Politecnico di Torino

Department of Mathematical Engineering

MSc Thesis

Machine Learning and Survival Prediction Models for Decision Support in the Private Equity Market

Candidates:
Marisa Hillary Morales
Vittorio Tiozzo

Supervisor:
Prof. G. C. Calafiore

A.Y. 2018/2019

A thesis submitted for the degree in

LM-44 Statistics and data optimization on networks

Abstract

The object of this thesis is the development of predictive algorithms for the support of investment decisions in Private Equity. In particular, the developed methodologies attempt to estimate the probability associated to a future state of a company, distinguishing among four possibilities: being acquired, go bankrupt, stay private, go public. Further, we model the evolution in time of the probability for a company to undertake an Initial Public Offering (IPO), which is one of the major events in the Private Equity market. The first objective has been realized by means of Machine Learning techniques, using Random Forest and Neural Networks models, in both the Multilayer Perceptron and Long Short-Term Memory configurations. The second objective was implemented by exploiting Survival Models, which are commonly used in the bio-medical environment, in particular Kaplan-Meier estimate, Cox models and the Accelerating Failure Times (AFT) model. Extensive numerical tests have been performed with R and Python, on the basis of an historical dataset available from Thomson Reuters.

Acknowledgements

We would like to thank Professor Calafiore for all his fundamental guide, both from a theoretical and a practical point of view. We would like to express our gratitude to Mr. Serge Marquie, for his invaluable guide throughout all this project, from its beginning in 2018 to this master thesis dissertation. Mr. Marquie provided us an exceptional insider point of view on the Private Equity market and in general on the financial environment, as well as countless occasions of thoughtful focus on the work done and the new enhancements to develop.
We would like to point out that this master thesis project has been supported by Eurostep Digital, both from the financial and the practical point of view, thanks to the precious suggestion of all its consumed financial professionals. It is due to thank them also for the trust that they have demonstrated in letting us design these experiments: we are extremely grateful for that.
Lastly, thanks to all who supported us in this work, especially to our families. It has been an amazing journey, with no lack of difficulties, spent with high level scientific and financial professionals, with which we evolved rapidly from a technical and personal point of view.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation of the thesis

The goal of the thesis is to design a predictive algorithm to drive investments made within the Private Equity (PE) market. Indeed, this work is the conclusion of a research started in 2018 aimed to explore the capabilities of the Machine Learning techniques in evaluating PE companies. The develop of this algorithm and the whole research work was supported by Eurostep Digital.
From a practical point of view, the Private Equity market is affected by a deep, chronic information asymmetry regarding the investment decisions. So, the algorithms designed attempt to exploit Machine Learning and Statistical Modeling in order to provide useful information to predict the future *outcomes* of investments on companies within the private market. These outcomes consist in the possible future "states" of the company: bankruptcy, public listing etc.

Before diving in the rigorous research questions, an introduction on Private Equity market must be provided in order to fully understand the practical domain of interest. Than, the objective and the mechanism of the algorithm are explained.

1.2 Private Equity market

A private company is a firm held under private ownership. Generally speaking, the fractions of these businesses ownership are called *shares* and are less liquid, so their valuations are more difficult to determine w.r.t. the public companies. Here, *"liquid"* equals cash reserve-based assets. In contrast, the ownership of a public company is spread amongst general public shareholders. This can be done through the free trade of stocks within public stock exchanges or over-the-counter deals market.

Most of the companies in the world start as privately held companies. Technically speaking, companies may be born from acquisition related events, but the large majority start as a privately held company. These companies range in size and scope: from the millions of individually owned businesses in the Europe and North America to the dozens of unicorn startups worldwide. Moreover, plenty of "giants"

worldwide players with upwards of $25 billion in annual revenue are privately held companies. Cargill, Ferrero, NM Rothschild and Sons, Deloitte and Ernst & Young are just some of the most well-known names. However, the investments raising for a private company are more difficult w.r.t. a public one, that is why many large private firms may choose to perform an Initial Public Offering (IPO). The IPO is the process of offering shares of a private firm to the public market through a new stock issuance.
The advantages of going public are evident: instead of having access just to bank loans and specific types of equity funding, public companies are able to issue and sell new shares through stock exchanges platforms or raise money by means of debit selling (bond offerings). These latter mechanism are faster than the first ones and are potentially able to raise a larger amount of liquidity, since they are reachable by the larger audience of the global investors.

A question arises naturally: **Why companies stay Private instead of undertaking an IPO?**
Companies born Private, then they may be Acquired, submit an IPO or go Bankrupt. Actually, a forth possibility exists, which is remaining Private.
There are several reasons for remaining Private. First of all, undertaking an IPO involves high costs because it requires one or more investment banks who arrange the shares to be listed on the stock exchange. Second, public companies have to perform more data disclosure and must publicly release some reports such as financial statements, on a regular schedule. These filings include annual reports (10-K), quarterly reports (4-Q), major events (8-K) and proxy statements. Last, some companies desire to maintain family ownership. Nowadays, many of the largest private companies have been owned by the same families for multiple generations, such as the aforementioned NM Rothschild and Sons, which is still property of the Rothschild family since Nathan Mayer founded it in 1811.
Moreover, public companies have to take into account its public shareholders decisions for the company's strategy and administration, as an example they may decide different members for the board of directors. This can be circumvented by the companies in two ways: by keeping the majority of the shares (i.e., to issue stocks for a value smaller than 50% of the total company value), or by issuing stocks that do not provide to the holders any decision power.
Going public can be thought as a final step for private companies. The IPO's benefits don't come without costs: undertake an IPO costs money and takes time for the company to set up. However, as for the investors point of view, this event would be enormously rewarding, so being able to tackle the connected problem (i.e. evaluate the future of a privately held company) would be really valuable.

1.2.1 Size of the PE market

The value (size) of the Private Equity market is shown on Fig.1.1 across the last 20 years. Its size is still much smaller in term of market capitalization w.r.t. the public stock exchanges. While the biggest stock exchanges are roughly 20 times the

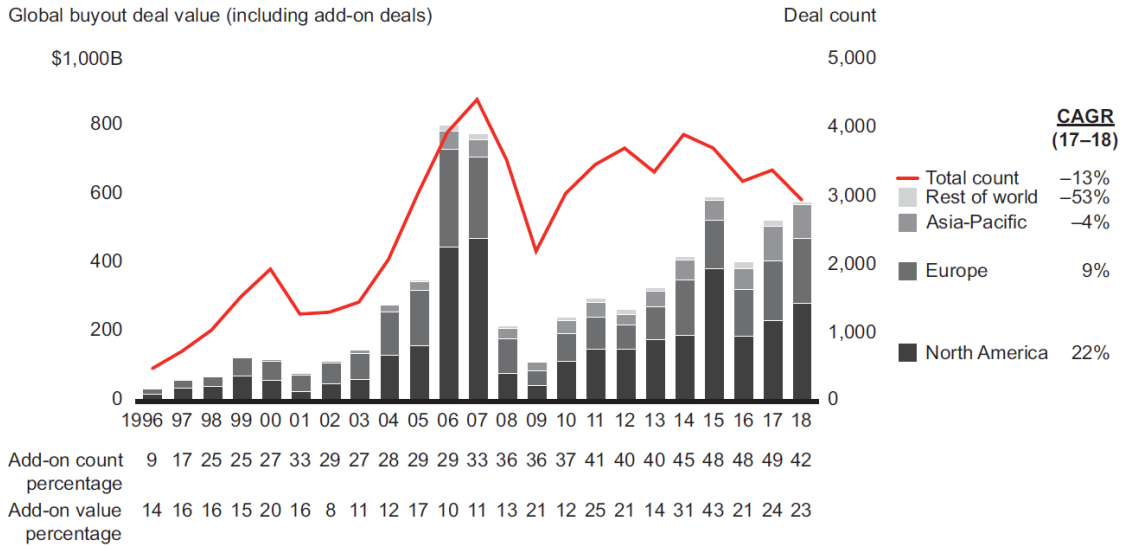

Figure 1.1: Private Equity market history since 1996. The CAGR simbols refers to the Compound Annual Growth Rate. Excludes loan-to-own transactions and acquisitions of bankrupt assets.

	Value (USD bn.)
New York Stock Exchange	22,923
Japan Exchange Group	5,679
Borsa Italiana S.p.A.	688
Global Private Equity market	**582**

Table 1.1: Global markets values. Sources: Wikipedia [1], Bain & Company [2]

volume of the PE market, the latter value is comparable to the one of the Italian exchange (Tab.1.1).
However, the private market registered an impressive increase in investment value in 2018 (black and grey bars in Fig.1.1). In 2018, because of the harsh competition among PE investment funds and the rise of companies shares prices, the number of individual transactions decreased by 13% w.r.t. 2017 (red line in the same plot). Nevertheless, the total buyout value (total investments value) arose to a 10% increasing, with $582 billion (including add-on deals), so completing the *"strongest five-year run in the private market history"* [2].

1.2.2 Private Equity investments & Information asymmetry

Within the PE market, any investment consists in the acquisition of a certain amount of share of a certain company, performed within a **deal**, an investment round in which a certain number of investment funds participate to the acquisition. Usually, the access to this round is not open to the general public: only big PE funds (sometimes part of large investment banks) or subjects closely related to company are able to get into the deals.
The investment evaluation equals to the evaluation of the company itself, in order to build an investment strategy able to meet the fund requirements in terms of per-

formance, volatility etc. As the PE market is increasing in volume, the main issue private equity investors are struggling with has not changed in decades: the absence of transparent, easily accessible valuation-related information. Actually, the investors, looking at possible investments in privately held companies are lacking information that are needed to build an effective investment case (e.g. financial information about the company, deal value of previous rounds etc.), for this reason investors resort to methods such as portfolio diversification to compensate this lack of company specific information.
Moreover, Private companies are able to sell their own shares through stock issuing, but these shares do not trade on public exchanges. As a result, they do not have to pass through the rigorous Securities and Exchange Commission's (SEC) filing requirements as public companies do.

In light of these issues, PE investors critically need structured, quantitative methods in order to infer basic future performance measures for their investments: the prospect for a company to go IPO in the future, the value of a private company as a potential acquisition target or the probability of a company to go bankrupt. Each of these possible outcomes correspond to very different investment returns. If an investor owns share of a company that is about to go bankrupt, the investment associated equals to a large loss. On the other hand, the return associated to a pre-IPO investment will be extremely big. How big?
Contrary to many conventional market strategies, successful investments in PE typically have extremely high rates of return: Peter Thiel's investment in Facebook in 2004 ($500,000) appreciated 693,3% by the time of Facebook IPO [4]. Softbank investment in Alibaba in 2009 appreciated 290,0% by the time of Alibaba IPO [5].

As a result, the combination of information scarcity and potential outsized returns, makes any performance forecasting indicators, even ones with marginal forecasting power, extremely valuable for private equity investors.

Investments Dynamics

The first investments a private company experienced provide some of the first and richest available information. Typically the newborn companies go through successive investment rounds (seed round, series A, series B, etc.) to which various types of investors can participate: Angel investors, Venture Capitalists, Private Equity funds, each with their own goals, expertise and history. The future developments of each company will deeply depend on these early investors, since the latter often have significant participation, receiving one or several board seats. So, this investment dynamics suggests that parameters such as the nature and the composition of a private company early investors contains significant information regarding the future of that company and therefore of the investments in it.

1.3 Research questions

Before stating the aims of the research, let's recap the key features of the Private Equity market and its main needs in terms of investment decision making. These have driven the decisions taken for this work.

1. **Expanding market:** The Private Equity market sees a constant increasing in volume since 5 years from the present day.

2. **Information asymmetry/Lack of information:** PE investors suffer from a total dramatic lack of information that are needed to evaluate private companies (i.e. the PE investments). Since these companies are private, they do not have to provide extensive information about their performance and/or their financial status.

3. **Lack of data driven strategy:** Consecutively, this absence of information affects investment's strategies that are based to the real world data. If the intelligence comes from knowledge, nowadays PE investors decisions lack of intelligence since they lack of knowledge on the companies statuses.

4. **Extreme investments returns variance:** Investing in companies that go bankrupt equal to a close to total loss of the investment itself. Instead, investments in private companies that will go IPO bring astonishing returns (most of the cases over 100%).

In light of all these, the research questions investigated have been:

I. **Private companies future statuses prediction.** The Random Forest algorithm and the Neural Networks framework have been explored and designed to output a probabilistic estimate of a company's future status.

II. **Time-to-IPO prediction.** The probability in time of the event IPO has been modeling using a Survival Analysis approach. In the end, the algorithm designed produces a company's *survival curve* representing probability of going IPO within each time point.

Given the practical needs enumerated above, these two questions are very valuable among the Private equity investors. The first one can drive the decision of which company investing in. The second one provides an extra information, that is the probability in time of a public offering: the rare, precious event that corresponds to the very high investment returns as described above.
From the mathematical point of view, automatic classification algorithms are plausible candidates for offering a solution to the problem of predicting private company's future performance. Indeed, Machine Learning algorithms are able to process large quantities of both qualitative and quantitative data within the fitting (*training*) phase. Their efficiency has been proved to produce accurate forecasts in areas as diverse as bio-statistics [6] or corporate finance [7]. However, within the PE framework, due to the pathological lack of investments related data, it is difficult to expect

high accuracy from any forecasting method, both algorithmic and human based.
As for the survival models, at the present day, this is the first study to use them within the Private Equity framework, even if their capabilities has already been explored in the economic field [8], [9].

1.3.1 Previous results

Despite the need of a analytic, data-driven approach to PE investment decision, the results in literature are still rare.
The only relevant results found has been the ones of the analytic branch of the Silicon Valley Bank (SVB), based in San Francisco. This is probably due to the bank location, at the heart of the global hi-tech and PE market: the Silicon Valley.

Particularly, two articles served as starting point of this research.
The first one explored mainly a *LASSO* regression model in order to forecast privately held companies financial performance [10].
The second is related to a work aimed to exploit the Random Forest algorithm to predict the future status of private companies [11]. The input data consisted in the names of investment firms that invested in each company, a choice pushed by the lack of information that affects the PE market: often, the only deal information available are indeed these names.

Although the quite high predictive accuracy reached by Bhat and Zaelit (B&Z), the performance has been very inhomogeneous across the metrics, resulting in a strong bias toward the negative class. However, because of the good quality of the paper, it has been chosen as the starting brick for the work, so it is worth to dive deeper in this work.

Bhat and Zaelit's framework

B&Z approach consists in a standard machine learning algorithm which has proved extremely efficiency in many similar finance related subjects, the Random Forest. In order to develop a forecasting framework, their work relies on a dataset extracted from the Reuters financial provider platform, *Thomson Eikon*.
The reference dataset consisted on the information about investment rounds of private companies happened between 1996 and 2011. The B&Z results have been replicated by the authors and expanded using similar data provided by the same platform. Obviously, the understanding of the exact composition of B&Z dataset is limited, since it is sensible and valuable information for SVB. The differences between B&Z results and the one obtained in the replication are probably due to the differences in the reference dataset retrieved. The full replication of the article and the comparison of the results achieved with the results of the article can be found in the Annex1.

After extensive tests conducted on the B&Z replication results which are out of the scope of this thesis, the reference dataset has been deeply changed in the events

time horizon, but the base covariates structure have been kept as they provided robust results and are easily retrievable for any PE investor, as it will deeply discussed in Chapt.2.

1.4 Structure of the thesis

This thesis is divided in two main parts, corresponding to the two research questions (Sec. 1.3):

1. **Estimation of the future status of a company.** This part has a double goal: to design a Machine Learning classification algorithm and to provide a basis for the Survival Analysis, as it would be explained in Chapt.4. Random Forest & Neural Nets are the two classifiers exploited. As for the RF, it had the aforementioned work of B&Z as building brick, while the NNs algorithms has been designed with no previous state of the art guidance. Two types of NNs have been explored:

   (a) **Multilayer Perceptrons (MLP).** The simplest net architecture available, so the most parsimonious in terms of number of parameters and training time. And that will be presented in Chapt.3.2.

   (b) **Long Short Time Memory (LSTM).** The state of the art type of net. Much more complex than the MLP, usually used for time structured data. Due to its related theory, the dataset has been manipulated as it will be described in Sec.3.5.1.

2. **Estimation of the probability in time of the IPO event.** A survival analysis procedure has been performed (Chapt.4). First, the Kaplan and Meyer *curve* has been described and designed 4.3. Then, the probabilistic and statistical properties of the IPO process are studied in order to highlight the survival model that fits best this framework. The Accelerated Failure Time models are chosen after their base hypotheses have been validated. Finally, they are tuned in order to maximize the fitting quality, exploiting their statistical properties and building an algorithm that computes the survival curve for each new input company 4.5. Moreover, the algorithm will be able to highlight, for each industrial sector, the most important *prognostic factors*: the covariates that, according to AFT models, are able to statistically *accelerate* or *decelerate* the time to IPO of a company 4.5.4.

Above the industrial sector company classification is mentioned. Even if this feature will be described in the next chapter, it is worth to point out that all the analysis and fittings produced through the Survival methods have been performed *sector-by-sector*: considering the fitting companies separately according to this category.

To briefly summarize, in the next chapter a deep explanation of the data source,

the dataset and the data pre-processing are provided. In Chapter 3, the Machine Learning models algorithms design process are presented: first the RF theory and the NNs one, then the tuning process and to conclude the comparison between their final predictive performance. Chapter 4 concerned the Survival Analysis exploration and finally in Chapter 5 the conclusions and global thoughts on this work will be drawn.

Chapter 2

Reference Dataset

In every data science work, the quality of the data is crucial to build an effective and consistent predictive capability. As aforementioned, this particular application domain, the Private Equity market, suffers from a global lack of information. This has been a major constraint both from the data source finding and for the type of variables to use as predictors.

The data source for this research is the Reuters "Thomson Eikon" (TE), a professional platform of financial analysis data, successor of "Venture Xpert". In particular, the data has been extracted from the Private Equity dedicated application of TE. Because of the structure of the Survival Analysis, the companies population analyzed has to be homogeneous w.r.t. the starting point in time of companies *life*. In fact, it is composed by the companies founded from 1998 to 2018, independently from the date of their investment rounds. the information of the first 3 investment rounds are retained. As for the geographic location, the dataset represents the Global Private Equity market: the selected companies have their headquarters in the Americas, Europe and Asia.

Before using the data for the automatic classifiers training and the survival analysis fitting, a deep manipulation was needed. This was performed in 2 parts:

1. **Data cleaning and labeling.** The base operation for any data science work. After a mindful reflection on the data issues, the data points that presented "NA"s (Not Available) values within their features or evident errors within the data registering process (e.g.: companies foundation dates after their first investment round dates) were removed. Then, the company status labeling procedure was performed according to the B&Z article [11] and to the application domain.

2. **Investments modeling.** The transformation from qualitative (investors' names) to quantitative data, for each company. This was performing according to [11].

It is worth to remark that before any data cleaning process, the dataset resulted in 101616 distinct company names.

2.1 Data pre processing & Labeling

The features extracted from TE platform, for each company, are:

1. Company Name.
2. Company Founded Date: in the "dd/mm/yyyy" format.
3. Deal Value: the total amount of US Dollars invested in each investment round. The value 0.00 corresponded to a not available deal value for that round.
4. Firm Name: the name of the investment firms that participated in any of the investment rounds of the company. It consisted in a list of names.
5. Fund Name: similar to the previous one, but it refers to the name of firm's fund that performed the deal.
6. Company IPO Date: in the "dd/mm/yyyy" format. The exact date of the company shares Initial Public Offering.
7. Investment Date: for rounds 1, 2, 3, in the "dd/mm/yyyy" format.
8. Company Status: categorical value, expressing the *present* status of the company. The available categories are:
   (a) Bankruptcy related: *"Defunct"*, *"Bankruptcy - Chap. 7"*, *"Bankruptcy - Chap. 11"*.
   (b) IPO related: *"Went Public"*.
   (c) LBO: *"LBO"*.
   (d) M&A related: *"Acquisition"*, *"Merger"*, *"Pending Acquisition"*.
   (e) Private related: *"Active"*, *"Other"*, *"In Registration"*, *"Private Company"*.

   This nomenclature will be described more in Subsec.2.1.2.
9. TRBC Economic Sector: a Thomson Reuters crafted classification of the economic sector the company operates in.
10. Company World Location: the location of the company's headquarters.

Some sample rows of the extracted dataset are represented in Fig.2.1.

2.1.1 Data cleaning

The data cleaning process regarded mainly the missing value found for the most important variables: the investment firms names, the foundation date and the IPO date. These variables were fundamentals both for the Machine/Deep Learning algorithms and the Survival Analysis.
The main problems encountered were:

1. Missing foundation date.
2. Missing IPO date, even if the company status was "Went Public".

	Company.Name	Company.Founded.Date	Deal.Value...USD.	Investment.Date
25	United Emergency Services, Inc.	01/01/1996	0.00	01/10/1998
26	United Leasing & Financial Services	<NA>	0.00	01/01/1998
27	United National Bancorp	01/01/1902	0.00	01/04/1998
28	United Online, Inc.	21/07/1997	0.00 \n0.00 01/10/1998 \n01/12/1998	
29	United Pet Group, Inc.	01/11/1997	0.00	14/08/1998
30	United States Technology Inc	01/07/1998	0.00	01/07/1998

	Firm.Name	Fund.Name
25	Cordova Ventures	Cordova Enhanced Fund, L.P.
26	Capital Investments Ltd	Capital Investments
27	Banc Funds Company LLC	Banc Fund III Trust
28	Clearstone Venture Management Services LLC	Clearstone Venture Partners - Unspecified Fund
29	TA Associates Management LP	Advent Atlantic & Pacific III \nTA/Advent VIII
30	Goldman Sachs & Co LLC	GS Capital Partners III, L.P.

	Company.IPO.Date	Company.Status	TRBC.Economic.Sector	Company.World.Location
25	01/08/2006	Acquisition	Healthcare	Americas
26	<NA>	Defunct	Financials	Americas
27	31/12/2003	Acquisition	Financials	Americas
28	23/09/1999	Went Public	Consumer Cyclicals	Americas
29	02/08/2004	Acquisition	Consumer Cyclicals	Americas
30	<NA>	Active	Technology	Americas

Figure 2.1: Sample rows of the reference, uncleaned dataset. Please note that the *Deal Value*, *Investment Date*, *Firm Name* and *Fund Name* columns can contain more than one value (i.e., a list). In that case the symbol $\backslash n$ separates them.

3. First investment round date before the company foundation date.
4. IPO date equal to the foundation date.

Actually, just the first two issues affected a substantial number of companies, as reported in Tab.s2.1, 2.2.

Variable	# missing values
Company.Name	0
Company.Founded.Date	23594
Deal.Value.USD	0
Investment.Date	0
Firm.Name	24
Fund.Name	24
Company.IPO.Date	64398
Company.Status	1
TRBC.Business.Sector	70
Round.Number	0
Company.Nation	0
Company.World.Location	0

Table 2.1: Number of missing values in the raw dataset. Please note the high value of missing foundation dates, equal to approximately 23% of the total.

The solution for the data issues presented above consisted in deleting the companies that presented them. Since the issues themselves regarded the time reference of companies events (particularly, the foundation date), any NA-value-substituting technique would have been poorly effective. Moreover, as it will be clearer in Chapt. 4, Survival Analysis is deeply affected by missing events time references.

Moreover, it is worth to point out that some companies presented no third investment date, but some investment firms in that round. In those cases, the investment

	# "Went Public" status companies
With IPO date	4783
Without IPO date	3356

Table 2.2: Number of IPOs companies with missing IPO date.

date has been set to 2 years after the second one, following to a statistical market metric and practitioners experience.

2.1.2 Company status and labeling

The "Company.Status" covariate is fundamental for this work, since it consists both in the classification target of the RF and NNs and in the final status label whose Surival Analysis is based on. That is why it has been of main focus during the initial data cleaning part.
The 12 possible labels that TE showed were reshaped into a 4 labels set:

1. "Defunct", "Bankruptcy - Chap. 7" and "Bankruptcy - Chap. 11" became *Bankrupt*.
2. "Went Public" became *IPO*.
3. "Acquisition", "Merger", "Pending Acquisition", "LBO" and "M&A" became *Acquisition*.
4. "Active", "Other", "In Registration" and "Private Company" became *Private*.

This choice is based on [11], since it is based on a reliable source on the application domain (the SVB experience). It is worth to remark that acquisition-related events are often referred as *M&A*s ("Merger and Acquisition"), a type of process in which two or more companies are combined together. Within an Acquisition process, one company buys at least the majority of the shares of another one, in order to have full control on it. Merger is similar, but the combination of the two firms does not presuppose a subsidiary relationship: the company resulting is a new legal entity under the banner of one corporate name [1].

Actual status relabeling

An interesting issue that came up is that a consistent number of companies within the raw dataset presented an IPO date in the "Company.IPO.Date", but the "Company.Status" column contained a label different from "Went Public".
At a first glance, this sounded strange. However, a plausible interpretation was found: the companies with this status-IPO date mismatch have been acquired by a public company, so they show a "Went Public" date even without experiencing a real went public-related event. So, these companies were relabeled as *IPO* and this will ease the reliability and fitting quality of the Survival Analysis, since it augmented considerably the number of IPOs in the dataset. This hypothesis is corroborated

[1] https://www.investopedia.com/terms/m/mergersandacquisitions.asp

by the high number of "Acquisition" labeled companies that showed this issue, as Tab.2.3 testifies. Actually, even some bankruptcy-related companies showed an IPO date, probably due to different acquisition processes.

Indeed, this choice is driven also by the goal of the analysis. As aforementioned, when privately held companies go IPO, the return of the investments for the initial shareholders is extremely high. On the other side, when a private company gets acquired by a public (and probably much bigger) one, the investment return is again very consistent. That is why considering the latter process as an IPO, from a practical point of view, does not cause any loss of generality also from the modeling side.

	# companies
Acquisition	15136
Active	284
Bankruptcy - Chapter 11	135
Bankruptcy - Chapter 7	152
Defunct	806
In Registration	120
LBO	11608
Merger	594
Other	13
Pending Acquisition	1522
Private Company (Non-PE)	4
Went Public	3356
Total	**33730**

Table 2.3: Number of companies with an IPO date, raw dataset. Please note the consistent number of "Acquisition" labeled companies.

2.1.3 Business Sectors

Generally, the economic activity is analysed through a classification that relies on the industrial sector in which each company operates in. Even if there exists a lot of classifications, one of the most reliable and complete one is the North American Industry Classification System (NAICS), that classifies the business activity using a 6 digits code [2]. It is the one used within the B&Z work and it classifies industries in 9 main sectors: Communications, Computer, Electronics, Biotech, Medical, Energy, Consumer, Industrial, Other. Each sector corresponds to the first digit of the code (e.g. "1xxxxx" stands for "Communications")
However, Thomson Eikon data source does not provide it in this numeric fashion, but with a text description that is difficult to handle for large datasets.

[2]https://www.census.gov/eos/www/naics/

In order to solve this issue, within this work the Thomson Reuters Business Classification (TRBC) code is used instead, but it is *translated* into the 9 NAIC business sectors. Even if this code is again a textual one, it is much less granular than the NAIC code reported on Thomson Eikon, so it was possible to translate it to the classic 9 sectors one through a text mining procedure.

2.2 Investments model & representation

This research, as the ones by B&Z previously cited, before designs some algorithms, boils down to model a network system: the Private Equity investment market. The representation of economics systems though networks it is widely recognized [12], [13] and it has indeed a lot of features in common with the social systems studied within the Network Dynamics theory. Each of the agents, the PE investors, is influenced by the decision of the other ones, all the agents act to reach the maximum utility and they compete (or collaborate) in the same framework, the PE market. That is why it comes natural to describe the PE investors as a network and to perform a *centrality* measure on it. In particular, an *Investors Ranking* was computed, relying also on the Random Forest B&Z article aforementioned. This procedure allowed to translate qualitative information to quantitative one, to be used as covariates for the automatic classification and the Survival Analysis.

Investors ranking computation and motivation

The Investor Ranking computation was performed as the following:

1. The investors who participated in the rounds of the reference (cleaned) dataset were listed in a vector inv_vector whose length is l.
2. The investor matrix I is initialized with a 0 matrix whose size is $l \times l$
3. For each investment round, if investors i and j are present, the (i, j) position of I is updated with a "+1"
4. The principal diagonal of I is forced to 0 and the row-sum is performed. Each sum corresponds to a value in inv_vector, that is the number of investments round each investor has participated in.
5. inv_vector is sorted in ascending order, and a rank r_{inv} is assigned to each investor.

Finally, for each company, its investors are aggregated on a list which will correspond to a set of investors ranking, translating the qualitative information (a list of names) in a quantitative one (a set of integers). As aforementioned, this set will represent the companies covariates used in this whole work.

This procedure relies on the graph interpretation of the PE market, whose I is the *adjacency matrix*. In fact, it is recognisable as a simple *degree computation*, performed on the graph that I represents. So, it comes natural to question if it models accurately the PE market, i.e., if this market is well represented by the Investors

Ranking.
In order to check this, the real-world performance (and industry reputation) of the top 10 Investors Ranking funds was explored (Tab.2.4). Indeed, it came out that they are really among the most important Private Equity investments firms on the global market.

Ranking	# investments	Investment Firm
1	243207	UndisclosedFirm
2	13912	NewEnterpriseAssociatesInc
3	12070	SVAngel
4	10508	IntelCapitalCorp
5	9265	U.S.VenturePartners
6	9148	3iGroupPLC
7	8436	DraperFisherJurvetsonInternationalInc
8	8360	AndreessenHorowitzLLC
9	7721	KleinerPerkins
10	7342	AltaPartners

Table 2.4: Top 10 investors ranking. Please note that the "Undisclosed Firm" has the highest rank, since in a large number of cases the name of the investor was not disclosed.

Actually, another strong rationale backs the usage Investors Ranking computation: on the long run, only the successful investors survive, i.e. the ones that performed sustainable returns investments (non-bankruptcy related, high performing private companies, IPOs etc.). So, if an investor performed an high number of investments, it is probably a strong performer within the market. This hypotheses has been verified also from the application domain point of view.

Lastly, from Tab.2.4 it is worth to note the large number of companies who had among their investors some firms that preferred not to be mentioned, registered by Thomson Eikon as "Undisclosed Firm". This is a peculiar feature of the Private Equity market: unlike the public one, the investors do not have to declare the holding of private companies share.

2.2.1 VIX index

The covariates setting originated by the B&Z work has been indeed enriched with a new type of information: a public market indicator within the investment information.
In fact, based on the application domain, the Private Equity market performance (i.e., the return of PE investments) is linked to the public market one. Indeed, it is widely understood that the stock market has deep influence on each aspect of the economic environment, in which obviously also the privately held companies must operate. That is why the Chicago Board Options Exchange (CBOE) Volatility Index ® (VIX ® index) has been used has a covariate. Specifically, the dates of investments rounds 1 and 2 have been joined with the VIX value of those dates.

The VIX value associated to third round was not included, since the weight of the market is inferred to be stronger for younger companies w.r.t. the more stable ones.

This index value is obtained through a measure of the 30-day expected volatility of the U.S. stock market. The base data are derived from call and put options whose underlying assets belong to the S&P 500® Index (SPXSM). For these characteristics the VIX index is recognised as a reliable measure of market volatility[3]. Roughly speaking, its value represents the degree of market uncertainty: the higher the value, the more the market is going through a *nervous moment*.
Why a volatility measure for this work? Because it is reasonable that the degree of the public market uncertainty has a strong influence on the willingness of investors to opt for the PE market, seeking "shelter" from the high level of volatility.

2.3 Final Dataset description

In the end, after the cleaning from all the data issues mentioned above, the final dataset is described in Tab.2.5.

Sector	Bankrupt	IPO	Acquisition	Private	**Total**
1 Communications	165	1084	49	856	2154
2 Computer	672	4926	144	11383	17125
3 Electronics	151	940	28	1734	2853
4 Biotech	84	1465	24	2704	4277
5 Medical	50	1279	13	2599	3941
6 Energy	5	492	10	852	1359
7 Consumer	82	1851	38	3444	5415
8 Industrial	108	2285	40	3755	6188
9 Other	96	1605	28	2866	4595
Total	1413	15927	374	30193	**47907**

Table 2.5: Status distribution across business sectors.

It is easy to note the severe unbalancing of the classes. Bankrupt label is approximately 3% of the whole dataset, while the Acquisition label do not even reach the 1%. This will be a major issue for the Machine Learning algorithms and so deeply discussed in Chapt.3.

An interesting descriptive statistics is represented by the companies foundations dates frequency (Fig.2.2). The plot highlights two important facts: there exists a periodicity of the foundations dates and this is common to almost all sectors. In fact, sector 1 (Communications) shows a strong peak in the 1998-2000 period, as the sector 3 (Electronics). Moreover, sector 6 (Energy) presents an increasing in frequency up to 2005 and sector 2 (Computer) an increasing from approximately 2002 to 2012, probably due to the relative *youth* of this industry.

