και κριτήριο τερματισμού για να εξερευνήσει αποτελεσματικά το χώρο των υπερπαραμέτρων και να βρει τον βέλτιστο συνδυασμό για το μοντέλο DeepSurv . Η εξωτερική δειγματοληψία με 10-πλή διασταυρούμενη επικύρωση εξασφαλίζει την αξιοπιστία της αξιολόγησης της απόδοσης του μοντέλου, και το τελικά επιλεγμένο μοντέλο αντιπροσωπεύει την καλύτερη απόδοση στο δεδομένο σύνολο δεδομένων.

Η Πίνακας 1 παρουσιάζει τα 10 μοντέλα που επέτυχαν τις υψηλότερες βαθμολογίες C-index μαζί με τον βέλτιστο συνδυασμό υπερπαραμέτρων τους και την αξιολόγηση της απόδοσής τους στα δεδομένα ελέγχου. Είναι σαφές ότι δεν υπάρχουν σημαντικά στοιχεία για μια συνεκτική τάση στο C-index όσον αφορά αυτές τις υπερπαράμετρους. Οι τιμές του C-index φαίνεται να ποικίλλουν χωρίς να εμφανίζουν έναν αντιληπτό προτύπων καθώς οι τιμές των υπερπαραμέτρων αλλάζουν.

Πίνακας 1: Μοντέλα DeepSurv με τον βέλτιστο συνδυασμό υπερπαραμέτρων

Weight Decay	Learning Rate	Optimizer	Dropout	Nodes/Layer	Layers	Activation	C-Index of Train Data	C-Index of Test Data
0.4736842	0.4210526	sgd	0.5789474	5	10	selu	0.7917773	0.7665178
0.4736842	0.4210526	adam	0.5789474	1	3	relu	0.7919148	0.7755227
0.4736842	0.7368421	sgd	0.8421053	3	14	relu	0.7871779	0.8047545
0.2631579	0.3157895	sgd	0.9473684	4	31	selu	0.7885051	0.7932329
0.2368421	0.8421053	adam	0	8	44	selu	0.7882624	0.7913179
0.1315789	0.3157895	sgd	0.5263158	7	42	relu	0.7862546	0.7867108
0.02631579	0.6842105	adam	0.8947368	10	1	selu	0.7885757	0.7685149
0.07894737	0.1578947	sgd	0.05263158	4	26	selu	0.7889386	0.8034955
0.5	0.5789474	adam	0.7894737	11	1	relu	0.7897789	0.7695737
0.2368421	0.9473684	sgd	0.8947368	3	10	selu	0.7881934	0.7810905

Ο διαδικασία επιλογής χαρακτηριστικών χρησιμοποιείται για να εντοπιστούν τα χαρακτηριστικά που συνεισφέρουν σημαντικά στην προβλεπτική απόδοση του μοντέλου DeepSurv . Σε αυτήν τη μελέτη χρησιμοποιείται

μια τεχνική ακολουθιακής επιλογής χαρακτηριστικών, η οποία χαρακτηρίζεται από την επαναληπτική της φύση. Αυτή η μέθοδος συμμορφώνεται με την προσέγγιση επιλογής χαρακτηριστικών με συνολική περικλείουσα προσέγγιση, δεδομένου ότι βασίζεται σε ένα μοντέλο μάθησης (αλγόριθμος εκμάθησης) για να καθοδηγήσει τη διαδικασία της επιλογής χαρακτηριστικών. Επιπλέον, αξίζει να σημειωθεί ότι αυτή η μέθοδος επιλογής χαρακτηριστικών λειτουργεί με επίβλεψη. Η διαδικασία ξεκινά με ένα άδειο σύνολο χαρακτηριστικών και προσθέτει σταδιακά χαρακτηριστικά με βάση την επίδρασή τους στο μέτρο απόδοσης, ειδικότερα το C-index , το οποίο αξιολογεί την προβλεπτική ακρίβεια του μοντέλου επιβίωσης. Ο στόχος είναι να εντοπιστεί το βέλτιστο υποσύνολο χαρακτηριστικών που οδηγεί στη μεγαλύτερη βελτίωση στο C-index. Παρατηρούμε ότι τα παρακάτω χαρακτηριστικά εμφανίζονται συνεχώς στα επιλεγμένα σύνολα:

Debtors/Operating Revenue : Αυτό το χαρακτηριστικό εμφανίζεται σε και τα δύο μέρη, υποδηλώνοντας την υψηλή του σημασία στην πρόβλεψη των χρόνων επιβίωσης. Υποδεικνύει ότι η αναλογία των οφειλετών προς τα έσοδα λειτουργίας έχει σημαντική επίδραση στον κίνδυνο εμφάνισης γεγονότων.

