UNIVERSITY OF LJUBLJANA

SCHOOL OF ECONOMICS AND BUSINESS

MASTER'S THESIS

Empirical Credit Risk Modelling: A Comparison of Traditional Statistical and Machine Learning Classification Methods for Corporate Credit Scoring

Ljubljana, September 2019 DOMEN BIDER

AUTHORSHIP STATEMENT

The undersigned Domen Bider, a student at the University of Ljubljana, School of Economics and Business, (hereinafter: SEB LU), author of this written final work of studies with the title Empirical Credit Risk Modelling: Comparison of Traditional Statistical and Machine Learning Classification Methods for Corporate Credit Scoring, prepared under supervision of prof. dr. Aleš Berk Skok

DECLARE

1. this written final work of studies to be based on the results of my own research;
2. the printed form of this written final work of studies to be identical to its electronic form;
3. the text of this written final work of studies to be language-edited and technically in adherence with the SEB LU's Technical Guidelines for Written Works, which means that I cited and / or quoted works and opinions of other authors in this written final work of studies in accordance with the SEB LU's Technical Guidelines for Written Works;
4. to be aware of the fact that plagiarism (in written or graphical form) is a criminal offence and can be prosecuted in accordance with the Criminal Code of the Republic of Slovenia;
5. to be aware of the consequences a proven plagiarism charge based on the this written final work could have for my status at the SEB LU in accordance with the relevant SEB LU Rules;
6. to have obtained all the necessary permits to use the data and works of other authors which are (in written or graphical form) referred to in this written final work of studies and to have clearly marked them;
7. to have acted in accordance with ethical principles during the preparation of this written final work of studies and to have, where necessary, obtained permission of the Ethics Committee;
8. my consent to use the electronic form of this written final work of studies for the detection of content similarity with other written works, using similarity detection software that is connected with the SEB LU Study Information System;
9. to transfer to the University of Ljubljana free of charge, non-exclusively, geographically and time-wise unlimited the right of saving this written final work of studies in the electronic form, the right of its reproduction, as well as the right of making this written final work of studies available to the public on the World Wide Web via the Repository of the University of Ljubljana;
10. my consent to publication of my personal data that are included in this written final work of studies and in this declaration, when this written final work of studies is published.

Ljubljana, _____ Author's signature: _____

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICIES

LIST OF ABREVIATIONS

angl. - angleško
AdaBoost – adaptive boosting algorithm
AI – artificial intelligence
AJPES – Agency of the Republic of Slovenia for Public Legal Records and Related Services
ANNs – artificial neural networks
ANOVA – analysis of variance
AUROC – area under the receiver operating characteristic
B2B – business-to-business
BFGS – Broyden-Fletcher-Goldfarb-Shanno
BS – Brier score
CAPEX – capital expenditures
CGI – corporate governance indicators
CIS – Classification of Institutional Sectors
CPU – central processing unit
CS – credit scoring
CV – cross-validation

DL – deep learning
DPD – days past due
DPP – data pre-processing
DT – decision tree
EAD – exposure at default
EBIT – earnings before interests and taxes
EBITDA – earnings before interests, taxes, depreciation, and amortization
ECRM - empirical credit risk modelling
ENN – edited nearest neighbours
FDP – financial distress prediction
FN – false negatives
FP – false positives
FX – forex
GA – genetic algorithm
GDP – gross domestic product
GPU – graphics processing unit
IT – information technology
KKT – Karush-Kuhn-Tucker conditions
KMV – Kealhofer Merton Vasicek
KS – Kolmogorov-Smirnov
k-NN – k-nearest neighbours
L-BFGS – Limited-Memory Broyden-Fletcher-Goldfarb-Shanno algorithm
LTCM – Long-Term Capital Management
LDA – linear discriminant analysis
LDP – low-default portfolio
LGD – loss given default
LR – logistic regression
LT – long-term
MAR – missing at random
MDA – multiple discriminant analysis
MDL – minimum description length
ML – machine learning
MNAR – missing not at random
MOS – majority oversampling
MUS – majority undersampling
NFL – no free lunch
PCA – principal component analysis
PCC – percentage correctly classified
PD – probability of default
PwC – PricewaterhouseCoopers
RBF – radial basis function
ReLU – rectified linear unit

RF – random forest
RFE – recursive feature elimination
ROA – return on assets
ROC – receiver operating characteristic
ROE – return on equity
RR – recovery rate
RS – Republic of Slovenia
SBITOP – Ljubljana Stock Exchange Market Index
SKD – Slovene Nomenclature of Economic Activities
SMEs – small and medium-sized enterprises
SMO – Platt's sequential minimal optimization
SMOTE – synthetic minority oversampling technique
SMOTENN - synthetic minority oversampling technique using edited nearest neighbours
ST – short-term
SURS – Statistical Office of the Republic of Slovenia
SVMs – support vector machines
TN – true negatives
TP – true positives
t-SNE – t-distributed stochastic neighbouring embedding
UK – United Kingdom
VIF – variance inflation factor
WCA – working capital
ZFPPIPP – Financial Operations, Insolvency Proceedings, and Compulsory Dissolution Act
ZGD – Companies Act

INTRODUCTION

The last economic crisis that started in 2008 in the United States and spread throughout the world has once again proven the importance of effective risk management across enterprises. Nowadays, stakeholders are aware that managing risk exposures across all parts of their organization is essential in order to succeed in the competitive business environment. Financial risk factors businesses face are broadly grouped into the market and credit risk. One aspect of credit risk is particularly important in the prolonged periods of economic distress – as a member of the Turnaround Management Association UK asserted: "As businesses struggle to survive into 2010, they are likely to put increasing pressures on their suppliers. Payments will be withheld for as long as possible. If and when a company fails, it is likely that other businesses it owes money to will get little or nothing in return. Unfortunately, the knock-on effect will be that other firms will also be starved of cash and more will find themselves under financial pressure" (Jackson & Wood, 2013, pp. 183-184). The type of credit risk mentioned in the preceding commentary is usually a consequence of trade credit arrangements that are extended to customers buying goods and services. The extension of trade credit inherently leads to the possibility of default on deferred payments. When granting trade credit firms must therefore manage this risk exposure by analysing the creditworthiness of their customers in order to distinguish between the ones who will pay and those who will not. The methodology of quantitatively assessing creditworthiness is known under the name of quantitative credit scoring (hereinafter: CS), which is a part of wider empirical credit risk modelling (hereinafter: ECRM) field. Credit scoring provides a basis for the development of effective credit risk mitigation strategies that enable insurance against the potential bad trade debts and so guarantee firm's liquidity. To measure the default risk involved by sales on credit, customers are assigned to risk classes based on their individual propensities to default on payment. As discussed later we use an approach known as machine learning (hereinafter: ML) in the computer science literature, which refers to a set of methods specifically designed to tackle computationally intensive pattern recognition tasks in large datasets. These methods are ideally suited for customer credit risk analytics because of the large sample sizes and the complexity of the possible relationships between payment behaviour and individual characteristics.

Hence, it is the **purpose of this thesis** to employ quantitative CS methodology to calculate the transaction-based probability of payment default for a range of corporate customers (i.e. B2B customer segment). More specifically, we carry out a comprehensive comparison of the traditional statistical and contemporary ML classification methods. The fundamental catalyst for this analysis is the fact that ML methods have been recognized throughout the academic literature to outperform statistical ones. Nevertheless, they are still not widely used in the industry due to their perceived complexity. Consequently, logistic regression (hereinafter: LR) remains the most popular method applied by practitioners (Crone & Finlay, 2012). A recent study by Lessmann, Baesens, Seow, and Thomas (2015) however reveals that the cost reduction of employing ML methods relative to LR amounts to around 5.0% of total losses. Similarly, Khandani, Kim, and Lo (2010) estimate the practical value of more

accurate predictions by summing the cost savings from trade credit reductions to high-risk customers and the lost revenues from the so-called "false positives" (i.e. the number of non-defaulted cases incorrectly predicted as defaulted customers). Under a conservative set of assumptions, they estimate the potential net benefits of these forecasts to be 6.0-25.0% of total losses. Although the link between predictive accuracy and firm's bottom-line is not straightforward, the fact provides some evidence that more accurate models facilitate sizeable financial returns, which serves as a motivation for our research.

The **aim of this thesis** is to find the best performing model on a given problem domain, i.e. payment default probability of B2B customer segment. Theoretical part of the master's thesis aims to present practical issues in quantitative CS as well as recent advancements in the ECRM field. These concern the following two dimensions: (i) novel ML classification algorithms, and (ii) improved performance measures to assess and compare different models. Empirical section of the thesis then tries to evaluate the impact of various CS methodologies on model's predictive performance with the focus of identifying the best performing model.

Based on the master's thesis aims we define **research questions** along the following lines:

(i) Which classification method provides the best generalization ability on the unseen data (i.e. prediction performance) – a comparison of the most commonly applied statistical (e.g. logistic regression) and ML methods (e.g. k-nearest neighbours (hereinafter: k-NN), decision trees (hereinafter: DTs), support vector machines (hereinafter: SVMs), artificial neural networks (hereinafter: ANNs)).

(ii) Multiple explanations principle in ML suggests that learning more hypothesis leads to higher predictive performance – do ensemble learning methods (e.g. random forest (hereinafter: RF), AdaBoost and gradient boosting algorithms) improve model generalization ability on the unseen data?

(iii) What is the effect of employing sampling techniques, i.e. does oversampling outperform undersampling in the context of generalization ability, i.e. predictive performance on the unseen data? Is SMOTEEN algorithm superior to the simple resampling techniques?

(iv) Which type of dimensionality reduction (i.e. feature selection/extraction method) performs best given our problem domain – comparison of embedded, filter, and wrapper methods in selecting the optimal subset of independent variables.

(v) Does the inclusion of the macroeconomic variables into the model (in a pooled cross-sectional manner) improve its generalization ability?

In order to satisfy the research questions of the thesis quantitative research is carried out using traditional statistical and ML methods. Firstly, **theoretical part** provides insights into the credit risk management and ML paradigm using descriptive research method. Following

is a systematic overview of techniques applied within the ECRM. We use comparative method to identify the best practices in the academic literature, which serves as a starting point for our own empirical analysis. Secondly, **empirical part** of the master's thesis uses ML workflow as defined in Mirjalili and Raschka (2017, pp. 11-13) and Thomas, Crook, and Edelman (2017) to empirically evaluate customers' creditworthiness. For the purpose of developing quantitative CS model, dataset comprising of business customers and their respective default status is used. Financial data is collected from the Slovenian Business Register managed by Agency of the RS for Public Legal Records and Related Services (hereinafter: AJPES), whereas the dataset on firm insolvency proceedings is obtained from the Information Centre of the Supreme Court of the Republic of Slovenia. Macroeconomic data was provided by the Statistical Office of the Republic of Slovenia (hereinafter: SURS) and global financial portal Investing.com.

The **structure of the master's thesis** is as follows. We begin our appraisal by introducing the theoretical background of the study, i.e. credit risk management and ML approach; this provides a basis for our further discussion concerning credit risk modelling. Firstly, trade credit arrangements between business entities are discussed. Furthermore, we establish the need for an effective management of the (trade) credit risk exposure and shortly describe various risk mitigation strategies. Then we turn our attention to the three components of credit risk and focus on the probability of default (hereinafter: PD); an important input into designing optimal risk management response. Next, we examine the practice of quantitative CS methodology that is typically used for modelling PDs. Secondly, our discussion continues with a high-level introduction into ML paradigm – we define general concepts and terminology as well as the benefits this novel predictive modelling approach delivers. Thirdly, we conclude the first section with a literature review aiming to present a holistic view of the state-of-the-art techniques used in credit risk modelling. In section 2 we discuss the three factors underlying the classification mapping function; firstly, the definition of default is considered. Then we look into the possible independent variables that can provide insightful information for the task at hand. Lastly, the representation (i.e. classification methods), evaluation (i.e. performance measures) and optimization components of classification algorithms are presented. Section 3 describes our experimental setup and presents the empirical results along with a concise discussion for each research question. The final section concludes, outlines the limitations of the study, and suggests some guidelines for the future research.

1 THEORETICAL BACKGROUND TO THE STUDY

This chapter starts with a gentle introduction into our research topic, namely quantitative CS and ML, as well as its importance in the context of enterprise risk management. Motivations for our study are discussed along with the practical aspects of dealing with (trade) credit risk and credit risk mitigation. Then, the chapter reports on the significance of ML, a field that has gained on its popularity in the last two decades. Finally, we conclude with a high-level literature review of the ECRM field.

1.1 Credit Risk in the Context of Risk Management

"No risk, no fun" says a well-known German proverb indicating to an inverse relationship between risk and reward we commonly face in our lives. Translating this into an economic sphere implies a risk-return trade-off principle stating that each economic entity must undertake some risks in the pursuit of value-enhancing business goals (i.e. return). Strategic-minded enterprises do not strive to simply minimize risks. Rather, they employ a modern view of managing risk exposures across all parts of their organization so that they incur "just enough of the right kinds of risk" (Curtis & Carey, 2012, p. 1). The winner in this race is the one capable of effective control of the risks in a continuously changing environment. Risk management is an integral part of the general enterprise management system for assessing the significance of each risk in achieving overall strategic goals (Rogachev, 2008).

Before delving into the potential benefits that the risk management offers, we should look into some risk-related terminology in order to establish the basis for our further discussion. As a Chicago economist Frank Knight observed in his classical economic text "Risk, Uncertainty, and Profit" not all future situations can be thought of as risky; some aspects of the future remain uncertain and no amount of complex modelling will enable us to fully disentangle this **uncertainty**, i.e. the variability that cannot be quantified due to the impossibility of determining possible outcomes. Given the fact focus should thus be on managing statistically measurable risk while having in mind the extent of our ignorance with respect to the uncertainty (Langlois & Cosgel, 1993). As already stated, **risk** arises from the "quantifiable" uncertainty and can be best described as "a chance of an event's occurrence in terms of its likelihood (i.e. the probability of different outcomes) as well as the negative impact it usually carries (i.e. evaluation of the outcomes)". **Risk management** on the other hand represents "the act of identifying, measuring and, reacting to those risks" (Hay-Gibson, 2008, p. 1). More recently the term **enterprise risk management** is being used to additionally recognize "the importance of prioritizing and managing the full spectrum of risks as an interrelated risk portfolio and to embed risk management in all critical decision-making processes" (Saunders & Millon Cornett, 2018, p. 20). Although we can trace the origin of risk awareness back to the ancient Mesopotamia, modern risk management only emerged as a formal discipline in the 1950s and has since been a bumpy affair; especially so over the last two decades with the rise of lucrative derivatives market as well as some extraordinary failures of the risk management related to the LTCM failure, Enron scandal, Lehman Brothers collapse, etc. Some financial commentators even go so far as to argue that the risk sharing strategies enabled by the sophisticated financial engineering played a significant role in covering up the true condition of poorly run companies and thus advanced the final failures (Curtis & Carey, 2012; Hay-Gibson, 2008). Nevertheless, according to Hoyt and Liebenberg's (2011) paper that tries to document the value relevance of enterprise risk management shows its implementation to be positively associated with the firm value measured as Tobin's Q. The main benefits of risk management can be summarized along the following lines (Berk Skok, 2016; Curtis & Carey, 2012): it (i) reduces the PD and cost of financial distress, (ii) provides additional insights and support to the management, (iii)

enables more realistic performance evaluation (e.g. performance per risk exposure) which improves resource allocation, (iv) boosts firms competitive advantages, (v) enhances customer service, etc. Lastly, in order to realize the potential value enhancing benefits firms must react to the risks at hand either by avoiding, transferring, mitigating and/or accepting them. So far, we have been discussing risk as a general phenomenon. Understanding the **typology of risk exposures** however is equally important as each category demands a different set of risk management skills as well as tools. Risk factors can be broadly grouped into the following categories: market risk, credit risk, liquidity risk, operational risk, legal and regulatory risk, business risk, strategic risk, and reputation risk. The first two risk factors fall under the scope of the financial risks and can be further subdivided into equity price risk, interest-rate risk, FX risk, and commodity price risk in the case of market risk, and transaction risk and portfolio concentration risk in the case of credit risk (Curtis & Carey, 2012; Saunders & Millon Cornett, 2018).

1.1.1 Trade Credit Arrangements as a Source of Credit Risk

As discussed in the introduction our aim is to employ the quantitative CS methodology to calculate the PDs for the range of individual business customers (i.e. B2B customer segment). Hence, our focus will be on measuring the credit risk usually referred to as the default risk in the context of non-financial institutions; in this study credit and default risk are used interchangeably as is often done in practice. **Default/credit risk** is most generally defined as "the risk that a counterparty does not honour his/her obligations" (Alexander & Sheedy, 2004, p. 211). The obligation may either be in the form of a payment or physical asset. The former definition suggests the risk of payment default is inherently a part of every transaction. In our case credit risk originates from the trade credit arrangements usually extended to the B2B customers (Alexander & Sheedy, 2004). Slovene Accounting Standards define **trade credit** as "a sale of goods or services on open account without an immediate payment" (Slovene Accounting Standards, 2016). For the supplier (i.e. seller) this represents an investment in accounts receivable, while for the buyer it is a source of financing recorded as accounts payable on the balance sheet (Garcia-Teruel & Martinez-Solano, 2010). Suppliers thus act as financial intermediaries by providing finance to their partners comprising of both time differential between a sale and a payment, as well as the proportional discounts for payments in bulk carried out before the due date. McGuinness, Hogan, and Powell (2018) argue that trade credit provides an important alternative short-term financing source to bank loans and is estimated to present approximately 14% of total assets among the European SMEs. It has been shown that such payment arrangement played a significant role during the post-crisis years, suggesting that it enabled many financially constrained firms to survive. There exists a whole body of literature with respect to the motives for extending trade credits. A nice overview of the topic is provided in Garcia-Teruel and Martinez-Solano (2010) where various financial, commercial, and operational motives are discussed. The most plausible reasons why more liquid firms use trade credit arrangements to help financially constrained ones are as follows (Garcia-Teruel & Martinez-Solano, 2010;

McGuinness, Hogan & Powell, 2018; Petersen & Rajan, 1997; Santos & Silva, 2014; Woodruff, 2019):

(i) **long lasting relationships**: large suppliers may provide stability to the customers in time of liquidity shocks to sustain sales and more importantly to support future business cooperation. They often have an advantage over banks in assessing customer's creditworthiness as they can acquire information more easily. Additionally, they possess higher bargaining power in forcing the repayment (e.g. cutting off future supplies) and an option to salvage the lost value from existing assets (i.e. the supplied goods);
(ii) **price discrimination**: typically, suppliers are motivated to increase sales, so they offer favourable trade credit arrangements to gain a competitive advantage over their competitors. This can be effectively thought of as an indirect price discrimination that is particularly useful when a direct pricing system cannot be legally established. Creditworthy customers that find trade credit terms overpriced are prone to repay it as soon as possible at a discounted price while risky customers use the provided credit as it may still be cheaper than other financing alternatives;
(iii) **product/service quality assurance**: the use of trade credit allows product quality verification by customers due to the deferred payment arrangement and thus represents an implicit quality guarantee instrument. Longer payment deadlines transmit the quality information. This is especially popular among smaller and younger firms since they do not enjoy an established market position.
(iv) **lower transaction costs**: suppliers and their customers can reduce their transaction costs through the use of trade credit. Buyers of goods and services may decide to aggregate multiple payments into one and thus reduce transaction costs whereas suppliers may more accurately predict their future incoming receipts (lower uncertainty and thereby increased operational flexibility). Additionally, suppliers may decrease their production cost by using trade credit as a trade policy instrument; e.g. businesses with seasonal activity may use it to smoothen production cycles and lower their storage costs.

Inevitably, the extension of trade credit leads to the possibility of default on deferred payments. When granting trade credit firms must therefore analyse the creditworthiness of their customers in order to distinguish between the ones who will pay and those who will not. Sources of such information can come from financial statements, credit reports, customer's payment history with the firm, etc. (CFI, 2019). The methodology of quantitatively assessing the customer's creditworthiness is known under the name of financial distress prediction (hereinafter: FDP) or alternatively quantitative CS. We discuss credit risk models in chapter 1.2. The rest of this section deals with the risk mitigation (i.e. insurance) part of credit risk management.

1.1.2 Trade Credit Risk Mitigation Strategies

Credit risk models discussed later on provide a basis for the development of effective **trade credit risk mitigation strategies** that provide insurance for the potential bad trade debts in order to guarantee firm's liquidity. In the case the share of trade debts gone bad increases

too much the firm supplying trade credit can face short-term liquidity difficulties that may result in insolvency over the long-term horizon given no risk management policies are implemented. Hence, due to the significant portion of receivables companies hold on their balance sheets it is crucial they consider at least some form of trade credit insurance. Optimal trade credit risk mitigation policy takes into consideration both the maximization of firm's sales as well as the minimization of loss originating from payment defaults and involves the cooperation between sales, IT, and financial departments. Firms extending trade credits must determine the acceptable level of credit exposure to their customers. This can be effectively done using **credit limits**[1] that are set for each respective customer based on their risk-adjusted future performance calculated using customer's creditworthiness evaluation (Bazzi & Hasna, 2015; Khandani, Kim & Lo, 2010). Bazzi and Hasna (2015) provide a comprehensive overview of experts' methods as well as modelling approaches of setting credit limits. The discussed methods depend mainly on the PD which is what we model here.

Apart from setting credit limits there exist other instruments to insure the kind of credit risk discussed herein. They provide additional control of risky trade debts and thus an increased likelihood of compensation in case of payment default. One example is taking out **trade credit insurance policy** provided by insurance companies. Such policies are flexible and allow policyholder to cover the entire portfolio (i.e. whole turnover cover) or just the key accounts against potential bad debts (Moorcraft, 2018). Another option to tailor credit risk exposure employed in the countries with well-functioning derivatives market is the use of **credit derivatives**[2]. The common types of credit derivatives are credit default swaps, collateralized debt obligations, total return swaps, credit linked notes, credit default swaptions, etc. Whereas the striking growth in the notional amounts of credit derivatives suggests their usefulness in risk management we have to keep in mind that they pose risk management challenges of their own, e.g. complex modelling risk, counterparty credit risk, and settlement risk (Gibson, 2007). The detailed discussion on implementation and advantages of trade debt insurance instruments is beyond the scope of this study. The Table 1 below nevertheless outlines some commonly used instruments for credit risk mitigation.

Table 1: Credit Risk Mitigation Instruments

Contractual Insurance	Unfunded Credit Protection	Funded Credit Protection	Payment Mechanisms
Contractual Penalty Damages	Guarantee Credit Derivatives	Financial Collateral Mortgage Lien	Promissory Note Letter of Credit Insurance Policy

Source: European Banking Authority (2018).

[1] Credit limit is defined as a threshold that a firm allows its customers to owe at any time.

[2] Credit derivatives are defined as instruments that "transfer credit risk related to an underlying entity from one party to another without transferring the actual underlying entity" (Chen, 2018).

1.2 The Three Basic Components of Credit Risk

Previous section described (trade) credit risk in the context of risk management establishing the need for quantitative customer's creditworthiness assessment in order to perform risk mitigation policies. This section layouts a useful framework to think about the credit risk, namely it discusses its three basic components with the focus on the PD that is the concern of this study. Central concept in measuring credit risk is the **probability of default** of a customer. It is defined as "the probability that the counterparty will fail to service its obligations" (Crosbie & Kocagil, 2003, p. 5). PD does not however represent a complete picture of the potential credit loss. Firms seek to measure two additional components characterizing the extent of default loss. Firstly, the magnitude of likely loss on the exposure termed as **loss given default**[3] (hereinafter: LGD) and expressed as a percentage of the overall exposure, and secondly, the amount to which a firm is exposed at the time of default known as **exposure at default** (hereinafter: EAD) (Alexander & Sheedy, 2004). These three components[4] provide a measure of individual default loss (L_i):

$$L_i = PD_i * EAD_i * LGD_i \quad (1)$$

The subscript i reminds us of the fact that we are discussing the transaction risk. Alternatively, risk managers may also be interested in measuring and controlling the risk of the whole portfolio of trade debts known as the portfolio concentration risk. To derive it we would need to sum up the individual default losses in equation (1) over all customers while also accounting for the interdependencies among them (i.e. default correlation structure). The quality of the entire portfolio is evaluated via portfolio credit risk models (Elizondo Flores, Lemus Basualdo & Quintana Sordo, 2010). We are only considering the transaction-based credit risk in our study. Additionally, while each three components (i.e. PD, EAD, LGD) are critical to the management of credit risk, this study focuses on the estimation of PD that is the most important input into designing the optimal trade credit risk mitigation strategies. The rest of the section thus outlines an approach that can be taken for PD modelling referred to as quantitative CS.

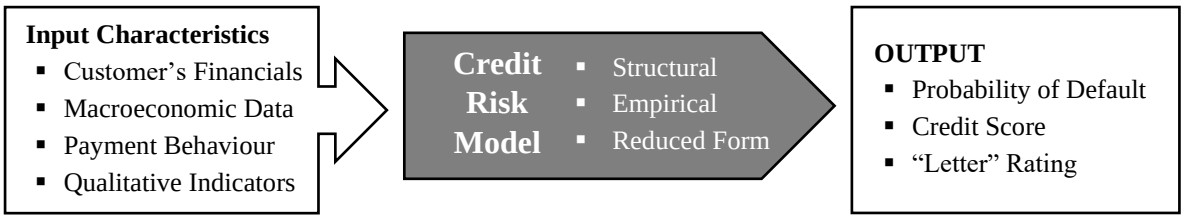
Prior to default, there is no way to discriminate unambiguously between the defaulted vs. non-defaulted firms. At best we can make some probabilistic assessments of the likelihood of default (i.e. estimate the PD) in order to implement risk mitigation measures proportional to the individual customer's PD. The typical firm has a PD of around 2% in any year. However, there is a significant variation in PDs across firms (Crosbie & Kocagil, 2003). This variation can be modelled using **credit risk models** that output the credit rating of an individual obligor which represents an accurate classification of firm's customers according

[3] LGD is often less than one, which reflects the fact that some proportion of EAD may be recovered in the bankruptcy proceedings – this amount is commonly referred to as the recovery rate (hereinafter: RR) and is calculated as follows: $RR_i = 1 - LGD_i$ (Alexander & Sheedy, 2004).

[4] Appendix 3 visually summarizes the three basic components of credit risk along with the modelling approaches.

to their creditworthiness by specifying their PD, credit score or "letter" rating. The structure of credit risk models is depicted in Figure 1.

Figure 1: Credit Risk Model Structure



Source: Adapted from Alexander and Sheedy (2004).

As can be seen in the figure above there are three approaches to model the relationship between realized defaults and their characteristics prior to default, namely structural, empirical, and reduced form approach (Chacko, Sjöman, Motohashi & Dessain, 2016).

(i) **Structural models** try to quantify the credit risk via replication of the firm and its capital structure using option pricing approach known under the name Black-Scholes-Merton model extended by the famous KMV model employed by Moody's. In such setting firm defaults on its obligations when the value of its assets reaches a sufficiently low level compared to its liabilities (debt). As the value of the firm (i.e. its assets) is hard to observe models use the information from the price of equity to determine the default risk (Kealhofer, 2003; Mosconi, 2015). Modelling firm structure is nevertheless difficult to implement in reality due to data availability issues and is usually only performed for large publicly listed corporations.

(ii) **Empirical models** evaluate historical information (i.e. firm's characteristics) of the defaulted companies and non-defaulted companies. In order to derive the classification rules from historical data various statistical and ML methods discussed later on are used. Empirical credit risk models come in two flavors: (i) **FDP models** with a focus on predicting the risk of companies going bankrupt, and (ii) **quantitative CS models** that are usually employed when making credit-related decisions and focus on the notion of payment default (Alexander & Sheedy, 2004).

(iii) **Reduced form models** associate the likelihood of default to an external signal and describe it as an abrupt event. The timing of the default is assumed to happen "by surprise" and the probability of such surprise during a given period of time is modelled using a Poisson type of distribution that outputs the event's arrival rate which is referred to as the default intensity in terms of credit risk modelling. Reduced form models thus assume that the default information lies outside the firm in contrast to the structural models. CreditRisk$^+$ published by Credit Suisse is an example where this actuarial science framework is used (Jarrow & Protter, 2004).

The rest of this chapter deals with the empirical credit risk models, or more specifically with quantitative CS. The aim is to find the most accurate model on a given domain as this enables the development of efficient credit risk mitigation strategies. There is a long tradition of FDP

and CS literature. One of the earliest empirical credit risk model goes back to Altman (1968), which was then extended by Altman, Haldeman, and Narayanan (1977). It uses discriminant analysis to derive a score to rank obligors. The first model that estimated default probabilities was employed by Ohlson (1980). Since, there has been an explosion in the use of various statistical as well as artificial intelligence (hereinafter: AI) methods. We outline the past developments and contemporary research studies in the literature review chapter. Next section presents the practical issues of quantitative CS and thus provides the basis for the empirical analysis.

1.3 The Practice of Quantitative Credit Scoring

So far, we have discussed credit risk in the context of risk management as well as various approaches to modelling credit risk (i.e. PD component). The focus of this study is to evaluate transaction-based credit risk arising from granting trade credits using quantitative CS methodology. We have briefly mentioned the approach in previous section. However, we have not formally defined it. **Quantitative credit scoring** is essentially "an objective way of classifying obligors into two groups – those who will default and those who will not default – using the characteristics of the obligor" (Thomas, Crook & Edelman, 2017, p. 3). Credit scoring methodology evolved from classification techniques[5] used in statistics as there was a need for automatization of credit-related decisions. Myers and Forgy (2012) report that the default rates dropped by 50% or more in some cases with the ascent of CS models. The philosophy underlying CS is pragmatism and empiricism. Hence the aim is to predict the risk in granting credit to a particular customer, whereas the transparency of the predictive model is not necessary (pragmatism). Empiricism on the other hand implies that any characteristic which aids prediction should be used. The advantages of CS go along the following lines: (i) ability to maximize risk/reward trade-off, (ii) improved assessment of credit risk and consistency across customers, (iii) increased quality of the overall portfolio, etc. There are some authors who criticize the underlying methodology saying that CS does not provide any explanation of the links between the characteristics (i.e. independent variables) and PD. Additionally, sample bias is often present in the learning dataset which hinders model generalization to the unseen data. Other problems highlighted in the literature are the problem of low-default portfolios (hereinafter: LDP), use of biased financial annual account information, etc.[6] (Kennedy, 2013; Kinda & Achonu, 2012; Schreiner, 2004; Thomas, Crook & Edelman, 2017).

1.3.1 Application vs. Behavioural Credit Scoring

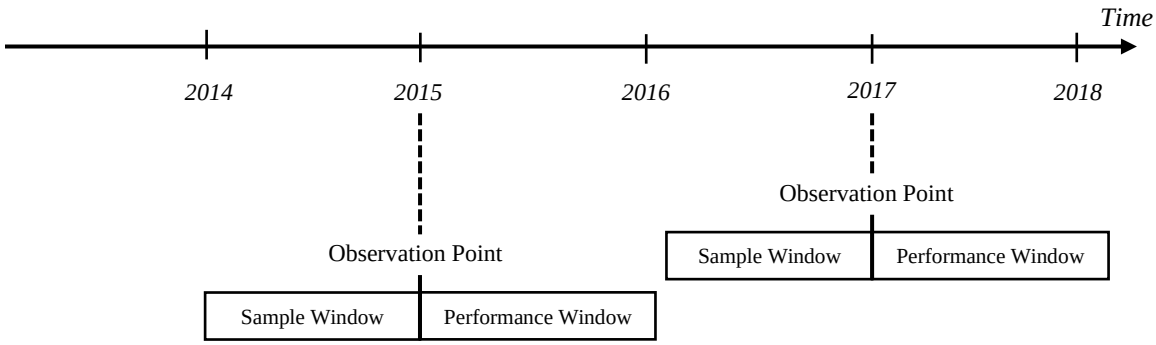
Based on the task and data used CS can be divided into two subgroups: (i) application scoring, and (ii) behavioural scoring. **Application scoring** tries to capture the credit risk associated with new applicants (i.e. customers), whereas the **behavioural scoring** focuses

[5] We define the set of statistical and ML classification techniques in chapter 2.3; they are used in expert systems that automate decision-making processes.

[6] Detailed discussion on the topic related problems may be found in Appendix 2 along with the possible solutions.

on assessing the ongoing risk of existing accounts; i.e. the probability of ongoing negative behaviour. One major difference between the two CS types is the data used in the development phase, e.g. in the application scoring phase we usually use the static data gathered at the arrival of new customer because there is usually no prior internal information. In contrast, most of the data used in behavioural scoring is performance based and comes from the past customer behaviour (Siddiqi, 2017). In order to perform the analysis two parameters must be set to construct the development sample: (i) **sample window** which refers to the "time frame from which selected customers' characteristics are gathered", and (ii) **performance window** which denotes the "time window where the performance of given customers is monitored to assign class (i.e. defaulted vs. non-defaulted) (Siddiqi, 2017, pp. 80-81). The point separating the two windows is known as the **observation point** that defines the existing customers used for the development of credit risk model. There exist several approaches to determining the three parameters. Observation point is usually set into the not so distant past in order to have a representative sample that incorporates current business conditions. In case the absolute number of defaulted instances in the dataset is too small to develop a robust model, the so-called **stacked sampling approach** where we define multiple observation points is taken. This approach also presents a way of including yearly macroeconomic variables in a pooled cross-sectional manner and thus enables the creation of a through the cycle credit risk model that accounts for seasonality, economic cycles and other abnormal periods. In the Figure 2 below you may find the graphical representation depicting the described approach. Performance window timeframe for application models is usually set to 6 or 12 months depending on the desired prediction horizon. Sample window on the other hand depends on the characteristics available and typically incorporates historical data for past 12 months (Siddiqi, 2017; Thomas, Crook & Edelman, 2017).

Figure 2: Dataset Construction Using Stacked Sampling Approach



Source: Own work.

The rest of this chapter deals with the issues concerning application CS such as data requirements, definition of default, etc. Before turning our attention to specific CS issues there is one more thing left to discuss for the sake of completeness. Broadly speaking, there exist two approaches to CS: the methods that employ static characteristics (i.e. classification methods), and the methods that incorporate dynamic aspect into the estimation of PD; the latter are referred to as **Markov chain probability models** and **survival analysis**. The idea

of the former is to identify different states the customer can be in and then estimate the set of transition probabilities among these states over given period of time. The latter focuses on the estimation of the duration until default (Thomas, Crook & Edelman, 2017). As it has been noted before, this study examines and employs statistical as well as ML classification methods, the dynamic approach is not considered here.

1.3.2 Definitions of Defaulted and Non-Defaulted Customers

When payment delinquency is in question then the task of CS model is to map the customers into two groups as already said, i.e. defaulted and non-defaulted with regards to their payment behaviour in the performance window. Members of the non-defaulted group are considered likely to repay their financial obligations, while members of the other group are in contrast considered as likely to default. The dependent variable in this setting is thus a dummy variable coded as one if the payment was late or zero otherwise. Exactly where the line between the two groups is drawn depends on the objective of a CS model as well as the company's views of success or failure (Kinda & Achonu, 2012). In the case of payment delinquency, the definition of default usually relies on the following considerations. First and foremost, the definition must be in line with **organizational objectives**, for example some companies would like to simply reduce the number of payment defaults in their portfolio, while others might be focused on increasing their profitability, market share, etc. Some are trying to apply new methods to achieve better predictive power, while others are building CS models to comply with the regulation. Specific objectives therefore require more/less stringent definitions. Secondly, the default definition must be easily **interpretable and trackable**. This condition does not only improve the development phase, but also makes for easier management, and decision making in the post-implementation phase. Last but not least, there might be **regulatory requirements** that govern the payment default definition such as Basel Accords in financial industry. In some cases, default definition is limited by the lack and/or length of the data the company gathers about its customers (Siddiqi, 2017).

Defaulted customers are usually the ones who fall behind on payment for 60 or 90 days, which is referred to as 60 or 90 DPD, i.e. days past due. A more stringent default definition would be 120 DPD or bankruptcy status. It provides a more extreme differentiation but may lead to a problem of low-default portfolio[7]. Once the defaulted customers are defined the rest can be categorized as non-defaulted. Some practitioners additionally specify the indeterminate group for customers that are in the "grey zone", where the classification is less obvious. They usually have some mild delinquency but a high cure rate. It is suggested that the proportion of indeterminates does not exceed 20% in the case of application scoring. Most common technique to deal with this "grey zone" customers is to discard them and build the model on the remaining instances (Anderson, 2007; Siddiqi, 2017). According to Anderson (2007) there are two possibilities in how to observe payment behaviour in the defined performance window: (i) a **current-status definition** "focuses upon the customer

[7] For more information on the problem of LDP please refer to Appendix 2 where we discuss relevant topic related problems. Furthermore, based on literature review we summarize possible ways to solve the issues.

payment status at the end of the period", while (ii) a **worst-ever definition** "uses the worst status over the performance window" (Anderson, 2007, p. 340). Both definitions try to specify something near the "point of no return", where the chances that the customer pays are low. In general, for credit risk assessments, 60 DPD dominates current-status definition, while 90 DPD dominates for worst-ever (Anderson, 2007). Siddiqi (2017) confirms this and further suggests the use of **roll rate analysis**[8] to determine the appropriateness of the default definition for specific project. Now that we have discussed the dimensions to consider when choosing our dependent or target variable, we should also clarify how to determine the set of potential independent variables.

1.3.3 Data Quality and Quantity Issues

In order to carry out the quantitative CS analysis in practice we need data that can discriminate between the two groups of customers. For the case of application CS historical data such as past financial ratios is used. Here the underlying assumption goes as follows: "future performance is reflected by past performance" (Siddiqi, 2017, p. 80). Since the global economic crisis in 2008 it has been recognized that the macroeconomic variables hold additional power when modelling the PD. Moreover Altman, Sabato, and Wilson (2008) highlighted the utility of including qualitative variables that describe firm scale, diversification, corporate governance, etc. They showed that their inclusion leads to a significant improvement in predictive accuracy of the model. Once we determine the types of variables (e.g. financial ratios, payment behaviour, macroeconomic, and qualitative variables) for the construction of the database we have to consider the issue of data availability in the context of quality and quantity. **Data quality** refers to the three dimensions, namely data accuracy, completeness and consistency. Data accuracy relates to "the degree of precision of measurements of a characteristic to its true value"; in practice poor data accuracy is usually a result of user input errors (also manipulations) and/or out-of-date characteristics. Data completeness deals with "the extent to which values are missing", whereas data consistency has to do with "the lack of standardisation among various data sources, which may lead to conflicting characteristics" (Kennedy, 2013, p. 70). There exist many techniques to handle poor datasets – they are generally known as **data pre-processing techniques** (hereinafter: DPP), e.g. data cleaning, instance/feature selection, normalization, transformation procedures, etc. These techniques enable us to prepare final training dataset and thus mitigate the "garbage in, garbage out" problem.

To ensure the development of a robust CS model a sufficient number of customer data is required, which is referred to as **data quantity**. Here the quantity in absolute (i.e. total number of data instances) as well as relative (i.e. the ratio between the defaulted and non-defaulted group) terms is important. According to Crone and Finlay's (2012) paper that looks

[8] Roll rate analysis helps us find some delinquency status beyond which the chances of recovery are very low. It shows the percentage of obligors who flow from a specific delinquency bucket (e.g. 30 DPD) into the next stage of delinquency status (e.g. 60 DPD) during a period of time (Zhe, 2017).

into the effect of both sample size and the ratio between the groups, it seems that for LR a sample of about 10,000 defaults results in the maximum Gini coefficient[9]. They add however, that more complex ML methods such as neural networks might require more to obtain the optimal performance. In this sense "more data is better". Usually there are no issues with respect to the absolute number of non-defaulted customers in the sample given the fact that the datasets are normally dominated by them. Some authors suggest that having 100,000 non-defaulted customers provide more than enough information. There is however another point, that we have already mentioned above, namely the relative odds between the two groups. The question at hand is whether we should keep the population odds (i.e. thus having imbalanced dataset) or adjust the sample to achieve the relative proportion closer to 50:50 using undersampling and/or oversampling approach. Again Crone, and Finlay (2012) experimental study provides some insights into the matter; oversampling significantly increases the accuracy relative to undersampling, especially so in the case of ML algorithms.

Given different types of data mentioned above the number of derived features (i.e. independent variables or characteristics) can grow quite fast. There are too many to deal with in detail, and so practitioners often use various procedures to identify the most powerful feature subset (Thomas, Crook & Edelman, 2017). We discuss and apply these in the empirical part of the study. Beforehand however it may be useful to screen all possible independent variables along the following criteria (Thomas, Crook & Edelman, 2017):

(i) **legal**: characteristics that are illegal according to the law should be removed. However, we can use them to assess potential benefits of inclusion and propose a change in regulation;
(ii) **intuitive**: we should use the characteristics that make at least some sort of intuitive sense;
(iii) **stable**: in order to develop a stable model, we should consider using only stable characteristics. One way of checking the robustness of a characteristic is to perform distribution analysis over a given time period;
(iv) **simple to obtain**: the cost of collecting data for a specific characteristic must be in line with the benefits it provides to the model performance. Therefore, we want to have characteristics that are obtained incurring relatively low costs;
(v) **verifiable and unambiguous**: we should try to use characteristics that are easily verifiable, as well as clearly defined - this leads to more accurate data;
(vi) **future availability**: we must ensure that every characteristic considered in the model will also be available in the future. Additionally, definitions of these items must be consistent;
(vii) **predictive**: at the end we should evaluate each characteristic according to its predictive power. In practice information value is calculated for every characteristic. Then ones with low power are eliminated. We must note at this point that this is a univariate

[9] Gini coefficient is a measure of CS model performance which summarizes the performance over all cut-off scores. For more see Thomas, Crook, and Edelman (2017, p. 190).

analysis, which does not take into account the possible partial associations and interactions among the input characteristics. Usually, other (more advanced) dimensionality reduction techniques are employed as discussed later in the empirical part of the study.

We can see that the feature selection process is not a purely statistical exercise; it also needs a solid business insight. Therefore, it must be carried out collaboratively among various entities/firm departments. So far, we have discussed the practical aspects of quantitative CS. The following chapter reports on the significance of ML in general and establishes a bridge between commonly used econometric toolbox and this rather new paradigm in the finance field.

1.4 Machine Learning Approach

Machines are increasingly being used for "intelligent" tasks that were not so long ago exclusively in the human domain (i.e. tasks where we cannot articulate our own knowledge explicitly such as speech recognition, trading bots, face recognition, fraud detection, etc.). They have been given the ability to improve the performance without humans telling them exactly how to accomplish the assignments they are given. Nowadays machines can achieve superhuman results in the disciplines ranging from engineering, medicine, biology, finance, etc. (Brynjolfsson & McAfee, 2017a). There are three important **factors contributing to the success of the machines** at discovering complex patterns that are not specifically pre-programmed, namely (i) ascent of big data, (ii) improved self-learning algorithms and (iii) cheaper computing power. According to an article in the Harvard Business Review data availability has increased as much as 1,000-fold, algorithm efficiency 10 to 100-fold, and computing power by at least 100-fold compared to two decades ago (Brynjolfsson & McAfee, 2017b). These improvements go hand in hand with one another in enabling professionals to develop, test and use their solutions in a time- and cost-efficient manner, and employ them in everyday applications. This in turn adds real business value to the field and therefore boosts capital investment that further propagates the advancements. Consequently, terms like artificial intelligence, machine learning and deep learning (hereinafter: DL) come up in countless articles inside as well as outside the technology sector.