[3]http://www.cboe.com/vix

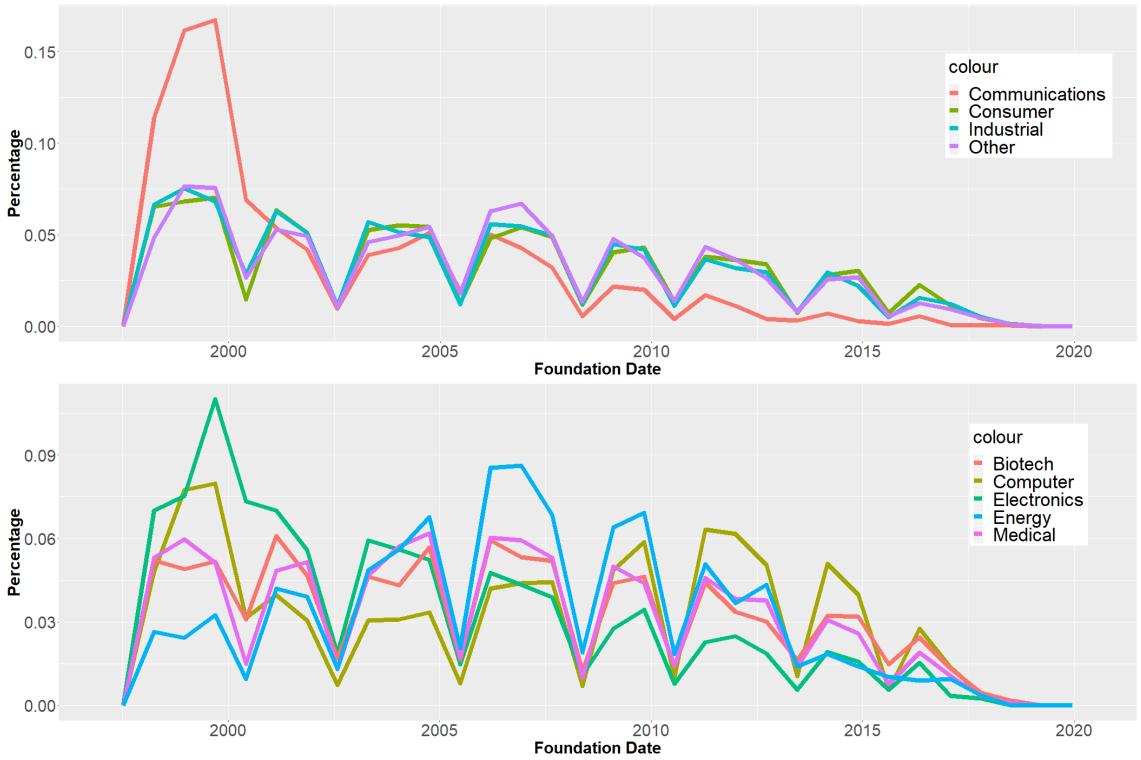

Figure 2.2: Foundation date normalized frequencies for the 9 business sectors.

It is worth to note how the frequency lines are subject to a constant decreasing towards 2018: much less company are founded w.r.t. the previous years. This should be compared with the Private Equity market growth in the last 3 years described by the Bain Report [2], depicted in Fig.1.1. In fact, it suggests that the investment rounds happened in the last period do not regarded young companies, but the more stable ones, whose market positions are probably more engraved than the younger companies ones. On the other hand, it could be a data quality issue relate to Thomson Eikon platform.

These market-related observations have to be considered in the commentary and conclusions related to the modeling results, in order to check if the mathematical tools outputs are coherent with the classical, practitioners descriptive ones or they highlight a different point of view. Moreover, the focus on a possible data quality issue have to be kept throughout all the work.

2.3.1 Final covariates

Finally, it is worth to enumerate the final covariates that will be used throughout the whole analysis. As aforementioned, for each company these variables are aggregated measures of the Investors Ranking (Sec. 2.2) related to investors who participated in the first 3 company rounds.
They are:

1. MeanX: average investor ranking at round $X \in \{1, 2, 3\}$.

2. MinX: minimum investor ranking at round $X \in \{1, 2, 3\}$.
3. MaxX: maximum investor ranking at round $X \in \{1, 2, 3\}$.
4. NumX: number of investors involved at round $X \in \{1, 2, 3\}$.
5. VixX: VIX index value at the time of round $X \in \{1, 2\}$.

2.4 Comments: why qualitative data?

Within the financial environment, quantitative data are widely used as a fundamental basis for prediction algorithms.
However, as broadly discussed in the first chapter (Subsec.1.2.2, Sec.1.3), within the specific Private Equity market the information asymmetry makes the quantitative approach unfeasible. The data about the investment amount in each PE deal and traditional financial key performance indicators are rarely available for privately held companies.
That is why in this work only qualitative covariates related to companies are used to build the analyses. Even if this choice brings some complexity to the whole work (need for a qualitative-quantitative transformation), it produces a double benefit:

1. The number of data points is as large as possible, since investors names are available for barely all the raw dataset (Tab. 2.1).
2. From an end-user point of view, the data needed to perform a prediction on the future company status and time-to-IPO will be easily available. If quantitative covariates, such as the invested amount in each company round or the company financial statements, were needed, probably a large number of users would not be able to gather these information.

Chapter 3

Outcomes probabilities estimation

This chapter focuses on the forecast of the future status of a private company. These statuses have been grouped in 4 possibilities: private, public (often written as IPO), acquisition and bankrupt. The goal has been achieved with two different machine learning techniques: the Random Forest (RF) and the Neural Networks (NN). As for the Neural Networks, two different typologies have been explored: MultiLayer Perceptrons (MLP) and Long Short-Term Memory (LSTM). Both algorithms provides a vector of probabilities, one for each possible class, so to end up with a classification, the maximum probability decision rule has been applied. This rule simple states that the label assigned to an observation has to be the one which results with the higher probability. All classifications have been evaluated with different performance metrics and compared with each others.
The chapter is structured as follow: a theory part which exploits both algorithms, an experiment section with the tuning procedure adopted and to conclude, a comparison of the results obtained. Please keep in mind that within this chapter the terms *"outcome"*, *"status"*, *"exit"* and *"label"* are used interchangeably.

3.1 Random Forest theory

The Random Forest is an ensemble tree-based method which can be used for both regression and classification problem. The basic element which composed this method is called *tree* since it partitions the predictors' space into small regions within a tree analogy which will be described later. The ensemble is than created *"growing"* a certain number of trees and the predictions of all trees are merged in order to provide a single consensus prediction. This method is based on the uncorrelation of the forest's trees in order to end up with multiple, different and independent splits of the features' space. Consequently, this means that predictions may differ from tree to tree.

Single decision tree

The theory behind the partitioning of the predictors' space is that at each step the algorithm evaluates different splits for each predictor and then it chooses the split

which minimize an error metric (e.g. Gini index measure, RSS). More precisely, for all predictors X_1, X_2, ...,X_j and all possible values of the cutpoint s the error measure for the two new regions of the space is computed. In the end, the pair (j, s) which minimizes the error is selected.
The procedure is repeated on the previously split predictors' space. The process continues until a stopping criterion is reached (e.g. no more than n observations for each terminal leaf, a level of purity for the terminal nodes, etc.)
Once the tree growing is ended, the new observations are classified according to the majority classification of the terminal nodes (*leaves*) whose these new points belong to.
This procedure is surely enough greedy, which means that the algorithm select the best split looking for a local minima, without evaluating if another choice could lead to a global minima in the next steps.

Algorithm 1 Building of a generic decision tree

1. Define decision criteria (e.g. RSS to minimize)
2. Define stopping criteria
3. Define region set space R
P = predictors set
S = cut points set
while stopping criteria **do**
  t = 0
  for $(j \in P, s \in S, R_i \in R)$ **do**
    $score_{js} \leftarrow$ decision criteria(j,s,R_i)
    if t = 0 **then**
      min_score $\leftarrow$ $score_{js}$
      j_min $\leftarrow$ j
      s_min $\leftarrow$ s
      R_min $\leftarrow$ R_i
      t = 1
    else
      if $score_{js} <$ min_score **then**
        min_score $\leftarrow$ $score_{js}$
        j_min $\leftarrow$ j
        s_min $\leftarrow$ s
        R_min $\leftarrow$ R_i
      end if
    end if
  end for
  1. Apply splitting (j_min,s_min) on region R_min
  2. Update set of regions
end while

Ensemble of trees

The Random Forest (RF) used in this analysis is a classification algorithm, which means that it is made of a set of classification trees. All trees have been built using the same observations, but due to the inherent randomness of the construction process (i.e. choosing of the features at each split), they result in a different partition of the space. This leads to a possible different classification of the same object by two different trees. Once a number n of trees is grown the RF predicts a new object's class based on the response of the majority of trees.
Example: There are 10 trees. The new point i is injected in the forest. If 7 trees classify i as "A" and 3 trees classify it as "B", then the object is labeled as class A.

3.1.1 Hyperparameters

The most significant hyperparameters of the RF are two. The first is the number of trees which composed the forest. The second is the number of predictors m_{try} that will be randomly picked at each split when the tree models are built.
The number of trees has been tuned considering the Out-of-bag (OOB) error, while for the number of predictors to use at each step it has been set equal to the square root of the total number of predictors.

OOB error

The OOB error is the error rate of the so called out of bag classifier on the training set. The procedure to calculate it consists in create a set of boostrap datasets which do not contain a particular record. This set is built starting from the training dataset removing at each time a particular observation and retaining the rest of the data. This is called out-of-bag set of examples and supposing that there are n observations in the training dataset, there are n of such subsets (one for each data record). The OOB error is a very interesting technique because it removes the need of a validation set. In fact, empirical evidence shows that the out-of-bag estimate is as accurate as using a test set of the same size as the training set [14]. Anyway, this is the *leave-one-out* OOB configuration: different fitting-validation set proportion can be used instead.

3.2 Neural Networks Theory

Neural Networks (NNs) are systems of interconnected artificial neurons (set of algorithms) which are modeled to recognize patterns. These systems can interpret, label and cluster raw data.
First, it is worth to note that these patterns must be numerical in order to be *learnt* by Neural Nets. They can be the numerical representation of whichever data can be translated in such fashion: images, time series, audio and video, text, representations of environments, markets etc. Usually, the way this translation is performed has a strong impact on the learning capability of NNs, so it has to be carefully evaluated and designed.

NNs are able also to perform automatic features extraction, which can feed other algorithms (or NNs) for clustering and classification purposes. Within such framework, it is possible to think about neural networks as components of larger Machine Learning algorithms.

An artificial neural network (ANN) is a structure of nodes and links that forms a directed, weighted graph.
Typically nodes are grouped in *layers* that can have different dimensions. All the nodes in a layer are equipped with the same *activation function* that can be thought as a layer's property and is often non-linear (Subsec. 3.3.1).
An input, which is a collection of observations, is provided to the input layer (i.e. to each node of the input layer). Each layer of the net performs an element-wise *affine transformation*, whose elements are computed within the network *neurons* (the nodes of the graph). That is, for each neuron, the output is computed according to the layer transformation and the activation function is applied. In modern NNs, because of the shape of the activation functions, the neuron's output value is propagated to the next layer only if it is over a certain threshold. The neuron's computed value is often referred as its *internal state*, since its propagation depends on its own value. The final layer (output layer) computes a response based on the output of the last hidden layer.

The training process consists in a numerical optimization problem whose aim is to find the set of network's *weights* that minimizes the classification or regression error, depending on the net response type. Within simple types of NNs, these weights consist in matrices and vectors that build the aforementioned affine transformations, but they can also drive the complications of more complex networks, such as the Long Short-Time Memory ones (Sec. 3.5). This optimization problem and the related solution techniques will be extensively described in Sec. 3.4.

From the architecture point of view, it is important to highlight that the structure of the graph's connections depends on the network type: fully connected (FeedForward Networks), sequential (Recurrent Neural Networks) etc. (Fig. 3.5). Each type of NNs (so, each type of graph), induces different properties of the network output and representation capability, along with the neurons types. Indeed, while Feed Forward Networks (FFNs) can handle whichever regression or classification task, Recurrent Neural Networks (RNNs) are specifically designed for time-structured data input. Moreover, the NN graph can be weighted in order to benefit from interesting properties, such as regularized training (Subsec. 3.4.2).
It is important to highlight that every Neural Network represents a computational graph: each neuron performs an operation whose result is propagated to a range of other neurons. It is worth to stress out this point: each neuron can receive the output of different neurons and send its own output to an extended or restricted number of child neurons.

In formulas, each layer l returns the following output:

$$\mathbf{y}_l = W_l \mathbf{x} + \mathbf{b}_l \tag{3.1}$$

where $\mathbf{y}_l$ is the layer output vector, $\mathbf{x}$ is the input vector and $W, \mathbf{b}_l$ are the layer's matrix and bias vector, respectively. Please note that the latter two elements represents the trainable weights of layer l, the ones that have to be fitted through the training process in order to minimize the net error.
From the neuron k point of view, this operation reads as:

$$y_k = \phi_l \left(\sum_{i=1}^{n} (w_i^k \cdot x_i) + b_k \right) \tag{3.2}$$

where ϕ_l is the layer l activation function, w_i^k are the elements of the kth row of matrix W_l, x_i are the elements of $\mathbf{x}$ and b_k is the bias element corresponding to node k within $\mathbf{b}_l$. An intuitive representation is depicted in Fig. 3.1, where the bias b is represented by w_0 and the activation function is a step function.

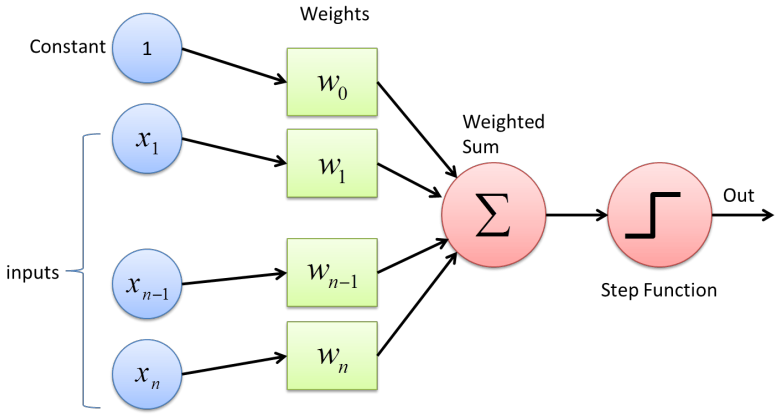

Figure 3.1: Single perceptron (or node) architecture.

3.2.1 Representation Power & Depth

The representation power of an automatic classifier describes its capability to represent (approximate) functions.
These functions are the real maps from input vectors $\mathbf{x}$ to output labels $\mathbf{y}$, namely $\mathbf{y} = f(\mathbf{x}; \theta)$. Vector θ represents the parameters of the mapping. The network's representation of f takes the form of the function $f^*(\mathbf{x}, \theta)$, that can be view as an approximation, an estimate of f. In fact, in all Machine Learning problems, f is unknown: only the data points and the labels are available, in a certain, limited amount.
As aforementioned, within the training process an optimization problem is solved in order to find the set of network parameters θ^* that minimizes the classification error. From the NNs point of view, the output layer must produce the desired output form, while the hidden layers have to *decide* how to use data points in order to learn the

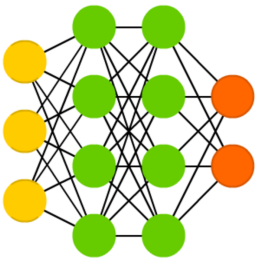

Figure 3.2: Feed Forward Network

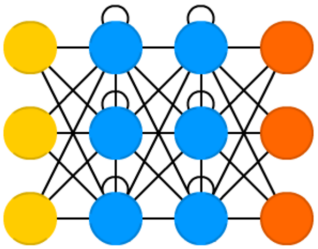

Figure 3.3: Recurrent Neural Network

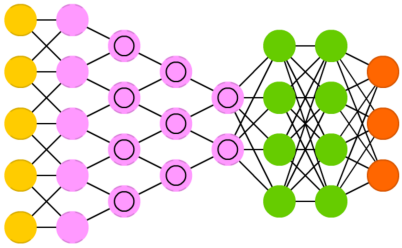

Figure 3.4: Convolutional Neural Network

Figure 3.5: The main types of Neural Networks. Yellow/Red dots: input/output neurons. Green dots: simple neurons. Blue dots: recurrent neurons. Violet dots: convolutional neurons. Please note the differences in both type of neurons and graph structure. Images adapted from `https://towardsdatascience.com`

representation function f^*.
In the following the particular Neural Networks' way to learn this representation is described.

Network Depth

The layers between the input and the output ones are called *hidden layers* and their number determines the *depth* of a neural network.
Since NNs are nothing more than composition of functions of affine transformation of the data, the depth of a network encodes an *a priori* belief about the function f that one has to estimate: it is composed by many other n functions f_i, with n equal to the number of the net layers. In formulas, this hypothesis reads as $f(\mathbf{x}) = f^{(2)}(f^{(1)}(\mathbf{x}))$ for a single hidden layer NN (represented in Fig. 3.7). This is an extremely powerful feature of deep neural nets, that expands considerably their generalization capacity. In other words, this point of view can also be interpreted as the belief that *"the learning problem consists of discovering a set of underlying factors of variation that can in turn be described in terms of other, simpler underlying factors of variation."* [15].
The main drawback is brought by the computational cost: the more f^* is complex, the more the optimization problem will be large in terms of number of parameters.

Below is provided the estimate f^* for a 1 hidden layer FFN (Fig. 3.6):

$$f^*(\mathbf{x}; W, \mathbf{c}, \mathbf{w}, \mathbf{b}) = \mathbf{w}^\intercal max\{0, W\mathbf{x} + \mathbf{c}\} + \mathbf{b} = \mathbf{w}^\intercal \mathbf{h} + \mathbf{b} \tag{3.3}$$

where $W, \mathbf{c}, \mathbf{w}, \mathbf{b}$ are matrix and bias terms, namely the net trainable parameters, the ones that will be optimized to find the minimum net error, while $\max\{0, \dots\}$ represents the hidden layer's activation function. Please note that the quantities in Eq. 3.3 are a mixture of affine (linear) transformations and a non-linear one, the latter represented by the activation function.

Non-linearity learning

Linear models are a comfortable framework, because they produce closed form or at least convex optimization problems.
However, what if the real classification function f is not linear w.r.t. to the input $\mathbf{x}$? Within the classical Machine Learning framework, a non-linear function $\phi(\mathbf{x})$ is applied to the data space populated by $\mathbf{x}$ before the training process, resulting in the learning of a non-linear map with the form of f. This is known as kernel trick and is widely (and effectively) used [16]. So, the problem shifts: how to choose ϕ?
There are two main options:

- Manually engineer ϕ, exploiting human practitioner knowledge of the research domain, being it a physical, biomedical, economical field or *guessing* ϕ, picking it from different families (polynomial, radial etc.). In most of the cases, the latter approach is extremely expensive in time of computational cost, making it often unfeasible. Actually, also the human practitioner exploiting can be

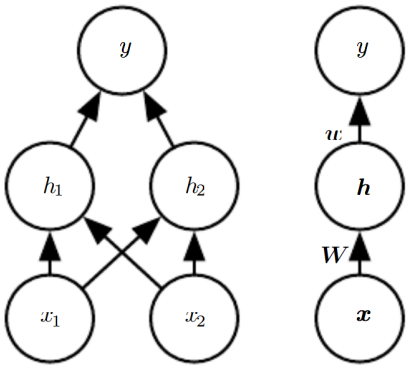

Figure 3.6: Example scheme for a 1 hidden layer FeedForward Net. Right: net computational graph: input vector $\mathbf{x}$ is provided to the input layer, the matrix W is applied to it to produce the hidden state vector $\mathbf{h}$. Finally, activation function is applied along with the scalar product with $\mathbf{w}$. Left: classical representation of a NN. Please note that the neurons are represented by the node of the graph, that "contain" their internal state. Picture taken from [3].

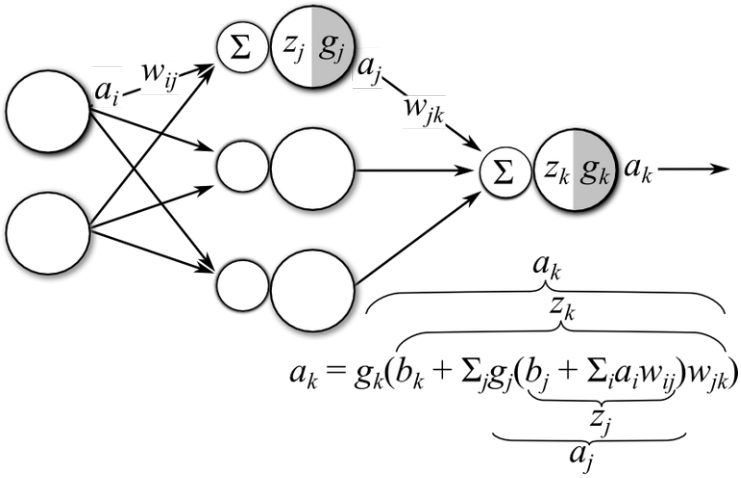

Figure 3.7: Neural Network as a composition of functions. g are the layers' activation functions, while w, b are the net weights that constitutes the affine transformation. Please note the nested computation of the a_k value. That is produced by the first and second layer, whose neurons output the a_i and a_j sets of values, respectively.

dangerous, since it introduce the practitioner personal knowledge bias within the learning process, resulting sometimes in ineffective results.

- The **Deep Learning** approach consists in **learning** ϕ. From a formal point of view, now the representation of f reads as $y = f^*(\mathbf{x}; \omega, \mathbf{w}) = \phi(\mathbf{x}; \omega)^\intercal \mathbf{w}$, where vectors $\omega, \mathbf{w}$ represent the set of neural nets trainable parameters θ, that will be the target of the training numerical optimization problem. Please note that Eq. 3.1 is coherent with this representation of f.
  Within this approach, f^* is ulteriorly parameterized in order to induce the learning of the non-linear map by the net itself. This increase tremendously the representation power of the learning algorithm: the capacity of representing the widest possible class of function. However, this does not come for free, since this approach gives up on the convexity of the problem, but the benefits outweight the harms.
  It is important to note that human practitioners can still leverage their knowledge by narrowing the number of families ϕ can be picked from, that is still a much *narrower* problem w.r.t. manually engineer it or guess it.

There are two important points to stress out. First, what described above is valid for all the Neural Nets types, so also for the ones used in this work.
Second, the convexity of the data space will not be subject to an extensive investigation. Since it is a numerical optimization problem and scope of this work is to model a social behaviour system (the Private Equity market), it is not possible to infer *a priori* any data space structure. If, as an example, the target framework were a physical one, than there would probably be some law or equation to drive a convexity analysis of the data point space, but this is not the case. In order to overcome this issue, optimization algorithms that are proved to perform effectively on non-convex sets will be used and extensively described within Sec. 3.4.

Universal Approximation Theorem

The representation power of NNs has been extendedly described previously, but it seems valuable to provide a more formal evidence of such capability: the Universal Approximation Theorem [17].
This result states that, under mild assumptions about the activation function, any FeedForward Network with a finite number of neurons (width) and one hidden layer is able to approximate continuous functions on compact subsets of $\mathbb{R}^n$. In other worlds, it assures that it is theoretically possible for a neural network with a finite number of parameters to find an estimate of any function f. Formally:

Theorem 1 (Universal Approximation Theorem)**.** *[17] Let $\psi(\cdot) : \mathbb{R} \to \mathbb{R}$ be a non-constant, bounded and continuous function which in this context is the activation function. Let $X \subseteq \mathbb{R}^m$ and X compact. Lets denote with $C(X)$ the space of continuous functions on X. Then $\forall \epsilon > 0$ and $\forall$ function $f \in C(X)$, $\exists\ n \in \mathbb{N}$, $\exists\ v_i, b_i \in \mathbb{R}$ two real constants and $\exists\ w_i \in \mathbb{R}^m$ real vectors, with $i = 1, \ldots, n$, such that:*

$$F_n(\boldsymbol{x}) = \sum_{i=1}^{n} v_i \cdot \psi\left(\boldsymbol{w}_i^T \boldsymbol{x} + b_i\right)$$

where n indicates the number of hidden neurons, it is an approximate realization of the function f, that is:

$$||f - F_n|| < \epsilon \tag{3.4}$$

This result has two main drawbacks. First, the *learnability* of the NN parameters is not assured. Second, the number of neurons needed to approximate f is bounded, but there is no assurance on the reachability of such number of neurons by a *real world* computation machine. Roughly speaking, the theorem states that the network *size* is finite in term of number of parameters, but this number can be so high to prevent any algorithm to find an optimal solution within the training phase.

On the other hand, Lu et al. [18] proved a universal approximation theorem version for deep neural networks: the width is kept bounded, but the network depth is allowed to increase. According to their work, any n-dimensional input Lebesgue integrable function can be represented by NNs with width equal to $n + 4$, arbitrally large number of layers and Rectified Linear Unit activation function (Subsec. 3.3.1). Again, formally:

Theorem 2 (Universal Approximation Theorem for Width-Bounded ReLU Networks [18])**.** *For any Lebesgue-integrable function $f : \mathbb{R}^n \to \mathbb{R}$ and any $\epsilon > 0$, there exists a fully-connected ReLU network $\mathcal{A}$ with width $d_m \leq n + 4$, such that the function $F_{\mathcal{A}}$ represented by this network satisfies*

$$\int_{\mathbb{R}^{\times}} |f(x) - F_{\mathcal{A}}| dx < \epsilon \tag{3.5}$$

where n is the number of input variables.

It is easy to see that Th. 2 can be seen as the dual version of Th. 1: in the first case the width measure is allowed to grow (in a limited amount) keeping the depth fixed, in the second one the width is kept fixed, but there is no bound on the depth. From the technical point of view, while the first theorem considers just continuous functions, Th. 2 deals with a broader class, the Lebesgue-integrable functions on the whole Euclidean space. This justifies the choice of using an L^1 distance in Th. 2 instead of the Th. 1 L^∞ one.

It is worth to note that neither the first nor the second theorem provides a *practical* boundedness indication of the the NN size needed to approximate a general target function. Even if some results about width efficiency is presented in the work of Lu e al. [18], the size of the network is not assured to be computationally reachable for a general f.

3.3 Units & Architecture types

The neurons (units) of a Neural Net can be designed in various ways, depending on the neuron layer type: input, hidden, output. Actually, as it will described below,

the difference between input and hidden neurons is purely conceptual, as the input layer can be seen as the first hidden one. As for the connections among them, the net architecture, the shape of the representing graph has a central role in determining the NN type and so the network aim (imaging classification, time series modeling etc.).

3.3.1 Hidden units

First of all, it is fundamental to state that the design and testing of NNs' hidden units is an active area of research, but with a very restricted number of theoretical basis or guiding principles [19]. This results in a design process that consists in a trial and error approach, often supported by the state of the art knowledge about a specific domain, such as time series modeling or image recognition.
Within the simple FeedForward type of network, the hidden units are often identified as the activation function used in their layer, since their internal operation is always the one in Eq. 3.3. Within other, more complex types of networks, the neurons may perform a wider range of operations, but their description is out of scope, except for the actually used one in this work: the LSTM (Sec. 3.5).

Before enumerating and describing the activation types, two important technicalities have to be remarked. First, the activation function choice must take into account also the optimization task that the net produces. In fact, some classes of activation functions may cause *gradient vanishing* issues, as it will described more deeply in Sec. 3.4.
Second, a broad range of activations are not differentiable in some points. However, since the training process is not expected to actually reach a 0-gradient point of the target function, having this function minima in undefined gradient points does not constitute an issue.

Finally, some of the most used activation functions are briefly introduced to the reader while in Table 3.9 their plots and equations are reported.

1. **Logistic Sigmoid and Hyperbolic Tangent.** Namely $g(z) = \sigma(z)$ and $g(z) = \tanh(z)$, respectively. When used as the base activation function, the one used to compute the unit output, they have a serious drawback: their $g(z)$ *saturates* when z is high, i.e. the output is almost the same for very distant input points. Moreover, within these regions, the first order derivative is almost null, that is why this issue prevents the gradient-based optimization algorithm from an effective learning result.
   The reader can now easily deduce that this type of activations are not widely used. An exception is represented by some type of networks, whose units actually use $\sigma(z)$ in their internal computation, as it will described for the LSTM in the dedicated section.
2. **Rectified Linear Unit (ReLU).** It uses the activation function $g(z) = \max\{0, z\}$, that has many advantages. The optimization process is *easy*, since this unit is very similar to the linear one: the first order derivative remains

not lower than 1 across half its domain, the second derivative is null almost everywhere and when the unit is active (i.e. when its output is non-null), its first order derivative is equal to 1. Since the optimization methods are gradient-based, these features assure that the gradient direction is much more *informative* on the minimum direction w.r.t. second-order effects activation functions [20].
The main drawback of ReLU is that it can cause the so called *dead neurons* phenomena. In fact, if the input value is below the threshold, the output is null. This, within the NNs computation graph can produce a cascade effect, in which a larger and larger number of neurons connected to the first dead one can move their own output to 0 through the learning process, resulting in a poor learning final output. However, the generalization of ReLU (leaky ReLU [21], PReLU [22]) and the development of SeLU have overcome the issue.

3. **SeLU.** Scaled Exponential Linear Unit [23] is part of a very recent work on Self Normalizing Networks (2017). It has the form (Fig. 3.8):

$$g(\alpha, x) = \lambda \begin{cases} x, & \text{if} \quad x > 0 \\ \alpha(e^x - 1) & \text{if} \quad x \leq 0 \end{cases}; \quad \lambda > 1 \tag{3.6}$$

where λ assures the slope of the line to be greater than 1 and α is the scale parameter.
This class of NNs are designed to perform a mean-variance scaling of the training points in each layer, resulting in a considerable increase of the training performance and the outperforming of ReLU-based networks.
More formally, let Ω be the training points' space, $\omega = \sum_{i=1}^{n} w_i$ and $\tau = \sum_{i=1}^{n} w_i^2$ be the first and second order summation of that layer weights, respectively. Then, according to the definition in [23], a NN is Self normalizing if each layer is represented by a mapping $l : \Omega \to \Omega$ that maps layers' weights mean and variance from that layer to the next one having a fixed point in (ω, τ). The original source provides also indication of the weight initialization and regularization technique required to make the net really self normalizing, however this is out of scope for this work.
In addition to the proved better results of SeLU-based networks w.r.t. to the ReLU-based ones, SeLU still benefits from the non-vanishing gradient feature and it prevents the network to show the deathunits phenomena, having non-null output for negative inputs as the aforementioned ReLU generalization. Because all of this, SeLU has been chosen as the default activation function within this work NNs.

4. **Linear.** The linear activation function (or identity function) can be seen as non using an activation function at all, letting the layer out be $\mathbf{h} = W\mathbf{x} + \mathbf{b}$. It can be proved linear unit layers results in parameters saving features w.r.t. the actual activation unit ones [24].

5. **Others: softplus, softsign, hard tanh.** These type of functions have empirically proved to be difficult to optimize, so they are hardly used in the present times [24].

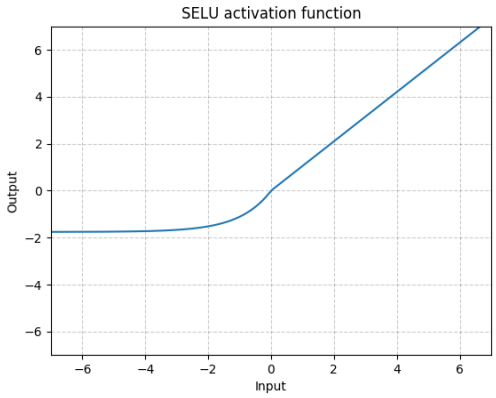

Figure 3.8: SeLU activation function plot. $\alpha = 1.6732, \lambda = 1.0507$

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figure 3.9: Activation functions' equations and plots.

3.3.2 Output units

The output unit selection is deeply correlated with the net output type, thus the net function that in turn depends on the target function whose the training net and the linked optimization problem is based on, namely the *cost function* (Subsec. 3.4.1). Anyway, this is not sufficient, since the output unit should not hamper the optimization task that is required by the net training, so it has to prevent the aforementioned gradient vanishing problem.
In the following subsection, the vector presented to the output layer, that has been computed by all the hidden layers of the net, is represented by **h**.

Multinoulli Output Distributions & Softmax Output units

If a classifier net is designed, the output that it must produce is a multinoulli probability distribution over n classes. In this case, the generally preferred output unit is the *softmax unit*, a generalization of the sigmoid one [25].
The requirement for the function that this unit represents consists in producing a vector $\hat{\mathbf{y}}$, with length n, whose each element $\hat{y}_i$ represents the probability of the data point i to belong to the class y given its covariates values, namely $\mathbb{P}(y = i|\mathbf{x})$. This is achieved by two subsequent operations: a linear transformation on the vector $\mathbf{h}$ and the application of the element-wise softmax function on the transformation itself. In formulas:

$$\begin{aligned} \mathbf{z} &= W^\intercal \mathbf{h} + \mathbf{b}, \\ \text{softmax}(\mathbf{z})_i &= \frac{\exp(z_i)}{\sum_{j=1}^{p} \exp(z_j)} \end{aligned} \tag{3.7}$$

where $W, \mathbf{b}$ are the layer's weights matrix and bias vector, respectively. Please note that the quantity $\text{softmax}(\mathbf{z})_i$ coincides with $\hat{y}_i = \mathbb{P}(y = i|\mathbf{x})$.
From the training mechanics point of view, the exponentiation in Eq.3.7 prevents gradient vanishing when the cost function is the log-likelihood. In this case, the training goal is to maximize $\log \mathbb{P}(y = i; \mathbf{x}) = \log \text{softmax}(\mathbf{z})_i$. So the optimization target function becomes:

$$\log \text{softmax}(\mathbf{z})_i = z_i - \sum_{j=1}^{p} \exp(z_j) \tag{3.8}$$

It is easy to see that the gradient of the Eq.3.8 function is strong everywhere and so it cannot *saturate*. Then, it should be clear the choice of the $\mathbf{z}$ exponentiation.
From the architectural point of view, within the output layer each unit provides the softmax function output of one class, e.g. if $n = 4$ the output layer will be composed of 4 softmax units, each one providing the corresponding probability y_i for each data point i.

