Edited by

Daniel Kaszyński Bogumił Kamiński Tomasz Szapiro

CREDIT SCORING IN CONTEXT OF INTERPRETABLE MACHINE LEARNING

THEORY AND PRACTICE

SGH Publishing House

CREDIT SCORING

IN CONTEXT
OF INTERPRETABLE
MACHINE LEARNING

THEORY AND PRACTICE

Edited by
Daniel Kaszyński Bogumił Kamiński Tomasz Szapiro

CREDIT SCORING IN CONTEXT OF INTERPRETABLE MACHINE LEARNING

THEORY AND PRACTICE

SGH Publishing House

WARSAW 2020

Reviewers
Jerzy Michnik
Paweł Miłobędzki

Editor
Daniel Kaszyński

Proof reading
Jakub Hamze

First Edition

ISBN 978-83-8030-424-6

SGH Publishing House
162 Niepodległości Ave., 02-554 Warsaw, Poland
www.wydawnictwo.sgh.waw.pl
e-mail: wydawnictwo@sgh.waw.pl

Cover design
Marcin Kamieniak

Cover production
Ad Depositum

DTP
Daniel Kaszyński

Print and binding
QUICK-DRUK s.c.

Order 147/XII/20

Contents

Foreword

In the present competitive and efficiency-oriented economic realm, credit scoring plays an important role, impacting numerous aspects of economic activities and driving their successes. From the level of self-employment, through micro, small, medium size and large companies and institutions, to even multinational enterprises and institutions up to entire countries.

A profound understanding of credit scoring nowadays has become a must for every person who is professionally involved in dealing with problems related to decision making or risk management in financial institutions.

Since its conception, credit scoring has traveled a long way from statistical and probabilistic analyses and decision-analytic approaches that have characterized earlier traditional methods. In recent years, maybe decades, the complexity of credit scoring and its related aspects has grown rapidly. Moreover, the wealth of data that can currently be collected by financial institutions and their customers, as well as various institutions and agencies, has changed the landscape by the emergence of the so-called data-driven technology, better access to a wider variety of data, increased computing power, etc. All these have initiated a new wave of novel approaches to credit scoring, as well as a possibility to include in the analyses further aspects such as the pricing of financial services to reflect the risk profile of the individual, company or institution, etc.

In general, credit scoring has evolved in recent years – as a result of the *data-driven revolution* – from traditional approaches based on statistics, probability, decision analysis, etc. to new ones built on artificial intelligence, notably machine learning. This sparked a great interest in these modern approaches to credit scoring, laying foundation for a considerable research effort that has resulted in many publications as well as practical applications.

However, the addition of more methods to the toolbox of a contemporary credit scoring modeler poses new challenges related to proper procedures of developing, validating, monitoring, and finally implementing such models and approaches in practice.

This volume tackles the problems related to the above-mentioned new directions, challenges, difficulties and opportunities in modern credit scoring in an interesting, innovative and constructive fashion. *Credit scoring in context of interpretable machine learning* is a rare, extraordinary entry in the world literature that presents combination of explanation of theoretical concepts and approaches, as well as contemporary scoring practices.

The text covers both classical credit scoring methods and new methodological and procedural approaches having their roots in machine learning. Such a vast range of topics cannot be developed by an individual – it requires a multidisciplinary team capable of combining individual expertise, experience and skills, as well as an ability to share the courage to respond to a need of the community for a monograph in this area that would be simultaneously methodologically sound and friendly for the reader.

The volume is, therefore, a collection of chapters which have been written by the members of the team having different backgrounds, expertise, experience, as well as professional and research interests. The authors share the research philosophy and policy of Decision Analysis and Support Unit at SGH Warsaw School of Economics that emphasizes that at the center of any application of mathematical models, there should be a clear vision of how they will be used to aid and support decision making.

The monograph starts with a presentation of the historical and organizational setting of credit scoring and a critical review of data-related processes that are relevant for preparing credit scoring models. Afterward, being aware of the recent *data-driven revolution*, the exposition moves to the presentation of selected machine learning methods that can be used for credit scoring, with a special emphasis on variable selection methods which concern one of the key challenges of the modeling practice. The third group of topics that is covered are analytical tasks that are typically undertaken when a credit scoring model has been built, that is the model's performance evaluation, model monitoring and methods allowing to understand how complex machine learning models produce their forecasts. This is a crucial issue due to the fact that virtually all machine learning methods are of a black (grey) box type in their functioning and hence the results obtained, may be not comprehensible to the user. Approaches to overcome this inherent difficulty attract much interest in recent years and a new field of the so-called Explainable Artificial Intelligence (XAI) has emerged. The monograph is completed with a review of key aspects related to the deployment of credit scoring models in complex IT infrastructures, with emphasis on the most important problems related to the performance and scalability of the scoring process, as well as architectures and processes that can be used while implementing credit scoring models in the development of decision engines.

The editors and authors of the particular chapters have to be congratulated for producing a captivating read and useful volume. Their contributions present well-chosen aspects of modern approaches to credit scoring up to the highest academic standards, yet in a comprehensible and constructive fashion. They focused on the new and promising data-driven approaches, notably based on machine learning which will certainly dominate in the years to come.

Professor Janusz Kacprzyk, Ph.D., D.Sc.
Polish Academy of Sciences
Spanish Royal Academy of Economic and Financial Sciences
Bulgarian Academy of Sciences
Finnish Society of Sciences and Letters

Preface

A deep understanding of credit scoring became a must for every person who professionally tackles problems involving decision making or risk management in financial institutions. The complexity of this area recently grows rapidly due to the increasing versatility of financial products and business models, a surge of uncertainty on the markets and rise of intricacy of interactions between players on these markets. That manifests itself by the wealth of data that are currently collected by financial institutions. While the concept of credit scoring remains indisputable, traditional credit scoring methods are currently challenged by machine learning approaches that are very successfully used in many other application areas. However, adding more methods to the toolbox of a contemporary credit scoring modeler poses new challenges related to proper procedures of building, validating and monitoring such models in practice.

The volume *Credit scoring in context of interpretable machine learning* presents a unique, and simultaneously balanced, combination of explanation of theoretical concepts and contemporary scoring practices rooted in these concepts. We assume that the reader has a working analytical knowledge in fields of finance, mathematics, applied statistics and data processing algorithms. This monograph is prepared for one of the three main target groups: 1) finance and economics students (graduate or postgraduate levels), 2) risk practitioners (e.g., risk managers, data scientists, risk modelers, regulators, consultants) and 3) independent researchers (including academics, freelancing risk specialists, and machine-learning experts).

This monograph's primary purpose and the ambition we had while writing it, were to present the development and maintenance of the creditworthiness assessment process, emphasizing the applicability of the *state of the art* data analytics methods (i.e., machine learning). The modern transformation allowing *big data* driven decision-making, especially in credit scoring, is present not only in universal banks but also in many other entities that offer solutions based on trade credit. Today's standards and business practices require full transparency and audibility of the analytical methods used. The era of *black boxes*, the actions of which could not be understood and decision-making processes in which stakeholders are not able to precisely explain the impact of individual factors, has already ended in many industries and in banking itself probably never came. For this reason, in this monograph so many pages are devoted not only to the machine-learning methods themselves (supported by the Explainable Artificial Intelligence - XAI) but also to other aspects of building robust and trustworthy quantitative models supporting business processes.

The structure of this monograph is as follows:

- In **Chapter 2**, we discuss the data utilized in the process of credit scoring; we also emphasize the importance and particular techniques of data management.

- In **Chapter 3**, we review different techniques used for variable selection. Here we present some of our findings in terms of modern approaches and their superiority over classical procedures.

- In **Chapter 4**, the selected methods of credit scoring are presented. In this chapter we introduced both the classical scoring approach using *logistic regression* and also its modern competitors.

- In **Chapter 5**, we present the most frequent data-related issues and the solutions used to overcome them.

- In **Chapter 6**, the validation procedures of credit scoring models are presented. Our discussion is model agnostic – both classical and modern credit scoring can be validated using the techniques discusses in that chapter.

- In **Chapter 7**, we investigate the methods of explainable machine learning that are the way to ensure the transparency and interpretability of the *black-box* models.

- In **Chapter 8**, we discuss the most common performance issues related to building and maintaining of scoring models. We also review technological platforms that provide a scalable technology stack allowing to overcome these problems.

- In **Chapter 9**, we present the techniques used for integrating scoring models into credit decision making processes.

In this textbook, we discuss the credit scoring of consumer finance products, small and medium enterprises, based on both financial and behavioral data. For the experimental purposes, we utilize simulated, see Chapters 3 and 4, as well as empirical data, see Chapter 7.

This monograph is the result of exciting discussions and scientific seminars conducted by the team of the Decision Analysis and Support Unit, SGH Warsaw School of Economics, with business practitioners from the banking sector. The research conducted at the Decision Analysis and Support Unit focuses on using quantitative methods to support decision-making. This research, basic and applied, is currently carried out in strong connection with applications in four areas:

1. The theory of **decision making**, with particular emphasis on describing the preferences of economic entities and supporting individual and group decision-making processes.

2. **Optimization methods**, especially methods of finding approximate solutions to problems with a high level of complexity and characterized by uncertainty.

3. **Forecasting** the consequences of regulatory and managerial decisions.

4. **Simulating** complex business processes, in particular using the multi-agent approach.

As authors and editors, we are thankful to all the people who supported us with the preparation of this monograph. In particular, we would like to thank our loved ones who supported our efforts.

We would also like to send our sincere thanks to all those who supported us in preparing this monograph – your support on various levels enabled the efficient creation of this book.

In particular, we would like to thank: **Beata Czarnacka-Chrobot** (Dean of Collegium of Economic Analysis), **Joanna Plebaniak** and **Ewa Łukasiewicz** (for support in publishing this monograph), **Jakub Hamze** (for proofreading), **Marcin Kamieniak** (for preparing an exciting cover) and the reviewers **Jerzy Michnik** and **Paweł Miłobędzki** who pointed out many imperfections in the initial version of the monograph.

Finally, we would like to thank the main initiator of this monograph – the person who supported us both in formulating the first research hypotheses, as well as gave us many accurate comments on the whole material. Mr. **Maciej Kwiatkowski**, thank you very much for your valuable support – without your energy and commitment, this monograph would have never been published.

Editors

Chapter 1

Background of the credit scoring

Daniel Kaszyński

Decision Analysis and Support Unit
Collegium of Economic Analysis
SGH Warsaw School of Economics

The word *credit* originates from the Latin *cred* which can be translated as to *believe*, *suppose*. In its roots, the *credit granting* was strongly associated with the trustworthiness of the debtors. The institution of credit (which is a reallocation of financial resources from those who *suffer* from the abundance of capital to those who experience its scarcity) has been accompanying humans since the dawn of time. As early as the time of the Sumerian civilization (around 3500 BC), within the first known urban civilization with over 85% of the population living in cities, the institution of credit was used, in particular as a source of financing and liquidity for agriculture.

Ancient Babylon was also one of the first reported to use credit; the literature on the subject (Lewis, 1992), indicates notes made on stone tablets from 2000 B.C.E.: *Two shekels of silver have been borrowed by Mas-Schamach, the son of Adadrimeni, from the sun-priestess Amat-Schamach, daughter of Warad-Enlil. He will pay the Sun-God's interest. At the time* *of the harvest, he will pay back the sum and the interest upon it.* Other early references to credit and lending are from Babylonian civilization (c. 1800 B.C.E.); the Babylonian law, written during the reign of King Hammurabi which formalized and systematized the laws of credit. This code sets the highest possible levels of interest rates that could be used to grant loans: 33.3% per annum for loans for the purchase of grains, 20% per annum for loans for the purchase of silver. Besides, the code also introduced principles regarding the technical aspects of granting loans (today we would call it corporate governance principles), e.g. a loan, in order to be binding, had to be certified by a public official and written down as a contract.

The process of credit granting is associated with the belief and presumption of the lender about the willingness and capability of the debtor in terms of repayment. In fact, from the perspective of the lender, the proportion of failures in repayments (for the sake of argument, let us assume that the credits are identical) forms the *cost of doing the credit business.* That being said, *the riskier* the clients, the bigger premium the lender should charge to cover: a) defaults of its customers, b) administrative margin, c) cost of money and d) managerial premium. To be able to estimate the riskiness of the client reasonably, the lender should possess some information about the client (e.g. whether this client had any problems with earlier credits, the profitability of the business, the risk of the entire industry) and the mechanism or model to transform that data into the *risk score* or *credit score.*

1.1 History of credit scoring

Creditworthiness scoring is related to the process of accepting credit applications and is one of the oldest tools in both *data analytics* and *risk management* (Thomas et al., 2017). Its systematic use dates back to the 1950s – an early application of the loan portfolio in terms of risk management and diversification. Credit scoring and accompanying tools were also used for data mining; it was one of the first applications of consumer behavior data analytics. However, one can find in the literature that credit scoring was used even earlier, in 1820. At its beginning in the 1820s, credit reporting began to modernize which was a result of the popularization of lending and the need to adjust regulatory standards to the new market regime. New bankruptcy laws made loans a risky transaction which resulted in attempts to standardize approaches to credit assessment. Established by a merchant Lewis Tappan in 1841, the trade agency searched for information from customers all over the country regarding the information on debtors' assets – this was the aftermath of the first economic crisis caused by a financial crisis (Morawski, 2003).

The above-mentioned credit scoring attempts were entirely subjective and human judgment approaches; when assessing creditworthiness, analysts were guided solely by their experience and expert knowledge (Thomas et al., 2017). Reports from that period indicate that, for example, one should be guided by racial criteria when granting loans (Cohen, 2012). From today's regulatory standards point of view, such behavior is unacceptable and the credit decision itself must be supported by *transparent quantitative* measures/features. However, it should be emphasized that the credit sector in this period was significantly smaller, and the lion-share of loans were used by enterprises. At that time, loans were granted using a principle of minimizing losses – only safe transactions were co-financed by the loan, with a high level of security for these transactions (Thomas et al., 2017). At its origin, credit scoring referred to the process of accepting loan applications in banks (Oesterreichische Nationalbank, 2004).

Nevertheless, credit scoring does not need to be identified only with the banking approval process. It is also used today in many other processes where the customer signs the contract, most often committing to regular financial burdens (e.g. telephone subscription, T.V.), has to be pre-assessed in order to prepare the best terms of the contract, so that the institution providing the services would not risk too much. In the context of Big Data, the scoring analyses are a well-established example of data analytics used in a straightforward business process.

Philosophically, credit scoring is an activity related to the estimation of the borrower's creditworthiness (dependent on given macroeconomic conditions) that belongs to the trend of *pragmatism* and *empiricism* (Thomas et al., 2017). The leading goal of credit scoring is effective (in terms of the quality of the returned forecasts) and efficient (in terms of the resources devoted) estimation of the credit risk forecast. Credit scoring derives its improvement primarily from objectivity based on statistical modeling methods and empirical data on customer behavior. The above-mentioned philosophical trends support the use of available data regarding the consumer or his environment in order to improve the performance of scoring models. However, this is in contradiction with some regulatory standards which explicitly prohibit the use of sensitive data (e.g. gender, race) as the criteria determining whether or not to grant a loan.

1.2 Classical scorecard

The origins of credit scoring, as mentioned above, date back to the 1950s, when a consulting company, Fair, Isaac and Company, today known as FICO, created the first commercial scoring system. The underlying principles behind this system were related to the cost efficiency (time and resources) but also to increase the objectivity of the scoring process. Lenders no longer needed to employ so many analysts to perform expert-based scoring – what took the human-based scoring several days/weeks, now could be performed in a matter of minutes or even seconds (assuming the availability of data). Today's perception of the credit scoring model refers to the statistical approach of scoring which usually involves the creation of the credit scorecards (Anderson, 2007; Matuszyk, 2004; Thomas et al., 2017).

Credit scoring models are, in principle, statistical predictive models used to forecast future events – the default of the customer. In particular, most of the banks use the basic logistic regression, see Chapter 4, as a function that inputs a set of features into the credit score. The popularity of the logistic regression model is due to its simplicity and interpretability – the aspect which is getting more and more attention from the regulatory point of view but also due to business needs. As of today, the logistic regression in banking is widely applied – across IFRS 9 PD calculation, to capital requirements determination (PD models used in the Basel III framework for the calculation of Risk-Weighted Assets).

1.3 Data analytics revolution and its challenges

The methods of granting loans (both cash and installment loans) have changed significantly; the combination of credit market growth, technology development and the constant striving to optimize the profitability profile of companies somehow forced the search for other, more effective (i.e. generating better prediction) and efficient (i.e. faster) methods of assessing creditworthiness. The current method of lending is rapid – today, even in a matter of seconds, the end-to-end scoring process is performed. Such external market requirements exclude manual work related to the assessment of each credit application by an expert – expert panels assess only the largest credit exposures and the credit decision itself is delegated to higher decision-making levels (including the bank's management board). This is reflected in the significant increase in the use of formal, technology-supported mathematical models of creditworthiness assessment observed over the recent period (Stein, 2005).

One of the significant dangers of using credit scoring is that, just as any model, it is some kind of generalization. Over time, the scoring model may become obsolete and the evaluation made on its basis is no longer reliable (Thomas et al., 2017). This argument, however, points to a more generic one regarding each model used; it points to the need for continuous improvement and quality monitoring of the results returned by the model, see Chapter 6.

Currently, institutions regulating the banking sector are beginning to pay more and more attention to issues related to reliability of algorithmic processes. The EBA report of January 2020 (European Banking Authority, 2020) aims to identify the critical areas related to the use of Big Data and advanced analytics (BD&AA) approaches in the banking sector. The report emphasizes that the use of BD&AA is currently on the agenda of most of the institutions performing a technological transformation. The report highlights the essence of the following areas within the use of machine learning:

- data management,
- technological infrastructure,
- organization and governance,
- analysis methodology.

The implementation of BD&AA class solutions is, in the perception of the EBA, an element that significantly enters the area of trust. As indicated by the guidelines, the critical aspect is to develop such technological and procedural solutions that it is possible to develop trust in the socio-technical environments into which such systems are integrated. The use of trustworthy artificial intelligence is associated with the desire to maximize the benefits of using artificial intelligence systems while preventing or reducing the risks associated with their use.

In June 2018, the European Commission established the so-called *High-Level Expert Group on Artificial Intelligence* which is an independent group of 52 experts and professionals deeply engaged in matters concerning civil society. This group aims to develop guidelines and recommendations in the field of ethical frameworks (using existing regulatory standards) that should be followed by designers, implementers and end-users of systems using artificial intelligence algorithms. This framework should enable equal ethical rules to be established in all Member States. In December 2018, this group published draft guidelines on the so-called *Trustworthy Artificial Intelligence.* As indicated, this document "[...] *is intended to start a discussion on Trustworthy A.I. for Europe*" (European Commission, 2019).

The report defines that trustworthy A.I. should have three characteristics throughout its lifecycle:

1. Compliance with the law understood as compliance with all applicable laws and regulations.

2. Respecting ethical principles and values.

3. Technical and social robustness.

The indicated guidelines address issues related to Ethics and Reliability. Based on the above-mentioned features, the High-Level Expert Group on A.I. has prepared seven requirements that A.I. systems should meet. These guidelines provide a horizontal framework that enables the construction and implementation of trustworthy A.I. – however, as the authors point out, *different situations lead to different challenges.* The authors point to the need to develop sectoral approaches that take into account the specific conditions and the context of the planned operation of artificial intelligence systems. This monograph aims to present the procedures that will enable addressing the requirements indicated, among other things, in the above-mentioned guidelines related to the technical aspects of using machine learning methods.

1.4 People and processes

Credit scoring process should be developed using all of the elements of data mining framework such as *Cross-industry standard process for data mining*, CRISP-DM, such as:

1. Understanding the business purpose.

2. Understanding the data.

3. Hypothesis stating and modeling stage.

4. Model evaluation / validation.

5. Model deployment.

In terms of the credit scoring process and associated roles/responsibilities, we refer to Naeem Siddiqi (2012), as the source materials. The following main participants should be planned:

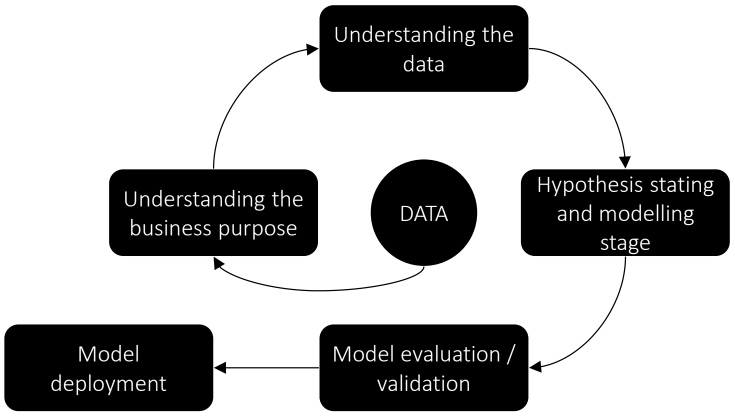

Understanding the data
Hypothesis stating and modelling stage
DATA
Understanding the business purpose
Model evaluation / validation
Model deployment

Figure 1.1: The Cross-industry standard process for data mining
Source: based on the Shearer (2000)

1. **Scorecard developer** – the expert in the domain of statistics or econometrics that performs the modeling stage. This role requires business knowledge related to the credit products that will be associated with the scorecard. Moreover, this expert should possess the knowledge about the data process in the institution but also some good understanding of the legal and regulatory requirements related to the model.

2. **Data scientist** – the expert who supports the data source management and extraction of necessary records/features. This role requires in-depth knowledge related to the process data within the institution but also data quality management practices followed by the organization.

3. **Risk manager** – the person responsible for managing the company's credit portfolio. This role requires to have the broad picture knowledge on the model itself (input data, development, implementation, monitoring, usage) but also have information on the company's risk appetite/tolerance.

4. **Product manager** – the person responsible for the management of the product/product branch for which the credit scoring model

is utilized. This person should have the information on the product-marketing strategies but also the target segmentation of the customers, as well as the scope and character of input data gathered. Nevertheless, the model formula is usually not shared with the product manager.

5. **Operational manager** – the person responsible for the on-going management of the model's related processes, e.g. claims, application processing, documentation or governance.

6. **Model validator** – the independent expert who overseeing the model's performance and model purpose (more on the validation process in Chapter 6).

7. **IT support** – the scoring model is always stored and used on some IT infrastructure. The IT support plays a critical role for the technical robustness, allowing the *data flow* to and out of the model (more on the technical aspects we cover in Chapters 8 and 9).

8. **Legal support** – a supportive role for the matters of regulatory compliance regarding the form and usage of the scoring model.

At the beginning of the scoring process, the legal aspects related to customer are verified – it is checked whether the applicant has provided correct information. Then, the rules that identify unreliable or suspicious customers are most often checked, which is referred to as the collective name – *verification with black lists.* In case the data from the application are incorrect, the additional steps of supervision needed to be performed (e.g. double-check with external source/credit bureau). All these types of activities should aim to minimize the risk of abuse.

The next stage in the process is to verify whether all the rules and recommendations made by the supervisor are respected, as the banking credit market is regulated and the supervisor has the right and obligation to protect the consumer. Most often, the recommendations refer to the guidelines limitation on too high debt to income ratio, too long tenor, too high price. Moreover, the *fair lending laws*, impose the obligatory unbiased treatment of all customers – they ensure that banking products and credit decisions are provided equally.

The next, essential stage is the use of the scoring model to determine whether to accept the exposure – approval of the single exposure should remain consistent with the organization's strategy in terms of risk appetite/tolerance. This process is handled within the banks' internal credit approval framework which includes the scoring cut-off threshold definition, i.e. defined explicitly and well documented. The approval process may include, apart from scoring itself, also additional factors (e.g. collateral which is not included in the scoring).

At that stage, a credit analyst may break the decision recommended by the system – it is called scoring's override. From a practical point of view, the bank usually, apart from the scoring cut-off threshold (i.e. a black scoring area), defines some thin range of scores (i.e. a gray scoring area) that allows individual decisions regarding credit application.

All the above-mentioned steps and interactions with the clients, as well as the accompanying documentation, should be registered and stored in the bank's database – usually, it is performed using the workflow system. Only then is it possible to learn from mistakes and continually improve the credit scoring process.

Chapter 2

Data processing for credit scoring

Maciej Kwiatkowski

In this chapter, data used for credit scoring and accompanying processes are presented and discussed. The structure of this chapter is as follows: firstly, the data sources are presented, with particular emphasis on the data sources available and used by the European financial institutions. Secondly, data management principles are presented; in the final section, data quality assurance procedures are discussed.

2.1 Data management

Data quality is a critical issue in credit scoring and the old saying "rubbish in - rubbish out" remains valid regardless of particular regression or machine learning technique used. With more and more explanatory variables added, the effects of using erroneous data (in terms of their quality) are becoming more complex and the final impact on the estimators/model parameters (i.e. its value or its stability) is hard to predict. On the other hand, the cost of cleaning such data increases proportionally to their amount.

In this monograph, we propose a set of rules and principles that may be implemented into the data quality assurance process:

- Understanding the target population.
- Understanding the data.
- Understanding the data feed process.
- Removing visible rubbish from the data set.
- Tight control on the quality of the outcome variable.

2.1.1 Understanding the target population

Over many years, the keyword for the target population in credit scorecards was *a homogeneous group of clients and products.* Nevertheless, the interpretation of what that *homogeneity* means was a somewhat arbitrary choice of the modeler, validator or regulator. Here, based on modeling experience and good market practice, we point out what choices have proved to be better and which of them turned out to be wrong.

The bad choices:
Building a scoring model on a product level (e.g. separate credit scorecards dedicated for cash loans, credit cards, overdrafts) is a reflection of product-centric silo organizations which leads to a complex model architecture where clients get inconsistent treatment depending on what product they require. In a modern omnichannel, customer-focused organization, it causes frustration of clients and sales force alike. Additionally, segmenting the target population may lead to a lower number of observations available for modeling and, therefore, lower predictive power and stability of resulting scorecards - we strongly suggest verifying results of the segmentation on predictive power and stability. Furthermore, as definitions of a “bad” client can significantly differ between products, it can lead to the aforementioned significant differences in predictive variables, especially for behavior scorecards; these differences in models have nothing in common with an underlying willingness and capacity, e.g. to repay the loan.

Development of the different scoring models for enterprises based on their balance sheet size or annual sales is an approach that is often a reflection of the internal classification of a client/company, assuming, e.g. their "small", "medium" or "large" sizes. This classification is subject to arbitrary credit policies and internal politics and is detached from the clients' characteristics that are further incorporated and analyzed with mathematical modeling. Moreover, as these credit policies may be frequently changed (sometimes several times a year), the models would have to be revised accordingly, adjusted and re-estimated.

The good choice:
Defining the target population based on available information. It eliminates the leading cause of instability of the model. The coverage of the input data may vary in time or across the portfolio (e.g. depending on company size), yet the predictive power of a particular input data set remains mostly unchanged and unrelated to particular product or business segmentation of clients.

Furthermore, in the case of different input data, different variables enter the model. A classic example is "reach hit" vs "thin file" vs "no-hit" segmentation of credit bureau scorecards. In the case of rich history in the credit bureau (usually defined as at least one currently reported credit product), socio-demographic data rarely enter the model and in the case of some very rich credit bureaus, they can be entirely skipped. On the other hand, scarce or no credit bureau information can usually be compensated with more socio-demographic information.

Another rule that should improve the data quality and data handling is to flag specific observations as "observation exclusions" or "outcome exclusions". The latter term refers to observations for which credit outcome, due to technical issues, cannot be determined with certainty (e.g. as a result of no credit activity of the client). "Observation exclusions" term specifies the clients with atypical behavioral profiles; including those clients into the training sample that might distort the model. Typical examples of "observation exclusions" include VIP, staff, fraudulent credit accounts, clients without relevant predictive information. While observation exclusions are generally a good practice, the fact that credit scorecard is used as an input to credit risk provisions under IFRS 9 and capital calculations under IRB opens a question of how PD parameters should be calculated for such clients. A common-sense solution of minimizing the number of observation exclusions in the first place and creating separate PD pools for those remaining usually works well. Clients with no relevant predictive information should be treated as observation exclusions anyway, which is in line with the concept of segmenting the target population based on available information.

It should be kept in mind that particular groups of clients can always be excluded from certain credit risk decisions even if the credit score is available for them (strategy exclusions, e.g. due to lack of customer consent for automated offering), therefore there is no point to overly limit the target population of the model.

2.1.2 Understanding the data

The data-mining approach based on sifting through terabytes is not welcome and appropriate in credit scoring (and broadly in credit risk). Even though sensational claims are made by some model vendors and top-shelf consultants about machine learning models finding predictive alcohol purchasing patterns and early indicators of a divorce, these claims did much damage to the perception of machine learning in the credit risk area. Firstly, to a credit risk expert, they sound much more sensational than credible; secondly, they spark concerns of the general public, journalists, regulators and members of parliament. Credit risk modeling has always been a delicate balance between the need to know the client and the privacy of the client. It is necessary to understand how to assess the client and that the client knows how he/she is assessed, the latter was even made a legal requirement in the European Union under the General Data Protection Regulation. It is also necessary to know which data are indispensable, as they exhibit cause and effect relationship to customer default and which are only correlated proxies.

A safe way to determine which of the vast databases available internally in the organization or on the market is relevant for modeling, is to look at what credit risk analysts were looking for over decades. Here we list some hints, separately for individuals and companies.

Individuals:

- the primary source of income and its stability,
- if salaried: quality and stability of the employer,
- if self-employed: profession, seasonality of the business, vulnerability to the economic cycle, entry barriers, business history,
- disposable income,
- an alternative source of income or support (e.g. a spouse, a family),
- the character of fixed expenses and their stability,
- assets (home, car, deposits, securities),
- saving rate,
- loans and credit lines currently held, their balance, installments and maturities,
- current and past delinquencies,
- means of contact,
- geolocation.

Companies:

- nature of business: seasonality, vulnerability to the economic cycle, entry barriers,
- business history,
- management quality,

- legal form,
- balance sheet (assets, liabilities, their character and length, financing sources in particular),
- income statement,
- cash flow,
- exposure to critical suppliers and key clients,
- concentration of suppliers,
- concentration of clients,
- current and past delinquencies,
- exposure to cross border risks (currency risk in particular).

The primary source of data sets for credit scoring is usually within the credit institution. Banks administer databases of account histories including information concerning the income, expense, financial assets, financial liabilities. Economic dependencies, concentrations and cross-border exposures can be discovered from fund transfers. Sales, as well as geolocation of clients, can be extracted from point-of-sale terminals (POS). Nevertheless, two critical factors have to be handled carefully: data completeness and privacy.

The data completeness refers to capability of determining the client's characteristics (with relatively small error) given a data set – the more completed data set, the better the predictions. We cannot make a spending profile of a client based on the data from a single POS terminal even if it is the leading shop in the client's village. We cannot tell about the client's willingness to pay if we see only delinquencies on our products and not those in other banks. In order to address the data completeness issue, various institutions have been established which will be described further in this chapter.

On the other hand, even a piece of isolated information like a hospital bill or a legal bill may indicate a severe life accident which is vital for credit risk assessment, but here is where privacy regulations come into play. Even though some general rules apply in the European Union, different compromises have been worked out in different countries. Therefore, it is crucial that the data set prepared for modeling does not contain prohibited information and even data strongly correlated with prohibited information. It means that the input data and final model always needs to be manually scrutinized for privacy issues and consultation with the compliance department is necessary.

The availability of some data is subject to customer consent. Especially the clients who have derogatory or sensitive information in some databases may attempt to withdraw previous consents. Such optional data should be identified before modeling. A safe approach would be to remove them from the modeling data set entirely. In the case of primarily positive information, it may still be considered as long as its withdrawal does not improve the client's risk assessment. In any case, optional data should be checked for stability and consistency over time.

Furthermore, the GDPR (European Parliament, 2016), imposes an obligation to inform each person that his/her data is processed. It is easy to do in the case of existing clients of the institution, as there is usually a convenient e-mail communication channel or in the worst-case scenario monthly paper account statements are sent regularly. In the case of prospects who do not have any relationship with the credit institution yet, it may be quite onerous, especially if paper letters need to be sent with a low (e.g. 0.1%) conversion rate to approved loans. In that case, the unit cost of acquiring a new client would be 1 000 times the cost of a letter.

Information duty may result in high costs the when processing of social or business networks takes place, as it may apply not only to the client itself but also his or her surrounding (both internal – e.g. employees, stakeholder but also external – suppliers, providers, vendors, etc.). It may also apply if, i.e. a payroll of a corporate client is processed in order to figure out the cost of its workforce. A solution to this might be through anonymization or pseudonymization of the data, as it is usually not the identity of a connected person that feeds the credit risk model.

Finally, the cost of information duty may lead to a complete removal of specific data fields and observations, e.g. personal data of company management board or records of sole company owners from a purchased commercial data register. This, in turn, may drive reconsideration of the model architecture and target populations of individual scorecards, mainly if this target population is driven by data availability.

2.1.3 Understanding the data feed process

Not only does the scope of data (e.g. predictive variables and target population) matters. What is equally essential is the refreshment frequency. Some databases (e.g. current accounts) are refreshed online, some others (e.g. financial statements) are refreshed only annually, some (e.g. country statistical office) are provided with a considerable delay. In most credit risk scorecards timing matters. As we cannot change the timing of external data sources, the development data sets should be prepared in a way that these timing differences and delays are reflected. The development data set should reflect explanatory data as they would likely look like in a production process when the predictions are calculated (e.g. it may be assumed that the financial statements are 6 months old or the observation window may be structured in a way that 1/12 of observations have financial statements 1-month-old, 1/12 have them 2 months old etc.; in production, the timing of the data feed should be monitored). A significant and permanent change in the schedule of the data feed may result in a need to redevelop the scorecard.

An important issue is the implementation of machine learning models. Prior to the implementation of modern decision engines which could be programmed by end-users in the credit risk department, implementation of simple credit scorecards with 10 to 20 variables took months, sometimes even more than a year. With modern decision engines, it can be reduced to weeks, yet the cost and complexity are more than proportional to the number of variables. Any manual coding, copying from one programming language (like SAS) to another one (like COBOL) is out of the question with machine learning model. New generation decision engines support copypaste model deployment or PMML (Predictive Model Markup Language). Nevertheless, none of these methods will cope with changing the names of variables. Neither will it cope with pre-processing. Therefore, the models should be developed on precisely the same data feed which will be used in production, with all the pre-processing steps.

Furthermore, the timing and the stability of the data feed, especially if various data sources are connected, should be monitored. There is no good credit scorecard without a good credit process in which it is embedded. It may be an application process, a monthly or quarterly portfolio monitoring process or an annual credit review process, yet there is little freedom on what data are provided to the model and how old they are. In some cases different processes will require different models, too.

2.1.4 Removing visible rubbish from the data set

The advanced modeling methods presented in this monograph will not eliminate the problem of incomplete, unstable and too granular data. Some of the discussed methods may become unstable and inefficient facing the data quality issues. Adding plenty of variables that do not exhibit explanatory power (even if it does not deteriorate the power of the model much due to regularisation mechanisms) will increase the computing cost and prediction time. Furthermore, it will increase the operational risk of failure in case of problems with the data feed. Last but not least, it can make the costs of model monitoring prohibitive. Therefore, while traditional scorecards with 10 to 20 variables were an oversimplification of credit risk assessment, models with thousands of explanatory variables seem to contain mostly irrelevant ones. Here are some good practices on how to identify irrelevant variables:

- Variables with coverage greatly varying in time.
- Variables provided optionally, either by the client or by the data vendor.
- Dummy variables indicating missing data (as long as they are highly correlated due to a data feed missing entirely).

- Customer identification data.
- Categorical data with a vast number of categories, for which no hierarchical data dictionary (coarse categorisation) exists.
- Data raising privacy concerns (e.g. revealing religious, political or sexual preferences).
- Variables reflecting treatment of the client and policies of an institution rather than the client's behavior (e.g. fees, interest rates).
- Unprocessed log files (e.g. application data log, collections communication log).
- Unprocessed transaction titles.
- Unprocessed transaction counterparties.
- Unprocessed contact details.
- Unprocessed addresses and zip codes.

It should be noted that commonly used machine learning algorithms poorly cope with unstructured categorical data with a vast number of categories. They are prone to overfitting as much as more traditional logistic regression or decision trees. Nevertheless, such data may bear important information and some pre-processing is necessary. Examples of pre-processing include:

- Categorising current account transactions (based on merchant codes, internal transaction codes or with text mining techniques).
- Creating summary variables (indicating count and volume) for each transaction category.
- Creating summary variables (indicating count and frequency) of contacts with the client and character of these contacts.
- Creating descriptive variables of transaction counterparties (e.g. nature of business, balance sheet size).

- Creating descriptive variables of geolocation area (e.g. urban or rural, local unemployment rate, local average salary, etc).

2.1.5 Tight control on the quality of outcome variable

While the quality of hundreds or thousands of potential explanatory variables might be challenging to handle, the target variable is the one that can decide on the quality of an entire model. Compared to traditional credit scorecards, the approach remains unchanged.

In terms of recent experience and good practices, here we provide some vital issues that have to be addressed:

- Delinquency definition is crucial, especially for individuals and small and medium enterprises. Some product definitions (balloon loans, bullet loans, credit lines with no significant monthly repayments) can compromise it severely. Some products are only delinquent when not renewed by the bank which makes any credit risk model a self-fulfilling prophecy. It cannot be addressed by the modeller alone. Product managers and credit risk managers should work on this issue otherwise business threatening credit risk losses might occur, as the authors experienced several times.

- A restructured loan is usually a bad loan. EBA guidelines on forbearance and the forbearance status it defines, is a good definition to follow.

- Last-in-first out (LIFO) is a bad Days Past Due definition. First-in-first out (FIFO) is much better. A client with one monthly installment overdue who pays regularly is a good client, even if 90 days past due under LIFO.

- Additional (no-DPD) default reasons as defined in EBA guidelines on default definition are a good further indication of *bad behavior*, especially for products with compromised DPD calculation and for corporate clients.

- It is better to use a less precise “bad” definition which is consistent across clients in the target population and over the observation window, rather than to use an exact and less consistent one. Inconsistencies will be reflected in predictors entering the model. They will predict these inconsistencies and not the client’s behavior.

- 12 months outcome window used in IRB PD models is too short, especially for 90 days past due trigger of *bad behavior*. Customers can juggle with cash loan consolidation and credit card balance transfers for much longer before they miss a payment. 18 to 24 months outcome window is a good pragmatic compromise between capturing such behavior and developing the model on reasonably recent data.

- Outcome variable must be defined on the customer level, taking into account all products. Otherwise, loan juggling and refinancing behavior may cover a bad outcome.

- Clients without significant credit activity (e.g. repaying their loans in less than 6 months or having only untouched credit lines or having only minimal balances due to charged fees) are not good clients. There is no evidence to say so. Prior inactivity is correlated with future inactivity and lack of activity will indicate a lack of risk if such clients are flagged as a good outcome. This is wrong, as it may indicate that the bank is not the first choice provider of funds and it will be used as the last resource lender. Inactive clients should be marked as outcome exclusions.

2.2 Data sources

As mentioned in the previous section, the principal recommended data source for the modern credit scoring process is the financial institution itself and various institutions integrating data: credit bureaus, public commercial registers, economic information integrators, public statistical offices, third party payment providers (acting under EU PSD2 directive). The main difference from modeling and implementation perspective is that internally available data come at no additional cost, whereas the external data may require customer consent to get them and they may have to be paid for.

2.2.1 Own data

Own data has a form of flat tables and it is usually made available in monthly and daily batches and contains customer transactions since the previous batch and end of day or end of month account status data. Using monthly data was a market standard for many years, except for collection scorecards, where the behavior score is best calculated on collection entry date with the most recent daily information.

In both cases the scorecards calculated on internal data are calculated in a monthly or daily batch, matching the monthly or daily structure of input data. In the case of scorecards combining internal information with external information received online, at least predictive characteristics are usually pre-calculated in a batch and combined with online information later, in order to save the computing power of online decision engines.

While there are attempts to calculate daily or even online behavior scores with robust modern IT technology, this will remain out of the scope of this publication as authors do not have enough evidence to evaluate added value compared to a more traditional monthly approach.

There are the following categories of internal data that are necessary input for credit scorecards and all efforts should be made to identify them. Their lack in the model should not be covered by other loosely correlated information:

- delinquencies,
- the mix of products held by the client,
- balances on products held,
- available credit lines,
- cash transactions, number and volume,

- incoming funds transfers (whenever possible, funds transfers between accounts belonging to the client and his/her household members should be shown separately).

Additional internal information which may be useful is the following:

- history of credit applications made,
- history of contacts with the client and character of these contacts,
- the character of incoming funds transfers (salary, invoices paid, social security, pension),
- identification of employer (via incoming salary transfer),
- internal transaction codes,
- merchant codes whenever transactions are made with a credit/debit card,
- geolocation of transactions made with a credit/debit card.

Annual history is most often used; nevertheless, the authors recommend using more extended history as long as it remains available and continues to be predictive).

Internal behavioral data can be used in the form of raw data (e.g. current loan balances), in a form of an indicator, count or summary variables (e.g. the number of loans held, the balance of new loans taken in recent 6 months, the fact of being 30 or more days delinquent in the past 3 months etc.).

The history of transactions on current accounts is first pre-processed based on transaction codes, merchant codes, transaction descriptions and sender or beneficiary of given funds transfer. The text mining techniques and classification techniques used for this pre-processing are out of the scope of this monograph.

2.2.2 Application forms

A commonly used source of information is an application form (in case of individuals) and qualitative assessment form (in case of enterprises). The quality of both varies greatly and it is to large extent dependent on the quality of data dictionary and attitude of salesforce supporting data gathering process. In recent years, the use of application forms and qualitative assessment forms is declining due to following reasons:

- Financial institutions compete in terms of minimizing the processing time of credit application and overly detailed questions may discourage clients as the required information may not be available on the spot.
- The number of questions the client can answer on a mobile phone screen is physically limited and lengthy application forms, due to technical issues, tend to crash.
- Sales force is no longer motivated to provide correct data in application forms or qualitative assessment. They are motivated to provide answers most likely resulting in credit approval.
- Application forms are easily manipulated by fraudsters, who can use internet channels to decode traditional credit scorecards inputting multiple credit applications.

We recommend using external data sources to compensate for the lack of application forms or qualitative assessment forms; in the following part, the additional external data sources are more broadly discussed.

2.2.3 Credit bureaus

The credit bureaus are institutions originally established to exchange credit information between banks; shared information is subject to dedicated regulations on sharing secret information (subject not only to privacy protection laws but also to the banking secrecy law). The mission of credit bureaus is to share credit information, in order to prevent clients from switching between banks and covering their bad debts, as well as to prevent excessive debt taking behavior. The credit bureaus can be run by the state (e.g. Belgian credit bureau owned by the central bank), privately owned (e.g. British Experian) or owned by a consortium of banks (e.g. Polish BIK). The ownership form is heavily dependent on the local regulations adopted in the country. In some countries more than one credit bureau exists which leads to differences in the coverage, data structure and scope of information. The scope of information also varies strongly by country, depending on what legal compromise was struck between the benefits of the banking sector and the customer's privacy.

Completeness of coverage is an important quality feature of a credit bureau. In countries with a high concentration of the banking sector, dominant banks can have a significant impact on how credit bureaus operate, what data are made available and for what price. In order to increase the coverage, in recent years also non-banking credit institutions got admitted to credit bureaus: specialized consumer finance companies, leasing, factoring, financial co-operatives, etc. As there may be separate laws governing data exchange between these institutions, the scope of available information may vary depending on institution type and this should be carefully examined before building the model. Information coming from non-banking institutions is best flagged separately for model development. The type of institution from which the client has taken a loan may also be predictive.

Initially, credit bureaus were set up to share information about individuals. Over time, small and medium enterprises were added to the scope, yet most banks withdraw information about large corporate clients. Information about enterprises is usually provided as a separate data set which may even have a different format of data or at least different data dictionaries. Small entrepreneurs may have some loans in the data set for individuals and some other loans shown in the data set for enterprises (in a different format). Some form of data integration is therefore recommended before developing a credit scorecard for this segment.

A credit bureau may share both negative and positive information. Negative information consists of records about unpaid loans; positive information consists of the records about loans paid regularly. Note that generally positive information may also negatively impact a customer's credit score (e.g. when it shows an excessive level of debt). Depending on the credit bureau, only negative information may be shown or clients may have an option to withdraw some of the positive information. This should be taken into account in modeling, by proper construction of predictive variables or by giving adequate coefficients to missing information in order to avoid manipulation of input data.

Credit bureau report is usually provided in XML format (the structure changes depending on the content) and it consists of following sections:

- customer identification,
- identification, timestamp and description of inquiry, in response to which the credit bureau report is provided,
- summary information (number of open loans, number of loans in the record, open balances, number of delinquent loans, balances on delinquent loans, days past due),
- history of credit inquiries,
- details of loans taken (original terms and conditions, current status, delinquency and balance),
- history of loans taken (in monthly or even daily intervals: timestamp, balance, status, delinquency status, days past due).

Some credit bureaus may contain information from courts, bailiffs, social security or tax institutions.

Most credit bureaus flag which records come from an inquiring institution. Some show pseudonymized identification of a credit institution so that records coming from the same contributor can be identified. No credit bureau known to the authors shares the name or publicly known id of the credit institution which contributes the records.

Many fields in a credit bureau report are optional and they have a considerably lower coverage than mandatory fields. The specification of credit bureau data should be checked for these optional fields prior to modeling and coverage of each optional field should be examined. Many areas may be discarded from modeling based on such analysis.

The data integrity in credit bureaus varies. The authors have experience of significant mismatches between summary information and details of loans taken, duplicate records, the same loan shown as separate records of different dates, etc. Even though the quality of information improves as credit bureaus mature, some form of data cleaning is practiced by the modelers in a bank. Authors do not recommend using summary information and instead suggest working on detailed data. Machine learning algorithms facilitate such an approach.

Credit bureau data can be obtained in two modes: online and offline. The price for both types of inquiries can vary and it often depends on the total annual volume of records purchased or contributed to the bureau. Some bureaus offer a considerably cheaper alerting service where pre-defined triggers (like delinquency, taking a new loan) are set and only clients hitting these triggers are reported to their credit institution. The cost of including credit bureau information in quarterly or monthly intervals for credit scorecards for all existing clients can be prohibitive. Therefore, full credit bureau information is often used only for clients applying for loans in an online mode and for a complete database of existing clients, a cheaper alerting service is used.

The data coming from credit bureaus can be used in their raw form (summary records), indicator variables based on full history records (indicating a specific event on a given loan type in given time), in a form of summary records defined by the modeler based on these indicator variables and in form of ratios, e.g. credit line utilization ratio, ratio of delinquent loans to all loans reported over the last 12 months etc.

2.2.4 Public commercial registers

Public commercial registers hold information about enterprises registered in each country. They are centralized on a country level. The scope may differ as some will only include companies while others will contain records of physical persons running registered trade activities (self-employed or sole traders). The range of information varies from one register to another but one should reasonably expect at least the following:

- identification of the company,
- legal form and a record of its changes,
- identification of the company owners and the history of changes,
- identification of company management and a record of changes,
- standardized code of economic activity (nature of the business) and a record of its changes,
- date of establishment,
- current status of activity (ongoing business, suspended, in liquidation) and record of its changes.

Furthermore, company financial reports are increasingly made available by these public commercial registers, as more and more companies deliver them in an electronic format. The discipline of providing them to public commercial registers in a timely manner is also increasing. Some public commercial registers also provide additional information like a number of employed staff which may be very useful for the modeler.

It is important to note that both recent information and record of changes are beneficial from a credit risk modeling perspective. Furthermore, identification of company owners can bring a piece of additional information like their age or companies owned or managed before. Personal credit bureau records of company owners may be verified. In case the owner is another company, more detailed information about that parent company can shed additional light on the credit risk of the company analyzed.

2.2.5 Financial statements

The financial statement is a set of information about the material situation and financial result of an economic entity. It is prepared for a given date and a given preceding period, most often the end of a calendar year for the previous calendar year. As the preparation of the annual financial statement may take some time, usually a quarter is allowed for preparation and a further quarter for all necessary approvals before the publication is made. Therefore, a financial statement is not an immediate source of information about the financial condition of a company and needs to be supplemented with more recent data, e.g. internal behavioral information or credit bureau information.

Most annual financial statements are audited, i.e. verified by chartered accountants who are not employees of the audited company.

Some companies prepare quarterly financial statements which are supposed to provide more recent updates. Nevertheless, modeling practice shows that they are of inferior quality and using audited annual reports, albeit old, leads to better quality models (with better predictive power and stability in time).

Financial statements comprise of:

- introduction,
- income statement,
- balance sheet,
- cash flow analysis,
- additional disclosures and clarifications.

The balance sheet is a summary of assets (i.e. what the company owns) and liabilities (i.e. what the company owes and how its activity is financed).

The income statement is a summary of revenues and costs. A recognition of revenue or cost in the income statement does not mean that it has been paid. An essential element of the income statement is depreciation, i.e. the cost of acquiring fixed assets is not immediately recognised but spread (amortised) over time.

The cash flow statement reflects what funds have been received and what have been paid. The main difference with the income statement is that it shows the ability to collect money from sold goods and services and the ability to defer payments for purchased raw materials. Additionally, the money spent to purchase fixed assets is not amortized over time.

All three elements of financial statements must be used together. Many companies went bankrupt despite showing high growth and income, as they were not able to finance and operationally manage further expansion. The financial statement not only helps to assess the probability of bankruptcy or default but it also helps to determine the financial needs of a company and propose a targeted product (a credit line or a loan of a particular tenor and amount).

It should be noted that in order to prepare a financial statement a company should run double-entry accounting, in which corresponding entries are made in the balance sheet and in the income statement. Not all companies are obliged to do so (depending on company type and annual turnover below the regulatory threshold). As a consequence, not all companies prepare and report financial statements.

It should also be noted that various companies (e.g. depending on the size or nature of business) report financial statements in different formats. Furthermore, the definition of specific fields may depend on whether local or international accounting standards are used. Information on the standard user may be found in additional disclosures.

Depending on the format, in which the financial statement is provided (flat table or XML) irrelevant fields may not be provided at all or they may be provided as null or even zero values. Especially zero values may be misleading, as in some cases they will represent a genuine zero and in some other cases missing value or an irrelevant field.

2.2.6 Alternative sources of information when financial statements are missing

Some of the alternative data sources may include tax statements, full transaction record (simplified accounting) or a full VAT transaction record (e.g. available in a standardised electronic format in Poland as a mandatory reporting to tax authorities). Especially the full transaction record and full VAT records are practical, as they show a complete picture of the company's turnover. For existing clients, current account records may provide similar accuracy as long as they are complete (e.g. the company has no current accounts elsewhere).

Full transaction record and a full VAT record need pre-processing before modeling, similarly to current account transaction records. The techniques used for such pre-processing are out of the scope of this publication.

2.2.7 Integrators of economic information

The core business of integrators of economic information is to integrate publicly available information from publicly available commercial registers, municipalities, courts about the company, its contracts and financial statements. For many years the key added value was to meticulously copy information available on paper or in free text electronic documents and integrate it into a standardised format while also, to some extent, eliminating format differences. As electronic forms of recording financial statements have been standardized, it made the job more comfortable, however the added value of integrators has decreased. The key benefits of integrators that remain in place are:

- Possibility to obtain the full database of companies registered in a given country and abroad (from countries in which the integrator operates).
- Possibility to track historical records.
- Possibility to obtain regular updates.

- Possibility to develop own applications which explore ownership links between various companies and individuals, also on an international scale.

Furthermore, many integrators also run other businesses, credit insurance in the first place. The credit insurance business gives them an additional source of information on customer behavior which they may share in a form of their credit scorecards.

2.2.8 National statistical offices

Even though national statistical offices gather information from individual companies, the published information consists of aggregated data. Nevertheless, the information from national statistical offices remains a useful source of information as it enables benchmarking of companies with their peer groups.

Note that financial statements are difficult to compare across different economic activities. A wholesaler may have an enormous turnover compared to assets and a low margin. A legal agency may have little or no assets at all, high wage bill and high margins. One possible way to address this issue is to compare financial results to benchmark (average) financial statements for each industry, as such simplified financial data are available from statistical offices. Then some measure of comparison against the benchmark can be used for modeling instead of raw variable.

Another way of using statistical office data is to attach specific characteristics to geolocation data (i.e. zip codes), e.g. local unemployment rate, population density and its changes and to use these characteristics in modeling instead of raw geolocation data.

2.3 Data quality assurance

The building blocks of data quality assurance are monitoring, identification of issues, remedial of issues and safety by design. First of all, the financial institution should have established processes to monitor data (and its quality) which is utilized for credit scorecards. Then the issues should be remediated at source, meaning where the error occurs in data processing or in the process of data collection. Furthermore, the models and credit policies should be constructed in a way that the impact of a single data failure is limited.

2.3.1 Data quality monitoring

Monitoring of data quality can be set up as two fundamental processes: an end to end manual check and automated scans. As the cost of running an entirely random manual test of thousands of input data may be prohibitive, a pragmatic approach is to use automated scans and then manually test suspected cases to identify the root cause of the problem.

The end-to-end manual test is to track suspected data elements (e.g. customer's income) from the primary source of data to the flat table feeding the calculation of the model, checking each intermediate calculation and documenting the result. The automated scans can be classified into the following categories based on records of an individual client:

- Detection of multiple inputs of the same initial data (e.g. financial statement, application form) whenever it leads to significant changes in a short period. *Short period* may mean different things depending on the data type. For example, it is not expected that the financial report significantly changes over a few days in December or that two children disappear from the application form in just two weeks.

- Values of single variables significantly different from typical values for the target population.

- Combination of variables significantly different from typical values for the target population (e.g. a young age person with a very high income),

- nonsensical values (e.g. days past due increasing from 0 DPD to 180 DPD over one month, a delinquent account with no reported balances, activity on a formally closed account).

- Missing values where the values are mandatory.

Based on a full batch of records feeding the calculation of credit scorecards:

- significant changes in the distribution of the input variable,
- a significant number of unjustified missing values in a mandatory field (note: missing values are justified when reasonably expected, e.g. utilization of credit lines is not expected if the client only has installment loans),
- significant increase in missing values in a non-mandatory field,
- sudden and significant loss of predictive power of any single and originally strongly predictive model variable.

The exact list of automated scans can consist of hundreds of positions and it is customized to the data sources, core banking systems and the model itself.

Automated scans present a summary record of issues detected, number of cases, percentage of cases with issues to the total size tested population and finally a list of few typical cases for manual investigation.

Specialized systems exist to support automated scans of input data. In most cases, they are marketed as fraud prevention tools or anomaly detection tools, yet they can also be used in a broader context of data quality assurance. Some of them utilize complex algorithms not only to pick up the problem but also to describe it. Details and examples of such tools are out the of scope of this publication.

2.3.2 Identification and resolution of issues

The identification of issues is based on a full tracking of suspected data elements through processing steps. Root causes vary and they may range from outsourcing of data entry activities, through the input of *"end of file"* character in a text field of input data set, lack of proper regression testing when ETL (i.e. extract, transform, load) process is modified, just to name a few from the author's practice.

As it may sound radical, it is recommended to remediate each data issue at the source, i.e. where it happened, rather than to build quick sequential fixes later in the process. After a few years, the latter approach becomes completely unmanageable; furthermore, it quite often leads to permanent destruction of data on which new models could be built.

Therefore, the rule is to detect early, act early and not to let the issues accumulate.

Fixing problems at the source is more difficult when data are obtained from external sources. Fortunately, most credit bureaus have similar processes of data quality assurance and they can track problems to their data contributors where the problems originate. There are penalties for errors in data feeding credit bureaus.

In the case of other data providers (integrators of economic information etc.), it is good to sign a service level agreement motivating the provider to listen to feedback and ensure data quality on their side.

2.3.3 Safety by design

Safety by design can be based on the following rules which reduce the probability of error and facilitate resolution.

- Keep data pre-processing logical and transparent and as simple as possible.
- Avoid multiple inconsistent ETL layers in multiple data warehouses and data marts.
- Avoid artificial adjustments, data impacted by window dressing and classifications impacted by internal politics, even if it is called "one version of the truth" in the company.
- Avoid additional data sources which add instability to the data feed and do not improve the power of models.
- Avoid processing of data on multiple technology platforms. Even small things like differing date format or floating-point representation of data may have annoying consequences.
- Eliminate the human factor in the processing of data, including data entry and manual transfer of data files.

- Automate all data feeds and have them appropriately scheduled.
- Implement automated data quality scans whenever data processing moves from one department of the institution to another.

2.3.4 Using robust regression or machine learning technique that does not overly rely on single variables

A traditional credit scoring recipe is to select uncorrelated variables with reliable univariate predictive power which are most appealing to the credit risk expert and at the same time robust in production implementation. The correlation effects and following statistical insignificance of additional variables reduce the number of variables to 10 - 20. Given a vast number of available data and considerable cost of obtaining them (paying external data vendors, collecting customer consents, fulfilling GDPR information duty), it is not a serious proposition. It is even counterintuitive. A traditional credit risk expert would look at customers income from the application form, compare it to the credit turnover on a current account. Furthermore, he/she would attempt to contact the employer, social security office or tax office (depending on the jurisdiction) to confirm it. Therefore, common sense tells us to use multiple data sources to extract the same information, take into account all of them and even to examine differences in more detail. Here machine learning can help by building similar mechanisms into the model. It should be noted though that, depending on the algorithm, the focus will be either to give equal balance to the same information coming from other sources or to select a dominant source and to look at differences. This should be understood (if not controlled) by the modeler. In any case, with machine learning technique, building models with few dominant variables can and should be avoided.

2.4 Data pre-processing

Regardless of the estimation techniques used for credit scoring, the following basic principles of data preparation should be taken into account.

2.4.1 Keeping the data consistent

Each field of the input data set should have the same meaning, regardless of other fields in the data set. A simple example is account balance which can be expressed in various currencies. For the purpose of modeling a single currency should be used and values in foreign currency should be converted accordingly. The same applies to gross and net income, positions of financial statements reported under various standards, etc. In case data cannot be converted to a single and comparable standard, it is better to split input data into two or more separate fields according to a different meaning, leaving missing data whenever necessary.

Records where the data were severely compromised (e.g. due to a failure in the data feed) should be entirely removed from the modeling data set.

2.4.2 Coding missing values

The critical observation referring to missing value is that missing values should not be confused with zeroes, either means or medians of the population. They should be considered as a separate category caused by the conviction that the data is not relevant, that the entire data feed is missing or that the client withdrew them on purpose.

Depending on the modeling algorithm or any particular implementation thereof, the treatment of missing values will vary. Some algorithms will discard missing values completely; some will attempt some form of deterministic or random imputation. As authors are not in favour of using random imputation in credit scoring area, they recommend the following approach:

- For categorical data, create a separate category with coded value for "missing" (i.e. technically it will no longer have a missing value).

- For numerical data, impute a number that will have the smallest impact on the modeling technique used. In the case of decision trees or ensembles of voting decision trees that would be an unusually high or unusually low number, beyond any range of commonly occurring numbers. In the case of linear regression, it would be the mean value (calculated after the treatment of outliers described above). A dummy variable indicating a missing value may also be used.

The approach recommended for linear regression may also be used for decision trees and ensembles of voting decision trees. It should be noted that in case the entire data source is missing (e.g. no credit bureau information), there may be an additional dummy variable introduced indicating missing values coming from the above-mentioned approach. It is recommended to replace them with a single dummy variable indicating a missing data source, in order to control/reduce the complexity of the model. Whenever the client has the right to withdraw information negatively impacting his/her credit score, the relevant dummy variable should be artificially adjusted to make sure that the missing data correspond to the worst possible value of the variable in question. When adverse discrimination based on such a withdrawal is prohibited, we recommend that the use of the entire variable is carefully considered, even if it remains predictive for credit risk.

2.4.3 Pre-processing data in production

Whatever pre-processing is made for modeling, the same pre-processing should be implemented in production. Additional data used in processing (such as data from the national statistical office or currency exchange rates) may need a regular refreshment. Depending on the decision engine used pre-processing of many input data may result in challenges for model implementation which are discussed further in this publication.

2.5 Conclusions

The data management process, including the identification of adequate data sources, along with data-quality assurance and pre-processing, was presented in this chapter. With the presented good general practices in this area, we emphasize the uniqueness and specificity of the process for various practices (e.g., the scoring model purpose, size of the entity, or product/customer segment). The set of good practices presented in the monograph are related to 1) understanding the business context of the process and target population, 2) understanding the data, as credit scoring requires a delicate balance between the need to know the client and the privacy of the client, 3) understanding the data feed process, 4) removing issues and problems of the data set, and finally, 5) setting tight control on the quality of the outcome variable. We argue that modern credit scoring may heavily benefit from machine learning techniques, but this cannot be at the expense of a lack of understanding of data processes. Responsible use of machine learning methods, particularly in model auditability and robustness, requires a thorough understanding of the data acquisition process and the related business framework. The first step in building a responsible credit scoring process is to understand your data!

Chapter 3

Variable selection methods

Karol Przanowski
Sebastian Zając
Daniel Kaszyński
Łukasz Opiński

Institute of Statistics and Demography, Decision Analysis and Support Unit Collegium of Economic Analysis
SGH Warsaw School of Economics

The variable selection methods, in classical econometrics, are used to extract the most important, in terms of statistical significance, features. Usually it is performed by an iterative a) removing the least important feature (*backward stepwise selection*) or b) adding the most statistically significant one (*forward stepwise selection*). In principle, these methods should lead to an iterative improvement in the result, i.e., the overall model's performance or at least to maintain its level. The termination criterion of the algorithm (backward or forward) is tied up to the dynamics of the cumulative model performance – the algorithm should be stopped when the model's predictive power begins to decline which can be observed an increase in test errors.

Those methods are based on a single-factor analysis (inclusion or exclusion of particular variables) which leads to certain risks, such as the fact that steps which have been already taken, cannot be undone and they determine what happens next. For instance, in the stepwise method, if a variable was once excluded, it will not be included again. We investigate the dummy coding as an alternative to Logit and WoE transformation in Section 4.3.5 (**see also**; Scallan, 2013; Scallan, 2011); it should be emphasized that dummy coding, in the context of credit scoring, is a complex technical solution which may lead to an unstable selection if not done correctly, especially when Wald test's p-value is used. Thus, employing the dummy coding combined with the standard implementations of stepwise methods is not recommended.