EBIT/Shareholders Funds: Ένα άλλο χαρακτηριστικό παρών σε και τα δύο μέρη, υποδηλώνοντας τη σχετικότητά του. Ο λόγος κερδών πριν από το επιτόκιο και τους κερδών των μετόχων φαίνεται να είναι ένας κρίσιμος προγνώστης των χρόνων επιβίωσης.

Non Current Liabilities/Total Assets: Αυτό το χαρακτηριστικό επιλέγεται συνεχώς, υποδεικνύοντας ότι η αναλογία των μη τρεχουσών υποχρεώσεων προς το συνολικό σύνολο των περιουσιακών στοιχείων είναι ένα ισχυρό δείκτη κινδύνου στο πλαίσιο της ανάλυσης επιβίωσης.

ln(EBITDA): Ο φυσικός λογάριθμος του EBITΔA εμφανίζεται σε και τα δύο μέρη, επισημαίνοντας τη σημασία του ως προγνώστη. Αυτό υποδηλώνει ότι ο φυσικός λογάριθμος των κερδών πριν από το επιτόκιο, των φόρων, της αποσβέσεων και της αποτιμήσεων έχει σημαντική επίδραση στην απόδοση του μοντέλου.

Profit Loss for period/Operating Revenue και Profit Loss for period/Shareholders Funds Και τα δύο αυτά χαρακτηριστικά επιλέγονται συνεχώς, υποδεικνύοντας τη σχετικότητά τους στην πρόβλεψη των κινδύνων γεγονότων.

Το μοντέλο DeepSurv με επιλογή χαρακτηριστικών επέτυχε συνεχώς τις υψηλότερες τιμές C-Index στην πλειοψηφία των διαδικασιών διασταυρούμενης επικύρωσης. Αυτό το αποτέλεσμα υποδηλώνει ότι η περίληψη της επιλογής χαρακτηριστικών συνέβαλε σημαντικά στη βελτίωση της προβλεπτικής απόδοσης του μοντέλου, καθώς πιθανόν βοήθησε στον εντοπισμό των πιο σχετικών και ενημερωτικών χαρακτηριστικών για την ανάλυση επιβίωσης.

Αντίστοιχα, το μοντέλο Cox Proportional Hazards παρήγαγε γενικά χαμηλότερες τιμές C-Index σε σύγκριση με τα μοντέλα DeepSurv. Αυτή η παρατήρηση υποδηλώνει ότι τα μοντέλα DeepSurv , εκμεταλλευόμενα τεχνικές βαθιάς μάθησης, διαθέτουν τη δυνατότητα να ανιχνεύουν πιο περίπλοκα πρότυπα εντός των δεδομένων. Συνεπώς, υπερτερούν του παραδοσιακού μοντέλου Cox Proportional Hazards στο πλαίσιο των εργασιών ανάλυσης επιβίωσης, υποδεικνύοντας τη δυνατότητά τους να αντιμετωπίζουν καλύτερα τις πολύπλοκες σχέσεις και εξαρτήσεις που υπάρχουν στα δεδομένα επιβίωσης.

Πίνακας 2: Σύγκριση Μοντέλων - Συγκέντρωση Αποτελεσμάτων

"	Μοντέλο	Μέθοδος Διασταυρούμενης Επικύρωσης	Επαναλήψεις	C-Index	Integrated Graf Score
1	DeepSurv (HPO)	CV	10	0.7841	0.1238
2	Proportional Cox Hazard	CV	10	0.7708	0.1319
3	DeepSurv (Επιλογή Χαρακτηριστικών)	CV	10	0.7896	0.1323

Ο Πίνακας 2 παρουσιάζει την αξιολόγηση της απόδοσης των τριών μοντέλων χρησιμοποιώντας δύο μετρικές: το C-Index και το Integrated Graf Score . Τα μοντέλα υποβλήθηκαν σε διασταυρούμενη επικύρωση με 10 διασταυρούμενες επικυρώσεις (CV).