Lastly, it is worth to notice that the softmax function creates a form of competition among its units: since their outputs sum up to 1, so if one unit increases its output value, than the others' outputs are necessarily "pushed" down to 0, producing the multinoulli-type output expected from this kind of layer.

In a theoretical approach, every unit used as hidden one can be used for the net's output. If the learning task is a classification one, then the number of neurons in the output layer should be equal to the number of classes.

3.3.3 Architectures

Within the Neural Nets jargon, *architecture* is referred as the design of the number of net's units and how they are connected among them.
The impact of the number of neurons, both in width and depth, has been extensively

discussed within Sec. 3.2.1, while here it will be described the variety of connections architecture (underlying graphs) that constitutes the main types of NNs. As aforementioned, the structure of the connections among neurons, across layers or in the same one, is directly related to the network aim, both in terms of classification target and modeling purposes. Although in recent and past years a great number of "exotic" architectures has been invented (Generative Adversarial Newtorks, Goodfellow et Al. [26], Deep Boltzmann machines, Salakhutdinov et Al. [27], Bidirectional Recurrent Networks, [28]), below the main type of nets are:

- **Multilayer Perceptron (MLP).** The fully connected architecture, used both for regression and classification. Each neuron accepts the output of all the previous layer's ones and transmits its output to all the neurons of the next one. This type of net is parsimonious in terms of number of parameters (and so of training time) and it is very versatile. However, results may be limited as for predictive performance because of the simple architecture: MLP accepts whichever data, so it is not crafted to model a specific process (time-based, images etc.).
- **Recurrent Neural Network (RNN).** Made to model sequential data, RNNs have a special pipe-line neurons' connection in every layer. The input process can be written as $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(\tau)}$, but RNNs are able to handle inputs of variable length. This input can be presented in many ways, but one of the most common consists in injecting in each input neuron a time step of the information, $\mathbf{x_t}$, that can be a numerical vector, a text, an image etc. The output can consist in a sequence of labels or just one value, corresponding to the last neuron of the last layer. This output is computed for each neuron or for just the last one in order to compute the loss function, along with the output response: $y^{(1)}, y^{(2)}, \ldots, y^{(\tau)}$ [29].
  From the formal point of view, the internal states $\mathbf{h}$ computed by each neuron are functions of the previous neurons' internal states *and* the input:

$$\mathbf{h^{(t)}} = f(\mathbf{h^{(t-1)}}, \mathbf{x^{(t-1)}}; \theta) \tag{3.9}$$

  where θ is the vector of neuron's parameters. From Eq. 3.9, it can be inferred that each neuron is trained to use a kind of summary of the past sequences of inputs up to its time step, t. As it will much clearer in the LSTM section, the net can be designed in order to selectively keep (or not) the parts of this summary. This causes an interesting feature of RNNs: parameters sharing. Each neuron's output is a function of the previous neurons' output and it is computed using the same update rule as the previous neurons' outputs.
  Finally, an important type of RNN, the one used within this work:

  - **Long Short Time Memory (LSTM).** [30] The LSTM is structured exactly as a common RNN, but its neurons are themselves made by other units. This design allows the network to keep the information of the internal states and input series, as well as to *forget* some part of it, resulting in a more complex learning process. More details will be provided in Sec. 3.5.

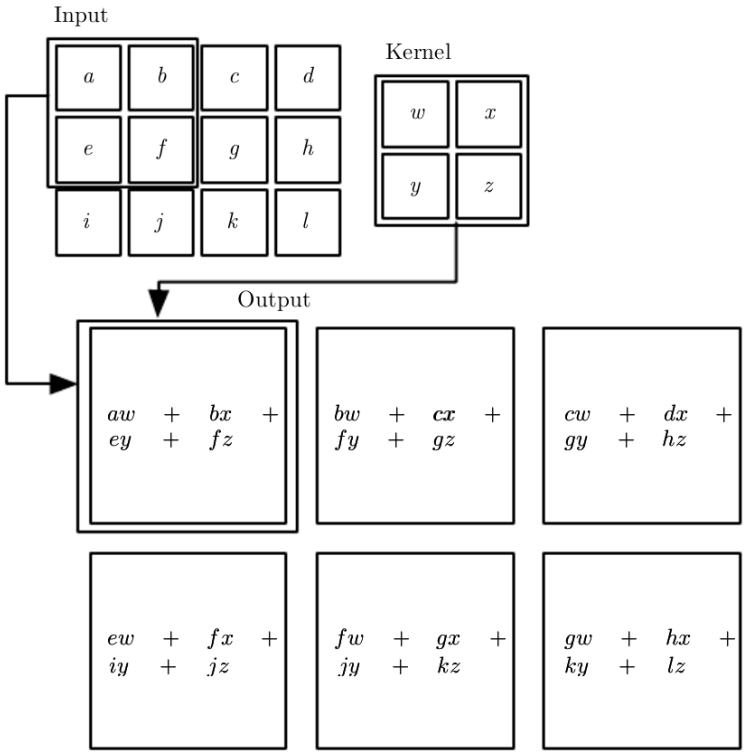

Figure 3.10: Example of a 2D convolution without kernel flipping. Shown just the output for positions where the kernel lies entirely within the image.

- **Convolutional Neural Network (CNN).** Deeply used for object recognition, are particular feedforward network. They are born to handle data with a grid-like topology, such as time series which can be interpreted as 1D grid of value or images as 2D of pixels. The main characteristic of a CNN is that in at least one layer a convolution is performed instead of the simple matrix multiplication [31]. An example of convolution is reported in Fig.3.10.

3.4 Learning & Optimization

Each time a Neural Network is trained, a complex optimization problem is cast to an advanced numerical optimization algorithm. Actually, every Machine Learning problem boils down to an optimization task, but NNs' training makes this task extremely hard, mainly because of the non-linearity introduced by its activation functions.
The goal of numerical optimization within NNs' training is to minimize $J(\theta)$, the expectation value of a cost function L, that, naively speaking, equals to the net *error*, whether it is a classification or regression one. Please recall that vector θ contains all the network trainable parameters.

It is worth to note that optimization in Machine Learning is kind of different from the classical one. In fact, the *generalization error* introduced by the cost function

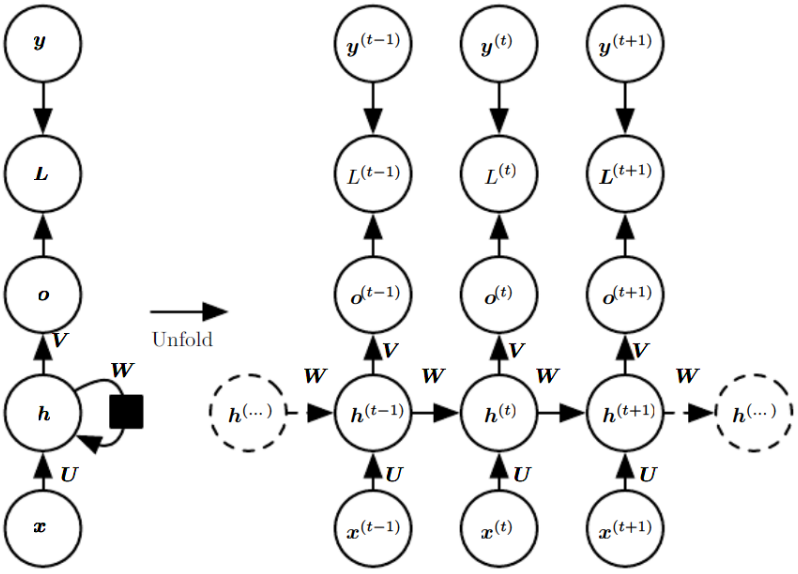

Figure 3.11: *(Left)* Condensation diagram of a recurrent network. *(Right)* Unfolded computational graph of the Condensation diagram. Information from the input **x** are incorporated into the state **h** that is passed forward through time. Each node is associated with one particular time instance. Each value of the input sequence generates a value of the output sequence **o**. Then, a loss function **L** evaluates the output **o** with the corresponding training value **y**. $\mathbf{U}, \mathbf{W}, \mathbf{V}$ are three weight matrices which parametrized the connections between the input and the hidden, the hidden to another hidden and the hidden to the output respectively. These matrices are updated at the end of each epoch in order to minimize the total loss value.

can be written as:

$$J(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{data}} L(f^*(\mathbf{x};\theta), y) \tag{3.10}$$

where L is cost (*loss* function), $f^*(\mathbf{x};\theta)$ is the predicted output depending on the covariates $\mathbf{x}$ and the net's parameters θ. The expectation is computed across the so called *empirical distribution* $\hat{p}_{\text{data}}$, the one built on the training data. If the *true data distribution* p_{data} were used instead, the classification or regression problem would be an *optimization task*.
Obviously, the latter is not available: Machine Learning is used to infer the data generating process. Instead, the training samples are available to build the empirical distribution $\hat{p}(\mathbf{x}, y)$ that in turn can be exploited to compute the *empirical risk*:

$$J(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim\hat{p}(\mathbf{x},y)} L(f^*(\mathbf{x};\theta), y) = \frac{1}{m}\sum_{i=1}^{m} L(f^*(\mathbf{x}^{(i)};\theta), y^{(i)}) \tag{3.11}$$

where $J(\theta)$ becomes the optimization problem objective function whose θ is the target vector. So rather than optimizing the *real risk* (i.e., the *real* cost function), it is optimized its empirical estimate based on the training set (Eq.3.11). If the training points are strongly correlated to the real data generating distribution, then by optimizing $J(\theta)$ it is found an optimal solution θ^{opt} that well approximates the *real* one.
Within this setting, the Machine Learning problem is driven back to a regular optimization task. The objective function described above do not include a regularization term, but the procedure that will be depicted in the following is easily extendable for the regularized case.

Within the numerical optimization framework considered in this work, the minimization of the objective function is performed through *gradient descent* algorithms. In particular, the optimizers described in the following exploit the so called *batch* or *mini-batch* optimization, that consists in computing the gradient at each step using just a random sample (the mini-batch) of the objective function points (i.e. the cost function evaluated on training set points). The class of gradient descent algorithms used is the *Stochastic Gradient Descent* (SGD) methods. Since evaluating the gradient of ML-related cost function (especially the Neural Nets ones) may be a computationally expensive task, computing this quantity on a subset of training set points can greatly speed up the optimization process. SGD methods exploit this mechanism in order to find a reliable estimate of the cost function minimum. Their procedure can be summarized as in Algo. 2.

$L(\cdot)$ is the cost function to minimize, η is the so called *learning rate* and the other quantities have the same notation as the previous sections. In the following, the role of such quantities will be clarified.

Before diving into the optimization algorithm's mechanism, it is worth to clarify few concepts and quantities regarding numerical optimization applied to Machine Learning.

- **Learning rate ϵ.** Scalar that determines the step size of each iteration towards the target function minimum. Roughly speaking, it determines how

Algorithm 2 SGD generic optimizer

Select a random initial weights vector θ
Define learning rate ϵ, mini-batch size m
while stopping criteria **do**
  Sample a mini-batch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$
  Apply update: $\mathbf{g} \leftarrow \eta \, \nabla_{\boldsymbol{\theta}} \sum_i L(f^*(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
end while

often the model weights are updated. It has a small positive value often in the range [0,1]. An high learning rate (e.g. 0.8) would shorten the training time and lower the accuracy, while a low learning rate (e.g. 0.001) would take much more time, but with a potential greater accuracy.

- **Mini-batch.** Subset of the training set used to update the model's weights within each gradient descent operation.
- **Epoch.** Number of iterations that the algorithm performs on the training set. Each time Algo. 2 reaches its end, another epoch starts, until the number of epochs specified at the beginning of the training procedure is performed.

3.4.1 Cost function

The cost function L plays a fundamental role in the training phase since it defines the target of the optimization algorithm. Roughly speaking, it is the *average* of the errors on the whole training set. The type of such average used is tightly coupled with the NN's task: if the net is of regression type, then the *Mean Square Error* (MSE) will be used, whether if it is a classifier, the Log-likelihood (LogLik) cost function will be applied. While the fist must be minimized during the training, LogLik have to be maximized.

- **Mean square error (MSE).** This cost function writes as: $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$, where Y_i is the actual response value of the i-th observation, while $\hat{Y}_i$ is the NN's response.
- **Log-likelihood.** In this case the cost function is actually the negative log-likelihood, that is equivalent to the *Cross-Entropy* (CE) between the training data and the model distribution [32]. CE cost function reads as $J(\theta) = -\mathbb{E}_{(\mathbf{x},y)\sim\hat{p}_{\text{model}}} \log p_{\text{model}}(\mathbf{y}|\mathbf{x})$, where p_{model} represents the probability distribution output from the Neural Network. For each data point i, given the covariates' values of i, p_{model} equals to the probability vector of length n of belonging to each of the n classes.

From a theoretical point of view any function can be chosen as a cost function, but from the practical side, since it is the optimization objective function, convex functions are preferred.

3.4.2 Neural Networks regularization

Regularization for Neural Network relies on classical machine learning regularization methods and NNs specific ones.

- **Norm Penalties:** Add a penalty term to the loss function J (the objective function that the NNs attempt to minimize through numerical optimization):

$$\hat{J}(w; X, y) = J(w; X, y) + \lambda\Omega(w) \tag{3.12}$$

  where w represents the NN weights, X the input matrix, y the output vector and λ the *penalty factor*. The bigger the lambda, the more the penalization term $\Omega(w)$ will *shrink* the norm of the parameters w.

  - L^2 **parameters regularization:** Applied if $\Omega(w) = \frac{1}{2}w'w$. It can be proven that this regularization is equal to an *early stopping* [33].

- **Dropout:** Ensemble method that derives from *bagging*, but it is specific for NNs. Dropout removes non-output units from the full net by multiplying their output by 0. Equals to train an ensemble of different networks that share the same "root" net. Training all the models would be unfeasible. Fortunately, from a mathematical point of view, by multiplying the output of each layer neurons by a fixed probability $p_{drop} \in [0, 1]$, one achieves the same result as bagging different NNs. The quantity p_{drop} is called *dropout rate*.
  The theoretical rationale behind this is called the **weight scaling inference rule**. *"There is not yet any theoretical argument for the accuracy of this approximate inference rule in deep nonlinear networks, but empirically it performs very well"* [34]. In modern nonlinear NNs this approximation has been empirically proved to "simulate" a nets bagging [35].
  The benefits of dropout are basically the same of bagging: it reduces training bias of the algorithm in order to prevent overfitting.

3.4.3 Neural Network Optimization issues

Training neural networks naturally involves the general, non-convex optimization case [36]. Anyway, even convex optimization may become not so comfortable. Below the most frequent (and most severe) optimization issues are briefly listed since they do not affect the case under study.

1. **Ill-conditioning.** Even in the convex case, a bad issue can occur: Hessian Matrix ill-conditioning. It can be derived that each optimization gradient's descent step is performed by computing the quantity $\frac{1}{2}\epsilon^2\mathbf{g}^\intercal H\mathbf{g} - \epsilon\mathbf{g}^\intercal\mathbf{g} = \epsilon^2\alpha - \epsilon\beta$, that is added to the cost function. The hyperparameter ϵ is called *learning rate*. Obviously such quantity must be negative in order to achieve a gradient *descent* iteration. So, if the hessian matrix is ill-conditioned (i.e. if its *condition number* is high), α may be similar in value to β, slowing dramatically the optimization. A possible solution is to adapt the learning rate throughout the optimization process, in order to contrast this phenomena. Moreover, the

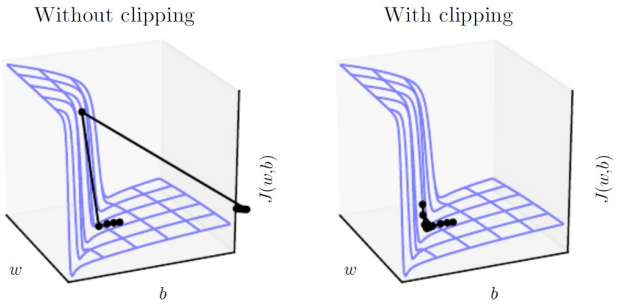

Figure 3.12: Effect of cliff on optimization, with and without gradient clipping. Within the neural network parameters space, the optimizer moves towards a cliff can catapult the actual optimization point very far from the previous trajectory (Left). Gradient clipping efficiently overcomes the issue by reducing the learning rate if the gradient norm overcomes a certain threshold (Left). Image taken from [3].

gradient norm can be monitored during the training, in order to check if it lowers its value or not.

2. **Local Minima.** Since Neural Nets training tipically involves non convex objective functions, they can present a number of local minima that can pose serious problems to optimization. However, deep learning practitioners believe that for sufficiently large NNs, most local minima have a low cost value ([37], [38]), so if the optimization algorithm get "stuck" in one of this point this should not hurt the training process.

3. **Flat regions.** Another significant threat to optimization process is the possible presence of flat regions of objective function, such as *plateaus* or *saddle points*. It is possible to prove that such regions, particularly saddle points, appear frequently in NNs training ([37], [39]). However, the empirical visualization provided by ([40]), shows that modern state of the art optimization algorithms, such as the *Adam* algorithm, are able to *escape* from saddle points within a sustainable number of iterations. More evidence can be found in [1].

4. **Cliffs and Exploding Gradients.** Deep NNs cost functions often result in the multiplication of several large weights together, that cause the presence of extremely steep regions. The steepest ones are called intuitively *cliffs*. When the optimizer encounters the edge of a cliff, a gradient descent update can move the parameters space point very far from the previous trajectory (Fig. 3.12). This can result in the loss of the most "optimization work" done so far. However, such dynamic can be avoided through the *gradient clipping* heuristic, that intervenes to reduce the learning rate when the gradient norm overcomes a certain threshold.

[1] http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

3.4.4 Optimization methods

Since the optimization within Neural Networks training is a non-trivial task, in which many bad topologies of the cost function may induce serious optimization issues, the choice of the optimizer is crucial.

The one used in this work is the *Adaptive Moment Estimation* (Adam), an update of the *RMSProp* algorithm. Adam performs an adaptive learning rate optimization, updating its ϵ according to the gradient's *momentum*. The momentum, named after a physics-related analogy, measures how large and how aligned are the gradients sequence in a determined optimization steps window. If it is used, it allows the optimization point to "move" across the objective function surface as if it were "heavier" than the non momentum-equipped point. These particular properties are designed in order to reduce the optimization process sensibility w.r.t. the topological issues that may occur, such as cliffs or saddle points.

The extensive procedure of ADAM optimizer is described in Algo.3.

Algorithm 3 ADAM optimizer [41]

1. Define step size ϵ (e.g. 0.001)
2. Define exponential decay rates for momentum estimates, ρ_1 and ρ_2 in [0,1)
3. Define small constant δ used for numerical stabilization (e.g. 10^{-8})
Set initial parameters $\boldsymbol{\theta}$
Initialize 1st and 2nd moment variables $\boldsymbol{s} = \mathbf{0}, \boldsymbol{r} = \mathbf{0}$
Initialize time step $t = 0$
while stopping criteria **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$
  Compute gradient: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
  t $\leftarrow$ t + 1
  Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \, \boldsymbol{s} + (1 - \rho_1)\mathbf{g}$
  Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \, \boldsymbol{r} + (1 - \rho_2)\mathbf{g} \bigodot \mathbf{g}$
  Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$
  Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$
  Compute update: $\Delta\boldsymbol{\theta}$ = -$\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}}+\delta}$ (operation applied element-wise)
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta \, \boldsymbol{\theta}$
end while

3.5 Long Short-Term Memory Networks

In 2017, approximately 4.5 billions of automatic translations have been performed by Facebook every day. The underlying algorithm was constituted by long short-term memory (LSTM) networks [2]. Actually, LSTMs are now pervasive within industrial

[2]https://www.theverge.com/2017/8/4/16093872/facebook-ai-translations-artificial-intelligence

applications, with a constant flow of enhanced implementations.

LSTM adds to the basic structure of recursive neural networks (Fig. 3.11) *self-loops* that allow the cost functions gradient (i.e., the information), to flow through the net for long durations [30]. Within the modern version, these self-loops are gated in order to control the amount of information flow and so the time scale of integration of the input sequence whose pattern must be learnt. Actually, the beauty of LSTM relies also on the fact that the control parameters of such gates are learnt during the training process, making the LSTM a largely adaptive model.
It is worth to note that deep LSTM architectures have been explored [42].

3.5.1 LSTM Architecture

The architecture of LSTM is exactly the same as the regular RNN one, but with each neuron constituted by a *cell* depicted in Fig. 3.13. This kind of cell is designed in order to build an internal recurrence (i.e., the self-loop), it has the same output and input of a regular NN neuron, but presents an internal system of gating units that controls the information flow, managed by cell-specific parameters that are learnt during the net training. Each gate controls the flow by means of a scalar between 0 and 1 that is multiplied to the internal state of the cell i, $s_i^{(t)}$ or to its internal hidden state $h_i^{(t)}$.
The gates can be summarized according to their functions:

- **Forget gate.** This unit produces the f_i^t according to:

$$\sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right) \tag{3.13}$$

  where $\sigma(\cdot)$ is the sigmoid function, $\mathbf{x^{(t)}}, \mathbf{h^{(t)}}$ are the input and hidden vector respectively. $\mathbf{b^f}, U^f, W^f$ are the bias, input and recurrent forget weights of the layer whose the cell i belongs to. Please note that within the RNN architecture, the vector $\mathbf{h^{(t)}}$ contains all the outputs of the previous cells.

- **External input gate.** Computed as the forget gate, but with its own trainable weights:

$$\sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right) \tag{3.14}$$

  it is noted as $g^{(t)}$.

- **Output gate.** Again, equipped with its own trainable weights:

$$\sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right) \tag{3.15}$$

  it is noted as $o^{(t)}$.

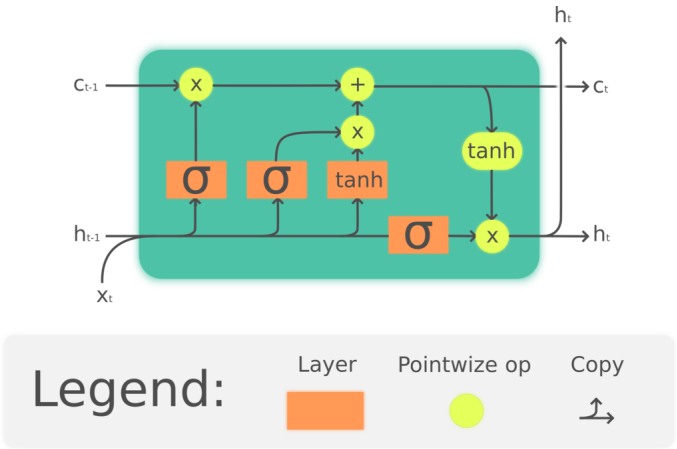

Figure 3.13: Structure of a LSTM cell. Orange spots represent the gates of the cell, while the yellow ones the operations needed to compute outputs and hidden states.

Finally, the cell output can be computed as:

$$s^{(t)} = f^{(t)} s^{(t-1)} + g^{(t)} \sigma \left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)} \right) \tag{3.16}$$

where $\mathbf{b}, U, W$ are the layer weights. Moreover, the hidden state is gated through the aforementioned output gate as:

$$h_i^{(t)} = \tanh s_i^{(t)} q_i^{(t)} \tag{3.17}$$

It is easy to note that when the forget gate is near 0 at time step t, the LSTM status in t will not be influenced by the one of the previous cell (i.e. the previous time step $t-1$). On the other hand, if $g_i^{(t)}$ is almost null, the internal state dependence on the input value $x^{(t)}$ will be significantly lowered. Same holds for the output gate. The rationale behind these non-trivial computations relies on the so called *forget-recall* dynamic. Each cell can learn how much information to accumulate (through $g^{(t)}$) or to forget (through $f^{(t)}$) in order to minimize the error. Roughly speaking, at each time step LSTM is designed to learn how much to forget or to recall from the pattern of the previous time step. This feature, along with the enhancements carefully crafted by the deep learning practitioners, has allowed the LSTM to become one of the most powerful neural net in a broad range of fields.

LSTM within this work

As described in the previous chapters, the input sequences x_i of each company is composed by the information about the 3 investment rounds of the company itself. This information, is composed by the minimum, the maximum and the mean investor ranking of the investors that participated in that ith company round j, along with the number of investors. E.g.: $x_{i_1} = Min_1, Max_1, Mean_1, Num_1$, $x_{i_2} = Min_2, Max_2, Mean_2, Num_2$, $x_{i_3} = Min_3, Max_3, Mean_3, Num_3$.

From the theoretical point of view, the LSTM architecture implies that the events represented by the time series $(x_{i_1}, x_{i_2} \dots)$ are consequential, that is they are bounded together in time from the less to the most recent. So, the LSTM is able to model the process as much as there is actually a link between the ranking of the investor of the past rounds and the ranking of the investors of the recent ones. I.e.: the underlying assumption is that the quality of the investors that invest in a company in a certain round do affect the quality of the investors in that company in the next rounds.
From the Private Equity practitioner point of view, the latter is a robust assumption, so the usage of the LSTM Networks is fairly corroborated.

3.6 Experiments

3.6.1 Introduction

To estimate the companies future statuses probabilities a dataset based on TR Eikon platform has been used. As largely described in Chapt. 2, it is composed by the investors information on investments rounds that occurred between 1998 and 2018. After the data cleaning process, the final dataset included investment info of 47907 companies, distributed in 9 business sectors.
It is worth to remark the data set exit's distribution in Tab. 3.1 since the classes are extremely unbalanced. This will be a serious issue within the algorithms' training phase.

	Private	IPO	Bankrupt	Acquisition
Companies	30193	15927	1413	374

Table 3.1: Number of companies within the four distinct classes in the input dataset.

The Random Forests have been trained on `R`, exploiting the `randomForest` package. As for the neural networks, the `Python` `keras` library has been used, along with the TensorFlow computational engine.

Training and test sets

The procedure adopted in all the experiments involves a training and a test phase. During training phase the algorithm attempts to *learn* the data pattern, while in the test phase the related set points are used to validate the model. The latter operation is performed comparing the model's output labels to the real ones in order to evaluate its predictive performance. The splitting ratio between train and test set can vary according to the experimental design.

Binary vs. Multi-class classification

Within the problem under study, there are 4 classification labels: Bankrupt, IPO, Privat, Acquisition. It would be straightforward to build a 4-labels classifier, both for the RF and NN algorithm. However, this approach can lead to poor results,

especially when dealing with a remarkable classing unbalancing, as in this case.
In order to overcome this issue, 4 binary classifiers have been built. Precisely one binary classification for each outcome, in a "one vs rest" (OVR) approach, so that the model has to model the realization of an event or its complementary. So, the four classifiers resulted to be: "Private" or "Not-Private", "IPO" or "Not-IPO", "Bankrupt" or "Not-Bankrupt", "Acquisition" or "Not-Acquisition". Given an input observation with vector of features x and label y, each binary classifier returns: $f_i(x) = \mathbb{P}\{y = i|x\}$ as well as the complementary probability, for $i = 1, ..., n$ which is the event that the company will not end up in the i-th class. In practial contexts, such as the one this work operates is, in order to produce a probability distribution over n classes, the OVR probabilities have been normalized so that they sum to one:

$$p_i(x) = \frac{f_i(x)}{\sum_{j=1}^{n} f_j(x)}$$

Please note that this procedure is necessary since the n OVR classifiers are trained separately, so in general the $f_i(x)$, with $i = 1, .., n$ does not sum to 1. Thanks to this transformation, the end user of the predictive algorithm, would be able to visualize 4 distinct probabilities of the 4 distinct bankruptcy, acquisition, private, IPO events.

3.6.1.1 Predictive performance indicators

The most common performance indicator used in machine learning to evaluate algorithms is the accuracy. However this can be a misleading indicator in some applications.
As an example, consider the classification of pixels in mammogram images as possibly cancerous [43]. A typical mammography dataset might contain 98% normal pixels and 2% abnormal pixels. A simple default strategy of guessing the majority class would give a predictive accuracy of 98%. However, the nature of the application requires a fairly high rate of correct detection in the minority class and allows for a small error rate in the majority class in order to achieve this.

Because of such issue and given the large amount of classes unbalancing in the reference dataset (Tab.3.1), the algorithms predictive performance has been measured measured with 5 standard indicators: Positive/Negative *Recall* and Positive/Negative *Precision* (Eq.s 3.18, 3.20, 3.22, 3.19, 3.21). In a nutshell, the Precision measures the ratio of elements correctly identified over the total elements predicted in that class, the Recall measures the ratio of elements actually identified over the total number of elements present in that class in the validation set and the accuracy represents the overall precision measure in both classes. A graphical representation of this rationale is depicted on Fig. 3.14.
For each OVR classifier, these indicators are based on the so called *confusion matrix*, that represents the number of correct/incorrect classifications: it allows to compare forecast and realization for each company in the test set. As for the bankruptcy classification, in which Bankrupt is the positive class, the confusion matrix appears as the following:

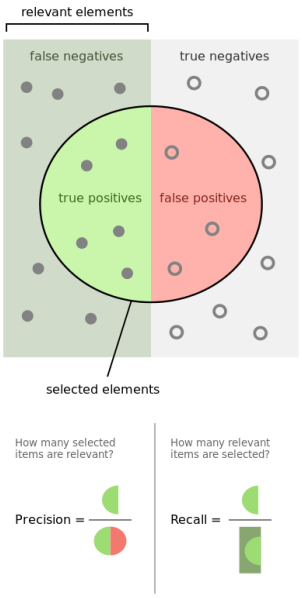

Figure 3.14: Graphical explanation of confusion matrix elements and performance indicators. Image taken from `https://commons.wikimedia.org/wiki/File:Precisionrecall.svg`

	Real Bankrupt	Real Not-Bankrupt
Predicted Bankrupt	# TP	# FP
Predicted Not-Bankrupt	# FN	# TN

Table 3.2: Sample bankrupt confusion matrix. T and F stand for *True* and *False*, while P and N for *Positive* and *Negative*.

$$Accuracy = \frac{mean(TP) + mean(TN)}{mean(FN) + mean(FP) + mean(TP) + mean(TN)} \tag{3.18}$$

$$+Precision = \frac{mean(TP)}{mean(TP) + mean(FP)} \tag{3.19}$$

$$+Recall = \frac{mean(TP)}{mean(FN) + mean(TP)} \tag{3.20}$$

$$-Precision = \frac{mean(TN)}{mean(TN) + mean(FN)} \tag{3.21}$$

$$-Recall = \frac{mean(TN)}{mean(FP) + mean(TN)} \tag{3.22}$$

Predictive performance example Let's suppose that there are 10 bankruptcies and bankruptcy is the positive class. If RF predicts 3 of them, then the positive recall equals to $\frac{3}{10} = 0.30$. If RF predicts also 5 more bankruptcies (which are actually non-bankruptcies) then the positive precision equals $\frac{3}{8}$.

3.6.2 RF procedure

The Random forest hyperparameters' space has been explored in a sequential way. First of all, the number of trees has been tuned evaluating the OOB error. Then, the number of selected covariates for each tree split (m_{try}) has been optimized with a built-in `R` function within the `randomForest` package, namely `tuneRF`. Its results are not worth to note, since it returned the default value of m_{try} suggested by the original work of Breiman [44], that is $m_{try} = \text{Floor}(\sqrt{p})$, where p is the number of predictors.
Finally, the labeling probability threshold has been optimized by the evaluation of the ROC curves, for each OVR classifiers. This value has been chosen as the one that corresponded to the curve *knee*, i.e. the optimal trade-off between True Positive and False Negative rates.

Cross validation

The cross validation is a statistical method which is used to obtain an estimation of the test-set prediction error and to investigate bias of the model's parameters. It consists in dividing the training set in a predefined number of folds K and fit the model K times, always holding out a different fold on which the model validation will be performed. The folds are randomly built and equal-sized.
So, the models is trained and tested K and its predictive performance indicators averaged across all the K values. Within RF training, K has been set to 10.

Balancing of the classes

Like other Machine Learning algorithms, RF accuracy is influenced by the distribution of the labels in the training set: the more the algorithm "sees" a label in the training set, the more it is able to recognize similar objects in the test set.
This will be a significant concern throughout all this work and various techniques will be used to address such issue. As for the RF, in the training set the classes have been balanced using an sampling with replacement of the minority class. Even if it is a simple technique, sampling with replacement provided good results within the Bhat & Zaelit work [11], that is closely related to this thesis' one.

Labeling probability threshold

The RF outputs consist in probabilities, one for each observation (i.e. company). According to the classification these are the probabilities for a company to go bankrupted, remaining private etc. However, in order to measure the predictive performance, these probabilities need to be translated to labels.
This is accomplished by setting the threshold p_{thr} for each classifier and labeling a company if its outcome probability is over such threshold. In this configuration, in order to label a company, the RF has to be sure at least at $p_{thr}\%$ to assign that label to a company.

Example The RF estimates that a company has 54% probability of being "Not-Bankrupt". If the threshold is 50% the assigned label is "Not-Bankrupt". If the threshold is 60% the assigned label is "Bankrupt".

3.6.2.1 RF hyperparameters selection

Below the hyper-parameters tuning procedure results, equipped with a brief commentary on the choice made.