Additionally, one of the most common problems of the single-factor analysis is the multicollinearity of dummy variables which jointly represent the same categorical variable. Also, their correlation is dependent upon the dummy coding type as well as the reference category, see Section 4.3.5.

To overcome some of the mentioned issues, in this chapter we discuss selected methods that can be utilized for the purposes of the vari able selections.

3.1 The importance of variable pre-selection

We start this chapter with an example that aims to present the underlying concepts – we recommend to study similar examples (also including the near multicollinearity problem – see Section 5.4). Let us assume our model is true according to the following equation:

$$\ln\left(\frac{p_i}{1-p_i}\right) = -4.8 + 2.3X_{i,1} + 1.8X_{i,2} + 0.2X_{i,3}$$

Assuming the independent standard normal distribution of the $X_{i,j}$, for all j, we have the $E[p_i] = 8.08\%$. Now, let us assume that we simulated n observations (i.e. the vectors $X_{i,\cdot}$ of i-th observation features, from standardized normal independent distribution). The model based on the above-mentioned formula is estimated. The distribution of the model parameters (500 simulations each) is presented in Figure 3.1.

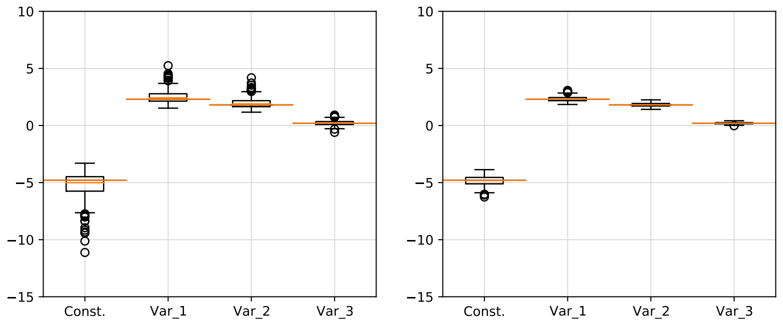


Figure 3.1: Variable selection – the impact of the number of observations. The left-hand side figure presents the estimation results for the $n = 200$ (small sample) and the right-hand side for the $n = 1000$ (large sample)
Source: own work

The left-hand side figure presents the estimation results for the $n = 200$ and the right-hand side for the $n = 1000$. Intuitively, the variability of the model's parameters is negatively related to the number of observations in the training set.

Now, let us assume that we do not exactly know the key features in this model (i.e. `Const.`, `Var_1`, `Var_2`, `Var_3`). We have a wide range of variables that include the three above-mentioned variables. In other words, we also consider variables which do not have any impact on the target variable (i.e. $\beta_j = 0$, for $j = 4, 5, \ldots, 103$ – we assume the 100 irrelevant variables). Figure 3.2 presents the correct specification of the model (estimation on the 3 key features) on the left-hand side and the misspecified model, i.e. all of the 103 variables with the constant terms, on the right-hand side.

As presented in Figure 3.2 ($n = 1000$ in both examples), the coefficients of the key features suffer from the inclusion of irrelevant variables – there is an impact on both bias (deviation from the true model coefficients) and efficiency (variability of the coefficients). This effect explicitly suggests the need for the model estimation on the filtered features (i.e., exclusion of irrelevant variables). In terms of classical credit scoring, one would perform the stepwise estimation to extract the most statistically significant

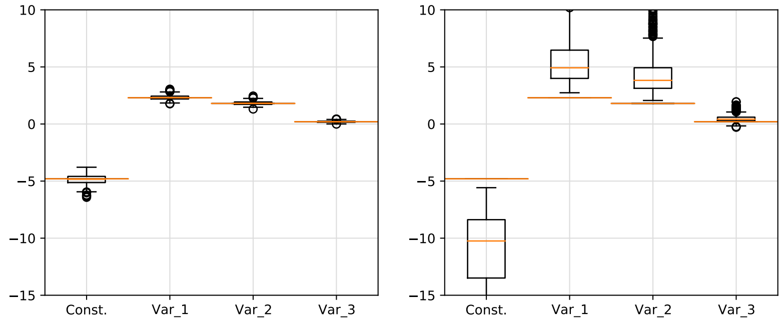


Figure 3.2: Variable selection – the impact of the irrelevant variables. Both of the plots present large sample examples (i.e., $n = 1000$); however, for the right-hand side example, additional 100 irrelevant variables has been included (apart from the constant and 3 relevant variables)
Source: own work

variables, as most of the variables may exhibit similar, uniform power (Gini of Information Value, as discussed later). Note that the coefficient of `Var_3` is equal to 0.2 which is close to zero. Because of this, in case of a lower number of observations, this variable may be stochastically excluded from the final set of features and another, more statistically significant variable will appear in one of the algorithm iterations even though there is no real relationship between this variable and the target variable.

Moreover, the multicollinearity problem (see Section 5.4) will further increase the magnitude of statistical inference, i.e., whether the particular feature should be included in the model or not. In the next example, we assume that all of the features are correlated, i.e., the relevant and irrelevant variables exhibit a high correlation – 80% of the correlation between all of the features.

Given the high correlation between features (right-hand sided plot in Figure 3.3), the variability of the parameters increased significantly (compared with the right-hand sided Figure 3.2). It should be noted that some of the methods discussed in this chapter are less resistant to the multicollinearity and a large number of relevant variables (e.g., stepwise methods, as they

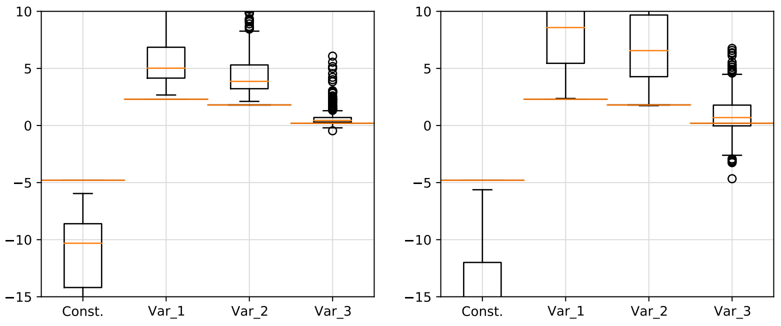


Figure 3.3: Variable selection – the impact of the multicollinearity on the model's parameters. Both of the plots present large sample examples (i.e., $n = 1000$), including additional 100 irrelevant variables has been included. However, in the right-hand sided plot additionally all of the variables are heavily correlated (80%)
Source: own work

are based on the p-values of particular variables and depend on the exact order of features inclusion/exclusion).

In Chapter 4, we discuss regularization techniques – e.g. ridge or lasso regressions. As discussed further in the monograph, those methods are dedicated to the problem of multicollinearity or small number of variables. In Figure 3.4, we present the usage of lasso regression to extract the key variables.

As presented in Figure 3.4, the parameter estimates are biased – for the left-hand side plot, the penalty imposed on the loss function is too small and for the right-hand side plot, the penalty is too severe. As discussed in Chapter 4, the regularization imposes the bias, but as a result, the model coefficients have smaller variance (i.e. bias-variance trade-off (James et al., 2013)).

The last example presents the model that almost perfectly replicates the underlying process – the lasso estimation was performed after internally selected (i.e. grid search) penalty term, see Section 4.2.2.

It should be noted that the regularization techniques can be utilized for the purposes of the variable selection, as it was presented in the above-

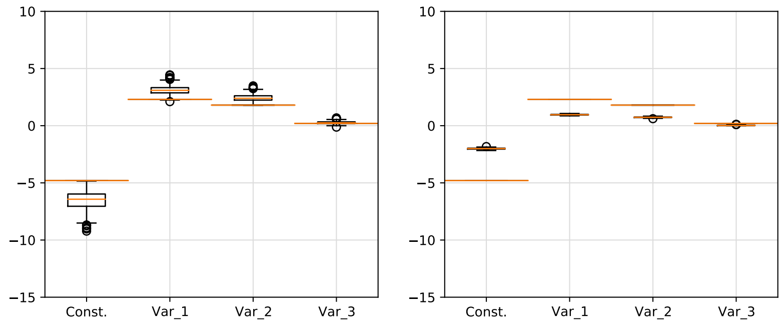


Figure 3.4: Variable selection – the impact of the regularization techniques (not calibrated penalty term). Both of the plots present large sample examples (i.e., $n = 1000$), including additional 100 irrelevant variables that are also heavily correlated
Source: own work

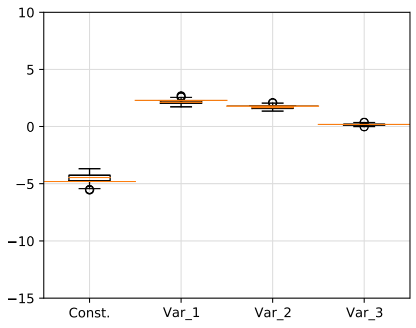


Figure 3.5: Variable selection – the impact of the regularization techniques (well-calibrated penalty term)
Source: own work

mentioned examples (for more on the regularization techniques, see Chapter 4). However, in practice one cannot assume the complete features set – the estimation is the process that should reveal the final set of variables (and weights) which should be included into the model. On the other hand, the penalty imposed on the loss function may be grid searched based on the validation set, see Section 4.3.2.

However, the regularization techniques shown in the examples, as in the case of stepwise, should be handled carefully while using the dummy variables (penalties imposed on the groups of the related dummy variable).

In this chapter, we discuss well-known feature selection methods in the case when models are generated as a classical logistic regression with logit transformation and categorization of all variables (binning). Particular methods discussed further in the text were tested based on the simulated data, as presented in (Przanowski, 2011). Therefore, we emphasize the need for performing an individual analysis – the generic data used in this monograph are not representative and independent efforts have to be made to evaluate impact in a particular case.

3.2 Comparison measures for the variable selection methods

One of the main problems of data analytics is still extraction of the key features – the abundance of variables only hinders this process, as the variety of possible feature selection grows. The computation of all possible models (i.e., with all of the possible numbers and combinations of features) for large data sets is unpractical or even impossible. More on the computational complexity is presented in Section 4.3.2. Given the current state of progress of computer science and technology development, we need to rely on the algorithms for variables selection – the brute force approach is no longer feasible.

In the case of a large number of variables (in relation to the number of cases), stepwise algorithms that select variables very often reach the local minimum which makes finding the right solution difficult.

For the purposes of presenting the numerical examples, we have an used artificially generated target variable generated baing on the input data after the WoE transformation (see Section 4.1.2). In the course of further analysis, we will monitor the following list of commonly used criteria:

- *Gini scaled* as a measure of the predictive power; for the technical purposes we utilized the scaled measure on every set of models: predictive power is linearly scaled up to 80% Gini, as each of the data sets (i.e., the `ABT_APP` and `ABT_BEH`) have a different underlying process (i.e., error included in the model); this measure allows to compare the distribution of Gini statistics for each method and each data set,

- *delta Gini*, the relative difference of Gini values between the training set and test set; this measure allows evaluating the stability of the model and potential overfitting,
- *NVar*, the number of variables; the smaller the number of variables, the more preferable model in terms of classical scoring,
- *Nbeta*, the number of negative betas in the logistic regression model (more on negative betas in Section 4.3.4); this measure allows to capture the confounding of variables; in the case of negative betas, we get different responses for a single variable model and the same variable in a model consisting of more variables,
- *p-value*, the classical Wald test of significance; as indicated earlier in this monograph, this statistic combined with the dummy variable coding is questionable, but we present its results as in classical approach it is still most commonly used measure,
- *VIF* – Variance Inflation Factor is the multicollinearity measure; multicollinearity problem has an impact on the variance of model's coefficients and is usually used in the model's diagnostic; for further discussion on the multicollinearity problem, see Section 5.4.

There is no best method today to prove the correctness of the argument that a given technique is the superior one. This problem is directly associated with having universal and comprehensive data, which is obviously impossible, so one can only generate plenty of random data sets (or gather publicly available data sets) and carry out a comparative process. Such a process should be carried out by an analyst in a particular data set so that one will be able to make final decisions about choosing a better technique for specific problems. This study cannot be regarded as evidence of the superiority of one technique over another but should rather be perceived as an example of a comparative method on a given data set.

In credit scoring practice, the *Analytical Base Table* (ABT) is utilized in data handling. ABT is a table that contains all the variables (and its interactions, dummy variable coding and binnings) for a single observation (e.g., customer on reporting date). The ABT table usually has thousands of columns that contain all the information for a particular observation – including the interactions between features, usually up to the 3rd level (see also Section 4.1.2 or discussion on the AdaBoost in Section 4.2.6).

3.3 Variable selection methods

In this section, we discuss the most commonly used methods of variable selection. The discussed methods are presented in the following order:

1. Principal Component Analysis and autoencoders.

2. Sorting and selection of the best set of features due to Gini statistics and Information Value for each variable treated independently.

3. stepwise method for iterative selection of variable in the backward and forward versions.

4. Other models used in machine learning to indicate the significance of variables using feature importance mechanisms as of decision trees or random forests. In those methods, all of the variables are analyzed simultaneously.

5. Analytical branch and bound method (Furnival & Wilson, 2000), implemented in the SAS procedure – `PROC LOGISTIC`. Due to the number of variables, SAS procedures first limited the number of variables to about 70 by stepwise selection or best 70 by Gini and than `B&B` is run for combinations of 4 to 14 variables (the number of possible variables is set by expert knowledge, in general more than 14 variables generate too big VIF), calculating best 20 models per every number of variables.

3.3.1 Principal component analysis and autoencoders

One of the most popular methods for a dimensional reduction is the *Principal Component Analysis* method (PCA). It is based on the linear transformation of all variables generated by the eigenvectors of the variance-covariance normalized matrix of features. Therefore, this method generates new variables that are linear combinations of the actual features. While the stepwise method supported with the Wald statistics does not allow to exclude correlated variables, PCA allows for decorrelation of features and then selecting a few main principal components (i.e., eigenvectors) that exhibit the highest predictive power against the target variable. Having selected those components, we can apply reverse engineering and extract the features that had the most predictive power and are decorrelated.

In scoring models, one needs to preserve the original variables due to their conversion into scoring cards and interpretability. Therefore, these methods, although useful and practical in the reduction of dimensionality, are not suitable for use in the construction of classical scoring models.

It needs to be emphasized that, in principle, one may use the kernel principal component analysis (kernel PCA) which is the extension of the basic PCA. This extension is related to the usage of kernel methods (Hofmann et al., 2008). The kernel PCA is especially useful when the basic PCA cannot reduce dimensionality, due to non-linear correlations between features. In principle, kernel PCA is recommended in case of the poor performance of basic PCA.

Analogously, one can consider using the neural networks to reduce the dimensionality of features. The models would take as an input the entire range of variables and would result in the most critical features, i.e. autoencoders (for more on the neural networks, see Section 4.2.3). However, these networks generate non-linear transformations of variables which significantly hampers the interpretation of the resulting variables.

3.3.2 Variable selection by best Gini statistics

The popular and straightforward method of selection is based on the best features with higher Gini statistics. The Gini statistic is calculated for each variable separately (more on the Gini calculation in Chapter 6). The method assumes the single-factor analysis and the selection of variables that exhibit the Gini at some predefined level. This way, one can obtain information about the quality and impact of a given variable on the target variable.

This method is very often used in the initial preselection of variables, where it can be used to remove variables whose impact is minimal compared to the rest of the features. The selection of the n best variables having the most substantial effect on a target variable often leads to a selection of linearly dependent variables. As a result, the models obtained by this procedure (compared to other methods) have relatively large statistic Gini of the entire model but they are burdened with multicollinearity.

As it is presented further in the text, this method is useful and efficient for scanning through features and rejecting insignificant variables, in the case of a large number of variables, at the initial phase of data processing (i.e. preprocessing).

The `VARCLUS` procedure is based on ABT. Firstly, the clusters of features are prepared, based on `VARCLUS` procedure (the number of the clusters is the input parameter). Based on that, one develops the table with the $1 - R^2$ and Gini of each particular variable. Among the variables representative for the cluster (based on the $1 - R^2$), we choose only one with the highest Gini, disqualifying variables that are not logical enough or are difficult to interpret. One chooses variables that are weakly related to the cluster and those with high Gini (allowing to obtain the shortlist of features as an input to the stepwise method). It is worth noting that this is not an automatic method and it requires a lot of work and internal discussions (especially when it comes to logic and variable interpretability) (Nelson, 2001).

3.3.3 Information Value

The information value IV (*information value*) can be a beneficial concept when choosing variables for the model. The IV indicator is based on the idea of information introduced by C. Shannon, closely related to entropy and information theory. The IV is defined in the following form:

$$\mathrm{IV} = \sum_i (\mathrm{distr}\,0_i - \mathrm{distr}\,1_i) \times \ln\left(\frac{\mathrm{distr}\,0_i}{\mathrm{distr}\,1_i}\right)$$

where:

- $\mathrm{distr}0_i$ is the distribution of values 0 (for target variable) in the i-th class for the variable,
- $\mathrm{distr}1_i$ is the distribution of values 1 (for target variable) in the i-th class for the variable,
- ln is a natural logarithm.

The value of $\ln\left(\frac{\mathrm{distr}\,0}{\mathrm{distr}\,1}\right)$ is defined as the WoE factor.

$$\mathrm{IV} = \sum_i (\mathrm{distr}\,0_i - \mathrm{distr}\,1_i) \times \mathrm{WoE}_i$$

Due to the sensitivity of logarithm and division on zero values, a well-done binning is critical for the correctness of the mathematical operation of this method.

3.3.4 Stepwise methods of variable selection

As discussed earlier, the stepwise methods consist of a) *backward stepwise selection* (starting with the entire set of features, reduction of the least important important features in subsequent steps) and b) *forward stepwise selection* (starting with no features, adding the most statistically significant in subsequent steps). In principle, those methods should lead to an iterative improvement in the overall model's performance or at least to a maintenance of its level. The algorithm should be stopped when the model's predictive power begins to decline (test error increases).

Those methods are based on a single-factor analysis (inclusion or exclusion of particular variables) which leads to certain risks, such as the fact that steps which were already taken, cannot be undone and they determine what happens next. For instance, in the stepwise method, if a variable was once excluded, it will not be included again. The research presents the dummy coding as an alternative to Logit and WoE transformation, see Section 4.3.5, or (Scallan, 2013; Scallan, 2011). It should be emphasized that dummy coding, in the context of credit scoring, is a complex technical solution which leads to unstable selection methods, especially when Wald test's p-value is used (in case of near multicollinearity problem). Thus, the dummy coding combined with the stepwise methods is not recommended.

The stepwise methods, as emphasized further in the text, have to take into consideration some technical issues, e.g. variable coding – dummy variables should be included/excluded as a complete set of variables corresponding to the same categorical variable. This case requires special handling, as the p-values are dependent upon previous steps.

3.3.5 Decision trees and random forests

Previous variable selection methods were based on the association of individual variables with the distribution of a target variable. It turns out that modeling methods using decision trees including random forests, in addition to generating the model, can also be used to create a list of variables ordered from the most important and affecting the target variable to ones that are irrelevant in the model.

Modeling with decision trees and random forests usually reflects a more complex nature of data and, at the same time, improves model predictions compared to the model obtained by logistic regression. However, due to the construction of the scoring card and interpretability (especially in the banking sector), these models are not used directly.

Variable selection in these methods can be very easily obtained in the *feature importance* procedures used in the `sklearn` library in the `Python` environment (Pedregosa et al., 2011). For further discussion on decision trees and random forests, see Chapter 4.

The random forest or gradient boosting models for all variables take into account (through the construction of many tree models) the non-linear correlation between variables. Gradient boosting is particularly efficient in detecting non-linear relations between variables. Unlike the random forests (that work better with deeper decision trees), the gradient boosting works best with the decision trees no deeper than 2 or 3, which is similar to the utilization of ABT features. One may only utilize the first few decision trees with the highest weights and those will suggest construction of variables in ABT, significantly reducing work effort on defining ABT variables by trial and error. Moreover, the gradient boosting with the logistic regression objective function is simply the linear model based on the ABT features.

3.3.6 Branch and bound

An analytical solution to the problem of choosing the optimal set of variables is the *Branch and Bound* algorithm (`B&B`) (Furnival & Wilson, 2000; Narendra & Fukunaga, 1977). This algorithm allows us to search through the set of variables without searching for all possible solutions. Due to the analytical nature of the algorithm, it is a method of implementation that takes quite a long time if there are many variables included.

The `B&B` method has been implemented in SAS software, the `PROC LOGISTIC` (Furnival & Wilson, 2000; SAS, 2013). Virtually all results (even a small set of variables – 4 to 14 variables) presented in Section 3.4 indicated the superiority of the `B&B` method (several percentage points in Gini) over other methods. Please note that within this monograph, we present results of two distinct methods: Branch and bound which is based on the Gini value (i.e., `B&B Gini`) and Branch and bound which support the further use of the stepwise (i.e., `B&B STEP`).

3.4 Numerical experiment of variable selection

In this section we present the results of a numerical experiment. Based on the simulated framework of the credit scoring data generated as in (Przanowski, 2011), we run particular variable selection models and evaluate individual measures as discussed in Section 3.2. The methodology of model building is compliant with Basel II documents such as (Basel Committee on Banking Supervision, Bank For International Settlements, 2005a) or (Basel Committee on Banking Supervision, Bank For International Settlements, 2005c). Further explanations are also presented in Chapter 4 (definition of the defaulted client).

We have utilized the two data sets for this purpose: a) `ABT_BEH` – ABT of the behavioral data and b) `ABT_APP` – ABT of the application data. Both of the data sets exhibit highly non-linear relations between features and the target variable. The taget variable was generated using logistic regression using the WoE transformation on features – similar to the examples shown in Section 3.1. As the simulated data sets, i.e., `ABT_APP` and `ABT_BEH`, have a different underlying process (i.e., error included in the model), we have used the linearly scaled up to 80% *Gini* as a measure of the predictive power. This allows us to compare the distribution of Gini statistics for each of the method and each of the data sets. From the implementation point of view, we have utilized the SAS framework as well as the codes prepared as presented in the (Przanowski, 2011). For the machine learning methods, we have utilized the `Python`-based libraries: `sklearn` and `Keras`.

In the graph below, we have used the following naming convention:

- `Random` – we choose and compute four logistic regression models for each fully randomly chosen features with 5-25 variables.
- `Gini` – selection based on Gini statistics.
- `IV` – selection based on Information Values.
- `TREE` – based on the feature importance of the calibrated decision trees.
- `FOREST` – based on the feature importance of the calibrated *random forest*s.
- `RFE` – based on stepwise methods.
- `LASSO` – with the parameter $C \in \{0.001, 0.01, 0.1, 1, 100, 10000\}$.
- `B&B Gini` – Branch and Bound method based on Gini values.
- `B&B STEP` – Branch and Bound method after stepwise selection.

3.4.1 Classical approaches to features preselection

In Figures 3.6–3.10 we present the numerical results of the particular feature selection methods.

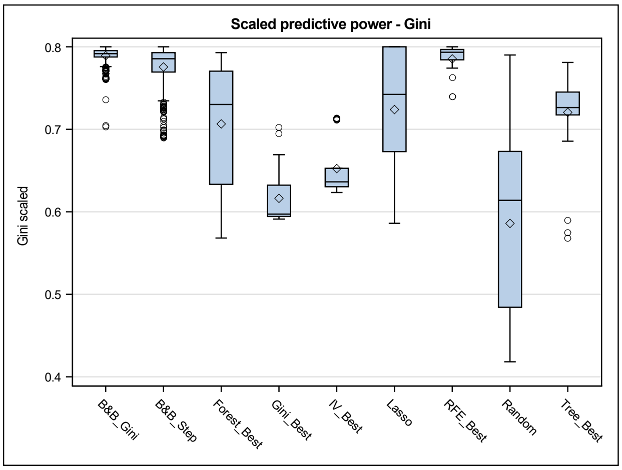


Figure 3.6: Single criterion – Gini – the predictive power
Source: own work

As presented in Figure 3.6, the best methods based on the predictive power are the ones based on the `B&B` algorithm; moreover, the lasso and random forests allow for a high predictive power features selection.

In terms of the number of variables, the `B&B` method allows for the smaller models – up to 14 variables, see Figure 3.7. This result combined with the previous, related to predictive power, suggests the superiority of this model – the high predictive power along with a simple functional form (i.e. small number of variables).

The previous results are also supported by the next values – in terms of multicollinearity problem, the `B&B` method allows to reduce the VIF value (performed after the stepwise method), see Figure 3.8. VIF for lasso method is not calculated due to infinite values.

In terms of the overfitting problem, also the `B&B` method exhibits relatively small changes – delta Gini between training and validation sets, see

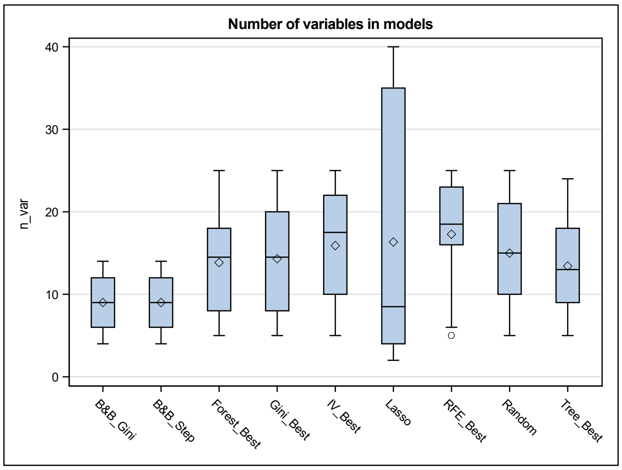


Figure 3.7: Single criterion – NVar – the number of variables
Source: own work

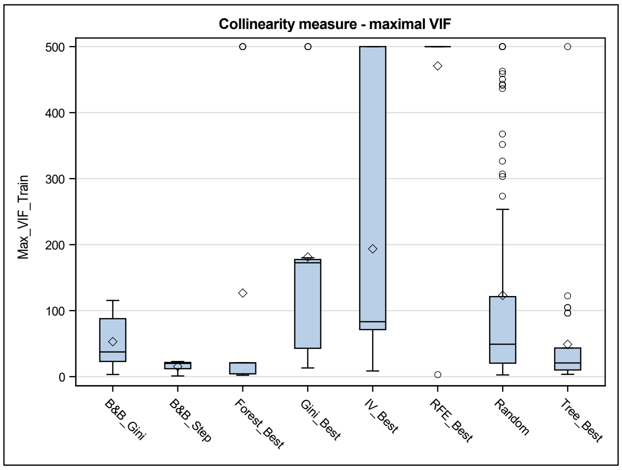


Figure 3.8: Single criterion – VIF – maximal Variance Infaction Factor
Source: own work

Figure 3.9. Similar superiority is achieved in terms of the Negative Betas, see 3.10.

The analysis presented in graphs 3.6-3.10, showed that `B&B` and lasso have many advantages. In general, apart from `B&B` and lasso, all other techniques have negative betas and statistically non-significant variables.

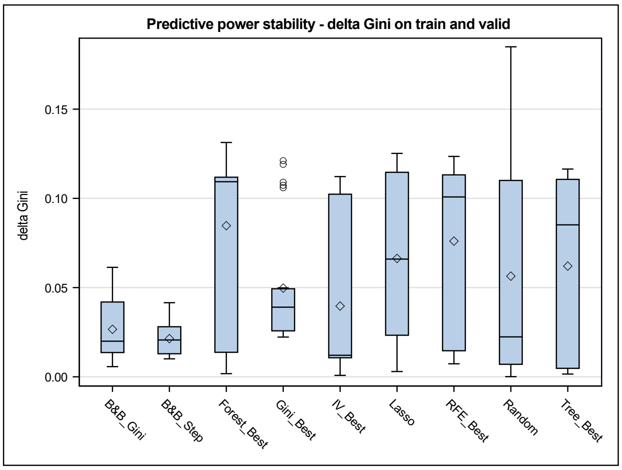


Figure 3.9: Single criterion – delta Gini
Source: own work

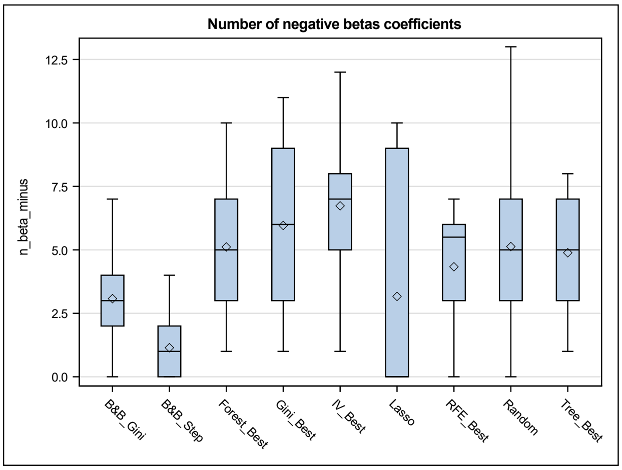


Figure 3.10: Single criterion – NBeta – the number of negative betas
Source: own work

It is a widespread problem of ABT data where many variables are similar to one another because they represent the same information collected in the history, e.g. the number of delinquency events during the last 9 or 12 months can be the same (and hence one may consider using the `VARCLUS`, see 3.3.2). That property usually produces the equal value of Gini and IV per few similar variables which results in poor estimates in logistic regression method (see also Section 5.4).

3.4.2 Feature selection with machine learning

Besides, we have also utilized the other methods discussed in Chapter 4 (e.g., XGBoost) as feature selection models. Based on all data sets, i.e. `ABT_APP` and `ABT_BEH`, we have used machine learning methods to generate models on pure data (i.e. no binning, no WoE transformation, the categorical data transformed by dummy coding). In contrast to classical logistic regression procedures and scorecard building, we did not make binning of data and we did not run any initial feature selection method.

That idea is opposite to a classical approach where WoE transformation was the main technique and Wald p-value and VIF collinearity measures were used. Here WoE method is replaced by missing values imputation with median values obtained for each feature separately. We also added new binary features in case the missing values have appeared.
Based on the previous criteria, we have compared the following models:

- `Lasso` models with $\alpha \in \{0.001, 0.01, 1, 100, 10000\}$.
- `Ridge` classifier with $\alpha \in \{0.001, 0.01, 0.1, 0.2\}$.
- `Decision tree` with $max\_depth \in \{10, 30, 50, 100, 150\}$.
- `Random forest` with $max\_depth \in \{10, 30, 50, 100\}$.
- `Stochastic gradient boosting` classifier with $\alpha \in \{0.001, 0.01, 0.1\}$.
- `XGBoosting` classifier with $max\_depth \in \{1, 2, 3\}$ and learning rate $\in \{0.1, 0.2, 0.3, 0.4, 0.01, 0.02, 0.05\}$ and n_estimators $\in \{10, 30, 50, 70\}$ (in total 84 different models).
- `Relu Network` with one, two and three hidden relu layers, see Section 4.2.3.
- `Logistic regression` – a classical approach presented here only for comparison purpose.

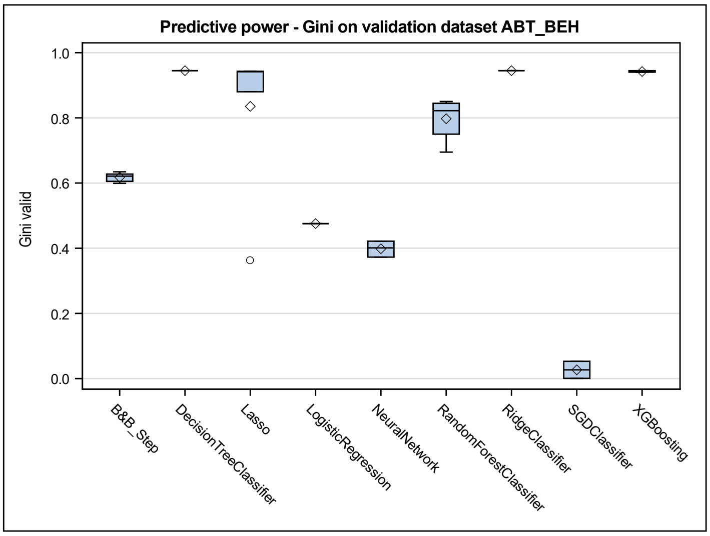


Figure 3.11: Gini on validation sample on `ABT_BEH`
Source: own work

- `B&B Step` – methods of scorecard building described in the previous section with conditions: only positive betas and p-value < 0.05, so only accepted by typical benchmarks – presented here also for comparison.

All techniques listed above were implemented using `sklearn` library, except for neural networks which were created using `Keras` library.

The first comparison is made on `ABT_BEH` data set, where Gini on validation sample and delta Gini – only two criteria are presented, see Figure 3.11 and 3.12. The classical approach represented by the `B&B Step` method can be compared with `DecisionTreeClassifier`, `Lasso`, `RidgeClassifier` and `XGBoosting`, where they have almost 90% of Gini. It means that the machine learning methods can be used instead of classical scorecards as we are able to achieve the same or even better predictive power and stability on modeling samples (training and validating data sets). Delta Gini for the `DecisionTree Classifier`, `Lasso`, `RidgeClassifier` and `XGBoosting` is small, so models are stable.

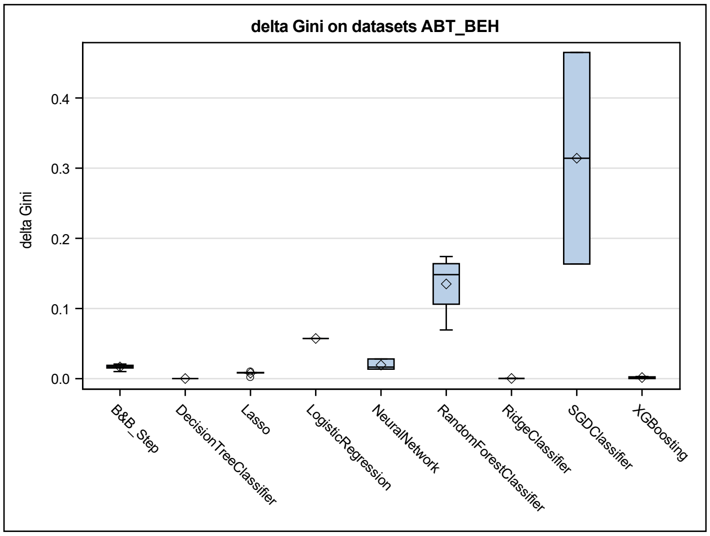


Figure 3.12: Delta Gini on `ABT_BEH`
Source: own work

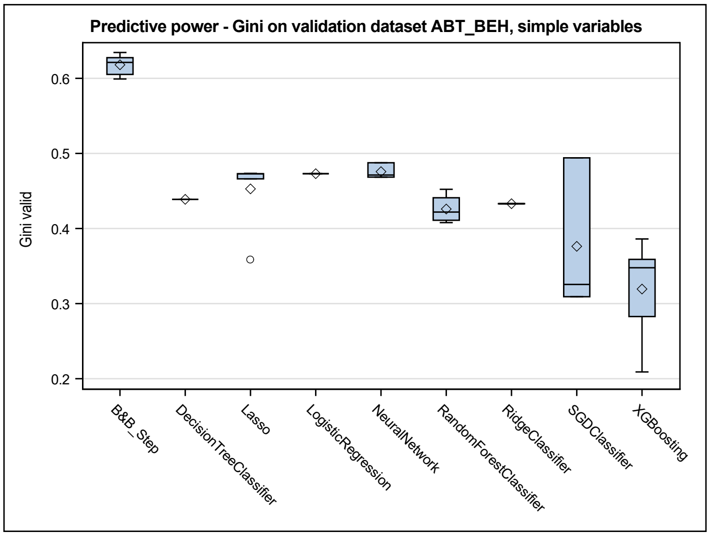


Figure 3.13: Gini on `ABT_BEH` only basing on simple variables
Source: own work

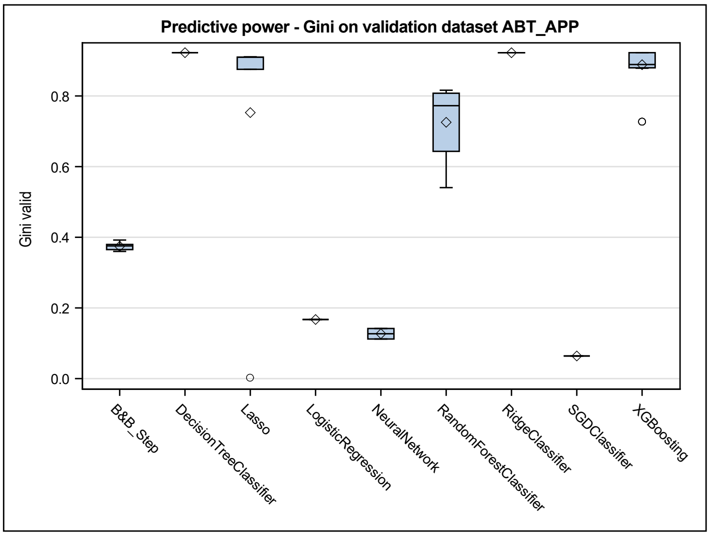


Figure 3.14: Gini on validation sample on `ABT_APP`
Source: own work

Moreover, now we can test a hypothesis that machine learning techniques do not need very complex variables, they do not need any aggregated information in time, only simple variables representing statuses and simple summary variables per particular historical month. That idea is tested in Figure 3.13, where unfortunately `DecisionTreeClassifier`, `Lasso`, `RidgeClassifier` and `XGBoosting` have significantly worse values of Gini than `B&B`. It should be emphasized that the `B&B` method is calculated on all available variables because this is the main idea of scorecard building. That problem needs to be analyzed in further research. The same conclusions can be formulated based on `ABT_APP` data, see Figures 3.14-3.16.

In summary, Machine Learning or Artificial Intelligence techniques can be used for the development of the scoring models in a similar way as traditional approaches with WoE logistic regression used; however, they require additional attention in terms of interpretability (see Chapter 7).

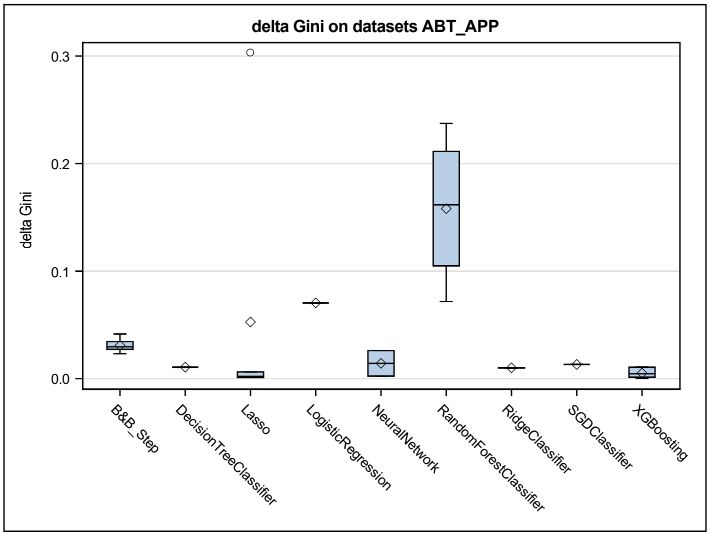


Figure 3.15: Delta Gini on ABT_APP
Source: own work

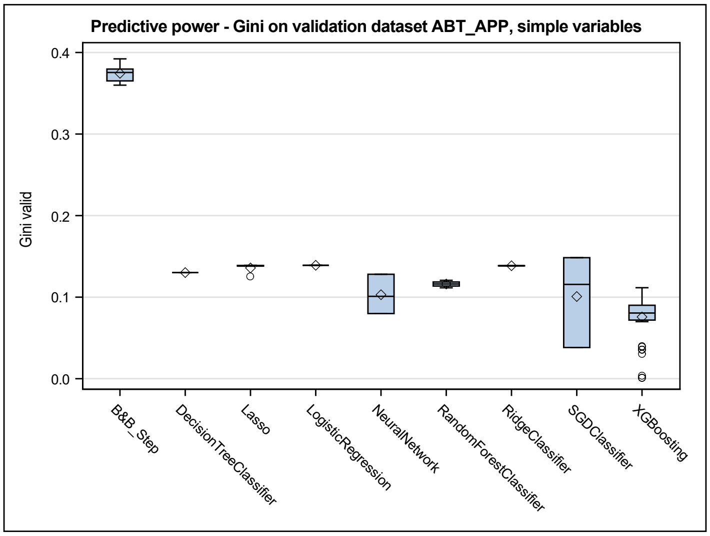


Figure 3.16: Gini on ABT_APP only basing on simple variables
Source: own work

3.5 Conclusions

This chapter presents selected methods of variable selection for scoring models and their impact on the quality of obtained models. These methods are based on statistical properties of individual variables (e.g. stepwise approach) or global properties of all analyzed variables (e.g. Lasso regularization). Both classical and machine learning-based approaches have been presented.

We emphasize the need for performing individual analysis – the generic/simulated data used in this monograph might not be representative and independent efforts have to be made to evaluate impact in a particular case. The optimal feature selection method may be strongly dependent on a particular case and the final decision in that matter should be made upon the given data set.

In general, when WoE, Wald *p*-value and VIF are used as a criterion for variable selection, then `B&B` with the stepwise method is very useful and produces the best results. However, our recommendation regarding the feature selection (i.e. the best strategy in that matter) is implementing various selection of methods and performing further performance analysis, based on the particular case or data set. Even though the results of Section 3.4 presents the `B&B` as a superior method, we claim that the exact choice of the technique used in practice should depend on the particular data set itself. Given the simple implementation of a specific method in free libraries, all the discussed methods can be quickly used and evaluated at various stages of modeling.

Moreover, we always recommend taking into account the model purpose. In the context of predictive scoring models, the main criteria should be Gini and delta Gini, as those measures reflect the predictive power. On the other hand, when the descriptive model is needed, one has to support the interpretability of the generated features actively – e.g. `B&B`, stepwise method, decision trees or `VARCLUS` method presented earlier.

In terms of other practical suggestions, we propose a set of recommendations regarding feature selection methods:

- Using other tests than Wald test to select variables, e.g., Lagrange Multiplier or Likelihood Ratio. As discussed earlier, we recommend testing and selecting the groups of dummy variables related to a single category variable jointly.

- Using the WoE transformation instead of using dummy variables for stepwise selection of variables (issues related to Wald test, as discussed earlier).

- Utilization of *marginal measures* for the forward feature selection methods, i.e., Marginal Information Value, Marginal Gini or Marginal KS (Scallan, 2013).

- Utilization of the *Generalised Variance Inflation Factor* (J. Fox & Monette, 1992), instead of using VIF. This measure is a generalized version of the VIF and is used for testing groups of features and generalized linear models.

Chapter 4

Selected machine learning methods used for credit scoring

Małgorzata Wrzosek
Daniel Kaszyński
Karol Przanowski
Sebastian Zając

Decision Analysis and Support Unit, Institute of Statistics and Demography
Collegium of Economic Analysis
SGH Warsaw School of Economics

In this chapter, we present *machine learning* models that, in the light of the current subject literature, indicate the possibility and adequacy of use in credit scoring. It should be emphasized that the subject-matter machine learning literature is currently: a) broad and rich in various methods and their variants, b) inconclusive in terms of recommendations which methods should and should not be used. As of the benchmark for machine learning, we also present the classical approaches in this chapter, i.e., logistic regression.

In this monograph, we cover the binary outcome type models, which result in consideration of random variable having two possible and opposite outcomes: *default* or *non-default* of a particular client. Let Y denote that random variable and let us assume that $Y = 1$ or $Y = 0$, with the value 1 as the reference to the *default* event. In terms of empirical observations, we assume that the default event has or has not been observed and the observation corresponds to the underlying customer status.

For credit scoring, the defaulted client may be defined differently than in the European Banking Authority (EBA) guidelines (European Banking Authority, 2017b), so that one may:

1. Obtain more default observations for model estimation.

2. Include longer than 12 months of the observation period for the target variable (over-crediting does not translate into the default within 12 months).

3. Utilize the contagion effect – the default flag is generated on the customer level, even though it is calculated on the exposure level.

In the practice of credit scoring, the particular definition of a bad client should be carefully investigated by the bank internally, e.g., taking into account the specificity of the clients and products. From a technical point of view, even shorter DPD definitions may be considered (i.e. compared to the 90 DPD imposed by the regulatory frameworks) as a bad client (e.g. 60 DPD as an indication of the *Unlikely full repayment*). The principle rule that should govern the practice of trustworthy and robust credit scoring is to reflect the trustworthiness of the clients appropriately (in terms of predictive power, see Chapter 3. The exact definition and further discussion on the default event are out of the scope of this monograph.

The EBA guideline (European Banking Authority, 2017b), regulates the definition of a default event significantly. Currently the notion of default plays a vital role, due to most of the banks transforming their previous default definition into a new one, i.e. *New definition of Default.* The EBA established tighter standards concerning the definition of default (CRR Article 178) which in principle is aimed at achieving greater alignment and standardization across banks and jurisdictions. Those new standards need to be implemented by the end of 2020. The main difference in the new definition is based on *Days Past Due* (DPD) and two thresholds: relative and absolute. To accomplish greater standardization across the banking sector, EBA also provided a list of the default triggers (i.e. *Unlikeliness to Pay – UTP*).

The models presented in this chapter base on the default definition that is calculated on a customer level (contrary to the exposure level). Each customer may have a few credit accounts, so in the default definition we consider all of the customer's accounts existing at the moment of observation. The following customer statuses are defined accordingly:

- **good** – a customer who has maximally only one due installment on his all credits in the next T months,
- **bad** – a customer who has at least one credit account with more than three due installments in next T months (in case of T=3, more than 2 installments),
- **Ind** – an indeterminate case the rest of customers, in a practical way a customer with more than one due installments but less than 3 (in case of T=3, less than 2 installments).

where T denotes the tenor of observation, i.e. the time horizon from the observation moment. The above-mentioned statuses are coded as follows: **good**=0, **bad**=1 and **Ind**=.i. For the scoring model estimation, we assume that all of the **Ind** cases are excluded for the training set; for the purposes of *Probability of Default* (PD) calibration, **Ind** cases are treated as the **good** clients. As a target variable (i.e. default flag) we prepared the variable *default_cus12* which represents the T=12 tenor. By the definition assumed in this Methodology, a bad client is a customer who has at least one credit account with more than three due installments in the next 12 months. The share of bad customers based on that default definition is defined as *Default Rate* (DR).

For further clarification, we assume the following notation:

- **features** – are the inputs of the models that are used for the classification of good/bad client groups. In this Methodology, we assume that features are represented mathematically in the matrix with a number of rows equalling the number of observations (clients) and columns to the number of variables. In terms of notation, we will also assume that the features matrix may also contain binary/dummy variables, as may be represented as an X matrix. As for the synonyms relating to features, we also use the terms: *exogenous* or *explanatory* variables.

- **target** – is the value that in the interest of modeling process – in case of credit scoring, we will relate that variable to the creditworthiness assessment, usually denoted as a Y vector and used interchangeably with *dependent* variable or *endogenous* variable.

In terms of further notation, we followed the original articles and studies, providing the necessary definitions of variables.

4.1 Classical credit scoring models

The main usage of statistical methods in credit risk management over many last years was focused on the binary logistic regression model. A default event was always transformed into a binary predicted variable, even though in many references, middle statuses such as indeterminate or dormant were introduced some.

The credit scores, in a similar form as today, were invented in the 1950s by a team of mathematicians led by Bill Fair and accompanied by Earl Isaac – the *Fair, Isaac and Company*, today's FICO, was established to prepare the standardized credit scoring system (Finance, 2013). The first versions of the system were based on a manual, paper-based form (hence today we call it *the scorecard*). Scorecard points were mainly estimated by the groups of experts (Thomas et al., 2017) due to the limited development of computer science at that time – the scoring process could not be fully automated.

Along with the computers becoming the best tools for data analysis and after discovering the logistic regression method, the practitioners started to use the scorecards estimated on the logistic regression method. The analyst could decide to choose a classical logistic model or risk scorecard depending on his or her preferences and bank policy.

4.1.1 Credit Risk Scorecard

The credit acceptance process in banks performs one of the critical functions in portfolio management (Oesterreichische Nationalbank, 2006), especially for retail portfolios with an extensive number of applications. In the case of a large number of requests per month, i.e. tens of thousands, statistical tools are needed or even necessary in this process. The more credit applications the bank receives, the more vital the automatic decisions making tools and advanced statistical models are. The most typical and utilized model used in that credit scoring process are scorecards; see an example of a scorecard in Table 4.1 below.

Variable (characteristic)	Category (condition)	Partial score
Age	$\text{Age} \leq 20$	10
	$20 < \text{Age} \leq 35$	20
	$35 < \text{Age}$	40
Income	$\text{Income} \leq 1500$	10
	$1500 < \text{Income} \leq 3500$	26
	$3500 < \text{Income}$	49

Table 4.1: Example of a scorecard with two features: Age and Income
Source: own work

The study of the scorecard structure allows us to understand the factors that drive the creditworthiness assessment. Without specialized knowledge or statistical skills, one cannot identify the most critical variable in the presented scorecard. It is possible due to the simple property of *partial score* calculation – taking the riskiest category of each variable, we get the lowest possible value of the scoring. The variable with the biggest partial score variability is the most important one because changing it can impact the score the most. Analysis of partial scores also can be useful to identify the best and the worst customer in the process. In our example, the best customer (i.e. the least risky), is one with age > 35 and income > 3500. That customer gets 89 points. Opposite customer (i.e. the riskiest) is one with age ≤ 20 and income ≤ 1500 having only 20 scoring points, the minimal value.

Moreover, it is worth emphasizing that the scorecards, in principle, are not identifiable; for explanatory purposes, let us assume the scorecard presented in Table 4.1. After we add any value x, e.g. $x = 1$, to all of the partial scores for the variable Age (e.g. 11, 21, 41) and subtract the same value x from the Income variable (e.g. 9, 26, 49), each customer will end up with the same score points as before. Therefore, only score changes between levels for a variable can be compared across variables.

On the other hand, this construction of the scorecard allows us to determine the creditworthiness of a customer quickly – and as a result, compare two distinct customers in that measure. In first scoring models, the assignment of particular partial scores to specific categories, was performed based on the expert judgment and assessment. However, this caused issues concerning objectivity, auditability of the decision or even unbiasedness (the national prejudices historically had a significant impact on the credit rating (Cohen, 2012)). To perform more data-driven scoring, the new analytical models were proposed.

4.1.2 Logistic regression

The classical linear regression models are known for more than a hundred years. The first authors, such as Adrien-Marie Legendre and Carl Friedrich Gauss, created their works in XVII century (Bretscher, 1997), in the form we know as theory of the analytical *Ordinary Least Square Errors* (OLS). It was a significant step in the further development of statistical theory. That method, however, allowed to use only one objective function, i.e. target variable as a linear function of the features (independent variables). The default event, as a binary variable, was not correctly covered by linear regression (hence the OLS assumes target as a linear function and is not bounded by 0 and 1). That theory was used with general approval and acceptance of its drawback (for the binary target variable modeling), mainly due to the lack of other, more adequate models. After the introduction of logistic regression (Cramer, 2002), which allowed intuitive and interpretable modeling of binary output, the consensus also changed and all credit analysts started to use today's most widely used model – the logistic regression.

The understanding of the logistic regression model requires, initially, an introduction of the binomial distribution. Let us consider a random event involving, i.e. default or non-default; that random variable, Y, may take two values: $Y = 1$ or $Y = 0$. Let assume that the probability that $Y = 1$ is equal to some p, i.e.: $p = P(Y = 1)$. Probability of the opposite event, i.e. non-default, can be calculated by the formula: $P(Y = 0) = 1 - P(Y = 1) = 1 - p$. Let us assume that the random variable Y has its instance y. In other words – its value has been observed (measurement was taken). Let us calculate the probability of observing this value. We can write it in two variants:

$$P(Y = y) = \begin{cases} p, & \text{when} \quad y = 1, \\ 1 - p, & \text{when} \quad y = 0, \end{cases}$$

or in one common formula:

$$P(Y = y) = p^y (1 - p)^{(1-y)}.$$

After the transformation into the final form we get:

$$P(Y = y) = \exp\left(y\ln\left(\frac{p}{1-p}\right) + \ln(1-p)\right).$$

The element defining the *logit* function appears here for the first time:

$$\text{Logit}(p) = \ln\left(\frac{p}{1-p}\right),$$

which is an important notion in the logistic regression method.

Let us consider a more general situation. In our random sample containing historical data, we observed N observations – instances of y. Each observation of the random variable Y_n associated with the status of the event default, has the value y_n, where n is the unique observation index. The model we are interested in is explaining the relationship between the probability of the event *default* which is often mathematically written as $p_n = P(Y_n = 1)$ and predictors denoted as a sequence of variables $x_n^1, x_n^2, ..., x_n^m$, where m is the number of variables in *Analytical Base Table* (ABT) – special data structure for modeling. At the beginning, the regression part is defined, namely a combination of predictors:

$$X_n\beta = \sum_{i=0}^{m} \beta_i x_n^i = \beta_0 + \beta_1 x_n^1 + \beta_2 x_n^2 + ... + \beta_m x_n^m.$$

This combination is associated with a non-linear relationship with probability p_n. Even though there are plenty of similar *sigmoid* link functions that can be defined and used in practice, the logit function has become the most popular.

Its popularity is due to the ability to interpret the $\frac{p_n}{1-p_n}$ term which is called the odds of default event which is the ratio of the probability of the event to the probability of the opposite event. So, we have a model that makes the natural logarithm of odds or logit of the probability of occurrence of the event default dependent on the regression part $X_n beta$. Finally, the following equation is estimated:

$$\text{Logit}(p_n) = X_n\beta,$$

where X_n are given predictor values, p_n are theoretical probability values of the event *default* for n-th observation and vector of coefficients β is searched for.

The logistic regression model (also known as logit model) uses logit function as a link function. Logit model and models with other link functions (along with its assumptions regarding the distribution of objective functions) are regarded as the *Generalized Linear Models* (Dunteman & Ho, 2005).

The β_i coefficients are calculated (estimated) based on the maximum likelihood method. It differs from the previously known least squares method and is unfortunately associated with a more complex algorithm for searching for the maximum function. They are found by the iterative method, approaching the result with increasing accuracy at every step. If the next steps cause the resulting change to be less than the set accuracy (convergent threshold), then the algorithm stops and a solution is found. Otherwise, the algorithm is not convergent and, unfortunately, the input parameters need to be slightly changed. The most popular algorithm is the *Newton-Raphson method*, in which successive iterations are determined by moving on the vector determined by the likelihood function *gradient.*

The maximum likelihood method, described by Fisher in the XX century (R. A. Fisher, 1922), is based on elementary, yet profound observation/intuition: the probability of obtaining observed values of variables must be the highest (thus *maximum likelihood* method). Discarding the effects of random fluctuations (which in unbiased estimation are expected to be zero), we can assume that in case of different observations we would have different probabilities of occurrence (and the maximal likelihood would be obtained for the observed set of variables).

Using the assumption about the independence of the observed events (i.e. that the probability of several events occurring simultaneously is equal to the product of their probabilities), we can calculate:

$$P(Y_1 = y_1, Y_2 = y_2, ..., Y_N = y_N) =$$

$$= P(Y_1 = y_1) \cdot P(Y_2 = y_2) \cdot ... \cdot P(Y_N = y_N) =$$

$$= \prod_{n=1}^{N} P(Y_n = y_n) =$$

$$= \prod_{n=1}^{N} \exp\left(y_n \ln\left(\frac{p_n}{1-p_n} \right) + \ln(1-p_n) \right) =$$

$$= \exp\left(\sum_{n=1}^{N} \left(y_n \ln\left(\frac{p_n}{1-p_n} \right) + \ln(1-p_n) \right) \right).$$

The likelihood function is the probability of observing all the y_n values together. By applying the logarithm function and inserting the appropriate regression parts instead of logits, one obtains the final logarithm form of the likelihood function – L:

$$\ln(L(\beta)) = \sum_{n=1}^{N} \left(y_n X_n \beta - \ln\left(1 + \exp(X_n \beta)\right) \right).$$

The essence of the maximum likelihood method is, therefore, to find a vector of β coefficients such that the logarithm of the likelihood function is the largest. The regression expression $X_n\beta$ is sometimes transformed as scorecard points – for more on the scorecard points calculation, see Section 4.3.1.

This may also be achieved by an analysis performed to verify the correctness of discovered rules, categories per variables and their relation between customer profile, indicated by a particular category and partial score. That structure of the model seems to be efficient because many properties can be seen without any special report or data visualization method. Everything can be read based on three information: description of a variable, category condition and value of the partial score

Binning and variable report

In practice, one has to transform the continuous variable into the categorical data (e.g. influential variable prevention). We present an example of a variable report, interpretation cause and effect analysis.

Number	Category (condition)	Number of cases	Percent	Risk (%)
1	Missing	11 564	70.5	5.6
2	167 < ACT_CUS_SENIORITY ≤ 227	1 165	7.1	4.1
3	ACT_CUS_SENIORITY ≤ 96	848	5.2	2.9
4	227 < ACT_CUS_SENIORITY	1 427	8.7	2.8
5	96 < ACT_CUS_SENIORITY ≤ 132	710	4.3	2.0
6	132 < ACT_CUS_SENIORITY ≤ 167	700	4.3	1.1

Table 4.2: Report of categories of variable ACT_CUS_SENIORITY – number of months since first credit, months on book
Source: own work

The WoE transformation is a well-known notion Weight of Evidence (Siddiqi, 2012). It is very similar to logit value because:

$$\mathrm{WoE}_k = \ln\left(\frac{G_k/G}{B_k/B}\right) =$$

$$= \ln\left(\frac{B}{G}\right) - \ln\left(\frac{B_k}{G_k}\right),$$

$$\mathrm{WoE}_k = \mathrm{Logit} - \mathrm{Logit}_k,$$

where k – means any variable category, G, B – the number of goods and bads customers in the entire population and G_k and B_k only in the category. Therefore, the Weight of Evidence for a category is the difference between the entire population logit and the category logit. Therefore, the WoE approach can be also implemented as a logit approach.

4.1.3 Drawbacks of the classical scoring models

The classical approaches to credit scoring, apart from clear advantages (i.e. straightforward implementation, auditability, interpretability), have some drawbacks. Among some, we would like to point out few most essentials (Johnston & DiNardo, 1997; Kennedy, 2003; Molnar, 2019):

- as discussed in Section 4.3, classical models require a burdensome, time-consuming process of variables maintenance, correction and at the modeling stage selection to the model; this usually involves the engagement of an entire team or even department in banks to handle those processes; moreover, these efforts are typically multiplied, for each of the several scoring models (e.g. products, clients) that are utilized in the bank.

- linearity and strict functional form of the models. The linearity of the model assumes the linear (usually affine) relation between features and the target variable. This property is generally perceived as an advantage, i.e. interpretability of the model, well-known properties, straightforward model implementation and maintenance, additivity but also the main drawback, i.e. inability to model complex functional relations, requires heavily input transformations to represent non-linearities.

- interactions between particular features to predict the target outcome, after considering the individual feature effects (e.d., distinct effects of Age and Income variables that are not captured by the sum of partial scores); in principle, classical models allows to model interactions between features, by simply adding new features that are compositions; this results, however, in an additional increase of the issues related to data management and handling (in case of behavioral ABT, the original number of variables – dummies and real-valued – may be equal to few thousands)

- in the context of credit scoring and required interpretability of model, some of the variables, especially those converted from categorical variables into the dummy should be included/excluded in groups. Moreover, we recommend the dummy coding that uses as the *reference category*, the one with the largest *cardinality*, i.e. the most observations' coverage.

4.2 Machine learning for credit scoring

Along with further developments in computer science, as well as lowering the costs of computation units (CPU, GPU), new models and techniques for data analytics were discovered and introduced in day-to-day operations.

This situation refers strongly to the IT sector which mostly uses the Business Intelligence software but also advanced analytic solutions – primarily for client acquisition, pricing or internal decision-making processes.

Big Data transformation, as this process is usually referred to, also embraces some of the banking processes (e.g. demand elasticity computation, text mining on the current accounts). However, as of today, the usage of more advanced methods in credit scoring is not widespread. This situation, as a result of the outlined constraints of classical models, will start to change as a new and more generally accepted framework supporting machine learning (especially in terms of interpretability and auditability) will be developed.

In this section, the formulas and principles of the most popular and widely-used machine learning models are discussed. The selection was also based on the adequacy and applicability to the credit scoring context – presented methods, in our opinion, are the most matching and relevant.

4.2.1 Machine learning methods – introduction

The *machine learning* refers to a wide group of statistical algorithms and methods (including the logistic regression) that allow, through parameters' estimation, find patterns in, usually massive, sets of data. To narrow down the scope of discussion, in this chapter we focus only on the supervised learning methods (i.e. the training set contains the values of the target variable) utilized for classification problems (hence the default flag is binary outcome variable). We strongly suggest the following references for the introductory texts on machine learning: (Chollet, 2018; Varian, 2014).

The beginnings of machine learning date back to the 1950s, when information and telecommunication sciences were developing intensively. Part of the leading research units, in addition to conducting scientific work in the field of computer science, also researched subjects related to questions that concerned the way the brain works and the similarity of this action to the mechanics of computer systems' work was shaped. Early computer programs contained only hardcoded programs – such programs do not modify their operation with each iteration. For quite a long time, many experts have believed that human-level AI can be achieved by implementing into computer programs enormous sets of rules and principles governing the operation of these programs. This approach is referred to in the literature as *symbolic artificial intelligence* and even today, it is an essential paradigm in artificial intelligence from the 1950s to the late 1980s.

Symbolic artificial intelligence reached its peak of popularity in the 1980s – with the so-called boom of *expert computer systems*. While it proved suitable for solving well-defined, logical problems, this approach showed significant difficulties in solving complex issues such as image classification and speech recognition.

Machine Learning has emerged as an alternative approach to solving complex problems. Machine learning is an attempt to answer the question positively – can the computer go beyond human-defined knowledge and principles and learn on its own how to perform a specific task. Rather than manually creating rules for how to approach a task for a given data set, the computer program automatically learns these rules based on the data. The above-mentioned example of a conceptual approach to solving severe problems is a *new programming paradigm*. In classic programming, the symbolic artificial intelligence paradigm is introduced by a human being by implementing them in a computer program – data is processed in accordance to these rules and responses are returned as a result. In machine learning, the input to an application is answers and data – the computer itself, presented with many examples relevant to the task, finds a statistical structure in this data which ultimately leads to creation of rules for automating task solving.