It is widely accepted by the business community that AI falls under the so-called **general-purpose technologies** (e.g. steam engine, electricity, internal combustion engine) that are fundamental for the future economic growth. Thus, AI is said to have a transformational impact on the whole economy. For example, the technology is being used by a company Deep Instinct to detect malware, by Amazon to optimize inventory and product recommendations, by PayPal to prevent money laundering, by Affectiva to recognize emotions of focus groups, an increasing number of financial companies are using it for trade executions on Wall Street, as well as for automated credit decisions. Healthcare industry is using it to look for early signs of lung cancer, and to predict whether a patient is susceptible to heart attacks and strokes (Brynjolfsson & McAfee, 2017a). These are just a handful of

most visible real-world applications. Based on the complex multi-stage modelling framework employed by PwC industry experts the cumulative expected marginal economic impact (as measured by GDP, i.e. gross domestic product) of the yet-to-be-implemented AI over the period 2017-2030 is estimated to be at around 14%, ceteris paribus. Similarly, McKinsey Global Institute estimates the AI potential while also considering negative externalities (i.e. transition and implementation costs) to about 16% or $13 trillion higher cumulative GDP by 2030. These figures will probably not materialize immediately; the adoption will most likely have a S-curve pattern with gradual acceleration of benefits over time. The impact will be driven via two fundamental transmission channels: (i) supply-side productivity gains (through process automation and augmentation), and (ii) increased consumer demand (due to product personalization, time saved, and higher product quality). The key factors for harnessing the full AI potential is in the willingness to invest, as well as in other barriers to adoption, e.g. regulation, human capital quality, consumer trust (Bughin, Seong, Manyika, Chui & Joshi, 2018; Cameron, Gillham, Rao & Verweij, 2018). AI undoubtedly presents an exciting source of wealth for the future, assuming the economic agents take strategic initiatives to capitalise on it. Additionally, it needs to be managed proactively otherwise the economic gaps among countries, companies, and workers may widened even more.

As stated in the previous paragraph, despite being used in numerous companies around the globe, most promises of the AI related technologies are yet to be tapped. According to McKinsey Global Institute most of the progress has been done in the "**narrow AI**", where ML algorithms are being employed to solve specific tasks, whereas the advancements in "human-level general intelligence[10]" still face significant challenges. The focus of this study is the former, so we focus on the specific challenges and risks facing narrow AI. ML, and even more so DL techniques as seen later are generally referred to as "black-box" methods due to their inability to assign weights to specific factors and consequently inability to interpret predicted outcomes, which is especially important in societal application such as criminal justice applications or financial lending (i.e. CS). This creates three risks. There are some concerns related to the way algorithms use provided data to learn, which can introduce new biases or propagates existing ones, derived from provided data. Secondly, unlike traditional expert systems that were built on explicit rules, ML systems are not guaranteed to work in all cases (lack of verifiability), which is a concern in mission-critical application such as controlling nuclear plant. A third risk has to do with difficulty of finding out why exactly errors were made, as the underlying predictive model is too complex. Furthermore, we can read about broader issues related to data privacy (i.e. use of personal information), cybersecurity, public opinion manipulation (e.g. influencing election results), etc. Nevertheless, the risks associated with narrow AI should not stop us from further advances.

[10] Human-level general intelligence tasks require machine to (i) reason, (ii) represent knowledge, (iii) plan, (iv) learn, (v) communicate in natural language, and (vi) integrate these skills towards given objectives, e.g. believable dialogue systems, human-level translation, natural-language comprehension, etc. (Goertzel & Pennachin, 2007).

After all we (i.e. humans) also have biases, make mistakes, cannot fully explain our decisions etc. It is critical that we embrace AI, while at the same time address possible misuse cases to achieve good outcome for all (Brynjolfsson & McAfee, 2017a; Bughin & Manyika, 2018).

To conclude this high-level introduction into ML section, we should add that there exist high short-term expectations related to AI technologies. However, as Chollet (2018, p. 13) stated: "Do not believe the short-term hype, but do believe in the long-term vision. It may take a while for AI to be deployed to its true potential – a potential the full extent of which no one has dared to dream – but AI is coming, and it will transform our world in a fantastic way."

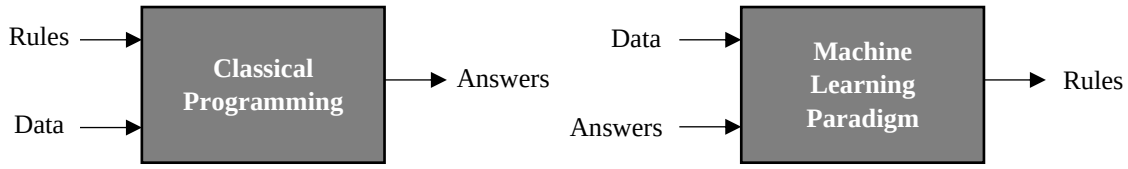
1.4.1 General Artificial Intelligence Concepts and Terminology

Before delving deeper into the specifics of ML methods that are used in the empirical part of the study, it is worth defining what exactly is meant by the terms such as artificial intelligence, machine learning and deep learning, as well as how do these concepts relate to each other. Additionally, the similarity of some ML methods to the traditional statistics and econometrics raises further questions: are ML algorithms merely employing standard techniques to big data or is there more to it? If so, how do these new empirical tools fit into the traditional frameworks used by statisticians and economists?

Developing and experimenting with AI can be traced back to the 1950s, when computer science pioneers simultaneously started exploring different possibilities of making computers "think". In the years since it has undergone multiple cycles of intense optimism followed by disappointments and thereby a dearth of funding ("AI winters"). As we have seen in the preceding section AI currently goes through a phase of intense optimism. Let us look at some definitions to get a clearer picture of the field. Chollet (2018, p. 4) defines **artificial intelligence** as follows: "the effort to automate intellectual tasks normally performed by human intelligence." More broad definition is given by PwC industry report, where AI is characterized as "… a collective term for computer systems that can sense their environment, think, and in some cases learn, and take action in response to what they're sensing and their objectives" (Cameron, Gillham, Rao & Verweij, 2018). In the beginnings AI was based on the so-called symbolic approach to AI, where programmers would hardcode the explicit rules governing the problem at hand (e.g. playing chess). Such approach was suitable for well-defined, logical tasks, but has proved to be intractable for more complex, fuzzy problems, where rules cannot be simply preprogramed. Consequently, a new subfield of AI was born, namely **machine learning**. ML focuses on the issue of how to get computers to program themselves (i.e. can it learn on itself from experience derived from data and perform tasks successfully in a changing environment) (Chollet, 2018). Harrington summarizes the new paradigm as follows: "Machine learning is turning data into information" (Harrington, 2012, p. 5). As can be seen in the Figure 3 with ML, humans input data and expected answers, and out come the rules. As such, one of the principal use of ML is automatic generation of knowledge bases for expert systems (i.e. knowledge discovery in databases), that help human experts with their work and in some cases replace them entirely. Additionally, ML is useful in applications, where humans poses poor domain knowledge, so

effective algorithms cannot be developed, as well as in applications that require dynamic adaptations to changing conditions (Kononenko & Kukar, 2007; Mitchell, 1997).

Figure 3: Classical Programming vs. Machine Learning Paradigm



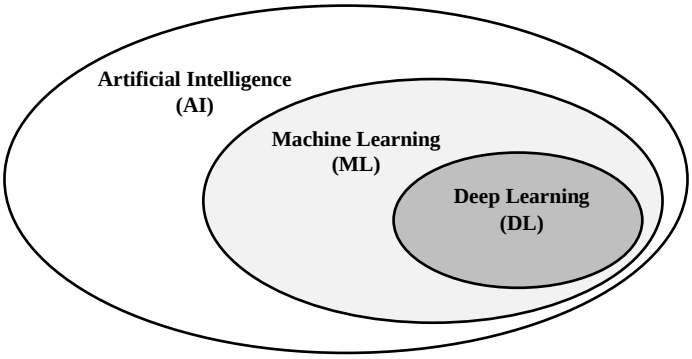
Source: Chollet (2018).

To generate useful knowledge for such applications system must automatically modify in a way to improve, i.e. learn. Central to the contemporary AI research therefore is learning, which can be defined as a process where a learner (in our case machine) improves its performance on a given task via practice, imitation, and trial and error. To be more precise, "we say that a system learns with respect to a particular task *T*, performance metric *P*, and type of experience *E*, if the system reliably improves its performance *P* at task *T*, following experience *E*" (Mitchell, 2006, p. 1). Thus, in order to have a well-defined problem (i.e. task) to apply ML algorithms to, we must identify three important features (Mitchell, 1997): (i) the class of task, (ii) the measure of performance, and (iii) the source of experience. Once we have that, ML algorithm tries to optimize given objective function (e.g. cost function) over certain performance criterion (e.g. number of correct predictions) via weight adjustments of the model's parameters using provided example data. The derived model (or a mapping from inputs to target) can then be used for predictive purposes to infer future predictions and/or for descriptive applications to gain knowledge from data (usually focus on former). Reader might therefore be tempted to equate ML with data mining. To a certain extent this might be true, since both fields use similar methods in their analyses. The main difference however is, that data mining applies those methods to large databases, and consequently also deals with database and data management related aspects, whereas on the other hand, ML is much more than just a database problem as we have already seen – as part of AI field it focuses more on developing intelligent data learning methods (Kononenko & Kukar, 2007). To further advocate the last claim, we must look at yet another promising field that we use in our analysis, namely DL. The Figure 4 displays the relation between the three main fields discussed in this section, namely AI, ML, and DL. **Deep learning**, as a subfield of ML, takes a rather new approach to learning, namely hierarchical representations learning via successive layers, and that is where the "deep" in the name DL comes from. As seen later, in contrast to the "shallow" ML methods, DL methods (i.e. various neural network architectures) also require a specification of the depth parameter[11]. Bengio, Hilton, and

[11] Depth parameter determines the number of layers stacked on top of each other usually organized as ANNs. Those layers try to learn representations of data with multiple levels of abstractions. For example, in image classification task first layer usually represents the presence or absence of edges, the second layer represents shapes construed from the edges and so on (Bengio, Hinton & LeCun, 2015).

LeCunn (2015, p. 1) define DL methods as "representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules each transforming the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned."

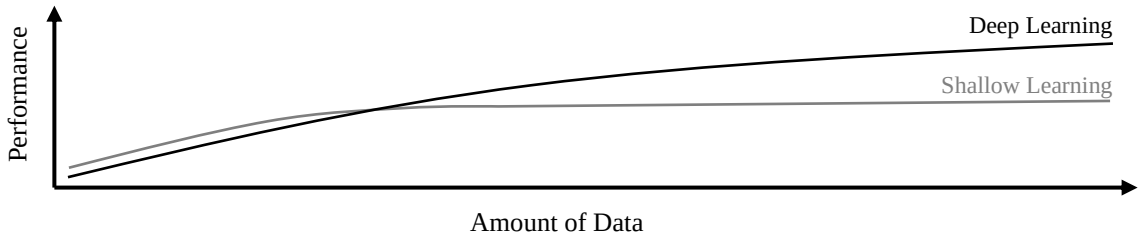
Figure 4: Artificial Intelligence, Machine Learning, Deep Learning



Source: Chollet (2018).

As Chollet (2018) notes, one of the main reason behind DL popularity in the last decade comes from the fact that it offers significantly better performance on many learning tasks. Furthermore, DL also automates one of the crucial steps in ML workflow: feature engineering. Prior to that model development required careful engineering backed by considerable domain knowledge to design appropriate input vector. To sum up, there are two fundamental characteristics that underline the way DL discovers complex patterns in data: (i) hierarchical knowledge representations, and the fact that these (ii) incremental representations are learned jointly (all parameter weights are simultaneously adjusted). These features enable DL algorithms to fully exploit massive amounts of data we have at hand today. Andrew Ng (2019), professor at Stanford and ML expert, tends to answer the question of why DL is taking off with the simple graph presented in the Figure 5 below.

Figure 5: Scalability Drives Deep Learning Progress



Source: Ng (2019).

To get an idea of the current landscape in ML and popularity of different algorithms we can look at the Kaggle website, an online community of data scientists, where people compete in numerous data science challenges. In recent years those competitions were dominated by two approaches, namely **ensemble learning** (e.g. gradient boosting machines) for solving

structured problems, and **deep learning** (e.g. deep neural networks) for unstructured applications such as image classification (Harasymiv, 2015). We have already looked at the latter, whereas the ensemble learning approach remains to be discussed in chapter 2.3.3.

1.4.2 Machine Learning Types and Basic Principles

In this section, we take a closer look into the three **types of ML methods** (Harrington, 2012; Kononenko & Kukar, 2007; Mirjalili & Raschka, 2017): (i) supervised learning, (ii) unsupervised learning, and (iii) reinforcement learning, in order to get a more complete view of the ML methods and their applications.

(i) **Supervised learning** (e.g. classification, regression) is the most common form of ML. Its goal is to learn a representation (i.e. model or mapping function) from labelled input data (i.e. independent variables) to the output variable (i.e. dependent/target variable) that allows us to make predictions about new, previously unseen data. The term supervised thus refers to a set of examples (or instances) that are pre-labelled with the expected answers provided by a supervisor (usually human agent). This enables direct feedback loop in the algorithm's learning process via minimization of error or distance between a given and desired output label. Learning stops when machine modifies internal parameters (called weights) in a way that acceptable performance is achieved (Bengio, Hinton & LeCun, 2015). Supervised learning problems can be further grouped into **classification** (target variable is discrete) and **regression** (target variable is continuous) tasks. Let us suppose we have a data instance (e.g. a patient characterized by various features or independent variables) and we try to predict whether this patient is healthy or ill (i.e. class label); such learning task falls under classification and the aim of the ML classifier is to assign the correct class given the mapping it has learned from the data (patients with known diagnosis). A second type of supervised learning, called regression analysis is the prediction of continuous dependent variable given some predictors or independent variables (e.g. relationship between studying time and exam scores). The supervised learning methods used in classification and regression tasks are listed in Table 2.

Table 2: Supervised Machine Learning Methods

Classification (discrete target variable)						
Decision Trees	Bayesian Classifiers	Nearest Neighbours	Discriminant Functions	Support Vector Machines	Neural Networks	Hybrid Algorithms
Regression Trees	Bayesian Regression	Locally Weighted Regression	Linear Regression	Support Vector Machines	Neural Networks	Hybrid Algorithms
Regression (continuous target variable)						

Source: Adapted from Kononenko and Kukar (2007).

(ii) In **unsupervised learning** there is no supervisor to input the target variable. Consequently, the aim is to find the regularities or structure in the unlabelled data in order to extract meaningful information. There are no correct answers as there is no supervisor that would provide feedback in the process of learning. In the field of statistics such applications are known as **density estimation**. One common ML density estimation technique is clustering where the aim is to find clusters or groupings in data and hence enables us to "organize a pile of information into meaningful subgroups without having any prior knowledge of group membership" (Mirjalili & Raschka, 2017, p. 7). According to Alpaydin (2004) clustering is widely used in businesses to infer the profile of its customers and adjust strategies accordingly. Another broadly used application of unsupervised learning is **dimensionality reduction** that is usually employed in the feature pre-processing stage to reduce high dimensionality of data (i.e. number of features), while retaining most of the relevant information. There are some other subfields of unsupervised learning that are not discussed here, namely associations, anomaly detection, and autoencoders (Salian, 2018).

(iii) **Reinforcement learning** "deals with the problem of teaching an autonomous agent that acts and senses its environment to choose optimal actions for achieving its goals" (Kononenko & Kukar, 2007, p. 14). The output in this domain is therefore not a single value, but rather a sequence of actions that determine a policy. In such setting an agent is attempting to find the optimal way (i.e. good policy) to improve on a specific task. An important part of the reinforcement learning is reward system, i.e. feedback loop that enables learner to improve from iteration to iteration and in that sense, we can relate reinforcement learning to supervised learning discussed before. The more rounds of feedback the agent receives the better it gets (Salian, 2018). A good example of applied reinforcement learning is robot navigating in an environment with a specific goal of finding certain object. At any time, this robot can take different actions with respect to a number of directions. After some training it should learn the optimal policy to reach the goal state (Alpaydin, 2004).

We end our overview of the ML field with three important principles that should lead the design of (supervised) ML systems according to Kononenko and Kukar (2007). As previously described ML uses knowledge extracted from input data as well as background knowledge (i.e. domain knowledge of model developer) to search for the most optimal representation (or hypothesis) given certain optimality condition. When searching through infinite space of those hypotheses we need some guiding principles. The first is the principle of simplicity or the so-called **minimum description length** (hereinafter: MDL) principle, that states "that one should prefer the model that yields the shortest description of the data when the complexity of the model itself is also accounted for" (Roos, 2016, p. 1). It is a formal version of the Occam's razor suggesting that the simplest explanation is usually also the most reliable. To put it differently, the optimal hypothesis is the one that is simple (i.e. has low variance), but also conveys enough information to capture original data accurately (i.e. has low bias). Unfortunately, it is typically impossible to do both, that is why ML

literature talks about variance-bias tradeoff that will be further explored in the empirical section. Another principle that is used less frequently is known as the **principle of multiple explanations**, which says that "all hypotheses consistent with the input data are to be kept" (Kononenko & Kukar, 2007, p. 65). Although it initially seems to be inconsistent with the MDL principle, they can actually both be used to supplement each other. To be more concrete, MDL principle is used in searching for the set of optimal hypotheses in the learning phase, while the principle of multiple explanations motivates us to combine generated hypotheses into hybrid (i.e. ensemble) models in the execution phase. Lastly, the **principles for the evaluation of learned hypothesis** (i.e. models) instruct us to independently assess the learning algorithms on a separate test set, whereas the model training and hyper-parameter optimization is carried out on training and validation sets, respectively. When comparing significance of the performance differences between various hypotheses appropriate statistical tests must be used (Kononenko & Kukar, 2007).

1.4.3 Machine Learning – Relation with Traditional Statistics and Econometrics

Preceding sections tried to establish a comprehensive description of ML field. Hence, we have reached a point where we can place this relatively new and vibrant discipline in relation to other scientific fields. Students of economics related programs are usually familiar with the probability and statistics, as well as econometrics courses. The question therefore is how do ML methods fit into this general framework usually being thought to economics students?

Alpaydin (2004) suggests that methods used in ML originate from different scientific domains – generally speaking, the field lies at the intersection of computer science, engineering, and statistics. For instance, statisticians work on similar types of modelling, especially ones in the subfields of applied statistics. But the underlying approaches taken by the two are fundamentally different in a way that ML practitioners use comparatively less mathematical theory and are more engineering oriented (Brownlee, 2018a). As Chollet (2018) notes: "It is a hands-on discipline in which ideas are proven empirically more often than theoretically." Although there are a lot of critics related to the fact that ML is biased towards theory-free approach, it is essential to take into consideration that without that pragmatic approach it had taken in the past, the field would have probably never reached such an enormous success (Brownlee, 2018a). This point was further advocated in the famous paper by Breiman (2001a) titled "Statistical Modelling: The Two Cultures". In the paper author underlines the fundamental difference between the **data modelling** (i.e. statisticians) and **algorithmic modelling** (i.e. ML practitioners) cultures. The former gives more focus on the stochastic data generating model inside the "black box" (e.g. the underlying structure of the relationship between inputs and outputs), whereas the later focuses on finding a function or algorithm that has the best prediction performance (i.e. generalization) on the test set, and therefore ignores what goes on inside the "black box". Breiman argues that the commitment to this data modelling approach prevented statisticians from using more suitable algorithmic models to tackle applied problems, where focus on finding a good solution for customer is essential. The workflow in applied statistics is usually as follows: (i) think about a parametric data model, (ii) estimate parameters of the model,

and (iii) perform goodness-of-fit on given instances, as well as tests of hypotheses, confidence intervals, etc. But as Breiman claims: "The quantitative conclusions are about the model's mechanism, and not about nature's mechanism. It follows that if the model is a poor emulation of nature, the conclusions may be wrong" (Breiman, 2001a, p. 202). He further notes that using goodness-of-fit tests and residual analysis to check the data model fit opens additional problems[12]. The most obvious way to tackle applied problem is therefore to check how well the model's box emulates the nature's box via estimation of predictive accuracy using cross-validation (hereinafter: CV) or separate test set that is put aside prior to learning. Additionally, while traditional statistics data models certainly have their own use cases, we should not limit our toolbox only to them. Approaching a wide range of problems by using solely data models a priori imposes serious limitations; we must use a larger set of tools advanced by ML community. The approach taken by them advocates that nature produces complex, partly unknowable data. The task is to find a mapping from inputs to outputs such that this mapping is optimal. Focus is therefore shifted from analysis of data models to properties of algorithms that make them "strong" predictors. (Breiman, 2001a). Over the last decade there has however been a trend toward collaborative work between the two fields, which is essential for the future success. As Hastie, Tibshirani, and Friedman (2017) write in the introduction section to a book on **statistical learning** that guides our empirical analysis: "The field of statistics is constantly challenged by the problems that science and industry brings to its door. … This book is an attempt to bring together many of the important new ideas in learning and explain them in a statistical framework."

An interesting observation on the relation between ML and statistics that further advocates our prior discussion is given in the article by Mullainathan and Spiess (2017), where an **applied econometrics view** is taken to analyse this rather new scientific discipline. They say that ML (or more precisely supervised learning, the focus of this study) mainly revolves around prediction tasks, where its success comes from the ability to uncover complex patterns not specified in advance, and consequently good out-of-sample generalization capabilities. On the other hand, econometrics generally revolves around parameter estimation or to put it plainly it deals with the estimation of β coefficients that specify the structural relationship between regressors and dependent variable. As we have seen before ML methods have primarily not been developed for that kind of problems, so "the danger in using these tools is taking an algorithm built for $\hat{y}$, and presuming their $\hat{\beta}$ have properties we typically associate with estimation output in econometrics" (Mullainathan & Spiess, 2017, p. 88). The problems of making inference about the underlying model structure fitted via ML approach is the lack of standard errors after data-driven model selection. Another

[12] Work by Bickel, Ritov, and Stoker (2001, in Breiman, 2001a) shows the omnibus goodness-of-fit tests have little power and typically do not reject hypotheses until the lack of fit is extreme. Residual analysis is similarly unreliable as argued by William Cleveland as it fails to uncover lack of fit in more than four to five dimensions where interactions between variables kick in. To get to the point, the main problem with traditional data modelling is the fact that "model fit itself is of secondary importance compared to the construction of ingenious stochastic model" (Breiman, 2001a, p. 5).

challenge comes from the fact that similar predictions may be produced using different variables due to regularization (i.e. penalty on the model complexity), which can introduce omitted variable bias. Authors hence emphasise the use of machine ML methods in the econometrics framework should fall into the toolbox marked with $\hat{y}$ rather than $\hat{\beta}$, at least as of now. A key area of future research in econometrics and ML should focus on extracting structure from accurate prediction models without making strong assumptions about the underlying processes. The advantages of ML tools highlighted by authors are in line with the previous discussion: (i) automated search for interactions between variables (e.g. in DL), and (ii) better out-of-sample prediction performance. The reasons behind these are twofold, namely flexible functional forms with inbuilt regularizer, and empirical tuning of hyper-parameters on validation sets prior to testing (i.e. regularizer parameters etc.), which solves the problem of overfitting to the data and establishes good generalization of the algorithms. We discuss the benefits of regularization from prediction perspective in the empirical section of the thesis as a part of the ML workflow.

Taking all this into consideration Mullainathan and Spiess (2017) advocate following **economics applications**, where improved prediction has large value:

(i) **big data**: unconventional high-dimensional data, where conventional econometric estimation methods would not work efficiently (e.g. using satellite data for determining economic development, classifying financial messages to explain market volatility, stock market prediction, CS applications in businesses, digital transformation);
(ii) **prediction in the service of estimation**: in the case of linear instrumental variables ML can help us predict the fitted values of $\hat{x}$ regressed on instruments that are used in the second-stage to estimate $\hat{\beta}$ (e.g. use of validation and testing sets to prevent overfitting and various nonlinear functional forms);
(iii) **policy/strategy prediction**: ML can help performing supervision tasks by detecting abusive behaviour, as well as guide strategic management decisions (e.g. marketing campaigns based on client clusters);
(iv) **testing theories**: test economic theories that are inherently about predictions (e.g. efficient market hypothesis, behavioural economics models).

1.4.4 Quantitative Credit Scoring as a Supervised Machine Learning Problem

Quantitative CS, which uses historical information to evaluate future default probabilities as discussed in chapter 1.3 can be framed as a **supervised learning problem** in ML context. In particular, each instance in the dataset is represented by a set of common independent variables and preassigned a discrete label (i.e. dependent variable) denoting either defaulted or non-defaulted status. Due to the binary nature of dependent variable we can further group the task as the **classification problem**. Employing state-of-the-art classification ML methods discussed in chapter 2.3.2 (e.g. k-NN, DTs, SVMs, ANNs, etc.) we can accurately model the complex relationship between the firms' characteristics and default status. This in turn provides a predictive model with high generalization capabilities to predict the PD for new customers.

1.5 Literature Review

Literature dealing with ECRM is substantial; during the last five decades many authors have examined several alternatives to predict customers' payment default (i.e. quantitative CS) or business' insolvency (i.e. financial distress/bankruptcy prediction models). As it is impossible to separate CS from FDP literature, we use generic term **empirical credit risk modelling** to refer to both. This section tries to outline historical developments in ECRM research as well as to highlight major advancements in the field. Aim is to provide a holistic view of the state-of-the-art techniques in the field and thus establish a background for our further discussion on ECRM issues, which is the topic of the next chapter. Finally, we employ the methodology for predicting the probability of default on the corporate dataset in chapter 3.

The literature on ECRM dates back to the 1930's, when initial studies concerning the use of ratio analysis to predict future defaults started emerging. Research up to the 1960's was generally based on univariate analysis with the studies of Fitzpatrick (1932), Smith and Winakor (1935), and Chudson (1945) being the most recognized. The first multivariate analysis for the corporate segment was applied by Altman (1968); the so-called Z-score model remains relevant up to this day. He identified the shortcomings of absolute comparison based on one financial ratio at the time and proposed an extension that combines "several measures into a meaningful predictive model" using statistical technique proposed by Fisher (1936) called multiple discriminant analysis (hereinafter: MDA) (Altman, 1968, p. 591). It is interesting to note however that it was Durand (1941) who carried out the pioneering work of utilizing MDA for financial purposes, i.e. credit worthiness prediction of used car loan applicants. Logit and probit analysis began to appear in the late 1970's with the work of Ohlson (1980) who employed a less restrictive LR model, and Zmijewski (1984) who adopted similar approach (probit model) to estimate the PD. More recently ML methods have been piloted due to the advancement in computational power, lower costs, as well as emergence of big data. For more detailed discussion on the history of ECRM please refer to studies by Gissel, Giacomino, and Akers (2007); Jackson and Wood (2013); Jayasekera (2018); Ravi Kumar and Ravi (2006); Sun, Li, Huang, and He (2014); Wu, Gaunt, and Gray (2010). Out of this literature have come a number of competing empirical credit risk models that employ different kinds of techniques for model development. One of the most explored issues in prior research is definitely the selection of classification method (i.e. classifier/learner).

Most frequently used **individual classifiers** nowadays are still parametric ones such as already mentioned LR. ML on the other hand introduced a wave of new semi-parametric (e.g. ANNs, SVMs) and non-parametric methods (e.g. k-NN, DTs). A number of conclusions emerge from reviewed literature: (i) advanced ML algorithms (especially ANNs and SVMs) outperform traditional statistical LR, and (ii) ensemble methods on average outperform the best individual ML classifiers, which emphasizes the need to implement ensembles – we discuss this point at the end of this review along with other recent advancements (Lessmann, Baesens, Seow & Thomas, 2015). There exists a considerable body of literature applying

ML methods to credit risk context. A number of authors have recognized that these methods present a flexible and powerful framework for default probability estimation with ANNs and SVMs scoring highest predictive performance. Tree-based algorithms and nearest neighbours are on average both inferior to former methods, but still perform relatively well compared to LR. However, previous studies have shown that the performance of the selected method strongly depends on the nature of datasets (Beque & Lessmann, 2017; Hand & Henley, 1997). Furthermore, prediction performance while important should not be the only criteria for method selection[13].

ECRM is a multi-stage process that generally involves DPP, model development phase, and consequent performance evaluation. Application of each classification method requires the presence of data in a mathematically feasible format to work efficiently; we follow recommendations on DPP from the literature (e.g. Anderson (2007); Crone, Lessmann, and Stahlbock (2006); Florez-Lopez (2010); Lessmann, Baesens, Seow, and Thomas (2015); Siddiqi (2017)). A number of authors have recognized the importance of **data projection** which aims at transforming raw data into useful representations. In particular, it is popular in CS to standardize/discretise[14] numeric values while categorical variables are usually transformed using one-hot encoding in case they are nominal or mapped to integer values in case they are ordinal. If numerical values are standardized it is a good practice to additionally winsorize data in order to account for outliers or other extreme values – this is particularly useful when dealing with financial ratios as in our case. It is not uncommon to have **missing values** in a real-world dataset. Several methods are reported in the literature to address the issue of missing values - many studies exclude the cases/features that do not attain a minimum level of availability (e.g. 80%) in the first step. Remaining missing values are then replaced with values specific to each group (e.g. a mean/median replacement).

Additionally, it is useful to analyse data using descriptive statistics and visualize them when possible. Principal component analysis (hereinafter: PCA) using only the first two or three dimension is a good start to visually inspect the patterns or clusters forming in the dataset, however it lacks efficiency when dealing with non-linear relationships. Therefore, authors suggest using t-Distributed stochastic neighbouring embedding visualization technique (Koutanaei, Sajedi & Khanbabaei, 2015; Mirjalili & Raschka, 2017; Šušteršič, Mramor & Zupan, 2009).

Corporate ECRM, the topic of this study, employs data from balance sheets (i.e. financial ratios), macroeconomic variables, and other qualitative variables to estimate credit worthiness for global enterprises, SMEs, or micro-entrepreneurs. The dimensionality of such data is often quite big; some authors have thus suggested that **feature selection** is used in order to examine collected variables for their representativeness in the chosen dataset. Most

[13] A more comprehensive description of a framework for tool selection may be found in the paper by Alaka et al. (2018).

[14] Standardization involves variable scaling so that they reside in a similar numerical range which enables more efficient learning. On the other hand, discretization transforms continuous variables into a limited set of bins.

studies employ filter (e.g. information value, Relief, correlation matrix, RF), wrapper (e.g. stepwise selection, genetic algorithm (hereinafter: GA) methods, particle swarm optimization), and embedded (e.g. regularization) approaches as well as **feature extraction** techniques (e.g. PCA, discriminant analysis, factor analysis) to facilitate model development phase (Anderson, 2007; Beque & Lessmann, 2017; Crone, Lessmann & Stahlbock, 2006; Koutanaei, Sajedi & Khanbabaei, 2015; Lessmann, Baesens, Seow & Thomas, 2015; Zhou, Lu & Fujita, 2015). According to Liang, Tsai, and Wu (2015, p. 260) there "is no exact answer for the best combination of the feature selection method and the classification technique. However, comparing average prediction results with the baseline models shows the models' performances can be improved if the feature selection method is carefully chosen." It should be noted however that for some classifiers feature selection does not always improve performance (e.g. DTs, SVMs). Some authors advocate for feature selection by using either domain knowledge or the integrated approach that embeds expert knowledge into quantitative methods. For more we advise you to refer to Lin, Liang, Yeh, and Huang (2014).

An important DPP decision concerns **data partitioning**. According to literature data is partitioned into training and test sets, which ensures proper model performance evaluation. Furthermore, prior to comparing different methods the best hyper-parameter configuration is identified for each method – this requires additional validation set. To obtain such validation set further holdout CV on the training set is usually performed. Contrary, it is a good practice in ML to use k-fold CV or nested CV, which ensures more robust model performance evaluation (Beque & Lessmann, 2017; Lessmann, Baesens, Seow & Thomas, 2015; Mirjalili & Raschka, 2017; Sun, Li, Huang & He, 2014).

The question of **sample size** has been dealt with in several studies; generally, the larger the sample size the higher the probability that a sample is representative of the population and the better the model performance. This is especially true for the complex ML methods. However, acquiring data might be costly which results in a trade-off between the increase in accuracy obtained from using larger sample and the marginal acquisition costs. The optimal sample size depends on dataset at hand. Several studies have however provided a guideline – for LR having 2,500 defaults is sufficient, while ML methods require at least 10,000 defaults (Crone & Finlay, 2012; Siddiqi, 2017). Another important issue is how to deal with the **imbalanced dataset**. Credit scoring datasets usually exhibit low proportions of defaulted customers – fraction of defaulted customers usually ranges from 0.01 to 0.10. It has been shown that such class imbalance impedes classification, so various under-/oversampling approaches have been proposed (Weiss & Provost, 2003). Chawla, Bowyer, Hall, and Kegelmeyer (2002) introduced a synthetic minority oversampling technique (hereinafter: SMOTE), which conducts a special form of oversampling the minority class. The majority of prior research (e.g. Batista, Prati, and Monard (2004); Crone, Lessmann, and Stahlbock (2006); Lessmann, Baesens, Seow, and Thomas (2015)) has consistently identified undersampling suboptimal to oversampling across different methods as well as datasets. Moreover, SMOTE outperforms both previous resampling techniques. ML community has

addressed the issue of imbalanced dataset via development of distribution insensitive algorithms such as AdaBoost (cost sensitive learning) and RF. A number of authors have recognized that most of ensembles perform well in dealing with samples where a large class imbalance was present (Brown & Mues, 2012; He, Zhang & Zhang, 2018).

Recent advancements in ECRM concern two dimension: (i) **novel ML classification algorithms**, and (ii) **performance measures to assess and compare models**. It is worthwhile to devote some time to discussion of these novelties, since we try to implement them through this study (Garcia, Marques & Sanchez, 2018; Lessmann, Baesens, Seow & Thomas, 2015).

In the past decade or so the academic ECRM research has been flooded with so called **ensemble methods** that integrate the prediction of multiple models. Much empirical and theoretical evidence has shown that model combination increases predictive performance. Ensembles are generally divided into homogenous classifiers that combine base learners of the same type in an independent (i.e. bagging) or dependent manner (i.e. boosting). Among **homogenous ensembles** RF is often credited as a very strong classifier. **Heterogenous ensembles** that combine multiple base learners however provide even better performance; even very simple approach of combining set of base learners through weighted majority voting achieves competitive performance. Thorough discussion on heterogenous ensembles is beyond the scope of this study[15] (Beque & Lessmann, 2017; Brown & Mues, 2012; Kruppa, Schwarz, Arminger & Ziegler, 2013; Lessmann, Baesens, Seow & Thomas, 2015; Sun, Li, Huang & He, 2014).

In general **performance measures** split into three types. Those that assess the discriminatory ability (e.g. area under the curve (hereinafter: AUROC), H-measure); those that assess the accuracy of the probability prediction (e.g. Brier score (hereinafter: BS)); and those that assess the correctness of categorical predictions (e.g. percentage correctly classified or classification accuracy (hereinafter: PCC), KS statistic). The PCC and the KS embody a local model assessment relative to a single cut-off point (i.e. threshold value), whereas AUROC and BS perform a global assessment over whole distribution. Hand (2009) defines the so-called H-measure that provides a normalized classifier assessment and overcomes the deficiencies that the AUROC suffers from – empirical evidence however shows that AUROC and H-measure do not differ much and give similar recommendations. Different measures embody different notion of model performance, which is why it is good practice to consider few measures for the purpose of comprehensive comparison between the methods – we discuss model performance measures used in this study in section 2.4.

ECRM authors sometimes employ performance comparison tests in order to statistically determine the difference between the methods' generalization ability – these tests are known as **multiple comparison tests** and are based on non-parametric hypothesis testing.

[15] A more comprehensive description can be found in a paper by Lessmann, Baesens, Seow, and Thomas (2015) that provides an overview of the state-of-the-art ML methods.

2 CREDIT RISK MODELLING

In previous section we saw that there are three main types of ML, as well as that quantitative CS falls into the category of supervised learning, more specifically classification. That being said the focus of this section is to present different supervised classifiers that are employed in this study and originate from statistics as well as ML field. The first problem each modeler faces is how to choose the right method among hundreds of different algorithmic variations. Domingos (2012) suggests that one conceptualizes the modelling problem as a combination of three components, namely representation, evaluation, and optimization. The optimization component of learning algorithms is discussed in Appendix 7 as it does not play an essential role in our study – we mainly employ standard optimization algorithms that are preprogramed in ML libraries. However, it is useful to devote some time to discuss them in order to get a complete picture. In the Table 3 below you may find the description of the three components of learning algorithms (i.e. classification methods).

Table 3: The Three Components of Learning Algorithms

Representation	Evaluation	Optimization
Choose a set of applicable classifiers that can represent the knowledge (i.e. hypothesis space).	Choose an internal evaluation function (i.e. objective function) to measure learning performance.	Choose a method to search and find an optimal hypothesis in the classifier's space.
Instances k-Nearest Neighbours Support Vector Machines Hyperplanes Logistic Regression Decision Trees Neural Networks Ensemble Methods	Predictive Accuracy Confusion Matrix (Error I & II) Precision, Sensitivity, Specificity Average Precision ROC Curve and AUROC H-Measure	First-Order Optimization (Stochastic) Gradient Descent Momentum Optimization Adaptive α Optimization Second-Order Optimization Quasi-Newton Methods (BFGS) Coordinate-Wise Optimization Sequential Minimal Optimization

Source: Adapted from Domingos (2012).

Technically speaking CS modelling deals with learning a mapping function $f(\cdot)$ from inputs (i.e. independent variables) that are denoted as matrix $\boldsymbol{X} \in \mathbb{R}^{n\,x\,p}$, where n and p stand for number of instances and independent variables (i.e. features or characteristics), respectively, to the output (i.e. dependant) variable denoted by $\boldsymbol{y} \in \mathbb{R}^{n}$, or in mathematical notation $\boldsymbol{y} = f(\boldsymbol{X})$. In order to get the whole picture of the modelling problem we must discuss three factors underlying the mapping itself. Firstly, the definition of default or $\boldsymbol{y}$ part is discussed in the next section. Afterwards section 2.2 looks into the possible independent variables that can provide insightful information for the discrimination between the two groups of customers ($\boldsymbol{X}$ part). Lastly, we consider the $f(\cdot)$ part (i.e. the representation, evaluation and optimization components of learning algorithm as outlined in the table above).

2.1 Development of the Definition of Default

In section 1.1.1 we defined default as the event where "a counterparty does not honour his/her obligations" (Alexander & Sheedy, 2004, p. 211). In classification task default is a binary variable denoted as follows in Equation (2).

$$\boldsymbol{y} = \begin{cases} 0\,;\ Non-defaulted\ Customer \\ 1\,;\ Defaulted\ Customer \end{cases} \tag{2}$$

In CS context we usually define the **defaulted customer** as the one who falls behind on payment for 60 or 90 DPD. In order to derive this definition, one needs a complete payment database to develop the dependent variable $\boldsymbol{y}$. If such data is not available, we can use more stringent definition of default, namely **insolvency proceedings** which is usually used in FDP. The Article 122 of the Financial Operations, Insolvency Proceedings, and Compulsory Dissolution Act, hereinafter referred to as ZFPPIPP (2007) defines insolvency proceedings as "the situation where debtor is not able to settle his/her liabilities falling due in an extended period of time (i.e. continuous illiquidity) or becomes insolvent." Hence, we define the default of a customer as the one who underwent the **bankruptcy proceeding** as defined in ZFPPIPP within one year from the observation point.

2.2 Types of Independent Variables Employed

As discussed in previous sections quantitative CS evaluates historical information of the companies in order to derive rules that enable a priori classification of new companies' obligors. A starting point in prevalent ECRM research is usually a derivation of **financial ratios** as a primary source of historical information. Financial ratio analysis is an important way to analyse financial health. There are usually hundreds of financial ratios measuring different aspects of company (Hajek & Michalak, 2013; Lin, Liang, Yeh & Huang, 2014; Zhou, Lu & Fujita, 2015). The financial features considered in this study are selected based on the frequency of prior occurrences in ECRM related studies. However, exclusive reliance on these financial ratios can be problematic as argued in Appendix 2. Hence, we explore the role of other non-financial features. There are significant risks coming from external environment such as macroeconomic changes as well as other industry/company specific factors. Bellotti and Crook (2009) explore the hypothesis that PD is affected by general conditions in the economy. They show that the inclusion of **macroeconomic variables** (e.g. interest rate, unemployment rate, all-share index, etc.) into the model improves predictive performance. Independent variables used in this study are summarized in Appendix 4.

2.3 Model Representation

In the literature review section, we saw that the first attempt to build a credible CS model began in the late 1960's. Since then there has been an ongoing quest to improve on the prediction performance using different methodological approaches (Jayasekera, 2018): **mathematical models** (e.g. hazard based models, gambler's ruin models), **statistical models** (e.g. discriminant analysis, LR), **artificial intelligence models** (e.g. ANNs, SVMs), and **market based models** (e.g. structural Merton model). Relative frequency of occurrence

in the prior literature suggests discriminant analysis and LR (i.e. statistical models) to be the most popular methods for studying payment defaults. In a systematic review of default prediction models Alaka et al. (2018) nonetheless demonstrate that the sole criteria of popularity for method selection usually results in improper use of tools. Hence the paper tries to identify key criteria to consider when choosing the set of possible methods. These set of criteria depend on the intention of model developer and can be categorized into three distinct categories: (i) **results related criteria** (e.g. accuracy, results transparency, deterministic output), (ii) **data related criteria** (e.g. sample size, types of variable, variable selection), and (iii) **tools' properties related criteria** (e.g. linear vs non-linear representation, generalization ability). Although all these criteria are important to take into consideration modeler usually focuses on the first category where he has to choose between models giving either accurate and/or transparent results. As prior research as well as our discussion of ML approach suggest there is usually a trade-off between the two dimensions - ML methods (particularly non-decision rules tools) are on average more accurate than statistical ones, whereas on the other hand statistical methods provide us with a transparent model (i.e. structural relationship) that can be interpreted for business purposes. Breiman (2001a) provides an alternative view on accuracy-transparency trade-off. He notes that framing the question in a trade-off manner is incorrect, the goal should not be interpretability, but gathering accurate information. Since statistical models are usually less accurate, they consequently provide less accurate information. In case interpretability is important it is recommendable to use either more accurate decision rules ML approaches (e.g. DTs) or decision rules-generating tools applied to the "black-box" ML methods such as SVMs and ANNs. The goal of such rule extraction approaches is to derive symbolic representations to support specialists gaining insight into the complex structure (Alaka et al., 2018). Since the aim of this study is to develop a prediction model, interpretability is not of primal importance. The selection choice of methods for the purpose of this study is hence based on the following two factors, the methods' occurrence frequency and performance achieved in prior literature, while also having in mind other data and tool related issues such as sample size, types of variables, generalization ability, etc. Consequently, next two sections explore two kinds of methods frequently used in the literature, namely **classical statistical methods** (LR) and **machine learning methods** (DTs, k-NN, SVMs, ANNs). Section 2.3.3 takes the research one step further and explores promising hybrid algorithms known as **ensemble methods**.

It is useful to present some notation and general remarks before our discussion of the selected methods. Although all methods described in the following section can be generalized for multinomial applications, we stick with the two-class derivations as this study deals with binary dependent variable. This way we can present general ideas behind methodologies without using too rigorous notation. In Table 4 you may find an overview of the notation used throughout this study.

Table 4: Mathematical Notation Used Throughout the Study

Mathematical Notation	Definition
$\boldsymbol{X} \in \mathbb{R}^{n\,x\,p}$	Input feature matrix, where n and p stand for number of instances and features, respectively.
$\boldsymbol{x}^{(i)} \in \mathbb{R}^{p\,x\,1}$	Vector of all the features values (excluding the label) of the i^{th} instance in the dataset.
$x_p^{(i)} \in \mathbb{R}$	Specific value of the i^{th} instance and p^{th} feature in the dataset.
$\boldsymbol{y} \in \mathbb{R}^{n\,x\,1}$	Output target vector (or dependent variable) of length n.
$y^{(i)} \in \mathbb{R}$	Label or specific output value for the i^{th} instance in the dataset.
$\boldsymbol{X} = \begin{bmatrix} (\boldsymbol{x}^{(1)})^T \\ (\boldsymbol{x}^{(2)})^T \\ \vdots \\ (\boldsymbol{x}^{(n-1)})^T \\ (\boldsymbol{x}^{(n)})^T \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & \cdots & x_p^{(1)} \\ x_0^{(2)} & & x_p^{(2)} \\ \vdots & \ddots & \vdots \\ x_0^{(n-1)} & & x_p^{(n-1)} \\ x_0^{(n)} & \cdots & x_p^{(n)} \end{bmatrix}$	The **design matrix** of independent variables and a set of instances, where each row represents an instance and each column corresponds to the feature; first column in the matrix is usually populated with ones in order to enable intercept (bias) estimation.
$\boldsymbol{y} = [y^{(1)} \quad y^{(2)} \quad \cdots \quad y^{(n-1)} \quad y^{(n)}]^T$	The column vector of dependent variable, where each element corresponds to a specific label for examples.
$\boldsymbol{y} = f(\boldsymbol{X})$	A mapping function $f(\cdot)$: $\boldsymbol{X} \rightarrow \boldsymbol{y}$ that associates each input example $\boldsymbol{x}^{(i)}$ of a learning set $\boldsymbol{X}$ to a single output element $y^{(i)}$ of another set denoted as $\boldsymbol{y}$.