Number of trees tuning

The plots in Fig.s 3.15, 3.16 represent the cumulative OOB error rates vs. the number of trees, grown in each OVR classifiers's Random Forest. The ith cumulative OOB is the error rate (i.e., $1 - Accuracy$) of all the trees in the forest up to the ith tree.
As for Fig.3.15, *Bankrupt classification* has an higher OOB error rate than the *Acquisition classification* when there are few trees in the forest. However, it sharply decreases within 30 trees and for both classifications the OOB error rate saturates close to 0, which means that the model has no doubts about the classification. (Note that it does not mean that the classifier makes no mistakes!). In this work, the number of trees has been set to 100. Both the aforementioned plots testify how the OOB rate decrease rapidly with the trees' number and saturates after approximately 30 trees. It is worth to note the difference scale between Fig 3.15 and 3.16: the latter saturate with a much greater error w.r.t. the first one. This is mainly due to the severe classes unbalancing, since the OOB, as the Accuracy, is a measure that does not take account of such issue as described in Subsec. 3.6.1.1). In fact, bankruptcy and acquisition classifications are the ones that suffer the most of this unbalancing (Tab. 3.1).

Probability threshold tuning

The probability threshold p_{thr} has been tuned evaluating the ROC (Receiver Operating Characteristics) curves. The ROC is designed plotting the TPR (*true positive rate*, named also Positive Recall) against the FPR (*false positive rate*) of the prediction performances of each model variant, in this case indexed by different values of p_{thr}.
Since these two measures are probabilistic, the ROC results to be a probability curve. This plot represents the capability of the classifier in distinguishing the classes: the bigger is the area under the curve (AUC), the stronger is the classifies. In light of such framework, it appears natural to choose the model by searching for the ROC *knee*, the point that optimize the TPR-FPR trade-off. In Fig.s3.17,3.20,3.18,3.19 there are two interesting things to observe. The first one is how fast the curve saturates to TPR = 1 (i.e. how similar it is to a perfect step), this characteristic reflects how well the model is able to separate the two classes. In fact, the higher the

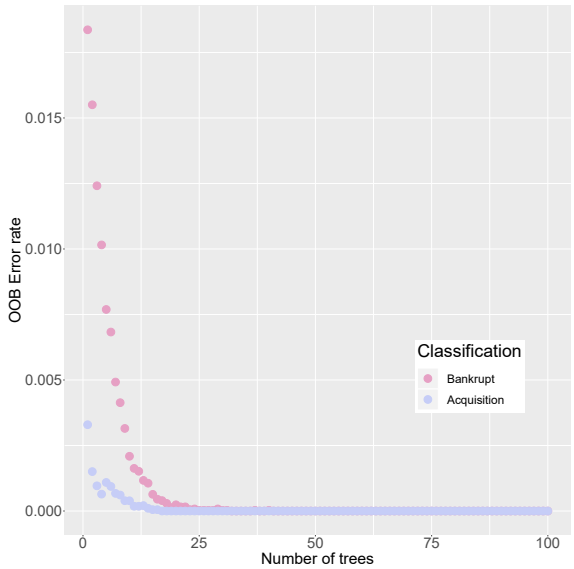

Figure 3.15: Out Of Bag error rate plot for bankruptcy and acquisition classification.

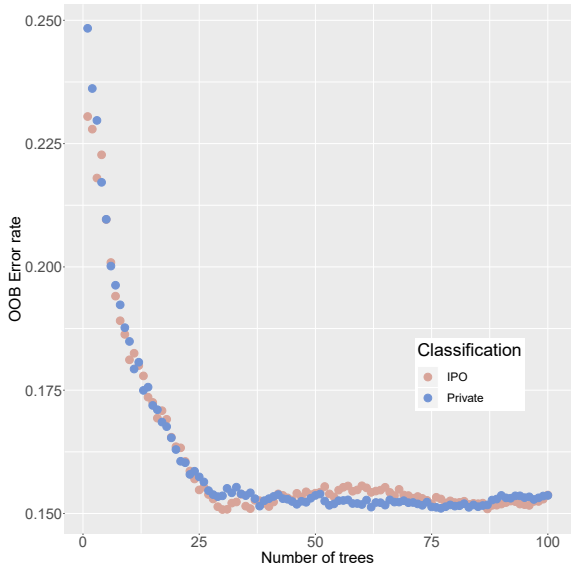

Figure 3.16: Out Of Bag error rate plot for IPO and Private classification.

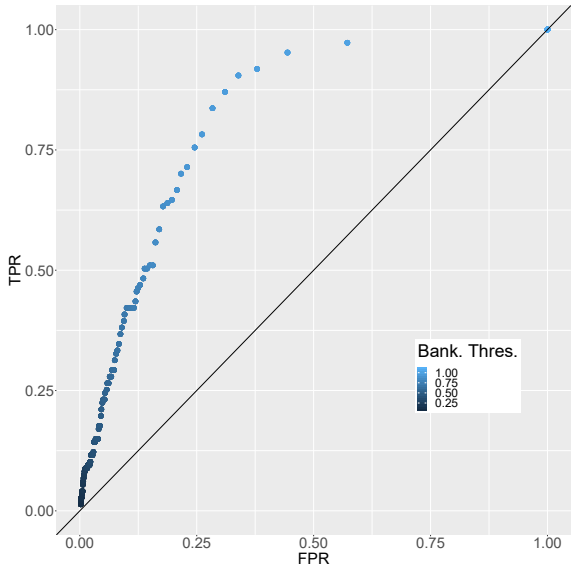

Figure 3.17: Bankrupt classifier, ROC curve on the bankruptcy probability threshold. Please note that this threshold is intended on the complementary event: the bigger it is, the easier is to predict a bankruptcy.

threshold, the more the algorithm will classify the minority class which resolves in a TPR increasing, however if the FPR increases too than it means that some points of the minority class are very much inside the cloud of the majority class points. The second thing to which pay attention is the color legend. Since the optimal threshold is a *ex post* hyperparameter, the ideally case is if the threshold (i.e. the color in the figures) is no lower than 0.50 when TPR = 0.50. As for Fig.3.17 the curve is quite satisfying and it turns out that the optimal probability threshold is in the [0.75-1.00] range. The ROC curves in Fig.s 3.18 and 3.19 are much more flat along the bisector, however in both cases for a FPR = 0.50 the TPR is greater than 0.60 which is a very good result. As for Fig.3.20, the curve is more similar the the Bankrupt's one and the optimal probability threshold results in the one which brings the TPR between 0.65 and 0.85.

3.6.2.2 Random Forest final results

Finally, the selected Random Forest models results, for each classification (Tab. 3.3). It is worth to note that the threshold probability selection procedure allowed to achieve good results as for the Positive Recall, without penalizing excessively the negative class prediction metrics.
While the accuracy is satisfying in each classifier, the Positive precision is low for the bankruptcy and acquisition one. Indeed, the acquisitions within our dataset constitute less than the 0.3%, so this result could be expected. As for the bankruptcy classifier, the severe unbalancing has an heavy influence on the minority class precision. In general, such results are in line with the Bhat and Zaelit's work [11].

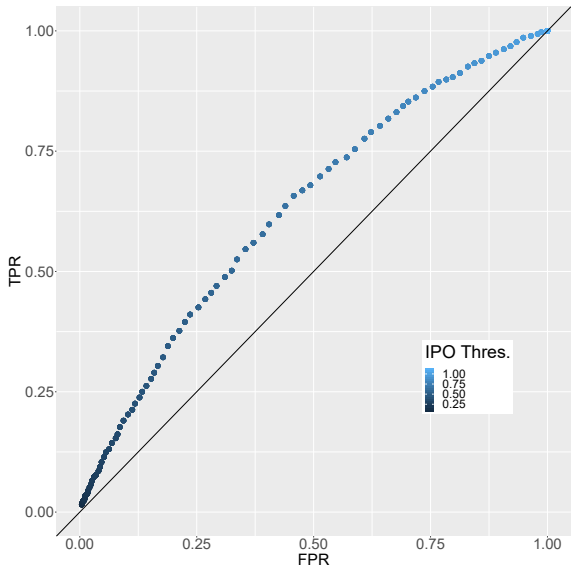

Figure 3.18: IPO classifier, ROC curve on the IPO probability threshold. Please note that this threshold is intended on the complementary event: the bigger it is, the easier is to predict an IPO.

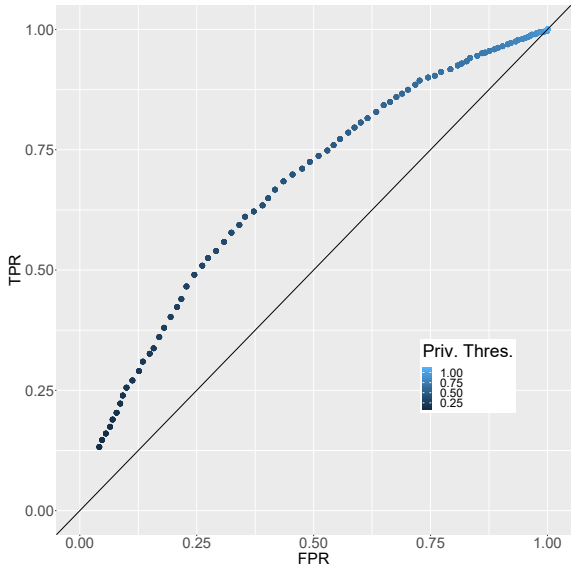

Figure 3.19: Private classifier, ROC curve on the stay-private probability threshold. Please note that this threshold is intended on the real Private event: the bigger it is, the harder is to predict a stay-private event.

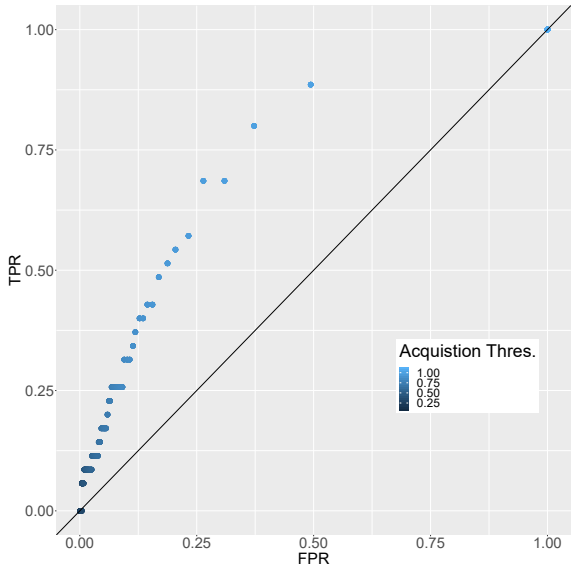

Figure 3.20: Acquisition classifier, ROC curve on the Acquisition probability threshold. Please note that this threshold is intended on the complementary event: the bigger it is, the easier is to predict an acquisition.

Class.	Precision+	Recall+	Precision-	Recall-	Accuracy	Prob. Thres.
Bankrupt	0.105	0.708	0.988	0.796	0.794	0.900
IPO	0.451	0.551	0.727	0.641	0.610	0.600
Private	0.714	0.691	0.492	0.519	0.629	0.450
Acquisition	0.017	0.477	0.995	0.778	0.775	0.950

Table 3.3: Final Random Forest predictive performance indicators, fro each OVR classifier. Please note that the probability threshold is the one that optimizes the TPR/FPR trade-off represented in ROC curves. Private probability threshold is considered on the *Private* event, while the others on the complementary events.

3.6.3 Neural Networks Experiments

The experimental procedure followed within the Neural Nets model tuning regarded both the hyperparameters selection as well as the model choice between MLP and LSTM. Several design choice had to be performed: network width and depth, regularization techniques and hyperparameters, training set oversampling techniques. As for the hyperparameters, each step is consequential w.r.t. the previous one, since it reduces the grid search w.r.t. the hyperparameter that minimizes the classification error (i.e., the one that maximizes the predictive performance on the test set). A similar approach has been applied to the oversampling techniques selection. Because of the complexity of the unbalancing impact on the predictive performance, the undersampling and oversampling procedure have been parallelized, in order to explore the NN models used in different data space environments.
Surely this is a *greedy* approach, but within NNs training the high dimensionality of hyperparameters grid make this choice the most feasible one in term of computational time.

Below a brief summary of the procedure:

1. **Base architecture** The base network architecture, both for MLP and LSTM, is composed of 14 hidden units and 14 output ones
2. **Training dataset undersampling** Training population is balanced by undersample the majority class until it has the same number of points of the minority one.

   (a) **Optimizer choice** Choice between SGD and ADAM optimizer according to the highest final epoch accuracy. The optimization algorithm selected is the one with the highest accuracy on the final epochs.

   (b) **Regularization** Dropout and L^2 norm regularization are performed with different dropout rates and λ, respectively.

3. **Training dataset oversampling** For each binary classification, the training set has been oversampled thanks to 3 advanced techniques: *SMOTE*, *ADASYN* and *SVMSMOTE*, described in the following subsections. Oversampling have been tested using always the base architecture's net. The best performing technique in the same classification has been chosen for the next step.
4. **Network width & Architecture tuning** After the comparison of the oversampling technique, a neurons' number optimization have been performed. That is, only for the best oversampling method, different combinations for the number of neurons in the hidden and output layer have been tested. For each combination, the training loss and the validation loss after 200 or 100 epochs have been compared.
5. **Regularization** Finally, different dropout rates have been tested on the models selected on each binary classification by the previous procedures.

The oversampling procedure has been performed only for the MLP, in order to reduce the total time of the experimental procedure regarding NNs tuning. It is worth to note that in Appendix A, deep architectures (with more than one hidden layer), has been explored.

Moreover, in Annex B, the gradient norm analysis has been performed to validate the optimizer robustness w.r.t. the optimization issues described in Subsec. 3.4.

3.6.3.1 Preprocessing

First of all the input variables have been first standardized through a simple mean-variance scaling. This is done in order to ease the optimization process NNs training relies on.

As for the response vector, the label y, the *one-hot encoding* transformation has been performed. The response vector is a categorical vector, but this cannot be used within the `Python` neural networks libraries, such as `keras`, the used one. Such codes accept only one-hot encoded response for classification algorithms. This method takes as input a label, e.g. Bankrupt and transforms it in a 2 elements vector, e.g. $[1, 0]$. When a Not-Bankrupt label is presented instead, the new response

vector will be $[0, 1]$. In a nutshell, one-hot encoding builds a simple map on the labels.

In order to overcome the classes unbalancing issues, simple and advanced sampling techniques have been tried.

Data-set augmentation: simple methods In order to perfectly balance the classes within a training dataset, the undersampling can be exploited. This approach undersamples the majority class training population until it has the same number of points of the minority one. It is a fairly naive method, especially when the number of minority class points is *low*, as in the case of the reference dataset under study. However, since it preserves the data space structure, as it does not manipulate the points, it has been tried within NNs experiments.
An example on the effect of undersampling is given in Tab. 3.4, where the final training set minority classes populations are depicted for each classification. Please note that they will be exactly the same as the majority one, that will coincide with their complementary (i.e. with Not-Bankrupt).

Classification	Training Minority Population
Bankrupt	942
Acquisition	231
IPO	10617
Private	10628

Table 3.4: Training set undersampling. For each classification, the number of observations of the minority class after the undersampling procedure is reported.

Dataset augmentation: advanced methods The simplest of the advanced data augmentation technique is the *Synthetic Minority Oversampling Technique* (SMOTE) algorithm [45]. It is a geometrical distance-based method whose working process can be summarized as:

1. Randomly pick a minority class point x_i, considering its k nearest-neighbors.
2. Randomly pick a point x_{zi} among them
3. Create a new point x_{new} on the segment between x_i and x_{zi} at a random distance from x_i. In formulas:

$$x_{new} = x_i + \lambda(x_{zi} - x_i) \tag{3.23}$$

   with $\lambda \in [0, 1]$ a random number.

Two main variants of the SMOTE have been developed. The *ADASYN* method [46] is similar to the SMOTE, but for each x_i in the minority class generates a number of x_{new} proportional to the other classes number of points.
The *SVM SMOTE* [47] exploits a SVM classifier is used to find support vectors that are then considered within the samples generation process.

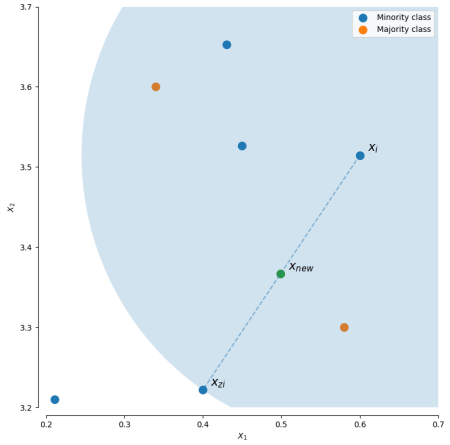

Figure 3.21: Scheme of the SMOTE working principle. Image from `https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html`

3.6.3.2 Dataset Augmentation tuning

Within the dedicated plots, the notation "(14,14)" stands for a network architecture composed by 14 hidden neurons and 14 output neurons. For each binary classification, SMOTE, ADASYN and SVMSMOTE oversampling have been tested using the base architecture's net. That is, one input layer with 14 outputs and one hidden layer with 14 outputs with the selected regularization.

As for the LSTM network, in Fig.s3.22, 3.23, 3.24 and 3.25 it is important to note which curve reaches the lowest error value in the train plot as the number of epochs increase. In the test plot is natural to expect a slight worsening in performance (i.e. a higher error), but this should a too large difference would imply an overfitting issue. Another feature to take into account of in the selection of the augmentation method is how noisy the test curve is. A fast oscillation makes the estimation of the test error final value very unreliable.
In light of these observations, within the bankruptcy classification, Fig.3.22 shows how the best method is the SVMSMOTE in the k = 2 configuration. As for the IPO models, in Fig.3.23 SVMSMOTE, outperforms the other method in the training loss plot, however SMOTE and ADASYN have a test loss more similar to the train one, within the first 100 epochs. The same holds also for Private classification plots Fig.3.24. To conclude, regarding Acquisition classification the SVMSMOTE technique works better than the others since the test loss results in a quite lower value, Fig.3.25.

As for the MLP network the results of the loss score within the first 200 epochs for different data augmentation techniques are reported in Fig.s 3.26, 3.27, 3.28, 3.29. In general, the MLP training curves saturate earlier than the LSTM-related ones. This is particularly evident in Fig.3.29. Moreover the difference between the starting training loss and the value after 100 epochs is much more significant w.r.t the LSTM. On top of that, the final loss score results to be lower w.r.t. the previous

architecture as well as in the test loss plots. In Fig.s 3.26 and 3.29 the SVMSMOTE technique leads to the best results, while for Fig.3.27 the method selected is the SMOTE since, although it performs slightly worse than the SVM one, its training loss score is much more similar to the test one and the initial fuzzy issue in the test plot completely disappear after less than 50 epochs. On the other hand, the MLP Private classifier's (Fig.3.28) the test loss score has been evaluated to be to much fuzzy across all epochs and moreover not particularly better than the LSTM result Fig.3.24. For sake of completeness, for this classification, the tables with the performance indicators have been reported in Tab.s3.5 and 3.6. The indicators performance of the LSTM are overall better than the MLP ones, in particular it is worth to note that the Negative recall and precision which refer to the Private class. For all these rationales the LSTM architecture is chosen for further tuning for the Private classification.

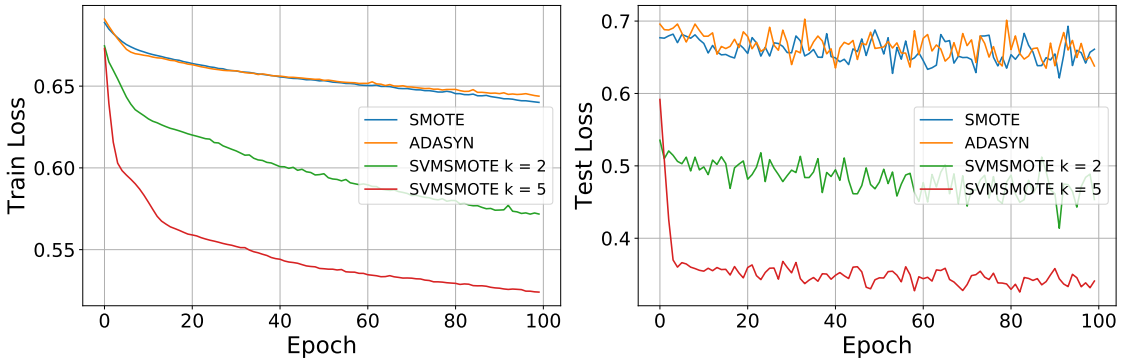

Figure 3.22: LSTM Network, bankruptcy classification. Train and test loss score across 100 epochs training, based on different advanced oversampling techniques. Network architecture: (14,14).

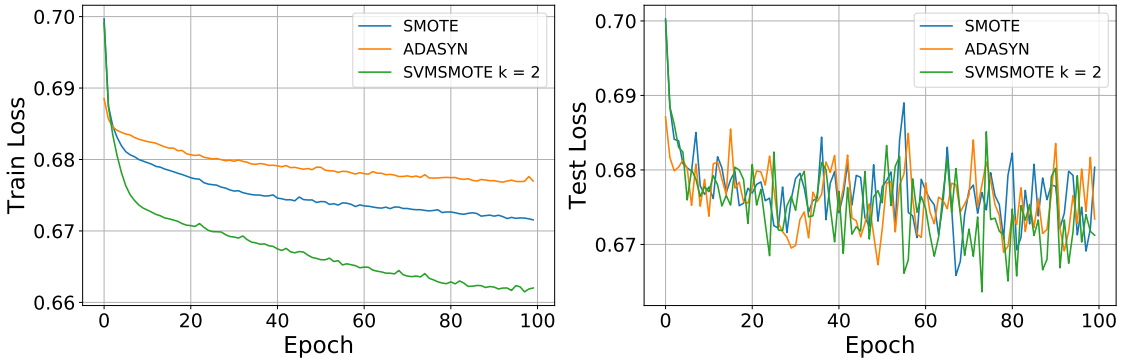

Figure 3.23: LSTM Network, IPO classification. Train and test loss score across 100 epochs training, based on different advanced oversampling techniques. Network architecture: (14,14).

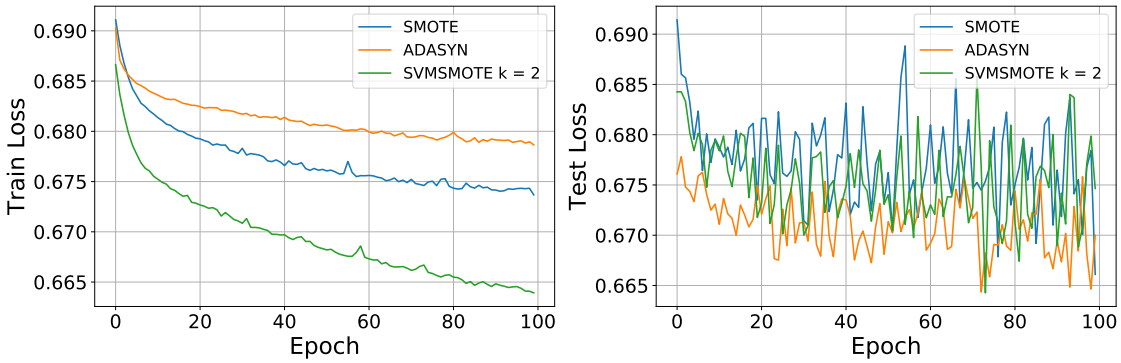

Figure 3.24: LSTM Network, Private classification. Train and test loss score across 100 epochs training, based on different advanced oversampling techniques. Network architecture: (14,14).

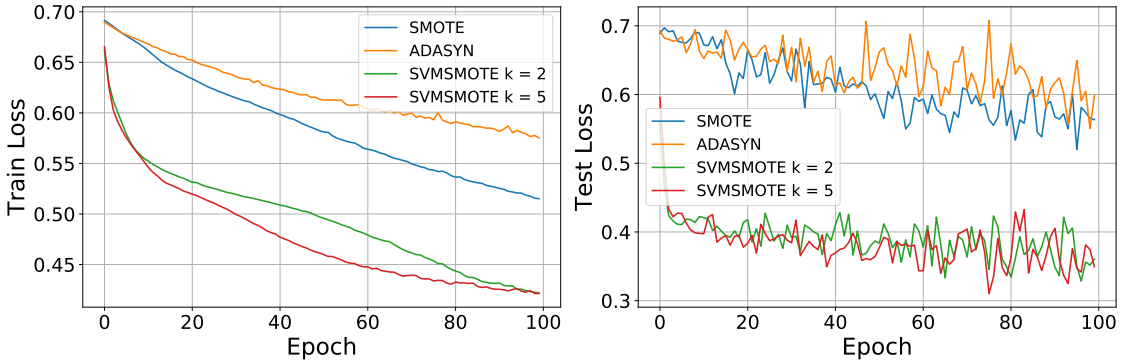

Figure 3.25: LSTM Network, acquisition classification. Train and test loss score across 100 epochs training, based on different advanced oversampling techniques. Network architecture: (14,14).

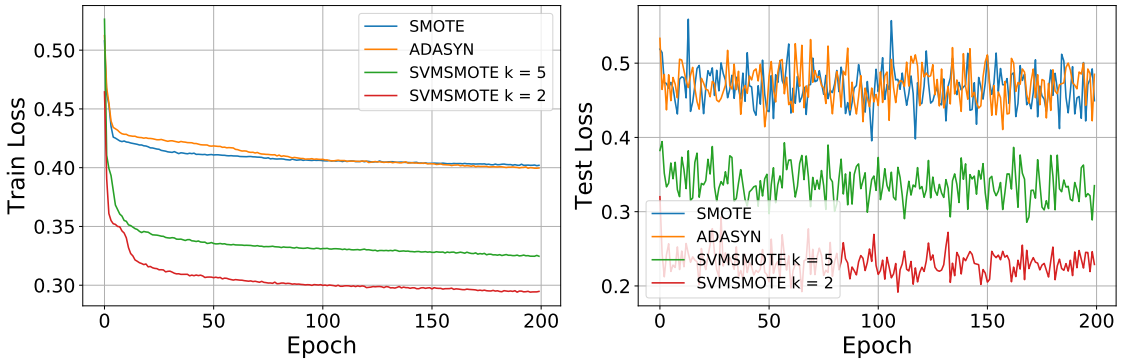

Figure 3.26: MLP Network, bankruptcy classification. Train and test loss score across 200 epochs training, based on different advanced oversampling techniques. Network architecture: (14,14).

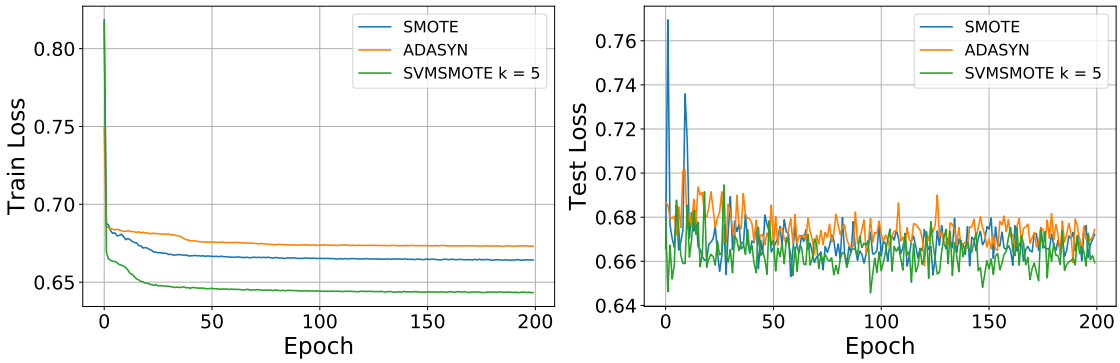

Figure 3.27: MLP Network, IPO classification. Train and test loss score across 200 epochs training, based on different advanced oversampling techniques. Network architecture: (14,14).

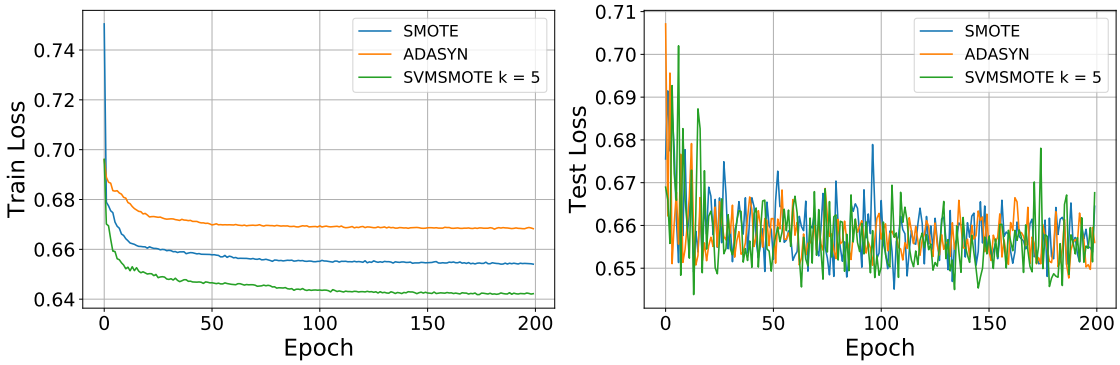

Figure 3.28: MLP Network, Private classification. Train and test loss score across 200 epochs training, based on different advanced oversampling techniques. Network architecture: (14,14).

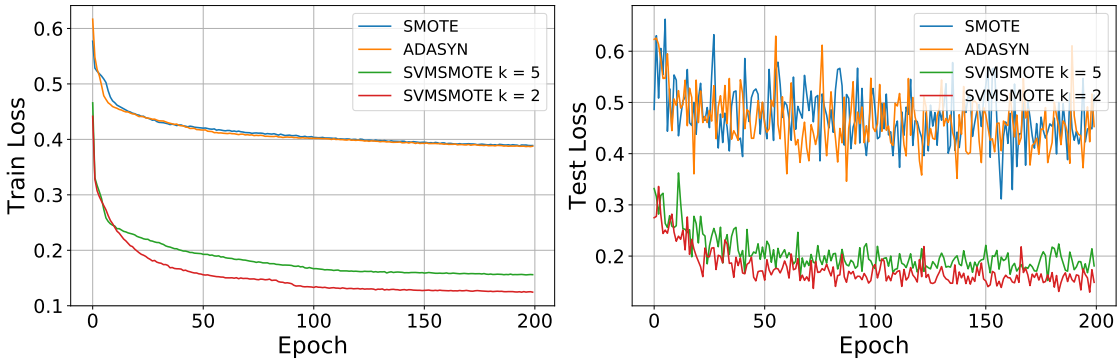

Figure 3.29: MLP Network, Acquisition classification. Train and test loss score across 200 epochs training, based on different advanced oversampling techniques. Network architecture: (14,14).

	Precision+	Recall+	Precision-	Recall-	Accuracy
SMOTE	0.28	0.38	0.53	0.41	0.598
ADASYN	0.28	0.42	0.53	0.38	0.608
SVMSMOTE k=5	0.28	0.39	0.54	0.41	0.597

Table 3.5: Private/Not Private MLP classification. Train set over sampled with different algorithms: SMOTE, ADASYN, SVMSMOTE. Train set distribution: 20218 Private and Not Private (for SMOTE), 20218 Private and 19389 Not Private (for ADASYN), 20218 Private and 20218 Not Private (for SVMSMOTE k = 5). Test set distribution: 5738 Private, 10072 Not Private. Please note that the positive class is Not-Private.

	Precision+	Recall+	Precision-	Recall-	Accuracy
SMOTE	0.435780	0.540189	0.687187	0.590877	0.572170
ADASYN	0.455258	0.448158	0.680111	0.686316	0.598419
SVMSMOTE	0.440801	0.520651	0.686365	0.613634	0.579317

Table 3.6: Private/Not Private LSTM classification. Train set over sampled with different algorithms: SMOTE, ADASYN, SVMSMOTE. Please note that the positive class is Not-Private.

3.6.3.3 Network width & Architecture tuning

The number of neurons for each selected classifier has been tuned trying different combination of them and comparing the classifiers performance. In the end the best models result to be:

- Bankrupt MLP classification: SVMSMOTE with k = 2 with (21,14) or (14,14)
- IPO MLP classification: SMOTE with (7,6)
- IPO LSTM classification: SMOTE with (28,4)
- Private LSTM classification: ADASYN with (28,14)
- Acquisition MLP classification: SVMSMOTE with k = 2 with (28,14)

In Fig.s3.30 and 3.31 the final training loss is reported for each LSTM net architecture tried. For sake of brevity, the MLP plots are not reported.
As for Fig.3.30 the lowest training loss score is reached by the model architecture (21,2), however its validation loss is higher and comparable with the ones of the other models. This rationale leads to the selection of architecture (28,4), which does not present such gap between the training and validation loss. On top of that, (28,4) model has a slightly lower validation loss than the (21,2) one.
The Private LSTM shows a constant trend in validation loss across different architectures, so it is chosen the one that minimizes the overfitting-validation loss trade-off: (28,14).