4.2.2 Regularization techniques

In econometric modeling regularization techniques are employed in the case of ill-posed problems or overfitting of the model; the architecture of those methods allows to reduce the variance of estimated parameters through a selection of explanatory variables and, hence, also a reduction of the variability of the model's predictions.

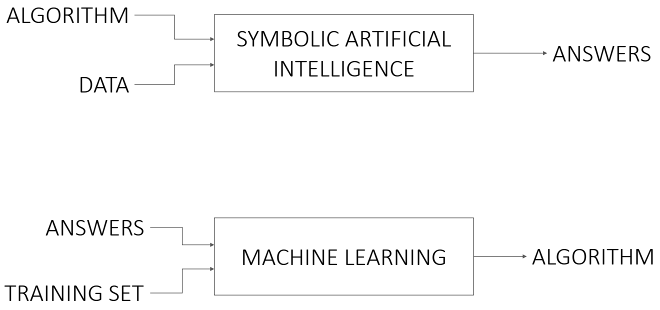


Figure 4.1: Comparison of two paradigms: symbolic artificial intelligence and machine learning
Source: own work

The first case, i.e. ill-posed problem, refers to the econometric models that exhibit instability of estimation. As a result, the instability of the model leads to an instance when a small fluctuation in the training set results in a significant deviation of the model's forecasts; this feature of the econometric model, especially within the context of credit scoring, is undesirable and in the extreme leads to an unacceptable model. In the context of ethical and trustworthy machine learning, we claim that the stability of the model constitutes a necessary condition of the technical robustness aspect. The ill-posed problem may have originated, among others, from highly correlated explanatory variables – the so-called multicollinearity (for the diagnostics tools, see Section 5.4). The examples of highly correlated features usually appear in behavioral scoring (i.e. using the same features with different observation horizon), for example: variables denoting the credit arrears in different tenors, different tenors of DPD, dummy behavioral variables.

The second case, i.e. overfitting of the model, refers to an instance of a model that exhibits a relatively high performance on the training set, but the prediction error expands siginificantly on different data sets. It is being said that the model has been overfitted to the initial data set (i.e. training set) and does not allow inferring on different data sets. The cause of such a scenario may have originated from the inclusion of too many independent variables (Everitt & Skrondal, 2010). The data-generating process that is the subject of econometric modeling is usually unknown; the use of too many explanatory variables on a training set can lead to a spuriously high level of predictive power in training set – the predictive power on test sets, however, is significantly lower (Johnston & DiNardo, 1997). Similarly to the ill-posed problem, the absence of overfitting establishes a necessary condition of the technical robustness on an algorithm (and hence often called algorithmic robustness).

Ridge regression

Ridge regression is a regularization technique used for estimating parameters in cases where the explanatory variables exhibit near multicollinearity. As mentioned in the Section 5.4, multicollinearity causes an increase in the variance of model parameters and thus an inability to interpret the model parameter values.

The ridge regression technique is a method of biased estimation – the estimated parameters are biased (i.e. $E[\hat{\beta}] \neq \beta$), but this allows to reduce the variance of estimated parameters. Therefore, the rationalization of the model is able to reduce the variance of the model predictions and parameter values at the cost of a small estimator bias. That trade-off (lower variance and little bias) is, from the business point of view, a desired property – unless it severely impacts the model's interpretability. Let $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \ldots, \hat{\beta}_p)^T$. For estimation, the values of explanatory variables are usually standardized, i.e. $\sum_i x_{i,j}/N = 0$ and $\sum_i x_{i,j}^2/N = 1$. The ridge regression parameters $\hat{\boldsymbol{\beta}}$ are defined as an optimal solution for the quadratic programming problem with inequality constraint (Tikhonov & Arsenin, 1977):

$$\arg\min_{\beta_i} \left[\sum_{i=1}^{N} \log(1 + \exp(-\sum_j \beta_j x_{i,j} y_i)) + t \sum_j \beta_j^2 \right],$$

where $t \geq 0$ is the tuning parameter of ridge regression (also referred to as a penalty term).

The selection of the appropriate t parameter (i.e. the one that generates the smallest prediction error) is made based on the validation data set's

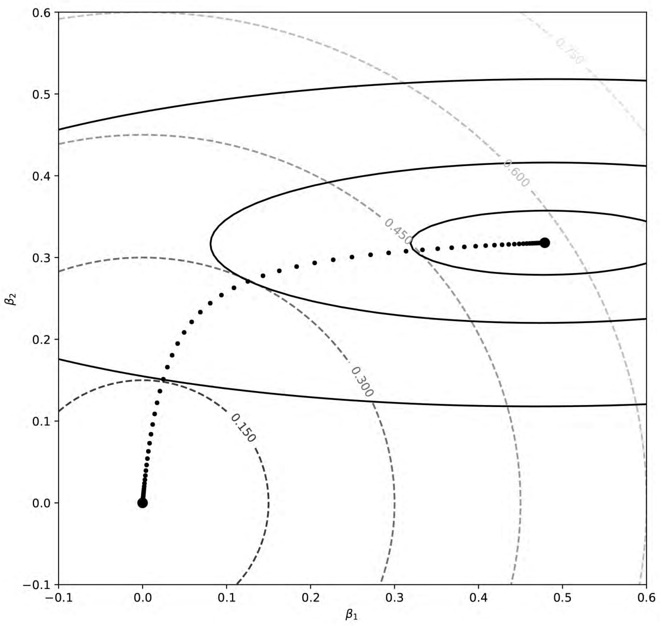


Figure 4.2: The contour curve of the ridge regression loss function
Source: own work

results; for a given set of t parameters, the choice of the target model (i.e. a model with specific tuning parameter) is based on minimization of the prediction error on the validation set.

Lasso

Tibshirani defined the *Least Absolute Shrinkage and Selection Operator* – lasso – as an optimal solution for the following constrained optimization problem (Tibshirani, 1996). Let $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \ldots, \hat{\beta}_p)^T$. It is assumed that the explanatory variables are usually standardized, i.e. $\sum_i x_{i,j}/N = 0$ and $\sum_i x_{i,j}^2/N = 1$. The regression parameters of lasso $\hat{\boldsymbol{\beta}}$ are defined as a optimal solution for the following constrained problem (Tibshirani, 1996):

$$\arg\min_{\beta_i} \left[\sum_{i=1}^{N} \log(1 + \exp(-\sum_j \beta_j x_{i,j} y_i)) + t \sum_j |\beta_j| \right],$$

where $t \geq 0$ is the tuning parameter of ridge regression (also referred as a penalty term).

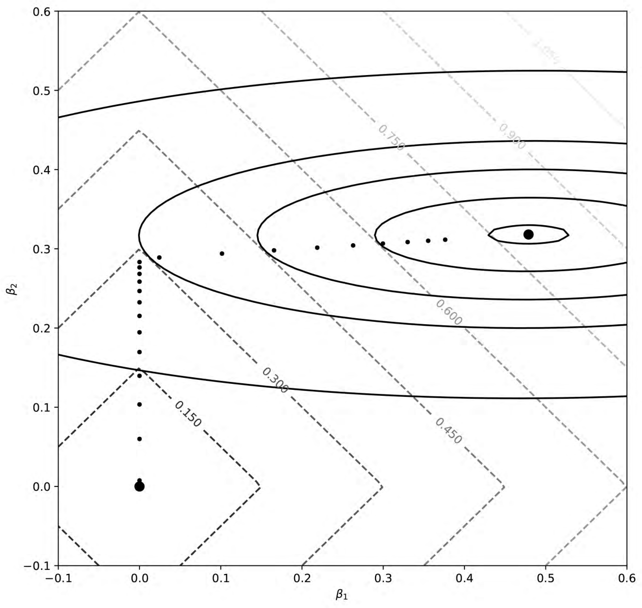


Figure 4.3: The contour curve of the lasso regression loss function
Source: own work

Similarly to ridge regression, lasso regression imposes a limit on the norm of model parameters; in the case of the lasso, however, this is the l_1 norm (also called the urban norm).

Despite the similar form of the models (i.e. loss function of the logistic regression with an additional penalty component), the following differences exist between ridge and lasso regression:

1. Lasso regression allows and it is a relatively common case, to exclude individual parameters (i.e. value $\beta_i = 0$. In the case of ridge regression, however, as a rule, the values of the model parameters are reduced to small values, nonetheless not equal to 0.

2. The lasso regression allows for less complicated and more straightforward parameters interpretation.

3. In the case of ridge regression, it is possible to observe the change of sign for the particular parameter.

4. In the case of lasso regression, the change of the model's parameters follows more linearly with the change of the tuning parameter.

5. Ridge regression imposes a higher penalty for bigger values of the model's coefficients.

6. In the case of two equal features (or strongly correlated), the ridge regression will produce equal weights and lasso will provide the coefficients with the same sign but not precisely similar (however, the sum of coefficients will be constant).

7. Both approaches allow reducing variance of the model's parameters.

Elastic net

The third type of regularization, used in a similar manner as lasso and ridge regression, is the *elastic net* regularization which is the combination of both penalties terms. As it was shown in (Zhou et al., 2015), the elastic net can be reduced to the linear support vector machine (SVM). It was shown that for every setup of the elastic net (i.e. parametrization of penalties) for the binary classification problem, the hyper-plane solution of linear SVM is identical to the solution of the elastic net. This result enables a use of highly optimized SVM solvers and GPU acceleration for elastic net problems (Zhou et al., 2015). The formulae for the logistic regression loss function with elastic net regularization is as follows:

$$\arg\min_{\beta_i} \left[\sum_{i=1}^{N} \log(1 + \exp(-\sum_j \beta_j x_{i,j} y_i)) + \lambda((1-\alpha)t \sum_j |\beta_j| + \alpha(\sum_j \beta_j^2)^{\frac{1}{2}} \right]$$

As discussed in this monograph, it is advisable to use the regularization techniques to eliminate over-fitting. In the beginning, it is always advisable to perform different regularizations and compare the results (see Section 4.3 for the discussion on validation set) – however, in the context of credit scoring, with a wide range of features, one can assume that preferable is the lasso or elastic net since they allow to reduce the redundant features' weights down to zero. In principle, the elastic net is preferred over lasso, as the lasso may be unstable as the number of features is greater than the size of the training set or some of the features are heavily correlated.

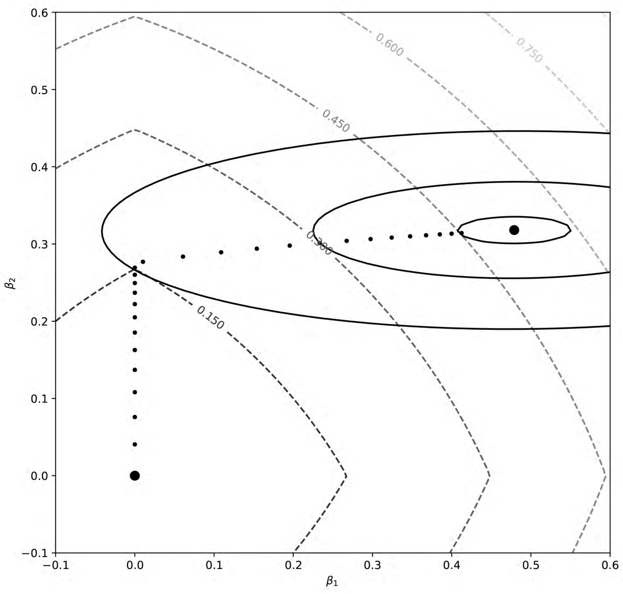


Figure 4.4: The contour curve of the elastic net regression loss function **Source:** own work

The regularization techniques described above are utilized in case of multicollinearity (which is the usual case in behavioral data) and a limited number of observations (compared to features).

4.2.3 Relu neural network

The *relu* stands from the *Rectified Linear Unit.* In the essence, relu neural network can be represented as a linear function nested in a non-linear (i.e. *rectifier*) *activation function.* In fact, if we would replace relu with the sigmoid function, we would end up with the logistic regression (assuming that the neural network is single-layer – only input layer exists). Moreover, due to the *vanishing gradient problem* that occurs in sigmoid activation functions, multilayer neural networks are initialized with the defaulted relu activation function.

The rectifier(relu) activation function has the following form:

$$f(x) = x^+ = max(0, x)$$

where x is the mentioned multiplication of node's inputs and its weights.

The structure of the neural network consists of the *layers* and *nodes*; the example is presented in the Figure 4.5.

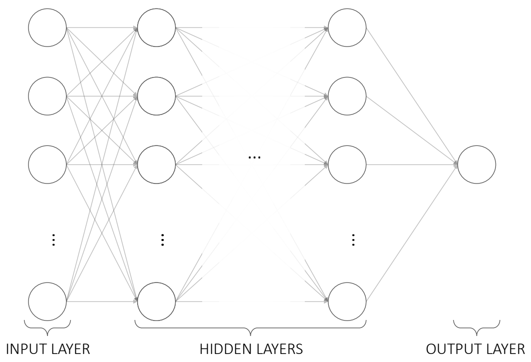


Figure 4.5: The architecture of a multilayer neural network
Source: own work

This neural network presented above has defined 4 layers (input layer, two hidden layer and output layer), with the set of 4, 4, 4 and 1 nodes corresponding to each of the layers. As a matter of fact, the presented architecture is universal (i.e. more layers, more nodes). The activation function, for each layer in the presented example, is assumed to be relu (apart from the input layer – those are simply input features).

On the level of a single node, the output of that node is simply the activation function evaluated on the summed inputs of the node multiplied with the node's weights. The most straightforward activation function that one can assume is the linear activation (no transformation performed) – that mapping, however, does not allow to model complex, non-linear patterns in the data. The relu activation function, as a non-linear, is more preferred as it allows the nodes to represent more complex patterns in the data. The training of the neural network is simply optimizing the weights (presented in Figure 4.5 as the edges) in terms of loss function.

In terms of the advantages of the relu neural networks, one may emphasize the following:

1. As the loss function is convex, the uniqueness of the solution (unimodality) is guaranteed.
2. Absence of the *vanishing gradient problem* and the possibility to generate fast predictive algorithms.
3. Absence of the randomness in the weights initialization and the problems with escaping from the local optima in terms of model's parameters.

For broader discussion on the neural networks, and subjects related (e.g., calibration), we recommend (M. A. Nielsen, 2015).

4.2.4 Decision trees

Decision trees are an algorithm of supervised learning, based on data for which all values of the dependent variable are known. These are nonparametric models in which no assumptions are made a priori concerning the relationship between a dependent variable and explanatory variables, nor any assumptions regarding the distribution of the variables or the model's errors. Therefore, unlike linear regression, they can reflect any non-linear relationship. However, in decision trees it is also possible to impose the monotony of some dependencies, which in the scoring practice is an important feature, both from a business and regulatory point of view. There are some order-preserving decision tree construction algorithms that can be applied to ordinal or continuous variables (Potharst & Feelders, 2002). Moreover, trees reflect well the interactions between variables without the need to process the data and prepare it for use in the model.

Trees are most often used in classification problems when the dependent variable is nominal and takes one of several values representing object classes. In the case of credit risk assessment, the dependent variable is binary and means default/bad or non-default/good for a bank client, i.e. $Y = 1$ for default and $Y = 0$ for non-default. A classification tree is a machine learning algorithm that subdivides a set of observations into subsets using tests conducted on attribute values. The purpose of the division is to obtain subsets that are as homogeneous as possible according to the categories of the dependent variable. The most popular algorithms for the construction of classification trees are those that minimize the measure of diversity in the subsets obtained (Hastie et al., 2009). This is how the Classification And Regression Trees (CART)[1] algorithms work (Breiman et al., 1984) and C4.5 (Quinlan, 2014). Another group consists of algorithms that divide the set based on statistical tests. This group includes, e.g. the CHAID algorithm that uses the χ^2 statistics (Kass, 1980) and algorithms based on permutation tests (Hastie et al., 2009).

Decision tree structure

The tree structure reflects the recursive splits of the set. Nodes and leaves denote the obtained subsets and tree branches show the rules of division. Trees are most often used in classification problems when the dependent variable is nominal and takes one of several values representing object classes. The divisions of sets of observations are determined based on the explanatory variables and tree nodes represent tests carried out on the values of the variables. The root is the first node covering all observations. The branches correspond to different test results. The leaves show the labels of the observations category. Each tree is a set of classification rules and each leaf on the tree is a separate classification rule.

A classification rule is an expression of the form:

IF conditions THEN decision.

Classification rules are used as a convenient way of representing knowledge with a very clear interpretation. They indicate the decisions that are appropriate when the conditions are met. In the case of the problem of classification, the rule can be called classification and read as follows:

IF conditions THEN category.

[1]The CART algorithm developed by Breiman enables the construction of trees in which the dependent variable is either nominal or real. However, due to the area of an application under investigation, trees with a real dependent variable, called regressive trees, will not be discussed here.

The left side of the rule is then a set of conditions imposed on the attribute values and the right side indicates the category of the object. The leaves of the classification tree can be equated with the classification rules for a fixed cut-off point. Changing the cut-off point may change some of the rules so that, under certain conditions, the objects will be assigned to a different class than before.

The classification of the example is based on successive tests, from the root to the corresponding leaf. The example is assigned to the class, the label of which has been indicated by the leaf. Tests carried out in tree nodes are functions defined on attribute values. Each test assigns a finite number of values – test results $t : A \rightarrow R_t$, where $R_t = \{r_1, r_2, ..., r_m\}$ to a fixed attribute. The trees in which each test has exactly two results are called binary trees. The available test results depend on the types of variables.

For continuous and ordinal variables, the breakdowns of sets are determined based on conditions described by inequalities or as ranges of values. For example, for a customer's age variable, these can be divisions that give the following possible test results:

$$R_t = \{\leq 28, (28, 35], ..., (55, 65], > 65\} .$$

Binary splits for numeric variables have the form

$$R_t = \{A(x) \leq a, A(x) > a\} .$$

For example, a binary split for a customer's age variable might produce the following two test results:

$$R_t = \{\leq 35, > 35\} .$$

For nominal variables, the divisions are determined based on conditions of adherence to separable subsets of variable values. For example, for a marital status variable, the following three test results are possible:

$$R_t = \{\{single\}, \{married, cohabitation\}, \{divorced, separated, widowed\}\} .$$

It may also occur that each value of a variable is a separate test result, i.e. $R_t = A$. Binary divisions are often based on the selection of one of the variable values:

$$R_t = \{A(x) = a, A(x) \neq a\}\,,$$

i.e. for marital status:

$$R_t = \{married, others\}\,.$$

This way of dividing the set leads to the creation of multidimensional "cubes" in the space of variables characterizing observations.

In each node that is subject to division, the optimum test is selected from a set of all tests that can be applied to the node. For this purpose, a numerical measure of test quality must be defined. In general, it can be said that the more homogeneous the subset obtained compared to the split node, the better the test is. Several test quality measures which are called splitting criteria, can be used in tree-building algorithms. The most common splitting criteria used for selecting tests are described in the next section.

In the process of tree construction, in addition to the test selection criterion, two other elements are also important. The first one is the criteria that determine whether the division of the set should be continued or completed and a leaf should be created. The set of such principles is called stop-splitting criteria. The most common stop criteria for trees will be presented in the following section. The second element is how to assign leaves to the classes. By default, the leaf gets the label of the class whose share in a given subset is the highest. However, it is possible to adopt different rules for assigning a class to leaves. The issue of labeling, i.e. assignment of classes, will also be discussed further.

Splitting criteria for classification trees

Choosing the optimal test to divide the set, means selecting the attribute to perform the test and possible test results. In each node of the same tree, the available test set may be different. It is also possible that tests in different nodes are based on the same variable but have different sets of results. In algorithms that minimize the diversity measure, usually the functions described below are used as criteria for partitioning a set. In practical use, in credit scoring the imbalance between the number of defaults and non-defaults is significant. Among the discussed methods, the Gini coefficient and Gain ratio are methods that are insensitive to the relative proportion of the two groups.

Entropy and information gain

One of the criteria for the selection of the test is to maximize information growth (Quinlan, 2014). The information or entropy contained in the set of labeled examples depends on the proportions of categories in this set. The entropy of the set is directly proportional to its impurity and is expressed by the formula

$$I(X) = -\sum_{d \in C} \frac{|X^d|}{|X|} \log_2 \left(\frac{|X^d|}{|X|} \right),$$

where:

C – a set of categories described by the dependent variable;

X – entire set;

X^d – a set of observations with category d.

The value expressed in this manner represents the amount of information missing for the correct classification of a randomly selected example. The minimum set entropy value can be 0 and is only achieved if the set is completely homogeneous. The maximum entropy value depends on the number of object categories but is always achieved when the distribution of categories in the set is uniform. The purpose of the test selection and division of the set is to reduce its entropy.

The Entropy in the set of observations with the result r in test t equals

$$E_{tr}(X) = -\sum_{d \in C} \frac{|X_{tr}^d|}{|X_{tr}|} \log_2 \left(\frac{|X_{tr}^d|}{|X_{tr}|} \right),$$

where:

X_{tr} – set of observations with the result r in test t;

X_{tr}^{d} – set of observations with the result r in test t and with category d;

The entropy of the entire set X after applying test t is a weighted average for all test results, i.e. for all values $r \in R_t$ (available results for a given test).

$$E_t(X) = \sum_{r \in R_t} \frac{|X_{tr}|}{|X|} E_{tr}(X).$$

The increase of information in the set after application of test t is expressed as

$$g_t(X) = I(X) - E_t(X).$$

The criterion will select as optimal the test that gives the maximum information increase in the set after its division $\arg\max_{t} g_t(X)$. This is equivalent to the choice of such a test that minimizes the entropy of the set after division.

Information gain ratio

Another criterion for the division of the set is the information gain ratio (Quinlan, 2014). This criterion uses the information gain described above. However, the choice of the test is determined by the value of the quotient

$$gr_t(X) = \frac{g_t(X)}{IV_t(X)},$$

where IV_t is the informational value of the test t, calculated as follows

$$IV_t = -\sum_{r \in R_t} \frac{|X_{tr}|}{|X|} \log_2 \left(\frac{|X_{tr}|}{|X|} \right).$$

For the Information gain ratio the test that maximizes this factor, that is $\arg\max_{t} gr_t(X)$ is chosen as the optimal one.

The information value does not depend on the distribution of classes in the obtained subsets. It only measures the uniformity of the distribution of a set into subsets. The value of this indicator is high when the set is divided into subsets of similar size. This value, on the other hand, decreases when one of the subsets is much larger than the others. Dividing the information gain by the information value plays the role of normalizing the g_t. The disadvantage of the information gain is that this criterion favors tests with a large number of results, even when this is not justified, i.e. the selected test is not actually more effective than others with a lower number of results (Quinlan, 2014). While the criterion based on gain ratio, thanks to the applied normalization, does not have this disadvantage. Moreover, the information gain, due to the preference for tests with many results and the most homogeneous sets, may lead to overfitting models. Using information gain ratio instead of information gain helps to avoid this problem as well.

Gini Index

The Gini index (for further information, see Chapter 6) is another criterion that indicates a test to minimize set impurity after its division (Breiman et al., 1984). The Gini index for any set of labeled examples is represented by the following formula

$$GI(X) = 1 - \sum_{d \in C} \left(\frac{|X^d|}{|X|} \right)^2 ,$$

where the indications used have the same meaning as for entropy.

The Gini index for the set after application of test t is the weighted average of the Gini indexes for all resulting subtests

$$GI_t(X) = \sum_{r \in R_t} \frac{|X_{tr}|}{|X|} GI_{tr}(X).$$

A test that minimizes the Gini index, that is $\arg\min_t GI_t(X)$ is chosen as the optimum test.

Twoing Criterion

Previously described test selection criteria can be applied to any number of subsets created by division. Twoing Criterion is a function used for binary trees, i.e. trees in which each division generates two subsets (Breiman et al., 1984). Since the division generates two subsets, one of them is denoted by the index L (left) and the other by R (right). The function assessing the purity of the resulting subsets is

$$\Phi(X) = \frac{1}{4} P_L P_R \left(\sum_{d \in C} \left| P_L^d - P_R^d \right| \right)^2 ,$$

where the designations are as follows

P_L, P_R – the share of observations assigned to the left or right subset respectively in the whole set;

P_L^d, P_R^d – the share of observations with category d in the left or right subset respectively.

A test that maximizes the above-mentioned measure will be selected for the application.

The χ^2 statistics

An example of an algorithm based on χ^2 statistics is CHAID (Kass, 1980). This statistic is used to study the independence of two variables with discrete distributions. The null hypothesis under test says that the variables are independent. In trees, it is used to assess the probability of independence of a dependent variable and with explanatory variables that are nominal or have been categorised. For each explanatory variable with m values, a contingency table with a dependent variable with k values is built. The table shows the frequency of common occurrences of i-th value of the explanatory variable and j-th value of the dependent variable for $i = 1, ..., m$ and $j = 1, ..., k$. The χ^2 statistics compares the actual frequencies of cumulative occurrences obtained in the table with theoretical values determined under the assumption of variables independence:

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{(n_{ij} - E_{ij})^2}{E_{ij}},$$

where

n_{ij} – frequency of common occurrences of i-th value of the explanatory variable and j-th value of the dependent variable;

E_{ij} – the expected value of the appropriate frequency of common occurrences assuming that the variables are independent.

If the variables are independent, the expected value of common occurrences frequency is

$$E_{ij} = \frac{n_i^1 n_j^2}{n},$$

where

n_i^1 – number of occurrences of i-th value of the explanatory variable;

n_j^2 – number of occurrences of j-th value of the dependent variable;

n – total number of observations. The statistics defined in this way have a χ^2 distribution with $(m-1)(k-1)$ degrees of freedom. It measures the difference between the actual and theoretical distributions assuming the independence of the variables. The greater the value of this statistic, the greater the difference between the distributions. Exceeding the critical value at a fixed confidence level rejects the null hypothesis. According to this criterion, the test with the highest value of the statistics will be selected.

Comments on the criteria

It is difficult to indicate among the above criteria one which always gives the best results. Using different criteria can lead to trees with different structures. The shape of a tree also depends on the number and structure of the classes in a set. The comparison of the test selection criteria for binary splits shows that significant differences in the obtained tree structures appear when there are many classes of objects in the set, i.e. when the explained variable has many values. In the case of credit scoring where there are only two classes and two values of the dependent variable, all the criteria give similar results (Breiman, 1996; Breiman et al., 1984). When using non-binary partitions, Twoing criterion does not apply while Gini index and Entropy and still give similar results. Both criteria tend to obtain homogeneous nodes but entropy tends to result in more balanced partitions with regard to the number of observations in the subsets, which may be disadvantageous in an unbalanced sample.

Binning and variable report

Binning is a data pre-processing technique which consists of discretizing a continuous variable by assigning the values to a set of bins. Such practice helps to make the variable more manageable, e.g. thanks to bucketing outliers in lowest or highest intervals of the range together with less extreme values. This way, outliers become no different than other values from the tail of the distribution. Binning can also solve complications arising from a high degree of skewness of a variable. The approach to binning can be either unsupervised (bins are based on the distribution of the variable alone) or supervised (bins are created using external information, e.g. target variable). One of the supervised methods of binning is fitting a decision tree aiming at forecasting target variables, i.e. default rates, using only the variable the analyst wants to discretize. In its technical aspect, it comes down to searching for a split minimizing the Gini coefficient or entropy, so the resulting leaves (or bins) naturally show decreased entropy (Kamiński & Zawisza, 2012). In practice, the binning of variables should be conducted for the logistic regression; in terms of decision trees, the binning of features is made within the algorithm (generation of leaves).

Stopping criteria

Stopping criteria control if the tree construction process should be stopped or not. An over-expanded tree is overfitted to the training set. Its quality at the test set is usually much lower. Preventing a tree overfitting is possible by using stopping criteria and pruning the tree. The stopping criterion may prevent the overfitting by restraining the node from further division.

The stopping criterion may also be of a technical nature. Such technical criteria are:

- homogeneity of a set, i.e. the situation when there are only observations from one class in a node;
- lack of available tests – when all observations in a node have the same values of the explanatory variables.

The other criteria, listed below, are applied to prevent overfitting of a tree:

- maximum depth of the tree – if a tree has reached the maximum depth indicated in the process parameters, the construction of the tree is finished;
- mininum node size – a node will not be split if child nodes are less than specified minimum node size;
- minimum leaf size – a node will not be split if obtained leaves are less then specified minimum leaf size, the node becomes a leaf;
- minimum leaf purity – a node will not be split if its purity is not less then a specified level, the node becomes a leaf;
- minimum increase in purity – a node will not be split if its purity improvement is less then specified level:

$$\max_{t\in T} \triangle P\left(t, X\right) < \beta$$

where

X – node to be split;

T – set of available tests;

$\triangle P\left(t, X\right)$ – the increase in the purity of the X set after the application of the test t;

β – the minimum required increase in purity.

If entropy is used as a measure of a node X purity, then $\triangle P\left(t, X\right) = g_t(X)$.

Advantages and disadvantages of classification trees

Classification trees have many advantages. First of all – the structure of the tree allows a clear interpretation of the obtained dependencies and classification rules, even for people without advanced statistical knowledge. The form of the tree indicates exactly which variables, how and in what order influence the affiliation of observations to particular categories. Furthermore, the tree clearly shows the interactions between the variables. It is also possible to assess the significance of individual variable in the tree structure and thus also in the classification process. The significance of a variable in a tree shows the extent to which a given variable contributes to the reduction of impurity of obtained subsets.

Classification trees show well the non-linear relationships in the data. They can be built using any type of variable – real, ordinal and nominal. They also do not require special variable processing before model building, although they can also be processed as in other methods, e.g. nominal variables can be replaced by binary variables or WoE. These algorithms also handle well outliers, missing values and mistakes in data. No pre-selection of variables is required. A tree construction method that is based on recursive divisions, makes it possible to select those variables that are most relevant for the classification process. They are also invariant to monotone transformations of numerical variables. Monotonicity constraints can also be imposed which improves the transparency of obtained models and their agreement with available knowledge.

The disadvantage of classification trees is that sometimes a very small change in the value of one variable can completely change the classification result. On the other hand, a small change in the data set may result in a change in the whole structure of the tree. A problem with using trees can also be their quickly decreasing interpretability as the number of nodes grows.

Trees can be components of more sophisticated and advanced models. They form the basis for other advanced algorithms such as random forests and boosting methods. These methods are based on sets of classification trees. Importantly, classification models built with the use of many trees

– as opposed to single trees – do not have most of the negative properties mentioned above.

4.2.5 Random forest

A random forest is a set of many classification trees. It is constructed according to a bagging algorithm, i.e. individual trees are built on independent bootstrap samples selected from the training set. The use of random forests allows to reduce the variance of the forecast in relation to the use of a single classification tree which has a large variance (Hastie et al., 2009). The reduction of variance results from averaging the results obtained from individual trees. The generalization error in a random forest depends on the number of trees, the predictive power of individual trees and also on the correlation between the trees (Breiman et al., 1984). In the case of that method, one can impose the monotonicity constraint on the particular features which in credit scoring practice is useful and essential (both from business and regulatory point of view). Moreover, this constraint can be implemented in both a single tree or entire forest granularity.

Construction of trees in a random forest

The trees in the random forest are built on independent bootstrap samples selected from the training set. This minimizes the correlation between the trees. All trees are built without pruning. During the construction of a single tree, a subset of variables is drawn in each node which will be used to select the optimal test that separates this node. Variables are selected independently for each node. The classification of observations is determined by each of the classifiers obtained and the final decision is taken by voting.

Single decision tree is characterized by a high variance of the forecast. Averaging results obtained from many trees in a random forest reduces the forecast variance. The mean variance for B identical random variables with a σ^2 variance and a correlation coefficient ρ between two variables is equal to (Hastie et al., 2009)

$$V(\overline{X}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

Therefore, the reduction of the variation of the average result depends on the number of trees B and the correlation coefficient ρ. The reduction of correlation between individual trees depends on choosing independent bootstrap samples for building trees and independent sampling of a subset of variables before each division. From among p variables, a subset of $m \leq p$ variables is selected, whereby it is usually assumed that $m = \sqrt{p}$.

To determine the error in the classification of a random forest, it is recommended to make a decision for each observation using only those trees that were not built on the sample containing that observation.

Significance of variables in a random forest

Similarly to individual classification trees, for random forests it is possible to determine the significance of individual variables in the model construction. Various methods of determining the significance of variables based on set heterogeneity indicators are described, for example, in (Breiman, 2001; Louppe et al., 2013). Usually, the significance of variables is measured using the Gini index and this measure is called Mean Decrease Gini (Louppe et al., 2013). This measure shows the average decrease in the Gini index in the set, which results from the use of a given variable. The most significant are those variables that contribute the most to the decrease of the Gini index in a set. Figure 4.6 shows a simple example of a ranking of variables that was based on the Gini index.

Other methods of determining the significance of variables in the random forest are Shapley values (Molnar, 2019) or Variable Importance Measure (VIM) calculations (Breiman, 2001). Both methods are described in Chapter 7. Figure 6.13 shows the Permutation Feature Importance indicators that were determined using the permutation method for determining VIM indices.

This very simple example shows a complete agreement between the indications of both rankings – the most important variable is 'financed amount' and the least important – the variable 'previous contracts'.

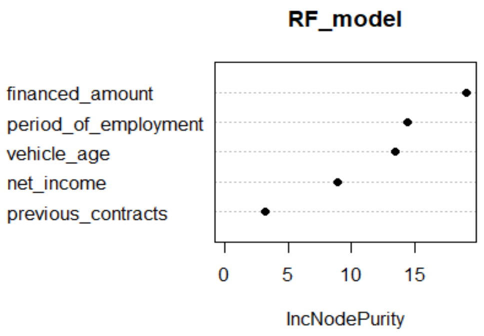


Figure 4.6: Ranking of variables in a random forest based on Gini index
Source: own work

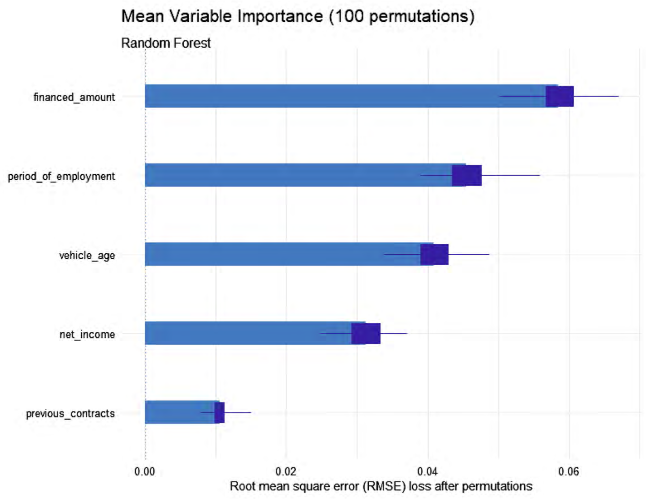


Figure 4.7: The VIM indices obtained by the permutation method for the random forest model
Source: own work

4.2.6 XGBoosting algorithm

Another widely used tree ensemble model is *XGBoost*. After its introduction in 2014, it has been proving itself as a fast and effective machine learning algorithm and a winning method of numerous data science competitions (D. Nielsen, 2016). Just as the random forest, XGBoost is an ensemble learning method, meaning that the result comes from an aggregated output of several models. Decision trees are associated with high variance and ensemble learning helps to reduce such behavior. It uses a concept that averaging a set of observations reduces variance (Hastie et al., 2009). In the case of that method also, one can impose the monotonicity constraint on the particular features (in both a single tree or the entire forest granularity).

There are two main kinds of ensemble learners: 1) *bagging* and 2) *boosting*. Bagging should be known to a reader from random forests. In such models, it does not matter which model was built first because they are trained in a parallel way. Boosting works differently – models are trained sequentially. Each of the models learns from errors made by the previous model.

Boosting is based on similar principles as bagging but it solves one of the bagging's limitations – all models may be wrong in the same area and make mistakes about the same observations. Boosting solves that issue by chaining the models – creating one tree leads to another. Each model focuses on those areas where its predecessor performed poorly (Freund et al., 1999).

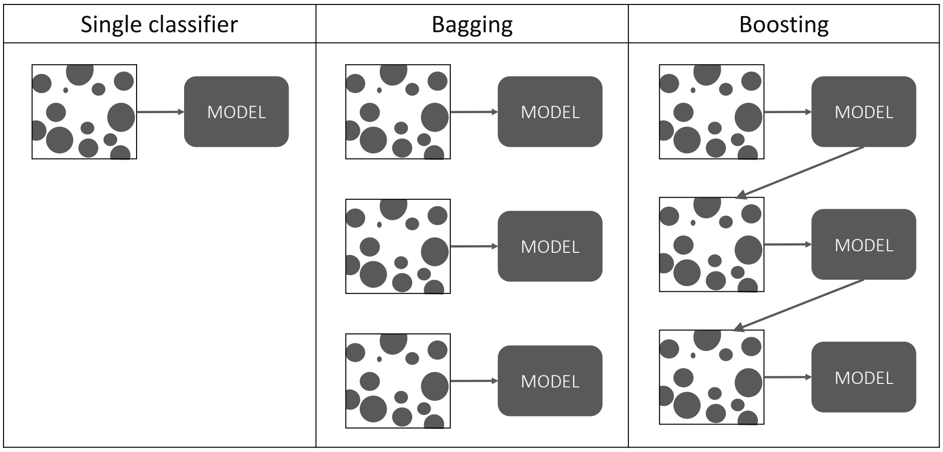


Figure 4.8: Decision trees ensemble methods comparison
Source: own work

In the case of *Adaptive Boosting* (or *AdaBoost* for short), consecutive trees learn from the previous model's mistakes by adjusting weights accordingly. The bigger the error, the bigger the weight for this observation, so the cost of a poor prediction is higher as well. These weights are not the only ones present in the algorithm, though. There are also weights of the trees, a so-called *Amount of Say* which stems from the tree's forecast quality. These weights are connected – the better the tree, the more significant boost will be given to the observations which were not correctly predicted (or had a significant residual) in this model. Let us see an example of how such a mechanism works. After starting from equal weights and fitting the first tree, the following measures are calculated:

1. Amount of Say:

   $\text{Amount of Say} = \frac{1}{2}\ln(\frac{1-\text{TotalError}}{\text{TotalError}})$

   where Total Error is a selected weighted error measure, e.g. weighted error rate. Notice that a classifier with accuracy equal to 50% will have an Amount of Say equal to 0 and thus does not contribute to the final prediction.

2. New Sample Weight for correctly classified observations:

   $\text{NewSampleWeight} = \text{SampleWeight} \times e^{-\text{AmountOfSay}}$

   where Sample Weight is the sample weight used in the last iteration of the algorithm.

3. New Sample Weight for wrongly classified observations:

   $\text{New Sample Weight} = \text{sample weight} * e^{\text{AmountofSay}}$.

Next, weights are adjusted, to sum up to 1. Please find below the visualization of the Amount of Say and new sample weights.

If the Amount of Say is high, the sample weight is scaled by a multiplier close to 0 in case of correctly classified samples (they almost disappear for the next tree) and close to 1 in the case of misclassified ones. The better the tree is, the bigger will be the importance given to misclassified observations. In the case of low value of the Amount of Say, multipliers will have very similar values – for the poorest tree possible with the Amount of Say equal to 0, both multipliers are equal to 1, therefore no change in weights occur.

In contrast to a random forest, where trees are grown to their maximum depth, in AdaBoost, usually trees with very few splits are constructed.

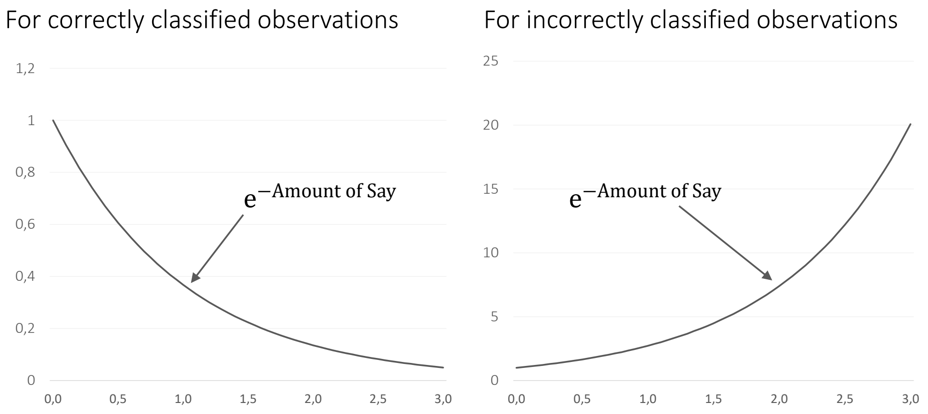


Figure 4.9: Visualization of the new sample weights calculation
Source: own work

In many cases they only have one split and are then called stumps. Combining multiple weak learners leads to creating a powerful, robust model with substantial predictive value. AdaBoost is extreme in this matter, but in Gradient Boosting, regular tree depth is a little bigger – usually between 2 and 8. It is especially adequate in credit scoring as variables in the ABT are usually constructed at depth 2-3. The algorithm allows us to find out how to construct particular variables in the ABT by looking at few of the trees with the biggest importance. Gradient Boosting with logistic objective function can be interpreted as a linear model built on automatically generated ABT variables. Another key difference between Adaptive Boosting and Gradient Boosting is that the latter does not only use weights to correct previous models' mistakes – models learn directly from predecessor's errors instead.

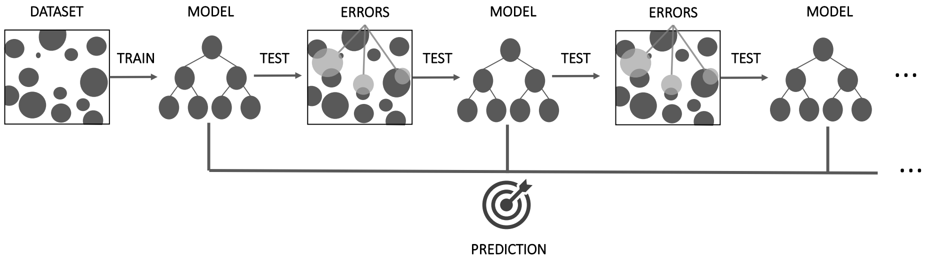


Figure 4.10: High-level architecture of the gradient boosting algorithm
Source: own work

The basic outline of Gradient Boosting can be described as follows in the case of a regression problem. We start from a base learner. Depending on the chosen implementation of the algorithm, it can be a stump, a small decision tree or even a linear model which constitutes our first guess. The simplest version of a base learner would be a single leaf with one value assigned to all observations – an average of the response variable. Once we have our rough estimate, we can calculate the residuals of the not-too-advanced model. In the next iteration, we will model these residuals, not the actual response variable. With each iteration, we'll be modifying the previous estimate with modeled residuals by adding the prediction. Importantly, a hyperparameter called learning rate is used when calculating the result to prevent overfitting and make only small steps in the right direction. For example, instead of adding the forecasted residual itself, we only add 10% of it. Because of this addition, we use even more decision trees to reach the final solution. After growing the next trees, when we make predictions, we sum up the original forecast from the base learner with all of the predicted residuals, each of them multiplied by the defined learning rate. The final prediction takes the form, as presented in Figure 4.11.

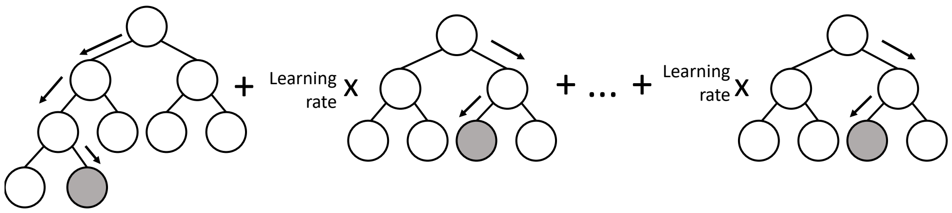


Figure 4.11: Visualization of prediction making in Gradient Boosting
Source: own work

Why is it called gradient then? In the description above, we trained the subsequent models straight on the residuals. This case of Gradient Boosting is the solution when one tires to optimize for a Mean Squared Error loss function. But Gradient Boosting is agnostic of the type of loss function – it works on all differentiable loss functions. We could see Gradient Boosting as a generalization of the algorithm we have defined in the said paragraph.

In a well-known iterative optimization algorithm, Gradient Descent, we try to optimize parameters (θ) of a function by minimizing a loss function (L) which takes the predicted values of the response variable $\hat{y}$ and y itself and returns a loss value. Hyperparameter η is the learning rate used to scale down the gradient of the loss function L.

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta L(y, \hat{y})$$

Let us compare that to our case.

$$F_m = \hat{y} - \eta \nabla_{\hat{y}} L(y, \hat{y})$$

$$F_m = F_{m-1} - \eta \nabla_{F_{m-1}} L(y, F_{m-1})$$

Instead of optimizing the parameters of a function, we optimize the function architecture F. A gradient of the loss function is called pseudo-residuals which in the case of MSE loss, become actual residuals as presented below (with a constant added for purposes of calcu lation simplification):

$$L = \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial L}{\partial \hat{y}} = -(y - \hat{y}) = (\hat{y} - y)$$

There are many modifications of Gradient Boosting such as Stochastic gradient boosting where at each iteration of the algorithm a base learner is fit only on a subsample of the training set drawn at random without replacement. The method proposed by J.H. Friedman was observed to improve the original method's accuracy (Friedman, 2002).

However, a Gradient Boosting implementation which gained the most popularity, is by far XGBoost which highly increased the capabilities of the algorithm. X in its name stands for extreme and it is for a reason. Upgrades and improvements are numerous, so is the number of steps, procedures and parameters a user can set (Chen & Guestrin, 2016). This is why here we will focus only on the basic functionalities of this method.

XGBoost's grand advantage is an intricate mechanism for preventing overfitting. Basic Gradient Boosting already includes a learning rate (eta in XGBoost) for decreasing weights. However, this implementation went much further in this aspect. A vital metric introduced in XGBoost is *Gain*. It is a relative contribution of the given feature to the model. The higher the value, the bigger the importance of a split. Except for stopping fitting if a node reaches a predefined maximum depth, it also allows for pruning trees in an automatic manner using, namely Gain. Starting from the leaves level, XGBoost checks whether Gain from a given split falls below gamma – a threshold of Gain improvement required to keep a division. If Gain is lower than this parameter, the node is pruned and the algorithm moves up branch by branch to the root. In the opposite case, when the node's Gain is higher than gamma, not only is the node left but the parent nodes do not have to be checked for pruning. This stems from the fact that ultimately it does not matter if a good split comes after a very poor one – it is still a split beneficial for the model.

A regularization parameter λ adds another layer to this process. It is a constant that is used during *Gain* and prediction calculations. It affects *Gain* by being added to *Gain's* denominator. Therefore, by decreasing *Gain*, each tree becomes more susceptible to pruning and less prone to overfitting. The higher the parameter is, the more conservative the model. It is worth noting that there is an asymmetry in how lambda affects Gain. The fewer observations are in the node, the higher impact of lambda is which is, of course, targeted at battling overfitting.

As said by Tianqi Chen, the author of the method, *xgboost used a more regularized model formalization to control overfitting which gives it better performance* (Hastie et al., 2009). However, there are still many more clever tricks used in XGBoost which were not described in lengths in this chapter. XGBoost allows for parallel processing and distributed computing, as well as custom optimization objectives. It contains an in-built procedure of handling missing values and does cross-validation at each iteration. It offers efficient memory usage and is highly scalable. The core of XGBoost is Gradient Boosting which was then enriched in terms of systems and algorithmic optimization.

4.3 Frameworks for model development

In this section, the frameworks for scoring model development are presented. In particular, the different process-steps are presented for both classical and machine learning model development.

4.3.1 Framework for the classical scoring model building

Almost all predictive models are built by a similar process called SEMMA, an acronym that stands for Sample, Explore, Modify, Model and Assess, a list of sequential steps developed by SAS Institute Azevedo and Santos, 2008.

- Data partition (Random samples). At least two data sets are created: training and validation, sometimes based on a simple random sample method or time sampling, where one data set comes from a different data period than another.

- Binning – categorization of every variable (regardless of the type of variable: nominal, ordinal, binary or interval). Mainly based on the entropy measure, each continuous variable is categorized into an ordinal variable. This is a typical method used in decision trees (classification) algorithms, see Section 4.2.4. Nominal or ordinal variables are also grouped together to form more numerous categories, more representative in the population.

- WoE or Logit transformation. Every variable is transformed based on created categories into interval variable having WoE or Logit values in categories. That idea is very useful to avoid many problems such as missing imputation, sensitivity to outliers or dummy variables in coding nominal variables. All the mentioned problems are discussed in detail in Chapter 5. The WoE transformation, compared to dummy coding, allows to save the degrees of freedom – in the case of behavioral scoring or application scoring based on the credit bureau ABT, the dummy coding may drastically reduce the number of observations that may be utilized in estimation. Moreover, the

WoE results in easier auditability and revision (i.e. negative betas problems – see Section 4.3.4). In the case of small samples, the WoE provides a smoother assessment of the body of distribution (dummies are producing more focus on tails).

- Variable preselection (feature first selection) – rejection of variables with low predictive power. At this stage, variables that are unsuitable for the final model are rejected based on simple one-dimensional criteria which have either a significantly weak prediction value or are very unstable over time.

- Multidimensional variable selection – model generator. In the logistic regression method, there are many different feature selection heuristics, such as stepwise: forward, backward or stepwise based on p-value criterion or branch and bound (Furnival & Wilson, 2000). The last one is a very convenient method because it allows you to create many models providing three options: *start*, *stop* and *best* which means: from what number of variables in the model to start, how many variables to end and how many best models to choose. In the case of the stepwise approach to variables selection, the potential issues that may arise should be noted, i.e. a) instability of variable selection or b) randomness of entering the model with many variables or c) disturbance of p-value of the statistical tests (e.g. Wald test which is used in variables statistical relevance testing). The stepwise method does not possess the property of independence of irrelevant alternatives. The regularization methods, each time consider the entire set of variables and do not have the problem of influencing the estimation of previous steps (i.e. previous variables exclusions); in stepwise, the variable that was already excluded will not be included again.

- Model assessment. There is no single criterion for assessing the model. Therefore, several criteria are used mainly related to predictive power: Gini, stability: Gini delta –a relative difference in prediction between training and validation sets, collinearity measures: maximum of VIF – Variance Inflation Factor, maximum of Pearson correlation coefficient on all variable pairs, maximum of Condition Index, significance measure: maximum of ProbChiSquare, maximum of p-value for all variables. In paper Lessmanna et al. (2013) several more are mentioned. This topic is still very current, many authors question relying on Gini statistics (Scallan, 2011). Moreover, some business criteria not connected to classical statistics should be emphasized, such as variable reliability – can the variable be verified, cost of data, of particular variable and complexity of variable calculation.

- Model implementation. It was not defined in SEMMA diagram but that step is very important and cannot be omitted. Awareness of the implementation of the model in a specific IT environment ensures consistency and forces the analyst to make many decisions during the model building process.

- Monitoring and backtesting. This topic will not be addressed directly in the present work and also is not included in SEMMA but in practice it is always an indispensable stage of the model lifecycle. You need to know how to test the correctness of the model's usage, how to make sure that the opportunity to build a better one has already appeared and how to compare both models. Finally, you need to know what criteria should be used when deciding to exchange for a new one. Also, you always need to reserve IT resources and other staff for model monitoring before it will be implemented.

Partial score calculation for risk scorecard

The general logistic regression equation can be written by:

$$\text{Logit}(p_n) = X_n\beta, \tag{4.1}$$

where β represents the regression coefficient vector. The X_n matrix can be written in detail as follows:

$$X_n = l_{ij}\delta_{ijn},$$

where l_{ij} is a logit of j-th category i-th variable and δ_{ijn} is zero-one matrix with 1 when n-th observation belongs to j-th category of i-th variable. Also, a simplified assumption is made that each variable has precisely the same number of categories so that we can use simpler notation – the number of groups is equal to v.

Note also that in the case of only one variable in the model, we have a fairly simple situation where the coefficient $\beta = 1$. This is due to the fact that both sides of the equation 4.1 have the same logits. It also means that the assumption of a linear relationship between the objective function and the predictors is met in advance. In the case of many variables in the model, the coefficients should be all positive, since logits and variable category numbers have such monotonicity. If the variable category number increases, the risk decreases. Therefore, if there is a change in the sign of the regression coefficient, then we deal with strong collinearity or a case of a confounding variable. Counting negative coefficients is ,therefore, one of the collinearity detection methods, see more in Section 4.3.3.

The product of the X matrix and β vector standing on the right side of the regression equation 4.1 is the score value for a given observation. It is not calibrated and is difficult to interpret. Usually, a few simple transformations are made to give it a more useful form. Note that if the probability value of p_n increases, then its logit will also do so and thus the score will increase as well. Therefore, the higher the score, the more likely the client is to pay back the loans. Most often, the score value is calibrated through a simple linear function:

$$\mathrm{Logit}(p_n) = \ln\left(\frac{p_n}{1-p_n}\right) = S_n = aS_n^{\mathrm{Calib.}} + b,$$

where $S_n^{\mathrm{Calib.}}$ is a calibrated score and S_n the raw one while a and b are coefficients. They are determined so as to obtain an additional property. To present the example, let us assume that: for a value of 300 points the chance of being a good customer should be 50 and when the odds doubles, i.e. it will be 100, then the score should be 320. Odds is defined as the quotient of the number of goods to bads customers or as the ratio

$\frac{p_n}{1-p_n}$. Odds 50, therefore, represents the customer segment, with 50 goods customers for one bad customer. Therefore, the system of two equations should be solved (Siddiqi, 2012):

$$\ln(50) = 300a + b,$$

$$\ln(100) = 320a + b.$$

This set of equations has the following solution:

$$a = \frac{\ln\left(\frac{100}{50}\right)}{20} = \frac{\ln(2)}{20} \approx 0.03466$$

$$b = \ln(50) - \frac{300 \ln\left(\frac{100}{50}\right)}{20} =\approx -6.4852$$

The final partial score is often rounded to the nearest integer number. In Table 4.3, we present the example of Partial Score calculation (for a similar example, see Mayes, 2004). The value of 30 in calculation formula column is associated with the number of score points required to double the odds (i.e. after how many score points the odds of being good clients doubles) – the division of ln(2) is required to obtain proper scores (Scallan, 1999).

Variable	Range	β	Calculation formula	P. Score
Age	<20	-	= 0 (Ref.)	-
	20-35	0.833	$= 0.833 \times (\frac{30}{\ln(2)})$	36
	35+	1.245	$= 1.245 \times (\frac{30}{\ln(2)})$	53
Income	<1500	-	= 0 (Ref.)	-
	1500-3500	1.528	$= 1.528 \times (\frac{30}{\ln(2)})$	66
	3500+	2.178	$= 2.178 \times (\frac{30}{\ln(2)})$	94
# 30 DPD alarms	0	2.139	$= 2.139 \times (\frac{30}{\ln(2)})$	92
	1	2.027	$= 2.027 \times (\frac{30}{\ln(2)})$	87
	2+	-	= 0 (Ref.)	-

Table 4.3: The example of partial score calculation
Source: own work

4.3.2 Framework for machine learning model development

The machine learning framework requires the division of initial data into three samples, i.e. *training set*, *validation set* and *test set*. This approach differs from the classical econometrics where the initial data set was only divided into a training set and test set (hence, one developed only one model from the very beginning and to evaluate its performance out of sample needed only one test set. In machine learning, due to the space of hyperparameters is wider than in classical model (i.e. the machine learning models have, on average, more parameters of training the model than classical model), there should also be additional data space where models will be compared out-of-sample and the best one will be selected. In terms of the framework of model creation in machine learning, it should at least cover the following steps (Chollet, 2018):

1. **Defining the problem, assembling a data set and choosing the evaluation protocol** – this stage is usually reduced when the problem is well known (e.g. credit scoring for mortgages), the data is already in the database (i.e. in a data center of a bank) – the only task is to define the evaluation protocol (i.e. what expectations one set for the model). For more on the model predictive power and other model performance measures, see Chapter 6.

2. **Explanatory data analysis** – each analytical process which is meant to end with the predictive model should include explanatory data analysis which covers among others single variable analysis (e.g. single factor Gini), the correlation between features, data artifacts and missings, outlier detection, endogenous variables analysis and analysis of the variables' distributions over time. The effects of this stage should provide the information, whether a particular model fits the purpose/expectations (e.g. in case of strong multicollinearity, the regularization should be performed, in case of interactions between variables – see Section 4.1.3 – the random forests or XGboost may be adequate).

3. **Development of the primary machine learning-based model** – this stage of the modeling consists of multiple hypotheses stating and evaluation. At this particular stage, different models are created to find a specific class of models that fits the best to the training set. As a result, one should end up with one class of models (e.g. XGboost models) that will be further explored and improved. It is worth noting that a different model should be compared on the validation set, using some of the measures (e.g. Gini, PSI, ROC curve), see Chapter 6.

4. **Scaling up the basic model** by tuning the parameters; it is usually performed on the *validation set* where the competitive models are being evaluated. In this phase usually one develops a model that exhibits the overfitting problem.

5. **Regularizing of the model** – this stage is performed to prevent the *overfitting* of the model. As of the previous step, the model may too strongly represent the training-set phenomenons and may diverge from the *process generating data* which in principle is in the interest of the modelers.

It is worth mentioning that in contrast to classical approaches, where one performs all the necessary features selection (i.e. removal of variables, single-factor analysis), binning and other data transformations (see Section 4.3.1) in machine learning, the diagnostics are shifted to the later stage (i.e. regularization). Moreover, it is assumed that the data preparation process is less demanding and engaging – e.g. in decision trees, the binning and interactions between features are embedded in the model itself and there is no need to prepare the ABT table. That, in principle, should be a cost-effective method since the data handling tasks are less requiring (e.g. there is no need to perform variable selection as it is performed on the model level), allowing to utilize methods highly efficient in terms of predictive power.

From the technological point of view, for implementation of the particular regularization technique, we emphasize that in Python 3.6 one can utilize an open-source package: `the scikit-learn` (Pedregosa et al., 2011).

In this environment, one can utilize the `linear_model` module as follows: `sklearn.linear_model.Lasso`, `sklearn.linear_model.Ridge` or `sklearn.linear_model.ElasticNet`. It is worth mpahsizing that this specification fits the linear model estimation (i.e. OLS type of model). For the purposes of performing regularization in logistic regression, one should use the `LogisticRegression` with additional parametrization of penalty applied: `L1` norm, `L2` norm or `elasticnet` – please note that l2 norm is set as a default parameter (i.e. if one uses the `linear_model.LogisticRegression`, by default uses also the ridge regularization).

As discussed earlier, it is advisable to use the regularization techniques to eliminate over-fitting. In the beginning, it is always advisable to perform different regularizations and compare the results (see Section 4.3 for the discussion on validation set). However, in the context of credit scoring, with a wide range of features, one can assume that the preferable is the lasso or elastic net since they allow to reduce the redundant features' weights down to zero. In principle, the elastic net is preferred over lasso, as the lasso may be unstable as the number of features is greater than the size of the training set or some of the features are heavily correlated.

What is also important is that, it was shown (Zhou et al., 2015) that the elastic net can be reduced to the linear SVM; for every setup of the elastic net (i.e. parametrization of penalties) for the binary classification problem, the hyper-plane solution of linear SVM is identical to the solution of elastic net. This result enables one to use highly optimized SVM solvers and GPU acceleration for elastic net problems (Zhou et al., 2015).

Computational complexity of discussed models

From the technical point of view, the computational complexity of the model may constitute an obstacle to use it in operations – that is usually one of the arguments to remain with well known classical approaches. Discussing the frameworks for machine learning techniques, we would also like to present simple comparisons between models in terms of their complexities.

Let us assume that we have a model that has n variables and m observations. Each of the iterations of a numerical optimization algorithm that searches the domain space of parameters in logistic regression is the order of $n \times m$. The same applies to the regularization techniques – each iteration of optimization results in the complexity of the order of $n \times m$.

On the other hand, in the case of the selection of variables algorithm, e.g. stepwise (forward or backward), it is required to create additional n models which results in the complexity of the order of $n^2 \times m$. In terms of regularization techniques, the complexity is the function also of the number of values of penalty term that is evaluated (also denoted as a *grid size* parameter), independent from n and m. The regularization techniques have a computational complexity of the order of $n \times m \times k$, where k is the grid size in search of optimal penalty term. Additionally, we assume the less efficient method of penalty term optimization (which can be further improved).

In the case of the n big enough (as in the case of ABT for the behavioral data) and the k respectively lower, the regularization techniques are less computationally complex than the logistic regression with the stepwise parameters selection.

In the case of the decision trees, the model complexity is the order of $n \times m \times 2^g$, where g is the maximal depth of the generated trees. It is worth noting that in practice, g is relatively smaller than n.

In the case of random forest, the model complexity is the order of $n \times m \times 2^g \times s$, where s is the number of decision trees in the random forest.

The computation complexity of the XGBoost is the order of $n \times m \times 2^g \times s$, where s is the number of iterations performed in the xgboost algorithm.

4.3.3 Other issues

In this section, we describe the special cases related to data or model specification. Those descriptions should guide technical issues related to data/model handling in the context of credit scoring.

Panel data

One of the most usual issues faced in behavioral scoring is the encounter of panel variables. The usage of the panel variables usually does not significantly affect the model's performance in practice but it impacts the critical levels of the statistical tests, e.g. used to select the variables. This situation is severe for classical as well as in terms of machine learning models. All the methods have similar sensitivity on the panel data – if possible, it is recommended to utilize the dedicated panel models (e.g. Generalized Linear Mixed Models (McCulloch & Neuhaus, 2014), classification tree using wavelet-transformation (X. Zhao et al., 2018)). In case of misspecification of the model (e.g. panel data), the model will encounter the issues with tests' distribution of p-values that were constructed assuming correct specification – if the model does not satisfy the assumptions, the p-values will be incorrect.