Source: Own work.

2.3.1 Classical Statistical Methods – Logistic Regression

Classical statistical methods are most commonly denoted as **parametric methods**[16], where certain assumptions about the underlying distributional form of $f(\cdot)$ are made. Such assumptions may greatly simplify and speed up the learning process and are particularly efficient in case we have less data. Parametric models are usually defined up to a small (fixed) number of parameters that describe the underlying probability distribution. If assumptions made are correct, then parametric models can give very accurate and precise estimates (i.e. have higher statistical power than non-parametric methods discussed in the next section). However, when assumptions are not met, we may incur a large error (Alpaydin, 2004).

The most common cross-sectional statistical method used for CS is **multiple discriminant analysis** that was first used in the study by Altman (1968). It consists of a linear combination of independent variables, which discriminate the best between defaulted and non-defaulted firms. These linear characteristics are combined in a single score that given certain cut-off point classify the firms in groups according to resemblance. MDA technique makes some restrictive assumptions such as multivariate normally distributed independent variables, equal variance-covariance matrices across the two classes, and the absence of

[16] Some examples of parametric methods are least squares regression, LR, linear discriminant analysis and related statistical tests (e.g. Student's T test, ANOVA tests, etc.).

multicollinearity (Balcaen & Ooghe, 2006). Therefore Ohlson (1980) suggested another group of models, namely **conditional probability models** (e.g. logit, probit models) that are considered less demanding than MDA. These models are based on certain assumptions regarding the probability distribution and use non-linear maximum likelihood estimation. The most popular among the methods is LR also known as logit model that is why next paragraph outlines theoretical background for this method (Balcaen & Ooghe, 2006). Regarding the decision between using MDA or LR Hastie, Tibshirani, and Friedman (2017) note that "it is generally felt that LR is a safer, more robust bet than the MDA model, relying on fewer assumptions."

Despite being named regression, **logistic regression** is a widely spread supervised classification method used to predict the probability $P(y = C|\boldsymbol{x};\boldsymbol{\theta})$ (i.e. likelihood) that an instance belongs to a certain class (denoted as C) based on given features and set of parameters $\boldsymbol{\theta} = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_{p-1} \quad \theta_p]^T$, where θ_0 is a bias term and θ_1 to θ_p input feature weights. LR is a special case of generalized linear models because it uses a weighted linear combination of features plus a bias term as an input into the logistic function that outputs a number between 0 and 1 corresponding to probability. This probability can then easily be transformed into a class prediction using a cut-off rule (Geron, 2017; Beque & Lessmann, 2017). LR is particularly attractive since it allows for categorical as well as numerical variables and enables the interpretation of the features' importance given that there is no multicollinearity. The probability $P(y = C|\boldsymbol{x};\boldsymbol{\theta})$ follows logistic distribution, which implies that "an extremely healthy (weak) company, as compared to a firm that has an average financial health, must experience a proportionally larger deterioration (amelioration) in its variables in order to deteriorate (ameliorate) its financial health score" (Balcaen & Ooghe, 2006, p. 69). To explain the idea behind LR we must introduce the odds ratio, that is defined as $p/1-p$ and calculates the odds in favour of a particular (positive) event we want to predict, where p denotes its probability. The logit function is then defined as a natural logarithm of the odds ratio (i.e. log-odds) and can take any value from $-\infty$ to ∞ (no restrictions are present contrary to modelling probabilities directly). In LR therefore one regresses log-odds on the linear combination of features and takes exponentials on both sides to derive the conditional probability p (Thomas, Crook & Edelman, 2017). This is the LR transformation[17] represented in equation (3).

$$\ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p = \boldsymbol{\theta} \cdot \boldsymbol{x}^T$$

$$e^{ln\left(\frac{p}{1-p}\right)} = e^{\boldsymbol{\theta}\cdot\boldsymbol{x}^T}$$

$$P(y = C|\boldsymbol{x};\boldsymbol{\theta}) = p = \frac{e^{\boldsymbol{\theta}\cdot\boldsymbol{x}^T}}{1 + e^{\boldsymbol{\theta}\cdot\boldsymbol{x}^T}} = \frac{1}{1 + e^{-\boldsymbol{\theta}\cdot\boldsymbol{x}^T}} = \sigma(\boldsymbol{\theta} \cdot \boldsymbol{x}^T) \tag{3}$$

[17] Logistic transformation $\sigma(\boldsymbol{\theta} \cdot \boldsymbol{x}^T)$, where $\sigma(\cdot)$ is a continuous logistic sigmoid function, assures that as weighted sum of features goes towards infinity ($\boldsymbol{\theta} \cdot \boldsymbol{x}^T \to \infty$) LR output approaches 1 and similarly, when weighted sum of features goes towards minus infinity ($\boldsymbol{\theta} \cdot \boldsymbol{x}^T \to -\infty$) LR output approaches 0 which is in line with the notion of probability.

Once the probability is estimated the class prediction is simply made using a cut-off point (i.e. threshold function) of let's say 0.5, which means that a class is 1 if $\sigma(\boldsymbol{\theta} \cdot \boldsymbol{x}^T) \geq 0.5$ and vice-versa (Geron, 2017).

Next step in deriving LR is estimating model parameters also called weights in ML via training on learning examples (i.e. n tuples) given as $\{(\boldsymbol{x}^{(1)}, y^{(1)}), \cdots, (\boldsymbol{x}^{(n)}, y^{(n)})\}$, where $y^{(i)} = 1$ for a positive class (e.g. defaulted class) and $y^{(i)} = 0$ for a negative class (e.g. non-defaulted class). $y^{(i)}$ is assumed to follow Bernoulli distribution (i.e. $y|\boldsymbol{x} \sim Bernoulli(h_\theta(\boldsymbol{x}))$, where $h_\theta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta} \cdot \boldsymbol{x}^T)$). The objective is to determine θ so that LR model estimates high probabilities for positive class and low for negative class, which can be presented via function (4) for single training instance (according to Bernoulli distribution).

$$P(y|\boldsymbol{x}; \boldsymbol{\theta}) = \big(h_\theta(\boldsymbol{x})\big)^y \big(1 - h_\theta(\boldsymbol{x})\big)^{1-y} \tag{4}$$

In order to derive cost function for LR maximum likelihood technique is employed assuming the instances are independent of one another. In practice it is easier to calculate the maximum of log-likelihood that is why we take natural logarithm of the expression in equation (5).

$$L(\boldsymbol{\theta}; \boldsymbol{x}) = \prod_{i=1}^{n} \mathrm{P}\big(y^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\big) = \prod_{i=1}^{n} \Big(h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big)^{\mathrm{y}^{(i)}} \Big(1 - h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big)^{1-\mathrm{y}^{(i)}}$$

$$l(\boldsymbol{\theta}; \boldsymbol{x}) = lnL(\boldsymbol{\theta}; \boldsymbol{x}) = \sum\nolimits_{i=1}^{n} \Big[\mathrm{y}^{(i)} ln\Big(h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big) + \big(1 - \mathrm{y}^{(i)}\big) ln\Big(1 - h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big)\Big] \tag{5}$$

Alternatively, we can rewrite the log-likelihood in (5) as a cost function $J(\boldsymbol{\theta})$ that can be minimized using optimization algorithms described in Appendix 7 (e.g. gradient descent). The intuition behind $J(\boldsymbol{\theta})$ in (6) is as follows, if we correctly predict that an instance belongs to class 1 then the first term in the brackets approaches 0 since natural logarithm of a number close to one (i.e. $h_\theta\big(\boldsymbol{x}^{(i)}\big) \approx 1$) is zero. In contrast, if prediction is wrong the cost function goes towards infinity. The same holds for class 0 (Mirjalili & Raschka, 2017).

$$J(\boldsymbol{\theta}) = -\sum\nolimits_{i=1}^{n} \Big[\mathrm{y}^{(i)} ln\Big(h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big) + \big(1 - \mathrm{y}^{(i)}\big) ln\Big(1 - h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big)\Big] \tag{6}$$

To determine the vector $\boldsymbol{\theta}$ we have to minimize the upper cost function by calculating the partial derivatives with regards to p^{th} model parameter θ_p as follows (Zupan, 2018):

$$\frac{\partial}{\partial \theta_p} J(\boldsymbol{\theta}) = -\sum\nolimits_{i=1}^{n} \frac{\partial}{\partial \theta_p} \Big[\mathrm{y}^{(i)} ln\Big(h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big) + \big(1 - \mathrm{y}^{(i)}\big) ln\Big(1 - h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big)\Big]$$

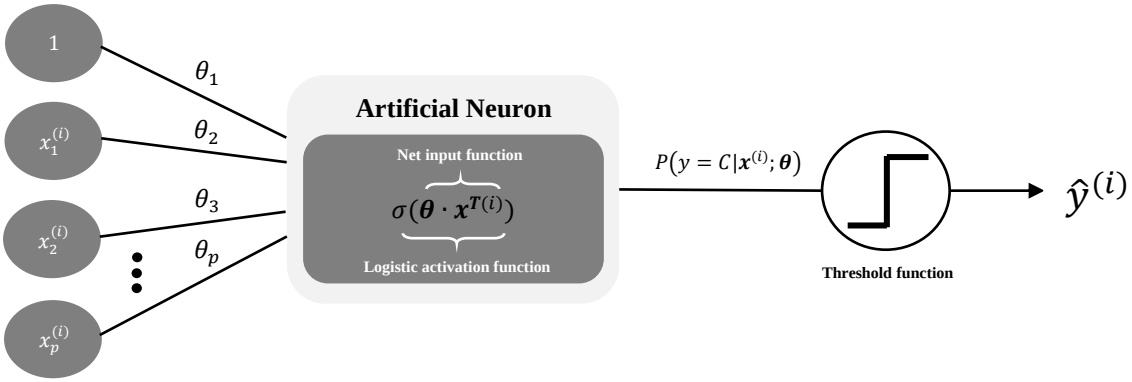
$$\cdots$$

$$= -\sum\nolimits_{i=1}^{n} \Big(\mathrm{y}^{(i)} - h_\theta\big(\boldsymbol{x}^{(i)}\big)\Big) \boldsymbol{x}_p^{(i)} = -\boldsymbol{X}^T (h_\theta(\boldsymbol{X}) - \boldsymbol{y}) \tag{7}$$

If we equate partial derivatives in (7) with zero, we get so called score equations. To solve these equations either Newton-Raphson numerical scheme or other optimization algorithms may be used since this cost function is convex so global minimum is guaranteed to be found

(Mirjalili & Raschka, 2017). Kononenko and Kukar (2007) provide an interesting viewpoint with regards to LR method. They say it can be viewed as a feedforward ANN with no hidden layers and only one artificial neuron – the building block of ANNs as seen in Figure 6.

Figure 6: Relation Between Logistic Regression and Artificial Neural Networks



Source: Adapted from Mirjalili and Raschka (2017).

In order to fit LR properly some assumptions about the underlying characteristics of the data must be met. Although LR relaxes quite a few assumptions usually made when fitting linear regression or MDA, there are still several that we must consider. Violation of these assumptions may lead to a detrimental effect on the model structural analysis. The main assumptions of LR are (Jackson & Schreiber-Gregory, 2018): (i) binary dependent variable, (ii) independent instances (requirement for maximum likelihood estimation), (iii) absence of multicollinearity among independent variables, and (iv) linearity of independent variables and log odds (line one in equation (3)). Table 5 below shows the pros and cons of LR method.

Table 5: Logistic Regression – An Overview of Pros and Cons

Pros: computationally inexpensive, easy to implement, knowledge representation easy to interpret
Cons: prone to underfitting (low accuracy), restrictive assumptions regarding data and functional form

Source: Harrington (2012).

2.3.2 Machine Learning Methods

Contrary to the classical statistical methods ML methods are usually referred to as **non-parametric methods**, although this distinction is not completely rigorous, and some methods may lie in the "grey zone" (i.e. semi-parametric methods). Non-parametric methods have an important advantage in the event we cannot assume particular probability distribution for the data; all these methods assume is, that "similar inputs have similar outputs" (Alpaydin, 2004, p. 153). From that perspective they are denoted as being free of assumptions, i.e. not subject to the stringent assumptions, which enables them to learn any functional form from the data, that leads to higher flexibility, power and consequently

prediction performance. Given the fact we can derive the second meaning of non-parametric techniques as follows: non-parametric methods do not assume the structure of a model to be fixed, it is determined by the data. That is why ANNs and linear SVMs are sometimes thought of as semi-parametric methods in this sense. Disadvantages usually related with non-parametric methods are bigger data requirements, slower training, and overfitting problems. That is why considerable amount of time is devoted to optimizing learning algorithms that can efficiently process large amount of data while also achieve low generalization error (Brownlee, 2016). Non-parametric ML methods include, among others: non-parametric regression, k-NN, DTs, SVMs, etc.

The most frequently used ML method is ANNs that was first introduced into ECRM research in early 1990s and performed significantly better than statistical methods. Another relatively new and efficient ML method used is SVMs, which is particularly attractive for its generalization capabilities. Both methods are discussed in this section with addition of k-NN and DT approach. We also spend some time to review ensemble methods that use different kinds of architectures (Sun, Li, Huang & He, 2014). Appendix 8 notes eight key ML lessons to have in mind when working with ML algorithms (i.e. methods).

2.3.2.1 Nearest Neighbour Approach

Nearest neighbour approach is one of the most basic supervised ML algorithms. It is interesting from the perspective that it does not learn any model (e.g. discriminative function), but rather learns all the examples by heart and makes predictions for new data instances using a similarity measure. This is called **instance-based learning**, in contrast to the **model-based learning** where a learner generalizes from data to build a model, and then uses it to make predictions. Since there is almost no learning this approach is a typical example of a lazy learner (Alpaydin, 2004). The principle behind nearest neighbour method is to find k in distance closest examples to the new data instance and predict the dependent variable using majority voting. That is why it is usually know as k-nearest neighbours method. The number of neighbours can be either user-defined in the process of model evaluation or vary based on the local density of points (i.e. radius-based neighbour learning) and some kernel function (e.g. Gaussian kernel). Despite its simplicity k-NN has been successfully applied to a number of use cases such as handwritten digits and satellite images scenes (Scikit-learn - Nearest Neighbors, 2019a). Parameter k determines the size of the neighbourhood based on cardinality principle enabling the algorithm to elegantly deal with sparsely as well as densely populated regions of the space spanned by features (Kononenko & Kukar, 2007; Beque & Lessmann, 2017).

The k-NN algorithm is straightforward and can be summarized by the following steps (Harrington, 2012):

(i) calculate distances between the new instance $\boldsymbol{x}^*$ from test set and training examples that is defined as $d(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*) = \sqrt[q]{\sum_p |x_p^{(i)} - x_p^*|^q}$ and known under the name Minkowski distance

(generalization of Euclidian and Manhattan distance where q is set to 2 and 1, respectively);

(ii) sort the distances calculated in (i) in ascending order as follows $d_1(\boldsymbol{x}^{(1)}, \boldsymbol{x}^*) \leq d_2(\boldsymbol{x}^{(2)}, \boldsymbol{x}^*) \leq \cdots \leq d_n(\boldsymbol{x}^{(n)}, \boldsymbol{x}^*)$;

(iii) take k examples (i.e. neighbours) with lowest distances;

(iv) find the majority class y^* of the instance $\boldsymbol{x}^*$ from the set of k nearest neighbours as shown in equation (8).

$$y^* = \underset{\boldsymbol{y} \in \{0,1\}}{mode} \{y^{(j)}\}_{j=1}^{k} \tag{8}$$

The upper algorithm suggests that all features as well as nearest neighbours contribute equally to the calculation of target class. In case some features have higher importance, the algorithm can be adapted to account for selected attribute quality measure. On the other hand, kernel functions can be used for calculating the majority vote (using Gaussian kernel closer neighbours contribute more to the overall vote). The main advantage of k-NN as an instance-based method is that it can continuously adapt to new training data. However, there are some disadvantages originating from algorithmic perspective. The computational complexity grows linearly with the number of examples in the dataset. Furthermore, all the learning examples must be stored for classification of new instances since no model is derived. Thus, this method requires a lot of memory (Mirjalili & Raschka, 2017). It is also very sensitive to the scale of the data, which implies that data scaling is a good practice before applying the k-NN method. Table 6 below shows the pros and cons of k-NN method.

Table 6: k-Nearest Neighbours – An Overview of Pros and Cons

Pros: simple nonlinear algorithm, high accuracy, insensitive to outliers, assumption free approach
Cons: computationally expensive, requires a lot of memory, sensitive to scale of the data, similarity defined solely on input variables

Source: Harrington (2012).

2.3.2.2 Decision (Classification) Trees

Some applications require an insight into the underlying structure of the problem (i.e. interpretability) as well as good prediction performance for unseen data. **Decision (classification[18]) tree** is a ML method that can provide both and is therefore commonly used in the industry as it can produce quality decision making guidelines. DT is a non-parametric nonlinear supervised learning method that predicts the value of target variable by learning simple decision rules from training on learning examples. DT consists of internal nodes - corresponding to features, edges - corresponding to the possible feature values, and terminal nodes (leaves) that represent a discrete class label. Each tree starts with a single tree root

[18] This study focuses on the classification aspects of DT algorithm; that is why we can refer to the method as classification tress as opposed to regression tree analysis used in regression setting.

(topmost node) and ends with multiple leaves (Kononenko & Kukar, 2007; Beque & Lessmann, 2017). In between, the set of learning data is repeatedly split into new subsets so that children nodes have as large difference as possible in the sense of class representatives (binary distinction which gives the most information about the class). The process of recursive partitioning stops when certain "level of purity" is attained. Each terminal node (i.e. leaf) is then classified based on majority voting scheme into the set of possible classes (e.g. in our case $y \in \{0, 1\}$). There are three important factors we need to consider when fitting a tree algorithm, namely, the **splitting rule**, the **stopping rule**, and **class assignment decision**. The last factor is the most straightforward to determine so we start with it. Normally, class is assigned to the terminal node based on the majority vote of the learning instances in the specific node. For the other two factors there exists multiple algorithmic implementations. Our further discussion is therefore based on the Python's Scikit-Learn classification and regression tree library implementation. The most common splitting rules are ones that look one step ahead to determine the results of the proposed split. They try to find the best split for each feature in the dataset according to some measure of homogeneity of the target variable within the created subsets. Most libraries (including Scikit-Learn) implement binary DTs for the sake of simplicity and reduction in combinatorial search space (Thomas, Crook & Edelman, 2017). In this setting the algorithm tries to split the training set in two subsets using specific feature (denoted as p) and a threshold t_p. Then it searches for the pair (p, t_p) that produces the purest subsets according to some impurity measure I weighted by their size as can be seen in the cost function for classification tasks in equation (9) where n is the number of instances in subsets. Once the best split is determined, the algorithm recursively repeats the splitting at the new subsets, and sub-subsets until some stopping rule kicks in (Geron, 2017).

$$J(p, t_p) = \frac{n_{left}}{n} I_{left} + \frac{n_{right}}{n} I_{right} \tag{9}$$

The three most common **impurity measures** are Gini impurity (I_G), entropy (I_H), and classification error (I_E). In practice there is usually little difference in results among the three measures, so it is often not worth spending time experimenting with the selection of impurity criteria. The Scikit-Learn library uses Gini impurity as a default calculated for the case of two classes as follows in equation (10), where t is the tree node and $p(i|t)$ is the proportion of the samples that belong to class i for particular node (Mirjalili & Raschka, 2017):

$$I_G(t) = \sum_{i=1}^{2} p(i|t)\big(1 - p(i|t)\big) = 1 - \sum_{i=1}^{2} p(i|t)^2 \tag{10}$$

DTs are often used in expert systems due to their simplicity, interpretability, and accuracy. However, they do have some limitations according to Geron (2017). Firstly, DTs usually produce orthogonal decision boundaries, which makes them sensitive to data rotation, and can produce instability. One way of dealing with this problem is via PCA. Secondly, trees are very sensitive to variations in training data, so each time we fit a tree different model may be produced. Additionally, as DTs make very few assumptions about the data resulting

trees will generally be too large which can lead to overfitting, that is why we need some stopping rule to prune it or constrain their size. This can be done via setting the maximum depth parameter or the minimum number of samples in the split, maximum number of leaf nodes, etc. There are also several algorithmic extensions that build on top of DTs such as RF method discussed in section 2.3.3.1. Table 7 below shows the pros and cons of DT method.

Table 7: Decision Trees – An Overview of Pros and Cons

Pros: nonlinear model, computationally cheap, interpretable results (i.e. "white box model"), can deal with missing values and irrelevant features (little data preparation)
Cons: prone to overfitting, tree structure usually not very robust

Source: Harrington (2012).

2.3.2.3 Support Vector Machines

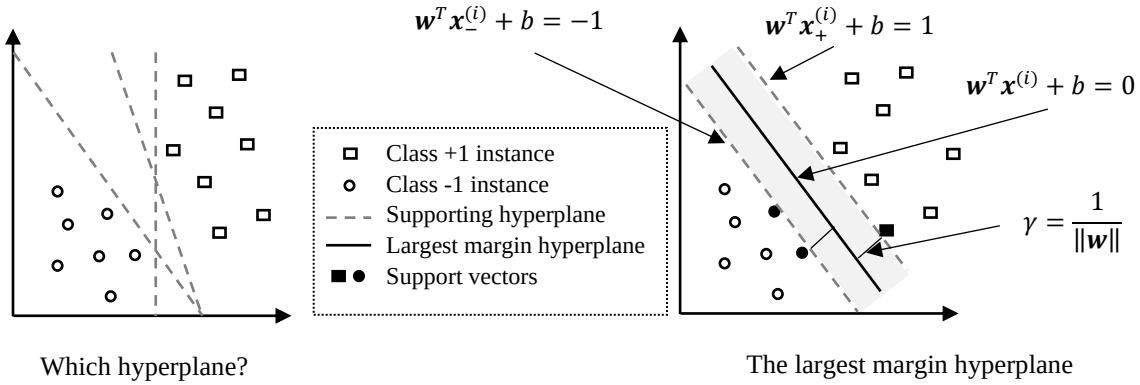
First, we have to introduce some new notation to talk about the **support vector machines** classification. In section 2.3.1 where LR was discussed we used a convention that parameter vector $\boldsymbol{\theta}$ includes the bias term as well as the input feature weights, i.e. the first column of design matrix $\boldsymbol{X}$ is populated with ones. This and the next section where ANNs method is examined use a different convention, which is more common in ML culture. The bias term is denoted by b and the feature weights vector $\boldsymbol{w}$ includes only the input variable weights (i.e. $\boldsymbol{w} = [w_1 \quad w_2 \quad \cdots \quad w_{p-1} \quad w_p]^T$). Additionally, the SVMs classifier directly predicts target variable $\boldsymbol{y}$ taking values 1 and -1 for positive and negative class, respectively (i.e. $\boldsymbol{y} \in \{-1, 1\}$) (Geron, 2017; Beque & Lessmann, 2017).

SVMs is a powerful learning algorithm developed by a Russian statistician Vladimir Vapnik in the 1990s. It is fundamentally different from other methods that construct a linear decision boundary to classify new data instances in the manner that SVMs produces the optimal linear separating hyperplane, as Haykin states: "Given a training sample, the support vector machine constructs a hyperplane as the decision surface in such a way that the margin of separation between positive and negative examples is maximized" (Haykin, 2009, p. 297). To put it differently we no longer minimize the misclassification errors, but rather the new objective becomes maximisation of the distance between the decision boundary and the training samples (i.e. instances) closest to this boundary. This can be presented in the Figure 7 below[19]. We can therefore think of the SVMs as fitting the widest possible street between the classes. Once this street is fitted, we no longer require data instances that are "off the street", since the decision boundary is fully determined (or "supported") by the data instances (which can be thought of as vectors in a multidimensional feature space) located closest to the edge of the street. These instances are known as support vectors – hence the name. Since SVMs learns support vectors by heart to make predictions for unseen data this

[19] The graph on the left represents different decision boundaries that successfully discriminate between two classes. However, SVMs finds the decision boundary with the largest margin (right graph) – it is a **large margin classifier**.

method falls under the **instance-based learning** methods. The rationale behind the large margin classification is that normally the generalization performance is superior to the small margin classifiers that are more prone to overfitting (Geron, 2017).

Figure 7: Linear Separation of Two Classes in Two-Dimensional Space Using SVMs



Source: Adapted from Mirjalili and Raschka (2017).

In order to develop some basic intuition behind SVMs, we only consider the linearly separable dataset first and introduce the **"hard" margin classifier**. This assumption is relaxed later to allow the SVMs to work on all kinds of data. Consider a training set of n instances $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ similar than in LR case, where $\boldsymbol{x}^{(i)}$ is a column vector of p features for each instance i, and y_i takes values 1 for positive class (e.g. non-defaulted class) or -1 for negative class (e.g. defaulted class). In line with our prior discussion we would like to construct a linear classifier with strong confidence of prediction (i.e. decision boundary with the largest margin between positive and negative class). The correct classification of positive (11) and negative (12) examples by large margin classifier can be described by two "hard" constraints:

$$\boldsymbol{w}^T \boldsymbol{x}_+^{(i)} + b \geq 1 \tag{11}$$

$$\boldsymbol{w}^T \boldsymbol{x}_-^{(i)} + b \leq -1 \tag{12}$$

Thus, if we can derive the optimal values of $\boldsymbol{w}$ and b under given constraints (i.e. optimal separating hyperplane) we could classify new data instances by finding the sign of expression $(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b)$; if the sign is greater or equal to 0 the instance belongs to positive class and vice-versa (Fletcher, 2008; Thomas, Crook & Edelman, 2017). Having the classes defined as -1 and 1 is mathematically convenient since we can rewrite the upper constraints with respect to the i^{th} training example using single equation (13) that is independent of the instance class due to multiplication with respective label $y^{(i)}$. For example, if we have a positive instance, then we require $(\boldsymbol{w}^T \boldsymbol{x}_+^{(i)} + b)$ to be a large positive number in order to have a correct and confident prediction. Conversely, in the case of negative example, the $(\boldsymbol{w}^T \boldsymbol{x}_-^{(i)} + b)$ must be a large negative number so that we get a positive number after multiplication with

-1. In both cases constraint (13) thus represents confident and correct prediction of the classifier (Ng, 2018).

$$y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 \; \forall_i \tag{13}$$

As previously said the goal of the SVMs is to find the largest possible orthogonal distance between the closest instances (i.e. support vectors) and the separating hyperplane. This distance is known as the margin γ and is fully determined by the negative and positive hyperplanes which are in turn described by support vectors as represented in equations (14) and (15).

$$\boldsymbol{w}^T\boldsymbol{x}_+^{(i)} + b = 1 \tag{14}$$

$$\boldsymbol{w}^T\boldsymbol{x}_-^{(i)} + b = -1 \tag{15}$$

If we subtract the upper two linear equations and normalize the derived expression by the length of vector $\boldsymbol{w}$ defined as $\|\boldsymbol{w}\| = \sqrt{\sum_{j=1}^{p} w_j^2}$ to prevent unmeaningful scaling of the margin we arrive at the expression (16) for the so-called margin which equals $1/\|\boldsymbol{w}\|$ (Mirjalili & Raschka, 2017).

$$\begin{aligned} \boldsymbol{w}^T\left(\boldsymbol{x}_+^{(i)} - \boldsymbol{x}_-^{(i)}\right) &= 2 \\ \frac{\boldsymbol{w}^T\left(\boldsymbol{x}_+^{(i)} - \boldsymbol{x}_-^{(i)}\right)}{\|\boldsymbol{w}\|} &= \frac{2}{\|\boldsymbol{w}\|} \\ 2\gamma = \frac{2}{\|\boldsymbol{w}\|} \Rightarrow \gamma &= \frac{1}{\|\boldsymbol{w}\|} \end{aligned} \tag{16}$$

The maximization of the margin basically boils down to "minimizing the Euclidean norm of the weight vector $\boldsymbol{w}$" (Haykin, 2009, p. 300). Additionally, minimizing $\|\boldsymbol{w}\|$ is equivalent to minimizing $\frac{1}{2}\|\boldsymbol{w}\|^2$. The later form enables us to apply quadratic programming optimization later on as well as the Lagrangian method[20].

The discussion on SVMs was thus far based on a restrictive assumption that the dataset is linearly separable. In real life applications we usually deal with data that exhibit linearly nonseparable patterns, so we cannot construct a separating hyperplane without any misclassification errors. That is why we have to reformulate our "hard" margin classifier in a way that allows for the violations of the constraint given in (13) for data instances that fall inside the region of separation or fall on the wrong side of the decision boundary (i.e. misclassification). To make SVMs algorithm less sensitive to such outliers we introduce new variables into the constraints, namely slack variables denoted as $\{\xi^{(i)}\}_{i=1}^{n}$ that measure the

[20] For a better understanding of the SVMs please refer to the first part of the Appendix 5, where we provide a derivation of the "hard" margin classifier. The main insight is that the optimal $\boldsymbol{w}$ vector is defined solely in terms of the training examples (i.e. support vectors), which is crucial for the application of kernel trick discussed later.

deviation of an instance to the ideal classification in case of linear separability. Slack variables relax the "hard margin" assumptions since they permit for some linearly inseparable instances to have margin smaller than 1; hence we talk about **"soft" margin SVMs**. New constraints for positive and negative examples are given below (Haykin, 2009).

$$\boldsymbol{w}^T \boldsymbol{x}_+^{(i)} + b \geq 1 - \xi^{(i)} \tag{17}$$

$$\boldsymbol{w}^T \boldsymbol{x}_-^{(i)} + b \leq -1 - \xi^{(i)} \tag{18}$$

In a similar manner as before the constraints can be rewritten into one equation as follows: $y^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b) \geq 1 - \xi^{(i)}$. The new objective function therefore takes in the account finding the largest possible "soft" margin for the separating hyperplane as well as the minimisation of the misclassification error averaged over the learning set. To solve the quadratic optimization problem defined in the second part of Appendix 5 Platt's Sequential Minimal Optimization (SMO) method is used (Harrington, 2012). Once $\boldsymbol{\alpha}$'s for some subset of training are derived we can easily compute the optimal set of weights $\boldsymbol{w}^o = \sum_{i=1}^{n} \alpha_i y^{(i)} \boldsymbol{x}^{(i)}$. Multipliers α_i are weights that determine which data instances contribute to the calculation of optimal margin. Those instance that do are referred to as support vectors. Having found the optimal weights, the calculation of the biased term using the constraint for support vectors is straightforward. Usually, the bias terms over all support vectors from $i = 1$ through to n_s are averaged to get a more stable value:

$$b^o = \frac{1}{n_s} \sum_{i=1}^{n_s} \left(y^{(i)} - \boldsymbol{w}^{oT} \boldsymbol{x}^{(i)} \right). \tag{19}$$

Given optimal solutions $\boldsymbol{w}^o, b^o$ the decision function to determine the class of new instance $\boldsymbol{x}^*$ may be written as follows (Hastie, Tibshirani & Friedman, 2017):

$$y^* = sign\left(\boldsymbol{w}^{oT} \boldsymbol{x}^* + b^o \right) = sign\left(\sum_{i=1}^{n} \alpha_i^o y^{(i)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^* + b^o \right) \tag{20}$$

There is one more extension of the SVMs that makes the method highly popular among the ML practitioners – **kernel methods**. If we rewrite the decision function using the equation for optimal value of weights as in (20) we find that the term depends only on the optimal alphas (which also determine the bias) and the inner product between new data instance and the support vectors. Furthermore, the optimization of alphas requires only the dot product of input vectors to be calculated (as seen in Appendix 5). This property of the SVMs algorithm (i.e. dependence on the inner products between input features) can be exploited to efficiently apply specific kernels and hence solve nonlinear classification problems (Ng, 2018). Kernel methods basically create nonlinear combinations of the original variables (e.g. similar to adding polynomial features in regression) in a higher-dimensional space using a mapping function usually denoted as $\phi(\cdot)$. Such transformation allows us to linearly separate the classes using SVMs linear hyperplane. There is one issue that comes to our mind when discussing kernels, namely the construction of nonlinear combinations seems computationally very expensive. Fortunately, the SVMs optimization algorithm depends solely on the inner products as discussed, so we can simply replace all $\boldsymbol{x}$ with $\phi(\boldsymbol{x})$ to get

inner products formulated as $\phi(\boldsymbol{x}^{(i)}) \cdot \phi(\boldsymbol{x}^{(j)})$. Then given a certain mapping function we can define kernel function $K(\cdot)$ to be

$$K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \phi(\boldsymbol{x}^{(i)})^T \phi(\boldsymbol{x}^{(j)}). \quad (21)$$

Every inner product is thus replaced by $K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ so that the SVMs is learning in the transformed feature space (Mirjalili & Raschka, 2017). "In ML, a kernel is a function capable of computing the dot product $\phi(\boldsymbol{x}^{(i)}) \cdot \phi(\boldsymbol{x}^{(j)})$ based only on the original vectors $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$, without having to compute (or know about) the transformation $\phi(\cdot)$" (Geron, 2017, p. 164). Such functions $K(\cdot)$ enable optimization which massively reduces the computational burden. The approach is therefore called "kernel trick". The popular choices for $K(\cdot)$ in the SVMs literature are linear, polynomial, and Gaussian RBF kernel. These kernels can be intuitively thought of as similarity functions between a pair of instances. For example, Gaussian kernel can be defined as $K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = exp\left(-\gamma \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2\right)$ where $\gamma = {}^1/_{2\sigma^2}$is free parameter to be optimized. The minus converts the distance measure into similarity score and the exponential term maps the score into range between 1 (similar instances) and 0 (dissimilar instances) (Mirjalili & Raschka, 2017). How do we generally know which kernels are admissible to use in SVM? According to Mercer's theorem they must follow certain mathematical conditions (i.e. $K(\cdot)$ is continuous, symmetric in its arguments, etc.). When the conditions are met, we can use $K(\cdot)$ as we know $\phi(\cdot)$ exists even though we do not know what it is (Geron, 2017; Crone, Lessmann & Stahlbock, 2006). Table 8 below shows the pros and cons of SVMs method.

Table 8: Support Vector Machines – An Overview of Pros and Cons

Pros: low generalization error, computationally inexpensive in linear setting, easy to implement, yields global optimal solution
Cons: sensitive to tuning hyper-parameters and kernel choice, kernel SVM computationally expensive

Source: Harrington (2012).

2.3.2.4 Artificial Neural Networks

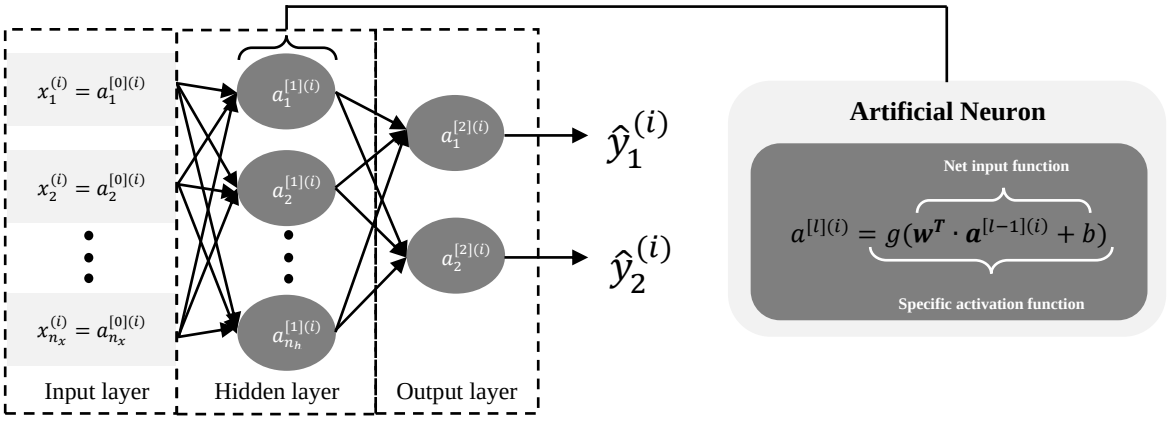
Methods discussed so far fall under the scope of "shallow" ML (i.e. methods that have one or at most a couple of hidden layers). This section covers the concept of **"deep" artificial neural networks** with many hidden layers that enable automatic feature engineering as well as hierarchical learning of complex mapping functions. ANNs were first introduced in the late 1940s and have since witnessed several waves of enthusiasm and disappointments. Currently ANNs enjoy a lot of attention due to their ability to efficiently tackle real-world problems, which was enabled thanks to some major breakthroughs in the previous decade (Geron, 2017).

The basic idea behind the ANNs is to simulate the learning mechanism of biological organisms, where electrical signals containing information travel across the nervous system

composed of neurons, i.e. cell bodies with many branching extensions. When a neuron gets enough signals from other neurons via synaptic connections it fires up and transmits the stimuli forward. The human brain has around 10 billion of such neurons, and when these "simple" biological units work as a part of a wider network complex information processing (i.e. learning) can be performed. Analogous to the biological neural network an ANNs compute a function of the input signals (p features) by propagating the computed values via the multilayer network (i.e. hidden layers of artificial neurons) to the output neuron(s). In such setting learning occurs by changing the weights connecting the artificial neurons. In order to improve the performance a sufficient number of learning examples $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ must be provided to the ANNs. Training data thus serves as a feedback to the correctness of the weights depending on how well the network predicts the output label given the pre-specified label $y^{(i)}$ (supervised learning). By successfully adjusting the weights between neurons in each training iteration good model generalization may be achieved. Hence the deep learners' attractiveness increases with the amount of training instances (Aggarwal, 2018; Beque & Lessmann, 2017; Geron, 2017; Thomas, Crook & Edelman, 2017).

The biological comparison is sometimes criticised due to ANNs being a very simple architecture compared to the real workings of the human nervous system. Therefore Aggarwal (2018, p. 2) looks at the neural networks as "higher-level abstractions of the classical models." This view is in line with our discussion of the LR method. We said that LR can be thought of as an artificial neuron or unit. ANNs gain their power by simply "putting together these basic units, and learning the weights of different units jointly in order to minimize the prediction error ... By combining multiple units, one is increasing the power of the model to learn more complicated functions of the data than are inherent in the elementary models of basic machine learning" (Aggarwal, 2018, p. 2). In order to understand the architecture of a multilayer ANNs we must first look at the basic structure of an artificial neuron also known as the perceptron. **Perceptron** computes a weighted sum of inputs and applies a specific activation function $g(\cdot)$ to that sum resulting in a neuron's activation (or output signal) as seen in Figure 8 (Aggarwal, 2018).

Figure 8: The Basic Structure of Perceptron



Source: Adapted from Aggarwal (2018).

In the case of a **single-layer ANNs**, input signal $\boldsymbol{a}^{[l=0],(i)}$ (where l stands for the hidden layer number, i.e. 0 in this case) represents the input feature vector $\boldsymbol{x}^{(i)}$ and $a^{[L=1],(i)}$ (where L is the number of ANNs layers, i.e. 1 layer) the predicted value $\hat{y}^{(i)}$ for the i-th data instance. The choice of **activation function** that transforms the linear sum of input features plays an important role in network's ability to learn new (complex) features. It is desired that such functions exhibit certain characteristics such as non-linearity, smoothness, as well as differentiability as we see later. Additionally, the choice of specific activation function also depends on the final output we would like to compute, e.g. if the target variable is real, then linear function is applicable, which makes perceptron the same as the least-square regression. On the other hand, if class probability needs to be calculated, sigmoid function is the right choice as in LR. The hyperbolic tangent function is preferred to the sigmoid, when output needs to be both negative and positive. Commonly used activation functions in applied ML are linear (i.e. identity), ReLU, sigmoid, hyperbolic tangent, and hard hyperbolic tangent functions (Aggarwal, 2018).

Stacking together multiple perceptrons into a feedforward neural network architecture generalizes the concept to the so-called **multilayer ANNs** also referred to as multilayer perceptron, that consists of multiple "hidden" layers placed in-between the input layer $\boldsymbol{x}^{(i)}$ (i.e. $\boldsymbol{a}^{[l=0](i)}$) and the output layer $\boldsymbol{a}^{[l=L](i)}$. From the output layer we can derive the predicted value $\hat{y}^{(i)}$ and compute the value of a prespecified loss function. Each hidden neuron in layer l individually computes a transformed weighted sum of the inputs (i.e. signals from the neurons in previous layer $(l-1)$ denoted as $\boldsymbol{a}^{[l-1](i)}$) and outputs the activation signal $a^{[l](i)}$. The outputs from all neurons in layer l, denoted as a vector $\boldsymbol{a}^{[l](i)}$, then become inputs for the neurons in the next layer $(l+1)$ and similarly for other layers. Generally, we can write the output of layer l in vectorized form as

$$\boldsymbol{a}^{[l](i)} = g\left(\boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1](i)} + \boldsymbol{b}^{[l]}\right), \tag{22}$$

where $\boldsymbol{W}^{[l]}$ is now a weight matrix connecting neurons from layer $(l-1)$ to l, $\boldsymbol{b}^{[l]}$ a bias vector, $\boldsymbol{a}^{[0](i)} = \boldsymbol{x}^{(i)}$, and $a^{[L](i)} = \hat{y}^{(i)}$ (Ng & Katanforoosh, 2019; Thomas, Crook & Edelman, 2017). The described process of feeding the patterns of the training data through the network is known as **forward propagation**. Based on some random parameter initialization we can compute the network's predictions and compare them to the correct labels using some predefined loss function. **Loss function** thus plays an integral part of the learning - there exist various forms of loss functions depending on the problem type; e.g. for classification model whose output is a probability in the set $\{0, 1\}$ the binary cross-entropy loss is typically employed, which is analogous to the cost function we used in the case of LR (Brownlee, 2019) and may be mathematically expressed as follows:

$$\mathcal{L}^{(i)}(\boldsymbol{W}, \boldsymbol{b}) = -\left[\mathrm{y}^{(i)} ln\left(a^{[L](i)}\right) + (1-\mathrm{y}^{(i)}) ln\left(1-a^{[L](i)}\right)\right]. \tag{23}$$

The goal of each training iteration is to minimize this composite loss function with respect to all parameters in the multilayer network using complex multivariable continuous

optimization. In the early years, efficient methods to adjust the parameters were not known, therefore ANNs stayed in the shadow of other ML methods. The first significant breakthrough was proposed by Hinton, Rumelhart, and Williams (1968) in the famous article titled "Learning Representations by Back-propagating Errors", where they formulated the **backpropagation algorithm**. With several computational, stability, and overfitting related advancements through the years it still remains the most widely used ANNs training algorithm. In essence, backpropagation can be thought of as a very efficient way of computing partial derivatives of a composite loss function by using the multivariate chain rule of differential calculus and dynamic programming to help us learn the weight coefficients of the network (Aggarwal, 2018). For each training instance the algorithm takes the output error $\mathcal{L}^{(i)}(\boldsymbol{W}, \boldsymbol{b})$ computed in the forward phase and reversely determines the error contributions from each neuron in the network by measuring the respective gradients in the backward phase. The analytical form of gradients $\partial\mathcal{L}^{(i)}(\boldsymbol{W}, \boldsymbol{b})/\partial\boldsymbol{W}^{[l]}$ and $\partial\mathcal{L}^{(i)}(\boldsymbol{W}, \boldsymbol{b})/\partial\boldsymbol{b}^{[l]}$ depends on the specific choice of activation and cost function, that is why it is important that such functions are well-behaved as already mentioned. Derived gradients are used to update the parameters using **continuous optimization strategy** such as simple stochastic gradient descent in the following way, where α is the learning rate (Ng, 2019):

$$\begin{aligned} \boldsymbol{W}^{[l]} &:= \boldsymbol{W}^{[l]} - \alpha * \partial\mathcal{L}^{(i)}(\boldsymbol{W}, \boldsymbol{b})/\partial\boldsymbol{W}^{[l]}, \\ \boldsymbol{b}^{[l]} &:= \boldsymbol{b}^{[l]} - \alpha * \partial\mathcal{L}^{(i)}(\boldsymbol{W}, \boldsymbol{b})/\partial\boldsymbol{b}^{[l]}. \end{aligned} \tag{24}$$

In practice mini-batch stochastic gradient descent is usually used as it aggregates the individual $\mathcal{L}^{(i)}(\boldsymbol{W}, \boldsymbol{b})$ that are too noisy for efficient training; aggregation is done as follows: $J_{MB}(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}^{(i)}(\boldsymbol{W}, \boldsymbol{b})$, where B is the number of examples in the mini-batch. The gradients with respect to the cost function $J_{MB}(\boldsymbol{W}, \boldsymbol{b})$ are then as follows: $\partial J_{MB}(\boldsymbol{W}, \boldsymbol{b})/\partial\boldsymbol{W}^{[l]}$ and $\partial J_{MB}(\boldsymbol{W}, \boldsymbol{b})/\partial\boldsymbol{b}^{[l]}$. Such implementation requires the equation (22) to be vectorized over the B training examples, as well. The outputs from a specific layer thus become matrices instead of vectors, and forward propagation performs the multiplication of the weight matrix with the activation matrix as follows:

$$\boldsymbol{A}^{[l]} = g(\boldsymbol{W}^{[l]}\boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}), \tag{25}$$

where $\boldsymbol{A}^{[l]}$ is now layer's l activation matrix. In the last decade several optimisation methods have been proposed that build on commonly used gradient descent method. It has been shown that selection of the optimisation method influences the quality of learning. Therefore we spend some time in the Appendix 7 to briefly examine the most common gradient-descent strategies implemented in real-world applications such as RMSProp, AdaDelta, Adam (Aggarwal, 2018; Mirjalili & Raschka, 2017).