DeepSurv (HPO): Αυτό το μοντέλο, που χρησιμοποίησε την Υπερπαραμετροποίηση (HPO), πέτυχε μέσο C-Index περίπου 0,7841 και μέσο Integrated Graf Score περίπου 0,1238. Το υψηλότερο C-Index υποδηλώνει ότι το μοντέλο εμφάνισε λογική συμφωνία μεταξύ των προβλεπόμενων και παρατηρηθέντων χρόνων επιβίωσης.

Αναλογικό Cox Hazard: Αυτό το μοντέλο, βασισμένο στην παραδοσιακή προσέγγιση του Αναλογικού Cox Hazard, πέτυχε μέσο C-Index περίπου 0,7708 και μέσο Integrated Graf Score περίπου 0,1319. Παρόλο που το C-Index του είναι ελαφρώς χαμηλότερο από αυτό του μοντέλου DeepSurv με HPO, υποδηλώνει ακόμα λογική

απόδοση σε εργασίες ανάλυσης επιβίωσης.

DeepSurv (Επιλογή Χαρακτηριστικών): Το μοντέλο DeepSurv με επιλογή χαρακτηριστικών εμφάνισε την καλύτερη απόδοση ανάμεσα σε όλα τα μοντέλα, επιτυγχάνοντας μέσο C-Index περίπου 0,7896 και μέσο Integrated Graf Score περίπου 0,1323. Αυτά τα αποτελέσματα υποδηλώνουν ότι η συμπερίληψη της επιλογής χαρακτηριστικών βελτίωσε τη δυνατότητα του μοντέλου να προβλέπει τους χρόνους επιβίωσης, καθιστώντας το πιο αποτελεσματικό μοντέλο σε αυτήν τη σύγκριση.

Συνολικά, το μοντέλο DeepSurv με επιλογή χαρακτηριστικών υπερτερεί των άλλων μοντέλων, συμπεριλαμβανομένου του μοντέλου DeepSurv με HPO και του παραδοσιακού μοντέλου Cox Proportional Hazards, βάσει των μετρικών αξιολόγησης (C-Index και Integrated Graf Score.

Σε μια συγκριτική ανάλυση μεταξύ του μοντέλου Cox Proportional Hazards και του μοντέλου DeepSurv με επιλογή χαρακτηριστικών, και τα δύο μοντέλα αποκάλυψαν σημαντικές συσχετίσεις μεταξύ διαφόρων μεταβλητών και του κινδύνου των γεγονότων, παρέχοντας πολύτιμες εισηγήσεις σχετικά με τους παράγοντες που επηρεάζουν τον κίνδυνο της λήξης για τις επιχειρήσεις στο σύνολο δεδομένων.

Το μοντέλο Cox Proportional Hazards εντόπισε αρκετές μεταβλητές που έδειξαν σημαντικές σχέσεις με τον κίνδυνο, συμπεριλαμβανομένων της Ηλικίας, του ln(EBITDA),του Current.Assets/Current.Liabilities,του Shareholders.Funds/Total.Assets,του (Long.Term.Debt + Loans)/Total.Assets,του Non.Current.Liabilities/Total.Assets, του EBIT/Shareholders.Funds,του Profit.Loss.for.period/Shareholders.Funds,του Operating.Revenue/Total.Assets,του Debtors/Current.Assets,του Region2, του Region3, του Legal_form2και του Legal_form3. Από την άλλη, το μοντέλο DeepSurv με επιλογή χαρακτηριστικών παρείχε πολύτιμες εισηγήσεις σχετικά με τα πιο σημαντικά χαρακτηριστικά για την πρόβλεψη των χρόνων επιβίωσης. Ειδικότερα, τα Debtors/Operating Revenue, EBIT/Shareholders Funds, Non Current Liabilities/Total Assets και ln(EBITDA) εμφανίστηκαν συνεχώς στα επιλεγμένα σύνολα, υπογραμμίζοντας την υψηλή τους σημασία ως προγνωστικοί παράγοντες των χρόνων επιβίωσης. Επιπλέον, τα Profit Loss for period/Operating Revenue και Profit Loss for period/Shareholders Funds, που σχετίζονται και τα δύο με την κερδοφορία, επιλέγηκαν συνεχώς, υποδεικνύοντας τη σημασία τους στην πρόβλεψη των κινδύνων.