Small number of observations

As described before, in the case of behavioral scoring with ABT based models, the usual setup results in a relatively small number of observations (clients) to the number of features (especially the *bad clients*). That problem is profoundly severe as the estimates and model's predictions will exhibit high variance as a result. That being said, in case of a small number of observations (as mentioned before, in relation to the number of features), one should: a) include the regularization (lasso or elastic net) or b) use the WoE transformation instead of dummy variable coding. The first approach is focused on the selection of the most statistically relevant features; the latter primarily allows to lower the size of features space.

4.3.4 Negative betas

In the case of the WoE transformation, the beta coefficients may be negative. The negative betas indicate the interactions between variables which are undetectable in the single factor analysis which is utilized to generate WoE. If the negative betas are estimated, it means that as a single factor, the variable has a positive impact on the target but jointly with other variables, it has a negative effect (so-called Simpson's Paradox). For more information on the higher order association, see Section 5.5.

4.3.5 Categorical variables coding

The categorical data coding is performed to allow the usage of the categorical data, both nominal (with no intrinsic order, e.g. geographical location, city, industry) and ordinal (with inherent order, e.g. income/age group, number of installments overdue). The *dummy coding* is a method of including such variables into the model and is the simplest (and probably the most common) coding system. The approach is to transform the categorical variable into the series of binary variables – e.g. for instance, the weekday may be processed using 6 binary variables indicating weekdays. The exclusion of one of the variables (why six, not seven days) is performed to avoid the perfect correlation problem (more on that in Section 5.4). The one category that is not represented by the series of binary variables is called the *reference level* or *reference category* – its value is embedded in the constant term of the model.

The dummy coding systems may differ in terms of choice of the reference category – relating to the above-mentioned example – which of the weekdays should be chosen. The answer may, in case of classical modeling, be less relevant but become a significant issue in regularization techniques. As for the credit scoring models, we recommend the dummy coding that uses as the reference category, the one with the largest *cardinality*, i.e. the most observations coverage. This coding system is safe for the regularization techniques, i.e. in regularization, the constant value is not penalized because it would result in a severe bias of the model. Moreover, the dummy coding allows for a straightforward interpretation of the model's parameters (which in the context of credit scoring allows the simple diagnostics of the final credit decision).

Moreover, it is worth noting that the type of dummies coding does not impact the classical scoring – and the final changes in partial scores of particular categories (but may result in different score levels). However, in the case of the preselection of variables, the dummy coding starts to matter significantly – if the particular variables are excluded from the model, the final scoring is also impacted. In that case, we recommend using the joint tests for the relevance of groups of dummies (relating to one initial categorical variable). Moreover, as pointed before, we recommend choosing the category with the largest cardinality as a reference category.

The literature also presents different categorical coding systems, e.g. simple coding, deviation coding, difference coding, Helmert coding or orthogonal polynomial coding. For more information on categorical data coding, we highly suggest the (Daly et al., 2016). The comparison of the dummy coding with effects coding (alternative approach) is presented in (Bech & Gyrd-Hansen, 2005). In practice, we recommend testing different coding strategies and selecting the one that produces a model with the best predictive power.

Expert correction and monotonic model response to the input parameter

In practice, both due to business purposes and regulatory recommendations, the bank should allow for the expert correction to make one the model. This action, even if it causes the model to be biased (and inefficiency of model's coefficients), may be justified, especially in case of scarcity of empirical data (e.g. Covid-19 crises is perceived as a world sensation, not observed in the recent banking history). In principle, the model may be backed by the expert knowledge, by the manual modification of the model parameters or imposing constraints on the variables (e.g. based on the `VARCLUS` which methods allow for the expert supported automated mechanism for the variable selection).

In principle, the above-mentioned constraints may also be related to assurance of the monotonic model response – which allows for a natural model interpretation and intuitive results. As one may use different heuristics approaches, we recommend the usage of the constrained regression procedure – the parameter estimation, in most of the examples (except for OLS) the coefficients are estimated using numerical algorithms. One may include the assumed and expected feature into the model, e.g. the set of variables that are preferred by the experts or the monotonic model response.

4.4 Numerical results of models

This section presents an example ofa model's documentation prepared in a classical way, i.e. when scorecard points are prepared and the model is built based on WoE approach. Here, we focus only on the most important scoring criterion – predictive power; however, we strongly suggest that the model documentation should also include the discussion on other criteria (as discussed in Chapter 6).

In this section, as in Chapter 3, we utilize the simulated data for all testing and model techniques comparison. The main description of data creation is presented in (Przanowski, 2011). To satisfy the purpose of our monograph, referenced algorithms were slightly changed to stabilize risk in time and to generate more variables – up to 2304 customer characteristics. Data can be treated as an example of credit bureau data of all available customers and their credits in the market. There are two data sets created:

1. ABT_APP – all customers without delinquency on the reporting date, with only one due installment in the last 6 months and not defaulted in the last 12 months – criterion is corresponding to customers who are applying for a new credit. On that data set acceptance models are often built to calculate the probability of default in case the customer will take a new credit.

2. ABT_BEH – all customers not in default on the reporting date. That data set represents a classical behavioral portfolio where PD model used in IRB or IFRS 9 methods are calculated.

4.4.1 Data structures

As it was mentioned earlier, data were randomly generated. The main information collected per every month of history is the following:

- Paid and not paid installments. Delinquency is aggregated into number of due installment. There is a collected number of paid and due installments per every credit account.

- Credits. All credits dates are collected: of granting and closing (with the final status: not paid or paid).

- Days. There are also collected dates of payments. They can be indicated how many days before or after the due date the payment is done.

- Events. Every customer can have some special events in his history connected with health, work, family and home. Those events can have an important impact on payments.

The methodology of model building is compliant with the base of Basel II documents, such as (Basel Committee on Banking Supervision, Bank For International Settlements, 2005a, 2005c). It is also based on famous methods described in known books and articles such as (Anderson, 2007; Janc & Kraska, 2001; Lessmanna et al., 2013; Oesterreichische Nationalbank, 2006; Siddiqi, 2012; Thomas et al., 2017; Verstraeten & Van den Poel, 2005). Extended ideas, formulas ordered structures and processes are defined in (Przanowski, 2014). Model building process consists of the following steps:

- Random sample, data partition. There are two data sets created: training and validating, based on a simple random sample method (without duplications) in proportions: 50%/50% and it is selected only 75% of all data.

- Binning, categorization, grouping. There are groups created for the categories of variables. Based on entropy statistics, every interval variable is split into categories. This is a typical method used in decision trees techniques. Nominal variables are also grouped, especially when the number of unique categories is too big. After binning, every variable is transformed into a logit variable.

- Variable preselection. Based on univariate statistics for every variable not important, variables are indicated in this way: too small predictive power, too unstable in time.

- Multifactor analysis. Multidimensional variable selection. There, a heuristic method, called branch and bound, is used (Furnival & Wilson, 2000).

- Model assessment. There are no unique model criteria. The most popular are the following: predictive power (Gini), stability such as: Gini or delta Gini –the relative difference between Gini on training data set and validating, collinearity measures: Max VIF – maximal variance inflation factor (Koronacki & Ćwik, 2008), MAX Pearson maximal Pearson correlation statistics on pair of variables and MAX Con Index – maximal condition index (Welfe, 2018) and variable significance measures.

- Model implementation. The model can be implemented in a few systems, so a dedicated document from every implementation is expected.

- Monitoring and testing. After every implementation, testing starts (especially after a few days) and then it is followed by a regular monitoring.

The model is built on all available historical data in the period 2004 – 2018 in ABT_BEH data set. All numbers of observations (customers) for training and validating data sets are presented in Table 4.4. Historical data per every year is presented in Table 4.5.

Data set	N. of obs.	N. of goods	N. of bads	N. of ind.	P. of goods [%]	P. of bads [%]	P. of ind. [%]
Training	52 841	31 010	15 378	6 453	58.7	29.1	12.2
Validating	53 070	31 514	15 281	6 275	59.4	28.8	11.8

Table 4.4: Numbers of observations for data sets used for estimation
Source: own work

The general idea of the variable selection in case of the future usage and business need is to identify all possible customer characteristics bas-

Year	N. of obs.	N. of goods	N. of bads	N. of ind.	P. of goods [%]	P. of bads [%]	P. of ind. [%]
2004	9 593	5699	2742	1152	59.4	28.6	12.0
2005	10 694	6282	3073	1339	58.7	28.7	12.5
2006	10 546	6247	3086	1213	59.2	29.3	11.5
2007	10 758	6370	3054	1334	59.2	28.4	12.4
2008	10 754	6408	3130	1216	59.6	29.1	11.3
2009	10 878	6485	3079	1314	59.6	28.3	12.1
2010	11 204	6577	3223	1404	58.7	28.8	12.5
2011	10 942	6362	3226	1354	58.1	29.5	12.4
2012	11 247	6536	3285	1426	58.1	29.2	12.7
2013	11 220	6660	3262	1298	59.4	29.1	11.6
2014	11 151	6563	3230	1358	58.9	29.0	12.2
2015	11 229	6664	3225	1340	59.3	28.7	11.9
2016	11 228	6560	3295	1373	58.4	29.3	12.2
2017	11 328	6617	3357	1354	58.4	29.6	12.0
2018	6632	4014	1830	788	60.5	27.6	11.9
Total	159 404	94 044	46 097	19 263	59.0	28.9	12.1

Table 4.5: Number of observation for all available data
Source: own work

ing on positive and negative information collected in various databases and systems. Often only negative information can differentiate bad customers, about 2% - 20% of all but the rest of them can have the same scorecard points. Only positive information collected in model variables can differentiate mentioned the rest of the customers, who have never been in delinquency.

Below we present the example of variables utilized in credit scoring.

1. **ACT_CUS_DUEUTL** – ratio: sum of all current due installments of all customer s credits over the sum of all number of installments.

2. **ACT_CUS_N_LOANS_ACT** – number of all current customers credit accounts.

3. **ACT_CUS_UTL** – ratio: sum of all paid installments of all customer s credits over the sum of all number of installments.

4. **ACT_STATE_16_CMIN_DUE** – example of very simple definition of the variable representing the state of a customer 16 months ago, exactly a minimal number of due installments of all customer's credit accounts.

5. **AGS21_SUM_CNCR** – number of granted credits in the last 21 months.

6. **AGS3_CSEV_ALL** – number of all kinds of events (work, family, health, home) which happened to the customer during the last 3 months.

7. **AGS9_PCTL75_CMIN_DUE** – 75 percentile of 9 months series of minimal numbers of due installments.

8. **APP_CHAR_JOB_CODE** – Customer's job code, connected with his income source.

As a target variable for modeling, it is set default_cus12. The share of bad customers based on default definition is defined as DR (default rate), it is an observed value.

4.4.2 Results for different methods

In this section, we present the summary of a brief numerical experiment. To be able to compare different results, we have utilized raw data, as discussed in the previous sections. In terms of classical models, due to the variety of different methodological assumptions (e.g. features selection, data pre-processing), here we present only the machine learning methods. Our findings are as follows:

- the XGBoost model, out of all other models, exhibits the highest predictive power, both on the training set and test.

- the regularization techniques exhibit similar behavior as the logistic regression – in that term, we emphasize the need for further penalty term optimization (in this work, we have utilized only two of them, assuming the notation of `sklearn`). As one can observe, for the low penalization (i.e. Lasso C=1), the results are almost identical as in Logistic regression.

- the random forest and decision tree perform much better in the training set but, as it appears, those methods suffer from significant overfitting. the As a consequence, the models do poorlyy on the generalization on the test (considerably lower delta Gini measure).

Model	Gini Train	Gini Test
Logistic Regression	0,46	0,41
Lasso (C=1)	0,46	0,41
Lasso (C=0.1)	0,47	0,4
Elastic Net	0,34	0,33
Decision Tree	0,48	0,37
Random forest	0,49	0,39
XGBoost	0,51	0,42

Table 4.6: The summary of Gini for machine learning scoring models
Source: own work

As it was mentioned in Chapter 3, we highly emphasize the need for further investigations. According to our best knowledge, the methods presented are significantly sensitive to the input data and pre-processing methods – each of the specific cases should be carefully investigate based on the particular data set. As we have mentioned earlier, the utilized data set was based on the simulation. In Figures 4.12 – 4.18, we present the ROC curves for the particular model.

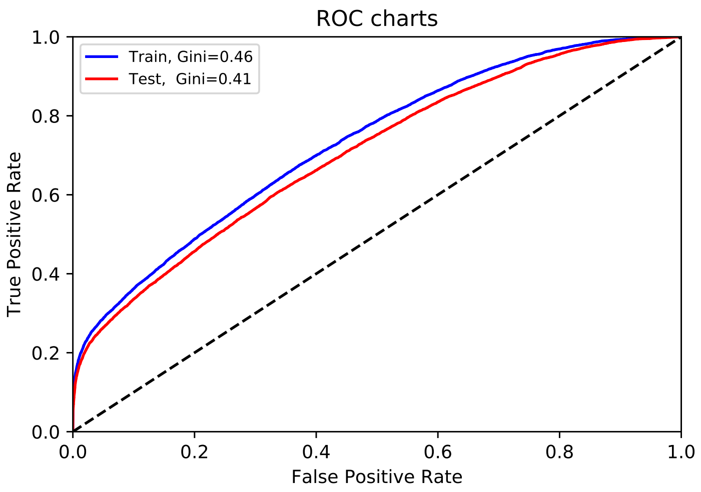


Figure 4.12: Logistic regression results – ROC curve
Source: own work

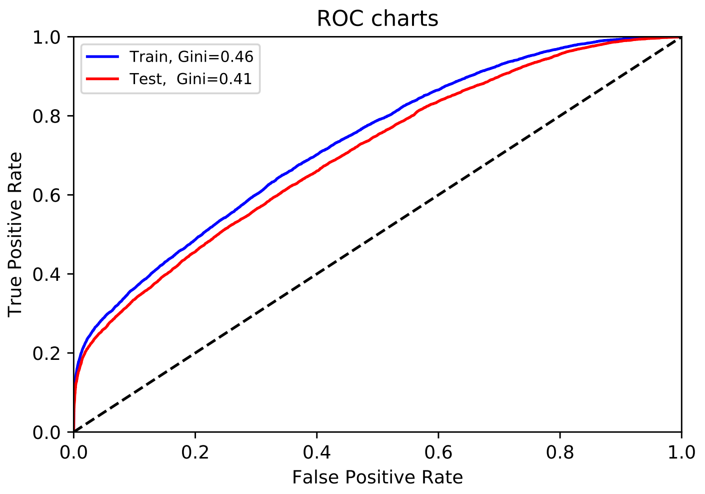


Figure 4.13: Lasso regression (C=1) results – ROC curve
Source: own work

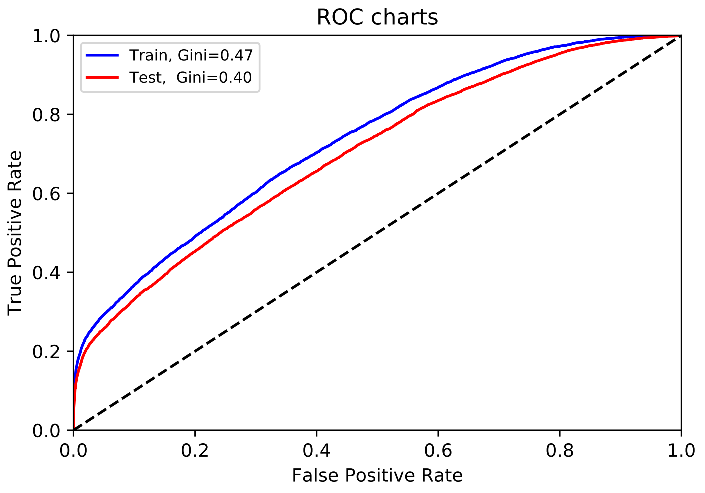


Figure 4.14: Lasso regression (C=0.1) results – ROC curve
Source: own work

4.5 Conclusions

This chapter presents the methods and techniques of data analysis that can be/are successfully used in credit scoring. In the first part of this chapter, we highlighted the classic credit scoring approaches common in many finan-

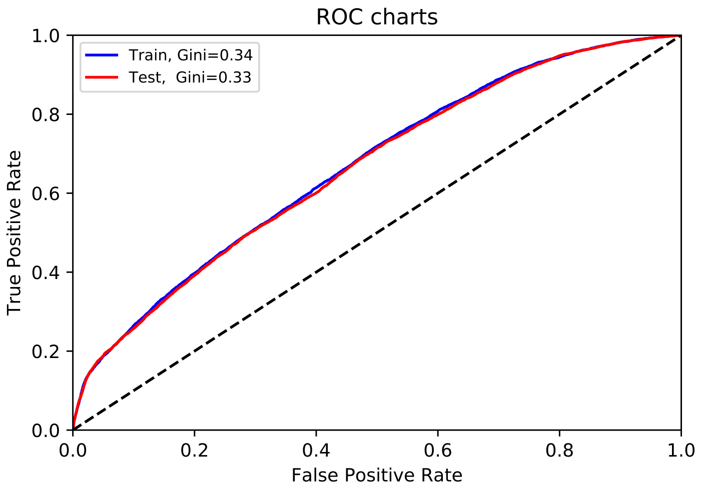


Figure 4.15: Elastic net regression results – ROC curve
Source: own work

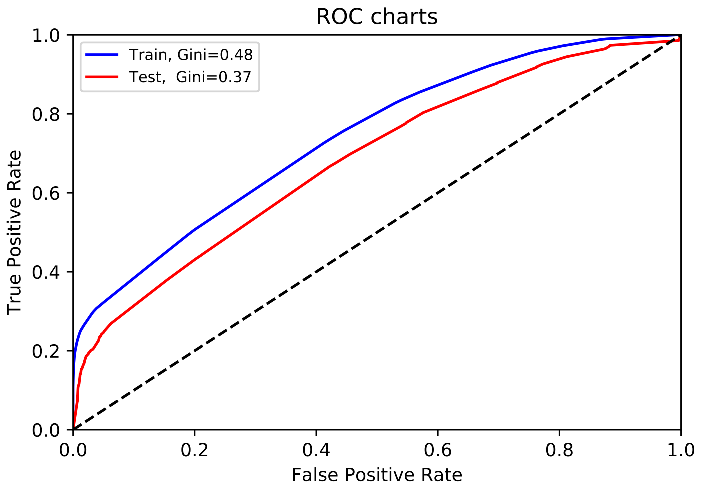


Figure 4.16: Decision tree results – ROC curve
Source: own work

cial institutions today; most banks now base their credit decision processes on these methods. The second part of the chapter presents machine learning techniques that may be used in modern credit scoring. These methods' advantages are undoubtedly related to the functional form's greater flexi-

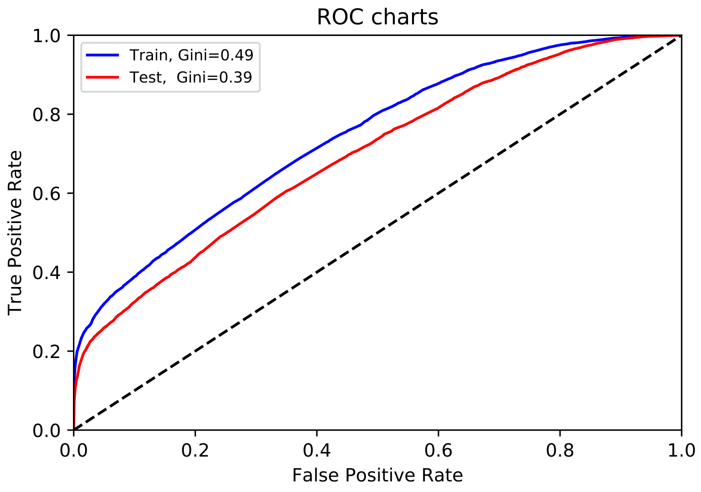


Figure 4.17: Random forest results – ROC curve
Source: own work

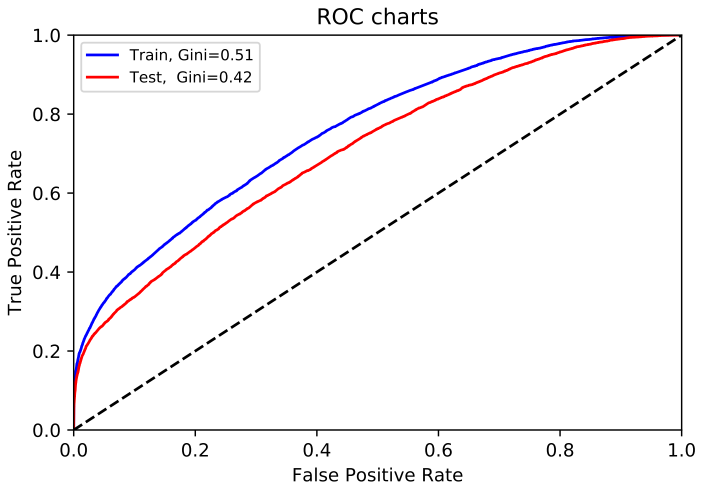


Figure 4.18: XGBoost results – ROC curve
Source: own work

bility (i.e., features and target variables) and automatic handling of issues related to the quality of data used for modeling. The chapter ends with experiments and numerical analyzes that were carried out based on simulation data emulating the actual data used in credit scoring.

This chapter focuses on introducing methods as analytical tools - in particular in the context of credit scoring. Detailed analysis of the sensitivity of machine learning models to data issues or maintaining the interpretability of models is the subject of Chapters 5 and 7.

Chapter 5

Sensitivity of machine learning methods to data issues

Daniel Kaszyński
Kinga Siuta
Bogumił Kamiński

Decision Analysis and Support Unit
Collegium of Economic Analysis
SGH Warsaw School of Economics

The credit scoring data sources enable us to obtain information about individual clients, i.e. socio-demographic data (e.g. age, source of income, employment sector) or behavioral data (e.g. number of days of delay in payment). As a rule, at the stage of preparing data for modeling, each intuitive variable may have some discriminatory power in terms of creditworthiness assessment; naturally, the relevance of each variable are different. In the case of most variables, their added value in terms of the discriminant power of the model as a whole may be negligible which may be related to either a low or even no discriminant power of a particular variable or a high level of correlation with other variables of the model – an example may be various behavioral binary variables that often have an almost perfect correlation.

Testing the influence of the dependent variables on the model may also lead to erroneous results – if we model another random variable with 100 random variables, then with a confidence level of 95%, the average of 5 variables will be obtained which will have a high discriminant power of the model. This means that when a large number of independent variables are used in the regression, with no underlying discriminatory power for the dependent variable, then only on the basis of the adopted confidence level, variables with no discriminant power may be selected for the final form of the model. To avoid overfitting the binary logistic regression model, the focus should be on the number of *events per variable* (EPV) rather than the total number of cases (i.e. events plus non-events).

The problems related to data are not only the issues of selecting variables for the final form of the model but also the issue of maintaining the required data quality. In general, the following data or model related issues are distinguished:

- outlying observations,
- missing data,
- definition of a *bad client*,
- multicollinearity problem,
- higher order associations,
- new categories of the features.

In this chapter of the monograph, the above-presented issues are discussed in more detail in the context of presented scoring models.

5.1 Outlying observations

As the commonly used quotation *garbage in, garbage out* suggests, any data-related issue may result in poor model design and performance – either the predictive power or stability. One of the most usually encountered cases in terms of data quality issues is one related to *outlier* observations, also called *influential observations.* Those are the numerical values that fall significantly far from the expectation, i.e. based on the population distribution. In principle, the occurrence of outliers may originate from a) technical issues or problems (e.g. decimal notation of monetary values, wrong currency assignment – 100 000 PLN earnings equals roughly to about 1 136 700 000 of *Iranian Rials*), b) changes in regulatory definitions (e.g. New definition of Default mentioned earlier), c) consolidation of different data sources (e.g. mergers of the banks/datacenters/department). In the list below, we specify the impact of outliers on the specific groups of models.

- Binned variables – in the case when numerical variables are first binned, outliers do not create an issue; they are added to the highest or the lowest of the bins,

- Decision trees and ensembles of voting decision trees – in principle, the decision trees, random forests and XGBoost algorithm, by default, have set minimum percentile/number of observations in each tree branch which results in robustness to outliers. However, when this constraint is removed, those methods may exhibit sensitive reaction on outliers (as the tree branch may contain only one observation),

- Linear regressions and regularization techniques – these are most affected by outliers as they may get extremely high or low scores, distorting both developments of the model and making resulting scores in production nonsensical.

The recommended approach for handling the outliers is to replace extreme values with certain minimum and maximum percentiles, e.g. if 99%of the population has an income lower than 40 000 PLN, then each income above 40 000 PLN is replaced with precisely 40 000 PLN. Relevant percentiles can be decided for each variable separately. Moreover, the outliers become no different than other values from the tail of the distribution with the application of binning.

5.2 Missing data

There is no data when there is no variable (s) for a given observation. The lack of these variables may result from: a) a human error (e.g. the bank analyst did not enter a value when filling in the customer's data), b) a technical error (e.g. due to a failure, some records have not been put back in the database), c) an observation of a given variable have not yet taken place (in particular in behavioral assessment, e.g. +90 DPD flag in case of new loan) or d) a deliberate action (e.g. the customer did not agree to provide some of his or her data).

While in the case of the first three situations, the fact of the lack of data does not constitute any additional premise / information that translates into the estimation of the customer's creditworthiness, the fourth case may already have such a translation. This situation occurs most often when the client wants to refuse to provide some information (e.g. earnings level) – in this case, the event related to the refusal to provide some information may in itself be a factor in assessing creditworthiness.

As a rule, there are 3 different types of missing data, i.e.:

- **Missing Completely at Random**, MCAR: This is the case of missing data if the probability of missing is the same for all observations. The occurrence of missing data of the MCAR type is not informative in terms of its translation into an explanatory variable.

- **Missing at Random**, MAR: Occurs when there is a systematic relationship between the propensity for missing values and the observed data. In other words, the probability of not observing depends only on the information available (i.e. other variables in the data set). For example, if introverts are more likely to hide information about their income than other people – assuming that the fact of being an introvert does not impact in any way to the creditworthiness (e.g. we assume that the levels of income and expenses of introverts are the same as extroverts). The fact of missing occurrence does not impact the assessment of creditworthiness but it may be more common in some of the sub-populations.

- **Missing Not at Random**, MNAR: if the occurrence of missing data is related to the assessment of creditworthiness – for example, people with a history of difficulties in paying off credit obligations may avoid reporting income; in the absence of earnings information, having a case of MNAR, it can be concluded that the person is less creditworthy.

In terms of data missings (MCAR or MAR), in classical scoring (logistic regression) but also regularization techniques, it is advisable to: a) impute missings (there are many possible approaches discussed in the literature like: population average, median or dominant, model fitting; testing several approaches and selecting one that gives highest predictive power of the model is recommended) and b) create a dummy variable (0 is no missing for a particular characteristic, 1 is missing occurs). That approach also allows us to incorporate potential information on the MNAR into the model.

On the other hand, one of the advantages of machine learning models, i.e. XGBoost, decision trees, random forests, is that they handle the missings systematically/internally. For example, in the case of XGBoost, missing values, by default, are learned during training (i.e. branch directions for missing values are learned during training), see `xgboost v1.2.0.1` (however, one should be careful in case of missing distribution different in training set and test set).

5.3 Selection of the target variable

As indicated earlier, this monograph covers the binary outcome type models which result in consideration of random variable having two possible and opposite outcomes: *default* or *nondefault* of a particular client. The European Banking Authority (EBA) guidelines European Banking Authority, 2017b regulates the definition of a default event significantly. As of today, the notion of a default plays a vital role, due to most of the banks transforming their previous default definition into the new one, i.e. *New definition of Default*. The EBA established tighter standards concerning the definition of default (CRR Article 178) which, in principle, is aimed at achieving greater alignment and standardization across banks and jurisdictions. Those new standards need to be implemented by the end of 2020. The main difference in the new definition is based on DPD and two thresholds: relative and absolute. To accomplished greater standardization across the banking sector, EBA also provided a list of the default triggers (i.e. Unlikeliness to Pay – UTP).

The exact definition of the target variable, including threshold on DPD but also other early warning signals (most commonly, e.g. job loss, insolvency, increase of off-balance utilization or lack of current financial documentation in case of corporate clients) should follow the guidelines and regulatory recommendations. In that manner, we highly suggest developing the definition of default which is consistent with the model purpose as well as further model usages (e.g. provisions calculations, debt collection).

Usually, additional recalibrations or add-ons that are performed on the original model are more cumulatively time-consuming than designing the model adequately from the very beginning. In principle, different model purposes require different model specifications – however, designing the scoring model that will also be utilized, e.g. for IFRS 9 or IRB, from the very beginning, will undoubtedly save time and efforts later-on; for instance, models used under Advanced Internal Rating Based regime require to have at least six non-default ratings and one default for corporates. In that matter, we highly recommend, along with scoring model development, the regulatory gap analysis which (when performed carefully) will support the incorporation of all regulatory requirements from the very beginning.

Moreover, in the case of the banks that possess a broader set of historical data (e.g. ten years of production data), it is advisable to perform the analysis of the recovery from various DPDs levels as well as vintage analysis (i.e. consecutive years of credit approvement). Combined results should allow to a) empirically authorize the default threshold, b) determine whether to include additional variables into the scoring model (e.g. dummy variables with different levels of DPD), c) confirm the stability of the model's assumptions across various cohorts.

In this monograph, we discuss credit scoring models that, regardless of the exact definition of the indicatory flag (target variable), allow us to model the binary output variable.

5.4 Multicollinearity problem

The phenomenon of multicollinearity of independent variables relates to multiple regression (i.e. in econometric models in which more than one explanatory variable explains one dependent variable) when there is a substantial linear relationship between individual explanatory variables.

The Gauss-Markov theorem states the conditions, the fulfillment of which guarantees that among all unbiased linear estimators, the least-squares method has the least variance of estimators (Greene, 2003). One of the conditions, apart from the linearity to parameters, random sampling, exogeneity and homoscedasticity, is the assumption that there is no perfect multicollinearity of explanatory variables, i.e. the matrix created from the values of the explanatory variables has a full order. This assumption has purely technical conditioning: in the case of perfect multicollinearity of explanatory variables (i.e. one of the explanatory variables is a deterministic linear combination of the other explanatory variables), the estimator values $\hat{\beta} = (X^T X)^{-1} X^T y$ cannot be obtained, where X is the matrix of values of the explanatory variables and y is the vector of the values of the explained variable ($X^T X$ matrix is irreversible). The perfect multicollinearity of explanatory variables, however, is an extreme case and often associated with *inadequate model specifications*, for instance including as many binary variables as the number of all categories of a variable when encoding categorical variables into binary variables (then one of the variables is a linear combination of the others). In practice, near multicollinearity problem – strong correlation of explanatory variables – is more often the case of consideration.

In the case of the logistic regression, the set of assumptions is as follows: 1) the target value is a dichotomous variable, 2) the relation between logit and predictors is linear, 3) there are no influential values, see Section 5.1,

4) the assumption that there is no perfect multicollinearity of explanatory variables. In principle, as discussed, the multicollinearity problem is in both linear and logistic regression an issue.

If the explanatory variables exhibit the near multicollinearity, the estimator values may strongly depend on the training set (i.e. the estimator values manifests a high variance) on which model parameters are determined. Near multicollinearity does not, however, affect the predictive power or reliability of the model or its whole but affects the values of individual estimators. Such a scenario means that the multiple regression model with correlated predictors as a whole possesses the predictive power but the results of individual predictors may be significantly disturbed due to the usage of the different training sets. The Gauss-Markov theorem refers to perfect multicollinearity (deterministic, linear relationships between explanatory variables). In this monograph, whenever there is a reference to multicollinearity and the consequences resulting from this phenomenon, one should understand near multicollinearity (i.e. stochastic).

Practically, the factors causing multicollinearity can be divided into the following categories:

- **the input data**; mainly due to the usage of highly linearly correlated independent variables but also poorly designed data collection method, low number of observations, a large number of explanatory variables, etc.,
- **incorrect model specifications**; among others, incorrect input of binary-coding: dummy variables used when converting categorical variables into binary variables, including as many binary variables as categories), including an explanatory variable which is the result of calculations carried out on other explanatory variables.

In the latter case, the model exhibits the perfect multicollinearity which, in principle, is easy to detect when the model's estimation. The former case is the subject of a further investigation and description presented below.

The variance of the logistic regression model's parameter is usually calculated numerically (i.e. the Fisher's information matrix) (Pruska, 2009;

Shao & Tu, 2012). To present straightforward effects of the mutlicollinearity on the model's parameter, we present the estimate for the OLS. The variance of OLS parameter's β_i, can be formulated as (O'brien, 2007):

$$\hat{\sigma}^2(\beta_i) = \frac{1}{n-k-1} \times \frac{1-R_y^2}{1-R_i^2} \times \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \tag{5.1}$$

where: n denotes the number of observations (i.e. number of rows of the X matrix), k denotes the number of model's parameters (i.e. the number of columns of X matrix), R_y^2 denotes the ratio of explained variance of y by the model, R_i^2 denotes the fraction of variance of X_i explained by the model constituted by other explanatory variables X_j, $j = 1, 2, \ldots, k$, $j \neq i$.

According to equation 5.1, the variance of the β_i parameter depends on:

- **the training set size** – n; the larger the training set, *ceteris paribus* the smaller the variance of β_i parameter,
- **number of model's parameters** – k; the more explanatory variables, *ceteris paribus*, the higher the variance of β_i parameter,
- **the amount of the variance of the dependent variable explained by the set of explanatory variables** – R_y^2; intuitively, the larger the amount of y-variance explained by the model, *ceteris paribus*, the smaller the variance of β_i parameter,
- **the amount of the variance of X_i depending upon the rest of explanatory variables** – R_i^2; the higher the variance of X_i depending upon the rest of explanatory variables, *ceteris paribus*, the higher the variance of the β_i parameter.

The first two results relate to the combined measure: degrees of freedom of the model – the higher the number of degrees of freedom, i.e. the difference between the number of observations and the number of independent variables of the model, the smaller the variance of the parameter β_i.

The last point relates to the problem of multicollinearity of explanatory variables introduced earlier; when for the particular independent variable, X_i, a large part of the variability can be related to the variability of other independent variables, the variance of the parameter of the variable X_i is inflated. As it was already indicated, the occurrence of multicollinearity in the model does not reduce the predictive power of the model (i.e. it does not introduce an estimation bias); however, it affects the ability to interpret the parameters. In the case when the variance of the variable X_i is inflated, the statistical significance of this variable decreases, to the level of the purposefulness of the exclusion of that variable from the model.

The main effect of multicollinearity is the inflation of the variance in the estimation of the model parameter which, as a result, may become statistically insignificant or change the sign of the model's parameter. The presence of highly correlated explanatory variables makes it impossible to reliably conclude about the impact of the particular explanatory variable based on the estimated value of this variable's parameter; multicollinearity does not affect the predictive power of the entire model. However, in this monograph, econometric models are used for assessing creditworthiness; from the regulatory point of view and, due to business requirements, it is necessary to have reliable estimates of the model's parameters as well as to select a set of variables that have an intuitive interpretation in the context of credit scoring. In classical approaches, one would include/exclude a particular variable in/from the model – in case of the methods described in this monograph, we allow (in some cases) to gradually increase the value of parameters along with gaining the "*material evidence*" of predictive power.

The occurrence of near multicollinearity in the case of classical models used to assess creditworthiness will result in unbiased results, while it will not be possible to reliably indicate to the client the reasons for taking individual credit decisions (i.e. acceptance or rejection of a loan application). In this sense, ensuring reliable parameter estimates (i.e. the low variance of the model's parameters), as indicated earlier, is necessary and is associated with the so-called technical robustness of scoring models.

As for the detection of multicollinearity, the literature and practice utilize two main approaches: the Variance Inflation Factor (VIF) and the Condition Index. The VIF indicates the increase of the variance of a particular explanatory variable due to its linear correlation with other independent variables. The value of the VIF for the i^th the variable is defined as:

$$VIF_i = \frac{1}{1 - R_i^2} \tag{5.2}$$

where: R_i^2 denotes the coefficient of determination for the linear regression of X_i over the rest of explanatory variables $j \in \{1, \ldots, m\} - \{i\}$. The subject literature reports different levels of VIF acceptance: from 5 (James et al., 2013) up to 10 (Hair et al., 1998). Also, some of the references Welfe (2018) suggest using the models as long as $R_y^2 > \max_i(R_i^2)$.

The *Condition Index* is the measure of multicollinearity in the data. It is calculated based on the full matrix of the normalized features – from that matrix, one calculated the eigenvalues and i-th index is calculated as: $CI_i = \left(\frac{\lambda_{max}}{\lambda_i}\right)^{\frac{1}{2}}$, where λ_i is the i-th eigenvalue of the normalized features matrix (Liao & Valliant, 2012).

In terms of interpretation, as indicated by Kennedy (Kennedy, 2003), as a rule of thumb for interpreting the value of Condition Index, one should set two limits for any of the CI_i: above 15 (may suggest the collinearity problems) and greater than 30 indicates strong collinearity. Different studies also confirm those rules, see *The IBM Knowledge Center*. In the table below, the example of the Condition Index calculation and values are presented.

Dimension	Eigenvalue	Condition Index
1	7.772	1.000
2	0.453	4.142
3	0.156	7.058
4	0.033	15.347
5	0.065	10.910
6	0.003	50.899
7	0.002	60.835

Table 5.1: Example of the Condition Index calculation
Source: own work

The above-described measures, VIF and Condition Index, are mostly referred to as the alternatives and it is recommended to use them jointly. The main difference in interpretation and usage is embedded in the formulas – please note the VIF is computed on the single variable level (e.g. the i-th variable explained by the rest of features) and the Condition Index is calculated on the model level.

For further explanation of the multicollinearity, let $X_1, X_2 \sim N(0, \Sigma)$, where Σ is the variance-covariance of X with assumed correlation ρ:

$$\Sigma = \sigma R \sigma = \begin{bmatrix} \sigma_{X_1} & 0 \\ 0 & \sigma_{X_2} \end{bmatrix} \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \begin{bmatrix} \sigma_{X_1} & 0 \\ 0 & \sigma_{X_2} \end{bmatrix} \tag{5.3}$$

Further, let us assume that $\sigma_{X_1} = 1$, $\sigma_{X_2} = 2$ and $\rho \in [0, 1)$. Let the dependent variable be defined as $y = 0.5X_1 + 0.3X_2 + \epsilon$, where $\epsilon \sim N(0, 1)$.

Figure 5.1 presents an example of the estimated values of the linear model's parameters (presented above) for a simulation instance of 500 observations (hereafter called a low sample case). Consecutive values of correlation coefficients between explanatory variables, X_1, X_2, are presented on the x-axis. As discussed earlier, the amount of correlation between explanatory variables affects parameter estimates – an increase in their standard deviation and a change in parameter value which, in extreme cases (i.e. low sample and high level of correlation between variables), can lead to a change of the estimated parameter's sign (in Figure 5.1, the variable parameter X_1 changes the sign).

Significant increase of the estimated parameters' standard deviation, along with variations of the estimated parameters (e.g. decreasing value of ($\hat{\beta}_1$) ultimately lead to a reduction of the variables' statistical significance; Figure 5.2 presents the t-statistics for both explanatory variables, accompanied by the significance test's threshold for subsequent values of correlation coefficients between variables.

The statistical significance (i.e. t-statistic) of the variable X_2 is higher for each level of correlation between variables; this can also be depicted on a more stable estimate of the model parameter value (see Figure 5.2). According to formula 5.1, as the variance of the value of the explained

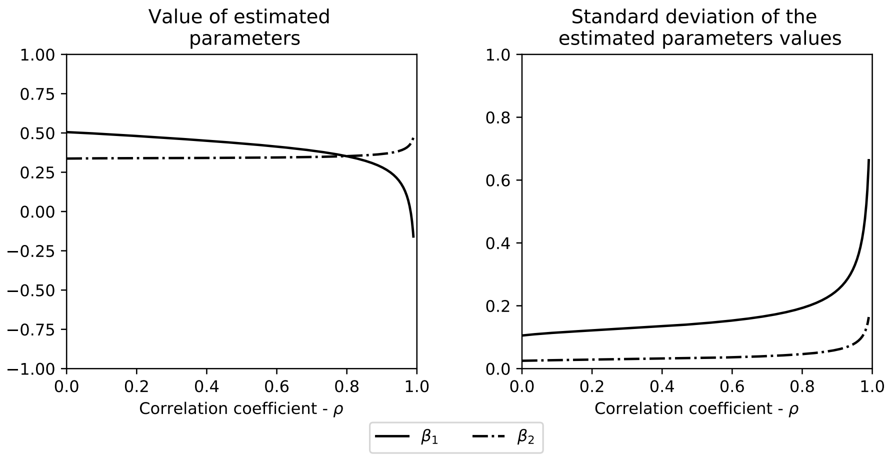


Figure 5.1: The effects of multicollinearity on parameter estimation – the case of small sample size
Source: own work

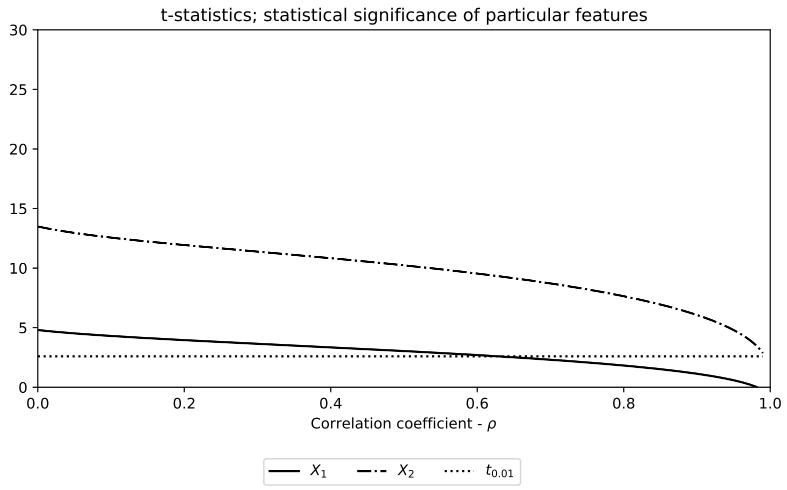


Figure 5.2: The effects of multicollinearity on statistical significance of model's exogenous variables parameter – the case of a small sample size
Source: own work

variable increases, the variance of its estimator decreases:

$$\lim_{\sigma(X_i)\to+\infty} \hat{\sigma}^2(\beta_i) = 0 \tag{5.4}$$

This means that the variables whose variability is higher are ceteris paribus more statistically significant in the presence of near multicollinearity. According to the results, the variable X_1 ceases to be statistically significant (1% threshold for the two-sided test) around the level of 50% correlation coefficient value.

The above-outlined simulation scenario has also been calculated for a bigger sample example where the number of observations equals to 700. The purpose of this analysis is to present a distinctive influence of the separate factors that shape the statistical significance of explanatory variables. Figure 5.3 shows an example of the estimated parameter values for a simulation instance of 700 observations. As in the previous example, the size of the linear correlation between explanatory variables affects parameter estimates – the increase in their standard deviation and the change in parameter value are, however, significantly less than in the case of a small-sample scenario (e.g. the parameter of the variable X_1 does not change sign).

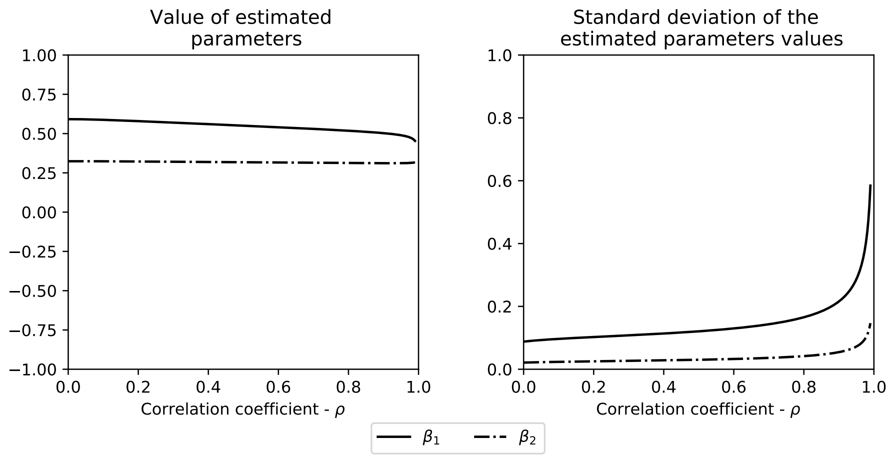


Figure 5.3: The effects of multicollinearity on parameter estimation – the case of a bigger sample size
Source: own work

A smaller change of the parameter's value, along with reduced standard deviation of the estimated parameter values, leads to an increase in the statistical significance of the model parameters; both variables are statistically significant up to the level of 82% of the linear correlation coefficient; see Figure 5.4.

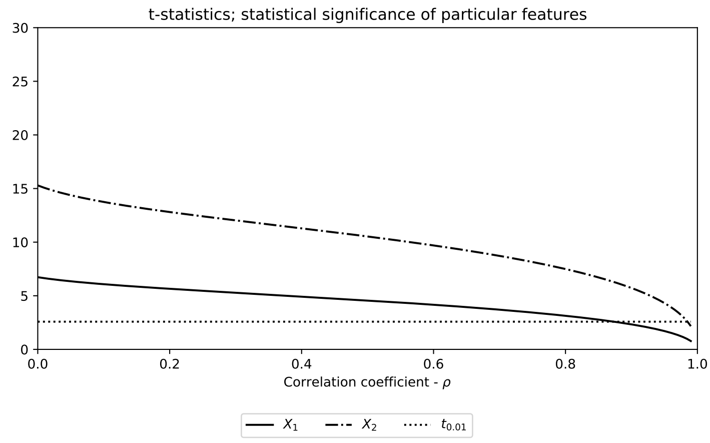


Figure 5.4: The effects of multicollinearity on the statistical significance of model's exogenous variables parameter – the case of small sample size
Source: own work

Figure 5.5 presents the Variance Inflation Factor for the richer data set. The results indicate the significance thresholds of the explained variables (t statistics) and the accompanying levels of Variance Inflation Factor (VIF) values. It should be noted that the variable X_1 ceases to be statistically significant already at the VIF level around 4.2, while the variable X_2 remains significant up to the VIF level around 33.7.

It should be noted, however, that the literature on the subject reports different values as VIF threshold levels. The most common Variance Inflation Factor values in the literature are 4, 5 or 10 (O'brien, 2007). It should be noted, however that artificial, uniform levels of the VIF should not be used; literature reports point to the need to distinguish between situations in which a variable is statistically significant or not (Belsley et al., 2005).

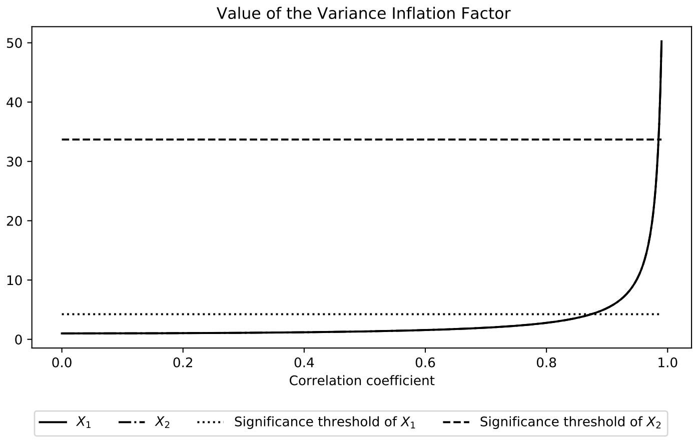


Figure 5.5: The effects of multicollinearity on the VIF parameter – the case of bigger sample size
Source: own work

5.5 The Simpson's Paradox

What happens when we *let the data speak for themselves*? Actually, in many cases, anything can happen. Depending on how the data is grouped and which variables are chosen for the model, the researcher may obtain the opposite results (Simpson, 1951). A statistical phenomenon where a trend appears in several different groups of data but disappears or reverses when these groups are combined is called a *Simpson's Paradox* and has manifested itself for decades in various scientific disciplines (Selvitella, 2017). Let us consider the paradox in two examples. First, let us assume that in the course of exploratory data analysis, the analyst has got the following contingency table:

		Default		Default Rate
		1	0	
Self-employed	Yes	39	1 992	1,92%
	No	46	2 114	**2,13%**

Table 5.2: The Default Rate distribution across job status
Source: own work

Based on such a table, one might conclude that since the default rate among salaried individuals equals 2.13% (larger default rates in the column are bolded) and only 1.92% among self-employed clients, full-time workers are riskier. Such a simple one-dimensional relationship hardly ever has enough forecasting power. In many actual problems in credit scoring or any other field, multivariate models have to be created to grasp the complexity of the data generating process. To do that, further stratification is performed and, as a result, two other contingency tables are produced. The first one contains only individuals whose income is lower than or equal to 3 500 and the second one – the remaining people.

Income $\leq$ 3500		Default		Default Rate
		1	0	
Self-employed	Yes	7	176	**3.83%**
	No	32	1 050	2.96%

Income $>$ 3500		Default		Default Rate
		1	0	
Self-employed	Yes	32	1 816	**1.73%**
	No	14	1 064	1.30%

Table 5.3: The Default Rate distribution across job status and income group
Source: own work

Contents of these tables might come as a surprise to the analyst because it turns out that even though full-time workers seemed to be riskier clients in general, after further stratifying, the trend is opposite. Among lower-income individuals, it is entrepreneurs who show a higher default rate (3.83%) compared to full-time workers (2.96%) and the same stands in the group of individuals with higher income – default rates equal 1.73% and 1.30%, respectively.

This phenomenon has an elementary mathematical explanation. Each default rate in the stratified tables shows a percentage of defaulted loans among a particular group and then they are compared regardless of sample sizes. Mismatch of sample sizes in our example is manifested by the fact that full-time working group is divided relatively evenly between lower- and higher-income groups in terms of a number of clients and not equally at all in terms of the number of bad loans in favor of the high-income group. For self-employed individuals, a different thing happened – very few clients fell into the lower-income bin and a relatively large proportion of them failed to repay their loans.

Simpson's Paradox is not a property of categorical data alone but can occur in other cases as well. Let us take a look at another toy example – a scatterplot of a continuous variable Income versus Probability of Default.

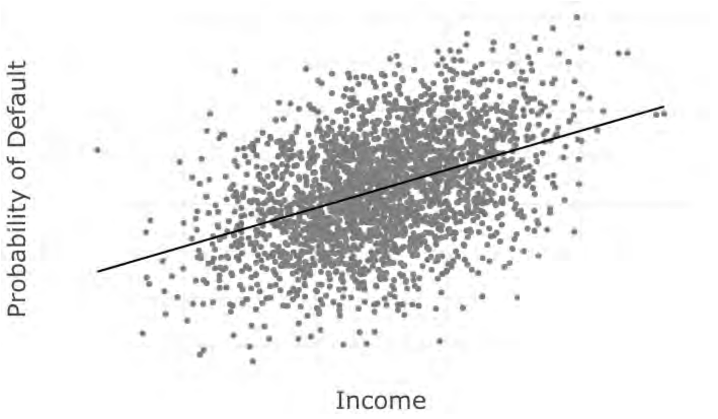


Figure 5.6: Distribution of the Probability of Default vs. Income level
Source: own work

The scatterplot, see Figure 5.6, strongly suggests a positive relationship between income and probability of a default, meaning that with an increase of the income, the probability of a default rises. Such a discovery would seem counterintuitive for any credit analyst as higher income usually tends to lead to steadier repayment of liabilities. However, after another layer is added and trends within age groups are estimated, not even one of the slopes is positive, see Figure 5.7. All relationships are either negative or neutral (non-existent) which in turn is in line with expectations.

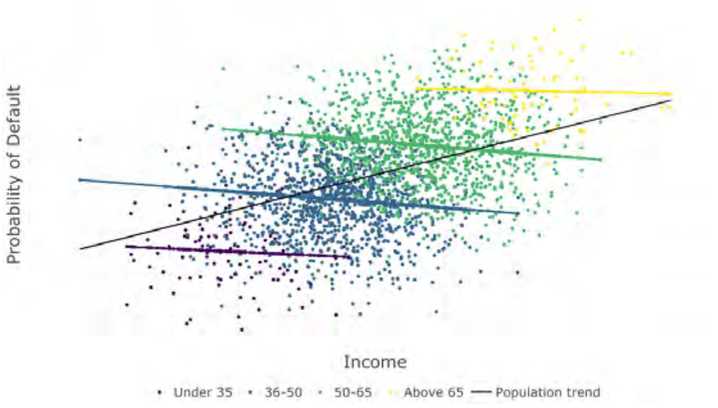


Figure 5.7: Stratified distribution of the Probability of Default vs. Income level
Source: own work

Simpson's Paradox in case of continuous data can arise either because of data being clustered into separate groups (e.g. if we only had the category of aged under 35 and above 65 from the toy example above) or because of something much more straightforward – because of an underlying stratification within this group. Both of these reasons occur quite frequently in practice.

5.5.1 What should a data analyst do when facing a Simpson's Paradox?

Let us go back to the original example with contingency tables. The question is, what is the truth in this situation? Are self-employed actually more reliable overall, even if they are riskier clients in both income groups? Unfortunately, there is no general rule that can be applied here. In this example, the same data can be used to show either the high or low risk of self-employed clients, depending on the model used. Therefore, statistics alone may not give answers here – what needs to be incorporated in the analysis, is a causal context.

Perhaps the causal model of the data looks in a way presented in the income stratified tables and the conditional relationships are true. It is also possible that the stratification shouldn't be kept by the analyst because the income threshold was chosen incorrectly, there should be more than one threshold or income does not play a part in this process at all. The number of possibilities is immense. This is why, for a human being who is testing various stratifications one by one, it would be easy to miss the correct option or believe in an incorrect one instead. Even though people are more aware of the data generating process than statistics is, machine learning algorithms can test significantly more data divisions than people can and they are unbiased while doing so. Sometimes people's theories are subjective or they are simply not advanced and deep enough. This flaw can be fixed by using methods such as decision trees or random forests which are capable of creating intricate rules beyond what could be thought of by credit analysts themselves. This is the reason why, in many fields, mathematical models eventually gained popularity over expert knowledge in the first place. Because of this, the analyst's role is not building the rules but evaluating them and relating them to reality which takes much less effort, time and resources and leads to more reliable models.

5.6 New categories in categorical variables

In practical application, one may encounter a situation when a model's categorical variables, e.g. for a new client, receive a new category. That situation, from a technical point of view, prevents using the model – one can not assign any value for that observation and hence one can no longer calculate creditworthiness. One of the required properties of *Trustworthy Artificial Intelligence* is the *technical robustness* (European Commission, 2019). In our opinion, a systematic handling of new categories embedded into the design of the scoring model is required under the technical robustness condition. As for the solutions, we recommend one of the following approaches:

1. when a new category has low cardinality, i.e. a small number of observations and the categories' catalog is fixed, it is advisable to reject this observation because it probably means a technical error,

2. for a nominal category with high cardinality and dynamic catalog, we recommend to group the low cardinality categories into one group (a so-called *other* category); then the new category, without any additional information, can be classified into this factor level,

3. if the embedding of the category space into a metric space is possible (usually R^n, with n small – 1 or 2) one can: a) estimate the model of the embedding (simplest representant is the WoE transformation) or b) estimate the model on the original values but based on the embedding, one approximates the value of new category based on the old categories (analogously to the nearest neighbor algorithm). As an example, let us consider the postal code – in that case, the usual embedding used is the one used based on the geographical coordinates (and then a) estimate model on the entire set of data or b) locally approximate the value of new category).

5.7 Complete separation problem

The *complete separation problem* (sometimes called *perfect prediction problem*) refers to a problem related to parameters estimation. It arises when one can perfectly separate binary target variables within the training set the observation, based on a few characteristics (usually 1 or 2), (Heinze & Schemper, 2002). That situation in the context of credit scoring may occur along with a low number of defaults within a portfolio or extremely correlated features with the target (probably caused by implementation errors – e.g. including target variable to features matrix). The severity of that situation results from the technical point of view – in case of complete separation, the *maximum likelihood* estimate does not exist. The abovementioned problem can also be generalized to a *quasicomplete separation* – a situation when some variable almost perfectly separates the observations. Let us consider the example presented in Table 5.4.

In the presented example, one tries to generate a model that predicts the default event based on two variables: Income and Age. As it appears, the Income variable is an almost complete predictor of the default event;

Default	Income	Age
0	3500+	35+
0	3500+	<20
0	3500+	35+
0	1500-3500	20-35
0	1500-3500	<20
0	<1500	35+
1	<1500	<20
1	<1500	<20
1	<1500	20-35
1	<1500	35+

Table 5.4: Example of the complete separation problem
Source: own work

if we calculate default rates within each of the categories of the Income Variable, then we may conclude that all of the defaults observed happened for the customers within the group of income lower than 1 500. Only one customer that has an income of lowest that 1 500 has not defaulted; hence this example presents the quasi-complete separation (if that customer fell into a different category of income, this would be the complete separation problem case). In that situation, the maximum likelihood estimates do not exist.

From a technical point of view, the `sklearn` function for logistic regression, as default, performs regularization – even in the case of complete separation, the loss function is not equal to zero and the maximum likelihood estimator exists. The first step, when complete separation is encountered, is to revise whether the implementation is correct or another variable linked to the output variable by definition is not included in the model itself. One of the possible actions to take is to exclude the variable that perfectly separates defaults; however, that variable may be simply the only adequate variable to include in the model (and also, this technique leads to a biased estimation of other predictors). On the other hand, in the case of complete separation, the maximum likelihood for different features is still valid – only the exact separator has not reasonable estimates. Another technique that may be useful is to merge some of the categories.

In practice, the complete or quasicomplete separation causes a significant estimation problem only in a logistic regression case – also, as in the case of WoE transformation, the complete separation problem may be minimalized by performing appropriate and careful binning (as indicated earlier). In the case of decision trees, random forest or XGBoost, only in case of one variable, the estimation procedure may indicate perfect fit; in the case of multivariate modeling, from the implementation point of view, the minimal number of observations (called minimum leaf size, see 4.2.4) is set so that the perfect fit may be prevented. In terms of regularization techniques, the penalty term is always explicitly present in the models, so from a technical point of view, a complete separation should not cause any estimation problems (as indicated earlier). As for the practical implementation of relu, it is usually the case that the model includes some regularization.

5.8 Too granular categorical data

Too granular categorical data do not help in improving the power of models. Some implementations of machine learning algorithms even limit the number of categories each variable can have. Therefore, less populated and similar categories should be merged, which requires considerable work effort as it is hardly possible to automate. The typical data with too many categories are codes of economic activity, transaction codes, product codes in internal data or in credit bureau, zip codes. The steps are the following:

- Calculate the number of good and bad observations by category
- Identify categories with similar meaning (e.g. different types of guaranteed credit lines)
- Identify categories with similar good/bad odds
- Merge categories with similar good/bad odds and similar meaning
- Out of whatever is left, merge tiny categories (with only a few goods and bads)

There is another method utilizing data from the statistical office which can help with merging zip codes. Zip codes can be mapped to municipalities and data about municipalities such as the unemployment rate, population density, population changes can be obtained. Then these municipality data (e.g. local unemployment rate) can be used in the model instead of zip codes.

5.9 Coding ratios

Ratio variables come from behavioral data and financial statements. Ratios may have missing values, missing numerators or denominators or zero denominators. The signs of numerators and denominators may also vary. Each of these cases may mean something different in terms of clients' behavior. E.g. take a ratio of customer's balances on all accounts to the same balances a year ago. Assume that balances on current accounts and savings are positive and balances on credit accounts are negative. If the client has no balances with us, we will get a division zero by zero. If the client is new and he/she had no balances a year ago, we get a division by zero. When the client had only deposits and increased them, we will get a positive ratio. Finally, when the client had loans and took a new one, we will again get a positive ratio. Therefore:

- It is better to create separate variables depending on whether the denominator is positive or negative

- Coding missing numerator and missing denominator as separate categories (separate artificial values or just a mean value and a corresponding dummy variable).

- Coding zero denominators as a yet separate category (another separate artificial value or just a mean value and a corresponding dummy variable).

5.10 Conclusions

This chapter presents common issues related to the data used in the process of building predictive models. This discussion's primary purpose is to present the critical issues in the data that every expert building a scoring model should pay attention to. These issues may have a significant impact on the quality and robustness of the supporting decision-making models. Linking this chapter with the considerations made in Chapter 4, we indicate that machine learning methods, unlike classic scoring methods, are less sensitive (i.e., more resistant) to data-related issues (e.g., collinearity, data occurrence gaps). At the same time, we emphasize that the issues described in this chapter are not specific to the area of credit scoring. These universal issues are related to data-quality areas - when building any predictive model, the constructor should consider the risks associated with artifacts occurring in the data.

Chapter 6

Model performance evaluation and model monitoring

Daniel Kaszyński
Małgorzata Wrzosek
Kamil Cerazy

Decision Analysis and Support Unit
Collegium of Economic Analysis
SGH Warsaw School of Economics

The scoring model validation process is aimed at verifying the correctness of this model. However, the correct operation of the model has many aspects and depends on various factors. The scope of the validation process depends on the stage in the model life cycle. The most extensive scope of analyses is related to the process of building and implementing the model. The full recurring validation performed at a fixed frequency also has a wide scope. The narrowest scope is that of the model monitoring carried out systematically, e.g. on a quarterly basis.

An important element of a model validation during its construction or modification is the verification of its correctness on a developed data set. It includes an assessment of whether the relationships in the model are logical and consistent with intuition, the observed impact of individual variables on the risk and the predictive power of the variables. These elements result from business knowledge and data analysis performed before modeling. Such an analysis should include dependencies of the dependent variable on individual explanatory variables, monotonicity and stability of these relationships, as well as a study of the predictive power of variables.

In the classic approach to building the model, stable variables, mutually uncorrelated, with sufficiently high predictive power and with a low VIF value were selected. In non-classical machine learning methods, the predictive power of single variables and lack of correlations between them are not so important. By using regularization methods or algorithms such as decision trees and random forests, some of the requirements for variables in the model can be omitted. In machine learning, models can achieve high performance by integrating many correlated variables with low individual predictive ability.