The above description of backpropagation is of course somehow simplified, and actual implementation usually requires additional adjustments to make the learning algorithm more efficient and stable. The purpose of this study is however not the detailed discussion of such adjustments. Therefore, we rather present some practical issues in training the ANNs in

Appendix 6[21]. Table 9 below shows the pros and cons of ANNs method.

Table 9: Artificial Neural Networks – An Overview of Pros and Cons

Pros: automated feature engineering, extremely flexible algorithm (nonlinearity), scalability, good generalization ability
Cons: non-convex loss function with multiple local minimums, a number of hyperparameters to tune, sensitive to feature scaling

Source: Ravi Kumar and Ravi (2006).

2.3.3 Ensemble Methods

When we were discussing guideline principles to search for the optimal model in the infinite hypothesis space the multiple explanations principle was mentioned. It states that all hypothesis that are consistent with the input data are to be kept, which basically provides motivation for an interesting topic of **ensemble methods** that combine multiple base learners into a meta-learner. Besides DL, ensemble learning has attracted a lot of attention due to its recent success in Kaggle competitions. The goal of ensemble methods is to achieve higher generalization performance than each individual learner by boosting weak learners into a strong meta-learner. Most generally we can divide the methods into two groups: (i) **homogenous ensembles** that employ learners of the same type, and (ii) **heterogenous ensembles** that use different learning algorithms (Zhou, 2012). More applied categorization splits ensemble methods into basic and advanced ensemble techniques. **Basic ensemble techniques** include majority voting and weighted majority voting, that simply outputs the class that has been predicted by majority of classifiers (weighted by some factors if necessary). It is somewhat surprisingly that even these simple approach to ensembles generally achieves higher performance than the single best learner. Even if the base learners are weak (i.e. slightly better than random guessing), the meta-learner can achieve high accuracy provided that the base learners are diverse and independent of each other. In case the learners in the ensemble have different accuracy, it is reasonable to employ the weighted majority voting (i.e. give more weight to the more competent learner).

The prediction for the new instance y^* can be written n mathematical terms as follows in the equation (26) below:

$$y^* = argmax_i \sum_{j=1}^{m} w_j \chi_A\big(L_j(\boldsymbol{x}) = i\big) \tag{26}$$

[21] Additionally, our discussion of ANNs was limited to the conventional fully connected feed-forward deep ANNs. For the sake of completeness, we would like to stress out that other ANNs architectures exist, such as radial basis function networks, restricted Boltzman machines, recurrent neural networks, and convolutional ANNs. However, discussion of these concepts is beyond the scope of this study.

Here, w_j is the weight associated with a base classifier L_j, χ_A is the indicator function and A the set of unique classes. If weights are equal than the formula boils down to a simple majority voting (Kuncheva, 2004; Mirjalili & Raschka, 2017).

Advanced ensemble techniques can be further divided into two groups, namely (i) sequential ensemble methods, and (ii) parallel ensemble methods in accordance to how base learners are generated. In **sequential methods** base learners are generated sequentially and try to exploit the dependence between one another in a way to boost overall performance and reduce bias of weak base learners (e.g. boosting). On the other side the basic motivation of **parallel ensemble methods** is to exploit independence between base learners to reduce variance of predictions (e.g. bagging) (Smolyakov, 2017). **Bagging** was introduced by Breiman (2001b) and stands for bootstrap aggregating which suggest a two-step process, i.e. bootstrapping the training set into more random subsamples and aggregating the learners' predictions using majority vote. The diversity necessary to create more independent learners thus comes from using different training subsamples. Bagging can significantly reduce the variance of an unstable learner. RF is the most used bagging technique that is built upon a DT classifier and is discussed in the next section. **Boosting** as a sequential method tries to boost weak learners by sequentially correcting the mistakes of earlier learners by focusing more on the prior mistakes (increasing their weights during training). As discussed by Breiman boosting can therefore attack both, the high bias and/or variance. The most common implementation of boosting in ML applications is Adaptive boosting or AdaBoost algorithm discussed (Kuncheva, 2004; Zhou, 2012). Bagging and boosting are so-called homogenous ensemble methods. Additionally, we can use multiple learners to build an ensemble for the problem at hand. The base level models are first trained on the whole dataset, then a meta-classifier is stacked and trained on top of them. We refer to this as **stacking** (Smolyakov, 2017).

2.3.3.1 Random Forest

As discussed in the previous section bagging is a technique that can significantly reduce the variance of the prediction function; that is especially true for the high-variance, low-bias classifiers such as DTs presented in section 2.3.2.2. Breiman (2001b) thus introduced an ensemble method that combines a large collection of tree predictors $\{L_i\}_{i=1}^{N}$, where each tree depends on a randomly sampled input vector from original dataset. This results in a greater diversity among predictors - an essential component to achieve higher performance. Breiman proved that the average accuracy of a tree ensemble relates to the correlation ρ between the predictors (i.e. trees). The lower the correlation the higher the generalization ability. That is why **random forest** method consist of an additional procedure that tries to lower ρ, namely instead of selecting all variables to split on, RF algorithm randomly picks a subset of variables for each tree. After a large number of trees are trained, they vote for the most popular class. It has been proved that the generalization error of RF converges to a limit as the number of trees grows and is therefore comparable to the AdaBoost ensemble algorithm

discussed in the next section (Breiman, 2001b; He, Zhang & Zhang, 2018; Thomas, Crook & Edelman, 2017).

The algorithm for RF application is as follows (Hastie, Tibshirani & Friedman, 2017):

(i) for $i = 1$ to N:
   a) draw a bootstrap sample of predetermined size from the training data;
   b) grow a RF tree L_i to the bootstrapped data, by repeating the following steps for each terminal node of the tree: select m variables at random from all available input variables p, pick the best split variable among the m, and split the node into two daughter nodes;

(ii) output the ensemble of trees $\{L_i\}_{i=1}^{N}$;

(iii) use majority vote principle to output a prediction for the new point.

There are generally three parameters that need to be chosen by the modeler: number of the trees N, the size of each sample, and the number of variables at each node m. Usually sample sizes of up to two-thirds of the original dataset are taken, whereas the number of variables is suggested to be equal to $m = \sqrt{p}$. The number of trees trained is usually tuned using CV (Thomas, Crook & Edelman, 2017). Table 10 below shows the pros and cons of RF method.

Table 10: Random Forest – An Overview of Pros and Cons

Pros: low generalization error (does not overfit), relatively robust to outliers and noise, extremely flexible (nonlinearity), easily parallelized, easily measurable feature importance
Cons: requires more computational resources

Source: Breiman (2001b); He, Zhang, and Zhang (2018).

2.3.3.2 AdaBoost Meta-Algorithm

Another popular approach to ensemble learning is boosting. In 1995 Freund and Schapire (1999) introduced the **AdaBoost algorithm**, that has since become one of the most used ensemble methods since it can consequently lead to a decrease in bias as well as variance and significantly increases the generalization performance. AdaBoost stands for adaptive boosting as the algorithm tries to adaptively correct the misclassifications of earlier base learners (also referred to as weak learners). This is carried out via reweighting of the training instances in each round. The misclassified training instances are assigned greater weight and thus given more importance in the next round of learning. Assuming L boosting rounds the resulting L weak learners trained on different reweighted training subsets are then combined using majority voting (Mirjalili & Raschka, 2017).

Using pseudo code, the AdaBoost algorithm can be summarized in following steps (Kuncheva, 2004):

(i) initialize the weight vector $\boldsymbol{w}$ to uniform weights such that $\sum_i w_i = 1$ and pick L, the number of learners to train;

(ii) for $j = 1, \dots, L$:
   a) train a weighted weak learner L_j using $\{\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}\}$ where $\boldsymbol{y} \in \{-1, 1\}$;
   b) predict class labels $\hat{\boldsymbol{y}}$ for $\boldsymbol{X}$ given L_j;
   c) calculate weighted error rate at step j as $\epsilon_j = \sum_{i=1}^{N} w_j^{(i)} l_j^{(i)}$ where $l_j^{(i)}$ is 1 for misclassified instances and 0 otherwise;
   d) compute the confidence coefficient $\alpha_j = \frac{1}{2} ln\left(\frac{1-\epsilon_j}{\epsilon_j}\right)$;
   e) update the individual weights as follows $\boldsymbol{w} := \boldsymbol{w} * e^{-(\alpha_j * \boldsymbol{y}^T \hat{\boldsymbol{y}})}$. If a prediction is correct than there will be positive sign and the corresponding weight is reduced, in the case of misclassification the weight is increased;
   f) normalize the updated weights to sum to 1: $\boldsymbol{w} := \boldsymbol{w} / \sum_i w_i$;

(iii) compute the prediction for new data instance: $\boldsymbol{y}^* = sign\left(\sum_{j=1}^{L} \alpha_j L_j (\boldsymbol{x}^*)\right)$.

AdaBoost is frequently referred to as meta-algorithm[22] as it is used in conjunction with other ML algorithms. As such it overcomes the disadvantages of individual learners, but also presents some new challenges as seen in the Table 11 below.

Table 11: AdaBoost Algorithm – An Overview of Pros and Cons

Pros: low generalization error, few parameters to tune, performs very well with most classifiers in practice, efficient in case of imbalanced datasets
Cons: cannot be parallelized due to sequential learning process, sensitive to outliers and noise

Source: He, Zhang, and Zhang (2018); Nikolaou (2018).

Another boosting ensemble method we use is known as the **gradient boosting algorithm**. It is similar to the AdaBoost but trains on the remaining errors to derive strong classifier. We implement it due to its recent popularity on Kaggle competitions.

2.4 Measuring Model Performance

Section 2.3 outlined a number of classification methods for quantitative CS. The question to ask at this point is therefore, which method is the best from the perspective of results related criteria, i.e. which method provides higher prediction performance? It is important to note one more time at this point that prediction performance is only one of the criteria used for selecting specific model. Sometimes results transparency might be of higher importance for the purpose of making informed business decisions; hence models with transparent results are preferred to more accurate "black-box" methods such as SVMs and ANNs[23].

[22] In this context meta-algorithm is an algorithm that combines a set of non-meta methods; i.e. in our case it sequentially combines base ML methods' outputs in order to achieve lower generalization error.

[23] There are some other data related as well as tool's properties related criteria discussed at the beginning of section 2.3. However, once these criteria are accounted for and the set of appropriate methods is picked it is essential to be able to compare their generalization performance – the topic of this chapter.

The process of evaluating methods' prediction performance is commonly referred to as **model evaluation**. It is good practice in ML to assess the model quality on a separate **test set**, which provides a less biased and more reliable performance estimate of a real-world functioning of the selected model on the unseen data (i.e. generalization ability); in statistics this process is known as **out-of-sample testing**. Setting the hyperparameters of algorithms should therefore be performed ahead on a separate **validation set**, which is a part of training set. This ensures the obtained performance is realistic (Kononenko & Kukar, 2007). This section reviews performance measures used in classification tasks as in our case. One of the most commonly used metrics to quantify the performance of a model is **classification accuracy** defined as the proportion of correctly classified test instances. However, evaluating the classifier solely based on high accuracy may be misleading, which is why dummy classifiers are usually employed to provide a "baseline" performance or the lower bound. This is determined by simply predicting the majority class for each instance in the test set estimated based on the training set. When we deal with highly imbalanced data (e.g. 90% of positive class) dummy classifier alone would achieve accuracy of 90%. This demonstrates why accuracy is not a preferred performance metric in such setting (Geron, 2017; Mirjalili & Raschka, 2017). Thus, it makes sense to focus on other performance metrics when evaluating and comparing a set of models. The rest of this section therefore introduces the following performance metrics: confusion matrix, precision, sensitivity, specificity, average precision, receiver operating characteristic (hereinafter: ROC) curve and related AUROC as well as H-measure. Note that the positive class represents defaulted customers, whereas the negative class represents non-defaulted customers.

(i) **Confusion matrix**: if one is interested in a classification accuracy for each class then **misclassification** (confusion) **matrix** provides additional information. Each row in the matrix represents an actual class, while each column represents a predicted class. For binary classification task a $2 \, x \, 2$ confusion matrix reports four separate counts for a given cut-off value as follows: True positives - TP (i.e. the number of positive cases correctly predicted as positive), False positives - FP (i.e. the number of negative cases incorrectly predicted as positive), True negatives - TN (i.e. the number of negative cases correctly predicted as negative), and lastly False negatives - FN (i.e. the number of positive cases incorrectly predicted as negatives). The "best" model may therefore be regarded as the one where the diagonal true cases (i.e. TP and TN) are maximized or off-diagonal false ones (i.e. FP and FN) minimized (Mirjalili & Raschka, 2017; Siddiqi, 2017). Off-diagonal elements can also be referred to as type I and II error. **Type I error** represents the number of negative cases incorrectly predicted as positive (i.e. the error associated with granting a nonperforming account) and is identical to the FP number in the confusion matrix, whereas FN related **type II error** depicts the positive class mispredicted as negative (i.e. the error associated with opportunity costs of not accepting a performing account). These errors provide us with specific threshold-based assessment of the correctness of categorial prediction (Garcia, Marques & Sanchez, 2018).

(ii) **Precision, sensitivity and specificity**: in the imbalanced dataset case it is usually preferred to look at the specific distribution of the classification accuracy rather than average accuracy as discussed before. Metrics to employ in such setting are **precision**, **sensitivity,** and **specificity**, all derived from the confusion matrix (and therefore related to the previously defined errors). The first calculates how precise our model is in detecting positive instances, the second calculates the share of the correctly classified positive instances, and the latter calculates the relative frequency of correctly classified negative instances (Kononenko & Kukar, 2007; Mirjalili & Raschka, 2017).

$$Precision = \frac{TP}{TP + FP} \qquad Sensitivity = \frac{TP}{TP + FN} \qquad Specificity = \frac{TN}{TN + FP} \tag{27}$$

In the customer default probability task, a sensitivity of 0.85 and specificity of 0.95 would imply that if a customer is defaulted, then the test will confirm the true state with probability 0.85. In case the customer is not defaulted, the test will wrongly proclaim the customer as defaulted with probability 0.005. Note that this corresponds to the notion of classifier's precision; when there is high sensitivity the classifier rarely overlooks an actual positive (low type II error) and vice-versa for high specificity (low type I error) (Kononenko & Kukar, 2007).

(iii) **Average precision**: the relationship between the precision and recall (i.e. sensitivity) can be observed in the stairstep area of the plot. Average precision summarizes such plot as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as a weight (Geron, 2017). This is calculated using the formula below, where n represents the thresholds.

$$Average\ precision = \sum_{n} (Recall_n - Recall_{n-1}) * Precision_n \tag{28}$$

(iv) **Receiver operating characteristics** (hereinafter: ROC) **curve and AUROC**: upper performance measures apart from average precision are all based on indirect minimization of type I and type II errors for an arbitrary chosen cut-off point. When searching for an optimal classifier we are interested in the whole range of possible cut-off points (i.e. thresholds) and corresponding relation between sensitivity and specificity. The **ROC curve** is one of the most widely used methods that displays this relation in a graph without any regard to class distribution or misclassification costs. On a horizonal axis it tracks the FP rate (i.e. $1 - Specificity$), whereas on the vertical axis we have a TP rate (i.e. $Sensitivity$). The ROC curve thus displays how the number of TP varies with the number of FP. By changing the threshold, we get the set of points in the ROC space. The point $(0, 0)$ means that all instances are classified as negative; this gives us sensitivity of 0.0 and specificity of 1.0. On the other hand, the threshold 1.0 means that all instances are classified as positive, which gives sensitivity of 1.0 and specificity of 0.0 at the top right corner point $(1, 1)$ on the graph. The diagonal connecting both points characterises a random classifier. Useful classifiers are the ones above the diagonal. An ideal classifier is represented by point $(0, 1)$ in the top left

corner. The ROC curve for a given classifier is plotted by extrapolation of points that basically represent the confusion matrices at different thresholds. When comparing two or more classifiers the one that is always further from the diagonal dominates. In practice it is usually the case that the respective ROC curves cross, which means that one classifier is better in one region and the other is better in the other region. In such case it is useful to calculate the quality of the classifier via the **area under the ROC curve**; the classifier with the largest AUROC is the best one. AUROC varies from 0.5 to 1 with the former representing a completely random (i.e. dummy) classifier and the latter representing a perfect classifier. It can be thought of as the probability that the classifier ranks a randomly chosen positive instance higher than a randomly chosen negative instance (Jackson & Wood, 2013; Kononenko & Kukar, 2007; Thomas, Crook & Edelman, 2017).

(v) **H-measure:** AUROC as defined in previous paragraph has an "appealing property of being objective, requiring no subjective input from the user" (Hand D. J., 2009, p. 103). Hand (2009) however notes that using AUROC may be potentially misleading as it is incoherent in terms of misclassification costs among various classifiers; effectively it evaluates different classifiers using different metrics[24] (i.e. AUROC depends on the classifier used). As an alternative measure of performance that is superior to AUROC Hand proposes a metric that satisfies both the objectivity (i.e. everyone obtains the same results) as well as the independency criterions (i.e. metric is independent of the empirical misclassification costs). The metric is known as **H-measure** and uses Beta distribution to specify the misclassification cost distribution (default values are usually $\alpha = \beta = 2$). As discussed in literature review, we use AUROC in our study as there is little difference between the measures in practice.

3 EMPIRICAL ANALYSIS

This chapter discusses the research methodology (i.e. the ML workflow) applied in this study. The development of a quantitative CS model requires historical data to derive the transaction-based PD for a given customer[25]. Data however rarely comes in a ready-to-use form for the classification methods presented in chapter 2.3 to work efficiently. Thus, data transformation techniques are one of the most crucial steps in modelling. We discuss this and other related issues in this section, which is organized as follows. First, we discuss the programming environment employed for the purpose of empirical analysis (i.e. Python and Jupyter Notebook) and present an overview of the workflow we follow in the rest of the study. Following is a thorough discussion of each step of the workflow starting with the dataset construction and partitioning as well as all the DPP steps. Second, before delving further into the empirical analysis, it is useful to describe the data using simple descriptive statistics, correlations, and visualization tools – this enables us to better understand the underlying structure of the data. Third, we review the model selection step and consider

[24] For a comprehensive discussion on the topic please refer to the original paper by Hand (2009).

[25] We have discussed the dependent and independent variables employed in this study in the previous section. For more please refer to sections 2.1 and 2.2, respectively.

various approaches to efficiently train the selected methods as well as look into the model performance evaluation phase. Finally, we present, compare, and discuss the results in subsection 3.5.

3.1 Experimental Setup Using ML Workflow

To conduct the empirical analysis and implement ML algorithms as already discussed we need a powerful, opensource, and user-friendly programming language. Additionally, we need a language with easily accessible as well as up-to-date libraries written for a number of tasks that we require. According to the reviewed literature **Python** is the best choice. Since its first appearance in 1991 it has become one of the most popular interpreted programming languages. It is commonly referred to as the scripting language since it can be used to quickly write small programs (i.e. scripts) to automate tasks (McKinney, 2018). Harrington suggests Python is a great language for data science applications due to the following (Harrington, 2012; Lutz, 2013):

(i) **clear syntax**: it carries many high-level data types, which makes abstract concepts easy to implement. It is also easy to process and manipulate the text – hence the name "executable pseudo-code". Python code is designed to be readable and understandable (i.e. straightforward);
(ii) **popularity**: there exists a lot of examples, tutorials, books, courses, etc., which makes learning it fast. Additionally, due to its low entry barrier it enjoys a large number of useful libraries for various applications. It is popular in scientific as well as financial communities and is increasingly being used in organizations;
(iii) **easy integration**: Python scripts can easily communicate with other applications via a variety of integration mechanisms.

Jupyter Notebook integrated development environment is a great way to run Python related data science experiments as it allows modeller to write and execute the code simultaneously, while also edit the corresponding text and results in an organized manner. Furthermore, a notebook enables to break up the analysis into smaller sections that can be run independently; this makes the development interactive and more manageable. Once the analysis is done one can easily share the code using different sharing options including HTML and PDF (Chollet, 2018).

Now that we have discussed the benefits of using Python and Jupyter Notebook for the purpose of data analysis we can turn our attention to the experiment set-up itself using ML workflow. The Table 12 below describes the main libraries we use throughout the empirical section for dataset manipulation and ML application. Majority of the methods we employ in the study are pre-programmed inside the Scikit-Learn library. In order to get raw data into the required shape we use Pandas, a high-performance data editing library. Lastly, Keras is used to train the ANNs as it provides a highly effective way of learning using computer's graphics card (hereinafter: GPU). It runs on top of the TensorFlow developed by Google also briefly discussed in the table below.

Table 12: Essential ML Python Libraries Employed in Empirical Analysis

Library Name [Version]	Description
Keras [2.2.4]	A deep-learning framework that provides a convenient way to define and train almost any kind of deep-learning model. It uses a well-specialized tensor library (e.g. TensorFlow) as a backend to perform various low-level operations (Chollet, 2018, p. 61).
Pandas [0.24.2]	A library providing high-performance easy-to-use data structures and data analysis tools designed to make working with structured data fast, easy, and expressive (McKinney, 2018).
Scikit-Learn [0.20.2]	A core ML library for Python that features various classification, regression, and clustering algorithms; it is designed to interoperate with Python numerical and scientific libraries NumPy and SciPy (McKinney, 2018).
TensorFlow [1.12.0]	A library developed by Google for dataflow and differentiable programming that is based on computational graphs. It enables Keras to run on CPUs and GPUs seamlessly (McClure, 2018).

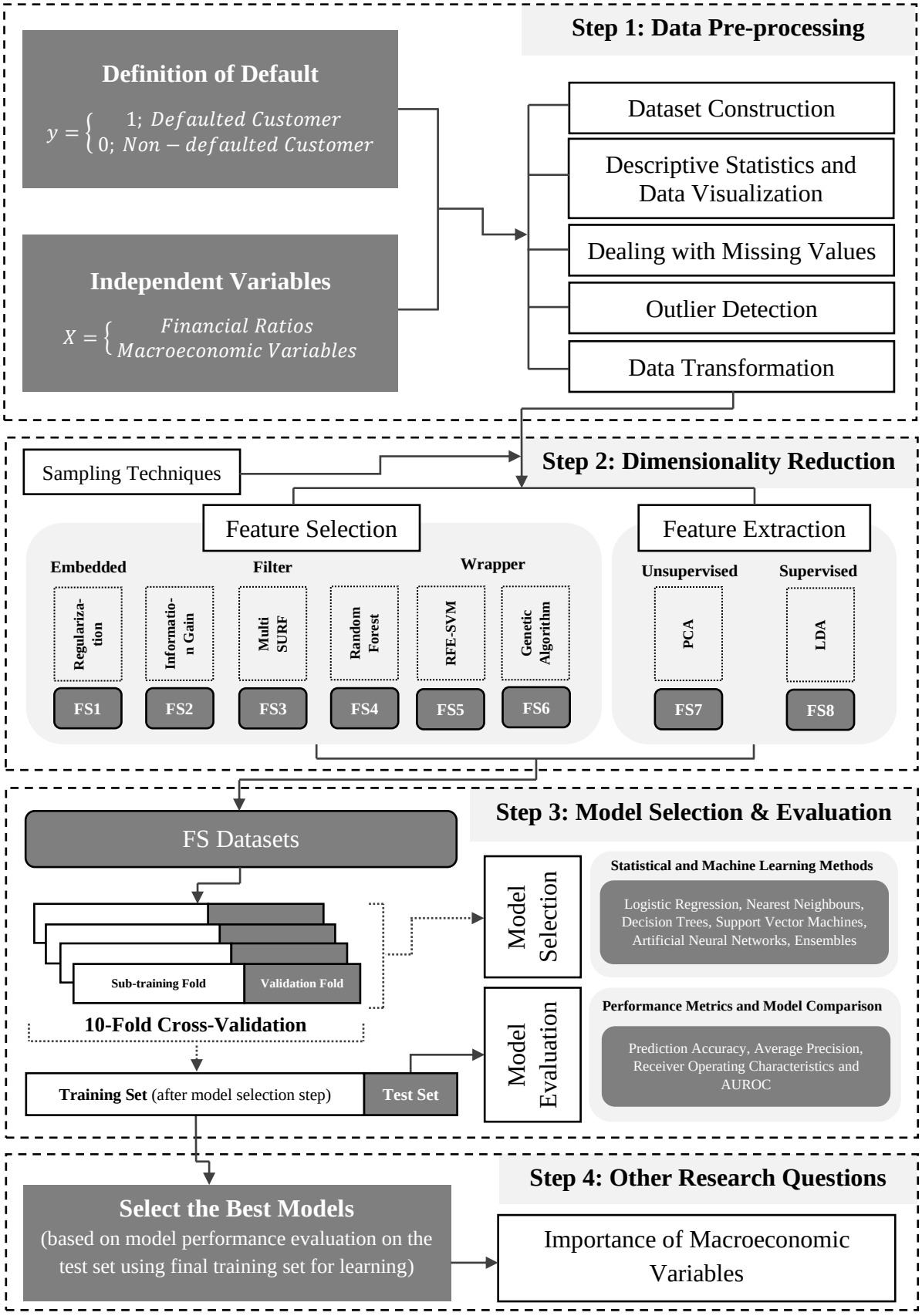
Source: Own work.

Typical workflow for using ML in **ECRM** consist of three steps, namely DPP, dimensionality reduction, and model learning phase. Raw data rarely comes in the desired format, which is why **data pre-processing** is one of the most crucial steps; we apply it in section 3.1. **Learning phase**, applied in section 3.3, includes training and selecting a set of applicable predictive models. Their performance is than evaluated using model evaluation (Mirjalili & Raschka, 2017)**.** Once the model is fully developed, we can use it for prediction of the customer's default probabilities as in our case. The Figure 9 on the next page summarizes the roadmap for applying ML methods in the case of quantitative CS.

3.1 Data Pre-processing Step

This section describes the development of a sample of defaulted and non-defaulted companies for our study as well as the collection of the underlaying data. In previous section we described the default event (i.e. our dependent variable) and specified the set of independent variables characterizing each company's status. This study considers the population of all Slovene commercial companies and big sole traders defined in Article 3 of the Companies Act, hereinafter referred to as ZGD (2006). According to ZGD (2006) Agency of the Republic of Slovenia for Public Legal Records and Related Services (hereinafter: AJPES) manages annual reports filings for national statistics as well as publication purposes. Company's financial datasets for the period from 2013 to 2017 were thus supplied by AJPES – Slovene Business Register. They provide a basis for the derivation of yearly financial ratios as defined in Appendix 4. Data on firm insolvency procedures for the purpose of assigning default event to each company in the business register is obtained from the Information Centre of the Supreme Court of the Republic of Slovenia for the period from 2014 to 2018.

Figure 9: The ECRM Workflow Employed in the Empirical Analysis



Source: Adapted from Mirjalili and Raschka (2017); Own work.

Macroeconomic data was collected from the Statistical Office of the Republic of Slovenia (SURS) and global financial portal Investing.com.

3.1.1 Dataset Construction

Once the data was collected from the relevant sources, we constructed the sample collecting input data referring to one year prior to the default event. Using **stacked sampling approach** discussed in the theoretical part we defined five observation points set at the end of each year ranging from 2013 to 2017; then five 1-year sample windows were defined for which we created a set of independent features. Analogously five 1-year performance widows were created for monitoring performance of given companies and assigning them the class label (i.e. default vs. non-default). This approach of sample construction increased the absolute number of defaulted instances in the dataset, which is especially beneficial when using ML methods. The original AJPES database comprising of 322,203 firm-year instances was then filtered to establish a coherent subset of companies relevant for our analysis. Firstly, based on the Classification of Institutional Sectors (hereinafter: CIS) we selected the companies that are classified as **non-financial companies**[26] (i.e. subsectors S.11001 – public non-financial companies and S.11002 – private non-financial companies) (Statistical Office of RS, 2019a). Secondly, using Slovene Nomenclature of Economic Activities (hereinafter: SKD) we choose the companies that have **similar balance sheet structure** (such segmentation is almost always necessary if we want to develop an accurate model); we selected the following sections according to SKD: C – manufacturing, F – construction, and G – whole sale retail trade, repair of motor vehicles and motorcycles (Statistical Office of RS, 2019b). Lastly, the size of the company influences the structure of the balance sheet - small companies usually do not report enough data points, which results in missing values when calculating financial ratios. Therefore, we only selected the companies whose **average revenue in the sample window exceeds 10,000 EUR**. We added the default variable derived from the dataset on insolvency proceedings. The summary of the final sample is presented in Table 13 below.

Table 13: Sample Size Statistics and Percentage of Defaulted Companies

Year [t]	**2013**	**2014**	**2015**	**2016**	**2017**	**Total**
Sample Size [t]	19,557	19,916	20,396	20,807	21,204	**101,880**
No. of Defaults [$t + 1$]	209	150	169	156	158	**842**
Percentage of Defaults	1.10%	0.80%	0.80%	0.70%	0.70%	**0.80%**

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Afterwards we calculated the financial ratios and merged the financial and qualitative data with the macroeconomic variables. Our final dataset included 53 variables; for more efficient

[26] Institutional units which are "independent legal entities and market producers with principal activity of producing goods and non-financial services" (Statistical Office of RS, 2019a).

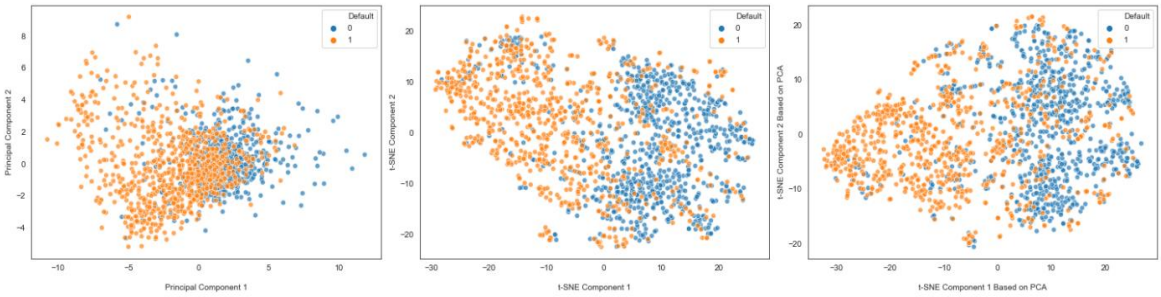
handling we assigned shorter names to the variables as follows: the 45 financial ratios were given code names ranging from *f01* to *f45* and the macroeconomic variables from *e01* to *e05*. The following analysis is based purely on the financial ratios. We introduce the macroeconomic variables into the modelling in section 3.4.

3.1.2 Descriptive Statistics and Data Visualization

Prior to dataset partitioning it is useful to get a quick **description of the data**, in particular means and standard deviations, the number of all non-missing values, variables' distribution characteristics as well as possible extreme values using *describe()* method in Pandas (Geron, 2017). After a careful inspection of the descriptive statistics found in the first table of Appendix 9 we concluded, that our data includes some missing values and extreme values. The last point was further validated by plotting histograms and box plots for each variable. The distributions appear to be normal or log-normal around the centre for most ratios, however data suffers from extreme values (i.e. outliers) that are common in ratio analysis due to near-zero denominators. We consider solving the issues of missing values and outliers for the purpose of modelling in the next section.

So far, we have only taken a quick glance at the data. It is time to get a little bit more in depth and employ different **data visualization tools**. First, we performed a fast DPP procedure on the whole dataset for the purpose of efficient application of different visualization techniques; more detailed discussion follows in the subsequent sections. In Appendix 10 you may find the correlation heatmap that uses coloured cells to show a correlation matrix between the variables. As expected, there are some highly correlated variables – we take care of the highly collinear independent variables in the second step of the analysis as there is no added value in having multiple variables with similar information in the model. Another useful tool for data exploration and visualization is t-distributed stochastic neighbouring embedding (hereinafter: t-SNE). It gives developer a feel or intuition of how the data is arranged in a high-dimensionality space. Similarly, to PCA t-SNE effectively reduces the dimensionality of dataset. However, in doing so it preserves small distances contrary to PCA, which is particularly useful when dealing with non-linear datasets (Violante, 2018; Derksen, 2016). The Figure 10 below shows three different 2D visualization settings. First figure from the left is a standard visualisation output of the first two principal components of PCA that explain 33% of the variation. We can see that the first two components definitely hold some useful information, however the two clusters are still overlapping. Second figure thus shows the output of a more powerful approach called t-SNE as discussed. We can see a significant improvement over the previous visualization. For the last case (right figure) we reduced the number of dimensions with PCA before feeding data into the t-SNE. For this we used the first fifteen principal components that explain approximately 82% of the variation. The results are similar. However, the fact that we only used fifteen dimensions rather than all 45 financial ratios should encourage us to consider some dimensionality reduction prior to applying statistical and ML methods. Furthermore, the fact that unsupervised visualization techniques managed to cluster the defaulted and non-defaulted companies considerably well, is a good sign for the rest of the analysis.

Figure 10: PCA (left) vs. t-SNE (middle) vs. t-SNE Based on PCA (right)

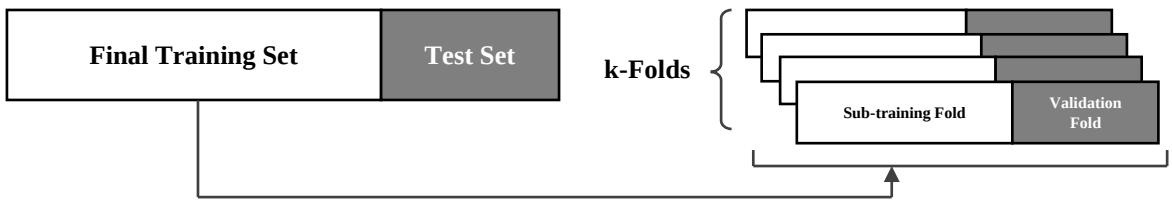


Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

3.1.3 Dataset Partitioning and Dealing with Missing and Extreme Values

It is a good practice in ML to **partition the original dataset** into three separate sets prior to conducting any DPP and dimensionality reduction steps for the purpose of modelling. This is to prevent data leakage, which would result in too optimistic performance evaluation. The **training set** is used to fit different models, whereas the **validation set** serves for model selection, i.e. the selection of the best model using various hyper-parameter settings (hyper-parameter tuning). Finally, the **test set** enables less biased model performance evaluation as it is completely independent of the previous modelling steps. This method is known as the holdout CV. According to Mirjalili and Raschka (2017) one potential disadvantage of the holdout method is the sensitivity of performance estimates to how we partition the original dataset. However, this is a less notable problem when dealing with large dataset[27]. Scikit-Learn provides a *model_selection.train_test_split* to perform data partitioning. A useful feature of the method when dealing with imbalanced data is stratified sampling that divides the original population into homogeneous subgroups; this guarantees that the derived sets are representative of the overall population. We partitioned the original dataset into two subsets, i.e. training and test sets where the first includes 80% of the original instances, while the test set is comprised of the remaining 20%. For model selection phase we use *model_selection.GridSearchCV* that enables exhaustive search over user-defined hyper-parameter values. Figure 11 illustrates the concept of hold-out CV with inner k-fold CV for model selection.

Figure 11: The Concept of k-Fold Cross Validation



Source: Adapted from Mirjalili and Raschka (2017).

[27] In future research section we discuss the nested CV that results in more robust final model performance estimates.

For each combination of values k-fold CV further splits the training set into k-folds without replacement, where $k-1$ folds are used for model sub-training, and one fold is used for model validation – the process is repeated k times so that we obtain a robust average model performance to select the best hyper-parameter values (Albon, 2018).

Table 14 summarizes the absolute and relative proportions of instances in the described sets. The training set used for final model learning consists of 80,987 instances, whereas the test set consists of 20,247 instances. Number of defaults in the sets is 644 and 166, respectively.

Table 14: Dataset Partitioning and Corresponding Statistics

Dataset Partition	Sample Size	No. of Defaults	Percentage of Defaults
Training Set	80,987	664	0.80%
Test Set	20,247	166	0.80%
Dropped Instances	646	12	1.80%

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

The quality of the data is the key factor in training an accurate model. Furthermore, most ML methods cannot work efficiently with missing data and extreme values (e.g. outliers), which is why we need to pre-process raw inputs in the training. As we have seen our data contains some **missing values**, which is why we employ a two-step procedure to take care of the issue. Firstly, we exclude the instances (i.e. dataset rows) and variables (i.e. dataset columns) that contain more than 20% of missing values denoted as "NaN" in our dataset. This operation decreased the original number of country-year instances for 646 instances and dropped 3 variables: interest coverage ratio, inventory turnover ratio, and net income growth. It is essential that dropped instances do not fall under the missing not at random scope. Dropping them would mean introducing a bias into the sample (Florez-Lopez, 2010). The percentage of defaults in the dataset remained roughly the same, which means that the likelihood of excluded instances coming from defaulted vs. non-defaulted group is similar. Secondly, remaining missing values are then imputed using simple median imputation strategy which is robust to outliers (Albon, 2018; Geron, 2017). It is important that the overall median values from the training set are saved in order to carry out the same strategy on the validation and test sets when selecting/evaluating the model.

Another essential DPP step is identifying the **extreme values** commonly referred to as outliers. According to Albon (2018, p. 69) outlier detection is "more of an art than a science". A common method is to look at individual variables and identify extreme values using percentiles. The identified outliers are than either excluded or winsorized. We use winsorization at first and ninety-ninth percentile which effectively clips the data at given values in a symmetric fashion (Badr, 2019). As in the missing values case the winsorization values are stored for later use on the validation and test sets.

3.1.4 Data Projection – Transformation of Numerical Variables

The last step before feeding the data into steps 2 and 3 to carry out dimensionality reduction and model learning is to perform **data projection**. As discussed in the literature review section data projection is essential for ensuring useful data representations. Hence, we transform variables in the following manner (Crone, Lessmann & Stahlbock, 2006). We carry out feature scaling on numerical variables (i.e. financial ratios) – in particular, we standardize them using *preprocessing.StandardScaler()* method as follows; mean and standard deviation are calculated for each variable, then conventional scaling is performed by subtracting the mean and dividing by the standard deviation in order to get mean centred values with unit variance. This ensures efficient learning since ML algorithms usually do not perform well when numerical inputs have different scales (Hearty, 2016). As before it is crucial to fit the transformers that learn parameters using only the training and sub-training data and not the full dataset in order to prevent data leakage.

3.1.5 Data Sampling Techniques

In the literature review section, we discussed the issue of imbalanced dataset and possible implications of LDPs for classification methods' performance. As there seems to be no ambiguous answer with respect to the effectiveness of using specific **sampling technique**[28], we examine their performance differences and try to identify the optimal one given our specific problem domain. First, we apply the two most common sampling techniques; one is known as random minority oversampling (hereinafter: MOS) and the other as random majority undersampling (hereinafter: MUS). In MOS data instances of the minority class are randomly duplicated, whereas in MUS instances of the majority class are randomly discarded from the dataset (Van Hulse, Khoshgoftaar & Napolitano, 2007). Both sampling techniques are implemented in Python using the Imbalanced-Learn package that is fully compatible with Scikit-Learn ML library. Chawla, Bowyer, Hall, and Kegelmeyer (2002) proposed another oversampling method called synthetic minority oversampling technique. It creates new synthetic minority instances by extrapolating between the existing minority instances rather than just duplicating original examples as in MOS. The method first identifies the k nearest neighbours of the minority class for each instance (recommended value of k is 5) and then generates an artificial instance in the direction of these neighbours. We use a combination of over- and undersampling methods called SMOTENN that performs oversampling using SMOTE and cleans the majority class using Edited Nearest Neighbours (hereinafter: ENN). ENN removes instances for which the class from nearest neighbours differs (Batista, Prati & Monard, 2004). The properties of the derived datasets are displayed in Table 15 below. Note that we only apply the sampling on the training set in order to facilitate learning using balanced data. The test and validation sets are left untouched (i.e. remain imbalanced) in order to perform realistic performance evaluation. The pre-processed datasets were than fed into the second step according to the empirical workflow in Figure 9.

[28] "Sampling attempts to balance original data based on a series of sampling algorithms by adjusting the number of samples in different classes" (He, Zhang & Zhang, 2018).

Table 15: Resampling the Training Set

Dataset Partition	Sample Size	No. of Defaults	Percentage of Defaults
Training Set – MUS	1,328	664	50.00%
Training Set – MOS	160,646	80,323	50.00%
Training Set – SMOTENN	156,053	80,322	51.47%

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

3.2 Dimensionality Reduction

Many ML tasks involve a high number of independent variables (i.e. features) in the dataset, which can make model training extremely slow as well as decrease its overall performance due to the "curse of dimensionality" as described in Appendix 8 – Lesson 6. This is also the case in our dataset that originally contained 50 independent variables; some of them may contain noise, which means they are not useful in predicting the PD. Domain knowledge can help us in selecting the most important variables. However, most commonly there is no agreed set of variables that are optimal for specific CS application. Fortunately, there exists several techniques that enable dimensionality reduction while keeping the most relevant information; section 3.2.1 discusses feature selection whereas section 3.2.2 deals with feature extraction techniques. The aim of this DPP step is to filter out unrepresentative features, i.e. the ones that bring relatively little predictive power to the model (Geron, 2017; Liang, Tsai & Wu, 2015; Zhou, Lu & Fujita, 2015). The main advantages of employing these techniques can be summarized as follows (Koutanaei, Sajedi & Khanbabaei, 2015; Salappa, Doumpos & Zopounidis, 2007): (i) decreasing the noise in dataset, (ii) reducing the computational costs, (iii) useful for data visualization and better understanding of the model, and (iv) simpler application.

3.2.1 Feature Selection Techniques

Generally speaking, **feature selection** (i.e. selection of an optimal subset of original variables) can be categorized into three subgroups. **Filter methods** select variables by ranking them using various statistical measures that assign predictive power to the variables. This search strategy does not involve any model training in the process of evaluation. In contrast **wrapper methods** consider feature selection as a part of model learning phase by assessing different variable subsets sequentially; that means that for each generated subset we train a model using specific classification method and evaluate the goodness of the selected subset. Similarly, **embedded methods** measure the importance of variables during model learning. However, these methods are embedded in the classification algorithms themselves which is why we discuss them in the next section dealing with model training (Zhou, Lu & Fujita, 2015). Generally, filters are much faster and more efficient compared to other feature selection strategies, but they ignore the dependencies among variables in the dataset. Finding the best subset using wrapper methods accounts for that dependency, however, it can become computationally intensive, which is why practitioners use various

heuristic search strategies that are able to identify suboptimal solutions (Chen & Li, 2010; Kozodoi, Lessmann, Papakonstantinou, Gatsoulis & Baesens, 2019). This study employs the following feature selection techniques:

(i) **Regularization** (embedded method) is a technique that modifies the classification method in a way that the complexity of the model is reduced which enables better generalization ability. Regularized techniques are among the most helpful feature selection procedures as they provide sparse solutions, where weaker features' parameters return to zero, leaving only subset of features with real coefficients. Approaches to regularization are method-specific which is why we discuss them later (Hastie, Tibshirani & Friedman, 2017; Zhou, Lu & Fujita, 2015).