Είναι σημαντικό να σημειωθεί ότι το μοντέλο DeepSurv με επιλογή χαρακτηριστικών επέλεξε ένα μικρότερο σύνολο μεταβλητών σε σύγκριση με το μοντέλο Cox Proportional Hazards. Ωστόσο, οι επιλεγμένες μεταβλητές ήταν ακόμα κρίσιμες για την ανίχνευση των πολύπλοκων προτύπων που σχετίζονται με την ανάλυση επιβίωσης. Αυτό υποδηλώνει ότι η διαδικασία επιλογής χαρακτηριστικών ήταν αποτελεσματική στον εντοπισμό των πιο σχετικών και ενημερωτικών χαρακτηριστικών, βελτιώνοντας την προβλεπτική απόδοση του μοντέλου DeepSurv με λιγότερες μεταβλητές.

Συνολικά, τα δύο μοντέλα παρείχαν πολύτιμες εισηγήσεις σχετικά με τα μοτίβα επιβίωσης των επιχειρήσεων στην Ιταλία. Το μοντέλο Cox Proportional Hazards αποκάλυψε συσχετίσεις μεταξύ πολλών μεταβλητών και του κινδύνου των γεγονότων, ενώ το μοντέλο DeepSurv με επιλογή χαρακτηριστικών εντόπισε ένα μικρότερο, αλλά εξίσου σημαντικό σύνολο προβλεπτικών. Τα ευρήματα από τα δύο μοντέλα συνεισφέρουν στην καλύτερη κατανόηση των παραγόντων που επηρεάζουν τους κινδύνους λήξης των επιχειρήσεων και υπογραμμίζουν τη δυνατότητα του μοντέλου DeepSurv με επιλογή χαρακτηριστικών για την προβλεπτική μοντελοποίηση στην ανάλυση επιβίωσης.

4 Συζήτηση

Στη διάρκεια αυτής της εργασίας, ερευνήθηκε και συγκρίθηκε η προγνωστική ικανότητα του παραδοσιακού μοντέλου Cox Proportional Hazards (PH) με του τεχνητού νευρωνικού δικτύου DeepSurv. Ο στόχος ήταν η εκτίμηση της επιβίωσης για επιχειρήσεις στην Ιταλία και η διερεύνηση ποια χαρακτηριστικά παίζουν σημαντικό ρόλο στην αποτυχία επιχειρήσεων. Για να επιτευχθεί αυτό, δημιουργήθηκαν τρία μοντέλα: ένα παραδοσιακό μοντέλο Cox Proportional Hazard, ένα μοντέλο DeepSurv με υπερπαραμετροποίηση (Hyperparameter Optimization, HPO) και ένα μοντέλο DeepSurv με επιλογή χαρακτηριστικών. Συνολικά, όλα τα μοντέλα έδειξαν παρόμοια προγνωστική απόδοση, με τα μοντέλα DeepSurv να υπερτερούν του μοντέλουCox PH όσον αφορά το Survival Concordance Index.