6.1 The importance of validation and monitoring

A scoring model that has been properly developed, i.e. taking into account the standards and issues described above and in the other chapters, should then be subject to extensive pre-implementation validation. This validation should cover all important elements related to the construction of the model, including:

- quality of input data – the data for building the model must cover an appropriate scope of information and be of appropriate quality,
- the quality of the model itself – the model should have appropriate discriminatory power and stability as well as calibration accuracy,
- the correct direction and strength of the influence of individual variables on the operation of the model and its predictive power,

- the results' quality – evaluation of the model's performance in terms of the quality of results generated by this model.

6.1.1 Validation in the model life cycle

The results of pre-implementation validation of scoring models determine the model's admission to the production environment. After this stage, a post-implementation validation should also be carried out. This includes the correctness of the technical performance and the integrity of the processes and allows one to verify that the model works as expected.

Pre- and post-implementation validations occur at the beginning of the model's lifecycle. Subsequent periodic validations and cyclical monitoring (usually quarterly) are used to assess the correctness of the model operation but also to verify the cyclical nature of the modelled relationship. The main purpose of validation is to verify the stability of the model's predictive power over time – because the portfolio, economic environment, bank strategy, risk appetite or other influencing factors may have changed since the development phase. However, periodic validation also includes other elements, not only subject to quantitative but also qualitative assessment. On the other hand, monitoring – usually quarterly – is aimed at generating reports on the operation of the model, including a simplified assessment, which allows one to monitor the operation of the model and the portfolio behavior on an ongoing basis. The catalog of analyses carried out as part of the above-mentioned validation processes will be slightly different in each case.

The validation of the credit scoring model is a mandatory process for banks, required and controlled by regulators. Moreover, credit scoring models must be accompanied by a methodology for validating those models (Polish Financial Supervision Authority, 2015). Such requirements, while maintaining standards related to the independence of the validation unit, are aimed at maintaining transparency and standardizing of the validation process.

Even the ancients wondered *"Quis custodiet ipsos custodes?"*[1]; in the case of banks and their models, a similar problem arises – how to ensure appropriate supervision over the correctness of processes. In this case, the solution to the problem is to adopt appropriate standards and procedures. After the development and implementation of a new model for credit scoring, the Bank also adopts and approves standards for the validation of a such model. Also, regulatory requirements related to the independence of the validation unit are superimposed, i.e. the administrative and functional direct subordination of the validation unit to the management board member responsible for the supervision of the model risk management area. Such institutionalization of the validation process and standardization of its scope, in the absence of other distortions related to the independence of the validation unit, allow constructive and substantive assessment of the models.

In this chapter, a description of the process and methods of diagnosing the operation of the creditworthiness assessment model is provided, which is carried out as part of the validation and monitoring processes. It should be remembered that the catalog of qualitative and quantitative techniques listed below is not complete and each process of development of a creditworthiness assessment model should include reflection and further investigation in the areas of requirements (i.e. *must have*) and recommendations (so-called *should have, nice to have*) relating to the scope of the model itself.

6.1.2 Model purpose

Before deep-diving into the validation process, methods, measures etc. it is an imperative to discuss the purpose of credit scoring models. The purpose dictates appropriate validation and monitoring process and relevant measures to verify desired model's properties. Let us start with the fundamentals – the business purpose.

Banks use credit scoring models for approving and pricing loans, estimating provisions, setting credit risk limits and managing overall credit

[1]Own translation: *Who will guard the guardians themselves?*

risk exposure. To achieve this, models should differentiate between borrowers with better and worse creditworthiness which translates into a lesser or greater chance of defaulting on the loan. However, depending on the realized aim, whether it is to decide if a particular customer should or not get a loan or assess risk of the credit portfolio of a bank, different analysis horizons apply which in turn demand different sets of data both in terms of history length and types of parameters.

For credit risk management, the model should take into account parameters related to business cycles over many years. In the case of a decision on a single loan, the model is based on a limited set and history of data and should decide whether, compared to similar clients, the risk of default is acceptable by the bank and whether the loan can be granted. Both issues relate to PD modeling. Howeve,r the first one is called the *Through-The-Cycle* (TTC) approach and the second is called *Point-In-Time* (PIT).

Once there was a need to set some overarching rules on the credit risk management, regulatory bodies consulted the major financial institutions to determine market practice. Based on different approaches, the consensus was reached and the concepts realizing so far business purpose trickled into regulatory requirements and incentives. A distinction between TTC and PIT approach is reflected in requirements for calculation of regulatory capital and the estimation of accounting provisions stemming from recently introduced International Financial Reporting Standard 9 (IFRS 9). Applying the same model in both cases is not easy because regulatory expectations differ for the models used for estimating regulatory capital and in accounting. In practice, however, one model and two different calibrations are used for the TTC and PIT approaches. The PIT model may also contain factors common to the entire population (macroeconomic variables), although they do not differentiate between good and bad clients. Therefore, they do not affect the scoring result but they do affect the estimated PD level.

Regulatory capital was put in place to ensure prudent management of banks and other financial institutions, i.e. they wouldn't take excessive leverage and risk becoming insolvent. This means holding enough capital against taken risk levels protects the company, customers, government and the economy as usually financial institutions are systemically important parts of it. Rules around capital requirements have been established in the form of Basel Accords published by the Basel Committee on Banking Supervision (BCBS) translated later into local laws. For smaller, less sophisticated institutions, there is an approach based on measures and coefficients proposed by the regulator, whereas more advanced institutions can opt for an internal rating-based approach of calculating capital requirements for credit risk. Following the purpose of regulatory capital being a buffer for resilience against changing economic conditions, Basel models are measured on the TTC basis to take into account the economy cycle.

IFRS 9 came into force on the 1st of January 2018 and changed accounting for credit risk. Beforehand, credit loss was recognized after a 90-days overdue period, under IFRS 9 framework *Expected Credit Loss* (ECL) is calculated at the origination of the financial instrument and later updated at reporting dates using forward-looking information. Moreover, assets are classified into different stages based on a change in the credit risk since the initial recognition. Such a change in risk can be measured by PD, CPD (*Cumulative PD*) or MPD (*Marginal PD*) values. This *snapshot* view of the current state based on expectations on the future is the above-mentioned PIT approach. In the case of model validation, the difference between PIT and TTC matters only when backtesting and assessing calibration, while the approach used does not affect the assessment of the quality of the scoring model.

6.1.3 Regulatory incentives

In the field of subject literature, this chapter is based on:

- Recommendation W on model risk management in banks issued by the Polish Financial Supervision Authority (Polish Financial Supervision Authority, 2015).
- Instructions for reporting the validation results of internal models issued by the European Central Bank (European Central Bank, 2019)

- Guidelines on PD estimation, LGD estimation and the treatment of defaulted exposures issued by the European Banking Authority (European Banking Authority, 2017a).
- Studies on the Validation of Internal Rating Systems issued by the Basel Committee on Banking Supervision, Bank For International Settlements (2005b).

The above-mentioned selected guidelines provide a regulatory perspective on issues related to the validation of creditworthiness assessment models – they have guidelines in both qualitative and quantitative validation techniques.

It should be noted that these regulations have different areas of application. The EBA is a supervisory institution whose regulations apply throughout the European Union. The ECB is the central bank for the euro area and its recommendations are binding in this area. Naturally, the recommendations of the Polish Financial Supervision Authority are binding for financial institutions operating in Poland. The Basel Committee on Banking Supervision (BCBS) brings together representatives of Central Banks and financial regulators from 27 countries around the world. However, BCBS is not a supervisory institution and its guidelines are not binding regulations but recommendations regarding good banking practices.

6.2 Validation and monitoring process

The validation of the credit scoring model is, in general, a process of assessment of the effectiveness of this model. It is carried out by an independent bank unit (model validation unit) in a comprehensive manner. The scope of validation of the credit scoring model should include assessment of the model's operation effectiveness, review of the methodological concept appropriateness, assumptions, correctness of its construction as well as its implementation method (Polish Financial Supervision Authority, 2015).

Cyclical monitoring of the credit scoring model is usually performed by the bank unit responsible for the construction, use and functioning of the model (the so-called *model owner* or *internal unit*), using statistical measures and numerical indicators (Polish Financial Supervision Authority, 2015). Monitoring of the credit scoring model is narrower in terms of the scope of evaluation than the validation process. It is mainly an assessment of the effectiveness of the model's operation, carried out mostly in the area of analysis of results generated by the model, compared to empirical data on the observed delinquency rate (for retail borrowers) or observed default rate (for corporates) of the loan portfolio for which the monitored model has been prepared (i.e. backtesting). At the same time, monitoring, thanks to its regularity and frequency, allows capturing any trends in changes in the portfolio or risk measures.

As for supervisory expectations, the process of assessing the effectiveness of the credit scoring model should consist of the following types of analyses:

- pre-implementation validation – performed after the development process but before implementation, its aim is to verify input data quality, statistical soundness of the model and quality of the initial results and answer whether the model should be implemented,

- post-implementation validation – associated with the implementation process and subsequent launch of the model in a production environment; during the first three months since the date of launch the correctness of model implementation should be assessed and the final decision for using the results of this model should be made,

- periodic monitoring – carried out at least once a year and in the case of data availability, it should be carried out at a quarterly or semi-annual frequency in usual market condition or even monthly when in a period of uncertainty and stress economic conditions or in few first months of operating after implementation or significant updates,

- periodic validation – performed usually once a year/once every two years depending on the significance and materiality of the model, it should be an external process (not conducted by the modeling unit) of assessing the quality of the operation of models functioning and used in the bank (also in terms of the model used to assess creditworthiness). The objectives of the model validation are verification of the correctness and effectiveness of credit scoring models, diagnosis of areas and issues related to the functioning of the model that should be improved, formulation and prioritization of recommendations related to the functioning of the model (which are elements of every validation), verification of the implementation of previous recommendations and promotion of high management standards (Polish Financial Supervision Authority, 2015),

- ad hoc validation – performed in response to identification (e.g. as part of model monitoring) of deterioration in model performance; the scope of ad hoc validation should always be defined, taking into account the issues identified; examples of analyses performed as part of ad hoc validation are verification of the adequacy of model assumptions, analysis of the distribution of variables supplying the model and identification of differences concerning the data sets on which the model was taught, comparing the results of the validated model to the results generated by the comparative model.

The scope of analyses carried out as part of the processes described above is presented in Table 6.1.

Type of evaluation	Internal unit	Frequency	Qualitative review	Quantitative review	Review of changes to the model
pre-implementation validation	✗	before implementation	✓	✓	✗
post-implementation validation	✗	after implementation	✓	✓	✗
periodic monitoring	✓	monthly / quarterly / semi-annually	✗	✓	✗
periodic validation	✗	at least once a year	✓	✓	✓
ad hoc validation	✗	depending on the identified needs			

Table 6.1: Analyses carried out over the life cycle of the model
Source: own work

As part of the periodic validation, an overall assessment of the credit scoring model should be prepared with the following scale (European Central Bank, 2019):

1. Adequate with no deficiencies – no deficiencies detected by the validation function (i.e. no follow-up needed).
2. Adequate with minor deficiencies – minor deficiencies detected that do not lead to any significant bias for risk estimates.
3. Major deficiencies identified – identified deficiencies indicate a significant bias for risk parameter estimates, e.g. for IRB models a potential quantitative impact on the risk-weighted exposure amount (RWEA) equal to or above +/- 5% but below +/- 10%.
4. Severe deficiencies identified – identified deficiencies indicate a severe bias for risk parameter estimates, e.g. for IRB models a potential quantitative impact on the RWEA equal to or above +/- 10%.

It is common practice to communicate validation results in a traffic-light approach. For the area of qualitative assessment, the following can be assumed:

- red traffic light – identified severe deficiencies in the model,
- yellow traffic light – identified medium-weight deficiencies of the model,
- green traffic light – no deficiencies identified or deficiencies identified are of a low significance.

Currently, the assessment of the model operation and its errors are often characterized by the model's materiality. This indicator may show a quantitative impact of the model on the company's operation. The materiality can also show the impact of the model errors on the institution's qualitative assessment, e.g. on its external rating. Materiality assessment in qualitative categories is often imprecise, it also uses expert knowledge and is therefore usually characterized by a descriptive scale, e.g. critical, high, medium, low.

6.3 Qualitative methods for credit scoring models validation

The first discussed area of the analysis is a qualitative assessment, which is related to the verification of the underlying assumptions of credit scoring models. It concerns checking whether the model is used in a correct and adequate way and whether the methodological assumptions that accompany each quantitative model are met (Engelmann & Rauhmeier, 2006). In this context, the fulfillment of qualitative assessment criteria can be seen as a prerequisite for undertaking a quantitative assessment – if significant inaccuracies related to the design or functioning of the model are identified, the quantitative analysis may provide unreliable results, e.g. indicate the erroneous, seemingly high quality of the results. The credit scoring model should be used only if it passes both the qualitative and quantitative assessment. Qualitative and quantitative analysis in this aspect are complementary to each other and none of them individually constitutes a premise or criterion for a sufficient assessment of the operation of the credit scoring model. Elements that are analyzed as a part of the qualitative assessment are (Engelmann & Rauhmeier, 2006):

- model design, carried out based on model documentation, aimed at assessing its scope, transparency and completeness;
- data quality, including its completeness and correctness;
- internal use test;
- rating process.

6.3.1 Model design

The assessment of the methodological assumptions of the credit scoring model is based on a model documentation analysis; in this sense, the scope, transparency and completeness are essential criteria. It is focused on assessing the adequacy of the adopted architecture and model assumptions, their compliance with the current environment (including regulatory, economic).

Model documentation should cover the following areas:

- delineation criteria for the portfolio or rating segment[2],
- whether expert knowledge was used and to what extent,
- description of the rating method/model type/model architecture used,
- reason for selecting a specific model type,
- completeness of the (best practice) criteria used in the model,
- data set used in rating development[3], data sources, quality control and archiving mechanisms,
- quality assurance for the data set,
- model development procedure:
  - model architecture and fundamental business assumptions,
  - selection and assessment of model parameters[4],
  - analyses for model development,
- quality assurance/validation during model development,
- description of all model functions,
- calibration of model output to default probabilities,
- procedure for validation/regular review,
- description of the rating process,
- duties and responsibilities concerning the rating model.

[2]This element covers the characteristics of the target population, including all data selection methods and rules for excluding data from analysis.

[3]In addition to the portfolio characteristics, variable selection resulting from technical constraints (e.g. too many missing data) or substantive reasons should also be taken into account.

[4]The meaning of model parameters can be interpreted in different ways. They can be understood literally when using parametric models, such as logistic regression. However, in the case of machine learning algorithms, this concept means learning parameters and algorithm settings.

Besides, we recommend considering extending the scope of the documentation by:

- description of the process and policies related to maintaining the credit scoring model – market practice suggests attaching detailed information on model maintenance to the model documentation (including criteria for recalibration or updating source data dictionaries),

- description of the procedures related to data quality management – if the data quality management methodology is effectively covering all the models, the documentation may be limited to a description of selected elements specific to a particular credit scoring model; data quality area should, in principle, include at least a description of the main characteristics of individual variables used in the model, i.e. max/min/median/dominant/mean/standard deviation/missing data/outliers; also, it is worth to include analyses in the field of logical data coherence (e.g. negative income, client aged 18 with higher education, pensioner aged 21),

- definition of the explained variable – as a rule, the definition of default used in individual financial institutions may vary, we recommend including a model definition of the explained variable (i.e. a bad customer) in the documentation along with the quantitative criteria for classification into the default category; besides, it is worth to indicate the results of a quantitative analysis that stood behind the adoption of the default definition (e.g. a negligible part of the exposure with a delay of over 90 days does not return to regular repayment); these criteria should, in principle, be sensitive enough to capture the risk relatively quickly but at the same time they have to make sure that estimation is accurate enough – for this purpose, one can use the Cure Rate curve analysis (percentage of exposures returning to regular repayment) for various baskets DPD delays (i.e. 30, 30-60, 60-90, 90+),

- description of the process of reporting the credit scoring model results,

- compliance with internal and supervisory requirements.

6.3.2 Data quality

The data quality assessment should cover both technical and substantive aspects. On the one hand, it is important that the data is complete, consistent and reliable. On the other hand, proper validation or archiving mechanisms are important. These issues are thoroughly discussed in Chapters 2 and 3 dealing with data used to build scoring models.

6.3.3 Internal use test

The internal use test aims to assess how the model is used in business processes, including for example risk management, internal capital allocation, management information. Due to the scope of this monograph, which is limited to scoring models, it is important to highlight the Basel Committee's recommendations for models and model-derived ratings. The recommendation in this area emphasizes the need to understand the ratings received and to use them effectively.

6.3.4 Rating process

Besides, it is worth noting that, as part of the qualitative assessment, information may be collected about the size of overwritten ratings (i.e. overrides), the occurrence of technical defaults, statistics on the occurrence of outdated ratings or deficiencies in customer documentation (e.g. financial statements), transferring ratings, data exclusions, average PD and DR parameter levels for excluded exposures, portfolio size (in terms of the number of clients and gross exposure) covered by the creditworthiness assessment model.

6.4 Quantitative methods for credit scoring models validation

Quantitative assessment of credit scoring models involves the use of appropriate procedures and measures, most of which are based on a comparison of model forecast and historical customer data. The issue of quantitative assessment of creditworthiness models covers three main areas:

- discriminatory power – assessment of the model's ability to distinguish bad and good customers,
- predictive power (or calibration) – assessment of the model's properties enabling estimation of the probability of entering the default state by client[5],
- stability – assessment of the stability of the model's operation over time.

The investigation of discriminatory power is the same, regardless of whether we identify customer groups as bad/good or default/non-default. Therefore, for simplicity, we will adopt a uniform designation of the quantities used in both cases. For the purposes of this chapter, the following notation will be used: let $PD_{i,t}$ denote the unobservable, actual probability of customer's default in period t, e.g. in the IRB regime the PD parameter is modeled over one year; let $s_{i,t}$ denote the awarded score for the i-th client in the period t (i.e. scoring); Let $\widehat{PD}_{i,t}$ denote the probability of entering the default state (or bad group) of the i-th customer in the period t determined by the creditworthiness assessment model. The general rule regarding creditworthiness assessment models is that the higher the credit score and thus the classification into the lower rating class, the smaller the proportion of exposures that will receive default status in the period t, and the higher the proportion of exposures that will remain non-default during period t.

6.4.1 Discriminatory power

Based on the notation defined above, the model's discriminatory power will be called model property related to the ability to distinguish between good and bad (or default and non-default) customer groups. The model distinguishes these groups correctly if for $PD_{1,t} > PD_{2,t} > \cdots > PD_{n,t}$,

[5]It is important to maintain the correct monotonicity of probabilities for the risk classes. However, the value of PD in classes and in the total population may fluctuate depending on macroeconomic factors.

the credit score points obtained from the model meet the relationships $s_{1,t} < s_{2,t} < \cdots < s_{n,t}$. As indicated above, the values of $PD_{i,t}$ are not observable, only the fact of entering the default state is observed. It is worth mentioning that the distribution of the $PD_{(i,t)}$ is not taken into account here. This means that testing discriminatory power does not answer the question whether the model is well calibrated, i.e. whether the forecast meets the condition $\widehat{PD}_{(i,t)} = E[PD_{(i,t)}]$.

Confusion matrix

The confusion matrix is a tool for assessing the correctness of the model forecast. It shows a comparison of objects' classes (good/bad) predicted by the model with their actual values. In the case where the model's forecast has a continuous character, e.g. indicates the probability of default or score, the classification of the model depends on the cut-off point. In this situation, we get different confusion matrices for different cut-off points.

The confusion matrix is a matrix with dimensions $N \times N$, where N is the number of states predicted by the model. In the case of creditworthiness assessment models, the response variable is a binary variable indicating the future state of the client – default or non-default, hence $N = 2$. When forecasting a binary variable based on a model, four situations can occur:

- the default forecast is in accord with the future status, i.e. the customer has defaulted: True Positive (TP);
- the default forecast is inconsistent with the future status, i.e. the customer has not defaulted: False Positive (FP);
- the non-default forecast is in accord with the future status, i.e. the client has not defaulted: True Negative (TN);
- the non-default forecast does not match the future status, i.e. the client has defaulted: False Negative (FN).

The confusion matrix for a binary classifier, such as the credit scoring model, takes the form presented in Table 6.2.

		Model prediction	
		Default	Non-default
Actual	Y=1 (default)	True Positive	False Negative (type II error)
	Y=0 (non-default)	False Positive (type I error)	True Negative

Table 6.2: Confusion matrix for a binary classifier
Source: own work

Particular values of the confusion matrix are determined by counting the observations in each of the four indicated groups. For example, consider the logistic regression model estimated for one exogenous variable with 200 clients. Let the threshold for the definition of a bad customer (i.e. one that defaults according to the model forecast) be 50% (i.e. if the forecast $\widehat{PD}_i > 50\%$, it is estimated that $Y_i = 1$). Figure 6.1 illustrates this example.

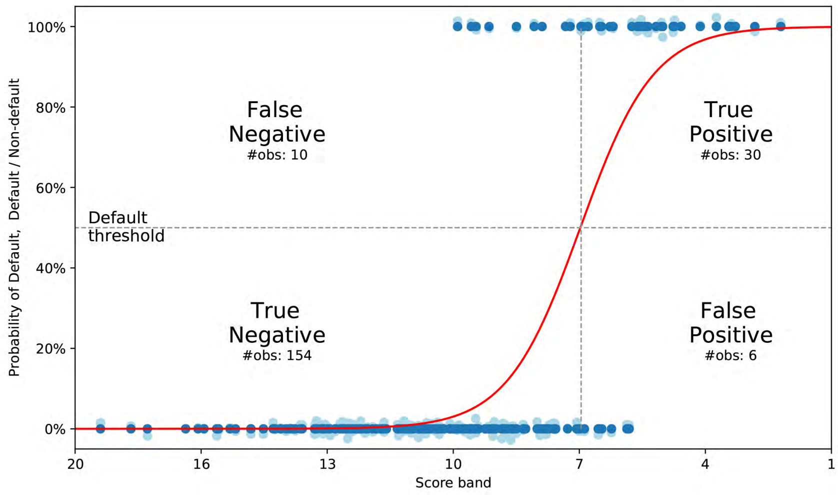


Figure 6.1: Confusion matrix; the threshold of the default definition – 50%, rejected ratings are below 7
Source: own work

In TP and TN cases, the model made the correct predictions – the forecast generated by the model is consistent with the client's state. There were 30 and 154 such observations, respectively. In FP and FN cases (6 and

10 observations, respectively) the model forecast was not correct, so the confusion matrix in this case has the form shown in Table 6.3.

		Model prediction	
		Default	Non-default
Actual	Y=1 (default)	30	10
	Y=0 (non-default)	6	154

Table 6.3: Confusion matrix for an example model with the threshold 50%
Source: own work

From a business point of view, the model error associated with False Positive and the False Negative error are different. In the case of the first error, False Positive (type I error), the model predicts that the client will enter the default state, which is counterfactual. In such a situation, the customer will have a high probability of entering the default state implied by the model. The bank has then two options: a) to grant the customer a loan with a high-risk margin or with additional collateral or b) not to grant a loan. This means that in the case of FP, the bank either imposes high credit costs on the customer (to compensate for potential credit loss) or simply does not grant a loan and therefore does not generate revenues on this account. It can be pointed out that the FP error is associated with the so-called missed opportunity cost.

In the case of the second error, False Negative (type II error), the model predicts that the client will not enter the default state but in reality the entry into this state is observed. From the bank's business point of view, this means incurring a credit cost associated with the need to create credit risk provisions. From an accounting perspective, this is a decrease in the asset receivable item, which is accompanied by incurring bad debt expense, which means a negative impact on the income statement.

Depending on the bank's risk appetite, i.e. the bank's maximum acceptable exposure level and associated risk level, the bank may set credit acceptance thresholds at different values. In the case of a liberal strategy, the cut-off point will be high (e.g. 75%) which leads to lower False Positive values and to greater False Negative values (i.e. lower missed opportunity cost at the expense of increasing the potential loss ratio of the portfolio). An example of such a situation is shown in Figure 6.2.

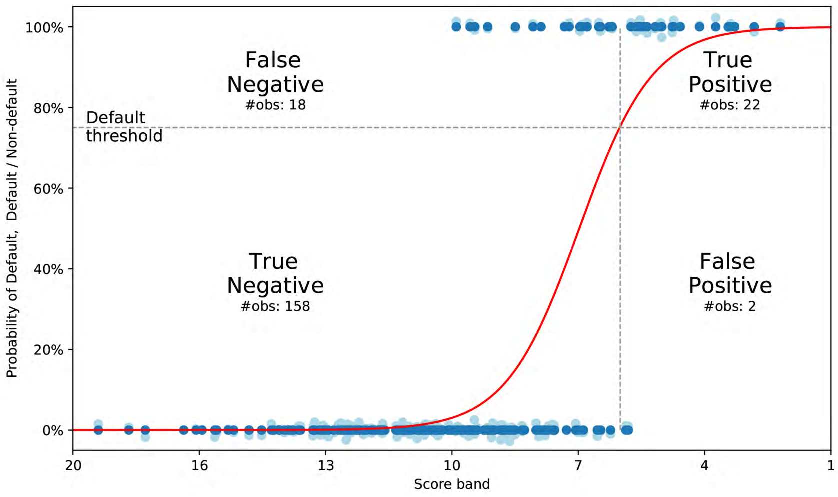


Figure 6.2: Confusion matrix; the threshold of the default's definition – 75%, rejected ratings are below 5
Source: own work

In the case of a conservative strategy, the threshold will be low (e.g. 25%), which will reduce the value of the False Negative error and increase the value of the False Positive error (i.e. reduce the portfolio loss ratio, at the expense of increasing the missed opportunity cost). This situation is presented in Figure 6.3.

Based on the data from the confusion matrix, many other numerical indicators can be determined that further characterize the quality of the model. A few of them, most often used, are defined below (the numbers are taken from the example with a cut-off of 50%):

- Precision which indicates the frequency of correct default forecasts:

$$Precision = \frac{TP}{TP + FP} = \frac{30}{30 + 6} = 83.3\%$$

- Sensitivity or True Positive Rate (TPR) or recall, indicating the frequency of the default forecast when the customer has defaulted,

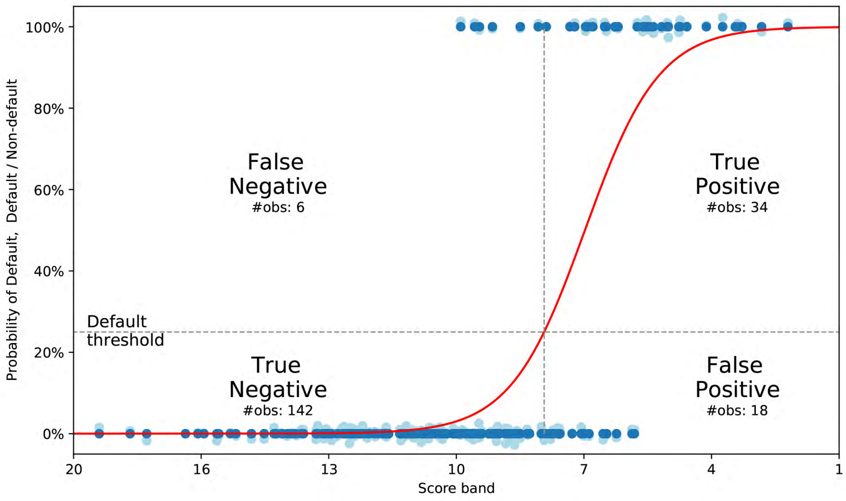


Figure 6.3: Confusion matrix; the threshold of the default's definition – 25%, rejected ratings are below 8
Source: own work

means the ability to correctly indicate a positive class:

$$Sensitivity = \frac{TP}{TP + FN} = \frac{30}{30 + 10} = 75.0\%$$

- Specificity or True Negative Rate (TNR), indicating the frequency of the non-default forecast when the client has not defaulted, means the ability to correctly indicate a negative class:

$$Specificity = \frac{TN}{TN + FP} = \frac{154}{154 + 6} = 96.3\%$$

- Accuracy, which means the proportion of observations which have been correctly classified:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{30 + 154}{30 + 154 + 6 + 10} = 92.0\%$$

- Lift shows how many times more effectively the obtained model indicates a positive class (bad/default) than the random model:

$$Lift = \frac{Precision}{(TP + FP)/(TP + TN + FP + FN)} = \frac{83.3\%}{36/200} = 4.63$$

- F-score, taking values from 0 (random model) to 1 (perfect classification model):

$$F{-}score = 2\times\frac{Sensitivity \times Precision}{Sensitivity + Precision} = 2\times\frac{75.0\% \times 83.3\%}{75.0\% + 83.3\%} = 78.9\%$$

The statistics indicated above illustrate the situation in a static dimension. All these numerical indicators are determined at a given cut-off point. Changing the threshold will change the confusion matrix and thus also the values of the listed indicators.

The listed indicators determined on the basis of the confusion matrix have different properties and emphasize different features of the classifier. It is important to mention the essential feature of TPR and TNR indicators. They characterize the classifier's ability to correctly indicate observations from the positive and negative classes, respectively, regardless of the share of these classes in the data set. Thus, they are good indicators of the effectiveness of the model even for portfolios with low or unstable default rate levels.

Receiver Operating Characteristic curve

The ROC curve presents the relationship between Sensitivity and 1-Specificity values or otherwise the relationship between True Positive Rate and False Positive Rate (FPR). The FPR value was not previously defined, along with other statistics for the confusion matrix. The False Positive Rate means the fraction of all non-default observations which were classified as default:

$$FPR = \frac{FP}{TN + FP}$$

Therefore, there is equality

$$FPR = 1 - Specificity$$

and we can notice that $1 - Specificity$ is a measure of the size of *mistaken* classification of non-default observations which have been assigned to the default class. In the ROC figure, the horizontal axis means $1 - Specificity$ and the vertical axis means *Sensitivity*. The entire curve is included in the area $[0,1] \times [0,1]$ and it connects the points $(0,0)$ and $(1,1)$. The straight line connecting these points, i.e. the diagonal of the rectangle, means the random classifier, while the line running along the sides of the rectangle through the point $(0,1)$ means the ideal classifier. A typical ROC curve runs between the diagonal (random model) and the edge of the rectangle (ideal model), wherein the higher and closer to the point $(0,1)$ the curve lies, the better the classifier is.

The determined confusion matrix sets only one point on the ROC curve. The entire curve is obtained by calculation of the TPR and FPR values for all possible cut-off points. To determine the ROC curve, we sort all observations in descending order of risk of a bad class (e.g. by probability). Then – going from the top – for each possible thresholds the TPR and 1-FPR values are determined as the coordinates of the next point located on the ROC curve. In the case of rating models, we obtain the ROC curve ordering observations by classes – from the highest to the lowest risk.

Figure 6.4 shows the ROC curves for the two models. The point marked on the blue curve indicates that for this model 60% TPR corresponds to 16% FPR. This means that for a given cut-off point, the model correctly predicts 60% of the bad class but at the same time incorrectly classifies 16% of the good observations as bad.

Besides the shape of the ROC curve itself, the confidence interval for this curve is also important which shows the range of possible curve runs. Various methods of determining the confidence interval for the ROC curve are used. An overview of such methods is provided, for example Macskassy et al. (2005). An example of a curve diagram with confidence intervals is shown in Figure 6.5.

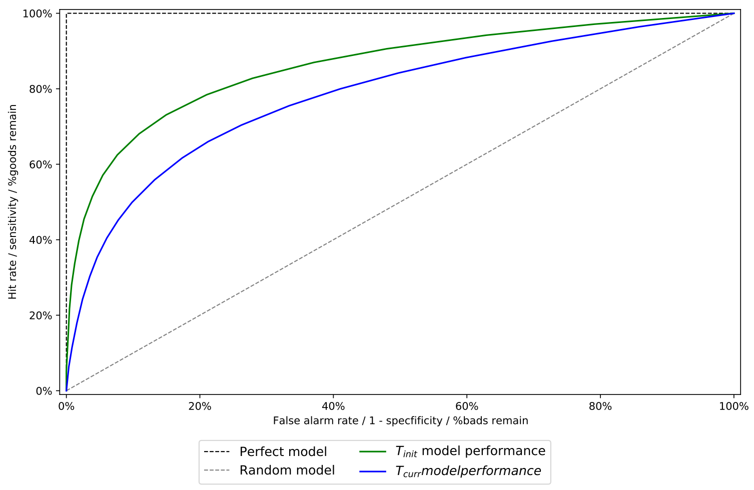


Figure 6.4: ROC curves for example of the random model, as well as perfect model
Source: own work

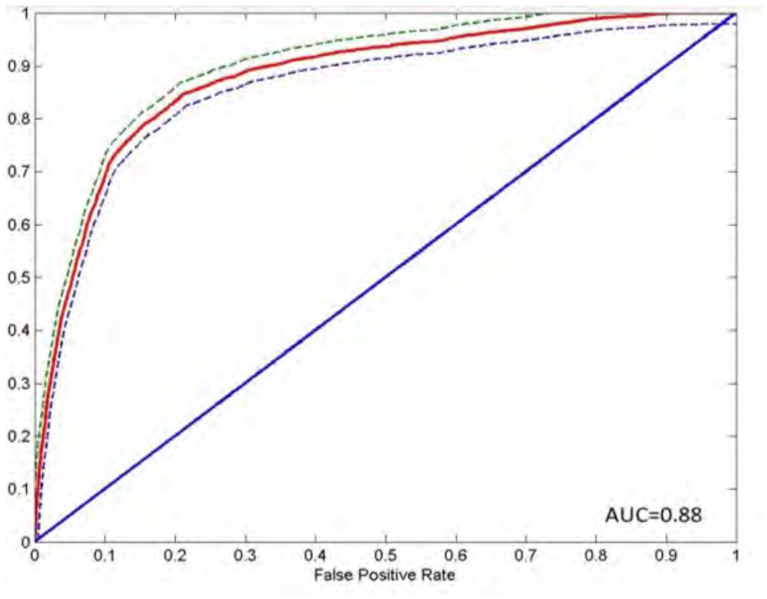


Figure 6.5: Example of the ROC curve with confidence interval
Source: own work

The smaller the confidence interval for the curve, the lower the variance of the forecast. However, it should be noted that the confidence interval of the ROC curve depends on the size of the good and bad classes. In particular, the confidence interval depends on the size of the bad class in a way that the smaller the number of bad observations, the wider the confidence interval. Thus, the sample size and the share of the bad class affect the determined confidence interval significantly.

An important feature of the ROC curve is its universality and wide application. It shows the discriminatory power of a classifier, regardless of how it was built – whether using statistical methods or machine learning methods. In addition, this curve shows how accurately the model indicates observations from the positive class (bad or default), regardless of its share in the set. This is particularly important for imbalanced data sets, as well as in situations where the share of the positive class is unstable over time, e.g. due to the business cycle.

Area Under the Curve ROC

The ROC curve is associated with a numerical quantity – the area under the curve, denoted by AUC ROC. This is the size of the area under the ROC curve and contained in the square bounding the graph of this curve. Thus, the value of this indicator may theoretically be in the range $[0, 1]$. However, given the fact that the diagonal of the square means the ROC curve for the random model, in practice the AUC ROC index values are greater than 0.5. On the other hand, in practice, the indicator does not reach the value of 1, which corresponds to the ideal model. The higher the value of this indicator, the greater the discriminatory power of the model.

The size of the area under the ROC curve has its interpretation which relates to the effectiveness of the model. This value means the probability that a randomly selected object from the bad class will have a higher PD forecast (or lower score) than a randomly selected object from the good class. This means that the model will correctly rank these observations.

In addition to estimating the AUC ROC, the confidence interval for this value is also determined. Similarly to the shape of the curve, different methods for estimating the confidence interval are also used for the AUC. An example of such a method is bootstraping, i.e. determining AUC ROC on many bootstrap samples. Engelman and Ranheimer presented another method for determining the confidence interval, using estimators of relevant probabilities (Engelmann & Rauhmeier, 2006). However, in Hanley and McNeil (1982), a method based on the Wilcoxon test of ranks was used, which simultaneously allows the determination of the size of both groups in the set needed to obtain the indicated confidence interval. According to this method, the confidence interval for AUC ROC is equal to:

$$AUC \mp z_\alpha SE(AUC)$$

where:
$SE(AUC) = \sqrt{\frac{AUC(1-AUC)+(n_1-1)(Q_1-AUC^2)+(n_0-1)(Q_2-AUC^2)}{n_0 n_1}}$,
$Q_1 = \frac{AUC}{2-AUC}$,
$Q_2 = \frac{2AUC^2}{1+AUC}$,
z_α – inverse of a normal dist. fun. for the confidence level α,
n_0 – the number of observations from class 0 (Y=0),
n_1 – the number of observations from class 1 (Y=1).

For the exemplary ROC curve shown in Figure 6.6, the confidence interval determined using the above method is as follows: $[0.873, 0.887]$, so according to the graph it is very narrow. The numbers of observations from different classes were about 4 000 (bad) and 16 0000 (good). However, for the ROC curve presented in the figure below, for which the respective numbers were about 1 000 and 21 000, the confidence interval for the AUC ROC size is $[0.826, 0.854]$.

The presented method of determining the confidence interval is based on the assumption of asymptotic convergence to the standard normal distribution which is true only with sufficiently large numbers of observations from the classes good and bad. For a small number of observations and especially for the low default portfolio, an alternative method is to determine the confidence interval using the bootstrap method. Engelmann points out that this is a computationally inefficient method (Engelmann & Rauhmeier, 2006). Together with his co-authors, Engelmann et al. (2003), he compared the operation of both methods on four portfolios with 100, 50, 20 and 10 defaults, respectively. The methods' comparison shows that

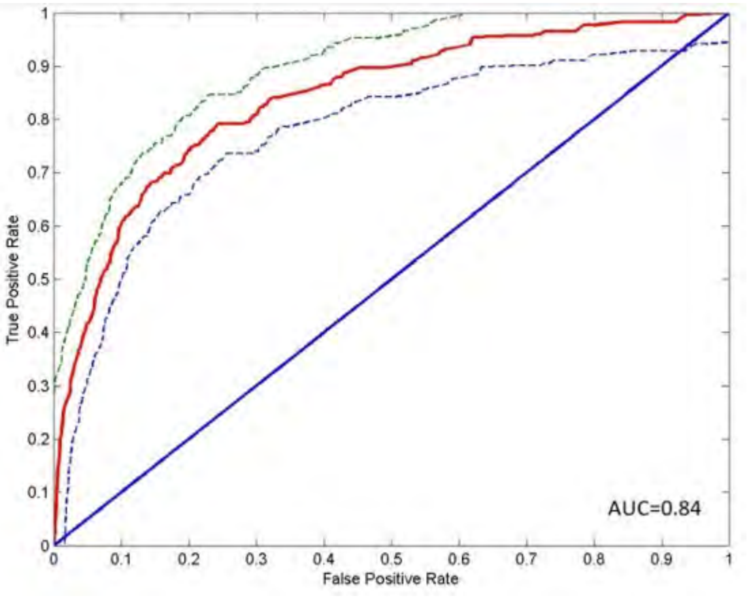


Figure 6.6: Example of the ROC curve with confidence interval
Source: own work

for 50 defaults both methods give an almost identical confidence intervals and even with a smaller number of defaults, the results are very similar.

Cumulative Accuracy Profile curve

Another curve that illustrates the discriminatory power of the model is the CAP curve. This curve is also contained in the rectangle $[0, 1] \times [0, 1]$ and has a shape similar to the ROC curve. However, its meaning is slightly different. Similarly to the ROC curve, the observations are first ordered according to the model forecast – from the highest to the lowest risk. The cumulative share of such ordered observations in the entire set is marked on the horizontal axis. On the vertical axis, the fraction of the bad class is marked, which belongs to a given group of ordered observations. Thus, it can be seen that on the vertical axis there are TPR values, assuming that the fraction of the population marked on the horizontal axis is a bad class indicated by the model.

The diagonal of the rectangle illustrates the random model. For the perfect model, all observations from the bad class have the worst ratings, so at the beginning CAP is shaped as a straight line that reaches level 1 when the cumulative part of the observations reaches the level of default rate or class bad share in the entire set. From this moment, the CAP becomes a horizontal line. In practice, the CAP curve runs between a random and an ideal model. Naturally, the closer the curve lies to the perfect model, the higher the discriminatory power of the examined model. The figure below shows a graph of the CAP curve of an example model with curves for ideal and random models marked.

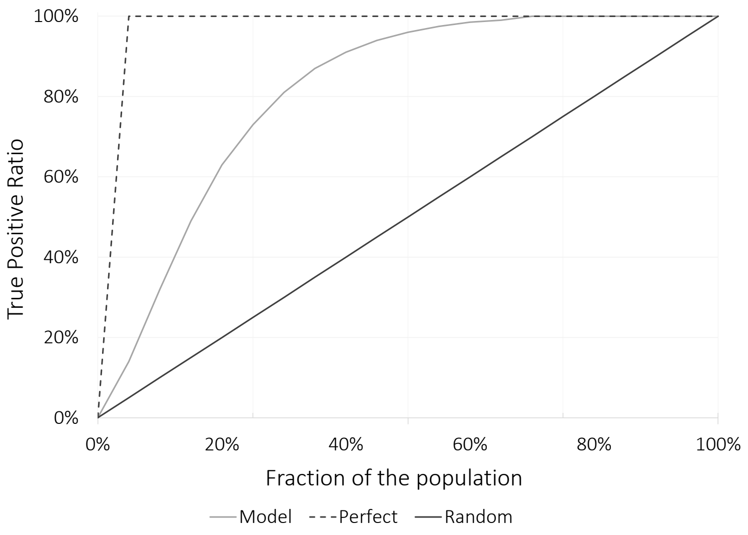


Figure 6.7: CAP curve of an example model with curves for ideal and random models
Source: own work

Gini coefficient (Accuracy Ratio)

The Gini coefficient is a numerical indicator of the discriminatory power of the model. It is associated with both of the above curves – ROC and CAP and can be determined based on each of them. In connection with the CAP curve, the name Accuracy Ratio and its abbreviation AR are more often used for this indicator. It is equal to the ratio of the size of two areas:

$$AR = \frac{a_R}{a_P}$$

The value of a_R means the area between the CAP for the validated model and the CAP for the random model (diagonal). The a_P value represents the area between the CAP for the perfect model and the CAP for the random model.

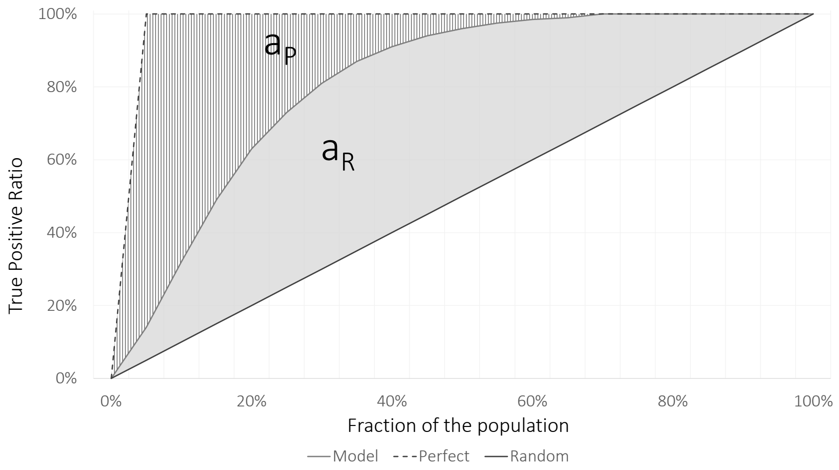


Figure 6.8: CAP curve of an example model and the areas used for AR calculation
Source: own work

The AR index values are in the range $[0, 1]$ and the closer the AR value is to 1, the greater the discriminatory power of the model. On the other hand, the Gini index value can be determined using AUC ROC, because the following equality is met

$$Gini = 2AUC - 1$$

The linear relationship of the Gini index with the AUC ROC value makes these indicators completely equivalent as they convey the same information. Using the above relationship, we can also determine the confidence interval for the Gini value. For example, for the model for which the ROC curve is shown in Figure 6.5, the Gini values and confidence interval boundaries are as follows:

$$Gini = 2 * 0.84 - 1 = 0.68$$

$$Lower_b = 2 * 0.826 - 1 = 0.652$$

$$Upper_b = 2 * 0.854 - 1 = 0.708$$

Therefore, we can say that the confidence interval for the Gini index (which is 0.68) is $[0.652, 0.708]$.

Lorenz curve

The Lorenz curve is another graphic indicator of the discriminatory power of a model. This curve was originally used in economics as an illustration of the distribution of the wealth or income of individuals in society. In credit risk, it was also used in the assessment of LGD models. In scoring models, it is the next curve after ROC and CAP associated with statistics dependent on the cut-off point and allowing to calculate the Gini index.

Similar to the previous two curves, the construction of the Lorenz curve requires all the observations to be ordered by risk estimation based on the model. We sort observations from the highest to the lowest risk and then consider in sequence the cut-off points. For subsequent cut-off points, we determine the TPR and FPR sizes and mark them on the horizontal and vertical axes respectively. In this approach, it is simply the reverse of the ROC curve. For the rating model, the classes are ordered from highest to lowest risk. Then, the cumulative percentage of rejected bad observations is marked on the horizontal axis and the cumulative percentage of rejected good observations on the vertical axis.

As well as for the ROC and CAP curves, the diagonal of the rectangle in which the Lorenz curve is contained means a random model. The perfect model means a curve that consists of two edges of the rectangle and passes through the point $(1, 0)$. In practice, the Lorenz curve lies in the area below the diagonal and the lower it lies and the more convex it is, the more discriminatory power the model has. The figure below presents Lorenz curves for two models, designated T_{init} and T_{curr} respectively. According to the interpretation described above, the T_{init} model to which the green curve corresponds has greater discriminatory power than the T_{curr} model for which the curve is marked in blue.

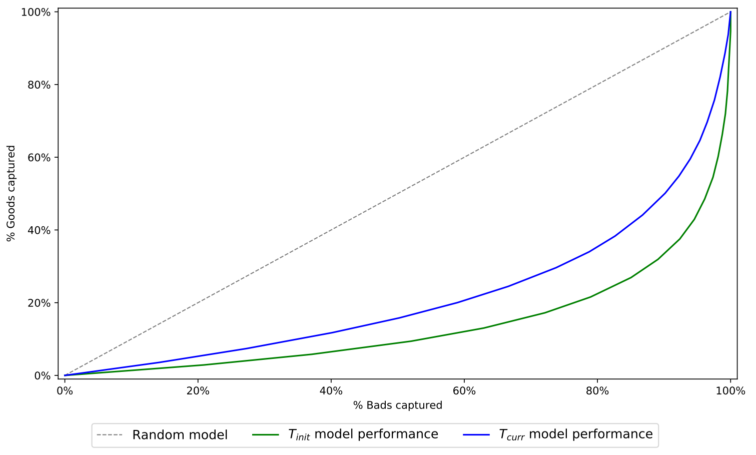


Figure 6.9: Lorenz curve of the credit scoring model performance T_{init} and T_{curr}
Source: own work

The numerical indicator of the quality of the model determined on the basis of the Lorenz curve is the area contained between the random model (diagonal) and the curve for the tested model. The higher the value of this indicator, the higher the quality of the model. In addition, if we mark the area below the Lorenz curve as A_L, then the relationship between A_L and the Gini index describes the equality:

$$Gini = 2A_L$$

The above descriptions of ROC, CAP and Lorenz curves show that they actually carry the same information and can be treated as equivalent. They all have a common feature that they are based on a comparison of model forecasts with the observed classes. They do not require additional information on the structure and operation of the model. Therefore, we can generate these curves for any models that predict the values of a binary variable, regardless of the algorithms used to build the models. Thus, it is possible to evaluate in this way not only models having understandable interpretation but also models for which we only know the resulting prediction. It is also important that the shapes of these curves and the assessment of the model depend only on the ordering observations by the model and the separation of bad and good classes. Therefore, we receive correct assessments of the quality of models even in the case of low or variable share of the bad class.

Lift curve

The lift curve (cumulative or non-cumulative) shows the effectiveness of the constructed analytical model. Similarly to other quality indicators, Lift is based on a comparison of the model forecast with the actual object class (good/bad). To determine the curve, all observations should be ordered according to the risk indicated by the model – from the highest to the lowest. Then the set is divided into equal subsets and the actual share of the bad class is determined in each of them. The Lift value can be expressed in a percentage form as a share of the bad class in a given group. On the other hand, Lift in numerical form indicates how many times the level of bad class in a given group exceeds the a priori level. Therefore, it is the value of the lift statistic (described in section “Confusion Matrix”) which – as it was indicated – shows how many times more effectively the obtained model indicates a positive class (bad/default) than the random model.

The ordered cases are marked on the horizontal axis of the graph and the Lift values on the vertical axis. Subsequent points on the horizontal axis represent a given group of observations, so they can be equated with cut-off points. If the model works properly, the curve (especially in the cumulative form) should be monotonically decreasing. Figure 6.10 shows an example of a Lift curve. The observations were divided into 50 equal buckets. The maximum lift value for this model is slightly above 4.5.

The value of 4.5 is only achieved for a very small set of the riskiest observations. However, even for 5% of the riskiest observations, Lift is still above 4. This means that the model in this group indicates the bad class four times more effectively than the random model. The a priori level of risk

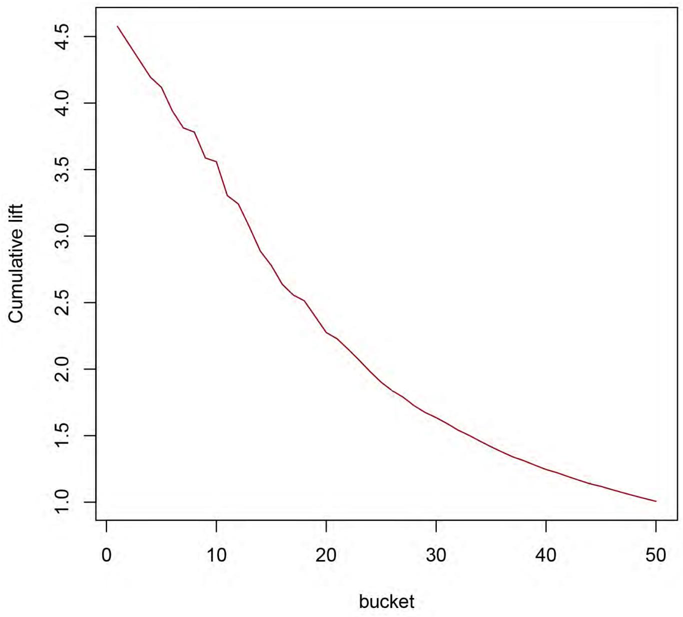


Figure 6.10: Lift curve of an example model
Source: own work

in the data set that was used to test the model is 21.85%. Thus, in the group of 5% of the riskiest observations indicated by the model, the share of the bad class is approximately 87.4%.

Kolmogorov-Smirnov test

The Kolmogorov-Smirnov test examines the distance between cumulative distribution functions for two distributions of random variables. In the case of credit scoring, the K-S test verifies the significance of the difference between empirical distributions for two classes of observations: good and bad. We organize the observations for the study according to an increase in $P(Y = 1)$, where $Y = 1$ indicates the bad class. Then we determine conditional (i.e. separately for classes $Y = 0$, $Y = 1$) empirical cumulative distribution functions for observations ordered in such a way.

The zero hypothesis (H0) is tested: the distributions do not differ significantly. The alternative hypothesis (H1) has the form: distributions differ significantly. Test statistics are proportional to the maximum difference between the distribution functions. The maximum difference between these functions for classes $Y = 0$ and $Y = 1$ is called the K-S (Kolmogorov-Smirnov) statistics and characterizes the correct separation of the good and bad classes:

$$KS = \max_x |F0 - F1|.$$

The K-S test statistics is equal to:

$$\sqrt{\frac{n_0 n_1}{n_0 + n_1}} KS$$

where

n_0 – the number of observations from class 0 ($y = 0$),

n_1 – the number of observations from class 1 ($y = 1$).

The (H0) hypothesis is rejected at the significance level α if the test statistics exceeds the critical value K_α of the K-S distribution, i.e. if the inequality is met

$$\sqrt{\frac{n_0 n_1}{n_0 + n_1}} KS > K_\alpha$$

The critical values K_α for the most commonly used significance levels of the test are presented in Table 6.4.

α	0.1	0.05	0.01
K_α	1.22	1.36	1.63

Table 6.4: The critical values of the Kolmogorov-Smirnov statistics for different values of significance level
Source: own work

The result of the test is the rejection or acceptance of the (H0) hypothesis. However, it should be noted that the shape of the K-S curve and the value of K-S statistics are important indicators of correctness and discriminatory power of the model. The greater the distance between conditional cumulative distributions and the higher the K-S value, the more effective the model is. An example of a K-S curve and statistics value is presented in Figure 6.11.

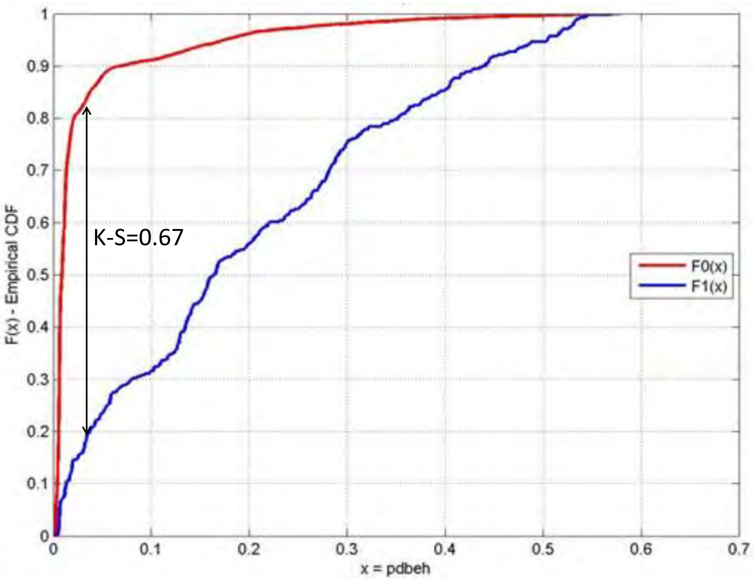


Figure 6.11: The example of the K-S distance between cumulative distribution functions
Source: own work

The disadvantage of the K-S statistics is that it is a local measure as opposed to the Gini index which globally characterizes the operation of the model. As a result, models with different discriminatory power and different Gini indices may have the same value of the K-S statistics. For the correct operation of the model, it is important that the maximum difference between the distributions occurs around proposed cut-off values. Moreover, K-S is sensitive to sample size and low number of observed defaults. For small samples, the curve is stepped, which distorts the assessment of the actual distance between the distributions.

Divergence – as a signal to noise ratio

Signal-to-noise ratio (SNR) is a physical indicator used in image or sound processing. This measure compares the level of the useful signal to the background noise level. Statistics also use the SNR as a measure of the ratio of useful information to background noise. For a single sample, it is the quotient of the sample mean by its standard deviation: $SNR = \frac{\bar{m}}{\sigma}$. The SNR index can also be used as a measure of the distance between two vectors of mean values $\bar{m}_1$ and $\bar{m}_2$ and the standard deviation σ_1 and σ_2 and then its value is

$$SNR = \frac{\bar{m}_1 - \bar{m}_2}{\sigma_1 + \sigma_2}.$$

This statistics can be used to assess the correctness of the binary classification and to determine the optimal cut-off point for the classifier.

Divergence, just as K-S statistics, is a measure of the model's ability to correctly separate observations from the bad and good classes. It illustrates the distance between the conditional distribution of assessments obtained from the model (score or PD) for two classes of clients. The numerical measure of the distance between distributions is the value

$$D^2 = \frac{(\bar{m}_0 - \bar{m}_1)^2}{(\sigma_0^2 + \sigma_1^2)/2},$$

where
$\bar{m}_0$ – average score of $P(Y = 1)$ for class 0,
$\bar{m}_1$ – average score of $P(Y = 1)$ for class 1,
σ_0^2 – score or probability $P(Y = 1)$ variance for class 0,
σ_1^2 – score or probability $P(Y = 1)$ variance for class 1.

Divergence can take any non-negative values, with a value of 0 meaning no discriminatory ability of the model. In turn, the higher the divergence values, the greater the discriminatory power of the model.

Figure 6.12 shows the PD value distributions separately for the classes $Y = 0$, $Y = 1$. A line with arrows indicates the distance between the means for conditional distributions. The divergence value for the distribution in the figure is 5.16.

Both divergence and K-S statistics can be used to study and illustrate the correct separation of classes by classification models built using various algorithms, e.g. statistical but also machine learning.

Influence of variables on the predictive power of the model

One of the elements of periodic validation is the analysis of the predictive power of individual variables in the model. In the case of the scoring card,

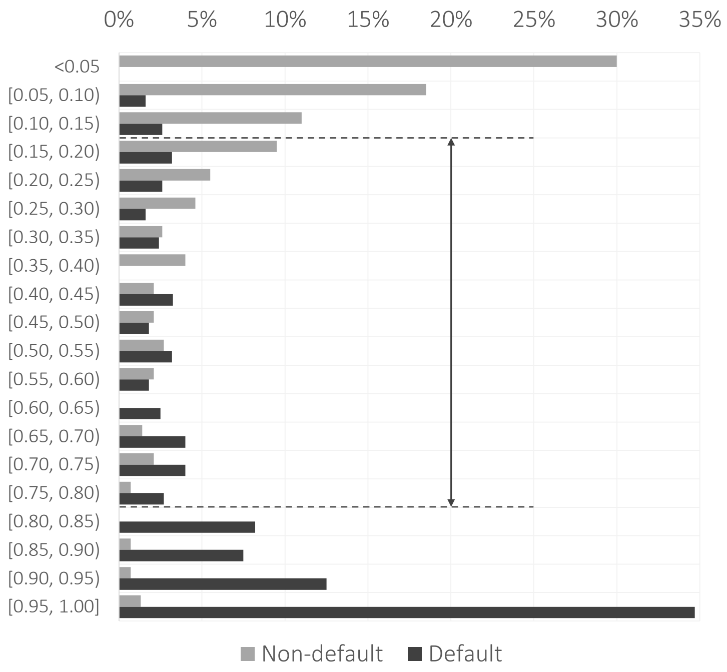


Figure 6.12: Distance between the means for conditional distributions of PD
Source: own work

the difference in the average score generated by a given variable for good and bad observations can be used as a measure of the discriminatory power of one variable – analogous to the divergence of the entire model. Note that the square root of the divergence is

$$D = \frac{\bar{m}_0 - \bar{m}_1}{\sqrt{\frac{1}{2}(\sigma_0^2 + \sigma_1^2)}}$$

Thus, there is a score difference in the numerator which can be decomposed into corresponding differences for the individual variables. Below is an example of a scoring card composed of 5 characteristics. Standard deviations of score distributions for good and bad in the studied population are respectively $\sigma_0 = 62.65$, $\sigma_1 = 53.12$. Thus, the denominator of the fraction that determines the value of D is

$$M_D = \sqrt{\frac{1}{2}(\sigma_0^2 + \sigma_1^2)} = 58.08$$

Table 6.5 shows the average score values in the good and bad groups for the entire model and for individual variables. The next column shows the average score difference. Of course, the difference for the entire model is the sum of such differences for individual variables. The last column of the table shows the quotient of the score difference by the denominator M_D calculated above. For the entire model, it means the square root of divergence. The remaining numbers add up to this value and can be interpreted as the share of a given variable in the model discriminatory power.

	avg_good	avg_bad	difference	sqrt_div
score	202.09	85.16	116.92	2.01
financed amount	58.12	15.75	42.36	0.73
period of employment	38.31	6.78	31.53	0.54
net income	28.67	13.7	14.96	0.26
previous contracts	16.05	13.36	2.69	0.05
vehicle age	60.94	35.57	25.38	0.44

Table 6.5: The average score values in the good and bad groups
Source: own work

The advantage of such a measure is its additivity – the predictive power indices of individual variables add up to the divergence of the entire model. More universal measures that can be determined in the case of any model are the Gini and Information Value (IV) indices. The Gini index for a single variable is determined in the same way as Gini for the model – based on the value of the variable and the bad or good class designation. The observations are then ordered by the value of the variable under study according to decreasing risk. The Gini index can be calculated from TPR, FPR, TNR and FNR values determined for any cut-off point. For the variables from the model described above, the values of IV and Gini are presented in Table 6.6.

The IV and Gini indices are not additive – the values for individual variables do not constitute the appropriate value for the entire model. This

Variable	IV	Gini
financed amount	1.45	0.58
period of employment	1.23	0.5
net income	0.53	0.38
previous contracts	0.14	0.15
vehicle age	0.74	0.42

Table 6.6: The values of IV and Gini indices for variables in a model
Source: own work

is a particularly important problem in the case of machine learning methods which also use highly correlated variables. In such a situation, Shapley values can be used as significance indicators. Shapley values are described and characterized in Chapter 7. There is also an example of a calculation and application of Shapley values there. The difference of the mean values of Shapley in the groups of good and bad indicates the discriminatory power of a single variable and, at the same time, these indicators have the property of additivity. Of course, Shapley values can also be used with classic models. Importantly, for the logistic regression model built on the WoE variables, the same role for individual variables play the indicators obtained as the products of the coefficients from the model (beta for each variable) by the appropriate IV values for explanatory variables.

Another method of determining the significance of variables in the model is the permutation method, which is described in Chapter 7 as Permutation Feature Importance (PFI). It is a model-agnostic method for determining the Variable Importance Measure (VIM). It consists in comparing the mean error rate for the observations before permutation of the selected predictor with the mean error level after permutation. This method is also described in detail in Chapter 7. The idea of the permutation method is that for an unimportant variable in the model, the permutation will not affect the classification and error rate. On the other hand, if the predictor is important, its permutation will have a significant negative impact on the classification and increase in error. Therefore, the higher the error rate, the greater the discrete force of the variable. The VIM indices obtained by the permutation method for the same as the above set of variables in the random forest model are shown in Figure 6.13.