(ii) **Information gain** (filter method) or mutual information measures the dependency between two variables; the higher the value the higher the dependency. Information gain is usually a good measure for deciding the relevance of features, however it does not consider the dependencies between features in the dataset (Ross, 2014).

(iii) **MultiSURF** (filter method) is a filter-style feature selection algorithm inspired by original "Relief" that evaluates the importance of variables according to how well their values distinguish between the instances of the same and different classes. In that sense it also "detects the information content of a variable that stems from the dependencies between all variables in the dataset" (Kononenko & Kukar, 2007, p. 164; Urbanowicz, Olson, Schmitt, Meeker & Moore, 2018).

(iv) **FS with random forest** (filter method) employs an ensemble classification method to measure implicit feature importance based on the average impurity decrease computed from all DTs in the forest (Chen & Li, 2010; Mirjalili & Raschka, 2017).

(v) **Recursive feature elimination** (hereinafter: RFE) **using SVMs** (wrapper method) is a greedy, iterative process that wraps around a learning method such as SVMs in order to find the best-performing feature subset. As the name suggests RFE selects features by recursively considering smaller and smaller sets and ranks features using weights assigned by the learning method (Hearty, 2016).

(vi) **Genetic algorithms** (wrapper method) are evolutionary search algorithms inspired by the principles of natural selection in biology. They enable solving various optimization problems including feature selection. The idea of GA is to combine a set of solutions (i.e. population) from generation to generation in order to extract the best genes (i.e. features) of a given chromosome (bit string of 0s and 1s that represents inclusion/exclusion of these genes) using crossover and mutation as genetic operators (Scrucca, 2013). Crossover creates two offspring chromosomes from parent chromosomes copying selected genes from each parent. On the other hand, mutation randomly changes the value of single genes given some small probability (Liang, Tsai & Wu, 2015). The quality of each chromosome is evaluated using fitness function that determines the probability of it surviving to the next generation – the larger the fitness value the higher the survival probability. The process ends once some stopping criteria is met (Lin, Liang, Yeh & Huang, 2014; Šušteršič, Mramor & Zupan, 2009).

3.2.2 Feature Extraction Techniques

Alternatively, we can reduce the complexity of the model using **feature extraction**, which derives a new feature set of lower dimensionality from the original feature space. By far the most popular unsupervised[29] feature extraction technique is the **principal component analysis**. We have mentioned it in the visualization section of the study. PCA projects data onto a lower-dimensional hyperplane that preserves the maximum variance in order to lose as little information (i.e. variability) as possible. The resulting orthogonal axes (i.e. principal components) can thus be interpreted as "the directions of maximum variance given the constraint that the new feature axes are orthogonal to each other" (Mirjalili & Raschka, 2017, p. 142). The first principal component preserves most of the variability of the original train set, the second preserves the second most variability, etc. Technically speaking each principal component is an eigenvector of the variance-covariance matrix of the original variables meaning that if we plot the components and normalize their corresponding eigenvalues (magnitude of eigenvectors) we get a graph known as the explained variance ratio – it indicates the proportion of original variability along each principal component. The number of new dimensions we choose usually relies on the rule of thumb (e.g. 80% of original variance) (Geron, 2017; Šušteršič, Mramor & Zupan, 2009). Another common feature extraction technique is **linear discrimination analysis** (hereinafter: LDA). This is actually a classification method, but in the process of training LDA derives the most discriminative axes between the classes that can be used as new features. In that sense it is similar to PCA as it also projects original data to a lower-dimensional space. However, LDA has an additional goal of maximizing the differences between classes (Albon, 2018; Chen & Li, 2010). One assumption in LDA is that the data is normally distributed. Furthermore, it is assumed that the classes have identical covariance matrices and that the samples are statistically independent. Even if one or more assumptions are violated, LDA still functions reasonably well in practice, which is why we consider it in this study (Liang, Tsai & Wu, 2015; Mirjalili & Raschka, 2017). The final selection of linear discriminants is similar to PCA; we pick the number of dimensions given ratio of discriminability. In our case we only have one linear discriminant since their number is limited to $C - 1$, where C denotes the number of class labels.

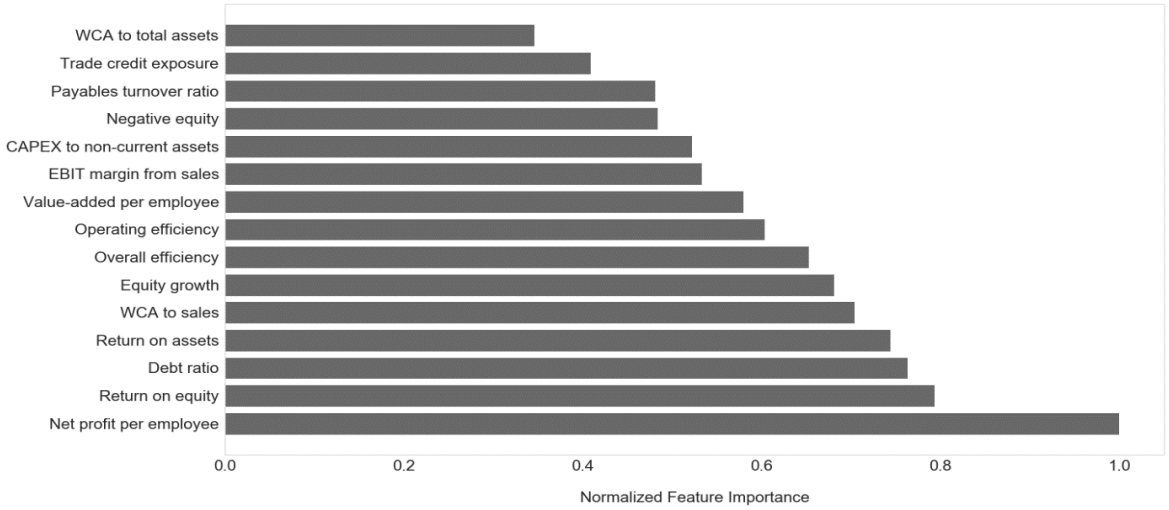
3.2.3 Application of Dimensionality Reduction Techniques

In general, final CS models usually consist of 8 to 15 variables; the exact number depends of the modelling goal (i.e. model transparency vs. prediction accuracy). When visualizing our dataset, we saw that the first fifteen principal components explained approximately 82% of the variation; the fact alone should encourage us to apply some dimensionality reduction techniques. As discussed in section 3.2.1 some feature selection algorithms (especially univariate filter methods) do not account for the dependencies between features in the

[29] PCA falls under the scope of unsupervised feature extraction techniques in contrast to the linear discriminant analysis that is a part of supervised techniques. The main difference is that supervised approach to dimensionality reduction also uses class information (Mirjalili & Raschka, 2017).

dataset. The top ranked features may hence exhibit highly dependence or **multicollinearity**. From the perspective of predictive modelling having multicollinear features does not affect the predictive capability of the model. Additionally, regularization used in ML usually stabilizes the coefficients, which tames its effects. Basically, multicollinearity seem to be an issue when we are interested in the underlying model structure (the case in traditional statistics) as it can lead to wrong insights when trying to interpret the model for business purposes (Gunipati, 2018). Although we are mainly concerned with the predictive performance in our study there is little use in having highly multicollinear features in the model, especially so when considering the costs of data collection. A highly effective way of dealing with multicollinearity is the use of variance inflation factor (hereinafter: VIF) that quantifies its severity using ordinary least squares regression analysis. A simple rule of thumb is to exclude the variable if its VIF is higher than 10 (Zhou, Lu & Fujita, 2015). We coded a sequential VIF algorithm that iteratively calculates the extent of multicollinearity for each feature in the dataset, excludes the one with the highest VIF, and repeats that until all VIFs are under the threshold. Doing so we excluded the following financial ratios from the dataset: Net profit margin from sales, Quick ratio, Equity ratio, and ST debt to total assets. In accordance with the defined ECRM workflow defined in Figure 9 we then applied eight dimensionality reduction techniques on the under-sampled sets to achieve higher computational efficiency. We ranked the features by their importance based on all techniques apart from regularization and GA; the former performs the selection inside model learning phase, while the latter only outputs a binary variable, where 1 indicates the specific feature is included in the model and vice-versa for 0. GA is implemented in R using GA package. As for the PCA and LDA we derived the corresponding feature ranks based on the magnitudes of their coefficients (loadings) in the derived components. The Figure 12 below shows the 15 **most important features** in our dataset based on the average rank calculated from the individual dimensionality reduction techniques.

Figure 12: 15 Most Predictive Financial Ratios Based on Dimensionality Reduction



Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Another interesting fact is the **category aspect** of the top 15 financial ratios. Namely, almost half of the ratios comes from the profitability category, which points at the importance of company's profitability in forecasting default events. Following are the liquidity and leverage ratios. Looking at the whole range of selected financial ratios the Table 16 below summarizes the importance of categories derived from the ranks we assigned to the induvial ratio – the lower the rank the more important the feature. We averaged together the ranks of all financial ratios in a specific category in order to eliminate the size effect. Similarly, profitability ratios seem to have by far the highest predictive power, followed by leverage ratios, growth indicators, and liquidity ratios. Interestingly, cash flow related ratios seem to be the least important in predicting PD.

Table 16: Importance of the Category in Predicting Default Event

Category	Averaged Rank
Profitability ratios	8.89
Leverage ratios	17.75
Growth indicators	19.00
Liquidity ratios	22.71
Efficiency ratios	23.80
Investment ratios	25.40
Cash flow related ratios	27.25

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

For a more detailed overview of the feature subsets selected by dimensionality reduction techniques please refer to Appendix 11. We have to note however that not all of the top 15 features are included in the final model; we only selected the top 10 features in order to get more transparent models. Furthermore, the final inclusion of an individual feature is method specific and depends on the learning phase, which is discussed next.

3.3 Learning Phase – Model Selection and Performance Evaluation

The generalization performance of a learning method to the unseen data is of primal importance in ECRM. This chapter thus discusses the practices of building good predictive models by fine-tuning their hyper-parameters[30] (i.e. model selection) and finally evaluating the models' performance (i.e. model evaluation) using four resampled datasets and eight feature subsets. Firstly, we have conducted a **simple ratio analysis** comparing the groupwise distributions of the defaulted and non-defaulted companies in our sample. The second table in Appendix 9 summarizes the results. We employed the Mann-Whitney U test that tests for the difference in distributions between two independent samples, i.e. defaulted vs. non-defaulted companies. It is a nonparametric, distribution free statistical significance test with

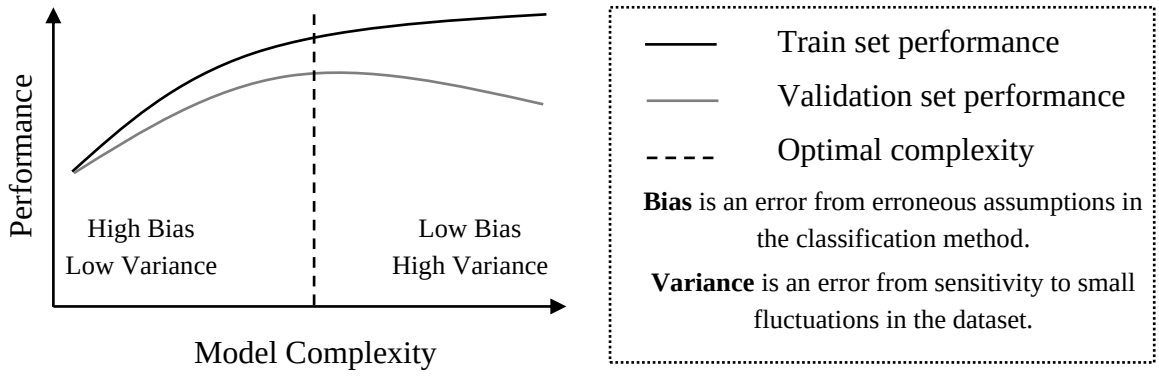
[30] Hyper-parameters refer to the parameters of a learning method that need to be optimized separately. A popular hyper-parameter optimization technique is called grid search (Albon, 2018).

the null hypothesis being that there is no difference between the distributions (Brownlee, 2018c). We can see that the null hypothesis is rejected for all financial ratios, which implies that there exists a difference in the distributions between the two groups of companies.

3.3.1 Model Selection Step

The first step in model learning is **model selection**, where different hyper-parameter settings of a given classification method are tuned and compared. The ultimate aim of this step is to find an **optimal bias-variance trade-off**; a term we have come across when different ML methods were introduced. We discuss bias and variance in Appendix 8 – Lesson 3, where we mention that the more complex the model is (i.e. the more it tries to fit the data) the lower bias we have at the cost of high variance which results in model overfitting and vice-versa, simpler models usually have high bias and low variance which results in underfitting – both phenomena lead to low generalization ability on the unseen data which is not desirable (Hastie, Tibshirani & Friedman, 2017). A useful tool to diagnose the issue of overfitting and underfitting are validation curves that plot train and validation set performances as functions of model complexity as seen in Figure 13 below.

Figure 13: Bias-variance Trade-off Diagnosis using Validation Curves



Source: Hastie, Tibshirani and Friedman (2017); Mirjalili and Raschka (2017); Own work.

One way of finding the optimal model complexity is via hyper-parameter tuning that directly controls the complexity of the model and thus helps us efficiently select the best model given the learning method at hand. As mentioned in previous sections we use **k-fold CV for model selection**. In order to select the best performing hyper-parameter set we must validate the fitted models in each CV iteration using performance metrics.

3.3.2 Performance Evaluation Step

The last and most important step is the evaluation of model performance. To assess the ability of classifiers' to generate accurate predictions, we employ three different evaluation metrics. Often the default performance measure (i.e. classification accuracy) used by Scikit-Learn algorithms is not the most appropriate metric to use. In the case of imbalanced data, it makes sense to focus on other methods as discussed in chapter 2.4. Table 17 lists evaluation

metrics we employ in this study for the purpose of estimating and comparing the generalization performance. As we see in the next section model performance is evaluated in the CV iterations to select the best model as well as on the separate test set for the purpose of comparing different statistical and ML methods. For the final comparison of implemented statistical and ML methods in the discussion section we use AUROC metric.

Table 17: Performance Evaluation Metrics

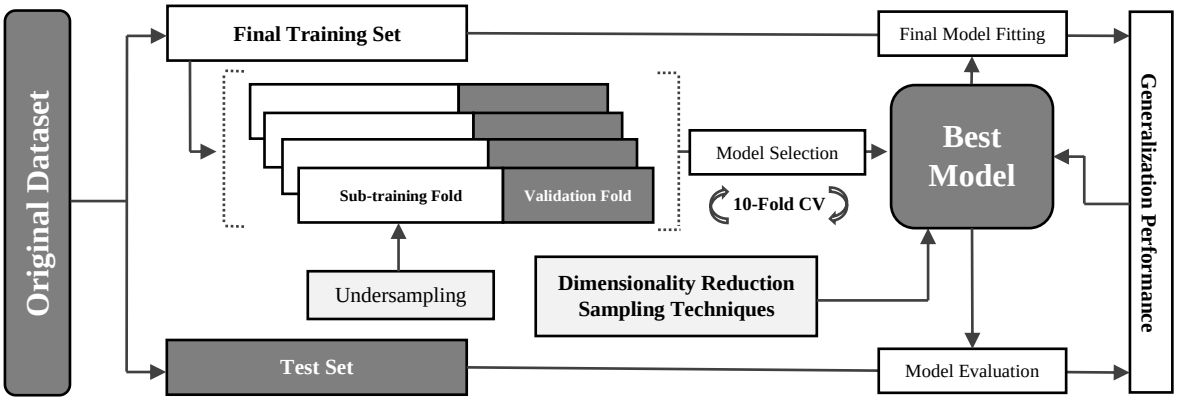
Evaluation Metric	Metric Type	Scikit-Learn Module
Classification accuracy	Correctness of categorical predictions	*metrics.accuracy_score*
Average precision	Exactness of positive predictions	*metrics.average_precision_score*
AUROC	Discriminatory ability	*metrics.roc_auc_score*

Source: Mirjalili and Raschka (2017); Thomas, Crook, and Edelman (2017); Own work.

3.3.3 Application of Model Learning on Our Problem Domain

The Figure 14 below describes the third step of our general workflow in a more detailed manner. So far, we have discussed the DPP steps including dimensionality reduction as well as specified particular methods and evaluation metrics. Instead of going through the described modelling steps separately, we combine them inside the pipeline function – a frequently used wrapper tool for chaining together individual data operations when employing ML (Mirjalili & Raschka, 2017). This chain of operations that includes data transformers (e.g. imputation of missing values, scaling, undersampling) and estimators (e.g. statistical and ML methods) was then fed into the 10-fold CV. The model selection was conducted on under-sampled sub-training sets using only methods' specific regularization as a dimensionality reduction. The fact that we decided to use these settings as our baseline scenario has to do with the high computational burden in the case, we would want to train thousands of models on an over-sampled data using all described dimensionality reduction techniques. Regularization is usually the preferred technique to combat complexity in ML models. Each method has its own regularization hyper-parameters; we do not discuss them here. However, you may find optimal settings in Table 18 on the next page.

Figure 14: Detailed Model Learning Phase Workflow



Source: Mirjalili and Raschka (2017); Zhou, Lu, and Fujita (2015); Own work.

The performance metric used in model selection phase was average precision score that is derived from the precision-recall curve (similar to ROC curve). After the best hyper-parameters on a baseline dataset were chosen for each method, we performed learning on the whole (final) training set in order to provide as much information to the learning methods as possible and finally evaluated them on the test set using the selected performance evaluation metrices. The results with the specifications on the best hyper-parameter settings as well as the test set performance values for each method are summarized in the Table 18 below. As seen from the table there are three categories of classifiers we used, namely statistical, individual ML, and ensemble ML methods. This table serves as a starting point for our discussion of the first and second research questions of the study that follows in section 3.5.1.

Table 18: Summary of Methods' Hyper-parameter Settings and Performance Evaluation

Classification Method		Hyper-parameters	Classification Accuracy	Average Precision	AUROC
Statistical Method	**Logistic Regression (LR)**	- Penalty: L2 - Regularization strength (C): 0.01	0.864	0.086	**0.868**
Individual ML Methods	**k-Nearest Neighbours (k-NN)**	- No. of neighbours: 25 - Distance metric: Minkowski - Power (p): 2 - Weight function: distance	0.862	0.067	**0.854**
	Decision Tree (DT)	- Criterion: Gini impurity - Max depth: 5	0.799	0.039	**0.835**
	Support Vector Machines (SVMs)	- Regularization strength (C): 0.05 - Kernel: linear	0.830	0.088	**0.872**
	Artificial Neural Networks (ANNs)	- No. of hidden layers: 3 - No. of nodes per layer: 100 - Dropout regularization: 0.5 - Optimizer: ADAM - Epochs: 100 - Mini-batch size: 250	0.806	0.090	**0.878**
Ensemble ML Methods	**Random Forest (RF)**	- Criterion: Gini impurity - No. of estimators: 600 - Max depth: 6	0.831	0.079	**0.886**
	AdaBoost – DT	- Base estimator: Decision tree - Learning rate: 0.5 - No. of estimators: 750 - Max depth: 5	0.836	0.124	**0.896**
	Gradient Boosting	- Base estimator: Decision tree - Learning rate: 0.01 - No. of estimators: 700 - Max depth: 3	0.839	0.127	**0.904**

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

3.4 Experiment Setup Related to the Other Research Questions

Before turning out attention to the results of this empirical analysis we should discuss three more issues related to the research questions set in the introductory part of the study – the choice of the optimal sampling technique, the impact of dimensionality reduction on generalization performance, and the inclusion of macroeconomic variables into the model. We analyse the relationships using the best hyper-parameter model settings for each method selected in the preceding step. While this might not provide us with the optimal results, it should give an indication of how specific modelling setting influences the overall generalization performance.

Firstly, we fitted each of the eight methods on the original (i.e. imbalanced) dataset. Then we repeated the exercise on the oversampled data as well as on the dataset that was resampled using SMOTEEN technique. The results from the respective analysis may be found in Appendix 13. The table summarizes the performance metrics per each classification method for four different resampling strategies, i.e. leaving the dataset imbalanced, undersampling and oversampling the dataset, and finally employing the combined SMOTEEN approach. Secondly, similar procedure was repeated for each of the six feature selection and two feature extraction approaches. The results are summarized in Appendix 14. Additionally, we looked at the implied feature importance based on the weights assigned to specific features by the classification methods. Lastly, we introduced the macroeconomic variables into the model. We assigned the values to each instance (i.e. firm-year) in the dataset in a pooled cross-sectional setting. As discussed, the sample window in this study ranges from 2013 to 2017 (i.e. we use stack sampling approach as discussed in section 1.3.1), which means that we gathered the macroeconomic variables for the defined period and applied similar DPP steps as in the case of financial ratios. Namely, we dealt with potential outliers and missing values, and scaled the data using standardization in order to facilitate the learning process. The descriptive statistics may be found in Table 19 below. All values are expressed in percentages.

Table 19: Macroeconomic Variables Descriptive Statistics

Variable[a]	Count	Mean	Std. Dev.	Min	25%	50%	75%	Max
e01	101,880	2.49	1.95	-1.10	2.30	3.00	3.10	4.90
e02	101,880	8.65	1.27	6.60	8.00	9.00	9.70	10.10
e03	101,880	2.56	1.83	1.11	1.12	1.64	3.19	5.99
e04	101,880	0.55	0.88	-0.50	-0.10	0.20	1.40	1.80
e05	101,880	5.60	10.31	-11.22	3.15	4.18	12.38	19.59

[a] Input variables used for default prediction: e01 – GDP growth, e02 – Unemployment rate, e03 – Government yield, e04 – Inflation rate, e05 – SBITOP index growth.

Source: Statistical Office of the Republic of Slovenia (2019); Investing.com (2019); Own work.

3.5 Results and Discussion

Credit scoring models are important for companies as risk management tools for preventing losses from suppliers with poor creditworthiness as argued in the theoretical part of the study. Customers' individual probabilities of default on payments are generally estimated using historical information from annual financial statements. The standard approach for estimation of these probabilities is LR. However, since we are dealing with the classification problem ML methods have proved to be useful. In this study we have used the standard LR (statistical method) as well as the ML methods to estimate the default probabilities. Next subsections discuss the results of the empirical analysis. Since we are dealing with the highly imbalanced dataset and thus comparing the model performance using metrics other than the plain classification accuracy it is important to have in mind the baseline values given by the **dummy classifier** that makes predictions using simple rules such as always predicting the most frequent class in the dataset. The baseline values are summarized in the Table 20 below. We focus on the area under the ROC curve performance measure since it is widely used in the ML literature for comparing different models and shows classifier's predictive performance in separating the two classes.

Table 20: Dummy Classifier Performance Baseline Values

Classification Method	Hyper-parameters	Classification Accuracy	Average Precision	AUROC
Dummy Classifier	Strategy: most frequent	0.992	0.008	**0.500**

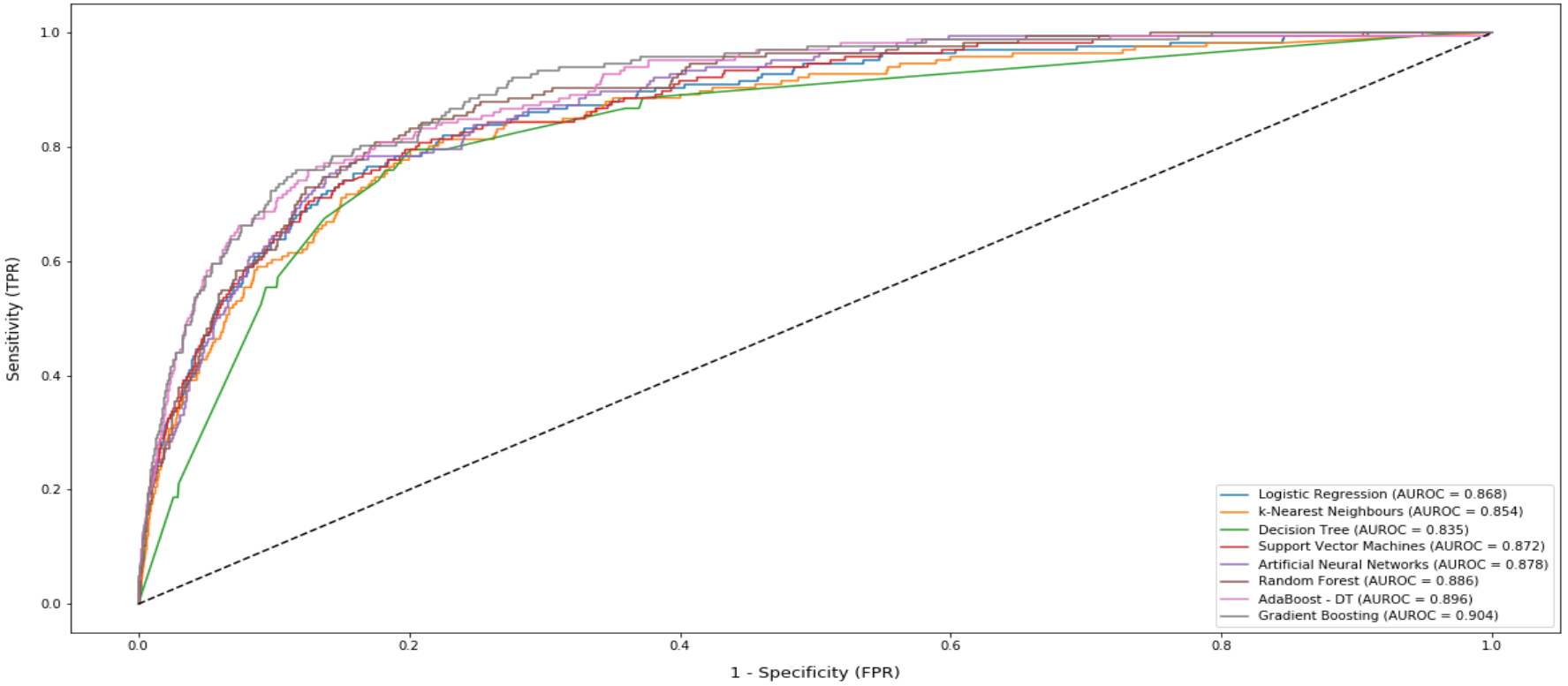
Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

3.5.1 Comparison of Traditional Statistical and Machine Learning Methods

First, we compare the performance of LR with other **classification methods originating from ML** field. The experimental results are summarized in Table 18. Additionally, Figure 15 on the next page depicts ROC curves for the classifiers at hand. We can observe that the classification accuracy for all classification methods is much lower than 0.992 achieved by the dummy classifier. However, as already argued looking solely at accuracy can be misleading. Firstly, classification accuracy gives us the performance evaluation at only one threshold (i.e. 0.5) which might not be the most optimal critical value from the business perspective of model errors. We should thus employ a more comprehensive measure. Secondly, sometimes it may be desirable to select a model with a lower accuracy but higher discriminatory ability. Hence, we compare the methods using AUROC curve metric. Looking at the AUROC value of 0.868 for the LR we can conclude that the statistical method performs relatively well compared to the ML methods. The worst performing classification methods are k-NN and DTs with AUROC values of 0.854 and 0.835, respectively. This is in line with the results of He, Zhang, and Zhang (2018) and Hand and Henley (1997). On the other hand, SVMs and ANNs both slightly outperform the LR method. Employing SVMs increases the overall prediction performance measured via AUROC metric by a mere 0.004 while the increase in performance using ANNs amounts to 0.010. While there is an evidence

Figure 15: Comparison of ROC Curves Across Different Classification Methods

The figure below plots ROC curves of the LR and other ML methods discussed in this study. We can observe that DTs and k-NN classifiers perform the worst at almost every threshold. Much better performance is achieved with LR, SVMs, and ANNs. As expected, ensemble methods (RF, AdaBoost – DT, Gradient Boosting) provide a further improvement in the AUROC metric (curves are more to the left).



Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

for the case that more advanced ML methods such as SVMs and ANNs lead to an increase in model generalization performance on our problem domain this increase is rather small compared to prior research. However, there is another group of promising methods in ML that perform considerably well according to the ECRM research, namely ensemble learning methods.

3.5.2 The Generalization Ability of Ensemble Learning

Lessmann, Baesens, Seow, and Thomas (2015) argue that on average **ensemble methods** outperform both the statistical methods as well as the individual ML classifiers. This is also the case in our study as seen in Table 18. In credit risk modelling section, we said that the DT method usually overfits the training data which leads to poor generalization performance – we saw that DT method performed the worst on our CS dataset. As suggested by Breiman (2001b) using bagging ensemble technique can significantly improve the variance of such unstable learner. Consequently, we first implemented the RF ensemble method that combines a large collection of individual trees in a parallel fashion. Furthermore, weak learners can also be trained sequentially using AdaBoost and Gradient Boosting algorithms. They both try to reduce the bias and variance as discussed earlier. Table 21 below reports the improvements of using ensemble methods over the DTs and LR. RF achieves the AUROC of 0.886, which amounts to a 6.11% increase in the metric over the DTs and 2.07% increase over the LR. Both boosting ensemble methods outperformed the RF with gradient boosting algorithm scoring the highest AUROC value of 0.904. This presents an 8.26% increase in the performance over the DTs and 4.15% increase over the LR. The results in favour of ensemble ML methods were robust at multiple iterations of performance evaluation. Appendix 12 presents ROC curves of the three ensemble methods discussed in this study and compares them with the ROC curves of DT and LR methods. It provides a further validation of the superiority of ensemble ML classifiers. RF, AdaBoost, and gradient boosting outperform individual classifiers at each threshold, i.e. the ROC curves depicting the ensemble classifiers are always left of the orange and blue lines that plot DT and LR methods' performance.

Table 21: Improvements in AUROC Metric when Using Ensemble Methods

Ensemble Method	Improvement in AUROC Over DTs	Improvement in AUROC Over LR
Random Forest	+ 0.051 (6.11%)	+ 0.018 (2.07%)
AdaBoost – DT	+ 0.061 (7.31%)	+ 0.028 (3.23%)
Gradient Boosting	+ 0.069 (8.26%)	+ 0.036 (4.15%)

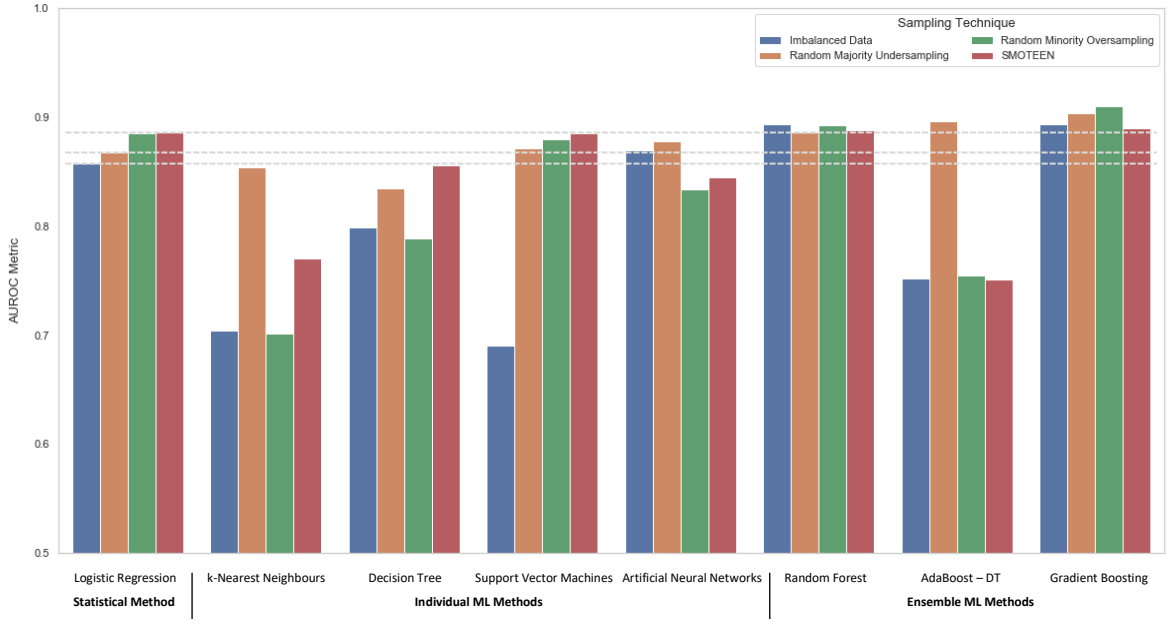
Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

3.5.3 On the Optimality of Employed Sampling Techniques

Credit scoring datasets usually exhibit low proportions of defaulted customers. The proportion of defaulted customers in our case is 0.8%, which means that our dataset is highly imbalanced. It has been shown that such class imbalance impedes classification, so various

under-/oversampling techniques have been proposed in the literature (Batista, Prati & Monard, 2004; Chawla, Bowyer, Hall & Kegelmeyer, 2002; Weiss & Provost, 2003). As discussed in the section 3.1.5 we employed three different **data sampling techniques**. The detailed results of the experiment may be found in Appendix 13. For easier interpretation Figure 16 below plots the AUROC metric for each classification method given four differently resampled datasets. When averaging together the methods' performance values across the four sampling techniques we get the following results. The highest average performance is achieved on the under-sampled dataset (AUROC = 0.874), following is the combination technique known as SMOTEEN (AUROC = 0.846). Oversampling technique and leaving the data imbalanced score 0.831 and 0.808, respectively. Furthermore, when looking at each method separately we can observe different patterns. In summary, k-NN, SVMs, and AdaBoost – DT seem to perform poorly on imbalanced data and require resampling; LR and DTs perform better but still worse compared to the other three sampling approaches. On the other hand, ANNs and the two ensemble methods (an exception is AdaBoost) seem to be insensitive to the class distribution which is in agreement with the results from prior studies (Brown & Mues, 2012; He, Zhang & Zhang, 2018). LR, DTs, and SVMs perform the best with the SMOTEEN sampling technique.

Figure 16: Methods' Performance Evaluation Using Different Sampling Techniques



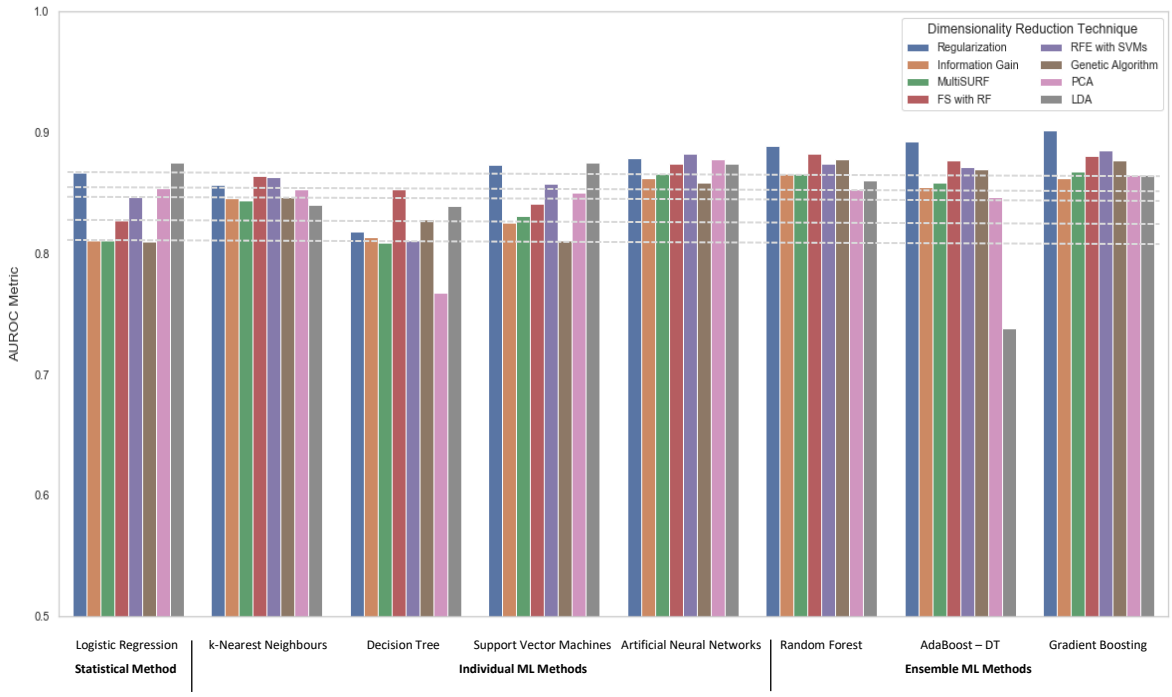
Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

3.5.4 Selection of the Optimal Feature Subset using Dimensionality Reduction

Our original dataset includes 45 independent variables; however, empirical studies have reported that usually a smaller subset of the most predictive ones may the optimal solution. As there is no pre-agreed set of variables, we should employ either feature selection or feature extraction techniques as discussed in section 3.2. The detailed results of the experiment may be found in Appendix 14. For easier interpretation Figure 17 below plots the AUROC metric for each classification method given eight different **dimensionality**

reduction techniques. Looking at the figure the most important conclusion is that embedded feature selection technique (i.e. method's specific regularization) performs the best on our dataset; the average AUROC metric value of implemented methods in the case of regularization is 0.872, following are the feature selection with RF (filter method) and recursive feature elimination using SVMs (wrapper method), which both score 0.862. The other methods have similar average performance, so it is impossible to distinguish them. It seems that in ML it is preferred to leave dimensionality reduction to the classification methods that find the best combination of feature parameters using regularization hyper-parameters. However, in real business applications data collection is usually costly, which implies that using dimensionality reduction before learning phase might be the best option even though there is a slight decrease in the classifier's prediction performance. Our experimental results show that there is no such thing as "the single best" dimensionality reduction technique. It is essential that we carefully choose the technique given the classification method we would like to use for predicting the PD.

Figure 17: Methods' Performance Evaluation Using Different Dimensionality Reduction Techniques



Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Based on the results in Appendix 14 and Figure 17 the effect of performing dimensionality reduction on classification methods is as follows:

(i) LR and SVMs combined with LDA feature extraction perform slightly better than the baseline classifiers combined with regularization. Employing other dimensionality reduction techniques perform worse than the baseline.

(ii) k-NN classifier combined with feature selection using RF (filter method) and RFE with SVMs (wrapper method) performs better than the baseline k-NN. Since k-NN does not include any pre-processing of the input features and is based on calculating distances, performing feature selection can have positive impact on its prediction performance.
(iii) DTs employ the feature selection step during the learning phase of the classifier so usually it does not require a separate dimensionality reduction. However, it seems that selecting the feature subset using RF yields the best solution in our problem domain.
(iv) ANNs exhibit similar prediction performance across all dimensionality reduction techniques with RFE using SVMs scoring the best predictive performance.
(v) Ensemble methods behave similarly across the eight dimensionality reduction techniques. Using embedded method (regularization) seems to be the best choice in our case, following are the feature selection with RF and RFE using SVMs. Employing LDA before training AdaBoost – DT classifier yields the worst predictive performance.

There is one more issue we must look at when discussing optimal feature subsets. Since embedded method produced the best prediction results on average, it would be interesting to see how the optimal feature subset selected by this technique compares to the subset in Figure 12 where we discussed other dimensionality reduction techniques. We calculated the normalized weights from individual classifiers that support either the calculation of implied feature importance from corresponding coefficients or directly implements feature importance method. As seen in Appendix 15 there is a lot of overlapping in the **top 15 variables**. Almost half of the ratios come from the profitability category, which again points at the importance of company's profitability in forecasting default events. Following are the liquidity, efficiency, and investment ratios. Looking at the individual variables trade credit exposure ratio seems to have the most important role in predicting the PD, followed by net profit per employee and return on equity.

3.5.5 Inclusion of the Macroeconomic Variables into the Model

In theoretical section of the study we argued that there exist significant risks for the normal business operations coming from the external environment. Macroeconomic factors should thus have an influence on the PD. Bellotti and Crook (2009) show that inclusion of **macroeconomic variables** into the model improves prediction performance. As described in section 3.4 we included five macroeconomic variables, i.e. GDP growth, Unemployment rate, Government yield, Inflation rate, and SBITOP index. When comparing model results there seems to be no improvement in the AUROC metric for any of the implemented methods. Appendix 16 lists the implied importance of the 15 least important features. We can see that all macroeconomic variables fall in this group. We can thus conclude that in our pooled cross-sectional setting macroeconomic variables do not play a significant role in predicting defaults. This is not in line with the prior research that is consistent in claiming the importance of macroeconomic variables. However, articles finding their importance usually employ dynamic modelling methods (i.e. survival analysis, Markov chains) that work with panel data and thus fully account for the time-series behaviour of the failure.

CONCLUSION AND FUTURE RESEARCH

In the aftermath of one of the worst financial crisis in modern history, it has become clear that managing risk exposures across all parts of the organization is essential in order to succeed in the competitive business environment. This study focuses on the aspect of credit risk that is particularly important for a day-to-day business operation and even more so in the prolonged periods of economic distress, namely the credit risk originating from trade credit arrangements that are extended to customers buying goods and services. Such arrangements inherently possess the possibility of payment delinquency or default. Firms should therefore manage the risk by analysing the creditworthiness of their customers in order to distinguish between the ones who are more likely to pay and those who have high PD. We employed the methodology of quantitative CS with a focus on implementing and comparing ML methods for the purpose of estimating PD, which serves as a basis for the development of effective credit risk mitigation strategies.

In this study, we developed ML based model for forecasting customer's credit default up to a year prior to it happening. Furthermore, we investigated the impact of employing different data sampling and dimensionality reduction techniques as well as the inclusion of macroeconomic variables into the model. Using AUROC performance metric we first compared the predictive performance of individual ML classifiers to our baseline LR classifier. The **empirical results** suggest that LR performed relatively well on our problem domain; k-NN and DTs performed worse than LR, while SVMs and ANNs outperformed the baseline by a small margin. We can conclude that there is a mixed evidence in favour of individual ML classification methods, which is in line with prior research. The performance of proposed methods usually depends on the specific problem domain. When looking at the ensemble ML classifiers, results are more robust and indicate that ensembles outperform both the LR method as well as the individual ML classifiers. We also analysed the behaviour of undersampling, oversampling, and SMOTEEN sampling techniques. Our results provide evidence that undersampling performs the best followed by SMOTEEN, while leaving the dataset imbalanced results in worst prediction performance on average. This contradicts results from prior literature that finds oversampling the most optimal choice. We can confirm however, that ANNs and ensemble methods seem to be insensitive to the class distribution. In addition, we examined the effect of dimensionality reduction in CS. Specifically, when comparing embedded, filter, and wrapper feature selection approaches, we found out that embedded technique (i.e. regularization) performs the best on our dataset, whereas there is no distinction in predictive performance between filter and wrapper techniques. Furthermore, our findings suggest that there is no such thing as the best dimensionality reduction technique. It is essential that we carefully choose the technique given the classification method we would like to implement. Lastly, there seems to be no added value in including the macroeconomic variables into the model using one-year (i.e. pooled cross-sectional) setting although these variables should have an important effect as suggested by prior studies.

In hindsight, ML approaches can be easily used for the estimation of individual (i.e. transactional) PD for a range of corporate customers as shown in this study. Fast, reliable, and simple to use implementations of the proposed methods are readily available and should therefore be considered serious competitors of the classical LR method, especially the ones that also improve the model's predictive performance. Although we have carried out a comprehensive study of various techniques used in quantitative CS, there are some **research limitations** with respect to the workflow of the empirical analysis. Firstly, model selection step was carried out on the under-sampled data using regularization as a baseline scenario; the selected optimal hyper-parameter set was then used throughout the rest of the empirical analysis. The reason behind this decision was that selection of the best set of hyper-parameters for each specific combination of classification method, sampling, and dimensionality reduction technique would be computationally to intensive given our resources as we would have to train and evaluate thousands of different models on big datasets. Secondly, according to a study carried out by Varma and Simon (2006) using nested CV provides better (almost unbiased) estimate of model performance relative to the holdout test set approach we used in the study. Thirdly, we employed a pooled cross-section analysis which might not be the best for evaluating the importance of macroeconomic variables; it would be advisable to apply panel data approach in order to infer more relevant results.