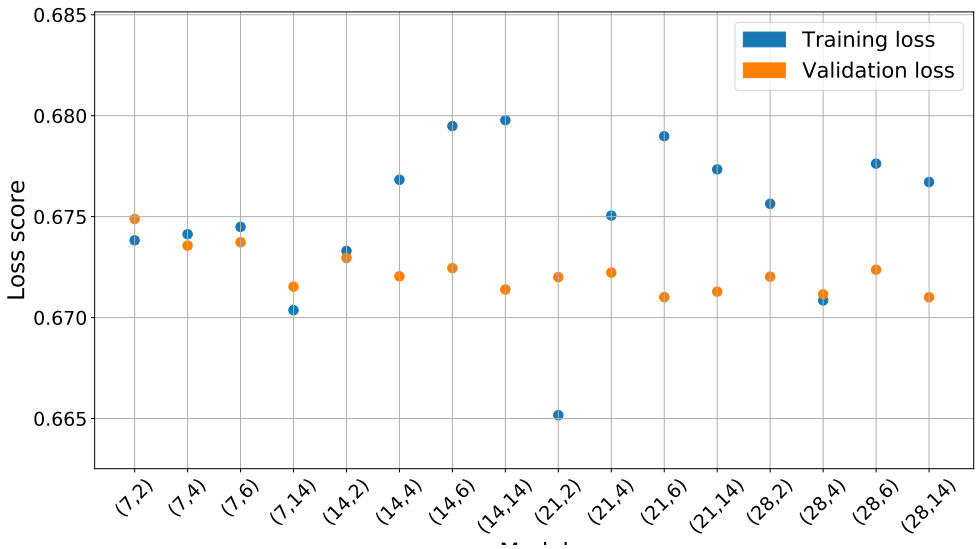

Figure 3.30: LSTM network, IPO classification. Final loss after 100 epochs for different neurons configurations, noted as (# hidden layer neurons, # output layer neurons) on the x axis. Best oversampling technique: SVMSMOTE with k = 2. Best model selected: (28,4).

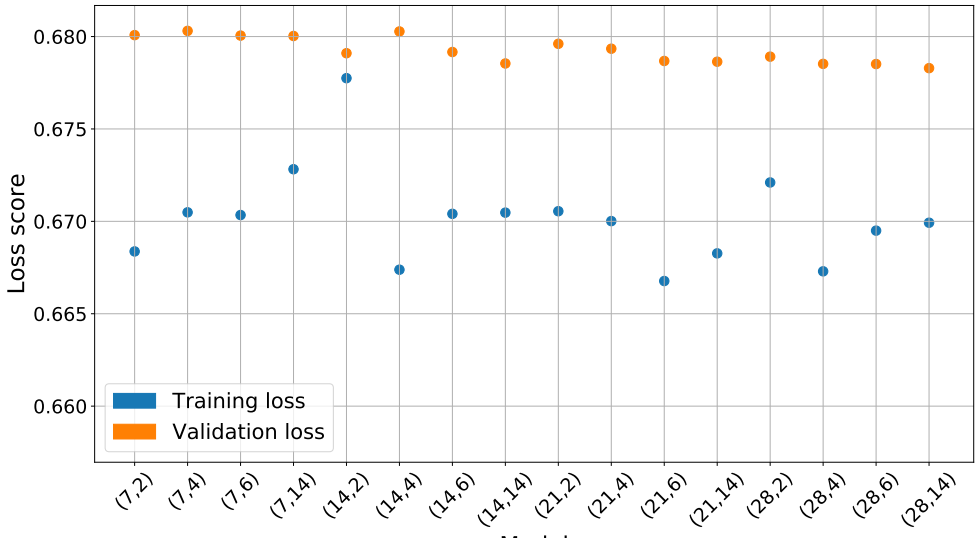

Figure 3.31: LSTM network, Private classification. Final loss after 100 epochs for different neurons configurations, noted as (# hidden layer neurons, # output layer neurons) on the x axis. Best oversampling technique: ADASYN. Best model selected: (28,14).

Final Dropout tuning

In Tab.s from 3.7 to 3.12 the performance indicators for the best models listed in the previous section, with different levels of dropout are presented. A key indicator is the positive recall, (*Recall+*), since it measures the models' capability to recognize the extremely interesting minority class events (bankruptcies and IPO).
This is particularly true as for the bankruptcy classification, since, from the practitioner point of view, the recognition of such pattern consists in the difference between a total investment loss and the other investment returns classes. In light of this, the MLP net (14,14) with a dropout = 0.5 has been chosen as the best performing model. Even if it presents the lower negative recall and the lower positive precision (without a big gap), it presents the highest positive recall with a very interesting absolute value (Tab.s3.7 and 3.8).
Regarding all the other classifications, the dropout configuration which results in a higher value of the *Recall+* has been selected. This with a mindful lower limit on the other indicators, so that they were not too muhc penalized w.r.t. the other configurations of dropout.

The best models selected are highlighted in the tables.
Along with the indicators performance tables, the loss for both train and test have been plotted within the number of epochs. Two representative examples are reported in Fig.s 3.32 and 3.33. In these figures it is important to note how the dropout rate increasing induces the loss curve becomes to be smother and less *"noisy"*. According to the results in Tab.s 3.7 and 3.11 the best curves are the one that correspond to a dropout rate equal to 0.5, as for bankrupt classification, and the one with dropout

rate = 0.3 as for the LSTM model IPO classification.

	Precision+	Precision-	Recall+	Recall-	Accuracy
No drop	0.129991	0.978032	0.482759	0.876953	0.864390
Drop = 0.1	0.129909	0.982070	0.593103	0.848720	0.839469
Drop = 0.3	0.128593	0.981771	0.586207	0.848720	0.839469
Drop = 0.5	0.109396	0.985979	0.708621	0.780302	0.776154

Table 3.7: MLP net with 14 neurons in the hidden layer and 14 neurons in the output layer. Bankrupt classification.

	Precision+	Precision-	Recall+	Recall-	Accuracy
No drop	0.121933	0.980155	0.548276	0.849639	0.840354
Drop = 0.1	0.128003	0.979770	0.532759	0.861786	0.851233
Drop = 0.3	0.128310	0.982025	0.593103	0.846553	0.837887
Drop = 0.5	0.112396	0.983840	0.653448	0.803480	0.797090

Table 3.8: MLP Net with 21 neurons in the hidden layer and 14 neurons in the output layer. Bankrupt classification.

	Precision+	Precision-	Recall+	Recall-	Accuracy
No drop	0.024233	0.993088	0.125000	0.961504	0.955155
Drop = 0.1	0.014028	0.992844	0.116667	0.937285	0.931056
Drop = 0.3	0.018982	0.993311	0.183333	0.927533	0.921885
Drop = 0.5	0.017446	0.993392	0.208333	0.910261	0.904934

Table 3.9: MLP net with 28 neurons in the hidden layer and 14 neurons in the output layer. Acquisition classification.

	Precision+	Precision-	Recall+	Recall-	Accuracy
No drop	0.413	0.740	0.590	0.582	0.585
Drop = 0.1	0.423	0.727	0.506	0.657	0.606
Drop = 0.3	0.437	0.714	0.401	0.743	0.629
Drop = 0.5	0.437	0.712	0.394	0.747	0.630

Table 3.10: MLP net with 7 neurons in the hidden layer and 6 neurons in the output layer. IPO classification.

	Precision+	Precision-	Recall+	Recall-	Accuracy
No drop	0.407860	0.724490	0.527307	0.618854	0.588425
Drop = 0.1	0.405980	0.723489	0.527117	0.616011	0.586464
Drop = 0.3	0.400189	0.727413	0.564225	0.578967	0.574067
Drop = 0.5	0.404415	0.725838	0.543863	0.601232	0.582163

Table 3.11: LSTM net with 28 neurons in the hidden layer and 4 neurons in the output layer. IPO classification.

	Precision+	Precision-	Recall+	Recall-	Accuracy
No drop	0.463643	0.676174	0.406512	0.724912	0.607400
Drop = 0.1	0.470184	0.676977	0.398629	0.737243	0.612271
Drop = 0.3	0.451418	0.680743	0.461011	0.672281	0.594307
Drop = 0.5	0.458014	0.673100	0.399143	0.723709	0.603922

Table 3.12: LSTM net with 28 neurons in the hidden layer and 14 neurons in the output layer. Private classification.

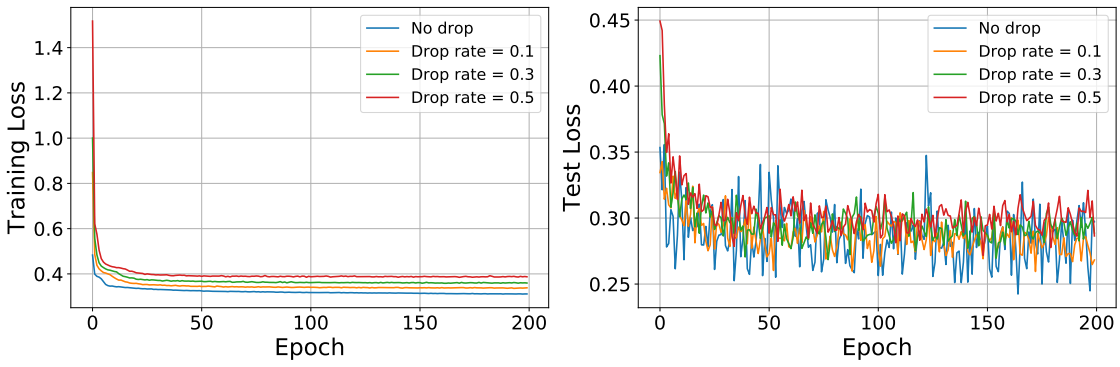

Figure 3.32: MLP network, dropout rate tuning on the selected bankruptcy model. Train and test loss for each dropout rate. Network architecture: (21,14)

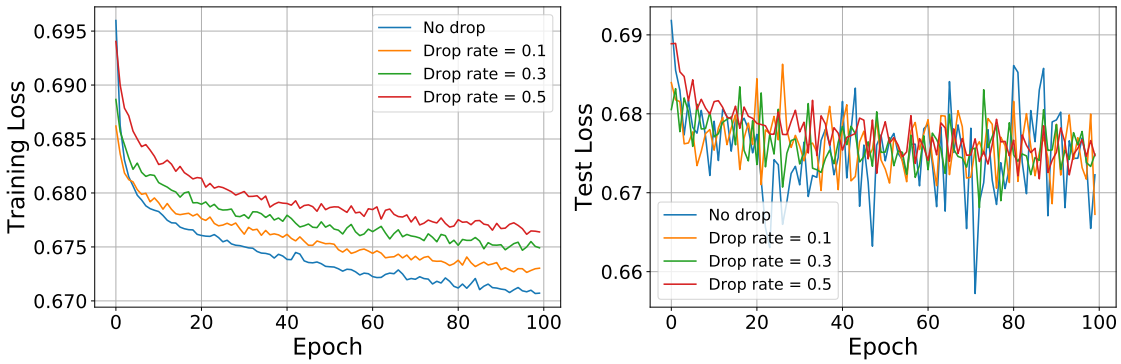

Figure 3.33: LSTM network, dropout rate tuning on the selected IPO model. Train and test loss for each dropout rate. Network architecture: (28,14)

3.7 Results

In this section, a comparison between the results of the final RF classification models and the final NNs models are presented for each classification. Both algorithms have been trained on the same dataset and then evaluated on the same test set. As for Bankrupt and IPO classifications, the NN results are coherent w.r.t. the RF ones. However, in order to consistently recognize the minority class (i.e. to have an high positive recall), RF needs to have a low minority class probability threshold, favouring the prediction of the this class itself. Only in this case the RF performance barely reaches the NN one.

- This is indeed not the optimal procedure: the probability threshold tuning is an *ex post* procedure, so it is possible that a periodic threshold tuning may be needed. Instead, the NN predictions are based always on a 50% (neutral) probability threshold, so it is a more robust result.

As for Private classification, NN basically averages the performance of the 2 thresholded RFs, so it confirms the RF results. As for Acquisition classification, the results are quite similar among classifiers. RF positive recall is extremely sensible w.r.t. probability threshold. Since the unbalancing of the Acquisition classification is extreme, the negative recall is very high for all the classifiers. In general, NN confirms the RF optimal results.

Th. Pr.	Precision+	Precision-	Recall+	Recall-	Accuracy
0.90	0.105	0.988	0.708	0.796	0.794
0.50	0.094	0.978	0.330	0.903	0.886
NN	0.109	0.991	0.711	0.801	0.809

Table 3.13: Bankrupt classification. Final results of the NN and RF. The NN in the MLP architecture with (14, 14) neurons' configuration and dropout = 0.5. The RF with both the optimal probability threshold and the 0.50 p.t. as benchmark. Bankrupt is the positive class.

Th. Pr.	Precision+	Precision-	Recall+	Recall-	Accuracy
0.60	0.451	0.727	0.551	0.641	0.610
0.50	0.4578	0.7092	0.3433	0.7975	0.6465
NN	0.412	0.738	0.596	0.576	0.582

Table 3.14: IPO classification. Final results of the NN and RF. The NN in the MLP architecture with (7, 6) neurons' configuration without dropout regularization. The RF with both the optimal probability threshold and the 0.50 p.t. as benchmark. IPO is the positive class.

Th. Pr.	Precision+	Precision-	Recall+	Recall-	Accuracy
0.45	0.714	0.492	0.691	0.519	0.629
0.50	0.5082	0.6968	0.4463	0.7466	0.6356
NN	0.451	0.683	0.460	0.673	0.594

Table 3.15: Private classification. Final results of the NN and RF. The NN in the LSTM architecture with (28, 14) neurons' configuration and dropout = 0.3. The RF with both the optimal probability threshold and the 0.50 p.t. as benchmark. Not-Private is the positive class.

Th. Pr.	Precision+	Precision-	Recall+	Recall-	Accuracy
0.95	0.017	0.995	0.477	0.778	0.775
0.50	0.0226	0.9927	0.0963	0.9672	0.9604
NN	0.021	0.992	0.228	0.906	0.898

Table 3.16: Acquisition classification. Final results of the NN and RF. The NN in the MLP architecture with (28, 14) neurons' configuration and dropout = 0.5. The RF with both the optimal probability threshold and the 0.50 p.t. as benchmark. Acquisition is the positive class.

Chapter 4

Survival Analysis

In this chapter a brief background explanation is provided, followed by the Survival Models theoretical basis exploration. The procedure and the experiments will complete the picture.

4.1 Goal & Background

Goal of this section is to estimate the probability for a company to go public within a fixed point in time. The estimation done is based on the so called *Survival Models* (SM), that relies on what is defined as a *Survival Data* (SD) data-set. This is composed by the date of the interest event and the subjects features (covariates). In the first part of this chapter, the SM type to be used is selected and tuned w.r.t. the selected covariates and to its parameters. The families of SM explored are the *Kaplan-Meier* models, the *Accelerated Failure Time* (AFT) models and the *Cox* models. Lastly, the estimation in time of the IPO probability is merged to the probability estimation of the other possible outcomes (Bankrupt, Acquisition, Private) provided by the other models used within this study (Chapt. 3), such as the Random Forest. This merging was necessary because of data quality, since dates of the other events, acquisition and bankruptcies, are not provided by the data source.

Background

The SM are broadly used within the medical environment, especially when the evolution in time of serious diseases (such as cancer) has to be modelled [48, p. 14]. In particular, SM has been used to study the probability of response to medical treatments in humans and animals, the development of diseases, the identification and evaluation of prognostic factors and risks linked to those diseases, [49]). Within this research, the subjects under study will be the companies within the reference dataset (Chapt. 2).

The generalization capacity of SM has been exploited also in many fields far from the

medical ones. Survival Data can be mined in a broad range of sectors: criminology, industrial production, reliability engineering, economics etc. [48, p. 14], [8], [50]. Roughly speaking, in every field in which there is something that *can fail* and there is an interest in modeling the probability in time of its failure, SD can be designed in that direction and Survival Models can be applied.
It is easy to infer that the notion of *failure* can be freely interpreted: if the event of interest is indeed favourable, the nomenclature used within the models will be different, but the mechanics of the models themselves will be the same.

An essential feature of Survival Models (SM) is their ability to handle *Censored Data* CD. In fact, what happens if a subject under study exits from the study itself and provides no more information about its status? The Survival Analysis method provide an effective way to include also this kind of data, whose link with the PE market data is explained in the following. In particular, how CDs are handled can bias in different way the result.

4.2 Theory & Procedure

Within this section the theoretical bases of the Survival Models are explored.
Every survival analysis is built on observations recorded during an experiment. The experiment can be fully designed by the analysts or it can be composed by the observation of historical data of a given environment. Within this study, the historical data are the ones provided by the Thomson Eikon platform and the environment is the US-Canada-Europe Private Equity market between 1998 and 2018 (Chapt.2). The scope of the SM (and consequently of the survival analysis) is to estimate the probability distribution of the random variable T (T is the time to IPO), for each company.

4.2.1 Survival Data

As mentioned above, the SM are fitted on Survival Data (SD), a data set composed by some dimensions that are specific of these models. SD can be structured in many ways, but need to contain at least two dimensions:

- **Survival time** T This quantity measures the time from the beginning of the study to the event of interest E. The event could be the failure of a mechanical component, the death of a subject in a medical trial, a divorce or a company that goes on IPO. T is a Random Variable whose probability distribution has to be estimated according to the experimental Survival Data. Each subject i has its own distribution of its T that has to be fitted by SM.

- **Censored Dummy** A dummy variable that states whether a subject's survival time is censored or not.

An important, but not essential type of dimensions are the **Subject Features** (SF). In fact, the SD matrix can be enriched by some features regarding the properties

of each subject, such as the type of medical treatment, the gender, the information about investors that invested in the company etc. Obviously the type of the additional features deeply depends on the application domain of the study.
Please note that Survival Time and Censored Dummy are *sufficient* to fit a SM. However, the presence of the Subject Features is essential in order to identify the *Prognostic Factors* as the following sections clarify (Subsec. 4.5.4).

4.2.1.1 Experimental Design

Survival Data are typically built using a simple procedure [51]. The analysis can rely either on a real-world experiment or on the observation of historical data. The observer that gathers SD needs to decide some key features of the experimental design:

- **Determine the subjects set.** The set of the subjects to study, in time and space. In this study, the subjects set are the companies of the reference dataset described in Chapt.2.

- **Determine the Observation Period (OP).** The time window in which the realization of the event of interest E of each subject is recorded. The initial t_{start} and final t_{end} times determine the length of the observation period.

Recording of the Survival Time. Each subject under study will experience E or not during the OP. The time of the entry in the study must be recorded as well as the exit caused by the realization of E or by the end of the OP determined by t_{end}. According to the order in time of these events, the censoring dummy variable will vary.

4.2.2 Censored Data

At this point, a description on what are the CD and their usage is a duty [52].
Within life or social sciences studies, it often occurs that a subject under study does not experience the event of interest by the end of the period of study. For example, in this context a company under study may not have been publicly listed before the end of 2018 or the data provider has not yet updated its status even if the company went public. The *censoring* can also occur when the subject information flow ends before the end of the study.
Indeed, there are 3 types of censoring:

- **Type I censoring** Suppose that some subjects have not experienced the event of interest before the end of the study or that some of them exit from the study in an *accidental* way. As an example, let's consider an experiment in which 6 rats are exposed to carcinogens [52]. Before the end of the study, 3 rats died by cancer. 1 rat died accidentally without tumors, 1 rat managed to escape from the laboratory and 1 rat survived (in its cage). So the first 3 rats survival times won't be censored, the ones of the latter 3 will be censored of type I.

- **Type II censoring** This type of censoring is used when the researcher has the full control over the experiment. It can be decided than that experiment will end when a number N_{stop} of subjects will experience the event of interest. The subjects that has not experienced the event by that time will have a censored survival time.

- **Type III censoring** In some life science experiments, the subject can enter in the study after the beginning of the study itself. Than, the survival time of each of them must be shifted by the date of entry in the study. If a subject is lost to follow-up or it has not experienced the event before the end of the study, its T is censored. This type of censoring is also called *random censoring*.

In a nutshell, censoring encodes the lack of knowledge about a subject whose information stream flowing to the observer ends up for some reason: the subject gives up the study or the study itself ends before the subject has experienced the event of interest E. In some sense, the latter case can be interpreted as "*the study gives up with respect to the subjects that still have not experienced* E". Both the events, subject or study giving up, generate a censored survival time T.

4.2.3 Functions of survival time

Each family of SM attempts to fit the functions of survival time T, that are indeed functions of its probability distribution.
Below the most important ones ([53]):

- **Probability Density Function $f(t)$** Since T is a continuous random variable, it has a density function $f(t)$ representing the probability of failure in an infinitesimal interval of unit time.

$$f(t) = \frac{\lim_{\delta t \to 0} \mathbb{P}(event\ E\ occurs\ \in (t, t+\delta t))}{\delta t} \tag{4.1}$$

  $f(t)$ is a non-negative function and the total area under its graph is equal to 1.

- **Survival Function $S(t)$** It is the probability that a subject under study experiences the event of interest after time t:

$$S(t) = \mathbb{P}(E\ occurs\ after\ t) = \mathbb{P}(T > t) \tag{4.2}$$

  From the basic rules of probability, the survival function is a non-increasing function of time t that has the following properties:

$$\lim_{t \to 0} S(t) = 1$$
$$\lim_{t \to \infty} S(t) = 0$$

- **CDF of T** From the rules of probability, $F(t) = 1 - S(T)$

- **Hazard Function $h(t)$** It is defined as the probability of experiencing E in an infinitesimal time interval conditioned to the survival of the subject until that time ([54]). in other words, for the subjects survived until t, $h(t)$ represents the probability that E will happen in the next, very short, time interval $(t, t+\delta t)$. In formulas:

$$h(t) = \frac{\lim_{\delta t \to 0} \mathbb{P}(E \; occurs \; \in (t, t+\delta t) | survival \; until \; t)}{\delta t} = \frac{f(t)}{1 - F(t)} \quad (4.3)$$

  Where the latter equality can be proved by the Bayes Theorem (REF?).

4.2.4 Procedure used & SM adaptation

The procedure adopted has been intended to adapt the Survival Models to a prediction model within the Private Equity market. That is why some nomenclature has been adapted to this specific application domain, as well as the experiment design.

4.2.4.1 Assumptions

From the mathematical and the Private Equity practical domain point of view, the analysis of this chapter needed some assumptions. Without them, the *real-world* problem under study could not have been captured by the model.

The assumptions done are:

- Companies will either go IPO, Bankrupt or they will be Acquired. The Private outcome has been modeled as a Censored Data type as described in the following subsection.

- The three possible outcomes for a company are disjoint events. Formally, these events represent the whole Ω events space.

- A company that at the end of the experiment observation period has not fallen on one of the three possible events is labelled as censored. This label does not contribute on the Ω space.

4.2.4.2 Ad-Hoc definitions

In the following, a certain number of ad-hoc definitions will be used, so a small recap of them is provided below. These definitions have been created in order to adapt the SM to the to the environment under study (the PE market).

- **Event of interest E.** For every company, the event of interest is the IPO event.

- **Survival.** According to the definition of E, the survival has been identified by the non-occurring of E, i.e. by the end of the study is still Private.

- **IPO Survival Time:** T_{IPO}**.** for each company, the event of interest is the IPO, so the survival time is the time from the foundation of the company to its IPO.

- ***e*1 event** the company went bankrupt or a Merge and Acquisition event has occurred. Details about acquisition and bankruptcy mechanisms has been discussed in Subsec.2.1.2. The presence of such companies within the subjects sets brought this study to establish a dataset conditioning as described in Sec.4.2.4.4.

4.2.4.3 Procedure

This procedure can be summarized as follows:

Observation period The observation period starts at the beginning of the 1998 and lasts until the end of 2018. In formulas, $t_{start} = 01/01/1998$, $t_{end} = 31/12/2018$. **Subjects** The companies with Foundation date in the Observation period. (Sec. 2). **Conditioning the dataset** as described in Subsec. 4.2.4.4. **Creation of the Survival Data matrix** The PE market from 1998 to 2018 has been observed. The bankruptcies and the acquisitions were excluded. For each company born in that period, the IPO date is recorded and its IPO time is computed as the time between the foundation of the company and the IPO itself. In formulas:

$$t_i^{survival} = (Foundation\ date) - (IPO\ date) \tag{4.4}$$

Where the *Foundation date* is the date on which the company has been founded. If a company went not on IPO, the its t_j lasted from its foundation up to the end of 2018 and it were marked as censored. More details on the theoretical rationale of censoring is described in subsec. 4.3.1. The **Subject Features** consisted on the investors ranking statistics described in sec. 2.

Kaplan-Meier (KM) method This model was applied in order to have an empirical estimate of the Survival statistics. As discussed in the following section, the KM method provides the *Survival Curves* (SC) with no need of subject features, since it relies on a simple counting statistics on the survival times of the subjects and their censoring labels.

Cox models The Cox method relies on a non-parametric model based on the Proportional Hazard Assumption (PHA) and they include the subject features. Before applying such models, the PHA has been verified (and indeed rejected) through dedicated statistical tests.

Accelerated Failure Time models On the other side, AFT models relies on the non PHA and they belong to the family of parametric models. AFT generates survival curves that depends on the subject features. Since the PHA has been rejected, AFT models has been chosen and their usage deeply exploited. In fact, these models are able to identify the *accelerating* and *decelerating* factors of the event of interest. No need to say, this particular feature is of great interest for an investor, since it highlights the features that statistically accelerates (or not) the time to IPO of a company.

4.2.4.4 Conditioning the data set

The first problem encountered within the survival part of this study has been the presence of Bankrupt and Acquired companies. That is: if SM are able to handle only binary outcome problems (censored/not censored data), how to manage multiple outcomes? That is, if the event of interest is the IPO, Private labels are the censored data, how to model a dataset that contains also Bankrupt and Acquisition ones?
The adopted solution is dataset conditioning: companies that went bankrupt or acquired have been excluded from the data-set and the survival models have been fitted with the remaining IPOs and privates. For each ith company, the distribution of T_i fitted in this way has been called *conditioned*, because it has been done on a conditioned dataset. Then, thanks to the Bayes Theorem, the distributions have been *unconditioned* using an independent probability estimate of e_1. In formulas, the conditioned survival estimates:

$$S_1(t) = \mathbb{P}(T > t|e_1) \tag{4.5}$$

A key assumption of this procedure is: $\{T \leq t \cap e_1\} = \emptyset$. From a practitioner (and indeed logical) point of view, it sounds fairly logical that if a company went bankrupt or acquired, it cannot go IPO. Nevertheless, a question may arise: are there public companies that go bankrupt? This is an event that could happen, however it is so rare than it was not worth to take account of.
So, by the rule of total probability, the following holds: $\mathbb{P}\{T > t\} = \mathbb{P}\{T > t|\overline{e_1}\}\mathbb{P}\{\overline{e_1}\}$
and hence,

$$\begin{aligned} F(t) = \mathbb{P}\{T \leq t\} &= 1 - \mathbb{P}\{T > t\} && (4.6)\\ &= 1 - \mathbb{P}\{T > t|\overline{e_1}\}\mathbb{P}\{\overline{e_1}\} && (4.7)\\ &= 1 - S_1(t)\mathbb{P}\{\overline{e_1}\}. && (4.8) \end{aligned}$$

Where $\mathbb{P}\{\overline{e_1}\}$ has been estimated, for each company, by the Random Forest algorithm described in Chapter 3.

4.3 Survival Curves & Kaplan-Meier method

The Kaplan-Meier method, also known as *Product Limit* (PL) estimate, is a non parametric method used to estimate the survival function $S(t)$ through $\hat{S}(t)$, regarding a set of subjects within a survival analysis [55]. This method allows to include the censored values within the estimate. It is classified as *non parametric* since, within the estimation, so it does not involve the Subject Features described in Chapt. 2.

4.3.1 Censoring type used

First of all, the type of censoring listed in Subsec.4.2.2. must be identified.
In this study, each company enters in the population at random times, i.e. when they are founded. So, the Type I censoring should be excluded. Moreover, the study ended even if some companies (indeed, the majority of them) have still not experienced any event, nor bankruptcy/acquisition or IPO.
The natural choice is then a Type III censoring: for each subject, the time of entry in the study is random and the study ends up leaving a subset of the population censored, since they remain private companies. The rational behind the Private Equity application of this censoring is the following: when the study ends up in 2018, the companies that are still private *could* go on IPO in a certain point in the future, but it is not known when and if this will happen. So, their survival times are censored according to Type III.

In a nutshell, if a company had not performed an IPO, its survival time lasted from its foundation up to t_{end} (2018) and it had been marked as censored. If a company performed an IPO at t_{IPO}, its survival time lasted from its foundation up to t_{IPO} and it had been marked as non-censored.

4.3.2 KM Theory

The KM method relies on the Product Limit estimate, that is indeed an interpretation of the survival probabilities within a population. Let's dive into it with a simple example [56].
Consider a clinical study starting at the beginning of 2017 that ends up at the end of 2018. 10 patients (g_1) enters the study and at the end of the same year 4 out of them are still alive. At the beginning of 2018, 20 new patients (g_2) enter the study. By the end of the study, just 1 patient belonging to g_1 is still alive and 5 out of the initial 20 belonging to g_2 are still alive. Sure enough, a smart method to estimate $S(t)$ would be the one that encodes within $\hat{S}(t)$ the knowledge regarding the different entry date of the subjects of g_1 and g_2.
Indeed, the KM estimate supposes that the survival probability $S(2)$ is composed by the probability of surviving for the first and then for the second year given the survival in the first one. In formulas:

$$S(2) = \mathbb{P}(T \geq 2|T > 1)\mathbb{P}(T > 1) \tag{4.9}$$

Whose KM estimation is:

$$\hat{S}(2) = (\hat{p_{2|1}})(\hat{p_1}) = \frac{1}{4} \cdot \frac{4+5}{10+20} \tag{4.10}$$

where $\hat{p_{2|1}}$ is the proportion of patients that survived two year among the ones that survived the first year, while $\hat{p_1}$ is the proportion of patients surviving the first year. It is now straightforward to generalize such computation.

Product Limit general estimate

In general, according to the KM Product Limit survival estimate, Eq.4.10 can be generalized as:

$$\hat{S}(k) = p_1 \cdot p_2 \cdot \ldots p_k \tag{4.11}$$

where p_1 is the proportion of subjects that survived at least until the end of period 1, p_2 is the proportion of them that survived between period 1 and 2 and so on. p_k regards the subjects that survived until the $(k-1)$th year up to the kth one.
From Eq.4.11 it follows that:

$$\hat{S}(t) = \hat{S}(t-1)p_t \tag{4.12}$$

A question arises naturally: **how the PL estimate can include the censoring?** The procedure to include such data derives straightly from the non-censored version. Let n be the total number of subjects, with uncensored or censored survival times t_i. These survival times can be ordered such that $t_1 \leq t_2 \leq \cdots \leq t_n$. Let r be the ordering index of the uncensored survival times such that $t_r \leq t$. Please note that if there are no censored observations, then the values of r are consecutive integers up to n. Otherwise in the censored framework, r stops up to an integer $m \leq n$.
Finally the KM survival estimate is:

$$\hat{S}(t) = \prod_{t_r} \frac{n-r}{n-r+1} \tag{4.13}$$

The PL estimates are maximum likelihood estimates.
The variance of this estimates is:

$$Var[\hat{S}(t)] \simeq [\hat{S}(t)]^2 \sum_r \frac{1}{(n-r)(n-r+1)} \tag{4.14}$$

from which it is possible to derive confidence intervals for $\hat{S}(t)$ [55].
A mock scenario should clarify the computation procedure.

Example of a KM estimate computation Let's suppose that the Private Equity market of DuckLand has been observed for 10 years. During the observation time, 10 companies have been founded. Each of them have been observed from its foundation up to t_{end} (the end of the 10 years observation period). 6 of them went IPO (the PE market in DuckLand is very rewarding). Their IPO dates were recorded and their IPO time were computed as the times between the foundation of a company and the IPO itself. If a company had not performed an IPO, its survival time lasted from its foundation up to t_{end} and it have been marked as censored. The others were still private up to t_{end}. Please recall the nomenclature of Subsec.4.2.4: the IPO event corresponds to *survival*.
The following table can then be built:

Surv. time (y.)	Rank	Censoring Label	r	(n-r)/(n-r+1)	$\hat{S}(t)$
1.4	1	1	1	$\frac{9}{10}$	0.900
2.3	2	0	-	-	-
3.4	3	1	3	$\frac{7}{8}$	$\frac{9}{10} \cdot \frac{7}{8} = 0.786$
3.5	4	0	-	-	-
5.8	5	1	5	$\frac{5}{6}$	$\frac{9}{10} \cdot \frac{7}{8} \cdot \frac{5}{6} = 0.656$
6.2	6	1	6	$\frac{4}{5}$	$\frac{9}{10} \cdot \frac{7}{8} \cdot \frac{5}{6} \frac{4}{5} = 0.525$
7.8	7	0	-	-	
7.9	8	0	-	-	
8.5	9	1	9	$\frac{1}{2}$	$\frac{9}{10} \cdot \frac{7}{8} \cdot \frac{5}{6} \cdot \frac{4}{5} \cdot \frac{1}{2} = 0.266$
9.9	10	1	10	0	0

Table 4.1: Example of Kaplan Meier estimate from a sample population.

where the censoring label is 0 when the survival time is censored (no IPO) and 1 if not (IPO).

4.3.3 Results: KM survival curves

Finally, the Kaplan-Meier estimated survival functions (*survival curves*) are computed for each industrial sector of the reference dataset. Please recall that within this fitting, the subject features of each company has not been considered, since the $\hat{S}(t)$ estimate of $S(t)$ needs only the survival times and the censoring label. These were both compute as described previously.

From the coding point of view, the `survfit` function within the `R` package `survival` has been used to perform the KM estimates.

The conditioned dataset curves (i.e. the bankruptcies and acquisitions free ones) are shown before the unconditioned plots. The all sectors curves are compared side to side in order to highlight the effect of conditioning on the survival patterns. Please refer to Subsec .4.2.4.4 for the details about conditioning.