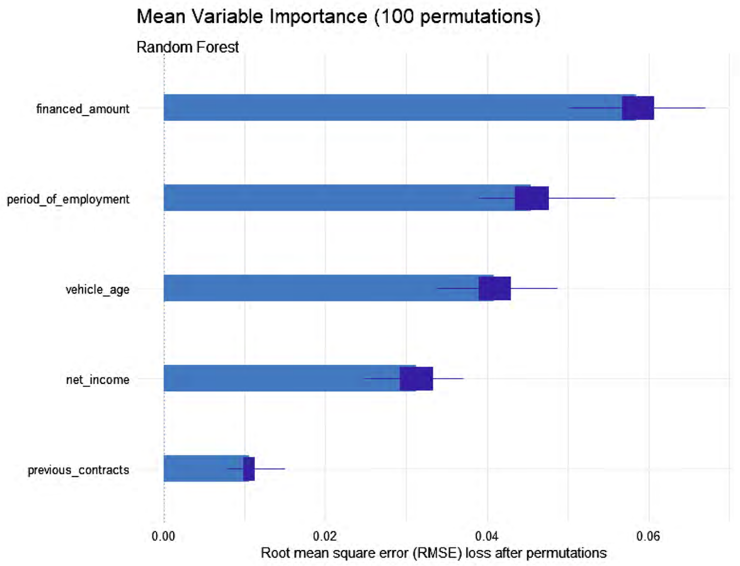


Figure 6.13: The VIM indices obtained by the permutation method for random forest model
Source: own work

The strongest variable in this model turned out to be the variable describing the amount of financing. The next one was the period of employment, the least importance has the variable concerning previous contracts.

6.4.2 Predictive power, calibration

The discriminatory power analysis outlined earlier gave an answer to whether, as a rule, the higher credit score granted to clients implied a lower observable frequency of the default event. As it was indicated, due to the lack of observability of $PD_{i,t}$ values at the level of individual clients, this measure is approximated by the observable frequency of the default event in given rating classes. Calibration tests are used to verify the correctness of the model's calibration, i.e. estimation of the PD parameter at the level of rating classes or the entire model.

It is worth emphasizing that, when assessing the discriminatory power of the model, shifting the result by a constant does not change the quality assessment. The situation is different for the evaluation of the calibration. Here, the most important thing is the correct assessment of the default rate in classes and in the population. Therefore, it is essential what approach was used for the calibration and whether, for example, macroeconomic factors were taken into account.

When testing model calibration, it should be remembered that, depending on the approach used, a change in the economic environment may cause a change in the default rates in bands or shifts of clients between rating classes. For example, in the case of the PIT model, the deterioration of the economic situation will move customers towards worse ratings. The economic recovery will bring about opposite changes.

Binomial test

Our subject of interest – modeled variable that refers to default/non-default flag – is a dichotomous variable that in practice is usually modeled using the class of generalized linear models (e.g. logistic regression). In such a case, the model estimates an expected value of an endogenous variable conditional to the set of exogenous variables (specific for a particular observation). The validation of such a model may be performed based on the frequency of default in a particular rating grade.

The most straightforward test that can be utilized is based on the assumption that the defaults appear independently and what we are aiming for is to verify the unconditional coverage[6] of default. In such a setup, we assume in H_0 that the $\widehat{PD}_i$ for i-th rating bin is correct, i.e. is equal to PD_i (which is unobservable – hence standard predictive error-based methods do not apply). In the binomial test, the distribution of a random variable (i.e. the probability of default) is assumed to be binomial. For a set of given exposures and within the particular time window (that refers to the time-horizon of PD_i), we measure the empirical frequency of the defaults. Based on the binomial distribution, we can statistically evaluate the

[6]The analog to the Value-at-Risk validation methodology (Kupiec, 1995)

observed rate of defaults. Usually, the alternative hypothesis refers to non-equality to assumed PD_i (especially in case of VaR backtesting) – from the business perspective, the one sided-test may be assumed (i.e. conservative estimates of $\widehat{PD}_i$).

From the practical point of view, such a test does not allow to model the performance of the entire rating system at once. The particular rating grades have to be tested separately. Also, it should be noted that in the presence of a multi-period data set, we are utilizing only a single time period for validation. Moreover, especially in the case of credit risk modeling, the assumption of defaults' independence is usually violated. The method's most significant advantage, however, is that the results of the test are very straightforward and intuitive.

As a consequence of such properties (i.e. if H_0 is correct, the well-calibrated test should reject H_0 at given significance levels, even if the process generating data is following H_0!), a binomial test may be used only for the narrow range of cases, e.g. when the bank starts to use the rating system for a new customers segment and hence does not possess the broader data set of historical observations.

Binomial test with correlation

Based on the results of the binomial test, the independence assumption can be further investigated. In terms of the binomial test with correlation, this assumption can be removed, but the correlation structure has to be imposed on the model – analytically or numerically.

In this test, it is assumed that the probability of default of a particular debtor depends on the: a) idiosyncratic factors of a debtor (specific exogenous variables reflected in the score), b) the correlation with the systemic factor (upon which all debtors depend on). The introduction of the correlation factor leads to the less intuitive model's interpretation and the more complicated model's formulae. The model incorporating the correlation structure is also known as the *Asymptotic Single Risk Factor model* (ASRF) which is used for the calculation of capital requirement under CRR.

The drawbacks of the approach stem mainly from the fact that a small variation in the sample can lead to significant deviation of the crucial values of the test (and hence also PD_i estimates). As a result, the binomial test with correlation shows that the type I error is higher than the significance of the test. Moreover, still, the binomial test with correlation is based on the singular rating-grade for a single time period evaluation.

On the other hand, from the regulatory perspective, it should be noted that this test is conceptually aligned with the formulae (i.e. ASRF model) used for the calculation of Risk-Weighted Assets (and hence credit capital requirements). In such a case, from the validation point of view, using the *aligned* test should be perceived as something desirable and expected (e.g. art. 153 of the CRR).

Normal test

To address the issues of testing the model calibration only for a single time period, we can utilize the normal test which evaluates whether the actual probabilities (PD_i) are higher than their forecast ($\widehat{PD_i}$). The test's statistic follows (asymptotically, when the number of periods goes to infinity) normal distribution. In this test, we are capable of utilizing the multiple periods for estimation of the empirical frequency of default for a rating grade, which in principle reflects the TTC philosophy of modeling PD models.

This test does not allow to evaluate multiple rating grades but does not assume the independence of the defaults in the particular period but assumes the independence of defaults between periods. In contrast to the other described tests, this one is a multi-periods test which, in case of availability of data, provides more robust and stable validation conclusions.

The Jeffrey test

The Jeffrey test allows for verification of the predictive power of credit scoring models at the level of individual rating classes as well as the entire portfolio (Brown et al., 2001). As part of the Jeffrey test, the values of predicted defaults are compared with the empirical defaults under the binomial model. The null hypothesis of the test assumes that the projected

PD values applied at the beginning of the period are greater than the values of the defaults realized. Assuming a priori distribution as a binomial distribution, the a posteriori distribution is set to be a beta distribution with the parameter $a = D\ + \frac{1}{2}$ and $b = N - D + \frac{1}{2}$, where N is the total number of customers, D is the number of customers who have entered the default state. The p-value of the test is $\beta_{D+\frac{1}{2}, N-D+\frac{1}{2}}(PD)$. The test may be carried out for each rating class as well as at the entire portfolio level.

The Hosmer-Lemeshow test

One of the most commonly used tests for PD calibration is the Hosmer-Lemeshow test (Hosmer Jr et al., 2013). It verifies whether the expected frequencies of defaults in particular rating classes are equal to observed ones. The null hypothesis is that the observed default rate is indifferent from the estimated PD, i.e.

$$H_0 : PD_g = DR_g, \forall g \in (1, \dots, G)$$

The test statistic is calculated as follows:

$$\sum_{g=1}^{G} \frac{N_g(PD_g - DR_g)^2}{PD_g(1 - PD_g)} \Rightarrow \chi^2(G-2)$$

where N_g is the number of observations in the rating grade g. Test statistic converges to a Chi-Square distribution with G-2 degrees of freedom (as the number of observations tends to infinity); also, this test assumes the independence of the defaults.

The Pluto-Tasche test

The confidence level based approach can be used for estimating PDs for portfolios without any defaults or with a very low number of them (Pluto & Tasche, 2011). This method delivers confidence intervals for the PDs in each rating grade, which can be adjusted by choosing the confidence level. The approach follows the most prudent estimation principle thus yielding monotone PD estimates. Low default portfolios are of special concern in the context of estimation of risk parameters for IRB as PD estimates for such portfolios based on historical performance or expert judgement may substantially underestimate capital requirements.

Let us start with the case of no defaults portfolio and assume that observations are distributed to rating grades A, B and C based on a set of predefined business rules from an expert-based credit risk model. Grade A is of the highest creditworthiness and with n_A observations followed by grade B with n_B observation and the lowest creditworthiness grade of C with n_C observations. Moreover, let us assume that default occurs independently (even if we consider no default case for now) and rating grades give us correct ranking, i.e. $PD_A \leq PD_B \leq PD_C$, where PD_i is (to be calculated) probability of default for rating grade i. In consequence of the most prudent estimate principle, we get that $PD_A = PD_B = PD_C$. With this relation in place, we can proceed to determining a confidence interval for PD_A at a confidence level α. The confidence interval can be viewed as a set of all values PD_A that the probability of not observing any default during the observation period is not less than 1-α which means that:

$$(1 - PD_A)^{n_A+n_B+n_C} \geq 1 - \alpha$$

therefore

$$PD_A \leq 1 - (1 - \alpha)^{\frac{1}{n_A+n_B+n_C}}$$

Consequently, the most prudent estimate of PD_B is obtained when $PD_B = PD_C$ and applying the same rationale we get

$$PD_B \leq 1 - (1 - \alpha)^{\frac{1}{n_B+n_C}}$$

and for PD_C we can use only observations in grade C which gives

$$PD_C \leq 1 - (1 - \alpha)^{\frac{1}{n_C}}$$

Naturally, apart from the confidence level α, the most impact for the upper confidence bound has the sample size. The smaller the sample the greater upper confidence bound. However, as the data set which gives the frequencies is given, the assumed confidence level is to be decided. Authors of the approach suggest to keep it below 95 percent, however keeping the confidence level between 50 and 75 percent was also suggested by Benjamin et al. (2006).

Let us expand the previous example on a case where no default was observed in grade A, three defaults were observed for grade B and one in grade C. Similarly, we start with determining the most prudent confidence region for the PD_A by assuming again that PDs of three grades are equal. Then, the confidence region for the probability of observing no more than four defaults can be determined by using the binomial distribution (under the assumption of independence of default events) in the following manner

$$1-\alpha \leq \sum_{i=0}^{4} \binom{n_A+n_B+n_C}{i} PD_A^i \left(1-PD_A\right)^{n_A+n_B+n_C-i}$$

Following the reasoning for PD_B and PD_C yields

$$1-\alpha \leq \sum_{i=0}^{4} \binom{n_B+n_C}{i} PD_B^i \left(1-PD_B\right)^{n_B+n_C-i}$$

$$1-\alpha \leq= \left(1-PD_C\right)^{n_C} + n_C PD_C \left(1-PD_C\right)^{n_C-1}$$

Those inequalities can be solved both analytically – using an appropriate beta distribution function – and numerically.

Even though there were no defaults in the grade A the results were impacted as defaults from grades B and C entered the calculation. They affect the upper confidence bounds which are higher than in the previous case. This approach can be expanded further to involve correlated default events by employing the Basel single-factor model and the asset correlations defined in the Basel Accord. It can also be calibrated to observed portfolio PD by introducing the scaling factor K in the following manner

$$PD_{Portfolio} = \frac{\widehat{PD}_A n_A + \widehat{PD}_B n_B + \widehat{PD}_C n_C}{n_A+n_B+n_C} K$$

In the above-mentioned examples, we have considered just one period. In the original article, also a multi-period setting was presented. As described above, each of the tests has some advantages and drawbacks. In Table 6.7, we present the summary of the main characteristics of the above-mentioned, predictive tests.

	Multiple rating grades	No-independence assumption	Multiple period testing	No rating grade homogeneity	No issues with type I error	Test simplicity
Binomial	✗	✗	✗	✗	✓	✓
Binomial with corr.	✗	✓	✗	✗	✗	✗
Normal	✗	✗	✓	✓	✓	✓
Jeffrey	✓	✗	✗	✓	✓	✓
Hos.-Lem.	✓	✗	✗	✓	✓	✓

Table 6.7: The main characteristics of the calibration tests
Source: own work

The particular usage of a test should also be supported by more broad investigation and reflection, e.g. model assumptions, availability of horizontal data (more observations for one period) and vertical (multiple periods). Statistical tests are based on specific assumptions about the number of observations, the independence of class defaults and the stability of the default rate. In fact, these assumptions are not always met, especially in the case of a small sample or a portfolio with a low default rate. This means that the actual confidence intervals for the PD estimates in the classes are wider than the calculations indicate. On the other hand, big data can also lead to wrong conclusions. With a very large number of observations, the test result may indicate the rejection of the H_0 hypothesis, even if practice and experience show that H_0 is true. For these reasons, often more importance is attached not to the statistical significance of the estimate but to the assessment of the materiality of the problem studied, which can be expressed, for example, as a percentage of exposure, provisions or Risk Weighted Assets.

6.4.3 Stability

Credit risk assessment models are built based on historical data, therefore one of the validation elements is testing the stability of the model. It aims to verify that the operating conditions of the model are still as assumed during the construction of the model. However, it should be noted that the stability of the model's operation depends on various factors. The lack of stability may result from a change in the dependencies and parameters in the model and then the model should be modified. However, the observed lack of stability of the model's assessments may also result from a change in the customer population or from a change in banking processes. Lack of stability is also not always a negative phenomenon, e.g. when a change in the portfolio indicates a higher share of reliable customers. The study of the stability of the model and population is not only to show whether the distribution of customer ratings is changing or not. An equally important element is understanding the reasons for any changes.

Customer migrations

One of the elements of the system stability test is the analysis of rating changes and customer migration between rating classes. The probabilities of transitions between rating classes in a given period show the migration matrix. Usually, the period indicated in the migration matrix is one year. The migration matrix at a set point in time can be determined based on two ratings for the same established group of clients. The first rating is used as the base classification and the second – made after a fixed period, e.g. 1 year, allows to determine the flow rates between classes that occurred in that period. An example of the migration matrix is presented in Table 6.8.

However, the values for the transition matrix in a given period are specific implementations of random variables that characterize the transitions between rating classes. The probabilities of transitions between classes are estimated based on historical data. For this purpose, for example, a cohort method can be used, which consists in determining the expected probabilities of transition based on appropriate matrices from many periods. The

		Rating after 1 year					
		1	2	3	4	5	Default
Base rating	1	91.85	4.87	1.34	0.92	0.62	0.41
	2	5.97	62.46	15.63	8.72	6.33	0.88
	3	2.31	13.87	45.87	22.47	14.26	1.21
	4	0.89	9.78	18.46	39.98	26.45	4.44
	5	0.48	4.26	15.34	19.34	51.99	8.59

Table 6.8: An example of the migration matrix (in %)
Source: own work

disadvantage of this method is the assumption of the independence of individual class transitions, which may not be true. However, this assumption is often made to simplify the estimation of transition probabilities.

The migration matrix informs about the stability of risk assessments resulting from the model. If the diagonal values are high, the ratings are stable. A large dispersion of values outside the diagonal indicates a lack of stability in assessments. It should be remembered, however, that various factors influence the stability of model assessments. The reason for changes in risk assessment may be for example a change in the economic situation that affects customer assessments. The stability of the model's ratings also depends on its design. The application model based on socio-demographic data at the time of application will show zero migrations during the lifetime of the loan. A model based on a small number of variables may show rare but large migrations. In contrast, the model based on many variables (as in machine learning) will show very frequent but small migrations.

Stability of the migration matrix

The answer to the question about the stability of the transition matrix is very important for the rating analysis. This stability means that the probabilities of transition between given classes are constant in time. The homogeneity of the transition matrix can be tested using the χ^2 test, assuming that the individual transitions are independent. The H_0 hypothesis is tested then that

$$P(t_1) = P(t_2) = \ldots = P(t_N)$$

It is often accepted that migrations between rating classes constitute the first-order Markov chain. This means that the migration matrix is time invariant. In addition, it is easy to determine the probabilities of transition between classes after k periods because

$$P\left(k\right) = P\left(t_1\right) P\left(t_2\right) \ldots P\left(t_k\right) = \left(P\left(t_1\right)\right)^k$$

However, we cannot always assume that migrations between classes reflect the Markov process. And not always transition matrices can be treated as constant, even if – as in the case of big data – tests can confirm the truth of such a hypothesis.

Importantly, changes of the transition matrix may affect the capital requirement and risk management. Therefore, an alternative to the application of statistical tests confirming our assumptions may be the assessment of changes in the transition matrix and their materiality. One of the metrics defined in the matrix space may be used for this purpose, for example the metric L_1 or L_2 as defined below. For square matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ where $i, j = 1, \ldots, n$, we define the L_1 and L_2 metrics as follows:

$$L_1\left(\mathbf{A}, \mathbf{B}\right) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left|a_{ij} - b_{ij}\right|$$

$$L_2\left(\mathbf{A}, \mathbf{B}\right) = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} - b_{ij}\right)^2}$$

Assessment of the materiality of changes in the transition matrix can be based on the distance determined by one of the metrics and the exposure for the portfolio.

Concentration in rating grades

The purpose of the concentration test is to check and assess whether significant structural changes can be observed between the periods of validation and the implementation validation within the rating classes, i.e. whether the classes have been dispersed or increased in concentration (the number and size of customer involvement in rating classes are more or less uniform). Levels of concentration should be measured in terms of the percentage of customers as well as the size of their exposure (European Central Bank, 2019). It is worth emphasizing that high concentration is not always clearly negative. In special situations, e.g. very high-quality model or for low default portfolios, there may be a problem of excessive concentration. It should be remembered that the high quality of the model has higher priority than low concentration.

Let K denote the number of non-defaulted exposure rating classes; let R_i be the relative incidence of the i-th rating class at the beginning of the observation period (weighted by the number of clients or the value of their exposure). Then the value of the coefficient of variation at the moment t (i.e. performing proper validation) is given as:

$$CV_t = \sqrt{K \sum_{i=1}^{K} \left(R_i - \frac{1}{K} \right)^2}$$

The null hypothesis of the test assumes that the Herfindahl coefficient value is lower at the time of proper validation than at the time of implementation validation. The p-value of the test is given by:

$$1 - \Phi \left(\frac{\sqrt{K-1}\,(CV_t - CV_{t_0})}{\sqrt{CV_t^2 \left(0.5 + CV_t^2\right)}} \right)$$

where: $\Phi\,(.)$ is the cumulative distribution function of the standard normal distribution, CV_{t_0} is the value of the coefficient of variation determined as at the date of implementation validation. As part of preparing validation reports (European Central Bank, 2019), next to the value of the coefficient of variation, values of the Herfindahl coefficient (Edward Miller, 1991), should also be reported:

$$HI_t = 1 + \frac{\log\left(\frac{CV_t^2+1}{K}\right)}{\log\left(K\right)}$$

As previously indicated, the test is carried out based on the relative frequency of occurrence of the i th rating class of observations based on the number and value of customer involvement in rating classes, i.e.

- weighted by the number of customers: $R_i = \frac{N_i}{\sum_{j=1}^{K} Nj}$, where N_j is the number of customers assigned the j-th rating class,
- weighted by the exposure value (only to calculate the HI_t parameter): $R_i = \frac{E_i}{\sum_{j=1}^{K} Ej}$, where E_j is the exposure value of customers assigned to the j-th rating class.

To illustrate the calculations related to the concentration test, let us suppose we have 20 non-default exposure rating classes. We determine the values of the R_i parameter and the expression $\left(R_i - \frac{1}{K}\ \right)^2$ for each rating class as shown in Table 6.9.

Based on these results, the values CV_{t_0}, CV_t can be determined:

$$CV_{init} = \sqrt{0.000671^2 \times 10} = 0.1159$$

$$CV_{curr} = \sqrt{0.000697^2 \times 10} = 0.1180$$

The p-value for this test, given the sample data, is as follows:

$$1 - \Phi\left(\frac{\sqrt{10-1}\left(0.1180 - 0.1159\right)}{\sqrt{0.1159^2\left(0.5 + 0.1180^2\right)}}\right) = 45.61\%$$

which means that we fail to reject $H_0 : HI_{init} < HI_{curr}$, i.e. the observed concentration measured using the Herfindahl index at the time of proper validation is lower than at the time of implementation validation. In addition, coefficients of variation and relative cardinal frequencies (i.e. R_i determined based on the number of clients) can be used to determine Herfind-

	N_init	N_curr	R_init	R_curr	$(\frac{\text{R\_init}-1}{K})^2$	$(\frac{\text{R\_curr}-1}{K})^2$
1	547	720	0.043197	0.046455	0.00004628	0.00001257
2	711	846	0.056148	0.054584	0.00003780	0.00002101
3	538	650	0.042486	0.041938	0.00005646	0.00006499
4	553	840	0.043671	0.054197	0.00004006	0.00001762
5	704	851	0.055595	0.054907	0.00003130	0.00002408
6	616	676	0.048646	0.043616	0.00000183	0.00004076
7	652	893	0.051489	0.057617	0.00000222	0.00005801
8	749	890	0.059149	0.057423	0.00008370	0.00005510
9	643	795	0.050778	0.051294	0.00000061	0.00000167
10	677	831	0.053463	0.053616	0.00001199	0.00001308
11	523	739	0.041301	0.04768	0.00007567	0.00000538
12	733	728	0.057885	0.046971	0.00006218	0.00000918
13	654	833	0.051647	0.053745	0.00000271	0.00001403
14	530	603	0.041854	0.038906	0.00006635	0.00012308
15	671	858	0.052989	0.055358	0.00000893	0.00002871
16	658	875	0.051962	0.056455	0.00000385	0.00004167
17	736	707	0.058122	0.045616	0.00006597	0.00001922
18	624	614	0.049277	0.039615	0.00000052	0.00010784
19	526	843	0.041538	0.054391	0.00007160	0.00001928
20	618	707	0.048804	0.045616	0.00000143	0.00001922
Port.	12 663	15 499	1	1	0.000671466	0.000696502

Table 6.9: An xample of the calculation of concentration measures
Source: own work

ahal index values as of the date of implementation validation and proper validation:

$$HI_{init} = 1 + \frac{\log\left(\frac{0.1159^2+1}{10}\right)}{\log(10)} = 0.45\%$$

$$HI_{curr} = 1 + \frac{\log\left(\frac{0.1180^2+1}{10}\right)}{\log(10)} = 0.46\%$$

The values of HI_{init} and HI_{curr} are low (less than 10%), which suggests that there is no problem with an excessive concentration within individual rating groups.

Population Stability Index

The examination of model stability has two main aspects – on the one hand, it covers the stability of assessments generated by the model and on the other hand, it also covers the stability of the population itself, i.e. the distribution of individual variables. The indicator that can be used in both cases is the *Population Stability Index* (PSI). It compares the distribution of the same characteristics in two different periods, one of which is treated as the base one and the other as the examined one. The base population is most often the model development sample. The examined feature can be a model score or rating but it can also be one of the variables. The PSI value is determined according to the formula

$$PSI = \sum_i (V_i - D_i)\, ln\left(\frac{V_i}{D_i}\right),$$

where

V_i – proportion i-th scoring grade in the new population,

D_i – proportion i-th scoring grade in the base population.

A PSI value below 0.1 means a stable population. A value exceeding 0.25 means an unstable population. Values from 0.1 to 0.25 mean noticeable changes in the population. For the score groups presented in Table 6.10, the value of the PSI is 0.05, so the distribution of this scoring in the new population is completely stable compared to the base population. Stability of the scoring can also be seen in Graph 6.14 which compares two distributions.

Score group	D_i Base (in %)	V_i New_pop (in %)	$V_i - D_i$	$\ln(\frac{V_i}{D_i})$	$V_i - D_i \times \ln(\frac{V_i}{D_i})$
score1	12.3	9.1	-0.0320	-0.3013	0.0096
score2	27.8	29.5	0.0170	0.0594	0.0010
score3	29.2	33.8	0.0460	0.1463	0.0067
score4	16.9	19.4	0.0250	0.1380	0.0034
score5	13.8	8.2	-0.0560	-0.5205	0.0291

Table 6.10: Scoring distribution in two periods and PSI calculations
Source: own work

In the same way, one can test the stability of nominal variables as well as numeric variables if their values are grouped. However, the PSI is heavily influenced by size fluctuations that may appear in the case of small categories for nominal variables or narrow grouping intervals for numeric variables. For this reason, instead of PSI, chi-square or Kolmogorov-Smirnov tests can be used to analyze the stability of variables. These tests compare

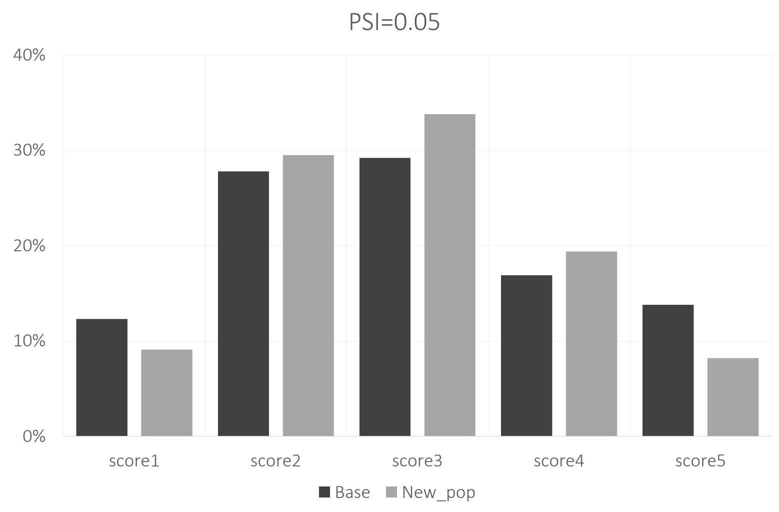


Figure 6.14: Scoring distributions in two periods
Source: own work

two distributions with each other. In the case of stability tests these are variable distributions from two periods. For scoring models, it is important to develop a characteristic analysis report. It allows comparing variable distributions over two periods but also shows how a change in the distribution of a variable affects a change in the operation of the model.

Characteristic Analysis Report and stability of variable's score

The characteristic report compares two distributions for each variable – for the test population and the base population. However, in addition to the difference in the distributions themselves, this report also shows what the average difference in scoring is for these two populations (Siddiqi, 2012). This makes it possible to understand the reasons for the change in the overall scoring determined by the model. In addition, we get the information which characteristics have the greatest impact on scoring changes.

The total score difference for the variable is determined based on the differences for individual values resulting from the change in population. In the first step, for each value of a given variable, the change in population is calculated as the difference between the examined population and the base population, that is $(V_i - D_i)$. Then, for each variable value, the point difference is determined as the product of $(V_i - D_i)$ by the number of points corresponding to the given value. Finally, the partial differences are added together and we get the total scoring points difference. Its sign shows in which direction the shift has taken place – whether customers with higher or lower scoring have come.

Variable values	D_i Base in %	V_i New_pop in %	$V_i - D_i$	Points	Score diff.
value1	22.10	18.20	-0.0390	66	-2.574
value2	29.40	27.10	-0.0230	42	-0.966
value3	17.50	16.70	-0.0080	28	-0.224
value4	17.70	25.40	0.0770	12	0.924
value5	13.30	12.60	-0.0070	32	-0.224
			Total score diff.		-3.06

Table 6.11: An example of the calculation of total scores difference
Source: own work

For example, variable presented in Table 6.11 there was a decrease in the share of categories to which higher scores correspond. The total score difference for this variable is -3.06 points. The determined difference can be interpreted in terms of risk for a specific calibration of the scorecard, e.g. if the calibration results in doubling the odds for every 20 points.

The above analysis allows assessing the stability of the assessments generated by the examined variable and its contribution to the overall score. However, as it stands, it only applies to scorecards. A more universal indicator of the stability of the scores generated by a given variable and its contribution to the global score is the change in the mean Shapley value for the variable. Importantly, the Shapley values apply to any machine learning models – we may say that Shapley values are model agnostic. The average value of this indicator for a given variable characterizes its significance in the model at a global level. The comparison of this indicator for the training and test sets – as indicated in Chapter 7 – allows for verification at the stage of model construction – whether it is stable and has not been over-trained. On the other hand, comparing the mean Shapley value for a variable in two different periods makes it possible to assess the presence or lack of stability of the variable in the population. It is also worth noting that in the case of the classic scorecard, the stability analysis of the Shapley value for the variable is exactly equivalent to the stability analysis of the scores. Because for the scorecard, the change in the mean Shapley value for a variable is equal to the difference in the mean score for the variable in the period under study and on the training set.

Apart from the Shapley values themselves, a broader analysis is also important, when together with the indices we consider their standard deviation and average value. The combined analysis of these values allows for a better understanding and more precise interpretation of the result obtained. Therefore, it is also important to periodically verify the stability of standard deviations and averages of the Shapley values of different variables. Similarly, the stability of Permutation Feature Importance indicators for individual variables should be verified. The indicators values themselves may fluctuate but in a stable model, the order of importance of the variables should be stable.

Stability of variables

Apart from testing the stability of evaluations generated by variables, the stability of the distributions and properties of the variables should also be periodically validated. It may happen that the average Shapley value for the variable is stable, but the distribution of the characteristics in the analyzed period is different than in the base period. The most basic method of assessing the stability of variables is to compile the distributions of a given variable in the periods compared. Such a juxtaposition makes it possible to assess the stability of various properties for variables – the distribution itself but also the range of values, the scale of missing data, the presence of outliers or atypical values.

In the case of continuous variables, the comparison should include basic statistics – the range of values, mean, quartiles, standard deviation but it is also worth comparing more indicators, e.g. deciles of distributions. For those purposes, we recommend to use the box or violin plots, which in a condensed way allows for the distribution presentation. Moreover, it is suggested to monitor the distribution of missing values. Table 6.12 presents the statistics for the variable denoting the amount of loan installments for companies from the SME sector, determined in the two compared periods.

Population	min	max	mean	quartile1	median	quartile3
Base_pop	69.44	247 637.11	5 672.19	1 726.63	2 759.38	5 536.64
New_pop	115.98	164 066.22	5 387.98	1 697.66	2690.02	5 448.34

Table 6.12: The statistics of the variable denoting the amount of loan installment in two periods
Source: own work

It can be noticed that despite the significant difference in the ranges of the value of the variable, its main statistics remain very similar in the compared periods. A similar conclusion can be drawn by analyzing the distribution of the variable into deciles – the border values for the deciles are very similar in both periods, as shown in Table 6.13.

Base population		New population	
Values	Observations in %	Values	Observations in %
<1 020	10.09	<1 080	10.05
[1 020, 1 510)	9.99	[1 080, 1 500)	9.93
[1 510, 1 910)	9.94	[1 500, 1 860)	10.11
[1 910, 2 285)	10.04	[1 860, 2 215)	10.11
[2 285, 2 760)	9,94	[2 215, 2 700)	9.93
[2 760, 3 400)	9.99	[2 700, 3 430)	9.93
[3 400, 4 600)	10.09	[3 430, 4 720)	10.11
[4 600, 6 420)	10.04	[4 720, 6 240)	10.05
[6 420, 10 250)	9.99	[6 240, 10 000)	9.99
>= 10 250	9.89	>= 10 000	9.80

Table 6.13: Distribution of the variable denoting the amount of loan installment into deciles in two periods
Source: own work

The PSI index can also be used to assess the stability of variables. In the case of nominal variables, its application is obvious – the percentage shares of individual variable values are compared. One of the values can also be a lack of data. Figure 6.15 summarizes the distributions of a variable that was created by grouping the number of employees for small businesses. After grouping, the variable has three categories: 1-2, 3-5, 6 or more and lack of data (null) was marked as an additional value. Table 6.14 shows the exact distributions of the variable over the two periods and the PSI calculations. For the variable presented in the table, the value of the PSI is 0.11, which means noticeable changes in the structure of this variable[7].

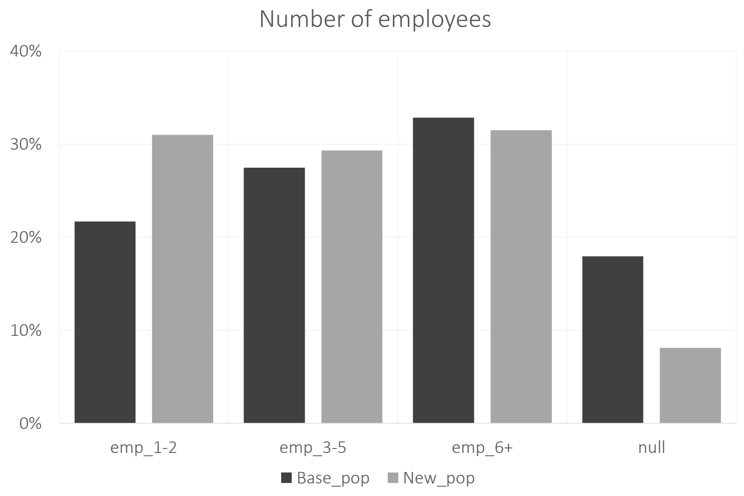


Figure 6.15: Distributions of the nominal variable in two periods
Source: own work

Values	D_i Base_pop (in %)	V_i New_pop (in %)	$V_i - D_i$	$\ln(\frac{V_i}{D_i})$	$V_i - D_i \times \ln(\frac{V_i}{D_i})$
emp_1-2	21.69	31.02	0.09336	0.358022	0.0334
emp_3-5	27.48	29.34	0.01854	0.065274	0.0012
emp_6+	32.87	31.52	-0.01344	-0.04176	0.0006
null	17.96	8.11	-0.09846	-0.79449	0.0782
		PSI		**0.11**	

Table 6.14: The PSI indicator calculations for the nominal variable
Source: own work

[7]Such a change does not always have to be a disadvantage. In the presented example, it probably results from the improvement of the data filling process.

For numerical variable, it is possible to test stability using the PSI indicator if the values of such variable are previously divided into predefined intervals. As an example, the above-mentioned variable denoting the loan installment and its division into deciles for the base population can be used. The decile breakpoints for the base population will be used to split the variable's value for the new population. We get the distributions shown in Figure 6.16. Calculations of the PSI index are presented in Table 6.15.

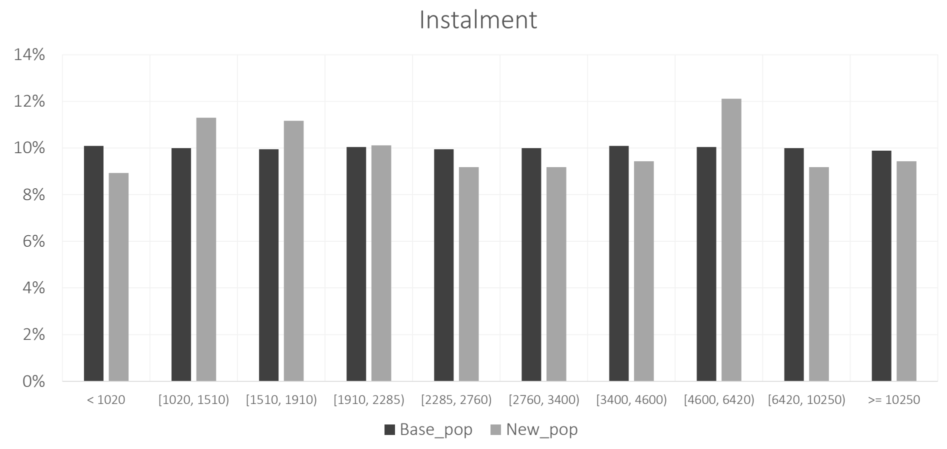


Figure 6.16: Distributions of grouped numerical variable in two periods
Source: own work

The determined value of the PSI indicator for the variable confirms the conclusions from the comparison of the variable's statistics for two periods – the variable is completely stable.

6.4.4 Computational complexity and implementation cost of the proposed methods of model monitoring

Three groups of quantitative analyses conducted as part of model validation or monitoring are presented above. These groups include discriminatory power, calibration correctness and model stability. The described tests and indicators include both simple and complex measures but almost all analyses within a given group have the same computational complex-

Values	D_i (in %) Base_pop	V_i (in %) New_pop	$V_i - D_i$	$\ln(\frac{V_i}{D_i})$	$V_i - D_i \times \ln(\frac{V_i}{D_i})$
<1 020	10.09	8.93	-0.0117	-0.1229	0.0014
[1 020. 1 510)	9.99	11.30	0.0131	0.1231	0.0016
[1 510. 1 910)	9.94	11.17	0.0124	0.1172	0.0014
[1 910. 2 285)	10.04	10.11	0.0007	0.007	0
[2 285. 2 760)	9.94	9.18	-0.0076	-0.0798	0.0006
[2 760. 3 400)	9.99	9.18	-0.0081	-0.085	0.0007
[3 400. 4 600)	10.09	9.43	-0.0067	-0.0684	0.0005
[4 600. 6 420)	10.04	12.11	0.0207	0.1873	0.0039
[6 420. 10 250)	9.99	9.18	-0.0081	-0.085	0.0007
>= 10 250	9.89	9.43	-0.0046	-0.0477	0.0002
		PSI		**0.01**	

Table 6.15: The PSI indicator calculations for grouped a numerical variable
Source: own work

ity. To determine the complexity of individual methods, let us denote the number of observations in the processed set by n.

In the group of analyses related to the study of discriminatory power, the following indicators were described:

- Confusion matrix and its statistics, such as precision, accuracy and others,
- ROC curve,
- AUC ROC and its confidence interval,
- CAP curve,
- Gini coefficient,
- Lorenz curve,
- Lift curve,
- K-S test,

- Divergence for model,
- Divergence for variables,
- Gini for variables,
- Shapley values for variables,
- Permutation Feature Importance.

Among these analyses, only the confusion matrix and its statistics have $O(n)$ computational complexity. All other indicators and curve plots in this group – except Shapley values – require the observations to be sorted by risk assessment (or by variable value when evaluating variables) and have the computational complexity of $O(n \log n)$. Determining the Shapley value is definitely more computationally complex. The algorithm consistent with the indicator definition has a complexity of $O(2^k)$, where k is the number of variables in the model. Therefore, various attempts are made to simplify the calculations, e.g. by performing them on sub-samples (Castro et al., 2017).

The second group of analyses includes calibration tests:

- Binomial test,
- Binomial test with correlation,
- Normal test,
- The Jeffrey test,
- The Hosmer-Lemeshow test,
- The Pluto-Tasche test.

In this group, all tests are based on statistics whose computational complexity is $O(n)$.

The third group consists of analyses related to the stability of the model and variables:

- Migration matrix,

- Stability of migration matrix,
- Concentration in rating grades,
- PSI,
- Stability of variable's score,
  - Characteristic Analysis Report,
  - Stability of Shapley value for a variable,
  - Stability of Permutation Feature Importance,
- PSI for variables.

In this group, almost all analyses, except for the Shapley value and PFI calculation, have a computational complexity of $O(n)$. Determining Shapley values has a computational complexity of $O(2^k)$, where k is the number of variables in the model. Calculation of PFI indicators has a computational complexity of $O(n \log n)$.

Apart from the Shapley value, all other indicators are characterized by low computational cost, even in the case of complex methods. It should be noted that the main computational cost of most methods is the determination of the score value, which is then used in the calculations. The implementation costs are also mainly related to scoring. It can be the most expensive to prepare an appropriate environment in which the assessments will be generated. On the other hand, all the presented analytical methods have their implementation in both commercial and open source environments.

6.4.5 Classic and alternative methods of validating scoring models

The use of non-classical analytical methods to build scoring models is still quite rare due to the requirement of interpretability of the credit decision. However, alternative analytical methods will be increasingly used in risk assessment. This is due to, among others, two factors: the very high efficiency of these models and the dynamic development of methods that allow us to understand and interpret the operation of models that were until recently treated as black boxes. This is confirmed by the EBA report (European Banking Authority, 2020) in which the regulator notes that the increasing use of big data is inextricably linked with advanced analytics and the use of machine learning methods. The use of non-classical analytical methods to some extent also affects the validation of the obtained models.

The first difference is visible already at the stage of preparing and selecting variables for model building. Although the methods used for this purpose do not change significantly, the approach to the issue of variable selection has changed dramatically. One of the elements of data validation at the model building stage was the study of the correlation between the variables and their discriminatory power. Classical modeling methods required the use of variables with high discriminatory power and not correlated with each other, therefore these features were the subject of verification. Non-classical methods can be correctly and effectively applied even with highly correlated variables that have low individual predictive power. Regardless of the methods used, the requirement of the stability of the variables used to build the model does not change.

The use of alternative methods of model construction does not affect the methods of validating the discriminatory power of the obtained model. All the above-described indicators and curves illustrating the discriminatory power are model agnostic and can be effectively applied to any classification model with a binary dependent variable. It is similar to the Gini index for variables. A new element of validation that appeared with the use of non-classical modeling methods are Shapley values and PFI calculations for variables. They show the contribution of individual variables to the discriminatory power of the model.

For the calibration tests, a change is proposed as described in the summary of Section 6.4.1. It consists of replacing the study of statistical significance with the study of the materiality of the problem. This approach was also presented in the ECB document (Bank, 2017). The ECB determines the materiality of the quantitative impact in the portfolio in terms of the Risk-Weighted Assets percentage. This is a new approach in validation methods however it does not result from the use of alternative ML methods. The main reason for the change is the problem with the actual realization of the sample assumptions underlying the statistical tests, e.g. about the independence of defaults in the rating class or the stability of real DR.

The methods of testing the population and model stability basically remain unchanged – they consist of comparing the distributions in different periods. The main change in this group of analyses, which is related to the use of non-classical methods, is the use of Shapley values to verify the stability of the influence of variables on the discriminatory power of the model.

6.5 Additional validation dimensions

The validation and systematic monitoring of credit risk models are essential for proper risk management. However, good test results obtained on internal data can sometimes be misleading. In fact, it may turn out that the risk has been underestimated and, therefore, risk management and the determination of provisions are based on erroneous estimates. This may result from the specificity of one's own portfolio but it may also be the case that incorrect assumptions are made at some stage of the process, e.g. regarding the lack of correlation of defaults. Therefore, it is worth carrying out additional analyses during the risk assessment, such as stress testing and benchmarking which allow for risk management in a broader perspective.

6.5.1 Stress testing

Stress testing is an analysis that reduces the uncertainty associated with risk and regulatory capital estimation. It allows to assess and take into account in risk management the risk resulting from very rare but feasible events that increase the risk. They can be political, institutional or economic events, but always result in a change in economic conditions. The most obvious example of such an event is the economic crisis, i.e. a significant deterioration in macroeconomic conditions, including a decline in GDP growth. However, the list of such events that may affect the portfolio risk is much longer. Such factors can be, for example:

- significant drop on the stock exchange and an increase in market risk,
- an increase in the correlation between borrowers and the concentration of defaults in rating grades,
- increase in the probability of moving to lower risk classes,
- drastic fluctuations in DR,
- legal regulations leading to changes in the business conditions of large borrowers.

Stress testing involves the inclusion in the risk assessment process of scenarios that take into account one or more factors influencing the risk parameters. Banks and regulators found out about the importance of the stress tests during the 2008 crisis. Following these events, the BCBS developed and published in 2009 the set of sound stress testing practices (on Banking Supervision, 2009). In 2018, based on several years of observation of banks' activities in this area, BCBS updated the set of stress-testing principles (on Banking Supervision, 2018).

6.5.2 Benchmarking

Benchmarking is the comparison of internal ratings and risk assessment with external information, such as ratings from rating agencies. The bank may obtain a comparison of risk ratings by using available external ratings for the joint sample or by generating its own ratings for the external sample. The use of external data for benchmarking allows the bank to verify the results of validating the discriminatory power of the model and the correctness of the calibration. Significant discrepancies in the validation results and benchmarking should be carefully analyzed in each case.

Performing benchmarking may not be easy to do. Comparing internal with external ratings requires identifying a common sample for which ratings are available and an appropriate mapping of the ratings to a common master scale. Using an external sample can be just as difficult. It is then necessary to verify not only the quality of the data but also the compliance of the definition of default and all the characteristics. Yet another difficulty is the correct interpretation of the results of the comparison. The high discriminatory power of internal risk models on the bank's own portfolio and, at the same time, the low rating obtained on the external sample are not unequivocally justified. The reasons for this may be different and must always be determined in a specific case. Perhaps these considerations make the issue of benchmarking still not sufficiently methodologically developed. An example of developing and conducting an extensive comparative analysis of models is presented in (Lessmann et al., 2015).

6.6 Conclusions

This chapter presents the process of validating the scoring model, taking into account the common distinction between qualitative and quantitative validation. The construction and maintenance of any model supporting the decision-making process should include an evaluation and verification component. In credit scoring, this component is even more crucial because these models directly impact the institution's profitability and in a susceptible and strictly regulated area (i.e., granting credits).

The procedures described in this chapter apply to both classical models (e.g., logistic regression model) as well as machine learning models. Validation of the scoring model performance, including such main areas as discriminant power assessment, calibration quality assessment, and performance stability, should be the basis of every scoring process, i.e., model development and monitoring.

Chapter 7

Model interpretability and explainability

MARCIN CHLEBUS[1]
MARTA KŁOSOK[1]
PRZEMYSŁAW BIECEK[2]

[1] *Quantitative Finance Department*
Faculty of Economic Sciences
University of Warsaw
[2] *MI2DataLab*
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw

The role of risk in the crediting process is crucial because inappropriate risk assessment may lead to inefficient estimations of risk costs which may cause issues for a bank and in consequence, to the whole banking sector which is one of the fundaments of the economy. In order to avoid such consequences, the Banking sector is deeply regulated by Basel Committee on Banking Supervision (BIS, 2020), local Financial Services Authorities, in European Union additionally by the European Central Bank (ECB, 2020). In case of the credit scoring, regulators expect that discriminatory/predictive power should not be a single measure to assess a model's quality – see Chapter 6. Similar importance should be assigned to the stability and interpretability of the results. Banks are obligated to know how their model works (*know your model* rule) and are required to be able to explain to their customers why a given credit was not granted (a so-called *right to know/explanation* rule) (European Parliament, 2016). In other words, it is expected not only to predict whether the client is less or more prone to default but also to know why a model predicts that.

According to that, the most commonly used model in the field is a logistic regression, see Chapter 4 or (Bolton et al., 2010; Siddiqi, 2012) which equalizes these two aspects. In many researches, it is shown that more flexible machine learning algorithms are providing better predictions, however in respect to their complexity and nested non-linear structure (Samek et al., 2017), they are hardly accepted. Usually, the main argument against flexible algorithms is their lack of interpretability (Bolton et al., 2010).

With respect to that, a trade-off between interpretability and predictability can be easily seen. So-called *white-box models* are easier to understand but may provide poorer predictions and so-called *black-box models* may find more non-linear and less obvious patterns (Bzdok et al., 2018).

Some, such as Rudin (2019), suggest using mostly transparent models, pointing out that they are often highly effective. However, others claim that elastic models are on average more efficient. For example, Louzada et al. (2016), performed a comprehensive comparison of credit scoring methods used in 187 scientific papers form years 1992-2015. In the paper, the ranking of methods was as follows: SVM, fuzzy logic, neural networks, decision trees and logistic regression. In a more recent paper (Lessmann et al., 2015), even more complicated machine learning systems, namely heterogonous (HCES-Bag, GASEN, Averaging) ensemble methods are pointed as the best; logistic regression was ranked 17th from 41 algorithms. Heterogeneous ensemble models are very complex, as they build their prediction based on a combination of homogenous model (single or ensemble).

Their important position has also homogenous ensemble algorithms based on decision trees (Caruana et al., 2008; Caruana & Niculescu-Mizil, 2006). The structures of such algorithms may be very complex, the most popular homogenous ensemble tree-based methods consist of many tree algorithms trained simultaneously or iteratively with different methods of regularisations, penalization, boosting and bagging are: AdaBoost (see Chapter 4, or Freund et al. (1999)), GBM (Friedman, 2001), Random Forrest (see Chapter 4 or Breiman (2001)), XGBoost (see Chapter 4 or Chen and Guestrin (2016)), Light GBM (Ke et al., 2017), CatBoost (Prokhorenkova et al., 2018) or Ngboost (Duan et al., 2019).

According to Hand (2005) and Fantazzini and Figini (2009), using complex algorithms might be unnecessary as gain relative to simpler methods is typically small and is at the expense of interpretability. Nevertheless, what if we could overcome the trade-off between prediction quality and interpretability? It seems that if we could understand how advanced algorithms work, we would also understand better the phenomenon standing behind the data which in turn would help to build a better model. Usually, it is stated that logistic regression or decision tree are easy to interpret because one can easily analyze estimated coefficients or variable splits. And this is true for a small or medium number of parameters, as very deep trees or regression models with a large number of coefficients are much more difficult to analyze.

Interpretable models such as logistic regression or decision tree are stiff which means that they are not able to capture all possible relationships between variables. It is tempting to use more flexible yet complex models that capture more sophisticated relationships. And then to extract knowledge about these relationships from the model. Having this in mind, it may be concluded that a similar approach may be applied to more complicated algorithms and with regards to that, a better understanding of the phenomenon may be revealed. Such an approach is a part of *Explainable Artificial Intelligence* (XAI).

The XAI methods are designed to understand better how the algorithms work and based on that, explain the decisions and predictions, improve systems, debugging and fixing data or models – both on global and single instance level (Barredo Arrieta et al., 2019; Biecek & Burzykowski, 2021).

In XAI toolbox, we may distinguish many different taxonomies for methods (Biecek & Burzykowski, 2021; Molnar, 2019):

1. **intrinsic and post-hoc methods**: intrinsic methods are specific for simple models (logistic regression is interpretable by its design), post-hoc methods concentrate on analyzing the model after its development (usually are based on predictions or residuals) and may be applied to complex models as well.

2. **model-specific and model-agnostic methods**: model-specific methods are designed using properties of a given model, thus are not useful for a thorough comparison between models (for example set of diagnostics tests and interpretation of coefficients as marginal effects for linear regression or goodness-of-fit tests and odds ratio interpretation for logistic regression). Model-specific methods may be fast, as they can use specific structure, properties or underlying assumptions for a given model. Model-agnostic methods do not use any specific properties of models and therefore may be used for all models satisfying basic assumptions (such as including finite number of features and providing a prediction as a single value).

3. **local (instance) and global level methods** : local level methods are designed for interpreting results for a single prediction or set of predictions and global level methods are designed to show how a model works in general.

Different stakeholders are involved in the life cycle of the credit risk model, such as risk managers, model developers, internal and external auditors, credit officers or bank customers. Because different stakeholders have different needs, they will also benefit from different XAI methods. A map showing which stakeholders need which methods was presented by Bücker et al., see Figure 1 in (Bücker et al., 2020). In the following section, we will focus on the needs of data scientists creating and validating credit risk models.

On the instance level, usually the purpose of the XAI methods is to decompose prediction of the model for a given instance with features contributions for each of the component (Hall, 2019). The enormous popularity of the model agnostic local methods began with the LIME algorithms introduced by Ribeiro et al. (2016a). In this method, a simple surrogate model is used to approximate an algorithm in some local area. Even though this approach is used frequently in Image and Text Recognition, its main drawback is that we have to estimate an additional model, instead of interpreting a trained algorithm directly. To overcome this issue, the *SHapley Additive exPlanations* (*SHAP*) values based on the game theory was presented (S. Lundberg & Lee, 2017). Slack et al. (2020), showed empirically that in case of unstable data (e.g. with highly correlated predictors) SHAP is a more reliable method than LIME and currently it is the most popular method for tabular data. To estimate SHAP values, different model-specific methods and model-agnostic methods are proposed. Similar to SHAP and relatively fast method to obtain similar decomposition of a prediction is Break-down proposed by Gosiewska and Biecek (2020). In this approach, instead of calculating the contribution of a variable in every or many permutations, only one order of variables (chosen based on local feature importance) is considered. The SHAP values for the classical scoring models (i.e. scorecard) is a normalized score for a selected feature.

In this chapter, we will discuss two popular frameworks used in the XAI: a) a comprehensive XAI framework for model-agnostic methods – DALEX (Biecek, 2018) and b) comprehensive XAI framework for tree-based model-framework - TreeSHAP values (S. M. Lundberg et al., 2020).

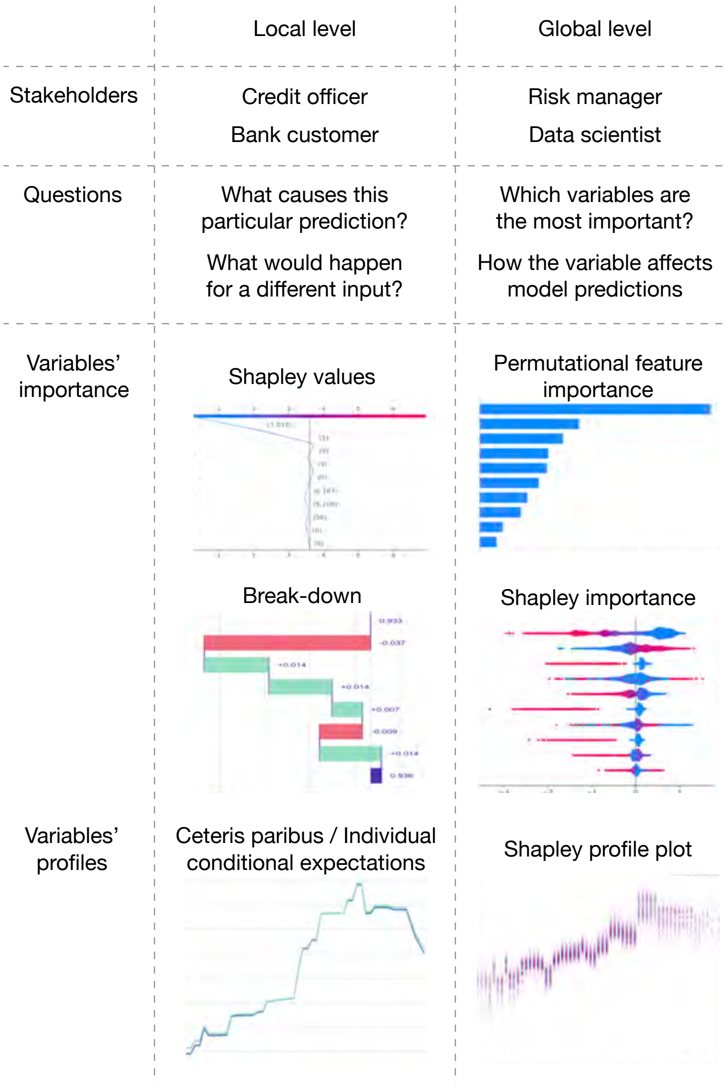

Local level
Global level
Stakeholders
Credit officer
Bank customer
Risk manager
Data scientist
Questions
What causes this particular prediction?
What would happen for a different input?
Which variables are the most important?
How the variable affects model predictions
Variables' importance
Shapley values
Permutational feature importance
Break-down
Shapley importance
Variables' profiles
Ceteris paribus / Individual conditional expectations
Shapley profile plot

Figure 7.1: A summary of the methods discussed in this chapter. We show examples of local and global methods for explanations. Local methods focus on questions asked by a credit officer or a bank customer while global methods are more often of interest to data scientists or risk managers. In Section 7.1, we will discuss Shapley values and the Break-down method, Section 7.2 is for Permutational feature importance, Section 7.3 shows Ceteris paribus profiles while Section 7.4 shows variable profiles
Source: own work

7.1 Shapley values and Break-down

This chapter brings the topic of methods available for explaining Artificial Intelligence[1]. What does it exactly mean that the black-box structure is interpretable for a human or that we can easily explain why the model made a certain prediction? There is no standard criteria and definition. Barredo Arrieta et al. (2019) underline that there is some urgent need for consensus regarding the definition of Explainable Artificial Intelligence. Authors mention also common pitfall when confusing interpretability and explainability terms. While interpretability (often referred as transparency) is related to the structure of the model which can be analyzed through the prism of individual coefficients in a way understandable for a human being, explainability refers to actions of post hoc exploration of models despite the complexity of its internal structure. Once the functioning of a model is understood and inner connections between distinct predictors explored, building trust in the model is the consequence. It is crucial for decision makers in achieving success. Trustworthiness and confidence of the model are essential, as its outstanding performance might be only a matter of a few patterns found in the small subset of data. Interpretable techniques explored in this paper focus only on post-hoc explanations. For their evaluation, a set of explanatory variables, prediction function (learning model generating predictions) and the target outcome is required (Arya et al., 2019).

Shapley values and decomposition of a single prediction

As mentioned in the introduction, the first goal of XAI methods on the instance level is to decompose the prediction of the model for a given instance. Most of the approaches available to do that are connected with Shapley values.

Shapley values are derived from game theory that studies interaction and actions of cooperative games. The payout function is convex which means that it pays off for the player to cooperate. The payout for a group is higher than the rewards they would have received without cooperation.

[1]Note that this chapter is an extended version of the working paper Kłosok and Chlebus, 2020.

However, the problem arises as to how to share the reward in a fair way. The Shapley values set such a fair distribution with good theoretical properties (Hotz, 2006).

In terms of explainable AI, game theory can be considered as follows: for every of k participants (predictors), the payoff is understood as prediction assigned to every possible combination of the players' movements. It is assumed that agents have different strengths in predicting the outcome. For every possible combination of strategies (effects of the features), different payoffs are possible. Shapley values purpose is: for a given set of features and prediction calculated for them, Shapley values calculate the contribution of each feature payoff to the final outcome.

Consider K agents who represent predictors in the data set. $S \subseteq N = \{1, 2, \ldots, K\}$ denotes the subset of agents and their number is set as $|S|$. The $v(S)$ is the function assigning the subset of agents S with a total payoff contributed by co-working. For every possible game (prediction), according to the Shapley formula, ϕ is a function on the fixed set N and $\phi(v) = (a_1, \ldots, a_K) \in R^K$ while a_i denotes the contribution of i-th player to the final output $v(N)$

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N \setminus S| - 1)!}{|N|!} \left[v(S \cup \{i\}) - v(S) \right], \quad i = 1, \ldots, n, \tag{7.1}$$

where $S \subseteq N \setminus \{i\}$ is the subset of all but $i - th$ features from subset S and $v(S \cup \{i\})$ is the function with the $i - th$ feature present (Strumbelj & Kononenko, 2010). The function $\phi_i(v)$ computes in some sense marginal contribution of each player. It is done by summing the effect of each possible coalition of agents drawn from the $N \setminus \{i\}$ set. However, it is done under the assumption that all agents arrive randomly. In order to fairly distribute their power, Shapley function averages the agents expected payoffs over $|N|!$ possible combinations. Another important assumption is that an empty set $v(\emptyset)$ is also valid in the calculations. Shapley assumes that $v(\emptyset) = 0$, as nothing is produced for free (Roth, 1988). In order to better capture the idea, let us consider a set of two players $N = \{1, 2\}$. There are possible four combinations consisting of the players: $\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$. Generally, for subset consisting of N players, there are 2^N possible combinations. Shapley function for each of the $i = \{1,2\}$ player is given by:

$\phi_1(v) = \frac{1}{2}[v(\{1,2\}) - v(\{2\})] + \frac{1}{2}[v(\{1\}) - v(\emptyset)]$,

$\phi_2(v) = \frac{1}{2}[v(\{1,2\}) - v(\{1\})] + \frac{1}{2}[v(\{2\}) - v(\emptyset)]$.

Thus, Shapley value for each of the player is a weighted sum of marginal contributions calculated as the difference of the payoff with and without a given feature.

Roth (1988) states that function $\phi(v)$ has some meaningful properties. The first one is known as efficiency state that total output $v(N)$ is the sum of K players contributions. In normal terms $v(N)$ would denote prediction for a single instance, however for the purpose of Shapley value calculations, it is stated as model outcome minus average model prediction

$$\sum_{i=0}^{N} \phi_i(v) = v(N).$$

The second property – symmetry – assumes that for a given subset of players S and two different agents i and j, so that $v(S \cup \{i\}) = v(S \cup \{j\})$, the function ϕ is symmetric in the sense that it does not depend on the player's name, only on its contribution to the function v. In other words, these two features contributions are equal (17)

$$\phi_i(v) = \phi_j(v).$$

The third axiom of dummy player – for any player i if $v(S \cup \{i\}) = v(S)$ for all $S \subseteq N \backslash \{i\}$ then $\phi_i(v) = 0$. If the contribution of feature i does not influence the prediction, then its Shapley value $\phi_i(v)$ must be zero.

Lastly, Shapley values satisfy additivity. Consider two games and two contribution functions v and w. The gain from combining these two functions is equal to the sum of individual gains for every player i:

$$\phi_i(v + w) = \phi_i(v) + \phi_i(w).$$

Additionally, $\phi_i(cv) = c\phi_i(v)$, where $c \in R$. This last property is also known as linearity (Aas et al., 2019; Štrumbelj & Kononenko, 2014).

Explanations standing behind Shapley values might be compared to the linear model where prediction function is given by

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{i=1}^{K} \hat{\beta}_i x_i.$$

The $\hat{\beta}_0$ denotes the intercept, i.e. average model prediction over data distribution. Shapley values decompose predictions for single instances in a similar manner to linear models, however the former approach is designed so that it is applicable for every class of models. Let x be an instance of interest, $\hat{f}(\bullet)$ prediction function and $\phi_0 = E[f(x)]$. Then prediction for the single data point is the sum of the intercept and K features contributions)

$$\hat{f}(x) = \phi_0 + \sum_{i=1}^{K} \phi_i.$$

The idea behind Shapley values for a given set of predictors is computing the difference between the feature effect and the average outcome. In order to calculate the contribution of predictor i to the score produced by the model for an observation x, it is necessary to derive a formula for the difference when regressor value is not known. The difference in the linear regression model would be an equivalent to feature contribution obtained with Shapley approach

$$\phi_i = \hat{\beta}_i(x_i - E[X_i]) = \hat{\beta}_i x_i - E\left[\hat{\beta}_i X_i\right], \quad i = 1, \ldots, K.$$

However, in practice no such explicit equation for non-linear models exists. In addition to that, linear models often assume no correlation of independent variables (i.e. absence of the multicollinearity problem in the data). In fact, vector of beta coefficients in linear regression might be inflated due to the correlated factors. When calculating the impact of single variables on the model output, we want to take into account the phenomenon of feature dependence. In order to overcome this problem, a general equation 7.1 has been proposed and it is a representation of Shapley value definition. Consider final output $v(N)$ that stands for prediction of a single data point. $|N|$ denotes the number of features for that object. Shapley value for each of the feature i from the set N is defined as $\phi_i(v)$, where v is a value of the feature i.