With regards to the **future research**, we recommend carrying out a similar empirical analysis on the dataset provided by a specific company. This would allow developer to work on a less imbalanced data as payment delinquency based definition of default is far less stringent than the one we used in our study (i.e. bankruptcy proceeding). Furthermore, it would enable the ML methods to be trained on a higher number of default instances, which has proven to be helpful in trying to improve predictive performance. Additionally, such analysis would enable inclusion of highly relevant past payment behaviour variables and other qualitative information about the customers. The kind of model developed for company's specific purpose also opens way to an important strategy of directly introducing the profit maximization to the model development. One approach is to modify the dependent variable in order to reflect profitability and change the nature of the task from classification to regression. Another approach toward profit scoring is based on using profit-related performance measures for model selection such as expected maximum profit that is calculated from the costs and benefits of extending customer credits. This would provide us with a direct answer to an important managerial question to what degree accuracy improvements actually add to the bottom-line (Kozodoi, Lessmann, Papakonstantinou, Gatsoulis & Baesens, 2019). Finally, it would be logical to extend the study to include some newer ensemble classification methods that have received considerable academic attention in recent years. For example heterogenous ensembles were not discussed in the study, but tend to perform considerably well in CS applications according to the paper by Lessmann, Baesens, Seow, and Thomas (2015). Another improvement might be the inclusion of textual and similar less structured data into the ML models.

REFERENCE LIST

1. Aggarwal, C. C. (2018). *Neural Networks and Deep Learning.* New York: Springer.
2. Alaka, H. A., Oyedele, L. O., Owolabi, H. A., Kumar, V., Ajayi, S. O., Akinade, O. O. & Bilal, M. (2018). Systematic Review of Bankruptcy Prediction Models: Towards a Framework for Tool Selection. *Expert Systems With Applications, 94,* 164–184.
3. Albon, C. (2018). *Machine Learning with Python Cookbook - Practical Solutions from Preprocessing to Deep Learning.* Sebastopol: O'Reilly.
4. Alexander, C. & Sheedy, E. (2004). *The Professional Risk Managers' Handbook: A Comprehensive Guide to Current Theory and Best Practices (Vol. III: Risk Management Practices).* Wilmington: PRIMIA Publications.
5. Alpaydin, E. (2004). *Introduction to Machine Learning.* Cambridge MA: The MIT Press.
6. Altman, E. I. (1968). Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. *The Journal of Finance, 23*(4), 589–609.
7. Altman, E. I., Haldeman, R. G. & Narayanan, H. P. (1977). ZETA Analysis: A New Model to Identify Bankruptcy Risk of Corporations. *Journal of Banking & Finance, 1*(1), 29–54.
8. Altman, E. I., Sabato, G. & Wilson, N. (2008). *The Value of Qualitative Information in SME Risk Management.* New York: New York University.
9. Anderson, R. (2007). *The Credit Scoring Toolkit - Theory and Practice for Retail Credit Risk Management and Decision Automation.* Oxford: Oxford University Press.
10. Badr, W. (2019, Mar 5). *5 Ways to Detect Outliers/Anomalies That Every Data Scentist Should Know*. Retrieved August 15 2019 from https://towardsdatascience.com/5-ways-to-detect-outliers-that-every-data-scientist-should-know-python-code-70a54335a623
11. Balcaen, S. & Ooghe, H. (2006). 35 Years of Studies on Business Failure: An Overview of The Classic Statistical Methodologies and Their Related Problems. *The British Accounting Review, 38*(1), 63–93.
12. Batista, G. E., Prati, R. C. & Monard, M. C. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explorations Newletter, 6*(1), 20–29.
13. Bazzi, M. & Hasna, C. (2015). Rating Models and Its Applications: Setting Credit Limits. *Journal of Applied Finance & Banking, 5*(5), 201–216.
14. Beaver, W. H., McNichols, M. F. & Rhie, J.-W. (2005). Have Financial Statements Become Less Informative? Evidence from the Ability of Financial Ratios to Predict Bankruptcy. *Review of Accounting Studies, 10*(1), 93–122.
15. Bellotti, T. & Crook, J. (2009). Credit Scoring with Macroeconomic Variables Using Survuval Analysis. *The Journal of the Operational Research Society, 60*(12),1699–1707.
16. Bengio, Y., Courville, A. & Goodfellow, I. (2016). *Deep Learning* . Cambridge MA: The MIT Press.

17. Bengio, Y., Hinton, G. & LeCun, Y. (2015). Deep Learning. *Nature, 521*, 436–444. Retrieved Februar 20 2019 from https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf
18. Beque, A., & Lessmann, S. (2017). Extreme Learning Machines for Credit Scoring: An Empirical Evaluation. *Expert Systems With Applications, 86*, 42–53.
19. Berk Skok, A. (2016). *Risk Management: The Big Picture.* Ljubljana: School of Economics and Business, UL.
20. Breiman, L. (2001a). Statistical Modelling: The Two Cultures. *Statistical Science, 3*(15), 199–231.
21. Breiman, L. (2001b). Random Forests. *Machine Learning, 45*(1), 5–32.
22. Brown, I. & Mues, C. (2012). An Experimental Comparison of Classification Algorithms For Imbalanced Credit Scoring Data Sets. *Expert Systems with Applications, 39*(3), 3466–3453.
23. Brownlee, J. (2016). *Parametric and Non-parametric Machine Learning Algorithms*. Retrieved March 10 2019 from https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/
24. Brownlee, J. (2018a). *The Close Relationship Between Applied Statistics and Machine Learning*. Retrieved March 10 2019 from https://machinelearningmastery.com/relationship-between-applied-statistics-and-machine-learning/
25. Brownlee, J. (2018b). *Analytical vs Numerical Solutions in Machine Learning*. Retrieved March 11 2019 from https://machinelearningmastery.com/analytical-vs-numerical-solutions-in-machine-learning/
26. Brownlee, J. (2018c, May 16). *How to Calculate Nonparametric Statistical Hypothesis Tests in Python*. Retrieved April 24 2019 from https://machinelearningmastery.com/nonparametric-statistical-significance-tests-in-python/
27. Brownlee, J. (2019). *How to Choose Loss Functions When Training Deep Learning Neural Networks*. Retrieved June 4 2019 from https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/
28. Brynjolfsson, E. & McAfee, A. (2017a). AI, For Real: The Business of Artificial Intelligence. *Harvard Business Review*. Retrieved Februar 25 2019 from https://hbr.org/cover-story/2017/07/the-business-of-artificial-intelligence
29. Brynjolfsson, E. & McAfee, A. (2017b). AI, For Real: What's Driving the Machine Learning Explosion? *Harvard Business Review*. Retrieved Februar 25 2019 from https://hbr.org/2017/07/whats-driving-the-machine-learning-explosion
30. Bughin, J. & Manyika, J. (2018). The Promise and Challenge of the Age of Artificial Intelligence. *McKinsey Global Institute: Executive Briefing*. Retrieved Februar 15 2019 from https://www.mckinsey.com/featured-insights/artificial-intelligence/the-promise-and-challenge-of-the-age-of-artificial-intelligence

31. Bughin, J., Seong, J., Manyika, J., Chui, M. & Joshi, R. (2018). Notes From the AI Frontier: Modeling the Impact of AI on the World Economy. *McKinsey Global Institute: Discussion Paper*. Retrieved Februar 15 2019 from https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-modeling-the-impact-of-ai-on-the-world-economy
32. Cameron, E., Gillham, J., Rao, A. & Verweij, G. (2018). The Macroeconomic Impact of Artificial Intelligence. *PricewaterhouseCoopers.* Retrieved Februar 17 2019 from https://www.pwc.co.uk/economic-services/assets/macroeconomic-impact-of-ai-technical-report-feb-18.pdf
33. Corporate Finance Institute (CFI). (2019). *What is Trade Credit?* Retrieved April 23 2019 from https://corporatefinanceinstitute.com/resources/knowledge/other/what-is-trade-credit/
34. Chacko, G., Sjöman, A., Motohashi, H., & Dessain, V. (2016). *Credit Derivatives: A Primer on Credit Risk, Modelling, and Instruments.* New Jersey: Pearson Education, Inc.
35. Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, P. W. (2002). SMOTE: Synthetic Minority Over-Sampling Technique. *Journal of Artificial Research, 16*, 321–357.
36. Chen, F.-L. & Li, F.-C. (2010). Combination of Feature Selection Approaches with SVM in Credit Scoring. *Expert Systems with Applications, 37*(7), 4902–4909.
37. Chen, J. (2018, 22 Januar). *Credit Derivative*. Retrieved April 23 2019 from Investopedia: https://www.investopedia.com/terms/c/creditderivative.asp
38. Chollet, F. (2018). *Deep Learning with Python.* New York: Manning Publications Co.
39. Chudson, W. A. (1945). *The Pattern of Corporate Financial Structure: A Cross-Section View of Manufacturing, Mining, Trade, and Construction.* Massachusetts: NBER Books.
40. Companies Act/ZGD. (2006, April 4). *Official Journal, 42*. Retrieved August 10 2019 from http://www.pisrs.si/Pis.web/pregledPredpisa?id=ZAKO4291
41. Coussement, K. & Buckinx, W. (2011). A Probability-Mapping Algorithm for Calibrating the Posterior Probabilities: A Direct Marketing Approach. *European Journal of Operational Research, 214*(3), 732–738.
42. Cramer, J. S. (2003). *Logot Models - From Economics and Other Fields* . Cambridge: Cambridge University Press.
43. Crone, S. F. & Finlay, S. (2012). Instance Sampling in Credit Scoring: An Empirical Study of Sample Size and Balancing. *International Journal of Forecasting, 28*(1), 224–238.
44. Crone, S. F., Lessmann, S. & Stahlbock, R. (2006). The Impact of Preprocessing on Data Mining: An Evaluation of Classifier Sensitivity in Direct Marketing. *European Journal of Operational Research, 173*(3), 781–800.
45. Crosbie, P. & Kocagil, A. (2003). *Modelling Default Risk: Modelling Methodology.* New York City: Moody's KMV Company.
46. Curtis, P. & Carey, M. (2012). *Risk Asseessment in Practice.* Durham: COSO.

47. Demšar, J. (2006). Statistical Comparison of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research, 7*, 1–30.
48. Denis, D. J., Denis, D. K. & Sarin, A. (1997). Agency Problems, Equity Ownership, and Corporate Diversification. *The Jorunal of Finance, 52*(1), 135–160.
49. Derksen, L. (2016, Oct 29). *Visualizing High-dimensonal Datasets using PCA and t-SNE in Python*. Retrieved August 18 2019 from https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b
50. Domingos, P. (1999). The Role of Occam's Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery, 3*(4), 409–425.
51. Domingos, P. (2012). A Few Useful Things to Know about Machine Learning. *Communications of the ACM, 55*(10), 78–87.
52. Durand, D. (1941). *Risk Elements in Consumer Instalment Financing.* Massachusetts: NBER Books.
53. Elizondo Flores, J. A., Lemus Basualdo, T. & Quintana Sordo, A. R. (2010). *Regulatrory Use of System-Wide Estimations of PD, LGD, and EAD.* Basel: Bank For International Settlements.
54. European Banking Authority. (2018, March 19). *EBA Report on the Credit Risk Mitigation (CRM) Framework.* Retrieved March 3 2019 from https://eba.europa.eu/documents/10180/2087449/EBA+Report+on+CRM+framework.pdf
55. Financial Operations, Insolvency Proceedings, and Compulsory Dissolution Act/ZFPPIPP. (2007, 12 17). *Official Journal, 13*. Retrieved August 10 2019 from http://pisrs.si/Pis.web/pregledPredpisa?id=ZAKO4735
56. Fisher, R. A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics, 7*(2), 1–11.
57. Fitzpatrick, P. J. (1932). A Comparison of The Ratios of Successful Industrial Enterprises with Those of Failed Companies. *The Certified Public Accountant*, 72–731.
58. Fletcher, T. (2008). *Support Vector Machines Explained.* Retrieved Februar 27 2019 from https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf
59. Florez-Lopez, R. (2010). Effects of Missing Data in Credit Risk Scoring: A Comparative Analysis of Methods to Achieve Robustness in the Absence of Sufficient Data. *The Journal of the Operational Research Society, 61*(3), 486–501.
60. Freund, Y. & Schapire, R. E. (1999). A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence, 14*(5), 771–780.
61. Garcia, V., Marques, A. & Sanchez, J. (2018). A Literature Review on the Application of Evolutionary Computing to Credit Scoring. *The Journal of the Operational Research Society, 64*(9), 1384–1399.
62. Garcia-Teruel, P. J. & Martinez-Solano, P. (2010). Determinants of Trade Credit: A Comparative Study of European SMEs. *International Small Business Journal, 28*(3), 215–233.

63. Geron, A. (2017). *Hands-On Machine Learning with Scikit-Learn & TensorFlow.* Sebastopol: O'Reilly Media, Inc.
64. Gibson, M. S. (2007). Credit Derivatives and Risk Management. *FEDS Working Papers, No. 2007-47*.
65. Gissel, J. L., Giacomino, D. & Akers, M. D. (2007). A Review of Bankruptcy Prediction Studies: 1930-Present. *Journal of Financial Education, 33*, 1–42.
66. Goertzel, B. & Pennachin, C. (2007). *Artificial General Intelligence* . Berlin: Springer.
67. Gunipati, T. (2018, 28 Jun). *25 Machine Learning Interview Questions & Answers - Linear Regression*. Retrieved March 13 2019 from https://www.upgrad.com/blog/machine-learning-interview-questions-answers-linear-regression/
68. Hajek, P. & Michalak, K. (2013). Feature Selection in Corporate Credit Rating Prediction. *Knowledge-Based Systems, 52*, 72–84.
69. Hand, D. J. (2009). Measuring Classifier Performance: A Coherent Alternative to the Area Under the ROC Curve. *Machine Learning, 77*, 103–123.
70. Hand, D. J. & Henley, W. E. (1997). Statistical Classification Methods in Consumer Credit Scoring: A Review. *Journal of the Royal Statistical Society, 160*(3), 523–541.
71. Harasymiv, V. (2015, November 3). *Lessons From 2M Machine Learning Models on Kaggle.com*. Retrieved March 13 2019 from https://www.linkedin.com/pulse/lessons-from-2mm-machine-learning-models-kagglecom-data-harasymiv/
72. Harrington, P. (2012). *Machine Learning in Action.* New York: Manning Publications Co.
73. Hastie, T., Tibshirani, R. & Friedman, J. (2017). *The Elements of Statistical Learning (2nd ed.).* New York: Springer.
74. Hay-Gibson, N. V. (2008). A River of Risk: A Diagram of the History and Historiography of Risk Management. *Interdisciplinary Studies in the Built and Virtual Environment*, 1–7.
75. Haykin, S. (2009). *Neural Networks and Learning Machines (3rd ed.).* New Jersey: Pearson Education, Inc.
76. He, H., Zhang, W. & Zhang, S. (2018). A Novel Ensemble Method For Credit Scoring: Addaptation of Different Imbalance Ratios. *Expert Systems With Applications, 98*, 105–117.
77. Hearty, J. (2016). *Advanced Machine Learning with Python.* Birmingham: Pact Publishing.
78. Hinton, G. E., Rumelhart, D. E. & Williams, R. J. (1968). Learning Representations by Back-propragating the Errors. *Nature, 323*, 533–536.
79. Hoyt, R. E. & Lienberg, P. A. (2011). The Value of Enterprise Risk Management. *The Journal of Risk and Insurance, 78*(4), 795–822.
80. Jackson, H. M. & Schreiber-Gregory, D. (2018). *Logistic and Linear Regression Assumptions: Violation Recognition and Control.* Retrieved Februar 10 2019 from https://www.lexjansen.com/wuss/2018/130_Final_Paper_PDF.pdf

81. Jackson, R. H. & Wood, A. (2013). The Performance of Insolvency Prediction and Credit Risk Models in the UK: A Comparative Study. *The British Accounting Review, 45*(3), 183–202.
82. Jarrow, R. A. & Protter, P. (2004). Structural Versus Reduced Form Models: A New Information Based Perspective. *Journal of Investement Management, 2*(2), pp. 1-10.
83. Jayasekera, R. (2018). Prediction of Company Failure: Past, Present, and Promising Directions for the Future. *International Review of Financial Analysis, 55*(C), 196–208.
84. Kaufman, S., Perlich, C. & Rosset, S. (2011). Leakage in Data Mining: Formulation, Detection, and Avoidance. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (556–563). San Diego.
85. Kealhofer, S. (2003). Quantifying Credit Risk I: Default Prediction. *Financial Analysts Journal, 59*(1), 30–44.
86. Kennedy, K. (2013). *Credit Scoring Using Machine Learning (Doctoral Thesis).* Dublin: Dublin Institute of Technology.
87. Khandani, A. E., Kim, A. J. & Lo, A. W. (2010). Consumer Credit-Risk Models via Machine Learning Algorithms. *Journal of Banking & Finance, 34*(11), 2767–2787.
88. Kinda, O. & Achonu, A. (2012). Building a Credit Scoring Model For the Savings and Credit Mutual of the Potou Zone (MECZOP)/Senegal. *Consilience, 7*, 12–31.
89. Kononenko, I. & Kukar, M. (2007). *Machine Learning and Data Mining: Introduction to Principles and Algorithms.* Chichester: Horwood Publishing Limited.
90. Koutanaei, F., Sajedi, H., & Khanbabaei, M. (2015). A Hybrid Data Mining Model of Feature Selection Algorithms and Ensemble Learning Classifiers for Credit Scoring. *Journal of Retailing and Consumer Services, 27*, 11–23.
91. Kozodoi, N., Lessmann, S., Papakonstantinou, K., Gatsoulis, Y. & Baesens, B. (2019). A Multi-objective Approach for Profit-driven Feature Selection in Credit Scoring. *Decision Support Systems, 120*, 106–117.
92. Kruppa, J., Schwarz, A., Arminger, G. & Ziegler, A. (2013). Consumer Credit Risk: Individual Probability Estimates Using Machine Learning. *Expert Systems with Applications, 40*(13), 5125–5131.
93. Kuncheva, L. I. (2004). *Combining Pattern Classifiers - Methods and Algorithms.* New Jersey: 2004.
94. Langlois, R. N. & Cosgel, M. M. (1993). Frank Knight on Risk, Uncertainty, and the Firm: A New Interpretation. *Economic Inquiry, 31*(1), 456–465.
95. Lehmann, B. (2003). Is It Worth the While? The Relevance of Qualitative Information in Credit Rating. *EFMA Helsinki Meetings*, 1–25.
96. Lemaitre, G., Nogueira, F. & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research, 18*, 1–5.
97. Lessmann, S., Baesens, B., Seow, H.-V. & Thomas, L. C. (2015). Benchmarking State-of-the-art Classification Algorithms for Credit Scoring: An Update Research. *European Journal of Operational Research, 247*(1), 124–136.

98. Liang, D., Tsai, C.-F. & Wu, H.-T. (2015). The Effect of Feature Selection on Financial Distress Prediction. *Knowledge-Based Systems, 73*, 289–297.
99. Lin, F., Liang, D., Yeh, C.-C. & Huang, J.-C. (2014). Novel Feature Selection Methods to Financial Distress Prediction. *Expert Systems with Applications, 41*, 2472–2483.
100. Lin, S.-M., Ansell, J. & Andreeva, G. (2012). Predicting Default of Small Business Using Different Definitions of Finanical Distress. *The Journal of the Operational Research Society, 63*(4), 539–548.
101. Lutz, M. (2013). *Learning Python (5th ed.).* Sebastopol: O'Reilly.
102. Maltz, A. C., Shenhar, A. J. & Reilly , R. R. (2003). Beyond the Balanced Scorecard: Refining the Search for Organizational Success Measures. *Long Range Planning, 36*, 187–204.
103. McClure, N. (2018). *TensorFlow Machine Learning Cookbook (2nd ed.).* Birmingham: Packt.
104. McGuinness, G., Hogan, T. & Powell, R. (2018). European Trade Credit Use and SME Survival. *Journal of Corporate Fianance, 49*(1), 81–103.
105. McKinney, W. (2018). *Python for Data Analysis (2nd ed.).* Sebastopol: O'Reilly.
106. Mirjalili, V. & Raschka, S. (2017). *Python Machine Learning (2nd ed.).* Birmingham: Packt Publishing.
107. Mitchell, T. M. (1997). *Machine Learning.* New York: McGraw-Hill.
108. Mitchell, T. M. (2006). *The Discipline of Machine Learning.* Retrieved Februar 20 2019 from http://www.cs.cmu.edu/~tom/pubs/MachineLearning.pdf
109. Montrichard, D. (2018). *Reject Inference Methodologies in Credit Risk Modeling.* Toronto: SESUG Proceedings.
110. Moorcraft, B. (2018, November 12). *What Is Trade Credit Insurance?* Retrieved April 7 2019 from https://www.insurancebusinessmag.com/uk/guides/what-is-trade-credit-insurance-116054.aspx
111. Mosconi, P. (2015). *Structural Models.* Milano: Bocconi University.
112. Mullainathan, S. & Spiess, J. (2017). Machine Learning: An Applied Econometric Approach. *The Journal of Economic Perspectives, 31*(2), 87–106.
113. Myers, J. H. & Forgy, E. W. (2012). The Development of Numerical Credit Evaluation Systems. *Journal of the American Statistical Association, 58*(303), 799–806.
114. Ng, A. (2018). *Lecture Notes: Support Vector Machines.* Retrieved Februar 27 2019 from http://cs229.stanford.edu/notes/cs229-notes3.pdf
115. Ng, A. (2019). *Neural Networks and Deep Learning*. Retrieved March 2 2019 from https://www.coursera.org/learn/neural-networks-deep-learning/lecture/praGm/why-is-deep-learning-taking-off
116. Ng, A. & Katanforoosh, K. (2019). *Neural Networks and Deep Learning*. Retrieved March 8 2019 from https://www.coursera.org/learn/neural-networks-deep-learning
117. Nikolaou, N. (2018). *Introduction to AdaBoost (internal material).* Manchester: School of Computer Science, University of Manchester.
118. Ohlson, J. A. (1980). Financial Ratios and the Probabilistic Prediction of Bankruptcy. *Journal of Accounting Research, 18*(1), 109–131.

119. Peavler, R. (2019, June 25). *Calculate the Solvency, Liquidity, and Viability of Your Firm*. Retrieved May 10 2019 from https://www.thebalancesmb.com/cash-flow-ratios-for-analysis-393116
120. Petersen, M. A. & Rajan, R. G. (1997). Trade Credit: Theories and Evidence. *The Review of Financial Studies, 10*(3), 661–691.
121. Platt, J. C. (1998, April 21). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.* Retrieved 25 Febraur 2019 from https://pdfs.semanticscholar.org/59ee/e096b49d66f39891eb88a6c84cc89acba12d.pdf
122. Ravi Kumar, P. & Ravi, V. (2006). Bankruptcy Preiction in Banks and Firms via Statistical and Intelligent Techniques - A Review. *European Journal of Operational Research, 180*(1), 1–28.
123. Režnakova, M. & Karas, M. (2014). Bankruptcy Prediction Models: Can the Prediction Power of the Models Be Improved by Using Dynamic Indicators? *Procedia Economics and Finance, 12*, 565–574.
124. Rogachev, A. Y. (2008). Enterprise Risk Management in a Pharmaceutical Company. *Risk Management, 10*(1), 76–84.
125. Roos, T. (2016). *Minimum Description Length Principle.* Retrieved Februar 10 2019 from https://www.cs.helsinki.fi/u/ttonteri/pub/roosmdlencyc2016.pdf
126. Rosner, R. L. (2003). Earnings Manipulation in Failing Firms. *Contemporary Accounting Research, 20*(2).
127. Ross, B. C. (2014). Mutual Information between Discrete and Continuous Data Sets. *PLOS ONE, 9*(2), 1–5.
128. Salappa, A., Doumpos, M. & Zopounidis, C. (2007). Feature Selection Algorithms in Classification Problems: An Experimental Evaluation. *Optimisation Methods and Software, 22*(1), 199–212.
129. Salian, I. (2018, August 2). *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* Retrieved Febraur 6 2019 from https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/
130. Santos, J. & Silva, A. (2014). The Determinants of Trade Credit: A Study of Portugese Industrial Companies. *International Journal of Financial Research, 5*(4), pp. 128-138.
131. Saunders, A. & Millon Cornett, M. (2018). *Financial Institutions Management: A Risk Management Approach (9th ed.).* New York: McGrawHill Education.
132. Schreiner, M. (2004). Benefits and Pitfalls of Statistical Credit Scoring for Microfinance. *Loans and Lending Procedures*, 1–33.
133. *Scikit-learn - Nearest Neighbors*. (2019a). Retrieved Febraur 19 2019 from https://scikit-learn.org/stable/modules/neighbors.html
134. Scrucca, L. (2013). GA: Package for Genetic Algorithms in R. *Journal of Statistical Software, 53*(4), 1–37.
135. Sharma, D. & Iselin, E. R. (2003). The Relative Relevance of Cash Flow and Accrual Information for Solvency Assessments: A Multi-Method Approach. *Journal of Business Finance & Accounting, 30*, 1115–1140.

136. Sharma, P. (2018, August 27). *The Ultimate Guide to 12 Dimensionality Reduction Techniques (with Pyhon Code)*. Retrieved July 5 2019 from https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/
137. Siddiqi, N. (2017). *Intelligent Credit Scoring (2nd ed.).* New Jersey: Wiley.
138. Slovene Accounting Standards. (2016). *Receivables.* Ljubljana: Slovene Institute of Auditors.
139. Smith, R. F. & Winakor, A. H. (1935). Changes in Financial Structure of Unsuccessful Industrial Corporations. *Bureau of Business Research, 51,* 1–44.
140. Smolyakov, V. (2017). *Ensemble Learning to Improve Machine Learning Results*. Retrieved March 14 2019 from https://blog.statsbot.co/ensemble-learning-d1dcd548e936
141. Statistical Office of RS. (2019a). *Classifications and Code Lists - SKIS*. Retrieved July 17 2019 from https://www.stat.si/StatWeb/en/Methods/Classifications
142. Statistical Office of RS. (2019b). *Classifications and Code Lists - SKD*. Retrieved July 17 2019 from https://www.stat.si/StatWeb/en/Methods/Classifications
143. Sun, J., Li, H., Huang, Q.-H. & He, K.-Y. (2014). Predicting Financial Distress and Corporate Failure: A Review From the State-of-the-Art Definitions, Modelling, Sampling, and Featuring Approaches. *Knowledge-Based Systems, 57,* 41–56.
144. Šušteršič, M., Mramor, D. & Zupan, J. (2009). Conusmer Credit Scoring Models with Limited Data. *Expert Systems with Applications, 36*(3), 4736–4744.
145. Thomas, L., Crook, J. & Edelman, D. (2017). *Credit Scoring and Its Applications (2nd ed.).* Philadelphia: Siam.
146. Tong, E. N., Mues, C. & Thomas, L. C. (2012). Mixture Cure Models in Credit Scoring. *European Journal of Operational Research, 218,* 132–139.
147. Urbanowicz, R. J., Olson, R. S., Schmitt, P., Meeker, M. & Moore, J. H. (2018). Benchmarking Relief-based Feature Selection Methods for Bioinformatics Data Mining. *Journal of Biomedical Informatics, 85,* 168–188.
148. Van Hulse, J., Khoshgoftaar, T. M. & Napolitano, A. (2007). Experimental Perspectives on Learning from Imbalanced Data. *International Conference on Machine Learning* , (935–942). Corvalis.
149. Varma, S. & Simon, R. (2006). Bias in Error Estimation Using Cross-validation for Model Selection. *BMC Bioinformatics, 7*(1), 1–8.
150. Verstraeten, G. & Van den Poel, D. (2005). The Impact of Sample Bias on Consumer Credit Scoring Performance and Profitability. *The Journal of the Operational Research Society, 56*(8), 981–992.
151. Violante, A. (2018, Aug 29). *An Introduction to t-SNE with Python Example*. Retrieved August 20 2019 from https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1
152. Weiss, G. M. & Provost, F. (2003). Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction. *Journal of Artificial Intelligence Research, 19,* 315–354.

153. Wolpert, D. (1996). The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation, 8*(7), 1341–1390.
154. Woodruff, J. (2019). *The Advantages & Disadvantages of Trade Credit*. Retrieved April 27 2019 from https://smallbusiness.chron.com/advantages-disadvantages-trade-credit-22938.html
155. Wu, Y., Gaunt, C. & Gray, S. (2010). A Comparison of Alternative Bankruptcy Prediction Models. *Journal of Contemporary Accounting & Economics*, *6*(1), 34–45.
156. Zhe, W. (2017, August 7). *Roll Rate Models & Vintage Loss Models*. Retrieved April 23 2019 from http://www.epsilonofwangzhe.com/?p=488
157. Zhou, L., Lu, D. & Fujita, H. (2015). The Performance of Corporate Financial Distress Prediction Models. *Knowledge-Based Systems, 85*, 52–61.
158. Zhou, Z.-H. (2012). *Ensemble Methods - Foundations and Algorithms.* Boca Raton: CRC Press.
159. Zmijewski, M. E. (1984). Methodological Issues Related to the Estimation of Financial Distress Prediction Models. *Journal of Accounting Research*, 59–82.
160. Zupan, B. (2018). *Uvod v odkivanje znanj iz podatkov (internal material).* Ljubljana: Faculty of Computer and Information Science, UL.

APPENDICES

Appendix 1: Summary in Slovene Language

Upravljanje s tveganji je ključno za dolgoročen obstoj podjetja na trgu. Zaradi tega morajo podjetja identificirati in meriti različne finančne pa tudi nefinančne nevarnosti ter se nanje odzivati. Med podjetji je ustaljena praksa, da svoje proizvode in storitve **prodajajo na odloženo plačilo** (angl. trade credit arrangement), za kar obstaja v literaturi cela vrsta finančnih, poslovnih in operativnih motivov, npr. ohranjanje dolgoročnih odnosov s strankami, cenovna diskriminacija, zagotavljanje kakovosti proizvodov in/ali storitev, nižanje transakcijskih stroškov itn. (Garcia - Teruel & Martinez - Solano, 2010). Slovenski računovodski standardi prodajo na odloženo plačilo definirajo kot »prodajo blaga ali storitve, ki ga (jo) kupec v trenutku prenosa nanj še ne plača« (Slovene Accounting Standards, 2016). Za prodajalca to pomeni povečanje terjatev do kupcev, medtem ko se v kupčevi bilanci stanja transakcija zazna kot povečanje obveznosti do dobaviteljev. Glede na nedavno študijo imajo evropska podjetja v povprečju 14 % celotne aktive v obliki poslovnih terjatev do strank, kar predstavlja znaten del likvidnosti (McGuinness, Hogan & Powell, 2018). Tovrstna prodaja tako za vsako podjetje pomeni **tveganje neplačila** (angl. credit/default risk), kar lahko ob neupravljanju s tveganjem vodi v likvidnostne težave, vsekakor pa imajo tovrstni kreditni dogodki negativen vpliv na dobičkonosnost poslovanja. Tako je ključno, da podjetja ocenjujejo kreditno sposobnost strank v portfelju in ločijo med tistimi, ki bodo bolj ali manj verjetno poravnali svojo terjatev, kar ne nazadnje omogoča zasnovo učinkovitih strategij za upravljanje s kreditnim tveganjem.

Metodologija, ki se uporablja za ocenjevanje kreditne sposobnosti strank oziroma omogoča izračun transakcijske verjetnosti neplačila, je znana pod imenom **kvantitativna analiza kreditne sposobnosti** (angl. quantitative credit scoring) in je del širšega področja modeliranja kreditnega tveganja (Chacko, Sjöman, Motohashi & Dessain, 2016). V splošnem se modeli kreditnega tveganja razvrščajo v tri razrede: strukturni (angl. structural models), empirični (angl. empirical models) in reducirani modeli (angl. reduced form models). Najprimernejši modeli za namen ocenjevanja verjetnosti neplačila poslovnih strank so empirični. **Empirični modeli** kreditnega tveganja v osnovi primerjajo med podjetji, ki so oziroma v nekem preteklem obdobju niso bila podvržena dogodku neplačila, ter iz te primerjave izpeljejo implicitna klasifikacijska pravila, ki temeljijo na historičnih informacijah, kot so: poslovni izkazi podjetij, makroekonomske in kvalitativne spremenljivke, podatki o plačilih itn. Pri tem velja dodati, da tovrstna analiza temelji na predpostavki »prihodnja dinamika zrcali preteklo«, torej da se vzorci oziroma izpeljana pravila do neke mere ponavljajo (Alexander & Sheedy, 2004). Filozofija kvantitativne analize kreditne sposobnosti tako temelji na premisi pragmatičnosti in empirizma v smislu, da je cilj modeliranja doseči čim boljšo napovedno natančnost na neodvisnem testnem vzorcu, in to kljub slabši interpretativnosti modela. Ključne **prednosti uporabe** te analize so naslednje: (i) omogoča maksimizacijo kompromisa med pričakovano dobičkonosnostjo in tveganjem; (ii) izboljšanje ocenjevanja kreditnega tveganja in večja konsistentnost ocen verjetnosti med strankami; (iii) višja kakovost portfelja strank itn. Na drugi strani nekateri avtorji opozarjajo na **pomanjkljivosti metodologije**, npr. pristranskost učnega vzorca, nizko

število dogodkov neplačila, uporaba pristranskih kazalnikov finančnega poslovanja itn. (Kennedy, 2013; Kinda & Achonu, 2012; Schreiner, 2004; Thomas, Crook & Edelman, 2017). Glede na vrsto vhodnih podatkov pri modeliranju poznamo dve vrsti ocenjevanja kreditne sposobnosti: ocenjevanje kreditne sposobnosti i) novih (potencialnih) strank (angl. application scoring); ii) obstoječega portfelja strank (angl. behavioural scoring). Pri prvem tipično uporabimo statične podatke, zbrane ob prihodu strank, medtem ko pri drugem model temelji na preteklih podatkih, zbranih v določenem **vzorčnem oknu**; tj. oknu, ki služi za izpeljavo želenih neodvisnih spremenljivk. Na podlagi **opazovalnega okna** definiramo odvisno spremenljivko; v našem primeru je to dogodek neplačila. Presečni dan je točka, ki ločuje obe okni in definira vzorec strank, za katere bodo zbrani podatki. Obstaja možnost **večkratnega vzorčenja**, pri čemer opisani postopek ponovimo še za nekaj preteklih let; to je priporočljivo izvesti, če je v vzorcu premalo dogodkov neplačila oziroma kadar želimo dodatno vključiti še makroekonomske spremenljivke, kar v model prinese dodatno robustnost in večjo napovedno moč (Siddiqi, 2017).

Pri določanju **definicije dogodka neplačila** je treba primarno upoštevati organizacijske cilje; treba se je vprašati, zakaj se razvija model kreditnega tveganja. Drugič, definicija neplačila mora biti jasna in sledljiva. To ne omogoča le učinkovitejšega razvoja modela, ampak tudi olajša zasnovo strategij upravljanja s tveganjem. Tretjič, če v industriji obstaja regulativa (npr. baselski sporazum v finančni industriji), je treba slediti smernicam, ki jih ta določa. Glede na povedano obstajajo bolj ali manj stroge definicije neplačila. Najpogosteje se kot neplačnika definira dolžnika, ki s svojimi obveznostmi zamuja za 60 ali 90 dni. Nekoliko restriktivnejša definicija bi kot neplačnika določila dolžnika, ki s plačilom zamuja več kot 120 dni ali pa je bil nad njim začet insolventen postopek (Anderson, 2007). Na drugi strani je treba za diskriminacijo med plačniki in neplačniki v vzorcu opredeliti nabor **neodvisnih spremenljivk**. Kot smo omenili, modeliranje kreditnega tveganja tipično temelji na finančnih kazalnikih, ki jih izpeljemo iz letnih izkazov podjetij, in pretekli plačilni dinamiki, če imamo na voljo podatkovno bazo računov. Številni avtorji ugotavljajo, da vključitev makroekonomskih spremenljivk v kreditni model pripomore k napovedni moči (Altman, Sabato & Wilson, 2008; Bellotti & Crook, 2009). Za robustnost napovednega modela je bistveno, da so podatki kakovostni (tj. natančni, popolni in konsistentni) in dovolj obsežni; to je posebej pomembno pri aplikaciji pristopa strojnega učenja, kot bomo videli v nadaljevanju (Kennedy, 2013). Glede na veliko število virov podatkov je – tipično – dimenzionalnost podatkovne baze velika, zato je smiselno poleg uporabe tehnik reduciranja dimenzionalnosti pri določitvi končnega nabora neodvisnih spremenljivk upoštevati naslednja merila: i) morebitne pravne omejitve; ii) intuitivnost, preverljivost in stabilnost spremenljivke; iii) stroške zbiranja glede na doprinos; iv) prihodnjo razpoložljivost; v) napovedno moč (Thomas, Crook & Edelman, 2017).

Namen magistrskega dela je aplicirati opisano metodologijo ocenjevanja kreditne sposobnosti strank na vzorcu slovenskih podjetij in primerjati napovedno natančnost tradicionalnih statističnih metod s pristopi strojnega učenja (angl. machine learning approach). Vzrok za analizo predstavlja dejstvo, da je dandanes uporaba strojnega učenja v

finančni industriji zaradi navidezne kompleksnosti še vedno zelo omejena, kljub potencialnim prihrankom uporabe pristopa pri ocenjevanju kreditnega tveganja. Lessmann, Baesens, Seow in Thomas (2015); Khandani, Kim in Lo (2010) v raziskavah ugotavljajo, da izboljšana napovedna natančnost tehnik strojnega učenja vodi v primerjavi z logistično regresijo v zmanjšanje kreditnih izgub za od 5,0 do 25,0 %.

Stroji oziroma računalniki se v zadnjem času vse bolj uporabljajo v različnih domenah za opravljanje »inteligentnih« nalog, pri katerih ljudje ne moremo eksplicitno artikulirati svojega znanja oziroma pravil igre. Ključno vlogo pri uspehu pristopa strojnega učenja imajo trije elementi: i) vzpon masovnih podatkov; ii) izboljšani algoritmi samostojnega učenja; iii) boljša/cenejša računalniška zmogljivost. To omogoča sorazmerno preprosto, predvsem pa stroškovno učinkovito aplikacijo pristopa strojnega učenja oziroma umetne inteligence (angl. artificial intelligence) kot celote na različnih področjih, npr. v strojništvu, medicini, biologiji, v financah itn. (Brynjolfsson & McAfee, 2017a). Strokovnjaki na področju umetne inteligence iz svetovalnega podjetja McKinsey ocenjujejo potencialno kumulativno vrednost uporabe te tehnologije na 16 % BDP oziroma 13 bilijard dolarjev do leta 2030, a hkrati opozarjajo na potencialne nevarnosti. Uporaba umetne inteligence namreč prinaša neslutena tveganja, povezana z varovanjem osebnih podatkov, s kibernetsko varnostjo, z manipulacijo javnega mnenja itn. Kljub temu nas te nevarnosti ne smejo odvrniti od nadaljnjega raziskovanja; z raziskovanjem namreč prispevamo h »gradnji« znanja, kar na eni strani omogoča boljše razumevanje priložnosti, na drugi pa učinkovitejše obvladovanje tveganj, ki jih prinaša umetna inteligenca oziroma stojno učenje kot njeno podpodročje (Bughin, Seong, Manyika, Chui & Joshi, 2018; Bughin & Manyika, 2018). Eksperimentiranje z **umetno inteligenco** sega v 50. leta prejšnjega tisočletja, ko so se znanstveniki ukvarjali predvsem z reševanjem dobro definiranih/logičnih problemov, kot je npr. igranje šaha (govorimo o simbolični umetni inteligenci). Tovrsten pristop pa se ni izkazal kot najoptimalnejši za uporabo v kompleksnejših nalogah, pri katerih pravil ni mogoče eksplicitno pred-programirati. Tako se je v okviru umetne inteligence razvila nova paradigma, ki se osredinja na samostojno učenje »agenta«, poznana kot **paradigma strojnega učenja**. Cilj algoritmov (metod) strojnega učenja je samostojno učenje na podlagi vhodnih izkušenj v smeri doseganja čim večje uspešnosti, ki jo navadno merimo z neko stroškovno funkcijo (Chollet, 2018). Znotraj strojnega učenja se v zadnjem času veliko pozornosti namenja **globokemu učenju** (angl. deep learning), ki obravnava globoke (večslojne) nevronske mreže. Te v določenih aplikacijah dosegajo oziroma celo presegajo zmogljivosti naravne inteligence. Bengio, Hilton in LeCunn (2015, p. 1) globoko učenje definirajo kot »več-nivojsko učenje predstavitev vhodnih podatkov, pri čemer vsak nivo vhodno informacijo pretvori na nekoliko višjo, abstraktnejšo raven.«

Kot smo omenili, je fokus magistrskega dela aplikacija pristopa strojnega učenja na primeru napovedovanja kreditne sposobnosti strank. Tovrstno nalogo lahko uvrstimo v domeno **nadzorovanega strojnega učenja** kot enega izmed treh vrst strojnega učenja: i) nadzorovano; ii) nenadzorovano; iii) spodbujevalno učenje. Pri nadzorovanem učenju se algoritem samostojno uči pravil na podlagi implicitnih vzorcev iz vhodnih podatkov in

želenih izhodnih rezultatov (oziroma vrednosti odvisne spremenljivke, ki jih predhodno določi človek/nadzornik). V našem primeru je odvisna spremenljivka binarna oziroma diskretna (tj. plačnik ali neplačnik), zato govorimo o **klasifikaciji**. Metode nadzorovanega strojnega učenja imajo apriorni cilj maksimirati napovedno natančnost na podatkih zunaj vzorca, zato so primerne za napovedovanje verjetnosti neplačila. Ker ja tako strojno učenje izhodiščno namenjeno nalogam napovedovanja, nam tovrstne metode ponujajo veliko večjo izbiro funkcijskih oblik, kar vodi v večjo natančnost modela in posledično boljšo učinkovitost implementiranih strategij za upravljanje s tveganji. Dodatno nam metode stojnega učenja ponujajo večjo fleksibilnost glede izbire neodvisnih spremenljivk, saj so algoritmi prilagojeni za učinkovito učenje na velikih podatkovnih bazah in večjem številu neodvisnih spremenljivk; predvsem globoko učenje, ki dodatno omogoča samodejno izpeljavo spremenljivk (angl. automated feature engineering) (Kononenko & Kukar, 2007; Mirjalili & Raschka, 2017; Ng & Katanforoosh, 2019).

V industriji se za ocenjevanje transakcije verjetnosti neplačila strank oziroma izpeljavo klasifikacijskih pravil še vedno najpogosteje uporabljajo parametrične statistične metode, kot so npr. diskriminantna analiza, pogojni verjetnosti modeli itn. – najbolj priljubljena metoda modeliranja je tako imenovana **logistična regresija** (angl. logistic regression s kratico LR), ki odnos med verjetnostjo neplačila in neodvisnimi spremenljivkami modelira prek logistične funkcije. Kot statistična metoda ima LR določene restriktivne predpostavke, prav tako je v osnovi namenjena pojasnjevanju strukturnega odnosa med neodvisnimi spremenljivkami in verjetnostjo plačila kot pa samemu napovedovanju. Posledično je napovedna natančnost sorazmerno nizka oziroma napovedna napaka višja, kar pomeni, da implementirane strategije upravljanja s kreditnim tveganjem niso optimalne. V nadaljevanju tako predstavljamo **metode strojnega učenja**, ki temeljijo na manj restriktivnih predpostavkah in tako omogočajo večjo sposobnost generalizacije na nove primere.

(i) **k-Najbližjih sosedov** (angl. k-nearest neighbour v nadaljevanju k-NN) je eden izmed osnovnejših algoritmov nadzorovanega strojnega učenja. Glavna značilnost metode je, da pri učenju ne izpelje modela, ampak si zapolni celotni učni vzorec in poda napoved za nov primer s pomočjo primerjave na podlagi mer podobnosti. Zaradi večje robustnosti metode se podobnost primerja s k najbližjimi sosedi (Harrington, 2012).