Η διαδικασία υπερπαραμετροποίησης ακολούθησε μια μεθοδική προσέγγιση, συνδυάζοντας αναζήτηση πλέγματος, διασταυρούμενη επικύρωση και κριτήριο τερματισμού. Αυτό βοήθησε να εξερευνηθούν αποτελεσματικά οι επιλογές των υπερπαραμέτρων και να βρεθεί ο συνδυασμός που προσφέρει το υψηλότερο Survival Concordance Index (C-index). Η επιλογή των υπερπαραμέτρων για βελτιστοποίηση, συμπεριλαμβανομένων του βάρους απομάκρυνσης (weight decay),του ρυθμού μάθησης (learning rate),τοudropout,του αριθμού κόμβων ανά επίπεδο, των επιπέδων, του βελτιστοποιητή (optimizer) και της συνάρτησης ενεργοποίησης, έγινε με προσοχή. Αυτές οι ρυθμίσεις επηρεάζουν τον τρόπο λειτουργίας του νευρωνικού δικτύου εντός του μοντέλου DeepSurv. Η χρήση αναζήτησης πλέγματος και διασταυρούμενης επικύρωσης διασφάλισε ότι η διαδικασία βελτιστοποίησης ήταν ισχυρή και ότι οι επιλεγμένες υπερπαράμετρες λειτουργούν καλά για νέα δεδομένα που το μοντέλο δεν έχει δει προηγουμένως. Αυτό οδηγεί σε καλύτερη απόδοση κατά την πρόβλεψη αποτελεσμάτων.

Παρόλο που καταβλήθηκε αρκετή προσπάθεια σε αυτήν τη διαδικασία, ήταν δύσκολο να βρεί κανείς ένα καθαρό μοτίβο μεταξύ παραγόντων και πώς επηρέαζαν το C-index. Αυτό δείχνει ότι ο τρόπος με τον οποίο λειτουργούν αυτοί οι παράγοντες μαζί είναι πολύπλοκος, και κάποιοι συνδυασμοί μπορεί να παρέχουν εκπληκτικά καλά αποτελέσματα.

Παρόλα αυτά, η διαδικασία υπερπαραμετροποίησης δείχνει πόσο σημαντικό είναι να εξερευνήσετε προσεκτικά τις επιλογές. Ακόμα κι αν δεν είναι πάντα απλό, αυτή η διαδικασία είναι κρίσιμη για να αποκομίσετε την καλύτερη απόδοση από το μοντέλο και τονίζει τον ρόλο της στην επίτευξη μιας καλύτερης απόδοσης του μοντέλου.

Η ενότητα για την επιλογή χαρακτηριστικών χρησιμοποιεί το DeepSurv για την αναγνώριση των πιο πληροφοριακών προγνωστικών για την ανάλυση επιβίωσης. Η μελέτη υιοθέτησε μια μέθοδο σταδιακής επιλογής χαρακτηριστικών, με στόχο τη δημιουργία ενός υποσυνόλου χαρακτηριστικών που μεγιστοποιεί την προγνωστική ακρίβεια του μοντέλου. Η διαδικασία επιλογής χαρακτηριστικών ενσωματώθηκε με διπλό σχηματισμό, εξασφαλίζοντας ανεκπλεκτή αξιολόγηση και μείωση του κινδύνου υπερπροσαρμογής.

Ωστόσο, ένα περιορισμός της χρήσης της βαθιάς μάθησης για την ανάλυση επιβίωσης είναι η δυσκολία στην ερμηνεία των αποτελεσμάτων. Τα νευρωνικά δίκτυα χρησιμοποιούν περίπλοκες σχέσεις μεταξύ των μεταβλητών, πράγμα που βελτιώνει την προγνωστική ακρίβεια, αλλά συνεπάγεται τη δυσκολία στην κατανόησή τους. Αυτό είναι ένα σημείο όπου η βαθιά μάθηση υστερεί σε σχέση με πιο παραδοσιακές μεθόδους. Η δυνατότητα ερμηνείας της εσωτερικής λειτουργίας των μοντέλων τεχνητής νοημοσύνης είναι μια τρέχουσα περιοχή έρευνας που υπερβαίνει το πεδίο αυτής της μελέτης.

Ως αποτέλεσμα, προτείνουμε μια συμπληρωματική προσέγγιση όπου παραδοσιακές τεχνικές συνδυάζονται με τη βαθιά μάθηση, αντί να τις αντικαταστήσουν εντελώς. Εάν μελλοντικές εξελίξεις στην έρευνα επιτρέψουν μια καλύτερη κατανόηση της "μαύρης συσκευασίας" των νευρωνικών δικτύων, ενδεχομένως θα αντικαταστήσουν στο μέλλον υπάρχοντα μοντέλα όπως το μοντέλο Proportional Cox Hazards.