As computing expression $\phi_i(v)$ with K features requires creating 2^K subsets and increases exponentially with the number of predictors, an alternative approach has been proposed by Štrumbelj and Kononenko (2014). Its computation is based on Monte Carlo sampling. Consider an observation x, number of features in the model K, prediction function $\hat{f}(\bullet)$ and an integer number m denoting the number of simulations. The output of the simulation is the contribution of a given feature j to the final prediction. Strumbelj and Kononenko present the following algorithm for Shapley value of a single predictor[2].

Another popular calculation method of Shapley Values is a SHAP approach (Shapley Additivie Explanation) proposed by Lundberg and Lee (2017). The authors proved that for a given simplified input mapping h_x, there is only one additive feature attribution method that has an explanation model which is a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_j^i,$$

where $z' \in \{0,1\}^M$, M is the number of input features and $\phi_i \in R$ and satisfies three base properties (local accuracy, missingness and consistency) and it is based on Shapley Values.

The SHAP values may be considered as Shapley values of a conditional expectation function of the original model. They are not calculated directly but are approximated. They may be approximated by model-agnostic (using presented above Strumbelj and Kononenko method or Kernel SHAP

[2]The modified version of the presented algorithm, which is especially recommended for the Gradient Boosting algorithms, is the TreeSHAP (S. M. Lundberg et al., 2018). This variant of the algorithm records the number of subsets of observation that flow into each of the node of the tree.

Algorithm 1: Shapley values $\phi_j(x)$ for a single predictor (x).

```
for 1, ...., m do
    Pick a random observation $w$ from the data set;
    Pick a random permutation $O \epsilon \pi(n)$, where $\pi(n)$ is set of
      ordered permutations of the feature indexes;
    Create two new instances $b_1$ and $b_2$ based on $x$ and $w$:
        • reorder features in $x$ and $w$ according to indices in $O$:
          $x' = (x_1,\ x_{2,}\ \ldots,\ x_j, \ldots,\ x_{n)}$ and
          $w' = (w_1,\ w_{2,}\ \ldots,\ w_j, \ldots,\ w_{n)}$
        • create $b_1 = (x_1,\ x_{2,}\ \ldots,\ x_j,\ w_{j+1}, \ldots,\ w_{n)}$ and
          $b_2 = (x_1,\ x_{2,}\ \ldots,\ w_j,\ w_{j+1}, \ldots,\ w_{n)}$
    Instance $b_1$ is created by taking feature values of $x'$ for
      $i = 1, \ldots$,j and features values of $w'$ for $i = j+1, \ldots, n$.
      Analogously, instance $b_2$ takes feature values of $x'$ for
      $i = 1, \ldots, j-1$ and features values of $w'$ for $i = j, \ldots, n$;
    Calculate $\phi_j^m(x)$ of the $m$th iteration from the formula:
    $\phi_j(x) = \hat{f}(b_1) - \hat{f}(b_2)$
end
Shapley value is the average $\phi_j(x) = \frac{1}{m}\sum_{i=1}^{m}\phi_j^m(x)$.
```

Source: own work

method) or model-specific (Linear, Low-Order, Max, Deep or Tree SHAP) methods. The Kernel SHAP method consists of 5 steps, (Molnar, 2019).

Among model-specific approaches, the approach for tree-based algorithms seems to be the most important, as those algorithms provide very often the best results, especially for tabular data (used in credit scoring, as well). S. M. Lundberg et al. (2020) proposed a TreeExplainer method that provides an exact calculation of Shapley Values in tree-based models in polynomial time. What is more, it also captures directly feature interactions and provides a new set of tools for understanding global model structure based on Shapley Values and hence creates a whole XAI framework for tree-based models.

Shapley values used for the purpose of local model explanations are very similar in their use and functionality to local surrogate models implemented

Algorithm 2: The Kernel SHAP method

```
1. Sample coalitions $z'_k \in \{0,1\}^M, k \in 1,\ldots,K$, where $M$
   is a maximum coalition size, 1 = feature present in coalition,
   0 = feature absent;
2. Get a prediction for each $z'_k$ by first converting $z'_k$ to the original
   feature space and then applying the model $f : f(h_x(z'_k)); h_x(z'_k)$
   maps original value for features in coalition and randomly
   selected value from data for absent features.;
3. Compute the weight for each $z'_k$ with the SHAP kernel; the
   weight is calculated as:
       $$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z\prime|}|z\prime|(M-|z'|)},$$
   where $|z'|$ is a number of present features in instance $z$.;
4. Fit weighted linear model optimizing the loss function:
       $$L(f,g,\pi_x) = \sum_{z' \in Z} [f(h_x(z\prime) - g(z')]^2 \pi_x(z')$$
   5. Return Shapley values $\phi_k$, the coefficients from the linear
   model.;
Source: own work
```

in LIME. By contrast to surrogate models that assume linearity of the classifiers, the concept deriving from coalitional game theory has a strong mathematical background. Efficiency, symmetry and additivity axioms assure that for each data points features contributions are distributed in a *fair way*. Feature contributions for all regressors must be calculated so that the sum of effects and the average adds up to the local prediction. Moreover, if a feature is not present among regressors, its Shapley value is surely 0.

Shapley values are not only powerful and reliable tools for local model inspection but can be aggregated into useful statistics for global model inspection. By summing features contributions for all of the instances and averaging the effect, obtaining marginal contribution is possible. Visualizing the Shapley values with respect to a chosen predictor value and obtaining partial dependence plot is also possible. One has to bear in mind that once calculating feature contribution ϕ_i, all explanatory variables are taken into consideration. Explanations produced by local surrogate models often allow to specify the number of predictors for the user. Comparing to LIME procedure, estimating K predictors requires 2^K combinations of payoffs. Calculations of feature effect would be time-consuming even for one observation, not to mention the whole data set. There are some concerns regarding sampling feature values when calculating payoffs with and without the player. In order to predict the model outcome, the missing regressor must be randomly sampled from the data. Slack et al. (2020), raise the issue that both LIME and Shapley values computed in different packages are perturbation-based methods whose results are always biased in some way. However, in order to overcome this problem, the algorithm based on the Monte Carlo simulations presented in the Štrumbelj and Kononenko might be executed with sufficiently high parameter m (number of simulations – which will on the other hand increase further the computational time).

A similar approach to decompose single prediction is a Break-down. It was proposed by Gosiewska and Biecek (2020). Its general idea is to start with an empty set and add all of the features one by one and calculate its average contribution to the prediction. Features are added respectively to their predictive power. This model-agnostic approach for explaining single predictions is presented in the algorithm below. It considers only the step-up approach when calculations begin with an empty set and the features are added. The step-down approach also exists and requires calculating the smallest distance between the full set of variables and the feature removed from it.

Once again, let us consider a set of k features and an observation x. Define V as an empty set storing variable indices. Features' indexes are added one by one depending on their local importance. For the purpose of that, it is necessary to define a function calculating the distance between prediction for original instance x and an instance with features from V

$$d\left(x, V\right) = \left|\hat{f}\left(x\right) - \hat{f}\left(x\left(V\right)\right)\right|.$$

The $\hat{f}(x(V))$ term measures prediction for x with feature values fixed on variables from V. For the rest of the features, it is assumed they follow the population distribution, conditional on V.

Algorithm 3: Calculation of Break-down values.

```
$V \leftarrow \emptyset$;
**for** $i$ *in* $\{1, \ldots, K\}$ **do**
    **for** $j$ *in* $\{1, \ldots, K\} \setminus V$ **do**
        For the observation $x$ calculate the distance with the
          feature $j$ added to set $V$ as
            $$d(x, V \cup \{j\}) = \left|\hat{f}(x) - \hat{f}(x(V \cup \{j\}))\right|.$$
        Choose the feature that maximizes the distance and denote
          is as $j_{max}$;
        Calculate contribution of the feature $\phi_i(x)$ as difference
          of prediction made with the feature $j_{max}$ and without it:
            $$\phi_i(x) = \hat{f}(V \cup \{j_{max}\}) - \hat{f}(V).$$
        Add the feature $j_{max}$ to the $V$;
    **end**
**end**
```

Source: own work

Variable contributions in this procedure are ordered with their decreasing predictive power. The algorithm chooses the most powerful feature at first and then adds iteratively less important predictors, (Gosiewska & Biecek, 2020). The main drawback of this method is that contribution of the consecutive variable is dependent upon previously added predictors. This result might be upset in the case of strongly correlated predictors (i.e. similar effect as in the stepwise method described in Chapter 3).

In DALEX model-agnostic, methods for calculation SHAP and Break-down (with and without interactions), values are available (Biecek & Burzykowski, 2021), whereas in SHAP (S. M. Lundberg et al., 2020), a set of model-specific and model-agnostic methods for SHAP values are available. What is more, in SHAP framework, additional local and global analyses are available based on SHAP values. Similar model-agnostic methods are available in DALEX and will be described later in the text. On the local level, the main methods are:

1. SHAP decomposition plot which is a plot showing how a final predication for an instance stimulated by different feature values.

2. SHAP force plot for an instance which is another plot showing features contribution to the final prediction for a single observation.

On the global level, the most useful methods are:

1. Features Importance Plot with mean of absolute SHAP values (similar method to PFI but based on SHAP values).

2. Local explanation summary that shows distributions of SHAP values for all the features.

3. SHAP dependence plot which shows how a single feature (or interactions of features) affects the output of the model (similar to PDP/CDP/ALE plots).

4. SHAP force plot for all instances showing groups of instances that are affected similarly by different features' values.

The SHAP framework for tree-based models seems to be an effective choice however, if a comparison between the algorithms for different families is needed SHAP, values calculation may be time consuming. To be able to efficiently compare algorithms from different families, DALEX framework seems to be appropriate. Model-agnostic methods available in DALEX framework are described in the next sections.

7.2 Permutation Feature Importance

Variable importance was first introduced by Breiman (2001), in a random forest algorithm. Further research was done to propose a model agnostic tool for calculating the contribution of individual features into prediction accuracy based on the Breiman approach (A. Fisher et al., 2019). Variable importance for feature i is calculated by randomly permuting the feature X_i and computing increase a selected loss function, i.e. mean prediction error, on the newly created data.

Denote a vector of K features $X = (X_1,\ X_2, \ldots,\ X_K)$, Y is an unidimensional vector of true outcomes (dependent variable). Then, the dataset with n observations $D_n = \ \{(x_1,\ y_1),\ (x_2,\ y_2), \ldots,\ \ (x_n,\ y_n)\}$ is a training set for the machine learning model. Once the prediction function of a model $\hat{f}\,(x)$ is obtained, prediction error of the algorithm is denoted as $R\left(\hat{f}\right)$. For example, for mean square error it would be $R\left(\hat{f}\right) = \ E[(\hat{f}\,(X) - Y)^2]$. Instead of compounding term $R\left(\hat{f}\right)$ – absolute error, root mean squared error or any other metric such as R-squared, adjusted R-squared, accuracy, Gini measure or any other metric incorporating predicted and true outcomes might be used in order to calculate the performance of the model.

Let us suppose an ensemble learning algorithm (e.g. random forest) composed of n trees is trained. Hence, the following estimator is proposed, where $\mathcal{L}$ is a selected loss function such as AUC, RMSE or other, $\bar{D}$ denotes a validation set and n is the number of observations in the validation sample

$$\hat{R}\left(\hat{f},\ \bar{D}\right) = \ \mathcal{L}\left(\frac{1}{n_{tree}}\sum_{t=1}^{n_{tree}} \hat{f}_t\,(X_i)\,, Y_i\right).$$

Function returning prediction error is an aggregation of predictions returned by all of the ensemble tree estimators $\hat{f}_1, \hat{f}_2,\ \ldots,\ \hat{f}_{n_{tree}}$ into the average value. Alternatively, for a model consisting of a single structure (e.g. logistic regression) n_{tree} is set to one and estimator $\hat{R}\left(\hat{f},\ \bar{D}\right)$ can be derived in a similar manner.

Once an error measure function and original model error is obtained, it is possible to formally define Permutation Feature Importance (PFI) of variable X_i. Given an original learning sample D_n, permute a feature X_i by shuffling its values randomly. Instead of rearranging the values, creating a new variable with a given distribution is also acceptable. After permuting the variable once again, calculate the prediction error with this newly created data set denoted as D_n'

$$PFI(X_i) = \hat{R}\left(\hat{f}_t,\ \bar{D}_n^{t'}\right) - \hat{R}\left(\hat{f}_t,\ \bar{D}_n^{t}\right).$$

Finally, the variable importance for predictor X_i is the difference between the prediction error of the estimator calculated with the shuffled data and prediction error with the original data. In order to compare the significance of all training features, $PFI(X_i)$ should be computed for all of $i = 1, \ldots, K$ features and sorted by descending values. The above formula assumes that there are n_{tree} tree-based algorithms that generate the ensemble model, so in order to obtain the contribution of a single variable to the model score, we must sum up the increase in prediction error for all member estimators and take the average value (Gregorutti et al., 2017).

There are some concerns regarding this method. Firstly, the permutation of predictor values is random in every iteration. As a consequence, shuffling the feature for the first time might result in different importance than for the second time. The solution for eliminating this randomness component might be performing the action many times and taking the average result. Plotting the standard deviation of PFI coming from different calculations is also desirable. Secondly, variable significance can be measured trustworthily only if the predictors are independent of each other. However, it is common in practice that independent variables are strongly correlated. This doubt was raised by Toloşi and Lengauer (2011) who noticed that permutation feature importance depends strongly on the correlation between predictors. They conclude that features belonging to a larger group of correlated features receive smaller importance weights even though they might significantly correlate with the dependent variable (similarly to the SHAP value, in case of the particular feature present in the model). Lastly, consider the learning algorithm performing very well on the training sample. It can be so good that we can speak of overfitting to the data. Such a machine learning structure might be missing some data properties when tested with the new data sample and PFI calculated on the testing set might be completely different than PFI with the training set. Mentch and Hooker (2016) raise arguments against this technique in more detail. They also mention two meaningful alternatives. The first one suggested primarily by Strobl et al. (2008), involves permuting predictor taking into consideration distribution conditional on the other regressors. The second method studied by Lehmann and Romano (2006), is leave-one-covariate-out (LOCO). In line with LOCO, the Feature Importance should be calculated by measuring the predictor error after calculating the model without feature of interest. An important difference affecting the calculation time is that LOCO needs to re-train the model again for each removed variable which can be time-consuming for complex models. Nevertheless, all these methods seem to produce comparable results (Mentch & Hooker, 2016).

7.3 Ceteris Paribus Plot/Individual Conditional Expectation

Ceteris Paribus Plot (CPP) or Individual Conditional Expectation (ICE) is designed for inspecting single observations of the model. For model response function $\hat{f}$ and observations set $\{(x_{i,S},\ x_{i,C})\}_{i=1}^{n}$, the n denotes the data points and a predictor x_s, $\bar{f}_{ICE}\ (x_{i,S})$ calculates predicted outcome of the model keeping x_C fixed and changing the feature of interest x_s (x_c is different for all observations). It requires plotting prediction as a function of x_S conditional on observed x_C. As a result, relation between the prediction and predictor is shown for N observations – taking an average of the response results in Partial Dependence Plot (which would be discussed later on).

In terms of comparing the curves among many instances of the empirical data, it might be desirable to set up a centered CPP/ICE curve. It is helpful especially in a situation when observations start with different predictions and it is hard to capture the differences between them. Centered CPP/ICE assigns a constant starting prediction for all instances (centers the observations in the anchor value x^*) and then plots the curve with respect to the following equation (Goldstein et al., 2015):

$$\bar{f}_{\,ICE.CENT}\,(x_{i,S}, x_{i,C}) \;=\; \bar{f}_{\,ICE}\,(x_{i,S}, x_{i,C}) \;-\; \bar{f}_{\,ICE}\,(x^{*},\ x_{i,C})\,,$$

where i denotes the observation index, so $\bar{f}\,^{(i)}_{ICE.CENT}$ presents formula for single centered ICE curve. The x^* is the anchor point, it might be either minimum or maximum value of x_S or another value. Goldstein et al. (2015), assures that in case x^* is set to minimum of x_S, then all centered ICE curves are anchored at 0. This approach enables comparing predictions for single instances versus predictions for some fixed feature value.

7.4 Partial Dependence Plot

Q. Zhao and Hastie (2019), state that Partial Dependence Plot (PDP) depicting marginal effect of an input variable and the model outcome is a black-box visualization tool most commonly used. This approach was primary introduced by Friedman (2001). With PDP relation between a model outcome and the feature values might be visualized on the plot and better understood. The PDP assumes the X_i feature constant while averaging model predictions (i.e. it shows how the predictions partially depend on features). This toolbox is especially useful when the direction of regressors influence on the model outcome is unknown and its assessment is required.

Writing the procedure formally, let $X = (X_1,\ X_2, \ldots,\ X_K)$ represent a vector of K variables and there are n observations in the learning sample. Suppose x_s represents an interest set for which PDP is to be calculated and x_c is its complement. Complement x_c refers to the observations not included in the set x_s , so $x_c = x \backslash x_s$. Prediction function of a model on data set x is denoted as $\hat{f}\,(x)$. Partial dependence function on a set x_s is defined as:

$$f_{PDP}\,(x_s) = E_{x_c}\left[\hat{f}\,(x_s,\ x_c)\right] = \int \hat{f}\,(x_s,\ x_c) dP\,(x_c)\,,$$

where $dP\,(x_c)$ refers to the marginal distribution of x_c. Since it is not precisely known, Partial Dependence might be estimated with the formula:

$$\bar{f}_{PDP}\,(x_s) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}\,(x_s,\ x_{i,c}).$$

What this method does is simply averaging over features values keeping x_s constant (not conditional on x) (Greenwell, 2017)).

The idea behind constructing the PDP estimator is rather simple. Suppose we have the training set $D_n = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. The first step in the calculations involves replacing all of the data points from the set D_n with the value for the first observation of that feature. Afterward, predictions with newly created data set are obtained and the average value of the predictions is calculated. The procedure is repeated by replacing all of the feature values with the second observation value and again an average prediction is obtained. Suppose the number of unique values for predictor is equal to a number of data points n. The procedure is repeated then n times, until all average effects are obtained. Finally, plotting average values calculated for all of k-feature values is possible.

Partial Dependence Plots are most popular for covariates in a one-dimensional space because plotting them for more dimensions is difficult to analyze, although 2-dimensional PDP visualizing interactions between explanatory variables are popular too. Friedman (2001) suggests that with a large number of predictors, it is recommended to have a measure of relevance such as permutation feature importance because selecting a couple of useful variables and their combinations might be time-consuming. Partial Dependence is also computationally expensive. Consider a feature j, number of data points n and number of grid points m. For every predictor, it is required to make $n * m$ predictions. Supposing there are hundreds of explanatory variables, it is reasonable to select a couple of the most desirable features for visualization. Reducing computational burden is possible with setting a suitable grid of points for evaluation of profiles. Instead of selecting the whole range of possible values for a feature, one can use quantiles of the feature distribution for PDP calculation.

Certainly, PDP shows the relation between model outcome and input variable both for categorical and continuous variables. The former is possible to obtain by simply plotting a single feature values against $\bar{f}_{PDP}(x_s)$. For the latter, the PDP might be presented as a bar plot, where each bar denotes the unique categorical value and height of the bar corresponds to the

PDP function value. For many unique values of predictor, it might be computationally expensive to calculate Partial Dependence estimator, in which case we recommend binning (see Section 4.2.4) based on the percentiles.

Comparing PDP with CPP/ICE curve might bring interesting insights into trained algorithms. As Friedman (2001) noticed, Partial Dependence might be reliable only in case of weak dependence between predictors. Balancing these two methods and their results against each other can help to detect interactions between the features. It might be true especially where CPP/ICE curves for many instances are significantly different than the average of them (Partial Dependence Plot).

Similarly as with PFI, correctness of this method might be questioned in case of the highly correlated features, i.e. *dilution* of the impact of correlated variables problem or the averaging within the non-existent combinations of variables (e.g. 20 years old with 20 years of work experience). The second problem is not present in the ALE or SHAP. Fortunately, an alternative method to partial dependence was proposed. *Accumulated Local Effects* (*ALE*) takes the correlation bias into account and, with slight modifications, calculates average predicted outcome with respect to the predictor value. Another problem with PDP is that $\bar{f}_{PDP}(x_s)$ is computed over marginal distribution of other features. This inflates the results because observations with hardly probable numbers are created, thus creating fake and unrealistic data and inflating the shape of the Partial Dependence curve.

Alternatively, instead of marginal density, a conditional distribution might be used. Marginal Plot was proposed in this manner. The function calculating MP values takes the form

$$f_M(x_s) = E_{x_c|x_s}\left[\hat{f}(X_s,\ X_c)\middle|X_s = x_s\right] = \int \hat{f}(x_s,\ x_c)dP(x_c|x_s).$$

Analogously, the estimator of this formula is obtained

$$\bar{f}_M(x_s) = \frac{1}{n(x_s)} \sum_{i \in N(x_s)} \hat{f}(x_s,\ x_{i,c}).$$

In order to calculate $\bar{f}_M$ it is necessary to define some neighborhood of the feature of interest x_s. $N(x_s) \subset \{1, 2, \ldots, K\}$ is the subset of observations where $x_{i,s}$ is in close neighborhood of x_s and $n(x_s)$ denotes the number of those instances (Apley & Zhu, 2016). Marginal Plot resolves the problem of incorporating marginal distribution, however does not account for correlation bias, as averaging still incorporates dependence between two features. The solution for this phenomena was proposed by Apley and Zhu and is named ALE plot.

For a given predictor, Accumulated Local Effects calculate average changes in prediction for observations in close neighborhood to the original one. Formally, ALE requires computing following integral (for non-differentiable cases we refer to the source paper (Apley & Zhu, 2016))

$$f_{ALE}\left(x_s\right) = \int_{x_{min,x_s}}^{x_s} E_{x_c|x_s}\left[\widehat{f^1}\left(X_s,\ X_c\right)\middle| X_s = z_s\ \right] dz_s - constant =$$

$$= \int_{x_{min,x_s}}^{x_s} \int \widehat{f^1}\left(z_s,\ x_c\right) P\left(x_c|z_s\right) dx_c dz_s - constant,$$

where $\widehat{f^1}\left(x_s,\ x_c\right) = \frac{\partial \hat{f}(x_s,\ x_c)}{\partial \hat{f}(x_s)}$ denotes the first derivative – change of $\hat{f}\left(x_s,\ x_c\right)$ with respect to a small change of x_s. When calculating the local effect of x_s ALE accumulates the effect of the prediction for a given predictor with the gradient.

For this purpose, dividing the feature of interest in many intervals is needed. Intervals are designated as $N_j\left(k\right)$ and number of instances falling into k-th interval is denoted as $n_j\left(k\right)$, where $j = 1, 2, .., d$ denotes the feature index and $k = 1, 2, .., K$ denotes the interval number. Ranges of intervals are set with the following notation: $N_j\left(k\right) = (z_{k-1,j}, z_{k,j}]$. Authors Zhu and Apley state that $z_{k,j}$ are chosen as $\frac{k}{K}$ quantile of the data distribution. Subsequent data points are denoted as $x_{i,j}$, where $i = 1, 2, .., n$. Observations in all intervals sum up to the total number of observations, so $\sum_{k=1}^{K} n_j\left(k\right) = n$. Additionally, let $k_j\left(x\right)$ indicate the range index number where a given observation x falls, $z_{0,j}$ is set in close neighborhood to the smallest observation and just below it, whereas $z_{K,j}$ is equal to the largest $x_{i,j}$

$$\bar{g}_{ALE}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{\{i:\ x_{i,j}\ \in N_j(k)\}} \left[\hat{f}(z_{k,j},\ x_{i,}) - \hat{f}(z_{k-1,j},\ x_{i,})\right],$$

for each $x \in (z_{0,j},\ z_{K,j}]$, where $x_{i,}$ denotes the observation with subscript i including all predictors but j. It is possible to obtain a formula for centered local effect:

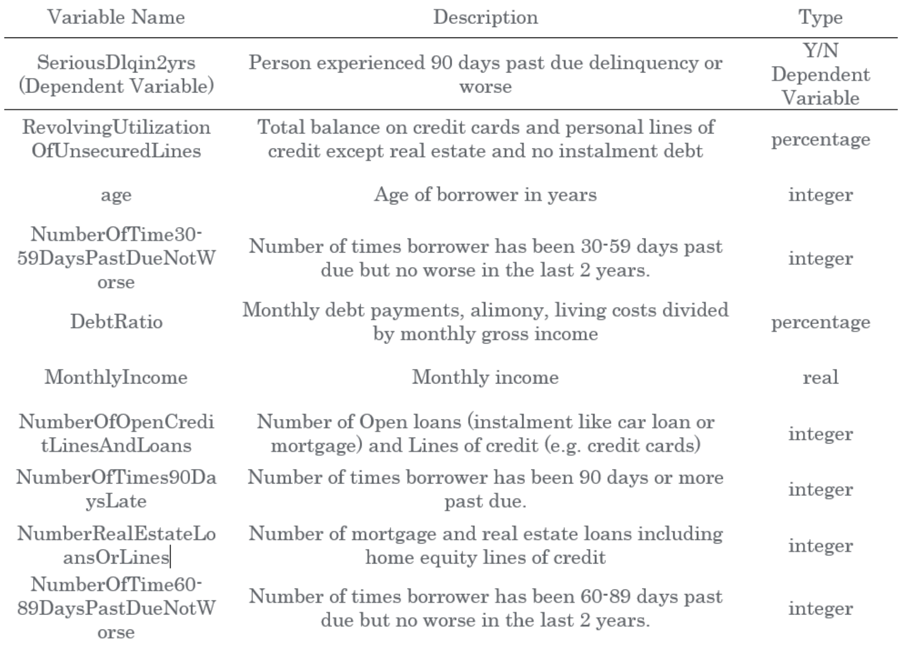
$$\bar{f}_{ALE}(x) = \bar{g}_{ALE}(x) - \frac{1}{n}\sum_{i=1}^{n} \bar{g}_{ALE}(x_{i,j}) = \bar{g}_{ALE}(x) - \frac{1}{n}\sum_{k=1}^{K} n_j(k)\, \bar{g}_{ALE}(z_{k,j}).$$

The $\frac{1}{n}\sum_{i=1}^{n} \bar{g}_{ALE}(x_{i,j})$ in the aforementioned equation accounts for average accumulated local effect for prediction, so the centered local effect measures how for a given predictor the prediction decreases or increases comparing to the prediction on average (Apley & Zhu, 2016).

Creating partitions of data based on $\frac{k}{K}$ quantile of the data is easy for continuous variables, as these predictors often have some predefined order and dividing them into ranges is obvious. For categorical data, especially without exact hierarchy (for example marital status), computation of ALE effect of the predictor is possible, however it is necessary to somehow define the order of values. As ALE plot for categorical features computes the effect across some definite intervals also, it is necessary to define some hierarchy of the feature levels, so that the effect can be accumulated through the categories and transmit information about the model behavior consistently with human logic.

7.5 An empirical example

In order to show how the XAI methods presented above may be used to better understand and explain trained the credit scoring model, an XGBoost algorithm with a logistic link function was trained on the *Give Me Some Credit* data set obtained from the `Kaggle.com`. In Table 7.1, all available independent variables and dependent variable (SeriousDlqin2yrs) are briefly described. The data set contains 150 000 observations with a default rate equal to 6.7%.



Variable Name	Description	Type
SeriousDlqin2yrs (Dependent Variable)	Person experienced 90 days past due delinquency or worse	Y/N Dependent Variable
RevolvingUtilization OfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no instalment debt	percentage
age	Age of borrower in years	integer
NumberOfTime30-59DaysPastDueNotW orse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years.	integer
DebtRatio	Monthly debt payments, alimony, living costs divided by monthly gross income	percentage
MonthlyIncome	Monthly income	real
NumberOfOpenCredi tLinesAndLoans	Number of Open loans (instalment like car loan or mortgage) and Lines of credit (e.g. credit cards)	integer
NumberOfTimes90Da ysLate	Number of times borrower has been 90 days or more past due.	integer
NumberRealEstateLo ansOrLines	Number of mortgage and real estate loans including home equity lines of credit	integer
NumberOfTime60-89DaysPastDueNotW orse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years.	integer

Table 7.1: Description of dependent and independent variables in the *Give Me Some Credit* data set.
Source: Data set "Give Me Some Credit" downloaded from Kaggle.com

In the modeling process, the data was divided randomly with 50% ratio into training and test samples and an XGBoost model was trained with number of iteration set to 70, max single tree depth to 3 and a learning rate to 0.3 (all other hyper-parameters were set on default values). Obviously, the hyper-parameter tuning process may improve the quality of the model. However, the trained model had good enough discriminatory power to be used in an example to show how the XAI frameworks help to understand how a complex model works. Note that here, the Gini value equals to 0.83 on the training sample and 0.71 on the testing sample. Moreover, the Gini 0.71 corresponds to the AUC equalling 0.855 which is close to being the best submission on Kaggle. For the sake of convenient interpretation of Shapley values, a probability of not being default was modelled, reverting common practice makes positive SHAP values interpretation as being a better client (less probable to default).

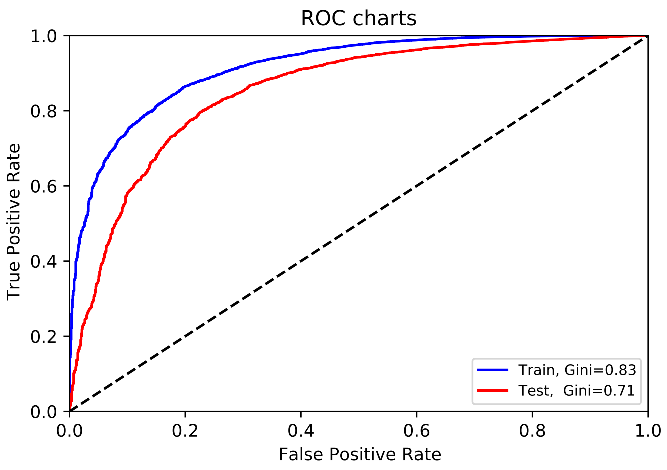


Figure 7.2: ROC curves for the XGB model on train and test sample. A global level explanation
Source: own work

7.6 Instance level

Stakeholders involved in the model development process can use XAI methods in different ways. In this section, we take the perspective of data scientists who create and validate the models (Bücker et al., 2020). From this perspective, on the instance level, two main important questions usually raise:

1. What was the main cause of model prediction? and
2. What would happen if the client would change somehow?

In the SHAP framework, the answer for the first question using the SHAP decomposition plot may be found, while in DALEX framework, the answers for both questions may be found. For the first, using Break-down or SHAP plots, for the second – a Ceteris Paribus Plot.

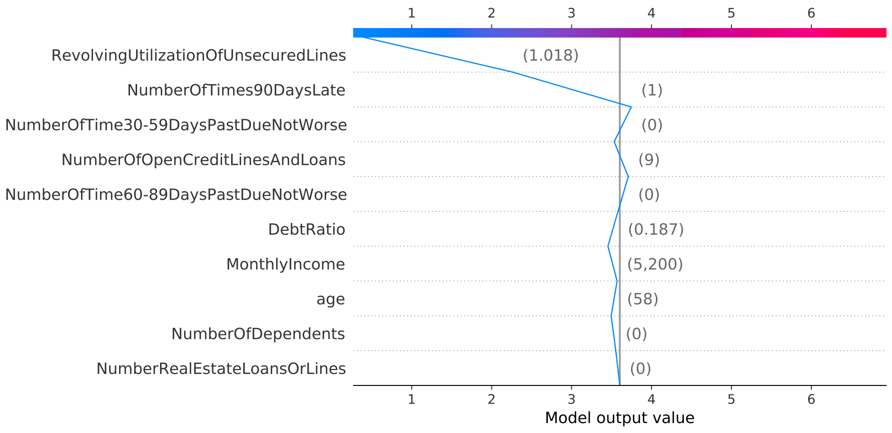


Figure 7.3: Shapley values for a single client. Calculated with the SHAP library. A local level explanation
Source: own work

Figure 7.3 presents SHAP decomposition plot for a given client. In the figure, the black horizontal line shows an expected value from a model (in log-odds scale). On the y-axis, all the features in contribution order are presented. A line showing contribution of each independent variable is drawn (color of the line depends on the final prediction, if it is negative, the color is blue, if the prediction is positive, the color is red). The length of the line between two features shows a contribution of a given variable. In the brackets, a value of a given feature for a given instance is presented.

In the example, it can be seen that two variables that affect the prediction most are Utilization of Revolving Limit (1.018) and Number of times that client was 90 days late (1). In both cases, the contribution is negative (client is riskier), which is in line with expectation, as the client has used more than 100% of available revolving limit and has bad experience in being late with payment of obligations.

In DALEX, two different plots may be used to understand how the features affect the prediction of the model for a given client. Figure 7.4 presents a Break-down plot and Figure 7.5 an SHAP plot (note that a different client is analyzed in the figures from DALEX and then for SHAP example). The Break-down plot shows how given values of features move

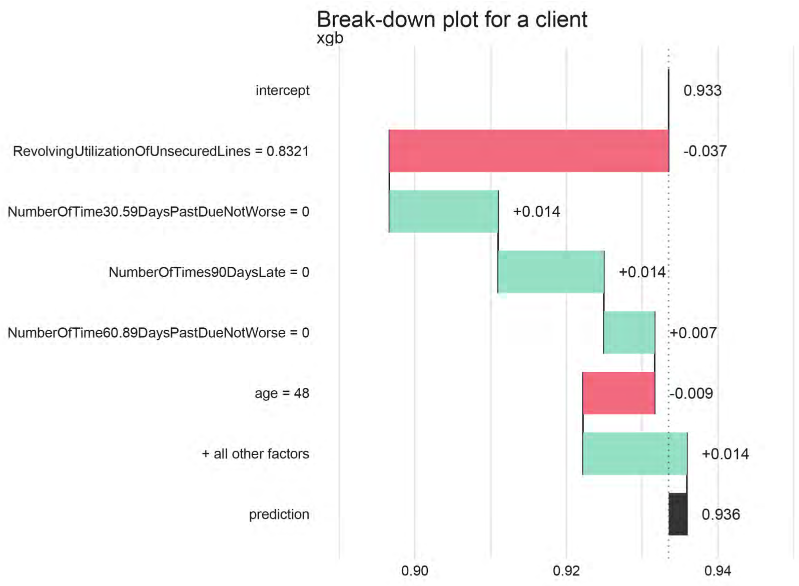


Figure 7.4: Break-down values for a single client. Calculated with the DALEX library. A local level explanation
Source: own work

the prediction from the sample average prediction (93.6% – the probability of being good) to the final prediction (note that probability scale is used). Features are ordered with respect to their local importance. For the analyzed client, the most important feature was again the Utilization of Revolving Limit (86% – relatively high utilization has a negative effect on the prediction), the next three features are telling us that client has no due experience and therefore they move the prediction toward higher probability of being a good client (all equal to 0).

In the DALEX framework, an additional Ceteris Paribus Plot is available (Figure 7.6) which is designed for what-if analysis for a given client. In the figure, a red line shows how the probability of being good would change if the client had a different number of open loans and lines of credit, keeping all other variables constant (*ceteris paribus*). The blue dot shows where is the real number of open products for the client. It can be seen that the client has around 10 open credit products, it may be also noticed that the probability of being good wouldn't change significantly if the client

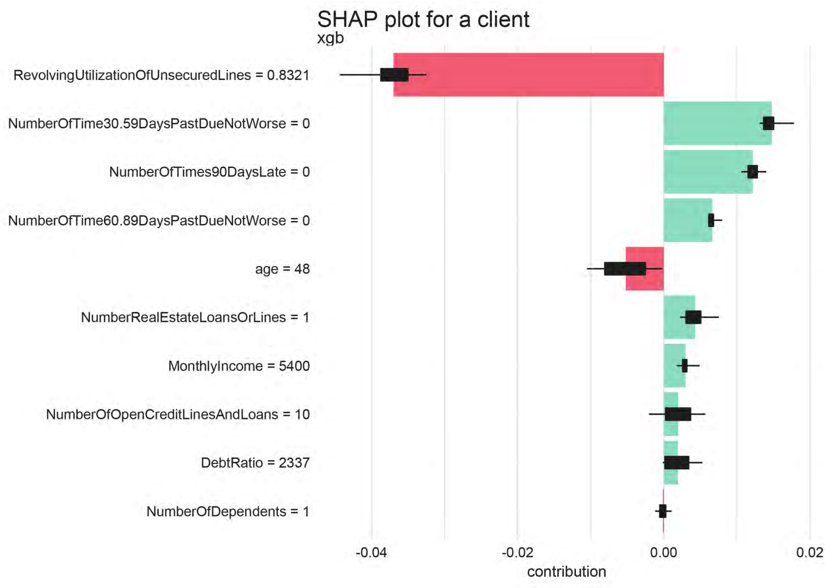


Figure 7.5: Shapley values for a single client. Calculating the DALEX library. A local level explanation
Source: own work

closed his/her products or opened a few more of them. Only after 20 credit products opened, we may expect a negative trend in probability of being a good client.

7.7 Global level

The XAI methods are also very useful to analyze a model on the global (whole model) level. Two main commonly used methods are feature importance and partial dependence plots. The first one helps to understand the role of all features in the model prediction, the second gives us an understanding how changes in a given feature affect the prediction of the model.

Both presented XAI frameworks contain methods for feature importance and partial dependence plots. However, in the SHAP framework, except for the Feature Importance Plot, the Local Explanation Plot is also

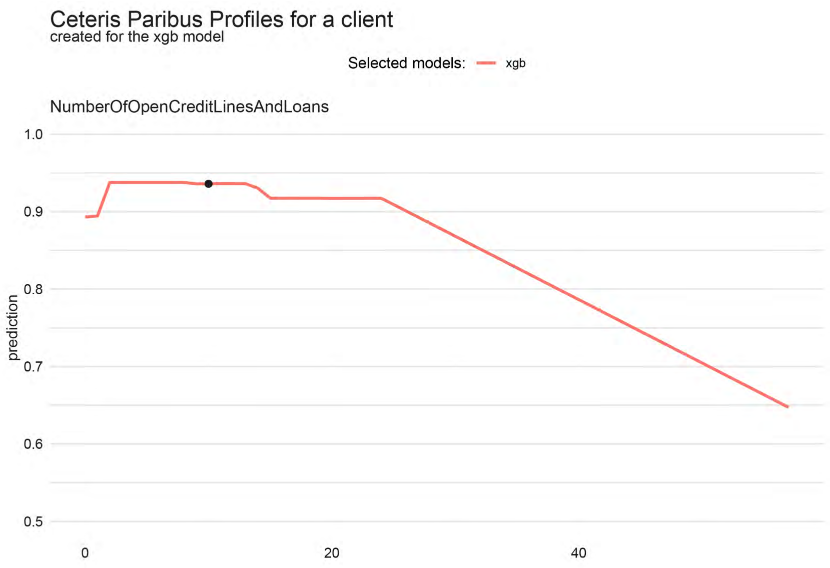


Figure 7.6: Ceteris Paribus profile for a single client. Calculating the DALEX library. A local level explanation
Source: own work

available which gives a bit more insight into the role of the all features in the model.

In Figures 7.7 and 7.8 Feature Importance Plots for training and testing samples from SHAP framework are presented. In this framework, the feature importance for each independent variable is calculated as an absolute mean of SHAP values. Differences in those values give us a good intuition in comparison of importance for each of the features (the higher value, the more important the feature), however a scale for absolute SHAP values is not so intuitive, therefore a role of given features in the prediction may be not so easy to assess. Then, the SHAP for the methods with logistic link function may be explicitly linked to the *points to double the odds*, so one can estimate the impact of the feature on the target variable. In Figure 7.7 it may be noticed that the most important variable is utilization of revolving lines which is twice as important as any of the rest. Then, a few following variables are similarly important (age, number of times borrower has been 30-59 days past due but no worse in the last 2 years, debt ra-

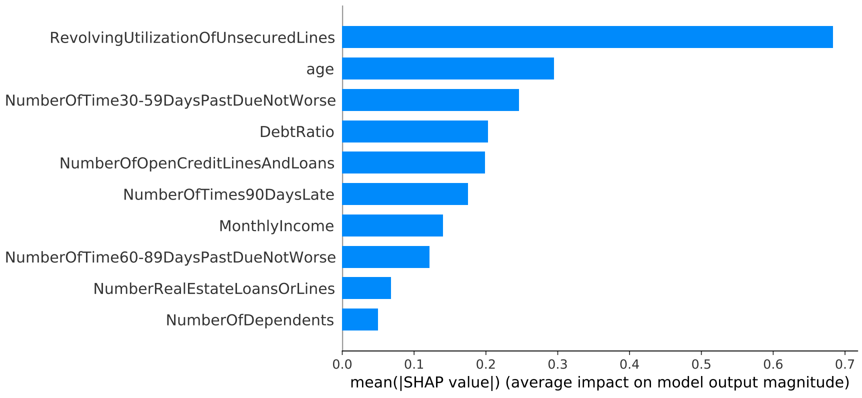


Figure 7.7: Feature Importance plot based on SHAP values for the training sample. A global level explanation
Source: own work

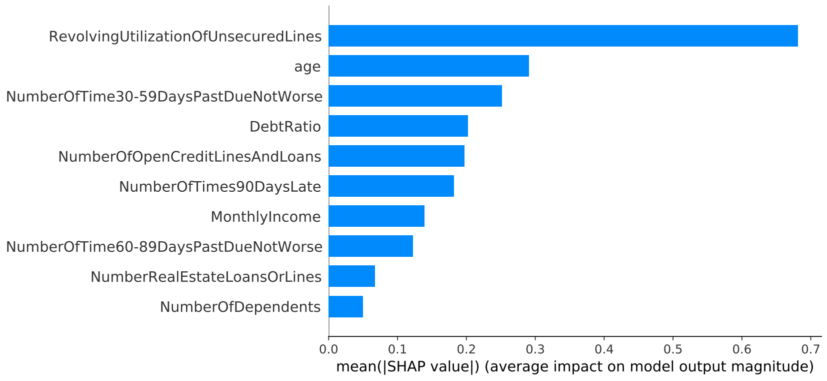


Figure 7.8: Feature Importance plot based on SHAP values for the testing sample. A global level explanation.
Source: own work

tio etc.). We may also notice that contribution of features to predictions is spread around many different variables, which is very often considered as crucial for a good credit scoring model (risk of fast deterioration in model quality is smaller, as prediction is diversified).

In the process of the model assessment, the comparison of features importance between the training and testing samples is very useful. To assess whether the model is rather stable and not over-fitted, the importance of variables should be similar in both samples. Analyzing Figure 7.7 and 7.8

it can be stated that for the trained XGBoost model, the importance for all variables in training and testing samples is very similar. Based on that, it may be assessed that the model should work similarly for observations that were not included in the training sample (however, coming from the same period of time).

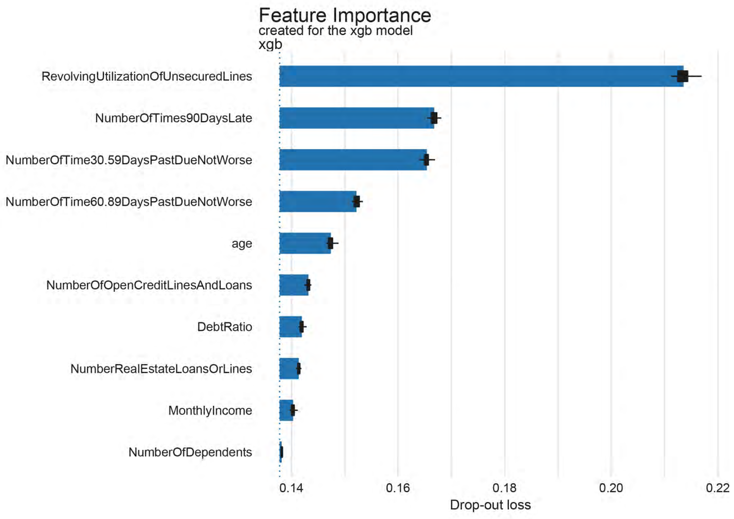


Figure 7.9: Permutation Feature Importance for testing sample. A global level explanation
Source: own work

The feature importance plot is also available in the DALEX framework, here the Permutation Feature Importance is calculated. This approach gives an opportunity to choose a metric that is easily interpreted. In Figure 7.9, the feature importance is calculated in marginal AUROC (area under ROC) drop scale. In this case, the role of variables may be assessed, both in relative and absolute terms. It may be observed that utilization of revolving products is again the most important feature (even more than twice important than next variables). However, based on the results obtained here, it may be also stated that if the utilization of revolving product would lose its discriminatory power (as if it was randomly assigned to clients), the drop in discriminatory power in AUROC scale would be equal to 0.08 (0.22 for the variable vs. 0.14 for the whole model) or 0.16 in Gini scale.

In other words, the AUROC for the model would drop from 0.86 to 0.78 or Gini from 0.72 to 0.54. As AUROC or Gini scale is usually the most important scale in financial institutions, an absolute interpretation becomes also very intuitive. Permutation Feature Importance values depends on the set of permutations that have been drawn in the calculation process, thus it may be different for different iterations. In order to present possible uncertainty in assessment, navy blue box plots are also drawn at the end of each bar in the figure. In this case, the uncertainty is rather small and it seems that the main disadvantage (potential instability) of the Permutation Feature Importance method is not essential here.

In the SHAP framework, the Feature Importance may be extended by an analysis of a Local Explanation Plot presented in Figures 7.10 (training sample) and 7.11 (testing sample). In the figures, horizontal violin plot for SHAP values (empirical distribution of SHAP values) is drawn for each variable. The SHAP values are colored red if the value of a given variable is high and colored blue otherwise. In both figures, we can see a very similar distribution of SHAP values for all features, this is another confirmation that the model should not be deeply over-fitted, as it works similarly on the training and testing samples.

Analyzing a plot for utilization of revolving products, it can be seen that the majority of distributions is placed around low values of the variable (blue color) and low values of utilization affects predictions positively (higher probability of being a good client). However, many high value outliers may be also observed that affects prediction negatively and its role in final prediction is higher (higher values in absolute terms).

The results for the utilization of revolving products seem to be intuitive and expected. Much more interesting results may be observed in the example for a number of dependents. It can be seen that a small number of dependents has a rather small impact on predictions. This stands in contrast to a high number of dependents seeming to have a higher impact on the prediction but the direction of the impact is not unequivocal. The impact for some clients is positive, for others is negative. Such results

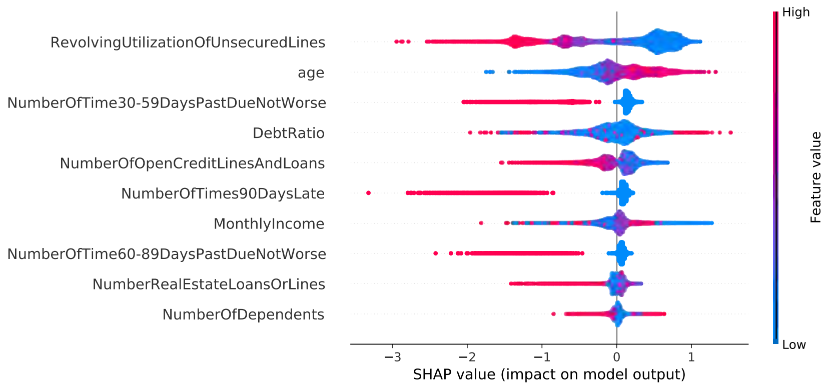


Figure 7.10: Shapley values for training sample. A global level explanation
Source: own work

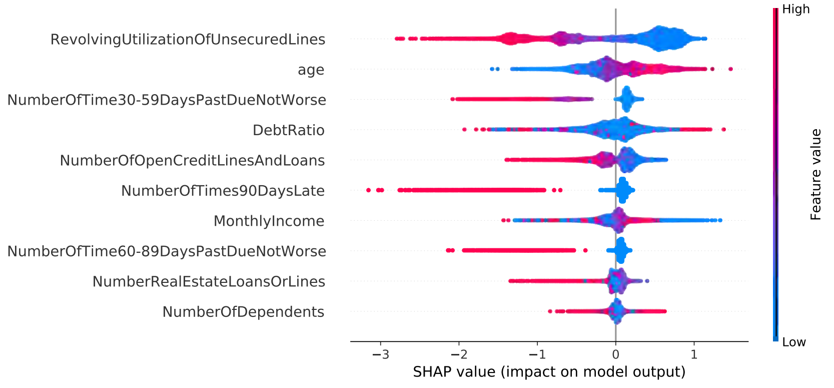


Figure 7.11: Shapley values for testing sample. A global level explanation. Note that it is similar to 7.10 which suggests good generalisation properties
Source: own work

may suggest an interaction with a variable that may possibly distinguish both groups.

Another set of methods at the global level of model explanation are partial dependence plots which show how the change of values for a given variable affects the predictions in the whole sample.

In the SHAP framework, the SHAP Dependence Plot is available. This plot shows how SHAP values are distributed for each value of a given variable and whether any specific profile of dependence between a given variable and prediction may be observed. Analyzing Figure 7.11, it may be noticed that being relatively young (30-40 years old) affects the prediction of being a good client negatively, whereas getting older affect positively the prediction (positive trend stops around 70 years of age).

In the DALEX framework, an analogical plot is Partial Dependence Plot (Figure 7.13, green line). It also depicts a profile of average dependence between a given independent and a dependent variable. In this plot, instead of SHAP values, an average prediction of dependent variable for each value of an independent variable is presented. In Figure 7.12 a similar positive tendency in probability of being good when getting older may be observed. Conclusion should be the same for both analyses.

Both methods work properly, if no collinearity in independent variables is observed (i.e. the averaging across the combination of features that are non-existent; the multicollinearity problem should be on the level of particular approach). There is no solution for such a case in the SHAP framework, however in the DALEX framework, two possible solutions are included. First is Conditional Dependence Plot (Figure 7.13, blue line) which shows how the analyzed independent variable and all collinear variables jointly affect the average prediction. Second approach is Accumulated Local Effect Plot (Figure 7.14) which extracts the effect of the collinear independent variables and shows only the relation between a given independent and dependent variable (by construction, it is on different scale, therefore for predicting probability of being good, it has to be presented on a different plot). Having all those 3 plots, an appropriate analysis of relation between independent and average prediction of dependent variables may be performed.

As a general rule, it may be assumed that if all 3 profiles have similar shapes, the PDP may be interpreted as if they are similar – the target value prediction is observed along with the change of the feature (accounting also the correlations between variables). In the case where all those profiles are different, usually the ALE plot is interpreted as most common, only a given variable role in prediction of dependent variable is analyzed (not all collinear variables jointly). In Figures 7.12 and 7.13, it may be observed

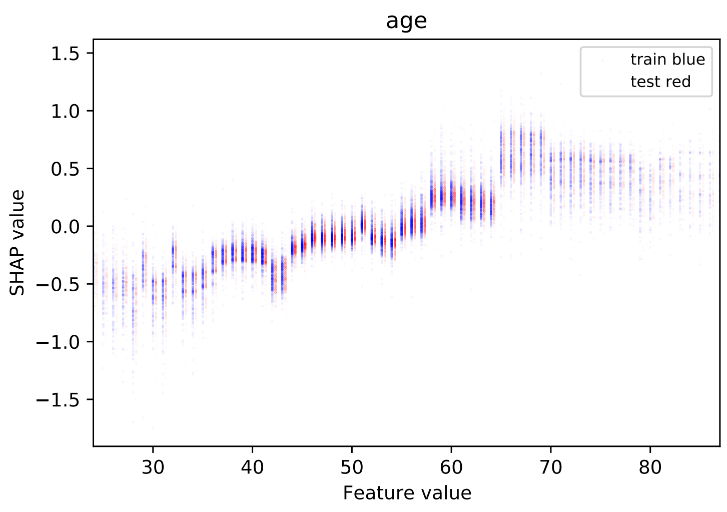


Figure 7.12: Variable profile based on Shapley values. A global level explanation
Source: own work

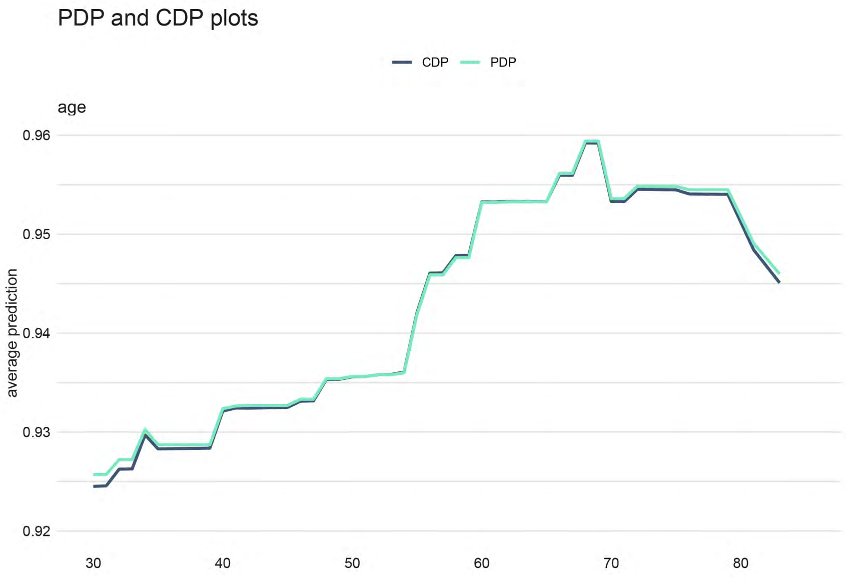


Figure 7.13: Partial dependence profile and Conditional dependence profile. A global level explanation
Source: own work

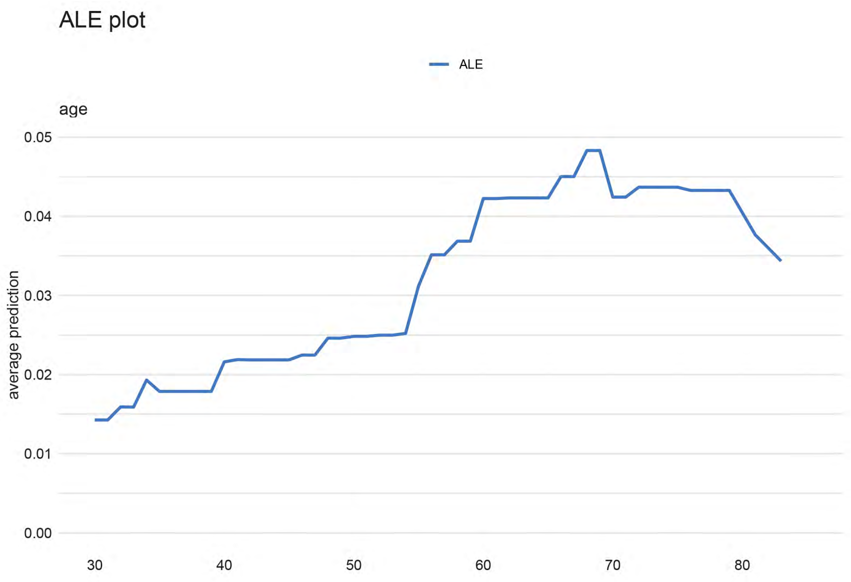


Figure 7.14: Accumulated local dependence. A global level explanation
Source: own work

that shapes for all 3 curves are very similar, so it may be concluded that age is not collinear with other variables and therefore PDP may be interpreted. Therefore, it may be stated that the young clients (92.5% for clients in age of 30) have the smallest probability of being a good client and the average probability is growing with age until around the age of 70 (95-96%), then the probability stabilizes.

To sum up, both XAI frameworks – DALEX and SHAP, provide a set of tools that enable to explain how the complex models work, both on local and global level. The methods presented above are not the only methods that are available in both frameworks, however they are a good foundation for making complex machine learning models explainable. The choice between the frameworks is not easy, as both give a comprehensive set of methods to explain and assess the quality of the model. The choice should be made including at least a few criteria:

1. the set of algorithms to be analyzed (SHAP works efficiently for example for tree-based model, DALEX is model agnostic, so many different algorithms may be compared),
2. preferable methods for analysis (SHAP has for example Local Explanation Plot which provides a lot of information at one plot, whereas DALEX, for example, has a what-if analysis for collinear independent variables),
3. intuition of interpretation (SHAP has all the methods based on SHAP values which is consistent, however DALEX has feature importance in AUROC/Gini scale which are well known in financial institutions),
4. the technology used and preferred by the institution; both SHAP and DALEX are available both in R and Python.

The set of criteria mentioned above is not exhaustive, many additional specifics for a given financial institution may be considered. What is important, no matter which one of XAI frameworks would be finally chosen, a comprehensive analysis of a model quality and explainability may be performed which is extending actual best practices in credit scoring models assessment. Having those frameworks, we may clearly understand the model and therefore confirm that credit scoring model based on machine learning algorithm may be used instead of classical credit scoring model without increasing the operational or model risk (i.e. related to the methodological assumptions of the model and interpretability of the model's parameters). Basing on methods presented so far, many additional analyses may be performed that extend insight into the analyzed model.

7.8 Extensions of base XAI analysis

Well-prepared XAI frameworks, as those discussed (SHAP and DALEX), are very intensively developed and new methods are consecutively implemented. In spite of that, not all useful analyses are available and therefore additional methods and extensions may be created in order to better understand how the model works. In the section, two very useful plots are discussed. Firstly, an extension for Feature Importance analysis is presented, secondly a plot showing the relation between standard deviation of SHAP values and mean difference between SHAP values for good and for bad clients.

In the extended Feature Importance analysis not only mean absolute SHAP values are analyzed but also:

- standard deviation of SHAP values, for which outliers (severe contributions of a given variable in both directions) weights more than in mean absolute SHAP values. Standard deviations are by definition centered around the mean value of the variable, so they may be used to assess a feature importance with extracting a role of a shift in mean of SHAP values,
- mean of SHAP values which for training sample should be equal 0, but for a testing sample may differ slightly (Figure 7.15) or severely (Figure 7.16). The shift may be either positive or negative, it depends on a data change.

In order to present how extended Feature Importance may be used in a model, two plots are compared. In Figure 7.15, the same testing sample that were used for previous analysis is presented, in Figure 7.16 a testing sample where a shift in distributions of clients may be seen. The shift in data distributions may be recognized by having mean of SHAP values not equal 0. If mean is around 0, it means that the average contribution of a given variable is similar in the analyzed sample as in the training sample. In the second testing sample, the mean of SHAP values is positive, it can be interpreted as a shift of clients distribution toward better clients (on average variable contributes more positively than in the training sample).

Acknowledginh mean of absolute SHAP (mean abs) values and its standard deviations (stdev), a plethora of useful information may be obtained. In the first testing sample, shorter bars for mean absolute values than for standard deviations show that we should expect a group of clients with severe SHAP values for most of features. Completely different conclusions

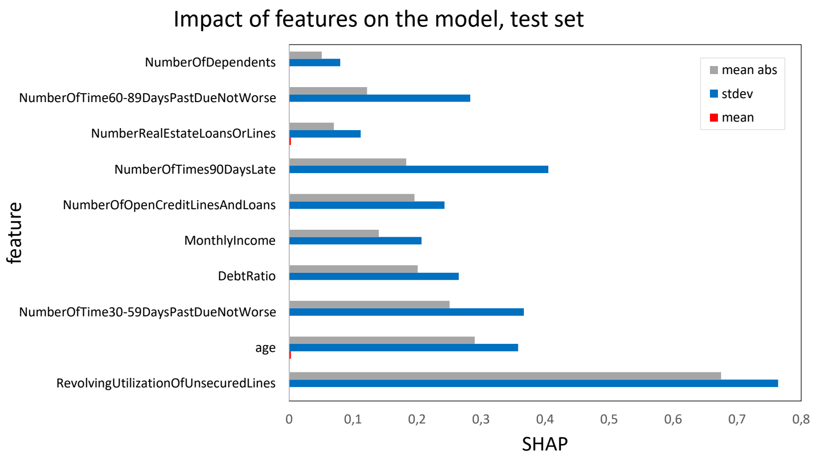


Figure 7.15: Extended Feature Importance analysis for a testing sample with not significant shift in data
Source: own work

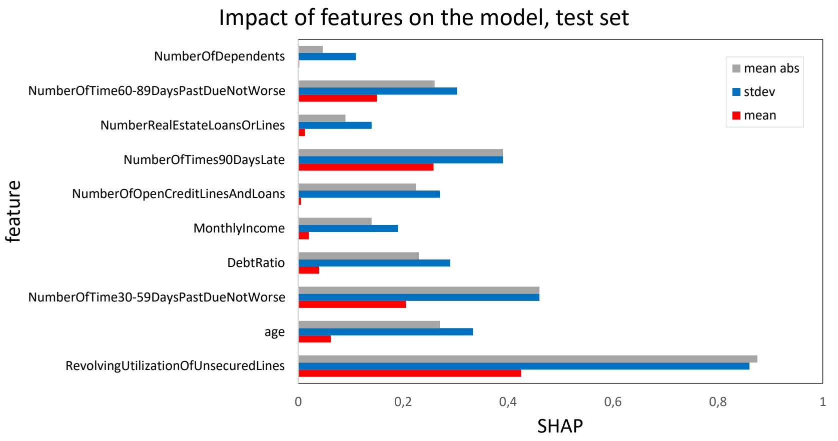


Figure 7.16: Extended Feature Importance analysis for a testing sample with significant shift in data
Source: own work

may be drawn from the second testing sample, here the standard deviations are close to mean absolute values, so the role of outliers is not important.

No matter which metric (mean absolute or standard deviation) we finally use to assess the importance of the features, we can state that the relative importance is the same in all cases (the most important is utilization of revolving products, then due experience, then age etc.). We may also observe that for both testing samples, the rank order of feature importance is similar, as well as its magnitude. Those results suggest that even though the second sample has a shifted distribution of clients, the role of each feature in final model predictions should be similar as for the first testing sample and the training sample consequently.

In this method, as we do not know what was the real outcome (Good or Bad client), it is assumed that if feature affects predictions in the same way in different samples, it may support a claim that the feature would perform similarly in terms of discriminatory power in all the samples considered. However, in order to confirm that, a joint comparison of predictors importance and discriminatory power should be performed.