Observations

First of all, it is interesting to note the different shapes of the curves across the sectors. In particular, sectors 1 and 2 curves are approximately bell-shaped, while sectors 6 and 7 ones show an exponential pattern.
This differences are quite interesting, since they highlight a difference across industrial sectors in the time-to-IPO probability pattern, which was by no means a foregone conclusion. As an example, sector 1 (Communications) companies tend to go IPO "faster" after a certain period from their foundation (see the sharp decreasing of the red curve in Fig.s 4.2, 4.3) w.r.t. the first years after their foundation. On the other hand, sector 6 (Energy) companies have on average constant "IPO-rate", since their KM survival curve decreases with a quite constant slope.

As for the conditioned/unconditioned differences, it is possible to note how the

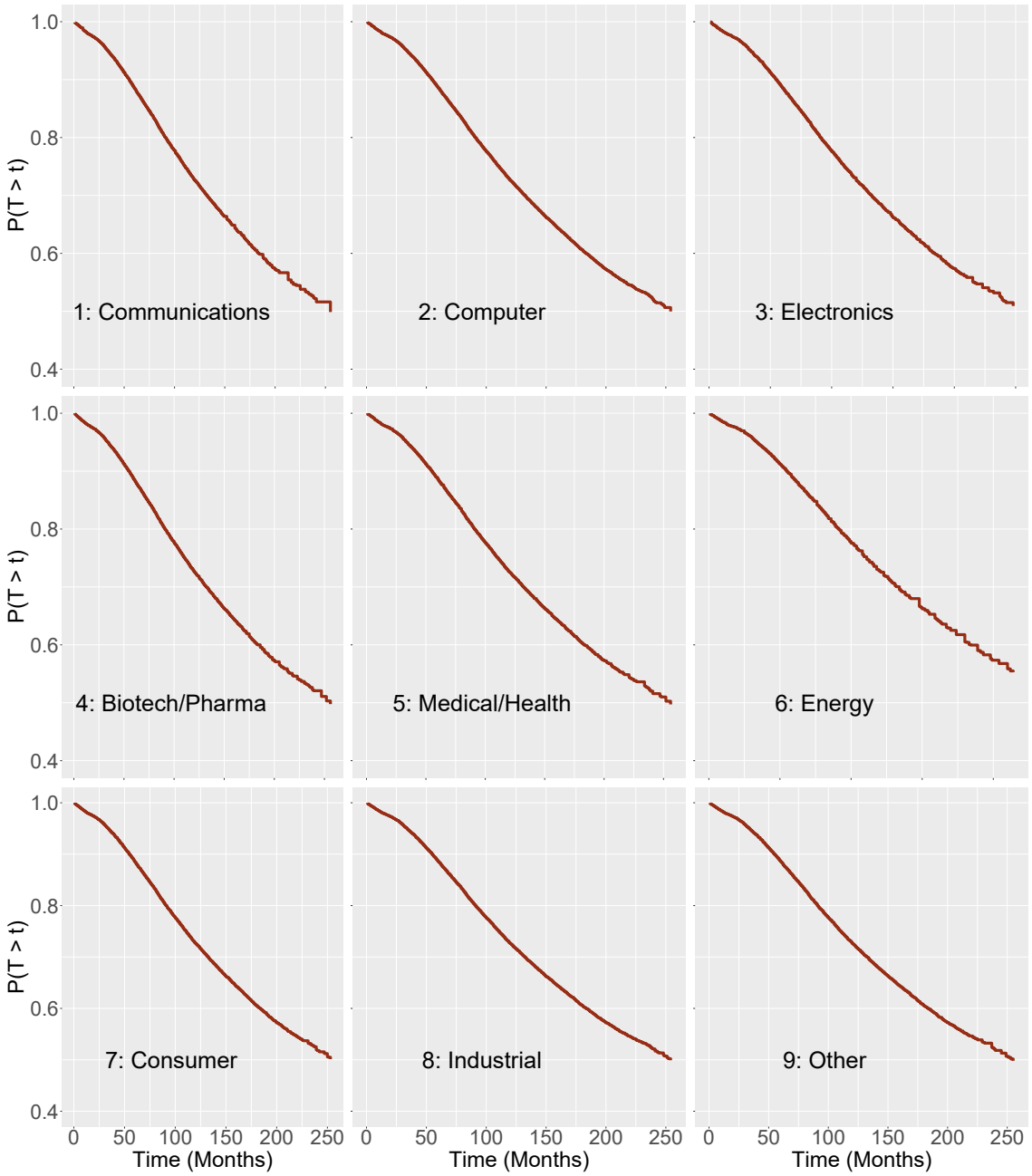

Figure 4.1: Conditioned Kaplan Meier survival curves for the 9 industrial sectors considered.

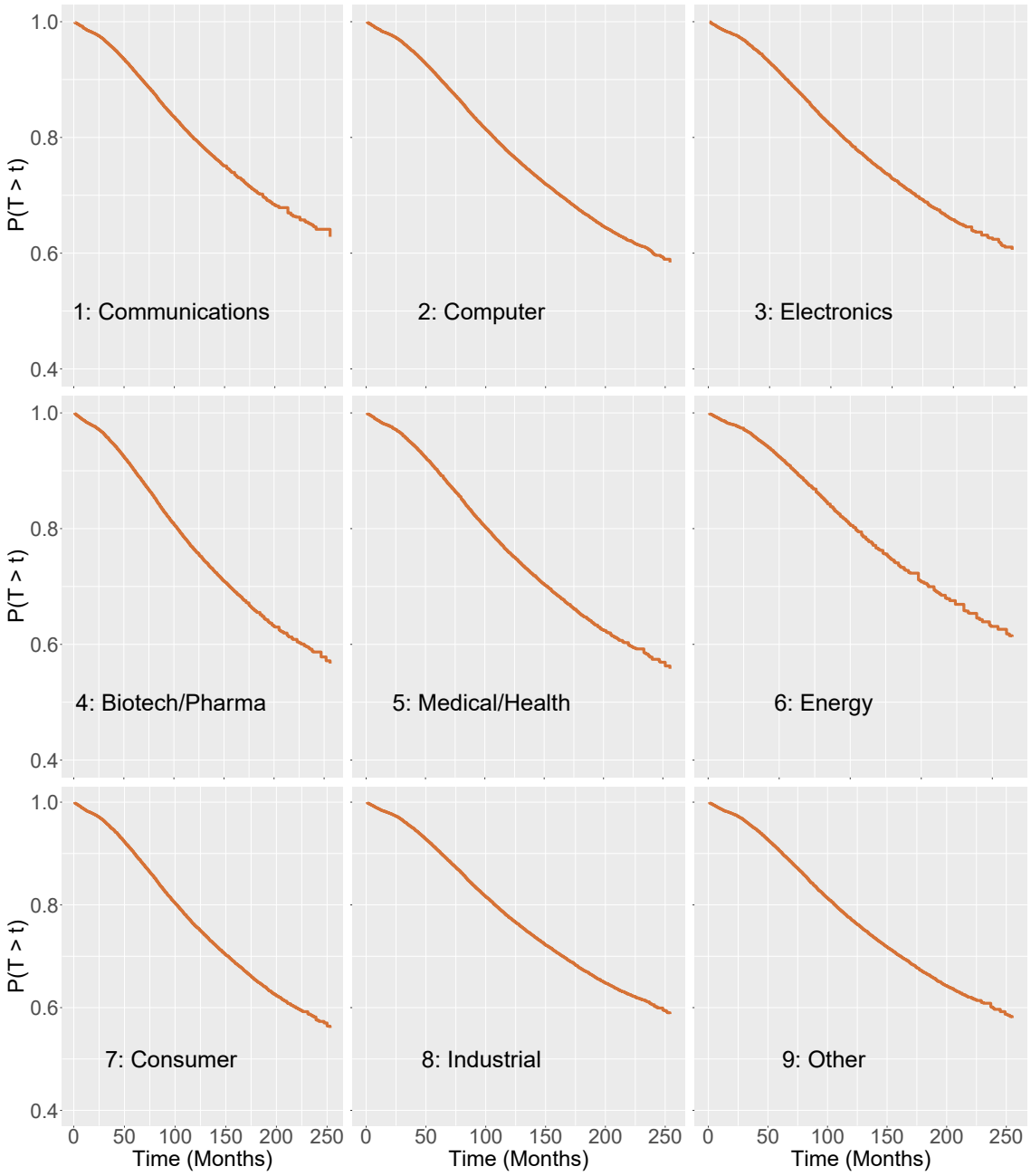

Figure 4.2: Unconditioned Kaplan Meier survival curves for the 9 industrial sectors considered.

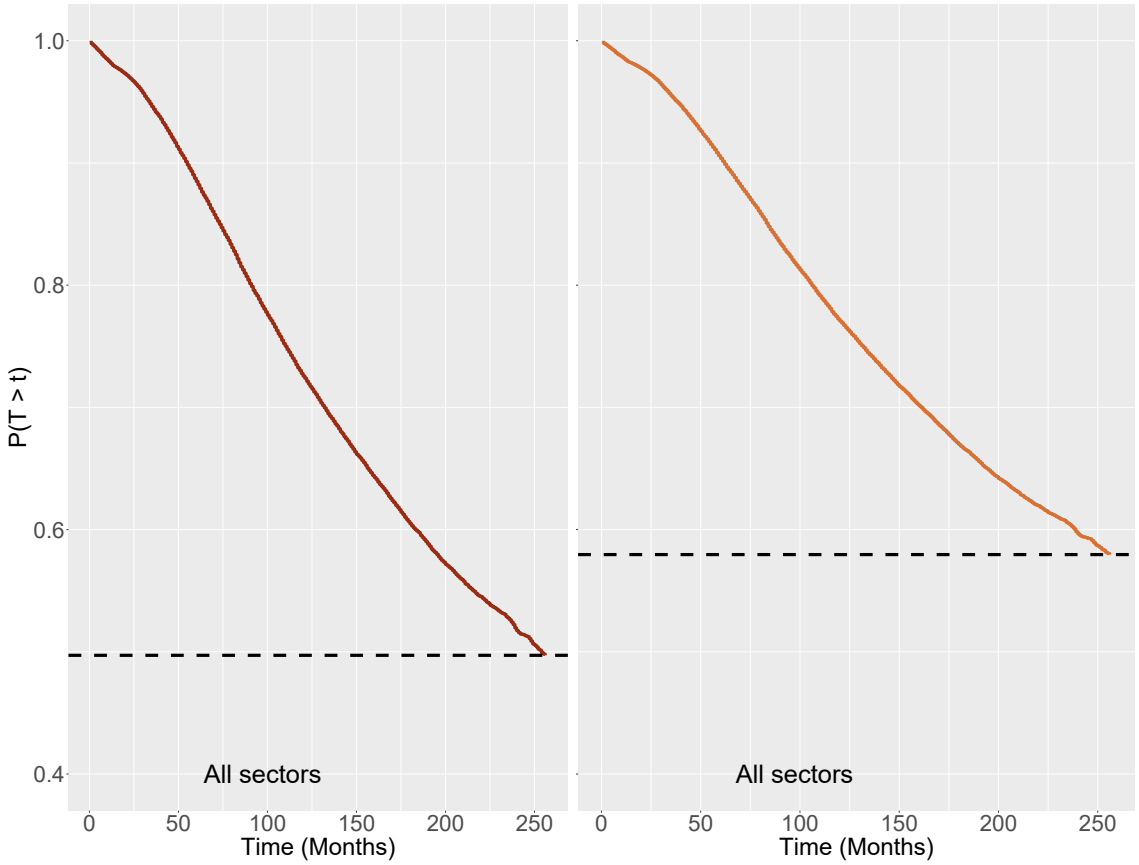

Figure 4.3: Survival Curves computed across all sectors. Dashed line: survival curve minimum point. Left plot: conditioned dataset. Right plot: unconditioned dataset. The y axis is the Kaplan Meier survival probability estimate, the x axis represent the time expressed in months. It is worth to note the remarkable different in the curve minimum point caused by the unconditioning procedure.

latter curves are "pushed" to the top of the graph w.r.t. to the conditioned ones. This is well pointed out by the red dashed line in Fig. 4.3. The effect is due to the inclusion of the Bankrupt and the Acquisition labels, that lowers the probability of going IPO of the whole population through time.

4.4 Cox Proportional Hazard Models

Cox models are non-parametric methods that do not require the knowledge of the underlying distribution that fits the survival curve built upon some Survival Data [57]. Despite this seems to be a *comfortable* framework, Cox models rely on a strong assumption: the hazard functions of different subjects are proportional and independent of time [58].

4.4.1 Proportional Hazard Assumption

The main assumption of this model is that the ratio of the hazard functions of 2 subjects drawn from the population is a constant not depending on time. This have a strong impact of the interpretation of the survival functions fitting.
In fact, within this study, the PH assumption would imply that the ratio of the risk of going IPO of two companies would depend just by the investors who invested in them (according to the Subject Features design). The time would not induce any increasing nor decreasing in the ratio of risk of the two companies.
Indeed, some probability distributions show this property, that is, in formulas:

$$h(t|x_1, \ldots, x_p) = h_0(t)g(x_1, \ldots, x_p) \tag{4.15}$$

where $x_1, \ldots, x_p$ are the Subject Features, $h_0(t)$ the *baseline hazard function* and $g(x_1, \ldots, x_p)$ a deterministic function of the subject features. It is straightforward how such an hazard function formulation induces the proportional hazard property. Let's consider the hazard function ratio of two companies:

$$\frac{h(t|\mathbf{x_1})}{h(t|\mathbf{x_2})} = \frac{h_0(t)g(\mathbf{x_1})}{h_0(t)g(\mathbf{x_2})} = \frac{g(\mathbf{x_1})}{g(\mathbf{x_2})} \tag{4.16}$$

where the x_1, x_2 vectors are the subject features of the two sample companies. Please note that the last term of Eq.4.16 is independent of time, as the PH assumption states.

4.4.2 Cox Model Theory

From the formal point of view, the Cox PH model relies on a key assumption about the survival functions structures.
The hazard function is:

$$h(t|\mathbf{x}) = h_0(t)g(\mathbf{x}) = h_0(t)\exp\{\mathbf{b}'\mathbf{x}\} \tag{4.17}$$

where $\mathbf{b}$ represents the vector of fitted covariates coefficients.
From this, it can be shown that the survival function is defined as:

$$S(t|\mathbf{x}) = [S_0(t)]^{\exp\{\mathbf{b}'\mathbf{x}\}} \tag{4.18}$$

where $S_0(t)$ represents the *baseline survival function.* Within the Cox models, this function is estimated in a non parametric fashion. Breslow [59] designed it in order

to include covariates:

$$\hat{S}(t|\mathbf{x}) = \prod_{t_{(i)} \leq t} \exp\left[\frac{m_{(i)}}{\sum_{l \in R(t_{(i)})} \exp\left(\mathbf{x}_l' \hat{\mathbf{b}}\right)}\right] \tag{4.19}$$

With some assumptions, $\hat{S}(t|\mathbf{x})$ can be proven to follow a normal distribution, thus confidence intervals can be built. However, as anticipated in Chapt. 1, since the Cox models will be discarded, these technicalities are out of scope.

In order to estimate the coefficients vector $\mathbf{b}$, Cox developed a *partial likelihood function*, assuming the absence of tied survival times [57]. Fortunately, further works [60],[59],[61] developed the Cox theory towards the inclusion of tied survival times. Even if the reference Survival Data show tied times, the gathering of these data was performed on a *discrete* time scale. Again, Cox [57] proposed an alternative partial likelihood function with tied discrete survival times.

4.4.2.1 Partial Likelihood Function

First of all, let's clarify the meaning of *partial* within this quantity. This is because the PLF is based on a conditional probability of failure that is built on a counting computation of subjects still alive in a point in time *conditional* to their past.
The PLF should be now described in a more formal way. The inclusion of tied survival time needs the introduction of more nomenclature [62]. Let

- t_i be the survival times of the k subjects of the study.
- $R(t_i)$ be number of subjects still alive at time t_i
- m_i be the number of tied observations at time t_i
- $\mathbf{u_{(j)}}$ be the representation of all the possible $m_{(i)}$ random selections on $R(t_i)$. U_i will contain all the $\mathbf{u_{(j)}}$s
- $\mathbf{u*_{(i)}}$ the set of subjects that *failed* exactly at time t_i.
- $\mathbf{z_{u*_{(i)}}} = (z*_{1\mathbf{u*_{(i)}}}, \ldots, z*_{p\mathbf{u*_{(i)}}})$ where $\mathbf{z*_{lu*_{(i)}}}$ represents the sum on the covariate l of all the $m_{(i)}$ subjects within $\mathbf{u*_{(i)}}$. $\mathbf{z_{u_{(i)}}}$ means the same, but associated to $\mathbf{u_{(j)}}$.

So, the PLF reads as:

$$L_d(\mathbf{b}) = \prod_{i=1}^{k} \frac{\exp\left(\mathbf{z}'_{\mathbf{u*_{(i)}}}\mathbf{b}\right)}{\sum_{u_{(j)} \in U_i} \exp\left(\mathbf{z}'_{\mathbf{u_{(j)}}}\mathbf{b}\right)} \tag{4.20}$$

The vector $\hat{\mathbf{b}}$ that maximizes $L(\mathbf{b})$ represents the maximum partial likelihood estimator of $\mathbf{b}$, i.e. the covariates coefficients to be inserted in Eq.4.18. *If* the Cox models had been applied, the mechanics related to the the search of $\hat{\mathbf{b}}$ would have deserved a deeper analysis. However, in order to apply such models, the PH hypothesis has to be verified: it would be useless to dive in technicalities before. Anyway, in the following subsection it will be clear that this hypothesis is not statistically verified.

4.4.3 Proportional Hazard Hypothesis Testing

To check if the PH Hypothesis holds, two kinds of validation procedures have been performed:

- The more *statistical* one, that is the *Schoenfeld residuals test* [63].
- The more *graphical* one, that is the *Cox-Snell* plot [64] to compare the Cox models with their alternative, the AFT ones, for which the PH assumption is not necessary, as will be clear in Sec. 4.5.

The second method relies on testing the survival data fitting with a *non-proportional-hazard-distribution*. In fact, for some probability distributions (such as the Weibull), Eq.4.16 is verified, while for others (such as the LogNormal), it is not. If the fitting for the latter group of distributions is robust, the PH assumption is rejected and the Cox models excluded from the analysis. Since this checking procedure is deeply related to the AFT models, it will be discussed within that section.

Schoenfeld Residuals (SR) test

The PH assumption requires the covariates not to be time-dependent. If just one subject feature varies with time, this assumption is false [65]. So, if the residuals of a fitted Cox model are scaled with time, if the PH assumption holds, no pattern should be recognised. This must hold for every covariate.

Within the survival analysis framework, the notion of residuals have to be modified w.r.t. the classical regression one. As for the Schoenfeld residuals, for each survival time t_i, for each datum referring to the SD matrix built by a number p of j covariates and a number n of i individuals:

$$R_{ji} = \delta_i \left[[x_{ji} - \frac{\sum_{l \in R(t_{(i)}} x_{jl} \exp(\hat{\mathbf{b}}' \mathbf{x_l})}{\sum_{l \in R(t_{(i)}} \exp(\hat{\mathbf{b}}' \mathbf{x_l})} \right] \tag{4.21}$$

where δ_i selects only the uncensored observations, since the Schoenfeld residuals are null for the censored survival times. As stated in [63], since $\hat{\mathbf{b}}$ is a maximum likelihood estimator, for each covariate the sum of the SR is null. Moreover, their asymptotic mean is 0 and they are not correlated.
In light of these properties, it is possible to build a SR statistical test for each covariate j (as proposed by Grambsch [66]):

$$H_0 : \textit{ the correlation between time and residuals is null} \tag{4.22}$$

Since the Cox models rely on the time-independence hypothesis, rejecting 4.22 for just one of the subject features equals to reject the possibility of using Cox models themselves.

Before presenting the results, a thoughtful reminder is due. In order to perform

the SR test, the Cox models had to be fitted within the reference Survival Data. So, the maximization of the PLF 4.20 had to be performed as well as the study of the behaviour of $\hat{\mathbf{b}}$. However, since the check of the PH assumption rejected the usage of Cox Models, for sake of simplicity these technicalities are not included in the main text of this work.

SR test results The results below are related to the Cox fittings on the datasets of industrial sectors 1, 2, 7 and on the all-sectors dataset. The code selected for the SR test only the statistically significant covariates according to the test related to the single covariate coefficient.

Covariate	p-val sec.1
max1	7.47e-01
min1	4.88e-01
num1	5.31e-02
mean2	9.16e-01
max2	7.68e-01
min2	9.88e-01
num2	1.81e-01
GLOBAL	9.56e-08

Covariate	p-val sec.2
mean1	3.85e-10
min1	1.24e-04
num1	1.51e-04
mean2	6.19e-01
min2	9.94e-01
num2	1.37e-01
min3	5.35e-14
num3	9.03e-03
GLOBAL	1.81e-74

Table 4.2: Sectors 1-2 SR test

Covariate	p-val sec.7
mean1	1.11e-04
num1	5.84e-02
GLOBAL	3.27e-04

Covariate	p-val all sec.
num1	1.19e-01
mean2	4.03e-08
min2	2.21e-09
num2	6.42e-08
mean3	6.22e-01
min3	7.11e-02
num3	5.01e-02
GLOBAL	5.91e-137

Table 4.3: Sectors 7 - All sectors SR test

In 3 out of the 4 sampled sectors, there are at least 2 covariates whose null hypothesis has been rejected with an $\alpha = 5\%$. In any case, in each of the industrial sectors studied (from 1 to 9), the *GLOBAL* p-value is far below the level of significance.
So, **the SR test rejected the Proportional Hazard hypothesis**.

4.5 AFT models

As investigated in the previous section, the Cox models non parametric approach cannot be used since the PH assumption does not hold. An alternative model, which does not require such hypothesis is the Accelerated Failure Time (AFT) model.

The Accelerated Failure Time models have been the key point of the survival part of this study. Since their base assumptions have been empirically proved and they showed statistically significant results, these models capabilities (such as the *accelerating factors*) have been studied and exploited. In the end of this section a brief presentation of the final appreciative practical usage of AFT models is showed.
This part of the survival analysis have been guided by the following procedure:

1. **Parametric Survival Fitting** The base of the AFT models relies on the fitting of the survival function $S(t)$ by means of parametric probability distributions (weibull, lognormal etc.). This was the initial brick of the AFT building. In order to develop a first insight on what parametric distribution fitted the Survival Data of each industrial sector, two validating approaches have been performed: Cox-Snell residuals and the Wald's statistic test (subsec.s 4.5.1.2, 4.5.2.1).

2. **AFT models fitting** AFT models were fitted and the subject features selected according to their statistical significance. The best performing probability distributions were selected according to numerical and graphical metrics.

3. **Validation plots** The consistence of the choice of the selected distributions has been validated according to a graphical metric.

4. **Accelerating Factors Analysis** A key feature of AFT models consists in highlighting the accelerating (or decelerating) factors affecting $S(t)$. This feature has been studied and exploited.

4.5.1 Parametric Survival Fitting

First of all, before adding the Subject Features to the analysis, it appeared natural to fit the $S(t)$ given by the survival times of Survival Data with a parametric distribution without covariates.
This is performed through a Maximum Likelihood Estimation, refined in order to be able to handle the censored observation. Given a parametric distribution, the MLE boils down to find a vector of distribution parameters $\hat{\mathbf{b}}$ that maximizes the probability of observing the survival times given that the distribution is shaped by $\mathbf{b}$
More formally, if there are right-censored survival times among data, the likelihood function is defined as:

$$L(\mathbf{b}) = \prod_{i=1}^{r} f(t_i, \mathbf{b}) \prod_{i=r+1}^{n} S(t_i^+, \mathbf{b}) \tag{4.23}$$

where **b** is the distribution parameters vector, r is the number of non-censored observations, n the total number of observations and t_i^+ the values of censored survival times.

4.5.1.1 Distributions used & Results

As stated above, the SD has been firstly fitted without including the Subject Features. Three distributions were included:

- **Exponential, Weibull** in order to try a *simple* and parameters-saving distribution
- **Lognormal** in order to catch a more sophisticated trend that the Kaplan-Meier curves (Fig.4.2) in some sectors
- **Generalized F** [67] The more complicated and parameters-rich distribution. This is the less interpretative distribution and indeed the most difficult to fit in terms of optimization of 4.23. However, since it depends on 4 parameters, the GenF distribution is also the most flexible one [68], [69].

Please note that these distributions represents, respectively, 3 different *families*: exponential, normal and Snedecor's F. This choice was performed in order to try to catch different types of patterns of each sector empirical survival distribution.

Generalized F focus With respect to the others used distributions, the Generalized F is an extremely complicated and very low-documented distribution, so its features will be studied separately. Its probability distribution reads as:

$$f(w) = \frac{\delta((\frac{s_1}{s_2})^{s_1} e^{\delta s_1 w})}{\sigma t(1 + s_1 e^{\delta \frac{w}{s_2}})^{s1+s_2}} \tag{4.24}$$

where:

- $w = \frac{(log(t)-\mu)\delta}{\sigma}$
- $\delta = \sqrt{Q^2 + 2P}$, $s_1 = 2(Q^2 + 2P + Q\sqrt{Q^2 + 2P})^{-1}$
- $s_2 = 2(Q^2 + 2P - Q\sqrt{Q^2 + 2P})^{-1}$

Roughly speaking, the GenF is basically a *scaled log-F*, where F is the common *Snedecor's F* distribution. More details can be found in [69].

Finally, an example: the parametric fittings of sector 1 (Communications) (Fig.4.4). Please note that the curves presented refer to the conditional set (cleaned from the bankruptcies and acquisitions as explained in Subsec. 4.2.4.4).

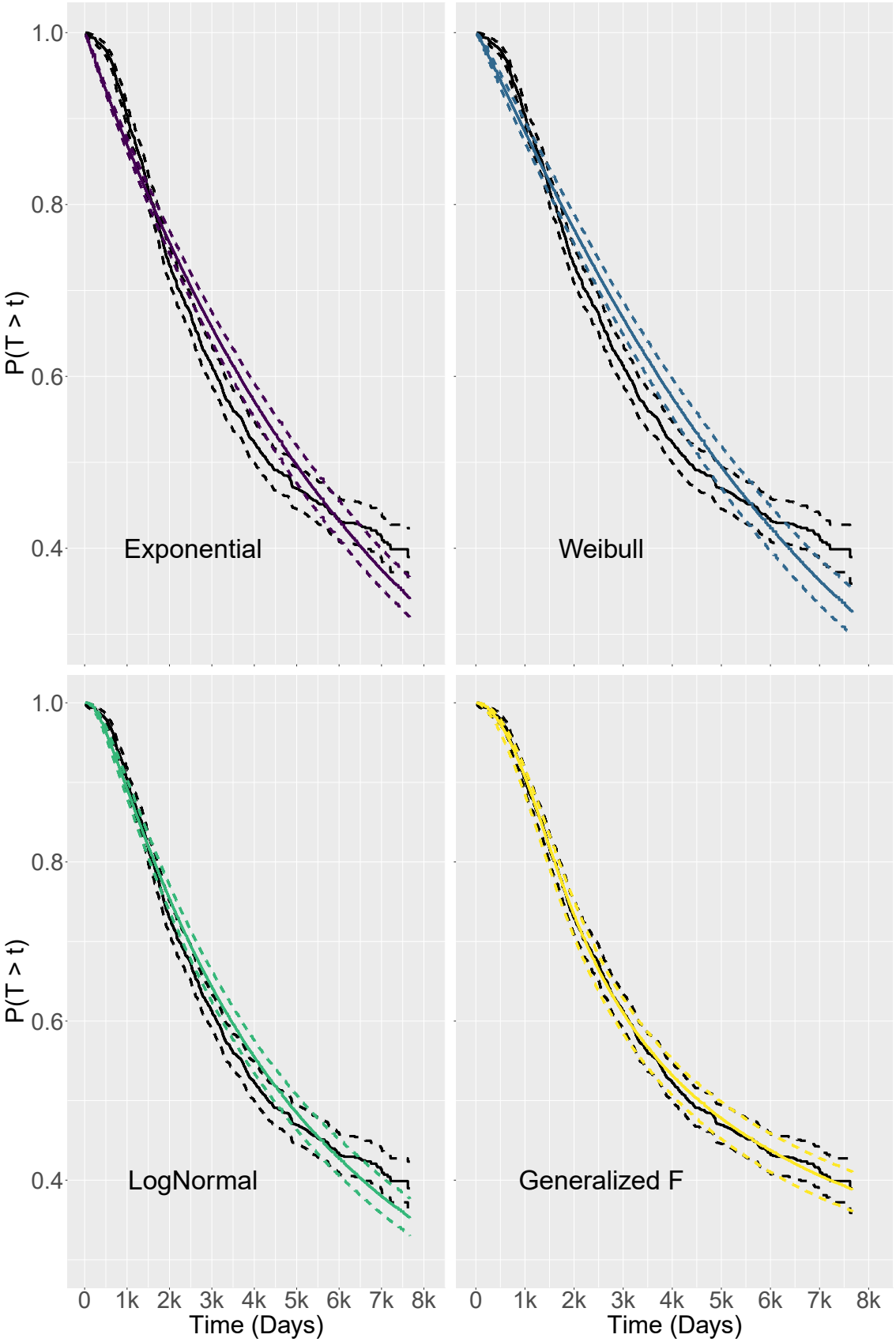

Figure 4.4: Parametric Survival Curves for sectors 1 (Communications). The black solid lines represent the Kaplan-Meier survival distribution estimate. The dashed lines are the confidence intervals. The y axis is the survival probability estimate, $\hat{S}(t)$.

4.5.1.2 Cox-Snell residuals

In order to validate the choice of the distributions that better fitted the SD of each sector, a Cox-Snell (C-S) residual method [64] is first applied.

The C-S method relies on a graphical approach. For each point of the fitted survival curve, the C-S residual is defined by:

$$r_i = -log\hat{S}(t_i),\ i = 1, 2, \ldots, n \tag{4.25}$$

where $\hat{S}(t_i)$ is the MLE parametric estimate of the survival distribution related to a specific distribution, such as the exponential. Since the cumulative hazard follows the property $H(t) = -\log S(t)$, therefore the quantity r_i is an estimate of the cumulative hazard curve at t_i. If t_i is censored, so it is its r_i. The C-S procedure is based on one important property: if the selected model fits the SD, then the C-S residuals r_i follow the unit exponential distribution with probability density $f_R(r) = \exp^{-r}$. So, if $S_R(r)$ denotes the survival distribution of the C-S residual r_i, then:

$$-\log S_R(r) = \int_r^\infty e^{-x}dx = -\log(e^{-r}) = r \tag{4.26}$$

In order to estimate $S_R(r)$, the K-M estimate $\hat{S}_R(r)$ discussed in the dedicated section is used. From this formulation, it is clear that, if the model fits the data, the plot of r_i vs. $\hat{S}_R(r)$ should approximate the unity slope line.
The results related to 4 sample sectors 1 (Communications), 3 (Electronics), 6 (Energy) and 7 (Consumer) are presented in Fig.s 4.5, 4.6, 4.7, 4.8. These 4 sectors have been chosen because they show different empirical (KM) patterns in the survival curve, so it is possible to appreciate the different fitting capability of the 4 different distributions.

Rejection of the PH hypothesis As stated in Subsec.4.4.1, a strong evidence against the Proportional Hazard assumption consists in the fitting of at least 1 parametric, non PH distribution. The Generalized F and the LogNormal ones are non PH distributions, so if the SD are *well* fitted by these distributions, then there is another proof favourable to the PH assumption rejecting. As the Cox-Snell residual plots clearly testify, the non PH distributions fit very well the empirical distribution in each sector (Fig.s 4.5, 4.6, 4.7, 4.8) so the PH assumption is again rejected. This situation is confirmed also within the CS plots industrial sector not showed.

Please note that the methods used to check the PH assumptions in this analysis are *industrial sector-based*: SR residuals and CS plots are computed sector by sector.

4.5.2 AFT theory

If covariates are included to the parametric models, Accelerated Failure Time are then considered. Within these models, the Subject Features play the role of the

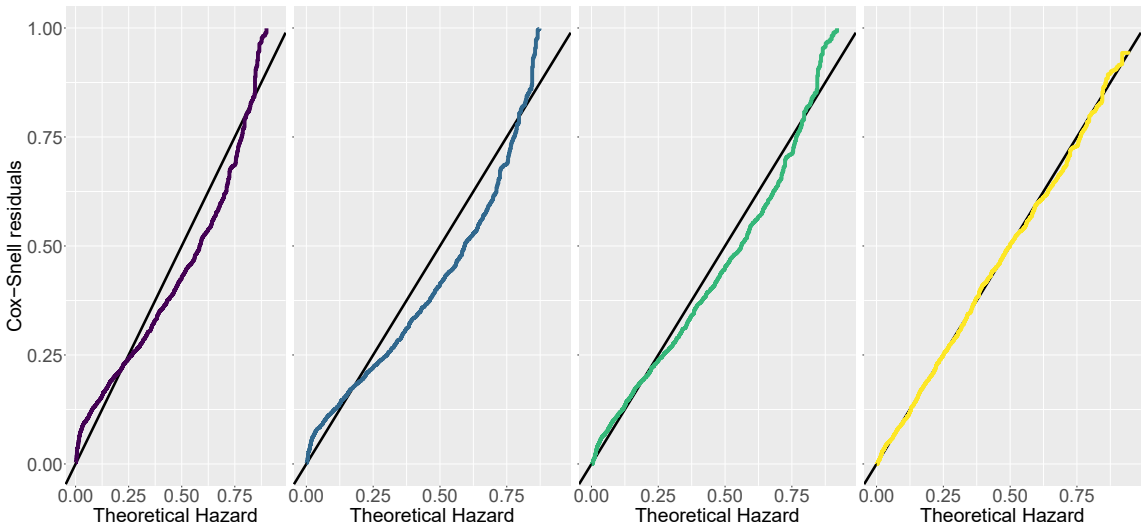

Figure 4.5: Cox-Snell residuals plot for sector 1 (Communications). From left to right: exponential, Weibull, LogNormal and Generalized F distribution.