(ii) **Klasifikacijska drevesa** (angl. classification trees v nadaljevanju DT) se uporabljajo v aplikacijah, ki poleg napovedne natančnosti zahtevajo tudi vpogled v strukturo problema (tj. transparentnost). Ta nelinearna metoda ponazarja relacijo med vhodnimi spremenljivkami in izhodnimi rezultati (tj. odvisno spremenljivko) prek notranjih vozlišč. Vsako drevo se začne z izbiro najpomembnejše spremenljivke in se nato razdeli v poddrevesa glede na vrednosti, ki zagotovijo najbolj »čiste« podmnožice glede na razred; posebna vrsta dreves so binarna klasifikacijska DT, ki vedno ustvarijo le dve podmnožici, kar zmanjša prostor iskanja rešitve. Kot mera čistoče se najpogosteje uporablja Ginijev indeks različnosti. Klasifikacija novih primerov temelji na primerjavi njihovih vrednosti z vrednostmi spremenljivk v vozliščih, dokler ne pridemo do zadnjega lista v drevesu, ki nam poda napoved. DT so izjemno občutljiva na vhodne

podatke, kar vodi v nestabilnost njihove strukture. Zaradi tega se priporoča uporaba ansambelskega pristopa, poznanega pod imenom naključni gozdovi (Geron, 2017).

(iii) **Metoda podpornih vektorjev** (angl. support vector machines v nadaljevanju SVMs) je metoda razvrščanja, ki jo je v 90. letih razvil ruski statistik Vladimir Vapnik. Zanjo je značilno, da množico učnih primerov razdeli v razreda, tako da maksimizira širino ločitvene meje med njima. SVMs si lahko zamislimo kot postavljanje najširše mogoče »ulice«, ki jo definirajo samo najbližji učni primeri oziroma vektorji; te imenujemo podporni vektorji. Ločitvena meja je pri tej metodi linearna, zato SVMs v osnovni različici ne more razvrščati primerov, ki so linearno neločljivi. V tem primeru lahko uporabimo nadgradnjo osnovnega algoritma s tako imenovanimi jedrnimi funkcijami (angl. kernel functions), ki osnovne podatke transformirajo v višjo dimenzijo, pri kateri je problem linearen. Ena izmed ključnih prednosti metode podpornih vektorjev je globalna optimalnost rešitve (Mirjalili & Raschka, 2017).

(iv) **Umetne nevronske mreže** (angl. artificial neural networks v nadaljevanju ANNs) so metode globokega strojnega učenja, ki delujejo po vzoru možganov v bioloških organizmih. Človeški možgani so sestavljeni iz približno 10 milijard nevronov (živčnih celic), ki se med seboj povezujejo v mreže prek nevritov, kar omogoča procesiranje kompleksnih informacij. Podobno delujejo tudi ANNs, le da so te v osnovi veliko preprostejše v primerjavi z delovanjem biološkega živčnega sistema. ANNs sestavlja množica umetnih nevronov, ki imajo več različno uteženih vhodov in en izhod. Učenje v kontekstu nevronskih mrež poteka prek spreminjanja uteži na vhodih nevronih, dokler ta ni zmožna optimalno rešiti nekega problema na podlagi izkušenj iz učnih podatkov. Ključni algoritem, ki prilagaja uteži vsakemu nevronu v mreži, tako da je napovedna napaka vedno manjša, se imenuje algoritem za vzvratno razširjanje (angl. backpropagation algorithm). Bistvena prednost ANNs kot metode globokega strojnega učenja je dejstvo, da se zaradi fleksibilnosti pristopa napovedna natančnost povečuje z naraščanjem števila podatkov, zato je primerna za aplikacijo na masovnih podatkih. Nasprotno učinkovitost drugih metod stagnira (Aggarwal, 2018).

(v) **Ansambelske metode** (angl. ensemble learning) so poleg globokega učenja v zadnjem desetletju prejele veliko pozornosti, saj na spletni strani Kaggle v različnih domenah dosegajo izjemne rezultate z vidika napovedne natančnosti. Cilj ansambelskih metod je združiti več istih oziroma različnih metod strojnega učenja v meta-algoritem, ki izkorišča prednosti posameznih metod. Trenutno je na voljo veliko različnih ansambelskih algoritmov; v okviru magistrskega dela bomo uporabili dve metodi. Pri opisu klasifikacijskih dreves smo opozorili na nestabilnost algoritma. Breiman (2001b) je tako predlagal nadgradnjo – namesto na enem drevesu učenje poteka na množici dreves, kar pomeni, da je končni model veliko robustnejši, napovedna napaka pa manjša. Ta metoda se imenuje **naključni gozdovi** (angl. random forest v nadaljevanju RF). Druga priljubljena ansambelska metoda, ki jo bomo uporabili za napovedovanje verjetnosti neplačila, je znana pod imenom **algoritem AdaBoost** (angl. adaptive boosting algorithm). Metoda poskuša postopoma popraviti napake prejšnjih krogov učenja, tako da v vsakem krogu pripiše večjo utež tistim primerom, ki so bili uvrščeni v

napačen razred (Freund & Schapire, 1999). Dodatno smo uporabili še metodo **gradientne krepitve** (angl. gradient boosting algorithm), ki se je izkazala za izjemno uspešno v preteklih raziskavah.

Proces napovednega modeliranja (angl. predictive modelling) v strojnem učenju v splošnem sledi trem korakom, ki zagotavljajo izbiro metode z največjo sposobnostjo generalizacije na nove podatke – govorimo o pred-procesiranju podatkov (angl. data pre-processing), fazi učenja izbranih metod (angl. learning phase) ter izbiri ter evalvaciji končnega modela (angl. model selection and evaluation). Opisani proces zagotavlja, da je ocenjena natančnost končnega modela realistična, kar je ključna informacija v fazi njegove implementacije v poslovni proces (Mirjalili & Raschka, 2017). Najpogosteje se v praksi za **evalvacijo modela** uporablja stopnja natančnosti (angl. accuracy), vendar pa ta ob neuravnoteženem vzorcu ni optimalna, zato smo za oceno diskriminacijske moči modela uporabili krivuljo ROC (angl. receiver operating curve), ki predstavlja razmerje med deležem pravilno razvrščenih plačnikov in deležem nepravilno razvrščenih neplačnikov, ter izpeljano mero AUROC (angl. area under the reciever operating characteristic), ki meri ploščino pod krivuljo. Vrednost AUROC pri popolnem modelu je enaka 1, medtem ko bi ta pri popolnoma naključnem modelu znašala 0,5 (Jackson & Wood, 2013; Kononenko & Kukar, 2007). Dodatno smo za primerjavo metod uporabili še metriko povprečna natančnost (angl. average precision).

Za namen **empirične analize** smo uporabili podatke iz letnih finančnih izkazov slovenskih gospodarskih družb in večjih samostojnih podjetnikov med letoma 2013 in 2017, ki jih v okviru Poslovnega registra Slovenije (v nadaljevanju PRS) zbira Agencija RS za javnopravne evidence in storitve (v nadaljevanju AJPES). Podatke o insolventnih postopkih za obdobje med letoma 2014 in 2018, ki služijo izpeljavi odvisne spremenljivke (tj. definiciji neplačnikov), je posredoval Center za informatiko Vrhovnega sodišča RS. Kot je razvidno smo pri zbiranju podatkov uporabili pristop večkratnega vzorčenja – to je omogočilo vključitev večjega števila dogodkov neplačila oziroma v našem primeru stečajev. Makroekonomske podatke smo pridobili na spletni strani Statističnega urada RS in na globalnem finančnem portalu Investing.com. Obdelava podatkov je potekala v programskem jeziku Python in razvojnem okolju Jupyter Notebooks, ki je prilagojen potrebam izvajanja podatkovne analitike. Del analize je bil izveden v programskem paketu R Studio.

Običajen potek dela v strojnem učenju se začne s korakom pred-procesiranja podatkov. Prvotni vzorec podatkov je vseboval 322.203 opazovanj tipa podjetje – leto za obdobje med letoma 2013 in 2017. Opazovanjem smo na podlagi podatkov o insolventnih postopkih določili status neplačnika oziroma plačnika, če se je v enem letu po objavi finančnih izkazov nad njim začel voditi stečajni postopek. Prvotni obseg podatkov smo nato zožili samo na **nefinančna podjetja**; to so tista podjetja, ki po klasifikaciji institucionalnih sektorjev (angl. classification of institutional sectors) spadajo v podkategoriji S.11001 in S.11002. Dodatno smo v vzorcu obdržali samo podjetja, ki imajo **podobno strukturo bilance stanja**, saj to omogoča razvoj natančnejšega kreditnega modela – skladno s standardno klasifikacijo

dejavnosti smo izbrali podjetja iz panog C (predelovalne dejavnosti), F (gradbeništvo) in G (trgovina; vzdrževanje in popravila motornih vozil). Nazadnje smo vzorec filtrirali še glede na velikost podjetij; obdržali smo samo podjetja, ki so imela v vzorčnem oknu **povprečen prihodek višji od 10.000 EUR**. Končni vzorec za analizo je tako vseboval 101.880 opazovanj, med katerimi je bilo 842 oziroma 0,8 % neplačnikov. Po izpeljavi 45 finančnih kazalnikov s področij dobičkonosnosti, likvidnosti, zadolženosti, gospodarnosti, investiranja, rasti itn. smo najprej lastnosti podatkov povzeli s pomočjo enostavnih opisnih statistik in histogramov. To je pokazalo, da so v podatkih prisotne manjkajoče in ekstremne vrednosti, kar je tipično pri tovrstnih kazalnikih, saj so vrednosti v imenovalcu velikokrat blizu ničle. Dodatno smo si opazovanja v vzorcu ponazorili s pomočjo **tehnik vizualiziranja visokodimenzionalnih podatkov**. Najprej smo uporabili metodo glavnih komponent (angl. principal component analysis), tako da smo izbrali samo prvi dve komponenti, ki pojasnjujeta največji del razpršenosti analiziranih podatkov. Nato smo podatke ponazorili še s pomočjo nelinearne tehnike t-SNE (angl. t-distributed stohastic neighbouring embedding). Vizualizacija je pokazala, da sta v podatkih prisotni dve gruči podjetij (plačniki in neplačniki), vendar pa je med skupinama tudi nekaj prekrivanja; vseeno dejstvo predstavlja dober signal za uporabo metod nadzorovanega strojnega učenja.

Pred nadaljnjim čiščenjem podatkov smo skladno z uveljavljenim potekom dela v napovednem modeliranju vzorec razdelili na **učno in testno množico** v razmerju 80 : 20. Prva je namenjena izbiri hiperparametrov modela (angl. model selection) in učenju (angl. fitting) klasifikacijske metode, druga pa omogoča nepristransko evalvacijo modela na (njemu) novih podatkih (angl. model performance evaluation). Za izbiro hiperparametrov modela na učni množici smo uporabili pristop k-kratnega prečnega preverjanja (angl. k-fold cross-validation), ki učno množico razbije na k disjunktnih podmnožic, pri čemer se $k - 1$ množic uporabi kot učno množico, k-ta množica pa služi evalvaciji modela ter izbiri optimalnih hiperparametrov, kot prikazuje graf 11 na strani 59. Pred aplikacijo tehnik v korakih 2 in 3 iz grafa 9 na strani 56 smo najprej iz vzorca odstranili tista opazovanja in neodvisne spremenljivke, ki vsebujejo več kot 20 % manjkajočih vrednosti; druge **manjkajoče vrednosti** pa smo nadomestili z mediano. **Ekstremne vrednosti** smo nadomestili z vrednostjo 1. in 99. centila. Tako urejene podatke smo nato standardizirali. Standardizacija (angl. standardization) je postopek, s katerim vrednosti spremenljivke transformiramo, tako da jim odštejemo aritmetično sredino in delimo s standardnim odklonom. To zagotavlja učinkovito učenje klasifikacijskih metod.

Naš učni vzorec vsebuje samo 0,8 % neplačnikov, kar pomeni, da imamo opravka z ekstremno neuravnoteženimi podatki. To zahteva implementacijo **tehnik vzorčenja** (angl. sampling technique), s katerimi uravnotežimo podatke. Najprej smo za uravnoteženje podatkov uporabili naključno podvzorčenje (angl. random undersampling), ki zmanjšuje število opazovanj v večinskem razredu, in naključno nadvzorčenje (angl. random oversampling), ki poveča število manjšinskih opazovanj s pomočjo naključnega podvajanja. Chawla, Bowyer, Hall in Kegelmeyer (2002) za uravnoteženje opazovanj v učni množici predlagajo uporabo tehnike SMOTE (angl. synthetic minority oversampling technique), ki

v vsakem koraku iz manjšinskih opazovanj generira sintetične primere. V delu smo dodatno uporabili nadgrajeno tehniko SMOTEEN, ki predstavlja kombinacijo podvzorčenja prek čiščenja večinskega razreda in nadvzorčenja z uporabo opisane tehnike SMOTE. V drugem koraku empirične analize smo skladno s shemo iz grafa 9 aplicirali osem **tehnik zmanjšanja dimenzionalnosti** (angl. dimensonality reduction techniques) naših podatkov. Cilj tega koraka je določiti optimalen nabor pojasnjevalnih spremenljivk, ki največ prispevajo k napovedni moči modela. V splošnem lahko te delimo v dve skupini: i) tehnike izbire spremenljivk (angl. feature selection); ii) tehnike ekstrakcij spremenljivk (angl. feature extraction). Navadno kreditni modeli vsebujejo od 8 do 15 spremenljivk; končno število je odvisno od cilja modeliranja (tj. transparentnost modela ali čim večja napovedna moč) in stroškov zbiranja podatkov. Korelacijska matrika v prilogi 10 nakazuje, da so nekatere spremenljivke med seboj močno korelirane; v takem primeru je smiselno izračunati statistiko, imenovano variančni inflacijski faktor (angl. variance inflation factor), ki omogoča identifikacijo multikolinearnosti med spremenljivkami v modelu in njihovo izločitev. Iz preostalih neodvisnih spremenljivk smo nato z uporabo tehnik izbire spremenljivk oblikovali šest podmnožic – vsaka je vsebovala deset najpomembnejših spremenljivk glede na določeno merilo. Aplicirane tehnike izbire spremenljivk obsegajo: i) regularizacijo; ii) izračun informacijskega prispevka; iii) MultiSURF; iv) metodo naključnih gozdov; v) rekurzivno eliminacijo z uporabo metode podpornih vektorjev; vi) genetski algoritem. Dodatno smo izvedli redukcijo dimenzij s pomočjo dveh tehnik ekstrakcij: i) metoda glavnih komponent, pri kateri smo uporabili 15 najpomembnejših komponent; ii) linearna diskriminantna analiza.

V tretjem koraku empirične analize smo skladno s shemo iz grafa 14 izvedli samo **učenje izbranih metod** (angl. model learning). Kot že rečeno, zajema proces učenja dve fazi: i) izbiro modela oziroma optimalnih hiperparametrov in validacijo z uporabo 10-kratnega prečnega preverjanja ter ii) končno evalvacijo metod na testni množici. Evalvacija metod temelji na treh metrikah, ki smo jih že predstavili. Pri tem smo se za končno primerjavo metod in diskusijo rezultatov osredinili na mero AUROC, ki je najbolj uveljavljena na področju stojnega učenja. Pythonova knjižnica Scikit-Learn za učinkovito in dosledno implementacijo strojnega učenja ponuja funkcijo »pipeline«, ki omogoča učinkovito veriženje opisanih postopkov pred-procesiranja podatkov in s tem samega učenja. Rezultati, povezani s prvim in z drugim raziskovalnim vprašanjem, so v tabeli 18 na strani 69. Predenj preidemo na diskusijo rezultatov, bi opozorili, da je potekala faza izbire optimalnih hiperparametrov samo na podvzorčenih podatkih in z uporabo regularizacije kot izhodiščne tehnike zmanjšanja dimenzionalnosti. Odločitev za ta **izhodiščni scenarij** temelji na dejstvu, da naša stojna oprema ne omogoča časovno učinkovitega učenja velikega števila modelov v postopku izbire hiperparametrov na tako veliki podatkovni tabeli. Tako smo dobljene optimalne vrednosti iz izhodiščnega scenarija uporabljali med celotno analizo. To ni najbolj optimalno z vidika rezultatov, vendar pa vseeno ponuja neki vpogled v odnos med uporabljenimi metodami/tehnikami in sposobnostjo generalizacije modela na nove podatke. Da lahko odgovorimo na zadnja tri raziskovalna vprašanja tega dela, smo v okviru empirične

analize izvedli učenje metod na originalnih (tj. neuravnoteženih) podatkih in podatkih, na katerih smo aplicirali nadvzorčenje in tehniko SMOTEEN – rezultati analize so na voljo v prilogi 13. Na podoben način smo analizirali vpliv tehnik zmanjšanja dimenzionalnosti. Rezultate lahko najdete v prilogi 14. Pri tem smo dodatno preverili pomembnost posameznih neodvisnih spremenljivk v končnem modelu; dotične rezultate najdete v prilogi 15. Nazadnje smo v model vključili makroekonomske spremenljivke (angl. macroeconomic variables) za vsako leto posebej in preverili njihov vpliv na napovedno moč modela.

V nadaljevanju povzemamo **rezultate izvedene empirične analize**. Kot smo omenili, temelji primerjava sposobnosti generalizacije metod na nove podatke (tj. testno množico) na metriki AUROC, saj imamo opravka z izjemno neuravnoteženimi podatki in tako mera klasifikacijske natančnosti ne bi omogočala robustne primerjave. Naj izpostavimo, da vrednost metrike AUROC pri popolnoma naključnem modelu (angl. dummy classifier) znaša 0,5. Vrednosti, višje od te, pomenijo večjo zmožnost modela v ločevanju med plačniki in neplačniki. Če si pobližje pogledamo rezultate v tabeli 18 na strani 69, vidimo, da znaša vrednost metrike AUROC pri statistični metodi LR 0,868, kar pomeni, da izhodiščna metoda sorazmerno dobro razločuje med razredoma. Najslabše se na naših podatkih obnašata metodi k-NN in DT – njune vrednosti AUROC znašajo 0,854 in 0,835. To je skladno s predhodnimi raziskavami He, Zhang in Zhang (2018) ter Hand in Henley (1997). Na drugi strani lahko opazimo, da naprednejše metode strojnega učenja, kot sta SVMs in ANNs, dosegajo nekoliko višje vrednosti AUROC, vseeno pa je vprašanje, če so te izboljšave statistično značilne. Robustnejše izboljšave smo zaznali pri ansambelskih metodah strojnega učenja. To je bilo pričakovano, saj smo v preteklih raziskavah zasledili, da ti modeli precej konsistentno beležijo večjo sposobnost generalizacije (Lessmann, Baesens, Seow & Thomas, 2015). Kot je razvidno iz tabele 21 na strani 73, aplikacija algoritmov RF, AdaBoost – DT in gradientne krepitve prinaša izboljšavo metrike AUROC. Vse tri ansambelske metode temeljijo na združevanju posameznih dreves metode strojnega učenja DT. RF združuje drevesa paralelno (angl. parallel ensemble methods), medtem ko preostala dva algoritma sekvenčno (angl. sequential ensemble methods) dodajata in izboljšujeta posamezna drevesa. Če primerjamo vrednosti AUROC teh treh metod z metodo DT, se njihova vrednost izboljša za 6,11 %, 7, 31 % in za 8,26 %. Izboljšave v primerjavi s statistično metodo LR so nekoliko manjše, vendar vseeno omembe vredne. Vrednost AUROC je pri RF za 2,07 % višja, pri AdaBoost – DT za 3,23 % in pri algoritmu gradiente krepitve za 4,15 %. Zadnje lepo ponazarja tudi graf krivulj ROC v prilogi 12. Nadalje je naša analiza pokazala, da ima pri učenju modela pomembno vlogo uporaba tehnik vzorčenja – povprečna vrednost metrike AUROC na neuravnoteženih podatkih znaša 0,808, medtem ko so pri podvzorčenju metode v povprečju zabeležile vrednost 0,874. V našem primeru nadvzorčenje in uporaba naprednejše tehnike SMOTEEN v povprečju nista prinesli dodatne izboljšave modelov. Za nekoliko podrobnejše rezultate dotične analize si lahko ogledate graf 16 na strani 74, ki ponazarja vpliv posamezne tehnike vzorčenja glede na izbrano klasifikacijsko metodo. Izpostavili bi, da so ANNs in ansambelske metode (z izjemo AdaBoost – DT) robustnejše pri učenju na neuravnoteženih podatkih, kar je skladno s

predhodnimi raziskavami (Brown & Mues, 2012; He, Zhang & Zhang, 2018). Vpliv uporabe tehnik zmanjšanja dimenzionalnosti je predstavljen v grafu 17 na strani 75. Opazimo lahko, da metode v povprečju najboljše delujejo v kombinaciji z regularizacijo, ki je specifična za vsako metodo, sledi tehnika izbire spremenljivk z naključnimi gozdovi in rekurzivna eliminacija z uporabo SVMs. Če si nekoliko pobližje ogledamo rezultate, ugotovimo, da »najboljša« tehnika zmanjšanja dimenzionalnosti vhodnih podatkov ne obstaja – vsaka metoda se obnaša različno glede na uporabljeno tehniko, zato je treba njeno izbiro prilagoditi uporabljeni klasifikacijski metodi. Vključitev makroekonomskih spremenljivk v model ni prinesla izboljšav modelov v smislu metrike AUROC, saj je vseh pet spremenljivk uvrščenih med najmanj pomembne, kot je razvidno v prilogi 16. Pri tem je treba poudarit, da smo vpliv makroekonomskih spremenljivk analizirali statično in ne z uporabo dinamičnih (panelnih) metod (npr. uporaba markovskih verig ali analiza preživetja), ki se navadno uporabljajo, ko imamo opravka s panelnimi podatki. Naša analiza vključitve makroekonomskih spremenljivk s tega vidika ni najoptimalnejša.

V splošnem rezultati nakazujejo na superiornost metod strojnega učenja, predvsem ansambelskih metod pri ocenjevanju verjetnosti neplačila poslovnih strank podjetja. Tako lahko sklenemo, da je uporaba teh tehnik smiselna, saj na eni strani to prinese višjo sposobnost generalizacije, na drugi strani pa je implementacija z uporabo že spisanih knjižnic preprosta, hitra in zanesljiva. Glede izvedbe empirične analize bi za **prihodnje delo** priporočali uporabo gnezdenega prečnega preverjanja, ki se odraža v robustnejši oceni generalizacije modela. Drugič, analizo bi bilo smiselno aplicirati na podatke specifičnega podjetja, saj bi to omogočilo vključitev dodatnih pojasnjevalnih spremenljivk pa tudi lažji odgovor na vprašanje finančnih prihrankov implementacije metod strojnega učenja. Tretjič, v proces učenja bi bilo smiselno neposredno vključiti koncept maksimizacije dobička in tekstovnih podatkov.

Appendix 2: An Overview of the Topic Related Problems

In practice quantitative CS faces many problems that are well-described in the literature. The purpose of this section is to briefly discuss them along with possible solutions.

Problem Topic	Possible Solutions	Literature
Arbitrary default definition – the payment default definition may be somehow arbitrary and depends on the objectives of CS model. Additionally, payment default is not a well-defined dichotomy in reality.	To confirm that our definition truly identifies defaulted cases we can use two methods: ▪ **consensus method** that is based on expert judgement; ▪ **analytical method** that involves roll rate analysis; Using **indeterminate class** is a way of solving dichotomy issue.	Balcaen and Ooghe (2006, pp. 72-73); Lin, Ansell, and Andreeva, (2012); Siddiqi (2017, pp. 89-103); Sun, Li, Huang, and He (2014, pp. 42-43); Thomas, Crook, and Edelman (2017, pp. 120-122).
Low-default portfolios – predictive modelling methods usually require large number of defaulted and non-defaulted cases. Usually the issue is with the low number of defaulted cases. This leads to the fact that algorithms do not reach optimal performance. It was shown that the optimal class distribution should contain between 50% and 90% minority class examples within the training set.	There are several ways to deal with LDPs: ▪ **stacked sampling** uses multiple observation points for dataset construction; ▪ **change default definition** to less restrictive one (e.g. use 60 DPD instead of 90 DPD delinquency); ▪ **statistical resampling techniques** such as random under-/oversampling, SMOTE method etc. are used to create more balanced datasets; ▪ **ensemble learning methods** that can deal with imbalanced data (e.g. RF, AdaBoost).	Batista, Prati, and Monard (2004); Brown and Mues (2012); Chawla, Bowyer, Hall, and Kegelmeyer (2002); Crone and Finlay (2012); Lemaitre, Nogueira, and Aridas (2017); Lessmann, Baesebs, Seow, and Thomas (2015); Mirjalili and Raschka (2017); Siddiqi (2017, pp. 91-95); Sun, Li, Huang, and He (2014, pp. 50-51).
Use of financial annual account information – the majority of empirical credit risk models build for B2B segment use annual account information in the form of financial ratios. They provide a hard, objective measure of company's health. However, annual account data suffers from creative accounting, lack of auditing, missing values. Moreover, if models are based solely on financial ratios then it is implicitly assumed that there are no other relevant failure indicators.	Although financial ratios have many drawbacks, they are nevertheless important in ECRM. In order to mitigate the discussed problems researchers suggest using other sources of data for measuring organizational success (internal as well as external): ▪ **cash flow-based ratios** present additional variables that can increase the modelling accuracy and may be less manipulated then accrual-based ratios; ▪ **macroeconomic variables** may also have an impact on firm's payment behaviour; ▪ **qualitative variables** such as age, type of business, firm size, CGI, region, etc.	Alaka et al. (2018, p. 178); Altman, Sabato and Wilson (2008); Balcaen and Ooghe (2006, pp. 82-85); Beaver, McNichols, and Rhies (2005); Denis, Denis, and Sarin (1997); Jayasekera (2018, p. 202); Lehmann (2003); Lin, Ansell, and Andreeva, (2012, pp. 540, 546); Maltz, Shenhar, and Reilly (2003); Rosner (2003); Sharma and Iselin (2003); Sun, Li, Huang, and He (2014, pp. 51-53).

Continues on the next page

Table continued

Problem Topic	Possible Solutions	Literature
Non-stationarity and data instability – ECRM is based on assumption that future mirrors past, i.e. the relationship between variables are stable. However, practitioners note that data generally exhibits non-stationarity and instability due to business cycles, different market conditions, etc.	To mitigate poor prediction performance and develop more stable model, following approaches can be used: ▪ **stacked sampling** uses multiple observation points to derive more robust through the cycle model; ▪ **feature screening** based on stationarity of the characteristic's distribution in time (i.e. characteristic distribution analysis).	Anderson (2007, p. 381); Balcaen and Ooghe (2006, pp. 74-75); Jayasekera (2018, p. 199); Siddiqi (2017, pp. 86-89).
Sample bias – model development is usually carried out on "known" customers, i.e. the ones that have been granted a service before (accepted customers). Using such models can be inaccurate, as the sample is not representative of the whole population; hence we talk about reject sample bias. Furthermore, sample selection bias may be present as a result of "complete data" selection criteria that eliminates companies with incomplete data.	The presence of sample bias is usually limited to application scoring, whereas behavioural scoring does not suffer from it. Furthermore, in environments with high approval rates the bias is also less apparent. There are generally two mechanism underlying missing (i.e. not accepted) customers. Firstly, missing at random (hereinafter: MAR), and secondly, missing not at random (hereinafter: MNAR). In case of MNAR **reject inference** is typically applied in order to develop a model that can be used for "through-out-the-door" population. Sample selection bias may be solved with using more sophisticated techniques that deal with missing values.	Albon (2018, pp. 76-79); Anderson (2007, pp. 401-418); Balcaen and Ooghe (2006, pp. 75-76); Florez-Lopez (2010); Lessmann, Baesens, Seow, and Thomas (2015); Mirjalili and Raschka (2017, pp. 107-111) Montrichard (2018); Siddiqi (2017, pp. 175-178, 215-235); Thomas, Crook, and Edelman (2017, pp. 141-149); Verstraeten and Van den Poel (2005); Zmijewski (1984).
Feature selection – the process of choosing an optimal subset of characteristics from the full feature set is often done based on subjective expert judgement (qualitative selection) or popularity in the previous research, which may not lead to the optimal subset selection given the fact that there are usually thousands of possible combinations. Another issue might be that the features included in the model exhibit multicollinearity, which can lead to unstable performance and inaccurate results (especially in case of traditional statistical methods).	There are three general classes of quantitative feature selection algorithms that can be used as an alternative to expert judgement: ▪ **filter methods** that apply a statistical measure (e.g. Chi squared test, t-test) to assign power to each feature, usually in a univariate fashion; ▪ **wrapper methods** consider feature selection as a part of model learning to find the best feature subset from (e.g. stepwise procedure, GAs); ▪ **embedded methods** try to combine feature selection and model construction using regularization methods. Alternatively, **feature extraction techniques** (e.g. factor analysis or PCA) can be used that reduce feature space dimensions.	Alaka et al. (2018, p. 178); Garcia, Marques, and Sanchez (2018, pp. 1390-1391); Kennedy (2013, pp. 75-81); Liang, Tsai, and Wu (2015); Lin, Liang, Yeh, and Huang (2014); Mirjalili and Raschka (2017, pp. 123-184); Sharma (2018); Siddiqi (2017, pp. 179-182); Sun, Li, Huang, and He (2014, pp. 51-53); Šušteršič, Mramor, and Zupan (2009, p. 4740); Thomas, Crook, and Edelman (2017, pp. 137-141); Zhou, Lu, and Fujita (2015).

Continues on the next page

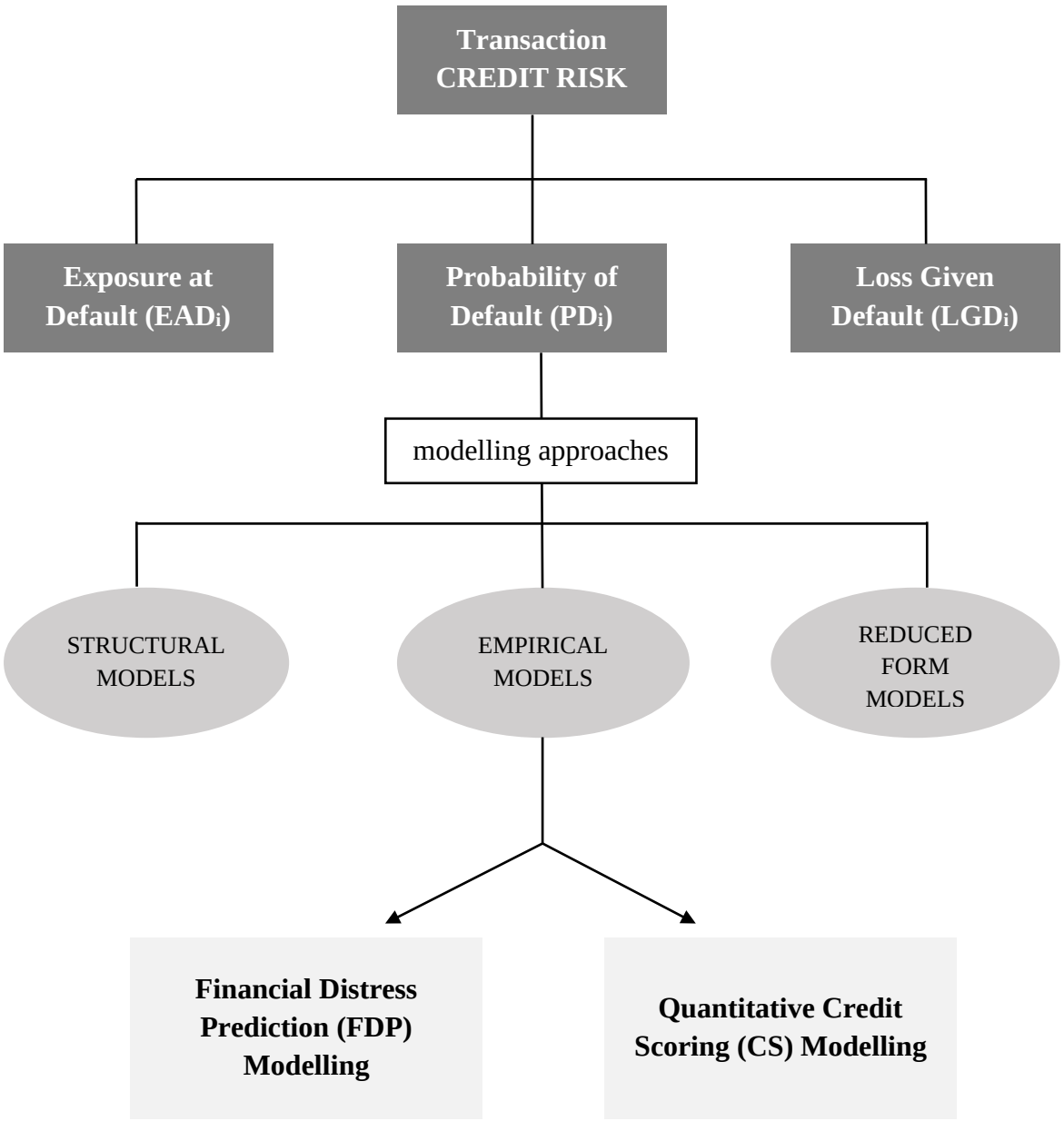
Table continued

Problem Topic	Possible Solutions	Literature
Method selection – many researchers note that the method selection for ECRM mostly depends on method's popularity or is based on modeler's professional background. The use of LR is thus still predominantly used in the industry. The ascent of AI and ML provides a handful of new methods. A comprehensive method selection framework should thus be employed to select the best method for given domain.	The set of criteria for method selection can be categorized into three distinct categories: ▪ **results related criteria** (e.g. accuracy, results transparency, deterministic output); ▪ **data related criteria** (e.g. sample size, types of variable, variable selection); ▪ **method's properties related criteria** (e.g. linear vs non-linear representation, generalization ability). Considering the relative importance of these criteria, we acknowledge that predictive accuracy is of pivotal importance in CS. Once we determine the set of possible methods based on upper criteria the final **multiple comparison tests** should be employed to statistically determine the best performing model.	Alaka et al. (2018); Balcaen and Ooghe (2006, pp. 77-79); Beque and Lessmann (2017); Brown and Mues (2012); Demšar (2006); Kononenko and Kukar (2007); Thomas, Crook, and Edelman (2017, pp. 25-99).
Black-box nature of advanced ML methods – traditional statistical methods provide us with less accurate but transparent model that can be interpreted for business purposes. Contrary, ML methods (particularly ANNs and SVMs) provide highly accurate predictions at the cost of lower transparency.	In order to gain accurate information into the underlying relationship between variables **rule extraction approaches** to derive symbolic representations can be applied. The experiments show that rule extraction techniques lose only a small percentage in performance compared with the original ML method.	Alaka et al. (2018); Breiman (2001a); Garcia, Marques, and Sanchez (2018, p. 1392); Thomas, Crook, and Edelman (2017, pp. 70-74).
Time dimension – classification methods used in this study use single observation (e.g. one annual account) for each customer in the dataset. This snapshot characteristic might not be sufficient to model default events, since it does not account for time-series behaviour of the failure process. Default is thus being modelled as a discrete event assuming the failure is static rather than dynamic process in nature.	One way of treating default process in a time-series manner is using **survival analysis** for analysing the expected duration of time until default happens. This allows us to model not just if a customer will default, but also when, since time is accounted for. One of the most used survival models in CS is Cox proportional hazards model that enables evaluation of specific factors on survival. Another type of methods that incorporate dynamic aspect into the estimation of PD are referred to as **Markov chain probability models**. Alternatively, one can include dynamic indicators based on annual accounts into the classification methods and thus partially account for the time dimension.	Balcaen and Ooghe (2006, pp. 77-79); Bellotti and Crook (2009); Jayasekera (2018); Režnakova and Karas (2014); Thomas, Crook, and Edelman (2017, pp. 157-177); Tong, Mues, and Thomas (2012); Wu, Gaunt, and Gray (2010).

Source: Own work.

Appendix 3: The Basic Components of Credit Risk and Corresponding Modelling Approaches

Central concept in measuring credit risk is the **probability of default** of a customer. PD does not however represent a complete picture of the potential credit loss. Firms seek to measure two additional components characterizing the extent of default loss. Firstly, the magnitude of likely loss on the exposure termed as **loss given default** and expressed as a percentage of the overall exposure, and secondly, the amount to which a firm is exposed at the time of default known as **exposure at default** (Alexander & Sheedy, 2004).



Source: Own work.

Appendix 4: Specification of Independent Variables Used in the Study

The table below lists and categorizes the independent variables we used in the study. Additionally, it provides a short description of each category characterizing its importance on business operations.

Category	Description	Selected Variables
Profitability Ratios Name range: f01 – f10	Assess company's ability to earn profits on sales, assets and equity (i.e. margin analysis). Critical in determining the attractiveness of investing in company.	Return on equity (ROE), Return on assets (ROA), Net profit margin from sales, EBITDA margin from sales, EBIT margin from sales, Overall efficiency, Operating efficiency, Value-added per employee, Net profit per employee, EBIT to total assets
Liquidity/Solvency Ratios Name range: f11 – f19	Assess company's financial ability to meet short-term/long-term obligations using short-term/long-term assets.	Quick ratio, Current ratio, Cash ratio, Receivables to payables ratio, LT financing of LT assets and inventory, Working capital (WCA) to total assets, Working capital (WCA) to sales, Trade credit exposure, Interest coverage ratio
Leverage Ratios Name range: f20 – f25	Assess the extent the debt is used in company's capital structure and evaluate the level of financial leverage as well as the ability to service debt obligations.	Equity ratio, Debt ratio, Debt/equity, Debt/EBITDA, LT debt to total assets, ST debt to total assets
Investment Ratios Name range: f26 – f30	Assess company's investments into various types of assets such as property, plant, and equipment, working assets (i.e. examines asset structure).	Property, plant, and equipment to total assets, Working assets to total assets, Financial investments to total assets, Current assets to total assets, Capital expenditures (CAPEX) to non-current assets
Efficiency Ratios Name range: f31 – f36	Use turnover measures to assess how efficient is company in its operations and use of assets.	Asset turnover ratio, Inventory turnover ratio, Receivables turnover ratio, Payables turnover ratio, Working assets turnover ratio, WCA turnover ratio
Cash Flow Ratios Name range: f37 – f40	Assess company's cash flow (i.e. the lifeblood of each business) and evaluate how solvent, liquid, and viable company is.	Current liability coverage ratio, Operating cash flow to sales, Asset efficiency ratio, Operating cash flow to debt
Growth Indicators Name range: f41 – f45	Measure company's year-to-year progress in essential areas of company performance.	Assets growth, Revenue growth, Equity growth, Negative equity, Net income growth
Macroeconomic Variables Name range: e01 – e05	A set of variables describing general conditions in the economy that have influence on normal business operations.	GDP growth, Unemployment rate, Government yield, Inflation rate, SBITOP index

Source: Bellotti and Crook (2009); Lehmann (2003); Lin, Liang, Yeh, and Huang (2014); Peavler (2019).

Appendix 5: A Deeper Insight into the Derivation of Support Vector Machines

PART 1: The constrained **"hard" SVMs optimization** problem is represented in (29) and has some advantages: reliable solution due to convexity, as well as higher computational efficiency via formulation of the dual problem.

$$\begin{aligned} &min_{\boldsymbol{w},b}\ \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &s.t.\ y^{(i)}(\boldsymbol{w}^T\boldsymbol{x}+b)-1\geq 0\ \forall_i \end{aligned} \tag{29}$$

In order to cater for the linear inequality constraints, we construct a primal Lagrangian under regularity conditions known as Karush-Kuhn-Tucker (hereinafter: KKT) conditions for the problem at hand:

$$L_P(\boldsymbol{w},b,\alpha)=\frac{1}{2}\|\boldsymbol{w}\|^2-\sum_{i=1}^{n}\alpha_i\left[y^{(i)}(\boldsymbol{w}^T\boldsymbol{x}^{(i)}+b)-1\right] \tag{30}$$

The additional α_i variables are nonnegative KKT multipliers that enable the reduction of a constrained problem into unconstrained one and play an important role in calculating optimal solution. According to the Lagrange duality theorem convex objective function and continuously differentiable linear constrains can be rewritten in a dual form that has the same solution. In order to formulate dual problem, we minimize $L_P(\boldsymbol{w},b,\alpha)$ and get the following first order conditions:

$$\frac{\partial}{\partial \boldsymbol{w}}L_P(\boldsymbol{w},b,\alpha)=\boldsymbol{w}-\sum_{i=1}^{n}\alpha_i y^{(i)}\boldsymbol{x}^{(i)}=0\Rightarrow \boldsymbol{w}=\sum_{i=1}^{n}\alpha_i y^{(i)}\boldsymbol{x}^{(i)} \tag{31}$$

$$\frac{\partial}{\partial b}L_P(\boldsymbol{w},b,\alpha)=\sum_{i=1}^{n}\alpha_i y^{(i)}=0 \tag{32}$$

Equation (31) provides an interesting insight into SVMs. The optimal $\boldsymbol{w}$ is defined solely in terms of the training examples, whose corresponding α_i differs from zero; as we already know these are our support vectors. Plugging (31) and (32) back into the primal Lagrangian postulates the following dual problem:

$$\begin{aligned} L_D(\boldsymbol{w},b,\alpha) &= \frac{1}{2}\left\|\left(\sum_{i=1}^{n}\alpha_i y^{(i)}\boldsymbol{x}^{(i)}\right)\right\|^2 \\ &\quad-\sum_{i=1}^{n}\alpha_i\left[y^{(i)}\left(\left(\sum_{j=1}^{n}\alpha_j y^{(j)}\boldsymbol{x}^{(j)}\right)^T\boldsymbol{x}^{(i)}+b\right)-1\right] \\ &= \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y^{(i)}y^{(j)}\boldsymbol{x}^{(i)}\cdot\boldsymbol{x}^{(j)}-\sum_{i,j=1}^{n}\alpha_i\alpha_j y^{(i)}y^{(j)}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(j)^T}-b\sum_{i=1}^{n}\alpha_i y^{(i)}+\sum_{i=1}^{n}\alpha_i \\ &= \sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y^{(i)}y^{(j)}\boldsymbol{x}^{(i)}\cdot\boldsymbol{x}^{(j)} \end{aligned} \tag{33}$$

Final optimization problem therefore becomes finding the vector $\boldsymbol{\alpha}$ that maximizes the objective function $J(\boldsymbol{\alpha})$.

$$max_{\boldsymbol{\alpha}}\ J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$$

$$s.t.\ \alpha_i \geq 0, \qquad i = 1, \dots, n\ and \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \tag{34}$$

Since SVMs optimization problem satisfies all KKT conditions the solution of the dual problem is consistent with the primal problem. Note that the algorithm requires only the dot product of input vectors to be calculated. This is important for the application of kernel trick discussed later (Fletcher, 2008; Haykin, 2009; Ng, 2018).

PART 2: In order to derive the **"soft" SVMs classifier** the "hard" SVMs optimization problem from (29) is supplemented as follows:

$$min_{\boldsymbol{w},b}\ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi^{(i)}$$

$$s.t.\ y^{(i)}(\boldsymbol{w}^T\boldsymbol{x} + b) \geq 1 - \xi^{(i)}\ \forall_i\ and\ \xi^{(i)} \geq 0, i = 1, \dots, n \tag{35}$$

The addition of the second term into the optimization problem sets the upper bound on the number of test errors. The trade-off between model complexity and the number of nonseparable points is controlled via parameter C, which can therefore be thought of as the reciprocal of the regularization parameter. Setting C to large values puts more emphasis on correctly classifying the training examples and can thus lead to overfitting, whereas small values of C allow for misclassifications. In any event, C must be selected by the model developer and is determined experimentally using separate validation set or CV technique (Haykin, 2009).