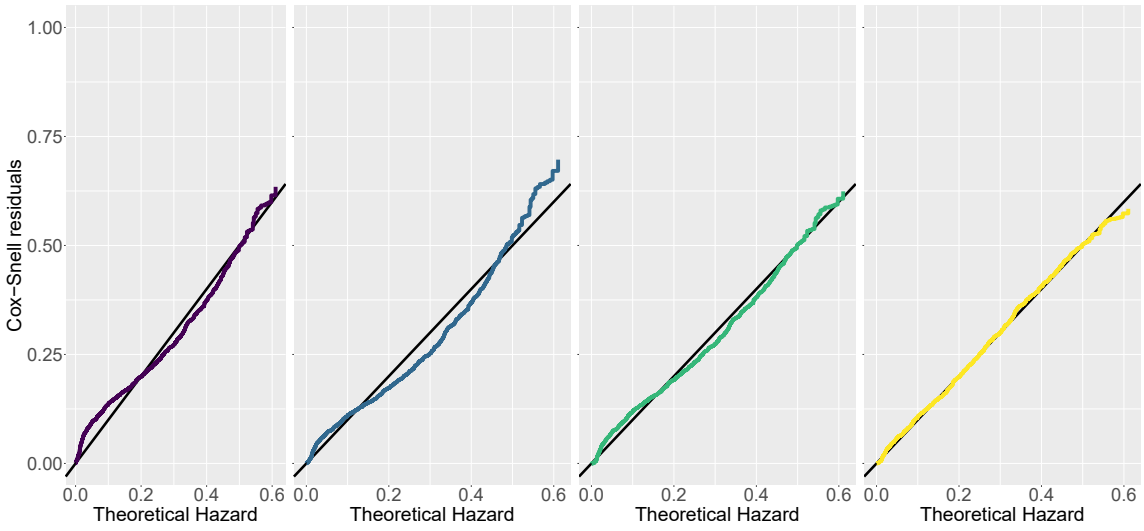

Figure 4.6: Cox-Snell residuals plot for sector 3 (Electronics). From left to right: Exponential, Weibull, LogNormal and Generalized F distribution.

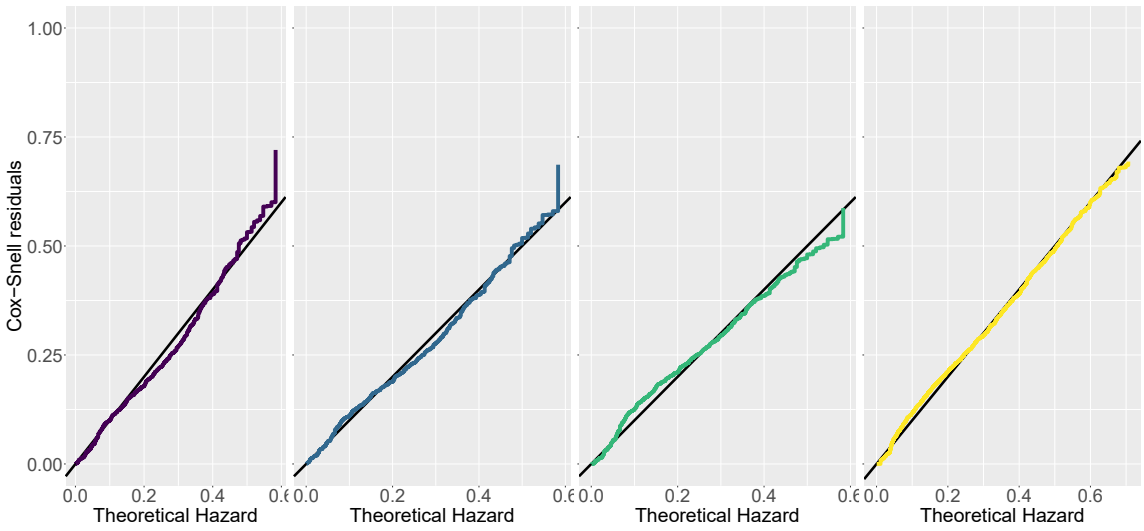

Figure 4.7: Cox-Snell residuals plot for sector 6 (Energy). From left to right: Exponential, Weibull, LogNormal and Generalized F distribution.

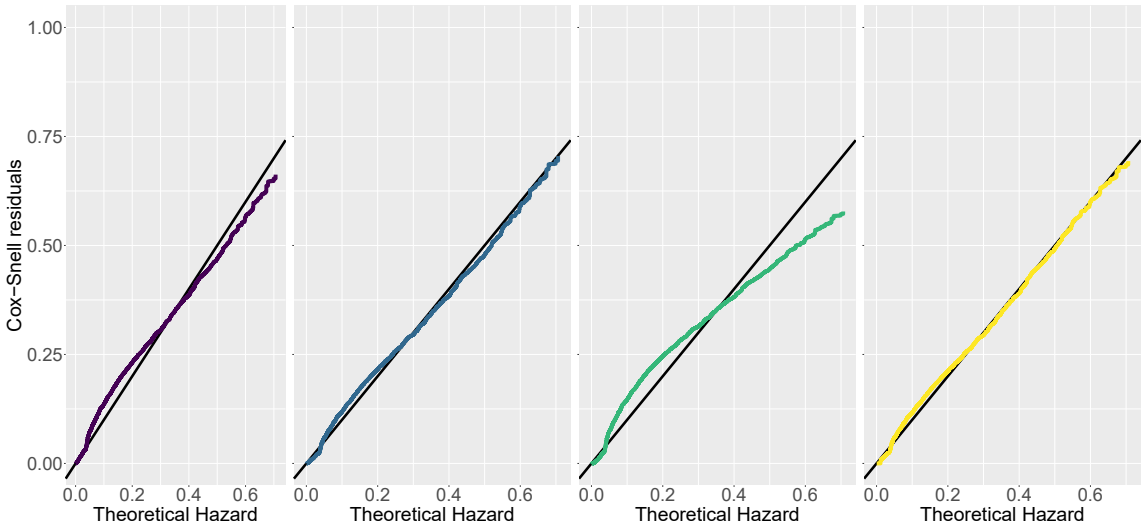

Figure 4.8: Cox-Snell residuals plot for sector 7 (Consumer). From left to right: Exponential, Weibull, LogNormal and Generalized F distribution.

prognostic variables or *prognostic factors*. Keeping the medical-related nomenclature, SF can have a theoretical, parametric relationship with the Survival Time probability distribution. So, SF can be used to perform a *prognosis* based on a probabilistic model.
Within the Private Equity context, this is equal to state that the set of investors that invested in a company are directly, probabilistically related to the time-to-IPO of the companies itself.

In formulas, the AFT model assumes that the logarithm of the survival time T is related to the covariates in a log-linear fashion [70]:

$$\log T = a_0 + \sum_{j=1}^{p} a_j x_j + \sigma\epsilon = \mu + \sigma\epsilon \tag{4.27}$$

where:

- x_j, a_j are respectively the p covariates and the p regression coefficients. a_0 is just an intercept parameter. All of these quantities can be summarized with μ. Please note that it is a deterministic term.

- σ is an unknown scale parameter

- ϵ is a stochastic error term, equipped with a density function $g(\epsilon, \mathbf{d})$ with unknown parameters vector $\mathbf{d}$. It is the actual *source of randomness* of the model. In the following subsections it will be clear how the distribution of ϵ induces the one of T.

4.5.2.1 Accelerating or Decelerating Factors

Let's suppose that an AFT model includes just the Subject Feature x, and that it can attain just discrete values, $x = 0$ or $x = 1$. Then, according to Eq.4.27, $T_{x=0} = \exp(a_0 + \sigma\epsilon)$ and $T_{x=1} = \exp(a_0 + a_1 + \sigma\epsilon) = T_{x=0}\exp(a_1)$, resulting in $T_{x=1} > T_{x=0}$ if $a_1 > 0$, and vice-versa.
From this basic calculus, the nomenclature of these models is straightforward: if a covariate takes on a positive value, it *decelerates* the survival time, while if it is negative the effect on T is an *acceleration* one.

The choice of the AFT models as time-to-IPO probabilistic models, indeed, was driven also by this practical need. In fact, this particular feature highlighted the role that the investors in each investment round had in accelerating or decelerating the time-to-IPO on a company, even if this acceleration caused no IPO within the 20 years length study.
However, it is worth to note that the robustness of this accelerating/decelerating interpretation relies on the statistical strength of the fitting of the AFT model used. This can be inferred as described in the next subsection.

Likelihood & Inference

In order to find an estimate of the regression parameters, equipped with confidence intervals, a likelihood formulation has to be stated as well as some inference on those parameters.

Likelihood formulation First of all, let's derive the likelihood function that has to be maximized in order to find $\mathbf{a}$ and $\mathbf{d}$. Let $\mathbf{b} = (\mathbf{a}, \mathbf{d}, a_o, \sigma)$ include all the unknown terms. Then, the log-likelihood w.r.t. the density function $f(t, \mathbf{b})$ and the survival distribution $S(t, \mathbf{b})$, similarly to Eq.4.23, is:

$$\begin{aligned} l(\mathbf{b}) = \log L(\mathbf{b}) = \sum \log(f(t_i, \mathbf{b})) + \sum \log(S(t_i, \mathbf{b})) + \\ \sum \log(\ -S(t_i, \mathbf{b})) + \sum \log(S(\nu_i, \mathbf{b}) - S(t_i, \mathbf{b})) \end{aligned} \tag{4.28}$$

where the first term is related to the uncensored observations, the second to the right-censored, the third on the left censored ones and the last on the interval censored observations, being ν_i the lower censoring interval. Please note that within this study, the observations are right censored, as stated in subsec.4.3.1.
Through the density function transformation rule, the relation that bounds the probability density function of T and ϵ is clearly:

$$f(t, \mathbf{b}) = \frac{g((\log t - \mu)/\sigma)}{\sigma t} \tag{4.29}$$

In order to obtain the MLE estimate of $\mathbf{b}$, that is $\hat{\mathbf{b}}$, for each element of $\mathbf{b}$:

$$\frac{\partial l(\mathbf{b})}{\partial b_i} = 0 \quad \forall\ i\ in\ \mathbf{b} \tag{4.30}$$

Since in many case Eq.4.30 cannot be solved in closed form, numerical optimization algorithms have to be used. Within this study, the `R` package used `flexsurv`.

Parameters Inference In order to test the linear relationship presented in Eq.4.27, the following test was performed:

$$\mathbf{L}\mathbf{a}' = c \tag{4.31}$$

where $\mathbf{L}$ is a constant vector or matrix that represent the linear hypotheses and $\mathbf{c}$ a constant vector. The test statistic used within this study have been the *Wald's statistics* [71]:

$$X_W = (\mathbf{L}\hat{\mathbf{a}} - \mathbf{c})'[L\hat{V}_a(\hat{\mathbf{a}})L']^{-1}(\mathbf{L}\hat{\mathbf{a}} - \mathbf{c}) \tag{4.32}$$

where $\hat{V}_a(\hat{\mathbf{a}})$ represents the sub-matrix that corresponds to $\mathbf{a}$ within the covariance matrix $\hat{V}(\hat{\mathbf{b}})$. The latter is the covariance matrix of the MLE estimate $\hat{\mathbf{b}}$.
Under some (mild) assumptions, X_W follows an asymptotic chi-squared distributions with η degrees of freedom, being η the rank of $\mathbf{L}$ [72].
Since this statistic has been used widely within the AFT study, an example of the related statistical test is presented below.

Single covariate Wald's test Let the number of covariates be equal to p. Let's suppose that the following test has to be performed:

$$H_0 : \ a_i = 0 \tag{4.33}$$

then, using Eq.4.32:

$$X_W = \frac{(\hat{a}_i)^2}{\nu_{11}} \tag{4.34}$$

being ν_{11} the variance of the MLE estimate $\hat{a}_i$. Since 4.34 has an asymptotic chi-squared distribution, the rejection rule becomes: $X_W > \chi^2_{1,\alpha/2}$.

4.5.2.2 Probability distributions

As stated in the previous subsections, within the AFT models, the distribution of the error term ϵ induces a distribution on the survival time T. This relation includes the shape of $g(\epsilon)$ and the parameters of the resulting distribution of T.
If $g(\epsilon)$ is properly shaped, $f(t, \mathbf{b})$ assumes a know parametric form, with parameters depending on the regression terms in Eq.4.27. For some distributions, such as the Weibull or the LogNormal, the distributions parameters are easily computed starting from $g(\epsilon)$ [73]. As for another distribution used within the study, the Generalized F, the computation degenerates fast. The detailed computation can be found in [68].

In any case, let's summarize the survival distributions parameters for the Weibull and the Lognormal distributions. The following parameters are related to the ith individual:

- **Weibull:** $\lambda_i = \exp \frac{-\mu_i}{\sigma}$; $\gamma_i = \frac{1}{\sigma}$
- **Lognormal:** $\mu_i^{distr} = \mu_i$; $\sigma_i^{distr} = \sigma$

where $\lambda_i, \gamma_i, \mu_i^{distr}, \sigma_i^{distr}$ are distribution parameters related to the survival probability distribution of the ith individual. The terms μ_i, σ represents the regression term in eq. (4.27) (μ_i represents the regression term of the ith individual).
From the relations above, it appears more clearly that the AFT models induce a survival distribution for each subject within the Survival Data.

4.5.3 Results

The results presented below are basically divided in two parts, both from the conceptual and the operative point of view.
First, an accurate, statistically significant parametric distribution fitting has been found for each industrial sector. The classical regression tests are performed, such as the covariates selection one. Then, a train-test approach is used to validate the models found.
Finally, an extensive elaboration of the results permitted a mindful interpretation of them. This is performed not just for sake of completeness within the study, but also in order to create a valuable practical resource. That is, not just a statistical model that provides knowledge of the underlying stochastic process (the IPO process in this case), but a framework that provides intelligence from that knowledge.

First fittings & Covariates Selection

First of all, a numerical and rigorous point of view on the fitting quality have to be provided.

In light of the structure of the fitting, that is based on a MSE estimate, the right indicator seemed the Maximum LogLikelihood Estimate (MaxLLE) achieved during the optimization process, in each sector and for each distribution. Please note that it is a *non-negative* likelihood, so the best fitting distribution is the one with the least value represented.

Sector	Exponential	Weibull	LogNormal	Generalized F
1	-9258.55	-9249.67	-9215.72	-9184.80
2	-43845.69	-43564.53	-43582.75	-43434.14
3	-8560.35	-8537.03	-8568.98	-8527.80
4	-12708.01	-12645.53	-12729.76	-12630.67
5	-11450.71	-11338.84	-11454.22	-11327.19
6	-4101.46	-4100.10	-4105.58	-4092.11
7	-16693.33	-16677.60	-16844.15	NA
8	-19923.64	-19921.53	-20067.58	-19896.60
9	-13910.47	-13903.96	-13984.55	-13900.13
10	-140752.78	-140418.35	-141171.65	-140288.54

Table 4.4: LogLikelihood of the AFT fitting. All covariates used. Best fits.

Tab.4.4 shows how Generalized F is the best fit for all sectors, when it exists. In fact, the optimization complexity brought by this distribution can sometimes prevent the algorithm optimization itself from converging. This is the case for sector 7 (Consumer).
These numerical results have to be considered along with the graphical ones presented in the plots below (Fig. 4.9). An example is indeed represented by sector 3 (Electronics): as the third plot in Fig. 4.9 testifies, it is indeed better graphically fitted by the exponential family, specifically by the Weibull distribution.
In light of these results, it is worth to note that in each sector, the numerical values in Tab.4.4 for some distributions are extremely close to each other. This points out that in some sectors, 2 distributions are almost *equally* well-fitted . Sector 3 is indeed a nice example: the least value is achieved by the Generalized F, but the plot shows that the survival curve pattern is better caught by the exponential family. In the end, it has been a matter of trade-off between the graphical interpretation and the likelihood value evaluation.
This trade off can be fully evaluated only after the statistically relevant covariates selection, that has been performed for each sector and for each distribution.

Wald's test covariates selection Finally, the covariates that presented a statistical significance at least 90% are selected, in each sector and for each distribution. The significance is tested using the Wald's statistic procedure described in Sub-

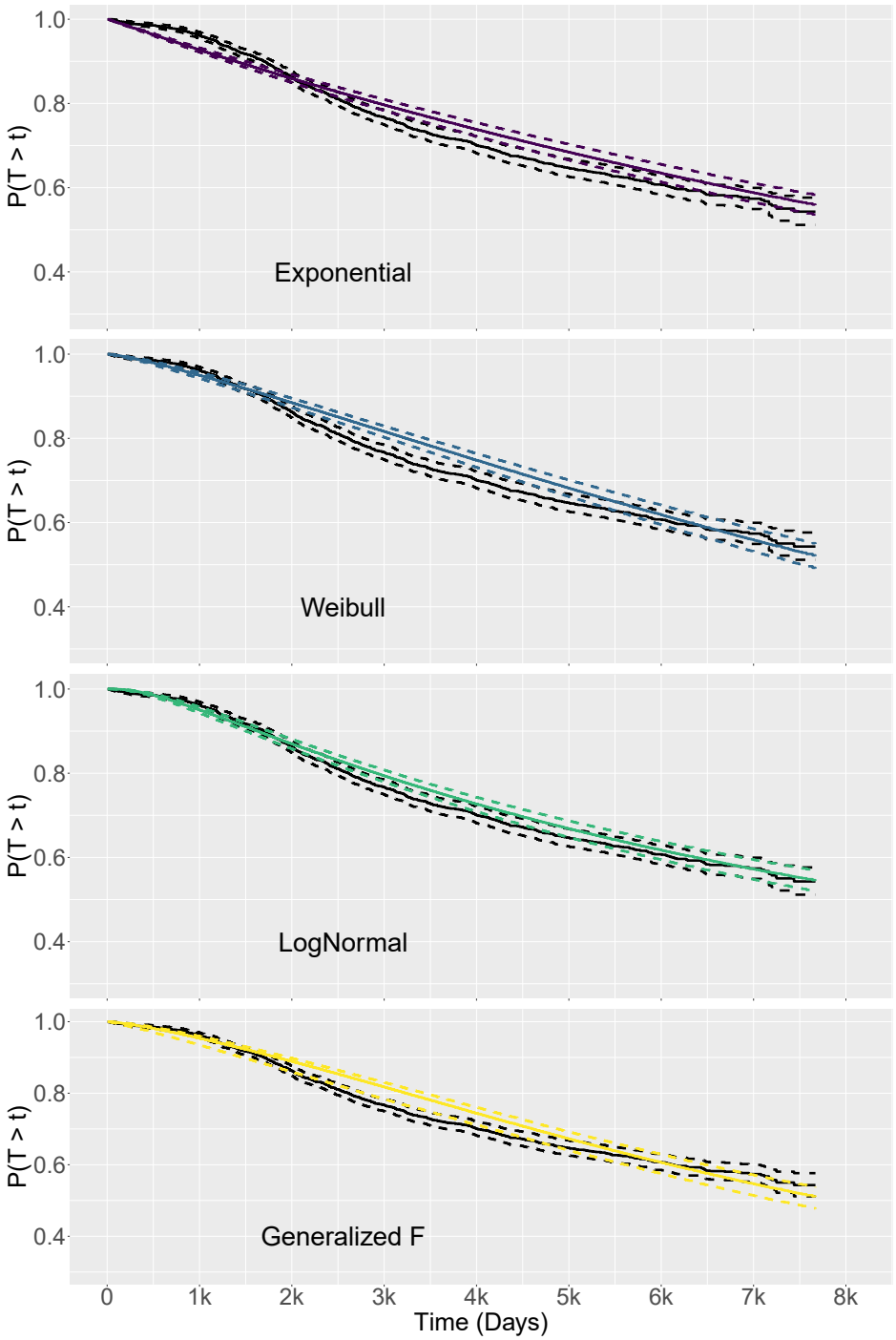

Figure 4.9: Sector 3, all covariates AFT survival probability distributions. Fitted distributions: Exponential, Weibull, Lognormal, Generalized F. The black solid lines represent the Kaplan-Meier survival distribution estimate. The dashed lines are the confidence intervals. The y axis is the survival probability estimate, $\hat{S}(t)$.

sec.4.5.2.1.
Below the number of the selected covariates for each sector and for each distribution.

Sector	Exponential	Weibull	LogNormal	Generalized F
1	8	10	8	6
2	10	9	8	7
3	7	7	7	7
4	8	8	6	6
5	3	3	3	3
6	0	0	1	0
7	2	2	2	0
8	3	3	3	0
9	2	3	2	3

Table 4.5: Number of covariates with a statistical significance over 90%.

It is worth to evaluate the *Akaike Information Index*, that measures the model likelihood and penalizes the models with the largest number of variables.
Even if this index should have been more *favourable* with the models that use the selected covariates, the difference w.r.t. the full-dataset AFT models is very narrow. Indeed, as Tab.4.7 testifies, the AIC is (very) slightly worsened w.r.t. the full-dataset models one.

Sector	Exp.	Wei.	LogN.	Gen. F	Exp.	Wei.	LogN.	Gen. F
1	18547	18531	18463	18406	18570	18544	18483	18442
2	87721	87161	87198	86904	87753	87209	87235	87126
3	17151	17106	17170	17092	17169	17128	17184	17113
4	25446	25323	25492	25297	25459	25341	25510	25328
5	22931	22710	22940	22690	22946	22722	22957	22704
6	8233	8232	8243	0	0	0	8230	0
7	33417	33387	33720	0	33430	33405	33738	0
8	39877	39875	40167	0	39872	39870	40166	0
9	27851	27840	28001	27836	27896	27890	28033	27887

Table 4.6: AIC before (left) and after (right) the covariates selection.

Conclusions, fitting plots and distributions selection

Some important question are due at this point of the study: are the AFT models consistent? Which distribution must be chosen in each sector, according to which metrics?
First, the graphical evaluation of the fitting presented within the selected covariates plots (Fig.s 4.10, 4.11) shows that the AFT models are able to model the empirical survival distribution (black solid lines in the plots) of the fitting and of the validation set. Sure enough this performance is sector-dependent: Tab.4.8 testifies how in some sectors the number of the significant regression coefficients is very low, just

Sector	Exponential	Weibull	LogNormal	Generalized F
1	-0.126	-0.071	-0.108	-0.200
2	-0.036	-0.055	-0.043	-0.255
3	-0.109	-0.129	-0.080	-0.126
4	-0.050	-0.071	-0.074	-0.122
5	-0.063	-0.056	-0.074	-0.058
6	0.000	0.000	0.164	0.000
7	-0.041	-0.053	-0.051	0.000
8	0.012	0.012	0.002	0.000
9	-0.161	-0.181	-0.113	-0.182

Table 4.7: Percentage variation in AIC before and after covariates selection.

1 in sector 6. Even if within sector 1 the Generalized F better catches the pattern of the KM curve w.r.t. the exponential by far (Cox-Snell residuals on Fig.4.5), the number of significant coefficients is much higher for the latter.

As for the distribution choice, it is evaluated through an integrated approach. The selected distribution in each sector must have the best MaxLLE and it must minimize the number of Wald's selected covariates. This without forgetting about the fitting quality insight provided by the validation survival plots.graphical and well approximate the KM curves. Sure enough the MaxLLE difference across distributions in each sector was extremely low compared to its order of magnitude (Tab. 4.4). It follows straightforwardly that this metric alone could not be considered as a reliable approach for selecting the best fitting distribution. On the other hand, it has been noted that the distributions with the highest MaxLLE coincides with the ones whose Wald's test has selected the least number of variables. This is true for all sectors, except for the ninth one. In light of these considerations, the choice of the optimal distribution deeply influenced the covariates choice, fo each sector.
The validation plots has been finally analyzed in order to choose the distributions that coincided whit the best KM curves pattern recognition *and* the least number of selected covariates, in *double validation* way. The result of such distribution and covariates procedure is resumed in Tab. 4.8.
On sectors 4 and 9, the selection criteria does not agree. As for sector 4, the LogNormal distribution approximates the validation set KM curve better than the Genaralized F, which is the greatest MaxLLE distribution. The right tail of the curve strongly highlighted this: the Genaralized F lied outside the KM confidence intervals. So, it has been preferred the LogNormal. As for sector 9 the Wald's test pointed to the Exponential or LogNormal as the best choices. Nevertheless, the Weibull distribution coincided with an higher MaxLLE w.r.t. the Exponential and a robust validation fitting plot. Actually, one could have expected some issue from the ninth sector (named "Other"), since it contains all the companies without a specific business activity connotation.

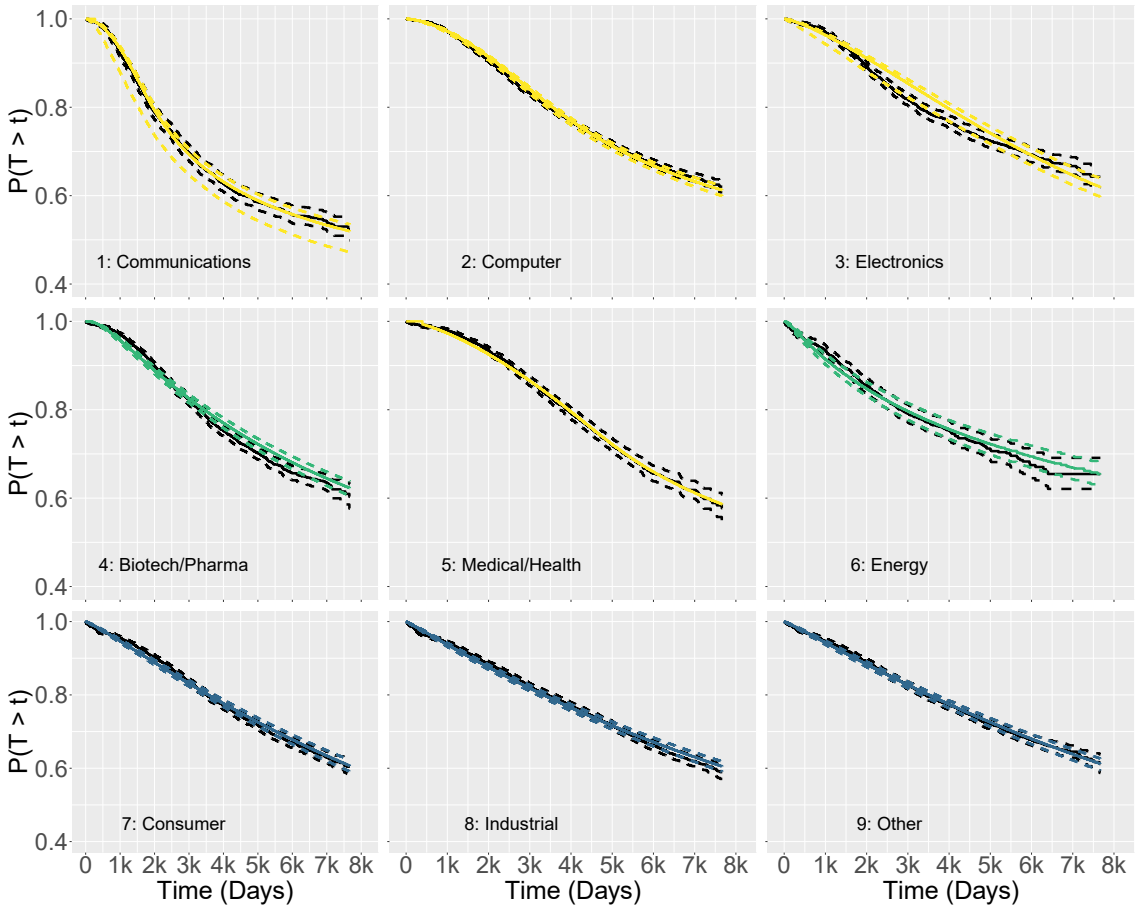

Figure 4.10: Unconditioned fitting survival plots, selected covariates models. Fitted distributions: Exponential (Violet), Weibull (Blue), Lognormal (Green), Generalized F (Yellow). The black solid lines represent the Kaplan-Meier survival distribution estimate of the fitting set. The dashed lines are the confidence intervals. The y axis is the survival probability estimate, $\hat{S}(t)$.

Sector (distr.)	1 (Gen.F)	2 (Gen.F)	3 (Gen.F)	4 (LogNorm.)	5 (Gen.F)	6 (LogNorm.)	7 (Wei.)	8 (Wei.)	9 (Wei.)
# sign. cov.s	6	7	7	6	3	1	2	3	3

Table 4.8: Number of covariates with a statistical significance over 90% and greatest MaxLLE in that sector. For each sector, the selected distribution

Validation plots Since the models selection procedure involved such plots, it is worth to describe them in more detail. Goal of these graphs is to *validate* the AFT selected models in a train-test set fashion. Even if this is a common practice in the Machine Learning environment, it has been useful also within this survival analysis, since it provided a deeper insight on the accuracy of the selected AFT models.
The procedure has been designed as the following:

1. The initial dataset is split between training and test subsets. Training set consists in the 90% of the total dataset.

2. For each sector: the selected parametric AFT distributions for each sector are fitted on the selected covariates training set and the Kaplan-Meier survival estimates of test set are performed.

   (a) The two curves are compared along with their confidence interval.

Within this time-prediction survival setting, the KM curves represent the empirical, *test* estimate of the survival probability $S(t)$, while the AFT curves represent its fitted estimate $\hat{S}(t)$.
The more the sector related $\hat{S}(t)$s will be closer to their KM estimates, the better the AFT fitting will be proven to be in that sector.

In each plot, the fitting curve is abundantly inside the space delimited by the confidence intervals of the KM estimate.
In sector 6 (Fig.4.10) these intervals are very broad, since the number of subjects within this sector is restricted w.r.t. the other business sector. This lack of companies produces a less accurate KM estimate.

4.5.4 Accelerating factors & Sensitivity analysis

As already stated, highlighting the factors (variables) that would made the IPO *most probable* in time is a valuable resource from the practitioners point of view.
Indeed, the AFT models provide naturally such features as described in Sec.4.5.2.1.
The table below presents the regression coefficients found for each sector, regarding the probability distribution selected for that sector.
Please recall that the negative coefficients are related to the covariates that accelerate the time-to-IPO, while the positive ones decelerate the survival curve.

For sake of completeness, even if these models won't be used, below the same table, but with the models fitted using the covariates selection approach provided by the Wald's test. Please note that the accelerating or decelerating value of each covariate changes w.r.t. Tab.4.9.

In order to recap, below the number of accelerating or decelerating factors for each sector, for the complete dataset:

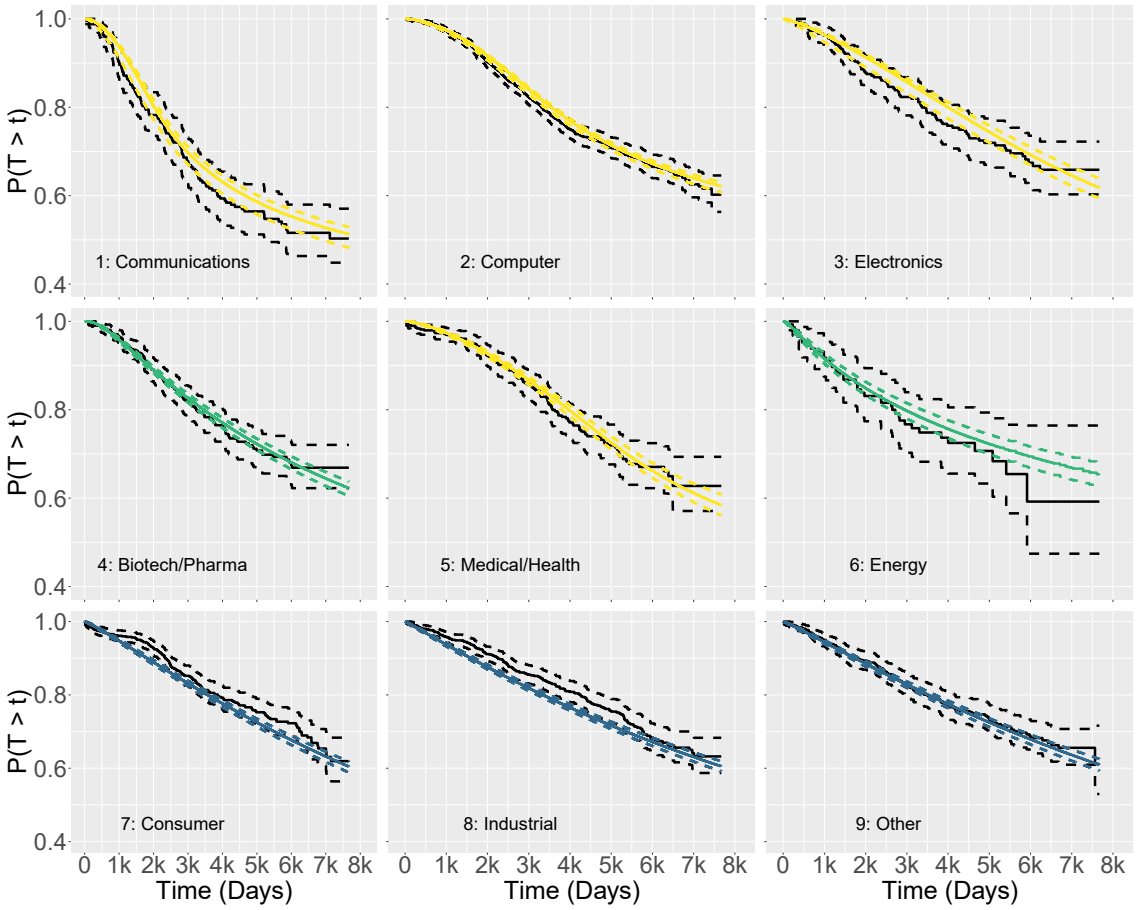

Figure 4.11: Unconditioned validation survival plots, selected covariates models. Fitted distributions: Exponential (Violet), Weibull (Blue), Lognormal (Green), Generalized F (Yellow). The black solid lines represent the Kaplan-Meier survival distribution estimate of the validation set. The dashed lines are the confidence intervals. The y axis is the survival probability estimate, $\hat{S}(t)$.