In similar fashion as before we may derive the primal Lagrangian function, where μ_i are Lagrangian multipliers with regards to the new constraint:

$$L_P(\boldsymbol{w}, b, \alpha, \mu) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi^{(i)} - \sum_{i=1}^{n} \alpha_i\left[y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) - 1\right] - \sum_{i=1}^{n} \mu_i \xi^{(i)} \tag{36}$$

Minimizing w.r.t. $\boldsymbol{w}, b$ and $\xi^{(i)}$ and considering the positivity constraints as well as the KKT conditions allows us to construct and solve the dual problem, which uniquely characterizes the solution (Hastie, Tibshirani & Friedman, 2017):

$$max_{\boldsymbol{\alpha}}\ J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$$

$$s.t.\ 0 \leq \alpha_i \leq C,\ i = 1, \dots, n\ and\ \textstyle\sum_{i=1}^{n} \alpha_i y^{(i)} = 0. \tag{37}$$

Appendix 6: Some Practical Issues in Training the ANNs

As Hastie, Tibshirani, and Friedman (2017, p. 397) suggest there is "quite an art in training neural networks. The model is generally overparametrized, and the optimization is nonconvex and unstable unless certain guidelines are followed." Some practical issues in training ANNs are as follows (Aggarwal, 2018; Hastie, Tibshirani & Friedman, 2017; Geron, 2017; Crone, Lessmann & Stahlbock, 2006):

(i) **feature pre-processing**: scaling of the inputs plays an important role in ANNs' training and can have a significant effect on the final solution. It is a good practice therefore to standardize the inputs; usually min-max normalization is carried out to scale the inputs into the range $(0, 1)$, which helps gradient descent strategies to perform efficiently. Another technique employed is referred to as whitening, in which the axes are rotated in a way to create uncorrelated features (e.g. typically PCA is used);

(ii) **parameter initialization**: choice of initial weights in the network is of particular importance due to the stability issues during the network training, known as the vanishing/exploding gradients discussed below. Setting all weights to 0 would lead to having all derivatives equal to zero, which implies perfect symmetry, and the algorithm gets stuck. Starting instead with large values may lead to poor solutions. ML practitioners suggest using more sophisticated rules that account for number of layers such as the Xavier/He initialization. Bias parameters are always initialized to zero;

(iii) **overfitting**: normally ANNs have too many parameters which leads to overfitting problems and poor generalization performance for the unseen data. Increasing the number of training data usually improves the performance. A good rule of thumb is to have at least two to three times more training instances than parameters. Additionally, there exists several design methods to mitigate the impact of overfitting. Firstly, regularization or weight decay that adds a penalty in the form of either $L_1 = \lambda\|\boldsymbol{W}\|_1 = \lambda\sum|\boldsymbol{W}|$ or $L_2 = \|\boldsymbol{W}\|_2 = \lambda\sum\boldsymbol{W}^2$ norm to the cost function. Larger values of λ (determined using CV or hold-out) tend to shrink weights towards zero and thus generalize the model. Secondly, early stopping that ends the training after a few iterations. The exact stopping time is determined with the help of separate hold-out set, i.e. when the hold-out generalization error starts increasing, the training is stopped. Thirdly, dropout method that is referred to one of the most effective regularization techniques. It consists of randomly dropping out a number of neurons in the layer, which introduces noise into the model and prevents the network to learn insignificant patterns. Dropout can also be thought of as an ensemble technique for ANNs. As we discuss in the next section, ensemble methods usually perform better than base learners, hence the popularity of this regularization approach;

(iv) **tuning hyper-parameters**: one drawback of ANNs' flexibility is that usually they require careful hyper-parameter tuning in order to derive well performing architecture. The hyper-parameters such as the learning rate, the regularization parameter, number of layers and neurons etc. are therefore optimized in a separate tuning phase that follows the initial training of the network. The tuning phase is carried out on a separate hold-out

data or via CV method using the so-called grid search technique. One issue with the procedure is that the number of hyper-parameters is usually large, which leads to an exponential increase in the number of points in the grid. Therefore, it is much better to use randomize grid search, where computational burden can be directly controlled;

(v) **the vanishing and exploding gradients**: deep ANNs usually face several stability issues with regards to the gradients, i.e. earlier layers receive small updates compared to the later ones using backpropagation algorithm. This is referred to as the vanishing gradient problem. On the other hand, some gradients may accumulate and result in very large updates, which is known as the exploding gradient. Both problems lead to unstable network. Some solutions to address the issues are as follows: specific choice of activation function (e.g. ReLU and hard tanh), adaptive gradient descent strategies, batch normalization, etc.

Appendix 7: (A Peak into) Model Optimization Procedures

The third component of learning as argued in the beginning of section 2 is **model optimization**. When the problem at hand follows some well-defined form, we can usually directly calculate the exact "closed form" analytical solutions (e.g. linear regression using linear algebra). However, in applied ML most of the methods that enable complex problem solving do not have an exact solution, but rather require numerical optimization. That being said the learning process in the ML domain essentially boils down to optimizing the set of model weights through the iterative process of minimizing the cost function via different gradient descent strategies; in that sense we can look at ML as an iterative search problem (Brownlee, 2018b). The rest of this section deals with high-level non-technical discussion of the optimization strategies used in this study. First, we look into the simple gradient descent update rule, which is part of first-order continuous optimization techniques. Then second-order quasi-Newton methods used in optimizing convex LR cost function are briefly discussed. The rest of the chapter deals with ANNs non-convex optimization strategies that are incorporated into the gradient descent rule. At the end coordinate-wise optimization (i.e. sequential minimal optimization) is mentioned as an efficient way of solving the SVMs dual problem.

When LR method was discussed we noted that the simplest way of optimizing the $\boldsymbol{\theta}$ set of parameters is via the **gradient descent update rule**. The general idea behind all gradient descent strategies is to use the negative direction of the gradient (i.e. multivariable derivative vector that points in the direction of steepest ascent) of a given cost function to gradually converge to a minimum. One important parameter of the basic gradient descent algorithm is the learning rate α that determines the size of the step; if α is too small, then the convergence to the optimal set of $\boldsymbol{\theta}$ is slow (i.e. it takes many iterations), alternatively choosing a large α might make the algorithm diverge and thus fail to find the solution (Geron, 2017). Gradient descent only uses the knowledge of first-degree derivatives. When we have a well-behaved convex cost function **quasi-Newton numerical methods** give superior performance in searching the minimum of a function as they also employ second-order derivatives (second-order optimization algorithms). That is why LR optimization is commonly carried out using L-BFGS (i.e. Limited-Memory Broyden-Fletcher-Goldfarb-Shanno) algorithm, which is an iterative method for solving unconstrained nonlinear problems (Bengio, Courville & Goodfellow, 2016; Cramer, 2003).

For ANNs where we face non-convex cost functions with ridges, plateaus and other irregularities, quasi-Newton methods impose significant computational burden (e.g. Hessian matrix of second-order derivatives). Additionally, in high-dimensional spaces they tend to converge to functional irregularities (i.e. especially saddle points). Since most problems in ANNs are difficult to express in terms of smooth convex functions, generally gradient descent with some additional course corrections on top the basic update rule is employed. Modifications help the update rule to at least partially overcome the issue of multiple local minima, saddle points, etc. and so enable faster learning in this "ill-conditioned" setting. One

method that can significantly speed up the learning process is known as **momentum optimization**. Momentum-based techniques try to smooth out the zigzagging by using an "averaged" direction of the last few iterations. This is formally done via introduction of the momentum term $\boldsymbol{m}$ into the update rule that is derived from an exponentially decaying average of previous gradients that consistently "move" through the parameter space and help in avoiding local minima and zigzagging. However, momentum optimization introduces another hyperparameter β that requires careful tuning. It influences the importance of $\boldsymbol{m}$, namely the higher the value the higher the friction (in case $\beta = 0$ there is no momentum). Other methods such as RMSProp and AdaDelta attack the zigzagging issue through introduction of the **parameter-specific (adaptive) learning rate**. The idea is that large parameters are often oscillating so the corresponding learning rates are diminished to achieve stability. **Adam optimization** which stands for "adaptive moments" is generally the best choice since it incorporates both ideas, i.e. momentum as well as adaptive learning rate. The additional hyperparameter η requires little tuning since the default value of 0.001 is used (Aggarwal, 2018; Bengio, Courville & Goodfellow, 2016; Geron, 2017).

The quadratic dual problem that was formulated for SVMs in (37) requires different kind of optimization strategy in order to find the vector of Lagrangian multipliers $\boldsymbol{\alpha}$ that minimizes the SVMs cost function J($\boldsymbol{\alpha}$). The **SMO** (sequential minimal optimization) algorithm developed by John Platt does the job. It builds on a simple idea of **coordinate-wise optimization** known as coordinate descent, that finds the minimum of a function by moving along the coordinate block. In a basic coordinate descent usually one parameter at the time is optimized leaving other fixed. However, the second SVMs constraint in (37) that can be rewritten as $\alpha_1 = -y^{(1)} \sum_{i=2}^{n} \alpha_i y^{(i)}$ defines α_1 completely in terms of other α_i's. Thus, holding the $\alpha_2, \dots, \alpha_m$ fixed we cannot change α_1 without violating the constraint. That is why SMO takes the problem and breaks it into sequential optimization of two alphas at the time in order to keep satisfying the constraint. The convergence of the procedure is guaranteed given the KKT conditions are satisfied (Ng, 2018). There are few more details with regards to the SMO that are beyond the scope of this study; for more information you can refer to Platt's (1998) article "SMO: A Fast Algorithm for Training Support Vector Machines."

Appendix 8: Eight Key ML Lessons to Have in Mind

Like any discipline, ML has a lot of "folk wisdom" that is not thoroughly documented in textbooks. Thus, it is worth reviewing them before delving into the empirical part of the study. This section therefore highlights key ML lessons to have in mind when constructing a ML related project (especially supervised one). They are based on our hitherto discussion as well as on insights provided by Domingos (2012):

(i) Lesson 1 – **it is generalization that counts**: the aim of ML approach (i.e. predictive modelling) is to achieve good generalization ability with respect to unseen data instances. This means that it is essential to test the model generalization performance on a separate test set prior to deploying it for day-to-day business use (according to the principles for the evaluation of learned hypothesis). From technical perspective modeler has to prevent data leakage, defined as an introduction of information from outside the training dataset into the learning phase, in order not to overestimate the expected performance. The two possible sources of data leakage are leaking features (i.e. variables that giveaway information observed after prediction is required) and illegitimate DPP procedures (Kaufman, Perlich & Rosset, 2011).

(ii) Lesson 2 - **data alone is not enough**: in the absence of some general assumptions there is no way learner can generalize beyond the data given. This was demonstrated by Wolpert (1996) in the famous "no free lunch theorem (hereinafter: NFL)" stating that if no assumption is made, then no algorithm can be preferred over any other. ML tools therefore employ some loose assumptions such as smoothness, similar examples have similar classes etc. in order to learn from given data. In practical terms NFL theorem suggest that no model is a priori better than others. Consequently, we must evaluate more reasonable models to find the optimal one (Geron, 2017).

(iii) Lesson 3 - **overfitting has many faces**: encoding too much data related noise usually leads to the problem of overfitting (e.g. model performs with ~100% accuracy on training set but much worse on testing set). One way of understanding overfitting is by decomposing the generalization error into bias and variance. If we assume $\boldsymbol{y} = f(\boldsymbol{X}) + \varepsilon$ where $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma_\varepsilon^2$ then we can we can derive expression (38) for total prediction error at some $\boldsymbol{X} = x_0$ as follows (Hastie, Tibshirani & Friedman, 2017):

$$\begin{aligned} Err(x_0) &= E\left[\left(y - \hat{f}(x_0)\right)^2 \middle| \boldsymbol{X} = x_0\right] \\ &= \sigma_\varepsilon^2 + \left[E\left[\hat{f}(x_0)\right] - f(x_0)\right]^2 + E\left[\hat{f}(x_0) - E\left[\hat{f}(x_0)\right]\right]^2 \\ &= \sigma_\varepsilon^2 + Bias^2\left(\hat{f}(x_0)\right) + Var\left(\hat{f}(x_0)\right) \end{aligned} \tag{38}$$

The first term is the irreducible error, the second term is squared bias (the amount by which the average of estimates differs from true mean) and the last term is the variance (measures algorithm's sensitivity to particular learning set). Bias a priori depends on the learning method and cannot be changed unless we choose different learning

algorithm (i.e. alter hypothesis space or use multiple methods), whereas variance originates from the learning examples. Generalization error is therefore composed of two opposing factors; hence a compromise must be found. Typically, if we make the model $\hat{f}(\cdot)$ more complex by adding parameters, we get lower bias and higher variance which results in overfitting and vice-versa, simpler models with less parameters results in underfitting - high bias and low variance (Kononenko & Kukar, 2007). There are many tools that help us solve this under- vs. over-fitting problem: perform CV (allows tuning model hyper-parameters within training set), train with more data, remove irrelevant features, use early stopping (stopping the learning process as it begins to overfit), include regularization term, and/or consider ensemble methods.

(iv) Lesson 4 - **intuition fails in high dimension**: "curse of dimensionality" as described by Bellman is one of the biggest problems in ML besides overfitting and refers to the fact that as dimensionality of data (i.e. number of features) increases, space increases so fast that data becomes sparse and many algorithms break down. Building a classifier is therefore much harder as we do not have the intuition supporting our development. In this sense "less is more", since having more features may be outweighed by the dimensionality issues. In practical applications however Domingos (2012, p. 82) suggests that the curse is partly offset by the fact that "examples are not spread uniformly through the feature space, but are concentrated on or near a lower-dimensional manifold (i.e. "the blessing of non-uniformity")." Some learners can implicitly benefit from the fact, while on the other hand various dimensionality reduction technics may be employed (i.e. feature selection and/or feature extraction).

(v) Lesson 5 - **feature engineering is the key**: one of the most important factors in designing a good ML classifier is feature engineering and is therefore a step that takes the most effort - it requires time, intuition, creativity as well as strong domain knowledge. Consequently, ML should be an iterative process of experimenting with various feature engineering techniques, analysing the performance, and repeating the whole process. Since this process is time-consuming and domain knowledge-related automation of feature engineering using ANNs may help.

(vi) Lesson 6 - **more data beats a cleverer algorithm**: when facing issues with model performance gathering more data is usually the solution. In case we have limited data at hand simple algorithm may beat a complex one since more evolved ML methods usually have much more parameters to tune and hence require bigger datasets (in line with Figure 5). It is a good practice to try simpler methods first to establish some benchmark and build on that using more sophisticated tools. Another data related issue is its representativeness, namely the problem of sampling bias and quality of the overall dataset that can be summarized in famous quote "garbage in, garbage out" (Geron, 2017).

(vii) Lesson 7 - **learn many models, not just one**: this in accordance with our hitherto discussion that ML is not a "one-way street", since developers must try multiple variations of learners to find the best one. Furthermore, as argued in section 2.3.3

ensemble modelling is now a standard, which means it must be part of a ML toolkit employed in each application.

(viii) Lesson 8 - **simplicity does not imply accuracy**: we are familiar with the theoretical aspects of Occam's razor suggesting that the simplest explanation is usually also the most reliable. However, hands-on experience implies that taking this recipe too serious would prevent us from using accurate but more complex ensemble models or algorithms like SVMs and ANNs. Domingos (1999) provides an interesting viewpoint on the matter. He argues that the original Occam's intent was not to advance the simpler models of the two with the same training-set error, but rather "given two models with the same generalization error (i.e. test-set error), the simpler one should be preferred because simplicity is desirable in itself" (Domingos, 1999, p. 410). Meaning that simpler models should not be preferred at the expense of lower accuracy, but rather due to simplicity being "a virtue in its own right."

Appendix 9: Financial Ratios Descriptive Statistics and Groupwise Medians

The table below presents the basic descriptive statistics of our final modelling sample.

Vars[a]	Count	Mean	Std. Dev.	Min	25%	50%	75%	Max
f01	101,865	-0.08	15.57	-3,714	0.00	0.06	0.20	1,287
f02	101,866	0.01	1.61	-375	0.00	0.02	0.08	132
f03	101,880	-0.05	5.18	-1,595	0.00	0.02	0.05	108
f04	101,880	0.05	0.48	-115	0.02	0.05	0.12	1
f05	101,880	-0.03	4.43	-1,385	0.00	0.02	0.07	1
f06	101,873	1.44	54.88	0	1.00	1.02	1.06	11,010
f07	101,869	1.47	54.87	0	1.00	1.02	1.07	11,203
f08	87,049	34,865.37	185,124.74	-5,434,140	16,533.75	24,173.66	36,637.67	42,490,385
f09	87,049	3,715.61	168,105.78	-30,853,520	143.10	1,480.28	5,860.47	20,372,425
f10	101,818	-0.68	153.91	-48,179	0.00	0.03	0.09	1,524
f11	101,496	3.76	97.33	0	0.55	1.07	1.96	18,158
f12	101,496	4.72	134.90	0	0.96	1.44	2.52	30,160
f13	101,496	2.01	70.17	0	0.05	0.23	0.78	18,098
f14	101,365	13.59	3,545.79	0	0.41	0.88	1.52	1,128,661
f15	94,918	62.91	10,350.20	-747,482	0.62	1.03	1.86	2,054,779
f16	101,818	-0.23	24.98	-6,558	-0.02	0.20	0.45	1
f17	101,880	0.07	7.11	-990	-0.01	0.12	0.33	343
f18	101,880	0.40	1.87	0	0.09	0.19	0.34	252
f19	54,105	2,138.16	78,296.36	-2,945,261	1.08	4.78	24.25	6,018,287
f20	101,818	-0.22	33.99	-6,558	0.15	0.37	0.62	1
f21	101,818	1.22	33.99	0	0.38	0.63	0.85	6,559
f22	101,861	7.00	193.08	0	0.62	1.70	2.87	39,318
f23	101,840	40.02	3,586.03	0	3.13	7.88	10.94	1,129,473
f24	101,818	0.27	17.00	0	0.00	0.00	0.18	3,615
f25	101,818	0.95	25.05	0	0.24	0.46	0.72	6,559
f26	101,818	0.24	0.25	0	0.02	0.14	0.40	1
f27	101,818	0.62	0.30	-1	0.39	0.66	0.89	1
f28	101,818	0.11	0.20	0	0.00	0.00	0.11	1
f29	101,818	0.70	0.28	0	0.49	0.77	0.95	1
f30	89,854	-1.77	415.18	-120,714	0.00	0.07	0.38	19,065
f31	101,866	2.13	12.63	0	0.79	1.43	2.44	2,483
f32	75,954	6,917.38	1,814,892.38	0	3.82	8.78	25.36	500,119,730
f33	100,899	974.00	288,148.29	0	3.04	5.38	10.40	91,481,572
f34	101,608	9.89	290.54	0	1.67	3.28	5.68	47,347
f35	101,791	7.42	364.27	0	1.40	2.40	4.01	95,736
f36	101,847	25.93	1,834.60	0	0.00	3.08	8.60	572,156
f37	101,496	0.06	83.11	-23,679	-0.05	0.13	0.47	3,270
f38	101,880	0.05	1.25	-144	-0.02	0.04	0.13	103
f39	101,818	-0.03	72.63	-18,338	-0.02	0.06	0.16	12,272
f40	101,571	0.21	36.08	-9,965	-0.04	0.10	0.32	3,270
f41	97,639	2.50	249.91	-1	-0.08	0.03	0.22	62,671
f42	96,428	10,684.63	1,656,092.51	-1	-0.13	0.04	0.30	493,933,579
f43	84,378	0.61	22.45	-1	0.01	0.07	0.24	5,461
f44	101,880	0.09	0.29	0	0.00	0.00	0.00	1
f45	67,128	54.01	11,781.14	-1	-0.38	0.11	1.10	3,050,575

The table below presents groupwise medians for defaulted and non-defaulted companies. We compare their respective distributions using non-parametric Mann-Whitney U test.

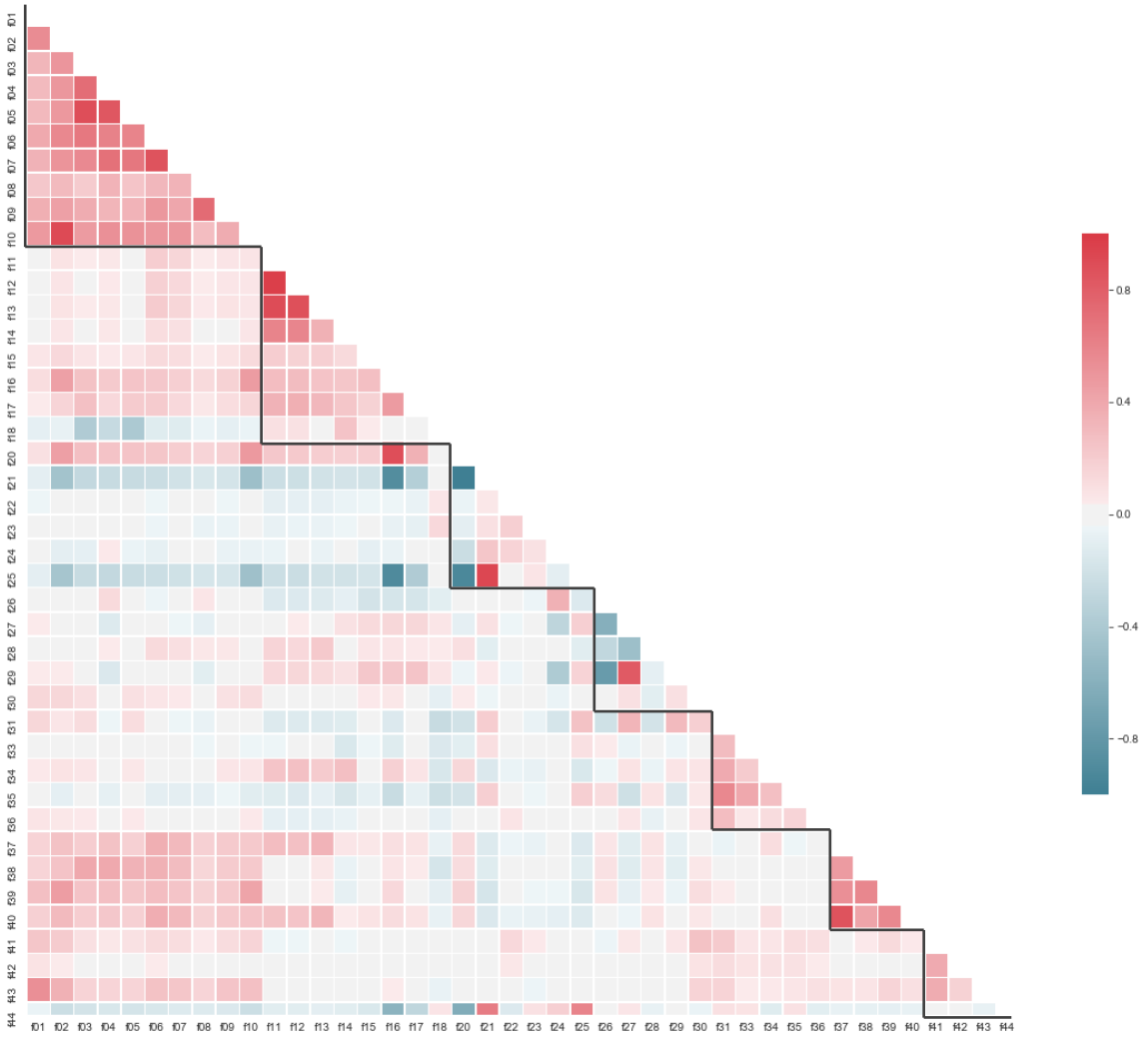
Vars[a]	Non-Defaulted	Defaulted	p-Value	Vars[a]	Non-Defaulted	Defaulted	p-Value
f01	0.057	0.000	**0.000*** **	**f23**	7.786	10.000	**0.000*** **
f02	0.024	-0.135	**0.000*** **	**f24**	0.000	0.033	**0.006** **
f03	0.016	-0.227	**0.000*** **	**f25**	0.453	0.964	**0.000*** **
f04	0.054	-0.053	**0.000*** **	**f26**	0.141	0.049	**0.002** **
f05	0.023	-0.164	**0.000*** **	**f27**	0.656	0.625	**0.007** **
f06	1.019	0.820	**0.000*** **	**f28**	0.000	0.002	**0.042** *
f07	1.024	0.859	**0.000*** **	**f29**	0.764	0.835	**0.000*** **
f08	24,272.374	11,690.141	**0.000*** **	**f30**	0.067	0.000	**0.002** **
f09	1,508.429	-9,806.000	**0.000*** **	**f31**	1.437	0.695	**0.000*** **
f10	0.032	-0.110	**0.000*** **	**f32**	8.812	5.334	**0.000*** **
f11	1.078	0.420	**0.000*** **	**f33**	5.390	3.034	**0.000*** **
f12	1.443	0.606	**0.000*** **	**f34**	3.297	0.770	**0.000*** **
f13	0.230	0.010	**0.000*** **	**f35**	2.400	1.867	**0.000*** **
f14	0.879	0.390	**0.000*** **	**f36**	3.112	0.000	**0.000*** **
f15	1.034	0.144	**0.000*** **	**f37**	0.158	0.000	**0.000*** **
f16	0.201	-0.316	**0.000*** **	**f38**	0.042	0.000	**0.000*** **
f17	0.120	-0.303	**0.000*** **	**f39**	0.057	0.000	**0.000*** **
f18	0.188	0.260	**0.000*** **	**f40**	0.096	0.000	**0.000*** **
f19	4.890	-6.595	**0.000*** **	**f41**	0.029	-0.154	**0.000*** **
f20	0.374	-0.102	**0.000*** **	**f42**	0.044	-0.310	**0.000*** **
f21	0.626	1.103	**0.000*** **	**f43**	0.071	-0.063	**0.000*** **
f22	1.085	2.000	**0.000*** **	**f45**	0.106	0.088	**0.000*** **

[a] Input variables (Vars) used for default prediction: f01 – Return on equity (ROE), f02 – Return on assets (ROA), f03 – Net profit from sales, f04 – EBITDA margin from sales, f05 – EBIT margin from sales, f06 – Overall efficiency, f07 – Operating efficiency, f08 – Value-added per employee, f09 – Net profit per employee, f10 – EBIT to total assets, f11 – Quick ratio, f12 – Current ratio, f13 – Cash ratio, f14 – Receivables to payables ratio, f15 – LT financing of LT assets and inventory, f16 – WCA to total assets, f17 – Working capital (WCA) to sales, f18 – Trade credit exposure, f19 – Interest coverage ratio, f20 – Equity ratio, f21 – Debt ratio, f22 – Debt/equity, f23 – Debt/EBITDA, f24 – LT debt to total assets, f25 – ST debt to total assets, f26 – Property, plant, and equipment to total assets, f27 – Working assets to total assets, f28 – Financial investments to total assets, f29 – Current assets to total assets, f30 – Capital expenditures (CAPEX) to non-current assets, f31 – Asset turnover ratio, f32 – Inventory turnover ratio, f33 – Receivables turnover ratio, f34 – Payables turnover ratio, f35 – Working assets turnover ratio, f36 – WCA turnover ratio, f37 – Current liability coverage ratio, f38 – Operating cash flow to sales, f39 – Asset efficiency ratio, f40 – Operating cash flow to debt, f41 – Assets growth, f42 – Revenue growth, f43 – Equity growth, f44 – Negative equity, f45 – Net income growth.

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Appendix 10: Financial Ratios' Correlation Heatmap

The figure below shows the correlation matrix between the pairs of features in the dataset. The darker the shade of colour the stronger is the linear relationship between the features, i.e. red colour depicts the positive relationship, whereas the blue colour corresponds to negative one. We can see that the financial ratios coming from the same group display higher pair-wise correlations (the groups are separated using black triangles). That is especially the case among the profitability (f01 – f10) and cash flow related ratios (f37 – f40). This gives a motivation to inspect the presence of multicollinearity and remove highly interdependent features.



Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Appendix 11: Feature Subsets based on Dimensionality Reduction in Step 2

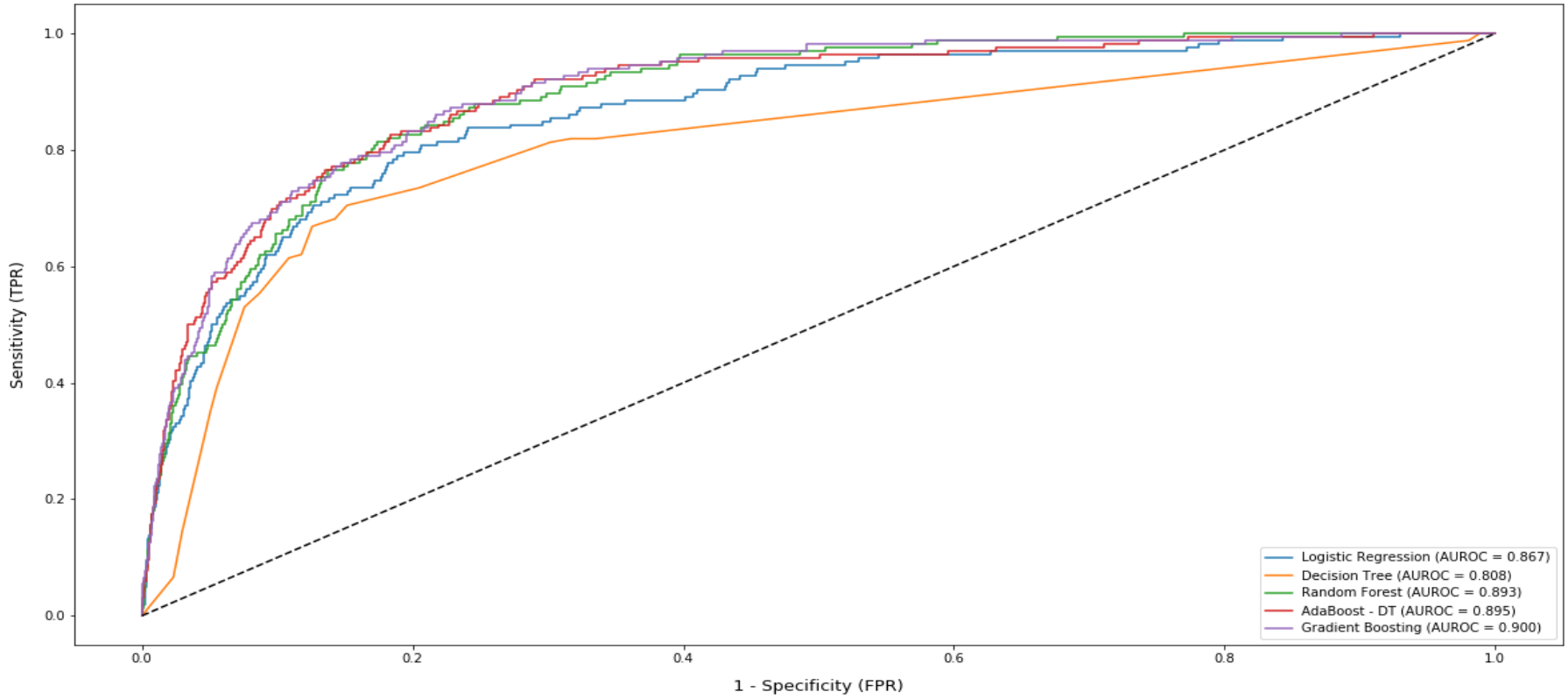
The table below lists the 15 most predictive variables based on the employed dimensionality reduction techniques. Additionally, we list the parameter settings for each technique.

Dimensionality Reduction	Selected Feature Subsets - TOP 15
Regularization	Depends on specific classification method employed
Information Gain - Score function: Mutual information - Number of neighbors: 3	Equity growth, Net profit per employee, Overall efficiency, EBIT margin from sales, Return on equity, Operating efficiency, Return on assets, Value-added per employee, LT financing of LT assets and inventory, Debt ratio, EBIT to total assets, Debt to equity, Current ratio, CAPEX to non-current assets, WCA to sales
MultiSURF	Overall efficiency, Operating efficiency, Return on assets, Value-added per employee, EBIT margin from sales, WCA to total assets, Debt ratio, Net profit per employee, EBIT to total assets, Payables turnover ratio, Equity growth, Negative equity, Return on equity, EBITDA margin from sales, Trade credit exposure
FS with Random Forest - Number of estimators: 100 - Max depth: 10 - Criterion: Gini impurity	Equity growth, Net profit per employee, Return on equity, LT financing of LT assets and inventory, Overall efficiency, Value-added per employee, Debt ratio, Return on assets, Payables turnover ratio, CAPEX to non-current assets, Operating efficiency, EBIT to total assets, Debt to equity, WCA to total assets, Revenue growth
Recursive Feature Elimination - Estimator: Support vector machines - Number of features to remove per iteration: 1	Negative equity, Net profit per employee, WA to total assets, Current assets to total assets, Trade credit exposure, WCA to sales, Return on assets, Debt ratio, CAPEX to non-current assets, PPE to total assets, Cash ratio, Return on equity, Payables turnover ratio, Asset turnover ratio, Value-added per employee
Genetic Algorithm - Fitness function: AUROC metric - Estimator: Random forest - Crossover probability: 0.8 - Mutation probability: 0.005 - Population size: 50	Debt ratio, WCA to sales, Net profit per employee, WCA turnover ratio, Value-added per employee, Revenue growth, Asset turnover ratio, Return on equity, LT debt to total assets, Equity growth, EBITDA margin from sales, Debt to EBITDA, CAPEX to non-current assets, WA turnover ratio, Debt to equity
Principal Component Analysis - Number of components: 15	EBIT margin from sales, WCA to sales, EBITDA margin from sales, EBIT to total assets, Debt ratio, WCA to total assets, Trade credit exposure, Operating efficiency, Return on assets, Overall efficiency, Return on equity, Operating CF to sales, Net profit per employee, CAPEX to non-current assets, Asset efficiency ratio
Linear Discriminant Analysis - Number of components: 1	Revenue growth, EBITDA margin from sales, WA turnover ratio, Debt ratio, Overall efficiency, EBIT margin from sales, Receivables to payables, Equity growth, Operating CF to sales, Assets growth, Current ratio, LT financing of LT assets and inventory, Assets efficiency ratio, Financial investments to total assets, LT debt to total assets

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Appendix 12: Comparison of ROC Curves for the Basic LR and DT Classifiers and the Three Ensemble Methods

The figure below plots ROC curves of the three ensemble methods discussed in this study and compares them with the ROC curves of DT and LR methods. We can observe that DT classifier performs the worst at each threshold. Much better performance is achieved with LR. As expected, ensemble methods (RF, AdaBoost – DT, Gradient Boosting) provide a further improvement in the AUROC metric.



Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Appendix 13: Effects of Employing Different Sampling Techniques

The table below summarizes the results related to using different data sampling techniques. The baseline scenario in this context is leaving the data imbalanced and comparing the methods' performance metrics with the scenarios where dataset was resampled.

Classification Method	Sampling Technique	Classification Accuracy	Average Precision	AUROC
Logistic Regression (LR)	Imbalanced Dataset	0.830	0.084	**0.858**
	Undersampling (MUS)	0.864	0.086	**0.868**
	Oversampling (MOS)	0.857	0.094	**0.885**
	Combination (SMOTEEN)	0.837	0.095	**0.886**
k-Nearest Neighbours (k-NN)	Imbalanced Dataset	0.992	0.059	**0.704**
	Undersampling (MUS)	0.862	0.067	**0.854**
	Oversampling (MOS)	0.933	0.034	**0.701**
	Combination (SMOTEEN)	0.871	0.036	**0.770**
Decision Tree (DT)	Imbalanced Dataset	0.863	0.062	**0.799**
	Undersampling (MUS)	0.799	0.039	**0.835**
	Oversampling (MOS)	0.866	0.062	**0.789**
	Combination (SMOTEEN)	0.840	0.052	**0.856**
Support Vector Machines (SVMs)	Imbalanced Dataset	0.992	0.019	**0.690**
	Undersampling (MUS)	0.830	0.088	**0.872**
	Oversampling (MOS)	0.854	0.093	**0.880**
	Combination (SMOTEEN)	0.832	0.096	**0.885**
Artificial Neural Networks (ANNs)	Imbalanced Dataset	0.992	0.081	**0.870**
	Undersampling (MUS)	0.806	0.090	**0.878**
	Oversampling (MOS)	0.959	0.071	**0.834**
	Combination (SMOTEEN)	0.942	0.084	**0.845**
Random Forest (RF)	Imbalanced Dataset	0.890	0.092	**0.894**
	Undersampling (MUS)	0.831	0.079	**0.886**
	Oversampling (MOS)	0.871	0.099	**0.893**
	Combination (SMOTEEN)	0.851	0.090	**0.888**
AdaBoost – DT	Imbalanced Dataset	0.991	0.071	**0.752**
	Undersampling (MUS)	0.836	0.124	**0.896**
	Oversampling (MOS)	0.991	0.060	**0.755**
	Combination (SMOTEEN)	0.983	0.071	**0.751**
Gradient Boosting	Imbalanced Dataset	0.992	0.116	**0.894**
	Undersampling (MUS)	0.839	0.127	**0.904**
	Oversampling (MOS)	0.870	0.133	**0.910**
	Combination (SMOTEEN)	0.869	0.093	**0.890**

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Appendix 14: Effects of Employing Different Dimensionality Reduction Techniques

The table below summarizes the results related to using different dimensionality reduction techniques. The baseline scenario is regularization on under-sampled dataset. Additionally, we employed five filter and wrapper feature selection techniques as well as the two feature extraction techniques as discussed in the study.

Classification Method	Dimensionality Reduction	Classification Accuracy	Average Precision	AUROC
Logistic Regression (LR)	Regularization	0.875	0.086	**0.867**
	Information Gain	0.888	0.060	**0.811**
	MultiSURF	0.890	0.062	**0.811**
	FS with RF	0.869	0.063	**0.827**
	RFE with SVMs	0.871	0.077	**0.847**
	GA	0.876	0.064	**0.810**
	PCA	0.859	0.076	**0.854**
	LDA	0.912	0.083	**0.875**
k-Nearest Neighbours (k-NN)	Regularization	0.877	0.073	**0.857**
	Information Gain	0.827	0.052	**0.846**
	MultiSURF	0.828	0.053	**0.844**
	FS with RF	0.826	0.067	**0.864**
	RFE with SVMs	0.835	0.071	**0.863**
	GA	0.830	0.058	**0.847**
	PCA	0.862	0.066	**0.853**
	LDA	0.778	0.061	**0.840**
Decision Tree (DT)	Regularization	0.816	0.038	**0.818**
	Information Gain	0.816	0.035	**0.814**
	MultiSURF	0.773	0.038	**0.809**
	FS with RF	0.825	0.040	**0.853**
	RFE with SVMs	0.802	0.040	**0.811**
	GA	0.792	0.042	**0.828**
	PCA	0.783	0.028	**0.768**
	LDA	0.790	0.041	**0.839**
Support Vector Machines (SVMs)	Regularization	0.863	0.081	**0.873**
	Information Gain	0.896	0.065	**0.826**
	MultiSURF	0.905	0.069	**0.831**
	FS with RF	0.860	0.058	**0.841**
	RFE with SVMs	0.865	0.081	**0.858**
	GA	0.878	0.058	**0.811**
	PCA	0.855	0.065	**0.850**
	LDA	0.901	0.083	**0.875**
Artificial Neural Networks (ANNs)	Regularization	0.810	0.076	**0.879**
	Information Gain	0.819	0.069	**0.862**

Continues on the next page

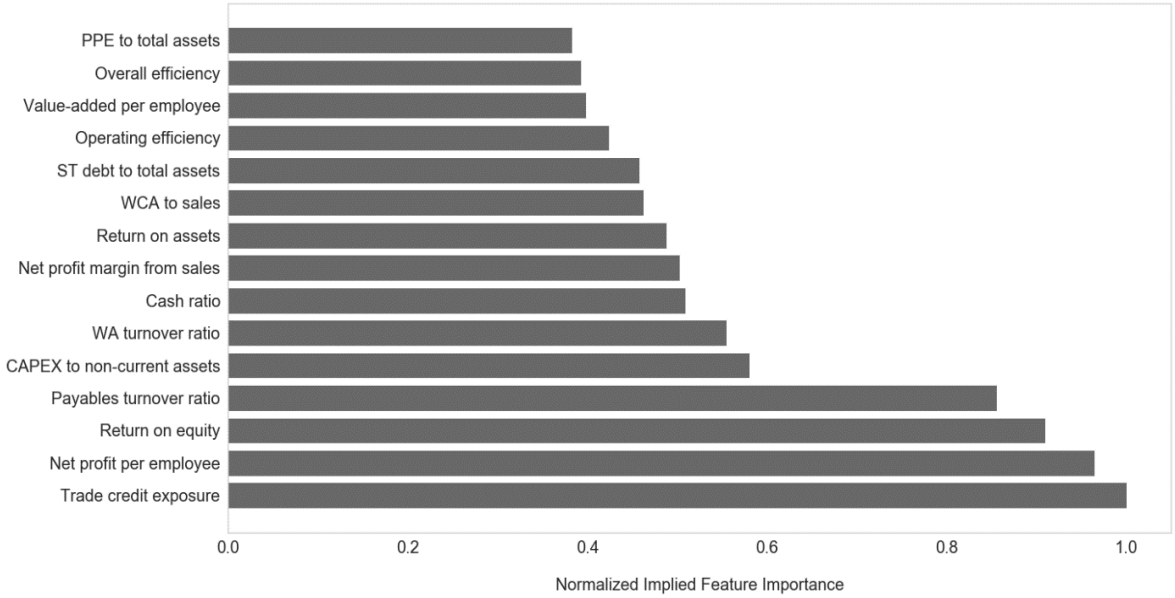
Table continued

Classification Method	Dimensionality Reduction	Classification Accuracy	Average Precision	AUROC
Artificial Neural Networks (ANNs)	MultiSURF	0.821	0.074	**0.866**
	FS with RF	0.818	0.083	**0.874**
	RFE with SVMs	0.789	0.079	**0.883**
	GA	0.743	0.069	**0.859**
	PCA	0.796	0.081	**0.878**
	LDA	0.771	0.083	**0.874**
Random Forest (RF)	Regularization	0.842	0.090	**0.889**
	Information Gain	0.831	0.070	**0.866**
	MultiSURF	0.818	0.092	**0.866**
	FS with RF	0.812	0.077	**0.883**
	RFE with SVMs	0.814	0.090	**0.874**
	GA	0.832	0.088	**0.878**
	PCA	0.822	0.067	**0.853**
	LDA	0.808	0.059	**0.861**
AdaBoost - DT	Regularization	0.835	0.096	**0.893**
	Information Gain	0.806	0.052	**0.855**
	MultiSURF	0.804	0.046	**0.859**
	FS with RF	0.808	0.079	**0.877**
	RFE with SVMs	0.811	0.079	**0.872**
	GA	0.817	0.087	**0.870**
	PCA	0.814	0.073	**0.847**
	LDA	0.741	0.034	**0.738**
Gradient Boosting	Regularization	0.841	0.125	**0.902**
	Information Gain	0.809	0.059	**0.862**
	MultiSURF	0.823	0.080	**0.868**
	FS with RF	0.816	0.073	**0.881**
	RFE with SVMs	0.819	0.075	**0.885**
	GA	0.815	0.091	**0.877**
	PCA	0.813	0.070	**0.865**
	LDA	0.794	0.067	**0.865**

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Appendix 15: 15 Most Predictive Financial Ratios Based on the Regularization Method

The figure below shows the 15 most important features in our dataset based on the normalized weights calculated from individual classifiers that supports either the calculation of implied feature importance from corresponding coefficients (as in the case of LR and SVMs) or directly implements feature importance method (as in the case of RF, AdaBoost – DT, and Gradient Boosting). Note that k-NN, DTs, and ANNs do not support this calculation and were therefore excluded from this part of the analysis.



Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.

Appendix 16: 15 Least Predictive Features Including Macroeconomic Variables

The table below shows the 15 least important features (including macroeconomic variables) in our dataset based on the normalized weights calculated from individual classifiers as in the case above. The output gives us an indication that macroeconomic variables do not have an important role in our predictive model. The reason for that might be that the variability in the macroeconomic factors is fairly low, since we only included half of the cycle into our analysis.

15 Least Predictive Features in Our Dataset
Current assets to total assets, Current liability coverage ratio, **SBITOP index growth**, Receivables turnover ratio, Quick ratio, Assets growth, EBITDA margin from sales, Revenue growth from sales, Operating CF to sales, **Inflation rate**, Debt to EBITDA, **GDP growth**, **Unemployment rate**, Equity growth, **Government yield**

Source: Slovene Business Register – AJPES (2019); Supreme Court of RS (2019); Own work.