Sec.	min1	max1	mean1	min2	max2	mean2	min3	max3	mean3	num1	num2	num3	vix1	vix2
1	0.16	0.17	-0.12	-1.41	-0.76	2.13	0.35	0.36	-0.84	-0.13	-0.05	-0.03	-0.12	-0.10
2	-0.26	-0.04	0.41	-0.36	-0.10	0.51	-0.22	0.00	0.14	-0.11	-0.07	-0.02	-0.06	-0.15
3	0.08	0.13	-0.22	-0.69	0.40	0.40	-1.31	-0.72	1.99	-0.09	-0.16	-0.09	-0.01	-0.08
4	0.02	0.27	-0.23	-1.09	-0.46	1.54	-1.51	-0.36	1.83	-0.09	-0.06	-0.09	0.00	-0.02
5	-0.08	-0.04	0.07	0.29	0.65	-0.92	-0.51	-0.15	0.60	-0.04	-0.04	-0.08	0.01	-0.04
6	-0.14	-0.10	0.16	-1.88	-1.06	2.92	1.47	1.61	-3.19	0.19	-0.06	0.09	0.07	-0.15
7	0.27	0.22	-0.59	-0.54	-0.35	0.90	-0.24	-0.30	0.53	-0.11	-0.03	-0.02	0.00	-0.04
8	0.44	0.06	-0.61	-0.74	-0.23	0.92	-0.08	0.19	-0.10	0.02	-0.13	0.00	0.02	-0.02
9	0.12	0.05	-0.29	-0.15	-0.13	0.30	-0.70	-0.01	0.66	-0.07	-0.11	-0.11	0.01	-0.00

Table 4.9: Accelerating or Decelerating factors for each industrial sector

Sec.	min1	max1	mean1	min2	max2	mean2	min3	max3	mean3	num1	num2	num3	vix1	vix2
1	-	-	-	-1.59	-0.95	2.56	-0.23	-0.11	-	-0.17	-	-	-	-
2	-0.29	-	0.42	0.07	0.02	-	-	0.05	-0.09	-0.14	-	-	-	-
3	-	-	-	0.10	-	-	-	-	-	-0.09	-0.10	-0.13	-0.01	-0.08
4	-	-	-	-1.22	-	1.18	-	0.04	-	-	-0.18	-	0.01	-0.03
5	-	-	-	-	-	-	-	-	-	-0.06	-	-	-	-0.04
6	-	-	-	-	-	-	-	-	-	0.26	-	-	-	-
7	-0.10	-	-	0.04	-	-	-	-	-	-	-	-	-	-
8	0.34	-	-0.45	-	-	-	-	-	0.03	-	-	-	-	-
9	-	-	-	0.04	-	-	-	-	0.01	-	-	-	-	0.01

Table 4.10: Accelerating or Decelerating factors for each industrial sector, selected covariates. Please note that the green values corresponds to accelerating factors, while the red ones are the decelerating factors.

If the meaning that *accelerates* have within this framework is still unclear, the next paragraph should better highlight it.

Sensitivity analysis In order to *show* the accelerating or decelerating influence that the covariates have on the survival distribution of each subject, a *sensitivity analysis* is performed.
A sample company i is drawn from a sector. The value of each covariate is then increased and decreased by a factor $x\%$. If the covariate a_j is an accelerating one and if its value is increased, the corresponding survival curve must be lower than the base one (the one plotted with the real a_j value).

Below and example from sector 1. The fitting procedure is the one regarding the selected covariates: it would have been very different to represent all the 14 covariates of the whole dataset, even if this will be the final model.

Please note that when every variable that has a negative value in Tab.4.10 corresponds to a survival distribution curve that is under the base one. So, for each time point of the curve, the probability for that company of not being IPO by that time is lower than it would have been if that variable were not increased.

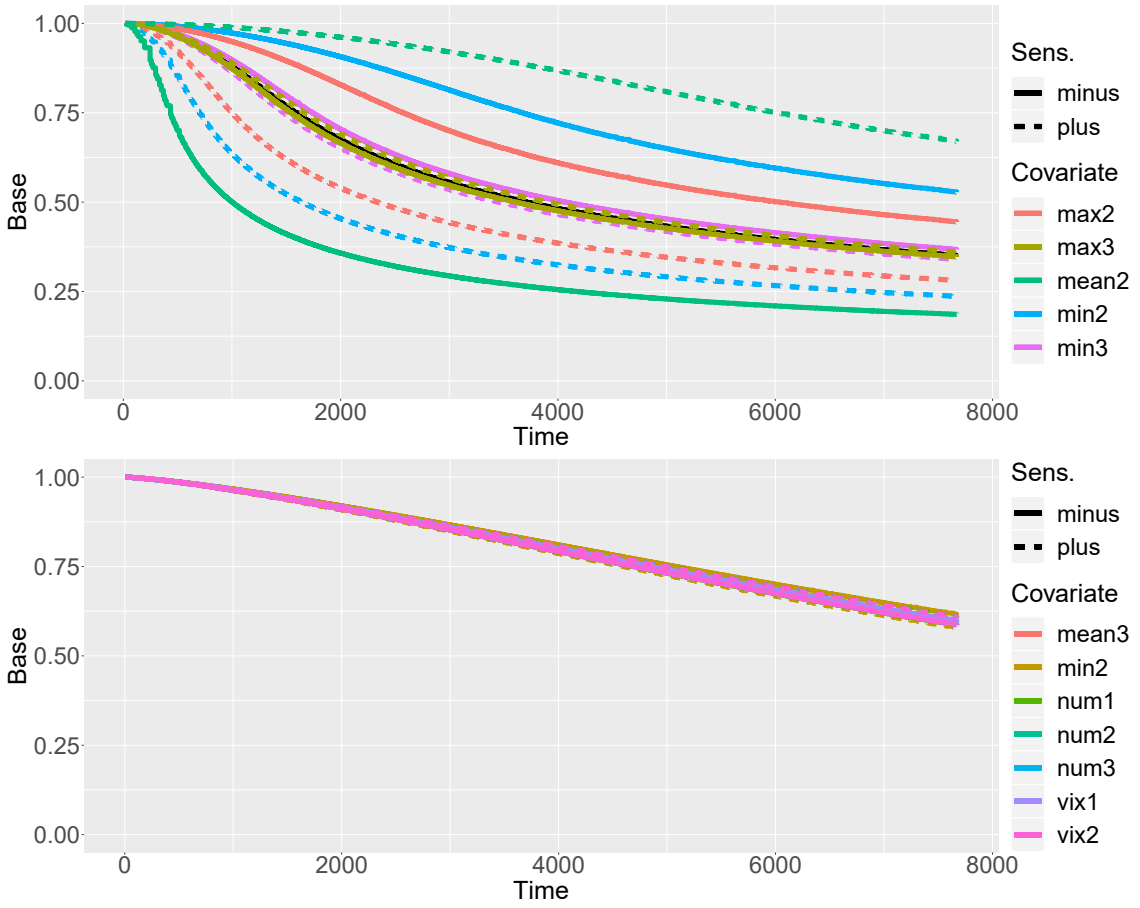

Figure 4.12: Sensitivity plots. Top plot: sector 1. Bottom plot: sector 3. Dashed lines: increased covariates. Solid lines: decreased covariates. Black line: original survival curve

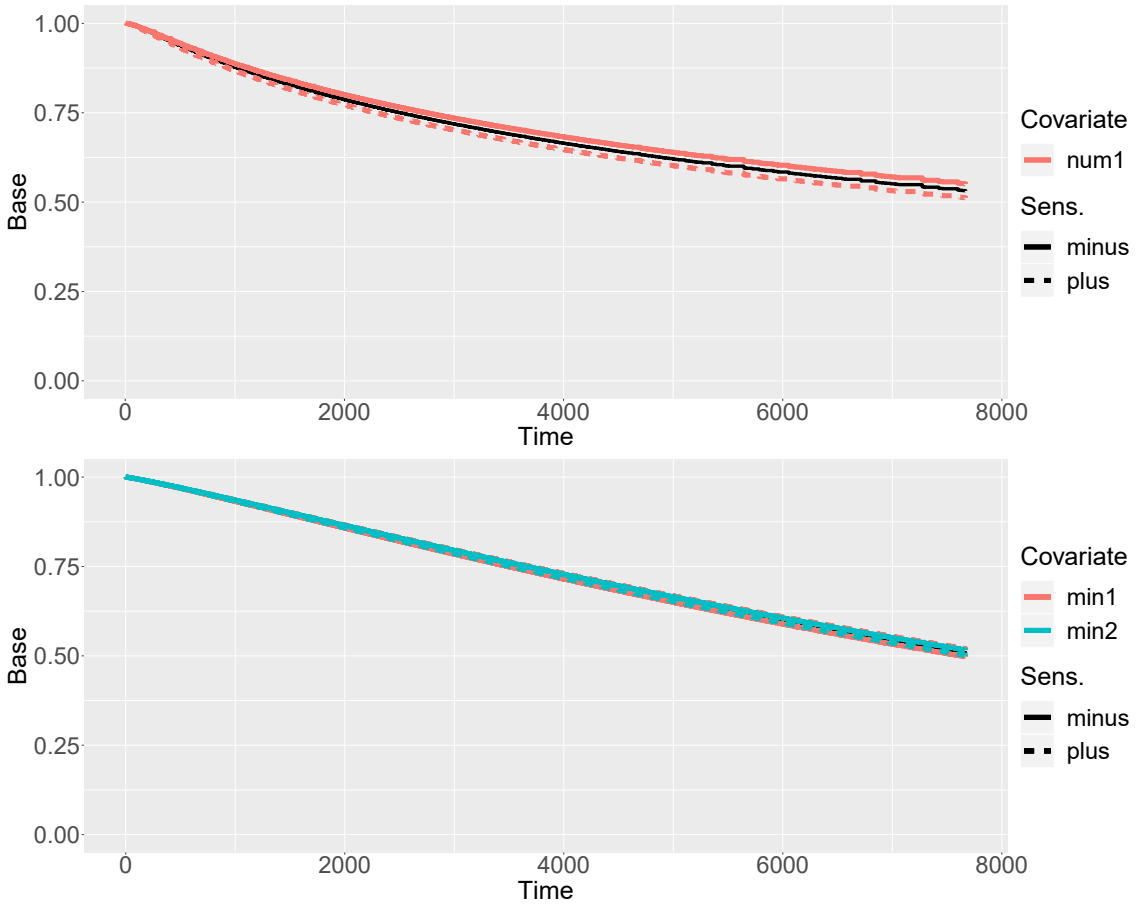

Figure 4.13: Sensitivity plots. Top plot: sector 6. Bottom plot: sector 7. Dashed lines: increased covariates. Solid lines: decreased covariates. Black line: original survival curve.

Factor Type	S1	S2	S3	S4	S5	S6	S7	S8	S9
# Decelerating	5	4	5	5	5	7	5	7	5
# Accelerating	9	10	9	9	9	7	9	7	9

Table 4.11: Number of accelerating factors for each sector.

Chapter 5

Conclusions

This master thesis work intended to develop a valuable tool, that exploits data to create knowledge Private Equity intelligent investments can be based on. Within PE environment, as well as in each investment framework, good strategies are able to increase investment returns average and to reduce variance. Because of the PE market chronic lack of data and the informative asymmetry described in Chapt. 1, it was, since its beginning, an hard goal to be achieved.
However, the algorithms and methods developed in this work provided robust answers to the research question related to this difficult endeavour. To recap, they were:

- Predict Private companies' future statuses.
- Model the Time-to-IPO event.

Let's develop a thoughtful analysis on such questions responses.

The neural networks and random forest tuned and trained provided both strong results in terms of predictive performance. The training dataset was strongly unbalanced, an issue that is widely known to have a critical impact on the Machine Learning algorithms' predictive capability.
Nevertheless, the Random Forest tuned models reached high accuracy on each OVR classification and very high minority class recall. It is due to point out that the classification threshold probability deeply influenced the increase of the latter indicator w.r.t to the 50% threshold. However this is interpretable as a practitioner related strategy, since for both extremely high or extremely low returns, even a small probability can act as a *decision threshold*. As an example, let's suppose that a PE investor is asked to evaluate the investment in a company whose predicted bankruptcy probability is higher than 10% (the probability threshold proposed in this work for the RF bankruptcy model). Within the PE framework, this is a sufficient threshold to *label* such company as a bad investment choice, since if it really goes bankrupt, the related investment would result in a total loss. Vice versa, if the company IPO probability is higher than 55%, any PE investor would be sure on the choice of whether to invest in it or not, given the enormous returns associated to such status.

On the other hand, the Neural Networks models carefully tuned in this work provided an extremely similar results in terms of predictive performance indicators, without the *ex post* tuning of the probability threshold needed for the RF. Sure enough the aforementioned rationale on the investment decision threshold is still valid for the NNs, but their performance appears to be more robust than the RF one. Such absence of sensitivity to *ex post* manipulation is always preferable in algorithms' design. From the computational point of view, the advanced oversampling algorithms (SMOTE, ADASYN and SVMSMOTE) succeeded in increasing the NNs classification quality. In fact, it has been already remarked that neural networks are widely recognised to suffer for narrow and unbalanced data. To confirm this, at the beginning of this work the performance of the NNs on the original dataset has been explored and it produced practical *no prediction*. In light of all these considerations, NNs have been chosen as the predictive company future statuses classifiers.

As regards the Survival Analysis, it is worth to highlight the central importance of the IPO event for any Private Equity investor. The investments in private companies present astonishing returns when the benefited company goes IPO (as the Peter Thiel's 2004 investment in Facebook, appreciated 693,3%). Actually, IPO is the ultimate goal of a number of large PE investment firms, so the inference of the time horizon of such event has a crucial importance, especially in an environment that suffers so much from lack of information as the PE one. In this work, a strong response to this need is provided through the Survival Analysis, in particular to the carefully tuned Accelerated Failure Time models presented in Chapt.4. These have proven, according to the validation plots and the statistically significant covariates selection procedure, to interpret reliably the probabilistic pattern of going IPO in time for the companies of all the 9 industrial sectors. Moreover, the accelerating/decelerating factor analysis crafted an informative investigation every PE practitioner would be interested in: how much the strength of the first round investors affect the time to IPO? How much they do the second round investors? The aforementioned procedure is able to reply to such important questions with an high statistical accuracy. Sure enough it could be pointed out that the validation plots show how the selected AFT models reliable fit empirical curves built with other models, namely the Kaplan-Meier curves. However, KM curves are simple counting methods whose reliability has been extensively testes within the medical environment, whose need for safe predictions is obviously great. So, validation plots seems to be in any case a robust testing method for the AFT models.

Within this work, possible further enhancements of the present models have been sketched. Some investigation to find out if any investment data is really available for the PE investor could lead to the inclusion of such variable to the existing models without a considerable effort. This could increase the classifiers accuracy, especially the one related to the minority class. If so, deeper neural architectures could be explored to further improve such metrics.

Finally, it can be stated that the algorithms and models designed and tested in

this work provide an informative, powerful tool to drive Private Equity investment decisions. The future status probability estimation constitutes a robust support to the investment evaluation, since it forecast the returns of such investment with an accuracy that is considerable within the PE framework. The survival analysis add an ulterior tool for the practitioners, that will be able to model with low error a time horizon for the highest PE investment return event: IPO.

Appendix A

Deep Neural Networks experiments

It seems due to show the NNs' results regarding deeper architecture w.r.t. the ones presented in Chapt. 3.

Deep NNs are often simply composed by staked hidden layers. A large number of state of the art NNs model, such as GoogLeNet or VGG, are indeed extremely deep nets, whose reliability and prediction power is widely known. The increase in depth corresponds to an increase in the total number of neurons, that is directly linked to number of the network trainable weights. Intuitively, the more the weights, the larger the computational effort needed to train the network and consequently the training time. As for the LSTM models, each neuron is indeed a *cell* that contains much more parameters w.r.t. a simple neuron, so the aforementioned increase is enlarged.

Scope of this procedure has been the investigation of the impact of the number of trainable weights on predictive performance.
From the architecture point of view, deep MLP models consists in simply staked hidden layers graphs, in which the output of each layer is the input of the following one. LSTM deep models are built in a similar way: the output of each cell in injected in the next cell of the same layer *and* the corresponding cell of the next layer.

Before exploring the adopted methods, it is important to remark that there is no theoretical background regarding the tuning of a neural network (i.e. a recognized step-by-step procedure to follow). The following experiment's result highlighted the need to tune *each layer*, with an extremely large need of computational power. As for the MLP, the procedure used consisted in a simple depth increasing method. The MLP selected models in Chapt. 3 have been explored within the Bankrupt, IPO and Acquisition classifications. For each OVR classification, the depth has been increased until the nets included more than 2.000 parameters. The LSTM procedure has been basically the same, but for computational time reasons just one deep model has been trained, with more than 14.000 parameters. Recurrent dropout [74] is used instead of the basic output one.

Deep models results and conclusions As the results testify, the increase on depth did not provide significantly improvement w.r.t the shallow models. This im-

# Hidden Layers	Precision+	Precision-	Recall+	Recall-	Accuracy
1	0.10	0.99	0.67	0.81	0.81
2	0.11	0.99	0.57	0.86	0.85
3	0.10	0.99	0.65	0.82	0.81
4	0.10	0.99	0.63	0.84	0.83
5	0.08	0.99	0.69	0.78	0.77
6	0.09	0.99	0.69	0.79	0.79
7	0.09	0.99	0.70	0.78	0.78
8	0.09	0.99	0.67	0.80	0.79
9	0.08	0.99	0.78	0.73	0.73
10	0.09	0.99	0.68	0.78	0.78

Table A.1: Bankrupt classification. Number of input neurons: 14. Number of hidden neurons per layer: 14. Depth of the model is expressed as the number of its hidden layers. Dropout rate = 0.50.

# Hidden Layers	Precision+	Precision-	Recall+	Recall-	Accuracy
1	0.02	0.99	0.23	0.90	0.89
2	0.02	0.99	0.24	0.88	0.88
3	0.02	0.99	0.28	0.87	0.87
4	0.02	0.99	0.44	0.83	0.83
5	0.02	0.99	0.18	0.94	0.94

Table A.2: Acquisition classification. Number of input neurons: 28. Number of hidden neurons per layer: 14. Depth of the model is expressed as the number of its hidden layers. Dropout rate = 0.50

plies that the numbers of parameters dependence on the test set performance is very low, suggesting that the deep architectures probably need enriched data structures, maybe equipped with variables related to quantitative investment information. In Chapt. 1, it has been clearly stated that such type of information is rarely available by investors. However, further, complex analyses on the data sources could be performed to explore possible enhancements on that path. Sure enough, such effort would be extremely extensive from the practical point of view.

	Precision+	Precision-	Recall+	Recall-	Accuracy
Priv./Not-Priv.	0.438417	0.685327	0.522194	0.608722	0.576787

Table A.3: Private classification. Number of input neurons: 28. Number of hidden neurons per layer: 14. Number of parameters: 14.254

# Hidden Layers	Precision+	Precision-	Recall+	Recall-	Accuracy
1	0.41	0.73	0.56	0.60	0.59
2	0.42	0.73	0.52	0.65	0.60
3	0.44	0.72	0.45	0.71	0.63
4	0.41	0.73	0.55	0.62	0.59
5	0.40	0.75	0.65	0.52	0.56
6	0.41	0.73	0.54	0.62	0.59
7	0.42	0.73	0.51	0.65	0.60
8	0.41	0.73	0.56	0.60	0.59
9	0.41	0.73	0.57	0.59	0.58
10	0.41	0.73	0.55	0.61	0.59

Table A.4: IPO classification. Number of input neurons: 7. Number of hidden neurons per layer: 6. No Dropout.

Appendix B

Neural Networks training gradient monitoring

Within this brief annex, it is provided an useful insight on the optimization procedure of one of the NNs selected models. In particular, the bankruptcy classifier, corresponding to a (14,14) MLP with dropout rate equal to 0.5, trained on an SMOTE oversampled set.

The metric this annex is focused on is the gradient norm. In fact, as in any numerical optimization task, the norm of the objective function gradient w.r.t. the decision variables provides a clear evidence on the optimization quality itself. In particular, it attempts to reply to the question: is the gradient really *descending*?
In formulas, the gradient L^2 norm GN used in this procedure reads as:

$$GN = \|\nabla_\theta J(\theta, \mathbf{x})\|_2 = \sqrt{\sum_{i=1}^{W} (\nabla_\theta J(\theta, \mathbf{x})_i)^2} \tag{B.1}$$

where $\nabla_\theta J(\theta, \mathbf{x})$ is the norm of the (discrete) cost function, that depends on the NN model. W is the total number of trainable parameters.

As described in Subsec. 3.4, within NNs training-related optimization process, a number of issues can occur: hessian matrix ill-conditioning, cliffs, etc. These problems can be monitored through the gradient norm. If it actually decreases throughout the training process, the hessian matrix is unlikely to be ill-conditioned. Moreover, if such metric does not have sudden *spikes*, probably the algorithm did not encounter a cliff in its path.
Please note that the gradient norm monitoring has provided through the NNs design process a powerful insight on the reliability of the optimization algorithm utilized, the Adam. From the literature-related point of view, Adam should overcome the aforementioned optimization issues. However, an accurate check is due.

In Fig. B.1, the bankruptcy NN model gradient norm evolution through training epochs is provided along with the test and training losses. The plot highlights

the absence of spikes and the actual descent of the gradient norm, excluding reliably the presence of bad topologies. However, it is worth to remark the extreme high value of the gradient norm itself and the stop of its descent after a few training epochs. This is in contrast with the stability of the loss value: if the gradient is high, the descent of the objective function value should be large as well.
Such behaviour can be imputed to the attainment of a local minima by the optimization algorithm. Again, as stated in Subsec. 3.4, this is a common issue within NNs training, but it has been proven that neural networks-related objective functions can show several local minima with extremely similar (and low) cost value [37], [38].

In light of these observations, the Adam optimizer is confirmed to be a reliable training engine for this work.

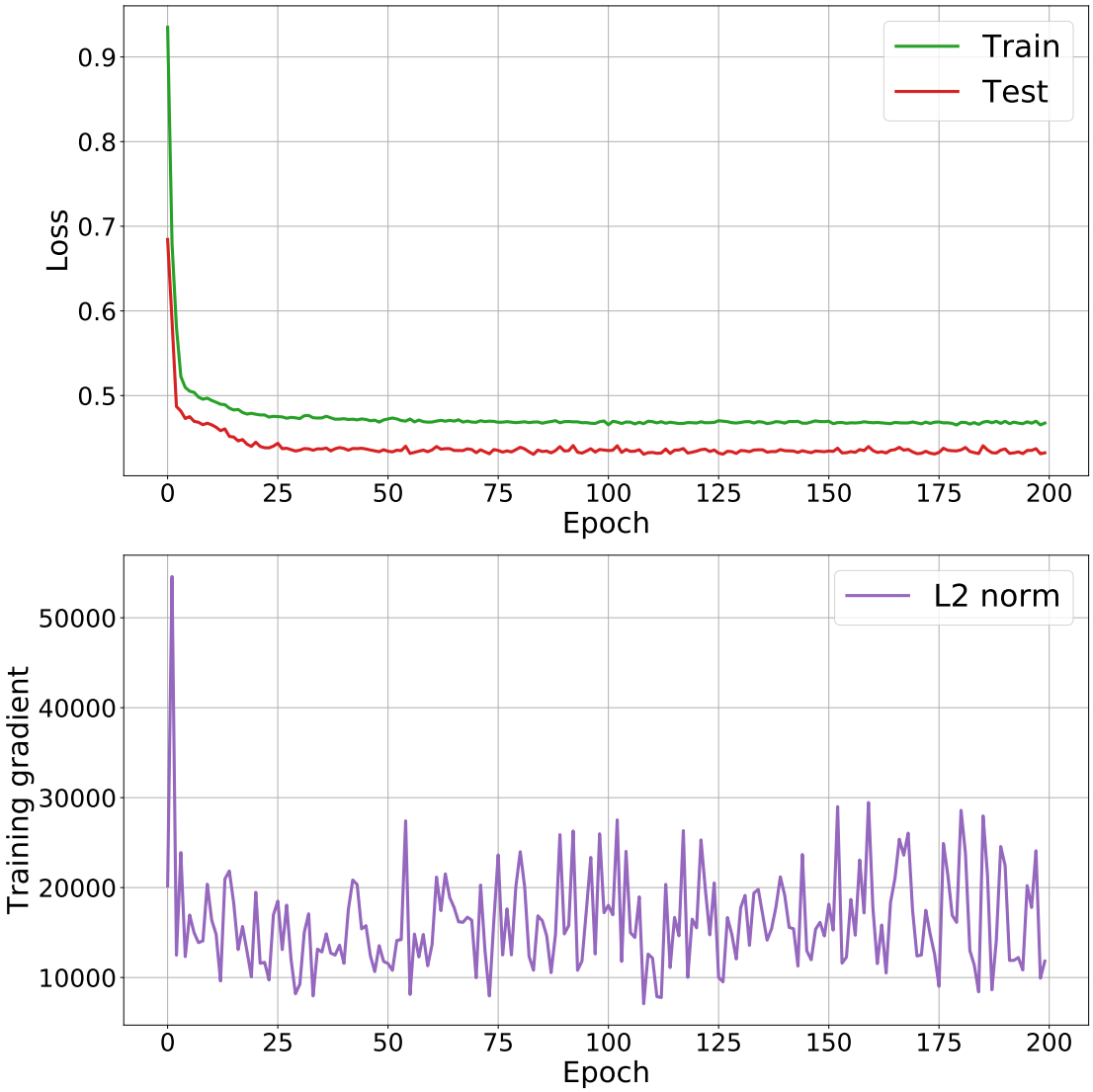

Figure B.1: Bankruptcy classification, optimization process monitoring through objective function gradient L^2 norm evaluation. Model architecture: (14,14) MLP, dropout rate = 0.5. Training set oversampling method: SMOTE.

Bibliography

[1] Wikipedia. List of stock exchanges. https://en.wikipedia.org/wiki/Listofstockexchanges.

[2] Bain & Company. Global Private Equity Report 2019. https://www.bain.com/contentassets/875a49e26e9c4775942ec5b86084df0a/bain_report_private_equity_report_2019.pdff. [Online; accessed 15-October-2019].

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Number 1. MIT press, 2016.

[4] Emil Protalinski. How much is Facebook worth? https://www.zdnet.com/article/how-much-is-facebook-worth/, 2012. [Online; accessed 23-August-2019].

[5] Bruce Einhorn. Masayoshi Son's $58 Billion Payday on Alibaba. https://www.bloomberg.com/news/articles/2014-05-07/softbanks-58-billion-payday-on-its-alibaba-investment, 2014. [Online; accessed 19-July-2019].

[6] Wolfgang Härdle, Yuichi Mori, and Philippe Vieu. *Statistical methods for biostatistics and related fields*. Springer Science & Business Media, 2006.

[7] Zhongsheng Hua, Yu Wang, Xiaoyan Xu, Bin Zhang, and Liang Liang. Predicting corporate financial distress based on integration of support vector machine and logistic regression. *Expert Systems with Applications*, 33(2):434–440, 2007.

[8] Ermanno Pitacco. Survival models in a dynamic context: a survey. *Insurance: Mathematics and Economics*, 35(2):279–298, 2004.

[9] Marc J LeClere. Preface modeling time to event: Applications of survival analysis in accounting, economics and finance. *Review of Accounting and Finance*, 4(4):5–12, 2005.

[10] Harish S Bhat and Dan Zaelit. Forecasting retained earnings of privately held companies with pca and l1 regression. *Applied Stochastic Models in Business and Industry*, 30(3):271–293, 2014.

[11] Harish S Bhat and Daniel Zaelit. Predicting private company exits using qualitative data. pages 399–410, 2011.

[12] Tom AB Snijders. Network dynamics. *The handbook of rational choice social research*, pages 252–279, 2013.

[13] Ahmad K Naimzada, Silvana Stefani, and Anna Torriero. *Networks, topology and dynamics: Theory and applications to economics and social systems*, volume 613. Springer Science & Business Media, 2008.

[14] Leo Breiman. Out-of-bag estimation. 1996.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 201. Number 1. MIT press, 2016.

[16] Giuseppe C Calafiore and Laurent El Ghaoui. *Optimization models*. Cambridge university press, 2014.

[17] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[18] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 191. Number 1. MIT press, 2016.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 193. Number 1. MIT press, 2016.

[21] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[23] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, pages 196–197. Number 1. MIT press, 2016.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 184. Number 1. MIT press, 2016.

[26] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[27] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455, 2009.

[28] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 361. Number 1. MIT press, 2016.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[31] Yann LeCun et al. Generalization and network design strategies. In *Connectionism in perspective*, volume 19. Citeseer, 1989.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 179. Number 1. MIT press, 2016.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, pages 246–252. Number 1. MIT press, 2016.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 263. Number 1. MIT press, 2016.

[35] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 282. Number 1. MIT press, 2016.

[37] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[38] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[39] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[40] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*, page 310. Number 1. MIT press, 2016.

[42] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.

[43] Kevin S Woods, Christopher C Doss, Kevin W Bowyer, Jeffrey L Solka, Carey E Priebe, and W Philip Kegelmeyer Jr. Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(06):1417–1436, 1993.

[44] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[45] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[46] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, 2008.

[47] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288, 2008.

[48] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[49] Regina C Elandt-Johnson, Norman Lloyd Johnson, and Mathematischer Statistiker. *Survival models and data analysis*. Number 312.0151 E37. Wiley Online Library, 1980.

[50] Mohammad Modarres, Mark P Kaminskiy, and Vasiliy Krivtsov. *Reliability engineering and risk analysis: a practical guide*. CRC press, 2016.

[51] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, pages 19–63. John Wiley & Sons, 2003.

[52] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, pages 1–5. John Wiley & Sons, 2003.

[53] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, pages 1–5. John Wiley & Sons, 2003.

[54] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, page 11. John Wiley & Sons, 2003.

[55] Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

[56] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, page 67. John Wiley & Sons, 2003.

[57] DR Cox. Regression models and life tables (with discussion). lr statist, 1972.

[58] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, page 298. John Wiley & Sons, 2003.

[59] Norman E Breslow. Analysis of survival data under the proportional hazards model. *International Statistical Review/Revue Internationale de Statistique*, pages 45–57, 1975.

[60] John D Kalbfleisch and Ross L Prentice. *The statistical analysis of failure time data*, volume 360. John Wiley & Sons, 2011.

[61] Bradley Efron. The efficiency of cox's likelihood function for censored data. *Journal of the American statistical Association*, 72(359):557–565, 1977.

[62] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, page 302. John Wiley & Sons, 2003.

[63] David Schoenfeld. Partial residuals for the proportional hazards regression model. *Biometrika*, 69(1):239–241, 1982.

[64] David R Cox and E Joyce Snell. A general definition of residuals. *Journal of the Royal Statistical Society: Series B (Methodological)*, 30(2):248–265, 1968.

[65] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, pages 326–337. John Wiley & Sons, 2003.

[66] Patricia M Grambsch and Terry M Therneau. Proportional hazards tests and diagnostics based on weighted residuals. *Biometrika*, 81(3):515–526, 1994.

[67] Antonio Ciampi, Sheilah A Hogg, and Louis Kates. Regression analysis of censored survival data with the generalized f family—an alternative to the proportional hazards model. *Statistics in Medicine*, 5(1):85–96, 1986.

[68] Yingwei Peng, Keith BG Dear, and JW Denham. A generalized f mixture model for cure rate estimation. *Statistics in medicine*, 17(8):813–830, 1998.

[69] Christopher Cox. The generalized f distribution: an umbrella for parametric survival analysis. *Statistics in medicine*, 27(21):4301–4312, 2008.

[70] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, pages 259–263. John Wiley & Sons, 2003.

[71] Peter J Huber et al. The 1972 wald lecture robust statistics: A review. *The Annals of Mathematical Statistics*, 43(4):1041–1067, 1972.

[72] Peter CB Phillips. The exact distribution of the wald statistic. *Econometrica: Journal of the Econometric Society*, pages 881–895, 1986.

[73] Elisa T Lee and John Wang. *Statistical methods for survival data analysis*, volume 476, pages 269–277. John Wiley & Sons, 2003.

[74] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.