It is interesting to analyze that by having a plot showing the relation between standard deviation of SHAP values (feature importance) and mean difference between SHAP values for good and for bad clients (feature discriminatory power). At the plot (Figure 7.17), on an x-axis, standard deviation of SHAP values for each variable is presented, whereas on the y-axis, deviation between average of SHAP values for good clients and average of SHAP values for bad clients are presented. The higher standard deviation, the more important the role of a variable in a prediction of a model. However, based on the standard deviation, it cannot be assessed whether the importance is linked with correct or not correct predictions.

In order to assess that, deviation between average of SHAP values between good and bad clients is calculated. It is expected that all variables affect prediction of being good for good clients positively and for bad client negatively, therefore the higher the deviance, the better variable in terms of discriminatory power. After merging this two dimensions together a plot may be drawn, where:

- features with high average deviance of SHAP values for good and bad clients and high standard deviation of SHAP values (upper right corner) should be considered as the most influential and highly separating. They play a crucial role in the model quality,
- features with a small average deviance of SHAP values for good and bad clients and small standard deviation of SHAP values (bottom

left corner) should be considered as not influential and not separating well. They are candidates to be removed from the model, as they not provide any valuable information into the model,

- features with small average deviance of SHAP values for good and bad clients and high standard deviation of SHAP values (bottom right corner) should be considered as influential but of poor separating quality. They should be removed from a model, as they not provide discriminatory power but affect the prediction significantly.

- features with high average deviance of SHAP values for good and bad clients and small standard deviation of SHAP values (upper left corner) should be considered as less influential but of high separating quality. They play also a crucial role in the model quality, even though they do not affect the prediction severely.

In the analysis, the difference between those two dimensions for training and testing samples may be included. We may check how the variables change their positions in the testing sample, in comparison to the training sample. If variable move more toward the x-axis than the y-axis, it means that discriminatory power drops more than the importance of the variable and therefore variable may be considered as riskier.

In Figure 7.17 we can see that the most crucial variable is utilization of revolting credits. It has the biggest importance as well as discriminatory power. However, it should be also noticed that in the testing sample its importance is not diminishing but its discriminatory power is. This results may raise a risk, as possible further deterioration in discriminatory power may lead to poor quality of the model. For most of the variables, in the testing sample discriminatory power drops, there is only one visible exception for variable counting number of times client was in 90+ days late. In this case, the discriminatory power of a variable and its importance grow jointly. In the figure, there are no variables in the bottom right corner, so there is not a need to remove any variable from the model, however variables, such as a number of dependents, seem not to have any special influence on the model and may be considered as redundant.

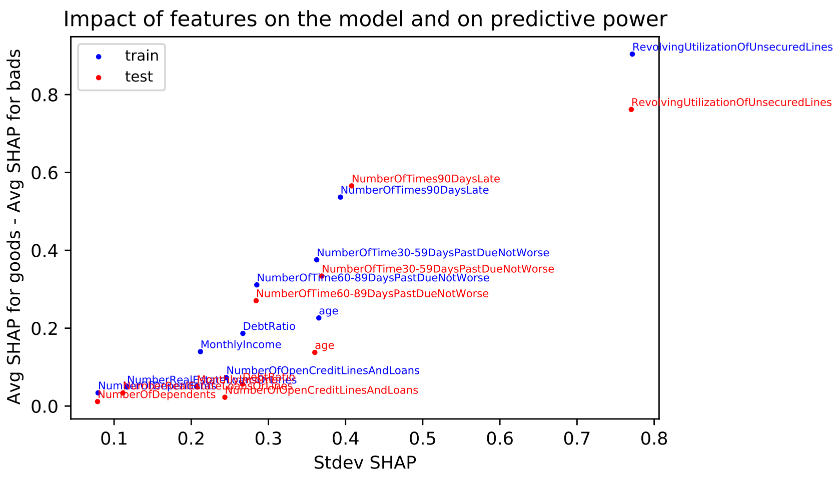


Figure 7.17: Plot showing the relation between standard deviation of SHAP values (feature importance) and mean difference between SHAP values for good and for bad clients (feature discriminatory power)
Source: own work

7.9 Conclusions

Explainable Artificial Intelligence methods were presented in this chapter. Different methods that help to assess and understand models on the local (prediction decomposition plots, ceteris paribus plot) and global level (feature importance, partial dependence plots) were analyzed. The methods presented are available in two XAI frameworks – SHAP and DALEX. The SHAP framework is SHAP values-oriented and is especially effective for tree-based algorithms. The DALEX framework provides methods using different metrics and is mainly focused on agnostic approaches. Both frameworks have their pros and cons however are comprehensive and ready to be proposed as tools for new best practices in credit scoring modeling using more complex modeling methods.

Chapter 8

Performance considerations and platforms for scoring models

Łukasz Kraiński

Decision Analysis and Support Unit
Collegium of Economic Analysis
SGH Warsaw School of Economics

In-depth knowledge of data processing, model building and assessment techniques is essential in creating a high-quality credit scoring application but additional technical knowledge is needed to implement the design in an efficient and scalable manner. This chapter focuses on computational issues connected with scoring models and highlights modern trends and platforms for time-effective processing architecture.

We begin with management and organizational perspective on computing performance in IT projects. Throughout years of experience, many frameworks dedicated to managing IT workflows, such as credit scoring applications, have emerged and shaped the standards for controlling time and resources. Afterward, we outline the history of computational resources and highlight how the approach to raising the components performance has changed since the creation of the first processor. After that, we follow the model creation process focusing on performance in consecutive stages: data gathering, model building and deployment. Each step has distinct characteristics and challenges which manifest themselves with increasing data volume, model complexity and deployed model utilization. The last part of the chapter focuses on Big Data services and cloud platforms as an example of modern computing environments suited for big-scale scoring applications.

8.1 Introduction to computation performance

8.1.1 Computation performance in the context of managing IT projects

Successful implementation of the scoring model requires also a consideration of computing and operational efficiency. Appropriate data processing and the construction of a high-quality model is not sufficient to successfully build a scoring application. The complexity of algorithms influence deployment effort for both initial launch and future features. Elaborate models may require meticulous testing process and additional computation resources during each release compared to simpler solutions. Additionally, during the service uptime, maintaining good performance of prediction both in terms of quality of prediction and responsiveness (latency) may pose a challenge and is linked with the complexity of the model as well as retraining process e.g. batch or online which will be discussed in more details in further sections.

Awareness of possible performance and quality caveats is essential during the application planning and designing phase. Formally, in the context of IT project management, there are three main planning factors (Frączkowski, 2003):

- cost, i.e. funds that can be allocated to project resources (hardware, computer software licenses, people and related costs),

- the time in which the project or project's milestones should be completed,

- scope including functionalities, used technologies and final shape of the product.

Technologies used in the financial sector are heavily influenced by legal regulations and the company's reputation strategy. Banks strive to be considered as cutting-edge tech companies associated with security, stability and innovativeness (Finlay, 2012). Modification of one of the above elements affects the others, e.g. by increasing costs, it is possible to reduce the project's implementation time or increase the planned scope. From the perspective of projects involving the creation of predictive models, it is essential to consider the required computing power. With the rise in the data volume and the use of increasingly complex models, it is necessary to use increasingly faster computers that allow more efficient calculations and technologies dividing the workload among many computers. In practice, the computational requirements can be met by increasing the budget (more efficient equipment, introducing shift work system), postponing the project completion date or by using appropriate technologies.

In the credit scoring system, a project completion date can be identified as a start of serving predictions for new clients. With proper set of technologies e.g. *Predictive Model Markup Language* (*PMML*)[1] and *DevOps* tools, the project's progress may be controlled in a more flexible manner resulting in a higher chance for completion on time. Model performance specification is mostly determined by the number of new credit applications and latency requirements. System design should take into account that most of the credit applications are generated through *mobile* and *internet channels* while few applications are filled in bank branches. On the other hand, in-branch applications need a faster response time while remote channels latency requirements are more relaxed. As for planning, an important computational factor is tied to the training type – *batch training* uses internal

[1]http://dmg.org/

bank data while *online learning* utilizes data provided by the clients. Batch and online learning are discussed in more detail in Section 8.2.3.

IT projects differ from projects implemented in other industries, mainly due to the variable scope of the project through duration (Liebert, 2017). This feature means that IT projects are often carried out using *agile* methodologies such as *Scrum* or *Extreme Programming* which allow to quickly incorporate new requirements in the project (Schwalbe, 2013). Requirements change may also demand increasing scope and volume of data used in order to provide a service for a much larger group of recipients or use a more complex class of models. The majority of specification changes occur before the launch of a new system but an additional scope may be added also during the service operations. Due to that, it is important to base the system on *scalable* and *extensible technologies*, in particular allowing harmonious and quick application adaptation to the growing intensity of requests (Roszkowski, 2015).

The cost and time of the project are not flexible which is why the increase in available computing power is currently gained by using appropriate computer programs based on Big Data principles or functionalities of programming languages and computer components (processors, RAM). Modern solutions, e.g. *cloud computing*, allow to smoothly modify the trade-off between the cost and time of calculations depending on current design requirements, both in model creation and utilization phases. This is also of great importance in terms of synchronizing the resources within the project. Apart from unexpected failures, computer equipment is available for use at any time however, project team members have designated working hours, therefore the ability to control the speed of calculations gives them the ability to perform tasks more efficiently and build a product. Synchronization is also important considered that standard model building process includes three environments: development, test and production. Data, code and configurations are migrated between environments to introduce new features and deploy model. A distinction into environments separates and organizes the workflow but also introduces additional computational overhead. With proper technology and computing infrastructure, migration cycles may be shortened and the system deployed more efficiently.

An additional area that impacts IT system performance and project costs is how data is collected – more on the data sources in credit scoring in Chapter 2. This is an extremely important layer of IT projects that affects the performance of the product both through the characteristics of the data device used (HDD, SSD, in-memory data) and the data storage system (in relational databases, in the form of binary files, JSON, Avro, HDFS, XML).

8.1.2 Historical outline of computer performance

Around 80 years have passed since the first digital computer was created, during which the computing power of computers has increased many times. For comparison, a modern laptop has 4 million times more RAM and 100,000 times faster processor than the on-board computer of the Apollo 11 mission. When analyzing the development of computers, several approaches to computing performance can be distinguished. Initially, the increase in machine performance was provided by reducing the size of transistors used in processors, so more elements were packed in the same space. According to the *Moore's law*, it is assumed that the number of transistors in microprocessors doubles every 2 years (Moore, 1965).

The term *Moore's law* is also used to change the characteristics of other computer components (e.g. RAM size, hard disk capacity) and related technologies (e.g. Internet bandwidth).

As already mentioned, one of the main reasons for which this exponential growth is possible, is the use of continuously smaller elements in the production process. However, taking into account the physical problems with the development and miniaturization of transistors, it is expected that in a few years the limits of element placement will be reached (ITRS, 2015). For this reason, the computer industry is currently focused on increasing computing capabilities by creating *multi-core processors* used in *parallel processing.* In the 1960s and '70s, parallel calculations were used only in supercomputers using specialized components. In the mid-1980s, as part

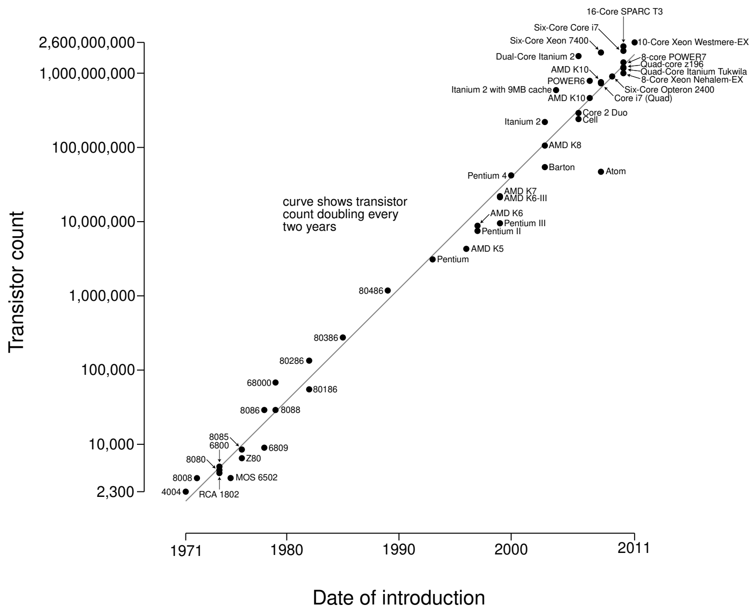


Figure 8.1: Number of transistors on processor chips
Source: own work

of a project funded by NASA, a supercomputer was created based on widely available Intel 8086/8087 processors which was a step towards the introduction of multi-core designs on the mass market (G. Fox et al., 1985). Finally, the two largest manufacturers of home computer processors included their first dual-core models to the product catalogue in May 2005 (Intel – Pentium D, AMD – Athlon 64 X2). In 2020, the number of cores in a processor is one of the main determinants of processor quality and performance. For example, processors from Intel i9 Series X have 18 cores while AMD Ryzen 9 3950X has 16 cores.

The main assumption of parallel calculations is to divide the instructions necessary to execute the program among the available processors/cores. Theoretically, it is therefore possible to reach computations speedup equal to the number of processor cores. In practice, only certain parts of the program can be parallelized, thus the maximum acceleration of calculations using multi-core architecture is limited. This problem is described by *Amdahl's law*, based on the Gene's Amdahl article published in 1967 (Amdahl, 1967).

The maximum calculation acceleration regardless of the number of cores is $\frac{1}{1-p}$, where p is the ratio of parallelized part in the program. For example, if 90% of calculations can be processed in parallel, using multiple cores will speed up the program by up to 10 times, see Figure 8.2).

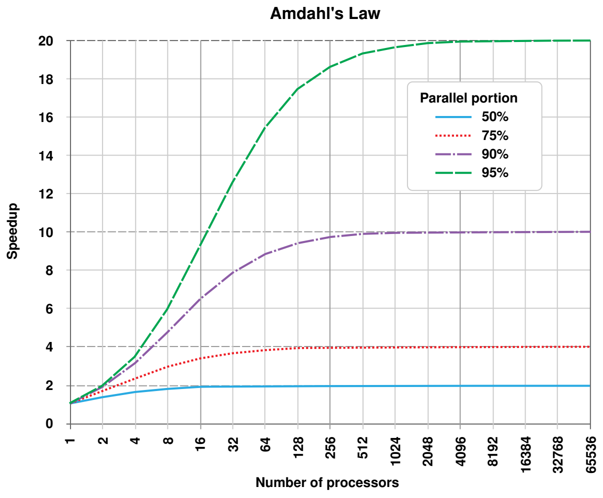


Figure 8.2: Grafical interpretation of Amdahl's law
Source: own work

The increase in demand for computing power resulted also in emergence of specialized components prepared for numerical calculations e.g. graphic cards. *Graphics processing units* (*GPU*) were created for the effective processing of images, films or animations required by computer games or programs for editing audiovisual files. Adapting GPUs for other type of calculations was named *General-Purpose Computing on Graphics Processing Units* (*GPGPU*) and allows to compute particular tasks much faster compared to the CPU, especially in problems with a high degree of parallelism. In November 2006, NVIDIA announced the launch of *Compute Unified Device Architecture* (*CUDA*) technology which defines the general architecture for calculations on GPUs. As part of the release, NVIDIA prepared a programming environment in C language for CUDA integration (NVIDIA, 2006). Currently, GPGPU libraries are available in many popular programming languages, such as Python, Java and C#. Calculations on graphics processors are used, among others, in simulations and modeling of physical processes, cryptography, mathematics and artificial intelligence.

In machine learning applications (including the construction of scoring models), it is also possible to use dedicated processors called *AI accelerators*. Accelerators perform calculations for neural networks much faster with relatively low power consumption compared to ordinary CPUs. For example, in May 2016, Google introduced an accelerator called the *Tensor Processing Unit* (*TPU*) which was used by the RankBrain search engine algorithm or during a match between AlphaGo and Lee Sedol – a Go game master (Google, 2016). Initially, Google used technology only for internal projects but since June 2017, TPU has been widely available as part of Google's cloud services (Google, 2017). TPU support is built into the popular machine learning framework – *TensorFlow*, making it easy for users to take advantage of this modern technology.

An alternative approach to increase computing capabilities is the technology referred to as *distributed computing* which consists in dividing tasks into smaller subtasks performed simultaneously by a group (cluster) of computers. Distributed computing is the foundation of the popular Big Data approach which makes it possible to control the speed of calculations by changing the number of machines in the cluster. Distributed processing was separated as a distinct area of information science in the early 1980s (the first Symposium on Principles of Distributed Computing conference took place in 1982), however, the start of the Big Data trend and the intensive development of distributed computing platforms is associated with the creation of *Apache Hadoop* in 2006[2] (Dean & Ghemawat, 2004; Tel, 2001). Twelve years after the creation of the Hadoop system, around 60% of companies declare using Big Data technology, when in 2015 it was only 17% (Dresner Advisory Services, 2018). This dynamic growth is caused by the progressive digitization of societies and easier access to data sets, which

[2]https://archive.apache.org/dist/hadoop/common/

creates demand for efficient and scalable data processing systems. In addition, the adoption of parallel processing has a low entry threshold, due to open-source distribution of many Big Data technologies (Spark, Kafka, Kubernetes, Hadoop) and the growing use of cloud services which allows to quickly adjust the number of units in computing clusters, which would not be possible in classic on-premise solutions.

8.2 Performance areas relevant to scoring models

While building credit scoring applications cost and performance considerations emerge in 3 main areas:

1. Gathering data
2. Preparing data and building scoring models
3. Model deployment

Each step has different compute and storage characteristics and rely on internal or external resources differently. In case of small scoring models performance issues may not be apparent but with large or dynamically growing application, performance bottlenecks on any of aforementioned stages may lead to usability degradation.

8.2.1 Gathering data

Gathering data for scoring application is a relatively complex process compared to other machine learning applications, more on data sources in Chapter 2. The underlying complexity is a result of factors specific to banking industry and characteristics of attributes used in modeling, namely:

- heterogeneity of used data sources, both in aspect of data structure (structured, unstructured and semi-structured data) and source storage system (FTP servers, relational databases, APIs)
- legal obligations to archive data for fixed amount of time and provide data for audit on demand

- legal obligations to remove or anonymise unused data for privacy reasons
- legal compliance with sensitive data handling e.g. GDPR and fair-aware (non-discriminatory, fair lending) regulations
- high volume and velocity of data due to increasing usage of social feed or clickstream data and availability of digitized information in general

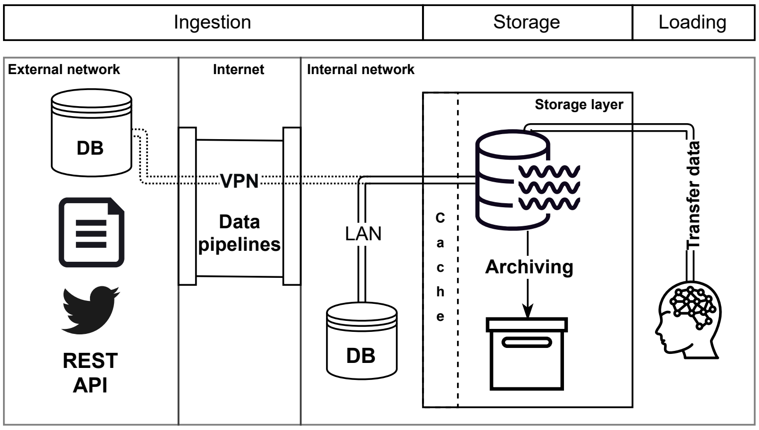


Figure 8.3: Data gathering process
Source: own work

Data gathering process breakdown may be summarized as in Figure 8.3. It consists of data ingestion from both external and internal sources, storing the data in appropriate service (relational or NoSQL database, data lake, flat files on filesystem) which includes moving the data inside storage system (e.g. compressing and archiving) and transferring the data to model building platform. Depending on assumed approach, transforming the data (also called *data wrangling* or *data munging*) may be considered part of data gathering but for the sake of the monograph it is bundled with model building subsection.

Ingestion phase is usually most vulnerable to performance issues due to external character of data sources and their heterogeneity. During ingestion of data, data pipelines built using integration tool (e.g. Apache Airflow, Informatica PowerCenter, Pentaho) extract data from external sources using internet connection or private VPN tunnel and LAN connectivity for sources in company's network. Performance bottlenecks may appear in three points:

1. Data sources throughput.

2. Internet connection speed and geographic latency.

3. Overload of machine hosting integration tool.

Storage and data feed applications differ in available throughput and scalability. Particular solutions may additionally impose quotas on the users, for example REST APIs often limit calls made in time interval (hour, day). In general, throughput of data source depends on underlying hardware (processor, memory) performance but applications may impose varying computing overhead. For example, simple file transfer should provide higher data throughput than relational database which perform additional data checks as part of transfer process. More important issue is limited ability to adjust performance provided by external vendors. Assuming that scoring application requires data from numerous providers and delivering complete data set to the model is highly expected, performance of all sources may be considered equal to performance of "the weakest link" as all data has to be digested before carrying on with the processing. Issue can be mitigated by investigating potential vendor infrastructure and including appropriate requirements in the contract.

Another limiting factor during ingestion is internet connection speed both on source and sink side. During ingestion, data travel through the internet so maximal throughput is constrained by connection provided by local *Internet Service Provider* (*ISP*). ISP services are typically bounded by long-term contract so potential increase in data throughput demand should be foreseen in advance. Final performance related to internet connection is mainly influenced by source system upload speed, sink download speed and geographic distance between them. It is worth noting that regular ISP service is asymmetric and usually provide 10x smaller upload than download speed. Symmetric connection is possible with dedicated internet line but this is costly alternative, require additional service contract and geographic proximity between data exchanging parties. Another popular connection service is a Virtual Private Network. VPN is a secure tunnel between two networks that traverses public internet. It provides another layer of security but additional encryption typically slow down internet throughput by 10%-20%.

All data ingestion activities have to be managed by data integration or orchestration application. As data pipeline workloads may be computing intensive, performance of data integration server is considerable issue. Monitoring usage metrics of the server (CPU and memory utilization, disk I/O) is important to keep ingestion effective. If overload is spotted it is important to split workloads to different machines or alternatively provide clusters of machines that scale and distribute work automatically with increasing utilization. Cluster management platform are often provided as built-in integration tool feature or additional product.

Internal data sources data may be transferred using LAN connection either through wireless WiFi or cable connection using Ethernet. Speed provided by modern Ethernet (e.g. IEEE P802.3 25 Gb/s Ethernet) standards is sufficient to eliminate LAN connection from performance considerations.

In storage layer, ingested data is stored and also archived for audit purpose or as a backup. In general, storage systems can be categorized by type of gathered data (Table 8.1). Modern application store both structured and unstructured data in data lake tailored to handle multiple data formats. Another approach to data processing and storage include streaming architectures (lambda and kappa architectures), in which data move between producers and consumers through message brokers (Lin, 2017). For example, popular distributed message broker Kafka buffer data in any type and can act as temporary storage before data is collected by proper permanent storage system. Described approach allows to process data as it is produced and build more reliable storage systems.

Independently of platform used, data must be stored on physical piece of hardware which greatly determines system performance (read and write speed). Typically three types of storage hardware are considered – persistent HDD and SSD and volatile RAM memory. They differ much in terms

Data type	Example	Data storage	Applications
Structured	relational data	SQL/NoSQL databases	MySQL, Cassandra
Semi-structured	JSON, XML	document databases	MongoDB, CouchDB
Unstructured	videos, text files	filesystem, HDFS	Hadoop, FTP server

Table 8.1: Data types and dedicated storage examples
Source: own work

of read/write speed and storage cost (Table 8.2). Hard Disk Drives are 10 times slower than Solid State Drives but also 10 times cheaper which make them suitable for archived data. For latency critical applications data may be stored in RAM which is more than 10 times faster compared to SSDs. On the other hand, RAM storage is most expensive solution and require additional software e.g. Memcached. Another downside of in-memory storage is RAM volatility which means that stored data is lost after reboot, although applications such as Redis with special architecture may provide persistent in-memory system. Common practice for performance-oriented projects is to use persistent disks as main storage device and boost speed by temporary caching for frequently used data in-memory. For scoring applications it may be beneficial to cache data obtained from third party APIs e.g. from credit reference agencies. API calls are slow and may be limited in number, thus if underlying data change slowly cache provide fast, temporary storage that can be updated with fresh data on demand.

Storage	Read speed	Write speed	Cost per GB
HDD	200 MB/s	150 MB/s	0,03$
SSD	2 500 MB/s	1 500 MB/s	0,2$
RAM	35 000 MB/s	30 000 MB/s	5$

Table 8.2: The average performance characteristic of storage hardware, based on offers on amazon.com, as of the March 2020
Source: own work

In highly regulated industries as banking, it is important to plan and optimize storage capacity in advance. Financial laws impose different retention periods for data, usually no shorter than 5 years. To effectively store the data proper compression algorithm should be applied to regular archiving process. For compression of flat files (CSV, Excel files, etc.) common *.tar*[3]*.gz*[4] format may be used and Parquet[5] for relational data. Another consideration is storage infrastructure planning which must provide redundancy to account for possible data loss and provide sufficient storage space for audit data.

Last step is loading data to modeling application which is usually present in the same network. In that case, performance of data transfer is determined either by LAN connection speed, write speed of storage system (e.g. data lake) or integration server capacity. Some applications for building AI models (e.g. Databricks) can be mounted directly on top of storage system, so data is immediately available after ingestion.

Described storage characteristics are relevant to on-premise system but many of above-mentioned problems may be tackled using cloud storage. Cloud solutions are often provided in one of three models:

- IaaS – Infrastructure as a Service.
- PaaS – Platform as a Service.
- SaaS – Software as a Service.

These systems differ in management capabilities of underlying software and hardware. For example in IaaS model, company can rent Virtual Machines and install appropriate storage application on them. This reduce infrastructure management work but gives full control over used application. In PaaS model, company can directly choose storage service e.g. SQL Server or MySQL database and all required software will automatically set up (Table 8.3).

Considering problems mentioned before, using cloud may be beneficial in few ways:

- resources in the cloud are connected with fast, dedicated connections, so accessing data sources in the same cloud and geographic region provides much better performance,

[3]https://www.gnu.org/software/tar/
[4]https://www.gnu.org/software/gzip/
[5]https://parquet.apache.org/

	On-premise	**IaaS**	**PaaS**	**SaaS**
Applications	X	X	X	V
Data	X	X	X	V
Runtime	X	X	V	V
Operation system	X	X	V	V
Servers	X	V	V	V
Storage	X	V	V	V
Networking	X	V	V	V

Table 8.3: Comparison of different IT resources management models; X – managed by you, V – managed by provider
Source: own work

- often PaaS services provide automatic scaling based on chosen metric, for example database memory may be increased if utilization exceeds fixed threshold for prolonged time,

- integration services clusters are managed dynamically to accommodate current load.

In context of data storage, particular cloud solutions called *blob storage* requires additional attention. Blob storage is unstructured data storage with virtually unlimited space and is billed per GB stored. With this approach, company don't have to plan for storage capacity and pay only for currently used storage. Additionally, blob storage offering distinguish *hot* and *cold* data where hot data is frequently accessed and cold data is read rarely but has lower price per GB. This map well with current data used for modeling and archived data stored for legal compliance. Important feature in terms of legal regulations is that blob storage automatically replicate and encrypt data, so data is secure and probability of data loss is extremely low. Cloud providers often guarantee compliance with financial standards as part of a product[6].

[6]https://cloud.google.com/solutions/financial-services

8.2.2 Preparing data and building models

For data preparation and model training often complex data transformations are required. While for copying data storage performance and internet connection speed were important factors, data transformations are relying heavily on memory and processor efficiency. To reach highest performance high quality components should be used along with modern parallel and distributed computing practices. Parallel computing may be perceived on different levels, namely:

- hardware parallelism,
- operating system/software implementation of parallelism,
- paralleled algorithms.

All the above stages must be in sync to provide an effective performance boost. Hardware parallelism is connected with the physical construction of processors and their design architecture. A commonly used distinction was proposed by Flynn and is presented in Table 8.4 (Flynn, n.d.). *Single Instruction Single Data* corresponds to the simplest computer where one instruction is applied to one data piece at a time. An important approach is *Single Instruction Multiple Data* in which the same instruction is applied to multiple data pieces e.g. elements of a vector. Most modern computer chips, although they are SISD type, incorporate vector processors on the chip and provide special instructions which can be used on a software level to utilize them. Multiple Instruction Single Data approach is used in uncommon processors such as Micron Automata. *Multiple Instructions Multiple Data* is architecture used in modern CPU where each processor (core) uses its own data and executes its own program.

A specialized type of SIMD processor is Graphical Processing Unit which contains many more cores than CPU. For example, Tesla V100 has 640 computing cores while a cutting-edge CPU has 18 cores. Although each core is slower than the CPU core, GPU computes highly parallel tasks such as matrix multiplication and other linear algebra operations quickly. On the other hand, GPUs lack more general instructions that are

	Single data	Multiple data
Single Instruction	SISD typical thread	SIMD vector processors GPU
Multiple Instruction	MISD rarely used Micron Automata	MIMD multi-core processor

Table 8.4: Flynn's taxonomy of parallel processors
Source: own work

implemented in CPUs. *Application-specific integrated circuit* (*ASIC*) such as *Tensor Processing Unit* from Google also relies on massive parallelism. Third-generation TPU may be rented for up to 1000 cores and perform AI-specific calculations even faster than GPUs.

Software level parallel computing can be further divided into three categories:

1. bit-level parallelism,
2. instruction-level parallelism,
3. task parallelism

Bit-level parallelism speed-up stems from doubling the computer's word size (the amount of information the processor can manipulate per cycle) in following processor architectures. For example, an 8-bit processor add two 16-bit integers by dividing them and adding in two steps while 16-bit processor would carry that instruction in one cycle. That parallelism type takes place only when the processor's word size increases (models with new architecture are introduced) and the software is adjusted. Currently, the standard word size is 64-bit.

In *instruction-level parallelism* (*ILP*) multiple instructions from the same instruction stream can be executed concurrently. In the software context ILP, is managed by a compiler deciding which instructions to execute in parallel. Parallel instructions are designated implicitly by an optimizer built into compiler or explicitly by a programmer. The degree of ILP in programs is very application specific, in certain fields, such as graphics and scientific computing, the amount can be very large.

Task parallelism is a form of parallelization that focuses on distributing tasks (concurrently performed by processes or threads) across different processors. In other words, multiple threads or instruction sequences from the same application can be executed concurrently. A common type of task parallelism is pipelining which consists of moving a single set of data through a series of separate tasks where each task can execute independently of the others. The exploitation of that approach began with the increased usage of multi-core microprocessors.

To fill the picture of both software and hardware parallelism *multi-threading* and *multiprocess*, computing must be introduced. On the hardware layer, multiprocessing requires multiple cores in processor and multithreading support for threading in processor architecture (e.g. hyper-threading technology developed by Intel). On the software level, both concepts have to be handled by a programming language and/or operating system. The main difference between multiprocessing and multithreading computing is memory sharing approach. Multiprocessing allocates separate memory and resources for each process or program while multithreading threads belonging to the same process share the same memory and resources of that process (Figure 8.4). Multiple processes generated act as separate instances of the program (e.g. Python interpreter) and are assigned individual IDs in the operating system.

Threads operate inside one process and share the memory during the execution, which improves tasks execution inside the process but also can create synchronization issues. As threads compete for resources they can block execution of other threads – in computer science this is referred to as *dining philosophers problem.* While the thread is using part of memory, it imposes a lock on it so that other threads can't modify the same memory segment until the lock is released. In extreme situations, it may lead to a *deadlock* – a situation when no thread can proceed further due to locks imposed by other threads. Due to that, multithread programming has to be handled carefully and it is a challenging part of software engineering.

Multithreading on the software level is often managed by an operating system (Windows, Linux) but there is an alternative approach called *green threading*. Green threads emulate multithreaded environments without relying on any native OS abilities and they are managed in user space instead of kernel space, enabling them to work in environments that do not have native thread support. That approach is supported in many programming languages e.g. Python, Julia, Go.

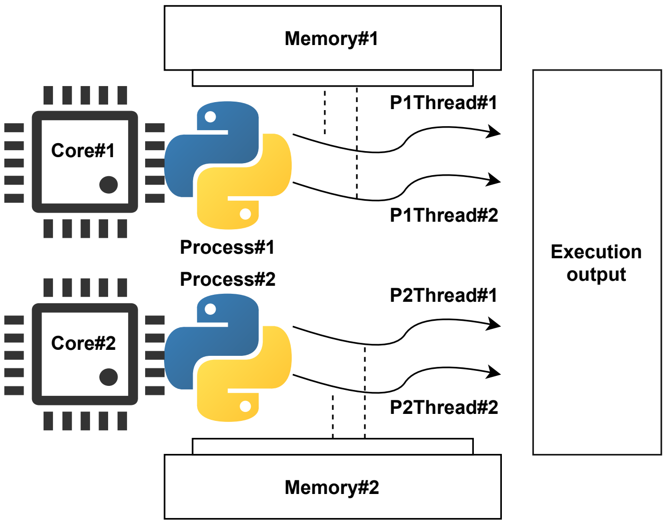


Figure 8.4: The relation between multiprocessing and multithreading computing
Source: own work

To fully benefit from hardware and software parallelism, applied computational tasks should also be highly parallel. Applications are often classified according to how often their subtasks need to synchronize or communicate into three classes:

1. fine-grained if the subtasks must communicate many times per second,
2. coarse-grained if the subtasks do not communicate many times per second,
3. embarrassingly parallel if they rarely or never have to communicate.

In the context of building a scoring model and machine learning models in general, two layers of application can be distinguished:

- outer layer – hyperparameter tuning,
- inner layer – training machine learning algorithm.

Hyperparameter tuning belongs to a class of embarrassingly parallel problems as every parameters combination may be calculated independently (Figure 8.5). Inner layer parallelization depends on the type of machine learning algorithm used. Highly parallel algorithms include:

- random forest – each tree may be trained independently,
- neural networks – especially efficient on GPU computing,
- gradient boosted trees – XGBoost library provide a parallel algorithm[7],
- cross-validation – each iteration of cross-validation may be computed independently,
- an ensemble of models – components of ensemble models may be calculated independently.

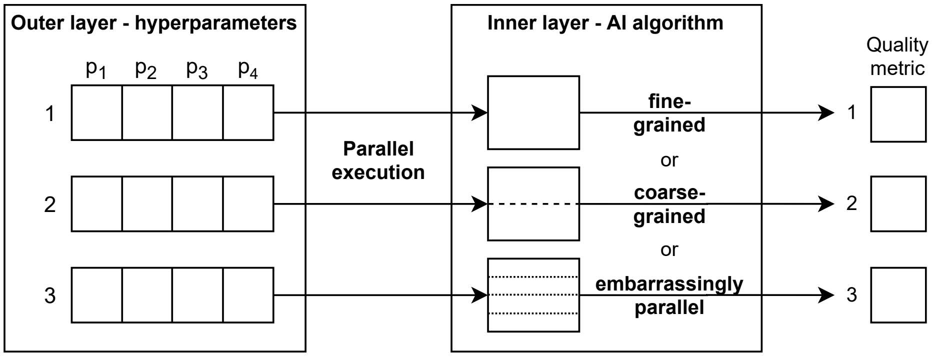


Figure 8.5: Parallelization in machine learning
Source: own work

[7]https://xgboost.readthedocs.io/en/latest/

The above subsection focused on the model's complexity and parallelization opportunities during training and validation phases but it is worth noting that algorithms' performance characteristics change while serving predictions. In effect, training and prediction capabilities should be addressed and planned separately during the system design phase. The next subsection introduces a few aspects of deployment performance considerations. In addition to training and prediction phases, machine learning models utilize additional components with distinct performance profile. In particular, scoring systems are often connected with interpretation algorithms such as SHAP, see Chapter 7, or (S. Lundberg & Lee, 2017)) or LIME (Ribeiro et al., 2016b). Prediction interpretability is a desired feature of credit scoring models, in some cases enforced by law (Goodman & Flaxman, 2017). Interpretation methods have their own performance specification and may need a closer inspection to be used effectively.

8.2.3 Model deployment

After training and testing the process, the scoring model is deployed for production use and enters the maintenance period. The deployment phase has a specific performance consideration depending on the retraining policy and predictions serving model. Additionally, the model should be monitored, in particular, for performance of predictions (e.g. latency of response) and quality of predictions (accuracy, precision or other metrics) – see Chapter 6. Performance issues are often intermittent and caused by a sudden hardware failure or a spike in request rate. Quality degradation is a long-term process caused by *concept drift* (Gama et al., 2004). The idea behind the drift is that the distribution of new data is changing compared to data on which model was trained on. For the scoring system, a possible increase in the number of bad credits may appear due to recession, thus the model should be retrained to reflect the shift in data.

Retraining approaches include three main categories:

- one-off training in which the model is retrained in irregular intervals, usually when predictions quality degradation is spotted or new input data is available,

- batch training requires regular model updates based on gathered data,
- real-time (online training) is applicable only to selected AI algorithms that can be trained on a continuously incoming stream of data.

In the context of retraining policy, computing resources allocation is the main factor (Figure 8.6). Online training requires small but constant compute supply, batch training is distributed in predictable spikes and one-off training requires varying supply. Additionally, online learning handles concept drift problem more effectively than batch or one-off training. On the other hand, online training is not always available, either due to algorithm design or data preparation limitations. For scoring applications, due to the long data gathering process, the most relevant training scheme is one-off and batch training with long retrain intervals. Often 12 to 18 months are required to mark credits as good or bad and prepare them for the model update. Furthermore, online training is mostly reserved for the retraining process in a production environment. In development and testing environments, models are trained with batches of data on demand.

Another specific trait of credit scoring models is inherent difference between training and production data. The data set on which the model is trained is wide and short (many columns, a small number of rows) due to inclusion of a broad range of predictors such as behavioral and demographic data. On the other hand, new data obtained for classification is narrow and long (a few columns, many rows). Reduced number of predictors stems from restricted access to use particular client's information or limited bank's knowledge about the new customer. From a technical perspective, the development/testing environment may vary from production and reverse migrations may be needed to fine-tune the model.

More consideration is required while planning infrastructure for specific serving model which can be *batch* or *real-time*. With the batch-serving approach, the system produces predictions for many records at once and often the response time requirements are relaxed. The batch serving works well if request batches have predictable size and frequency (similar approach as in batch retraining). In real-time approach, a prediction is provided for a single record on demand and low response latency is a requirement,

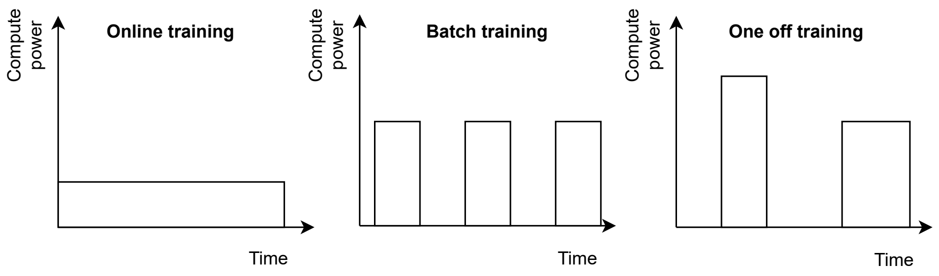


Figure 8.6: Load characteristics of retraining policies
Source: own work

thus an appropriate infrastructure and technology may be required to control the performance. For real-time prediction, machine learning models are usually served as a web-service (REST API), coupled with message broker (Kafka, Pub/Sub) or as a built-in application feature. Streaming (message) applications may handle an increased load well due to internal buffering features (e.g in Pub/Sub which is Google Cloud Platform message broker). In case of a web server or in-app prediction high scalability and load balancing, it may be needed to provide low-latency predictions. For this task, the Kubernetes cluster in a cloud environment with horizontal scaling enabled is an effective solution: containerized machine learning model is deployed on each node in a cluster, load balancer distributes traffic and AutoScaler accommodates spikes in requests by increasing nodes count. More information on model deployment may be found in Chapter 9.

8.3 Platforms for building scoring models

Another approach to boost model building is distributed computing. In distributed computing, workload is divided between nodes (separate machines) in a cluster. It is worth noting that each of the nodes can additionally perform parallel computations on assigned subtask (e.g. some nodes may use CPU multithreading and other GPUs accelerations). Typical properties of modern distributed systems include the following:

- the system handles failures of individual computers and may continue computations after a node exclusion,

- structure of the cluster network is not known in advance, the system may consist of different kinds of computers, network links and may change during the execution of a distributed program,
- each node has only a limited, incomplete view of the system,
- working in a master-slave architecture, in which one node is elected as a master and coordinate the work of all other nodes (slaves).

Distributed computing platforms have rich software ecosystems with both open-source and commercial applications. In the context of machine learning processing, the following open-source components are often used:

- Hadoop – software framework for distributed storage (HDFS), processing of big data using the MapReduce programming model and cluster management (YARN)
- Spark – is an open-source cluster-computing framework prevalent in machine learning domain
- Kubernetes with Docker containers – open-source container-orchestration system for automating application deployment, scaling and management

Many modern products are built on top of the above-mentioned solutions, adding additional functionality. For example, Databricks built on Spark provides collaborative notebooks, integrations tools or additional performance tweaks (Delta Lake format). Kubernetes is used as a foundation for Kubeflow – AI models deployment framework. Another extension of Spark is Sparkling Water from H2O.ai company which enables H2O algorithms in high-performance Spark cluster.

Hadoop is a framework that started the Big Data era and is still a foundation for modern, cutting-edge applications. Software built on top of Hadoop is sometimes referred to as *Hadoop ecosystem* and few of these (e.g. Spark) has ecosystem of its own. Three core components of Hadoop are:

- HDFS – Hadoop Distributed File System provides distributed storage.

- MapReduce – algorithm for distributed processing.
- YARN – Yet Another Resource Negotiator is a layer governing computational and storage resources within a Hadoop cluster.

HDFS provides a logical structure with files stored in directories (folders) – which resembles traditional file systems. The main difference is that files in that hierarchical structure are stored in multiple machines instead of one. From an architectural perspective, HDFS cluster consists of NameNode (master) and DataNodes (slaves). NameNode is a machine organizing operations and storing system metadata while DataNodes store files and communicate with the master if any changes occur. HDFS is designed to be used on top of commodity hardware and handle system failures e.g. node OS or disk failure, Internet connection outage. Each file stored in HDFS is divided into blocks (usually 128MB in size) which are replicated in accordance to replication policy (3 replicas by default). Every replica is stored on a different machine (node) or even different machine racks to handle both node outage or whole rack unavailability (Figure 8.7). To detect problems in the cluster, each DataNode sends a Heartbeat message to the NameNode periodically. The NameNode marks DataNodes without recent Heartbeats (10 minutes is the time-out by default) as dead and excludes blocks stored on the machine from the file system. As there are replicas on the other machines, the system's functionality is not affected and the dead node can be healed automatically (e.g. rebooted) or manually by the operations team. NameNode stores information (metadata) about all files in HDFS and coordinates both operations within a cluster and communication with client applications. To further improve robustness, NameNode can operate on multiple machines kept in sync.

HDFS provides a framework to distribute storage but Big Data workload won't be possible without distributed computing. For that purpose, Hadoop contains MapReduce framework which may process huge amounts of data in-parallel while maintaining reliability and fault-tolerance (Dean & Ghemawat, 2004). A MapReduce job splits the input data set into independent chunks which are processed by the *map* tasks in a parallel manner. Map phase output is sorted and passed to *reduce* tasks which may aggre-

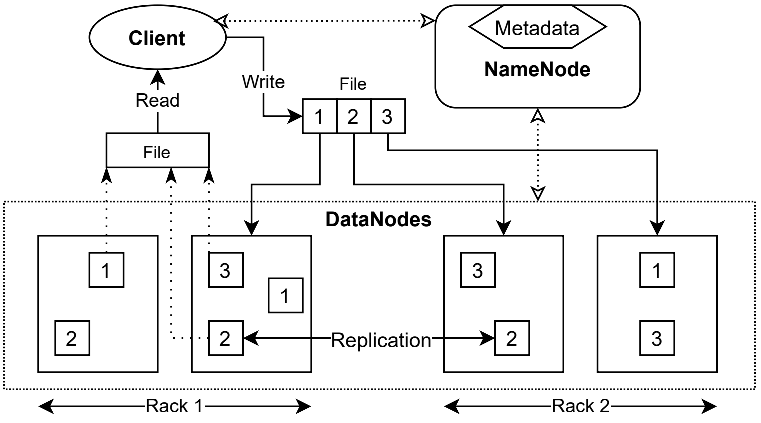


Figure 8.7: HDFS architecture scheme
Source: own work

gate intermediate results to final output (Figure 8.8). Typically both the input and the output of the job are stored in HDFS and the framework takes care of scheduling, monitoring and re-execution of the failed tasks. To effectively execute scheduled tasks, MapReduce and the HDFS are running on the same set of nodes, which allows to do the processing on the nodes where data is already present.

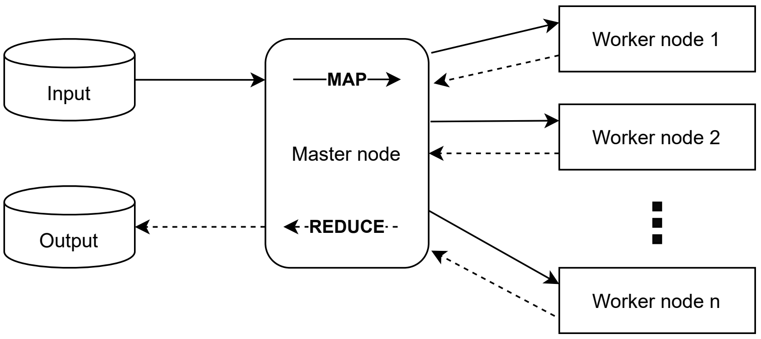


Figure 8.8: MapReduce algorithm
Source: own work

In the first version of Hadoop, MapReduce was responsible for everything above the storage layer, including assignment of cluster resources and data processing. Consequently, higher-level frameworks had to be built on top of MapReduce which is designed for batch workloads. It became obvious that a more general and more flexible platform was necessary. In order to generalize processing capability, resource management and fault-tolerance, components were transferred into YARN, effectively decoupling cluster operations from the data pipelines. The emergence of YARN democratized usage of HDFS and led to the creation of many purpose-built frameworks (Figure 8.9). MapReduce was altered to run on top of YARN as all other application frameworks.

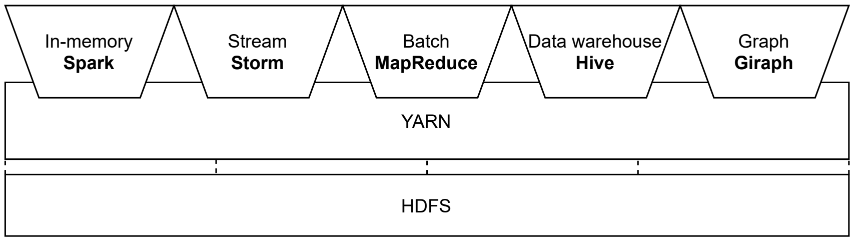


Figure 8.9: High-level architecture of Hadoop-based platforms
Source: own work

One of the commonly used Hadoop-based frameworks for machine learning workloads (scoring applications in particular) is Spark[8]. From the performance perspective, the main Spark's advantage lies in in-memory computations that can be many times faster than MapReduce jobs based on hard drives storage (compare IO speed in Table 8.2). Furthermore, Spark contains an extensive set of libraries including *Spark SQL*, *MLlib* for machine learning, *GraphX* for graphs processing and *Spark Streaming*. With these built-in functionalities, Spark serves as a unified platform for multiple types of analytical workloads.

Spark operates on its own abstraction structure called *Resilient Distributed Data set* (*RDD*). Data loaded into Spark is partitioned (distributed) and then can be further processed using two types of operations: transformations which create a new data set from an existing one and actions which return a value to the core engine after running a computation on the data set. All transformations in Spark are *lazy*, which means they are not carried out when called in code but instead when action is later applied to defined transformations. This design enables Spark to run more efficiently and

[8]https://spark.apache.org/

optimize the execution. Beside RDDs, Spark also exposes Dataframes and Data sets structures. Spark Dataframe is conceptually equivalent to a table in a relational database or a data frame in R/Python. Compared to basic RDD, Spark performs additional optimization on Dataframes based on data content and requested transformations. Dataframes provide a fast and flexible tool for interactive analysis and preprocessing of data which may be combined with MLlib and GraphX algorithms into a single workflow using *ML Pipelines* module.

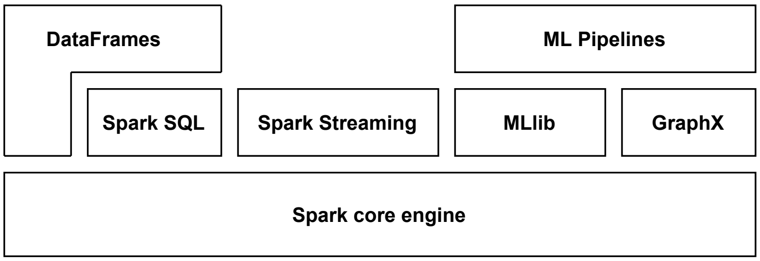


Figure 8.10: Main Spark components
Source: own work

Another big advantage of Spark is easy integration both with sources and target applications. As aforementioned, a Spark cluster can be run on Hadoop YARN but also on other platforms e.g. cloud services such as AWS EC2 or Kubernetes. Data can be ingested directly through one of many connectors and exposed outside of Spark with commonly used drivers. Development in Spark is based on a flexible, multi-language approach. The framework itself is written in Scala but provides interfaces in Python, Java, R and SQL which allows developers to quickly learn Spark workflow.

Hadoop and Spark can be launched on on-premise clusters but many companies decide to use PaaS solutions offered by all major cloud providers. The main reason for that is high infrastructure maintenance overhead in an on-premise setup. Multiple machines used as cluster nodes require operational support both in the hardware and software context. Moreover, the computation capacity must be planned far ahead due to the non-elastic character of proprietary IT infrastructure. In a cloud environment, VMs maintenance and ensuring machines uptime is on the provider's side. Confidence in clusters stability is reflected in strict SLAs. Azure HDInsight and AWS EMR provide 99,9% uptime guarantee and GCP Dataproc 99,5%. All software updates are handled automatically, including operating systems, security patches and enabling new Spark/Hadoop versions. Furthermore, compute resources can be scaled dynamically to provide capacity during peak demand and release free machines when not needed. Flexible scaling leads to distinction to interactive clusters for analytics and job-based clusters created from scratch and destroyed after finishing the workload. In the cloud, the processing is mostly separated from storage. Data may be stored on blob storage instead and mounted on a cluster on demand, which allows for more effective budget management compared to on-premise.

Cloud distributions of Big Data open-source frameworks also contain many custom extensions and improvements compared to basic versions. They integrate well with other cloud services such as AI and Machine Learning platforms (Azure ML Studio, GCP AI Platform, AWS SageMaker). AWS and Azure also provide Databricks service built on Spark but with a focus on collaboration and handling well data analysis, data science and data engineering. A wide range of interconnected cloud products may easily cover all tasks needed in building a credit scoring model.

An important feature of cloud solutions in the context of scoring applications is built-in security. As data for AI and Big Data frameworks usually reside in cloud storage, these platforms benefit from automatic encryption in the storage layer. From the network perspective, all cloud services are hosted in a cloud network separated from the public Internet and may be secured with additional firewalls, proxies and other network-based mechanisms. Banks have to take special precautions in storing and exposing sensitive data which in the cloud may be handled with dedicated redaction and de-identification tools such as Google's Cloud Data Loss Prevention or dynamic data masking in Azure. From the financial industry standpoint, another important feature is cloud vendors compliance with multiple regulations, in particular information security standards ISO/IEC 27001 and SOC 2, GDPR compliance and financial specific standards such as Payment Card Industry DSS or FINRA in the US. For the sake of the auditing process, audit logs are gathered as a part of cloud-wide logging services such as Stackdriver in GCP and AWS or Azure Monitor.

With an abundance of available modeling frameworks and data processing technologies, preparing modern credit scoring application architecture requires extensive expertise and continuous tracking of new solutions. Another factor to consider is an ongoing shift from on-premise computing to hybrid and cloud services which are heavily influencing the financial sector. Research has found that 54% of banks adopted or prepared a strategy to adopt cloud solutions by the end of 2021 (Finastra, 2019). Amongthe main reasons for the switch, institutions mentioned simplification of the IT infrastructure, cost reduction and increase in cloud regulatory compliance. Moreover, following the innovation and current IT architecture trends is much easier with a cloud-based approach, as vendors pursue cutting-edge technologies and constantly upgrade their offerings.

Chapter 9

Techniques for implementing models in decision engines

Aleksander Nosarzewski

Decision Analysis and Support Unit
Collegium of Economic Analysis
SGH Warsaw School of Economics

Even after the credit scoring model has been trained and extensively validated, it is far from being a finished product. The last step on its journey is being deployed in a production environment, so that it can start helping solve the problem it was designed for. Unfortunately, deploying the model adequately is not an easy task. Moreover, the art of model deployment is also just beginning to establish and is far from being a well-defined field.

In this chapter, we first present challenges which might arise during the model deployment. Later, the most popular formats utilized for exporting the model are described. Consequently, we present available methods of model deployment, along with some guidelines with regard to choosing the proper one. Finally, we provide a review of good practices for model deployment, as well as a variety of most popular tools utilized in this field.

9.1 Challenges in model deployment

Producing a final model is not the end of its business life cycle – after it has been built and trained, it needs to be properly implemented into production systems and deployed which can be a cumbersome task. Reports show that up to 88% of corporate machine learning and artificial intelligence initiatives are struggling to move beyond test stages (Sallomi & Lee, 2017). This is due to the fact that machine learning (ML) code is just a small fraction of real-world ML systems which additionally is not straightaway compliant with production environments (Sculley et al., 2015). Such a mismatch stems mainly from differences between data science and production environments.

Data scientists are experts in algorithms and associated mathematics. The key component of their work is to test a vast range of different approaches to a given problem in order to find the right solution to it. To do so, they need tools which are relatively easy to use and flexible with regards to their needs – data scientists use high-level programming languages, such as Python or R, or dedicated statistical software (e.g. SAS) (Talagala, 2018).

On the other hand, IT developers, who are responsible for deploying the models, often have little or no ML knowledge (Talagala, 2018). However, they are experts in deployment and management of software and services in production. They use low-level programming languages, such as C++, C# or Java which are more complex than ones utilized by data scientists but offer better performance which is crucial for production systems. As a result, there are two types of mismatches between data science and IT teams – expertise and environment mismatches.

There are two main approaches to overcoming these mismatches as presented in Figure 9.1 (Braun, 2017).

In the first one (presented at the top), both teams are kept separately. Data scientists work independently while they develop the final model using the environment of their choice (e.g. Python or R, along with preferred libraries). When finished, they produce a detailed specification document which then needs to be implemented by the developers in the production environment. Such an approach works but has several disadvantages.

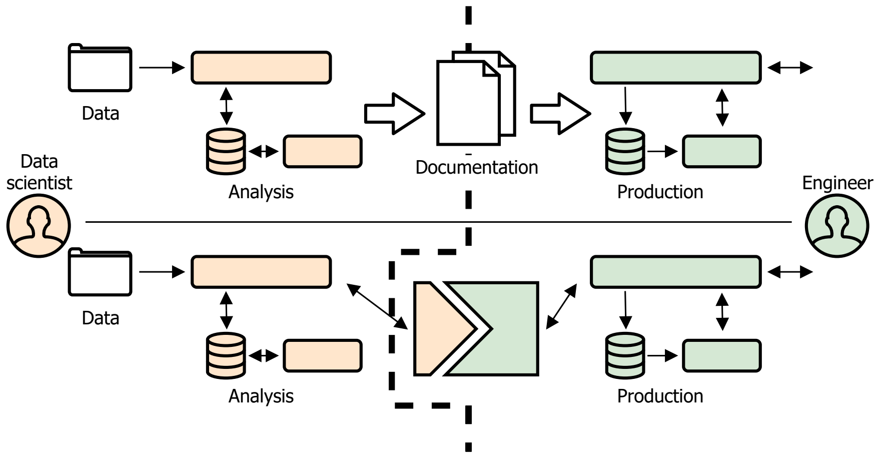


Figure 9.1: Two collaboration modes between data scientists and engineers
Source: own work based on the Braun (2017)

Firstly, it requires the two teams to communicate efficiently. This means having experts in the team who know both the environment used to create the model and the production environment or ensuring that the documentation provided by the data scientists is comprehensive and exhaustive.

Secondly, if a need to use another modeling method emerges (e.g. random forest instead of logistic regression), the production environment needs to be reconfigured from scratch. Moreover, the cost of implementing the model increases with the complexity of the statistical method used. Linear or logistic regression are relatively simple to implement but other methods, such as gradient boosting or neural networks are much more complex. Furthermore, a custom implementation of a utilized algorithm might be inefficient performance-wise, compared to the well-developed and optimized libraries which were used to train the model.

Thirdly, models do not always use raw data only – feature engineering (i.e. applying various transformations to the original data) can be used to transform the data into a desirable format. If the model is to be directly implemented in a production system, each data transformation needs to be handled separately as well. Moreover, if the new features are used in future iterations of the model (e.g. $\ln(x_1)$ instead of x_1^2), they need to be implemented before the model can be deployed. Various operations utilized for data transformations can be implemented differently in particular languages as well (e.g. one can use strict inequalities, while the other utilizes non-strict ones) which can lead to disparate model predictions.

Finally, this approach is error-prone, since it requires a lot of manual work. It can be also time consuming – implementing an algorithm's logic can be a lengthy process. In extreme cases, the model can be no longer up-to-date after it is finally implemented – behavior patterns of clients can change overtime and there can be some shifts in the data affecting the model (Samiullah, 2019).

The second approach (presented at the bottom of Figure 9.1) is to introduce an intersection between the two environments – a part of the production system with a clearly identified interface which can be used by data scientists to *plug-in* their models.

The code which communicates with the production system would naturally need to follow stricter software development practices than usual data science code (Braun, 2017) – it should accept input data and return its forecasts in a predetermined way, compliant with production system requirements.

However, engineers would no longer need to implement the model manually. The data science team would provide them with an *object* (i.e. a file in a predetermined format) containing algorithm's logic and executing its scoring based on data provided by the production system. Ideally, the object could also handle all required data transformations. This way, engineers do not need to know how the models work, they just need to provide an infrastructure which would handle the *model object* execution properly.

Moreover, such an approach allows data scientists to push models for deployment by themselves, without the need of the engineers' work. They can just provide an updated model object to the production system, provided that there is a proper logging process implemented.

Finding the best practices for collaboration and communication between data science teams and information technology (IT) professionals while automating and productizing machine learning algorithms, is a new discipline called *MLOps*, see (Talagala, 2018). The name is a compound of machine learning (ML) and operations (Ops). This is still a rapidly developing discipline and lacks well-defined good practices and standards, however it already offers solutions which are useful for conscious and reliable model deployment (Dong, 2017; Gallatin, 2019).

9.2 Methods of exporting *model* object

Before the finished model can provide any business profit, it needs to be implemented in a production environment. This section provides an overview of methods used for creating a *model object* introduced in Section 9.1 and exporting it from the modeling environment, so that it can be used for deployment.

9.2.1 Code porting

The simplest method of model deployment is the so-called *porting* – rewriting the code of a model we want to implement from the language used by the data science team to a production environment language. This is equivalent to the first approach described in Section 9.1.

Once the algorithm's logic is implemented in the desired programming language, the most popular practice is to export model parameters to a JSON file (Baatout, 2017). Doing so provides a relatively easy method of updating the model's specification, such as its coefficients, as long as the same algorithm is used.

There is a tool which allows automating the porting of the model to some extent – a Python library `m2cgen`[1]. It translates Python code to another desired language. Currently, it supports export to C, C#, Go, Java, JavaScript, PHP, PowerShell, R and Visual Basic. The majority of machine learning algorithms are supported: linear models, SVM (Support Vector Machine), decision trees, random forests and gradient boosting

[1]https://github.com/BayesWitnesses/m2cgen/blob/master/README.md

methods, with implementations in a variety of libraries (e.g. scikit-learn). However, the `m2cgen` library does not support exporting data transformations, so they still need to be handled by the engineers manually.

9.2.2 PMML

PMML (Predictive Model Markup Language) is the natural answer to the problems associated with code porting. It is a cross-platform standard for saving forecasting models and is based on the XML language. PMML's development began in the late 1990s by Data Mining Group (DMG). Its main propagator was Professor Robert Grossman et al. (1999).

PMML deos not only sets the standard for saving various types of models but also enables the export of various data transformations (used for both pre- and post-processing). The current version of PMML standard (4.4) supports most machine learning models, excluding deep neural networks (although simple neural networks are supported). It also provides support for basic data transformations (normalization and discretization of variables, value mapping and aggregation), as well as some basic operators and built-in functions.

PMML[2] is supported by most of the tools currently used in the data science (Levitan, 2018) – SAS products (Base and Enterprise Miner) and R (`r2pmml`, `PMML` and `PMMLTransformations` libraries) and Python languages (export to PMML format is supported by libraries, including popular scikit-learn, using `sklearn2pmml`), as well as many production systems (e.g. Microstrategy, Amazon AWS oracle, Spark and Teradata).

Unfortunately, PMML is a legacy standard – many versions have appeared over the years, each covering a different range of supported models and data transformations. Furthermore, different products support only certain PMML versions. Therefore, it is possible that the production environment will not be able to support the exported model, although it supports the PMML standard (but only in the older version).

In addition, PMML supports only a limited number of model classes, while new ones are introduced late. It also does not support increas-

[2]http://dmg.org/pmml/products.html

ingly popular deep neural networks. Moreover, only basic data transformations are supported, without the possibility of defining custom, more complex ones.

PMML is therefore a good solution only when standard models and data transformations are used. It is also crucial to verify whether modeling and production environments support compatible versions of the standard.

9.2.3 ONNX

The ONNX (Open Neural Network eXchange) format is a more modern equivalent of PMML. It is supposed to provide *interoperability* of models – the possibility to use them regardless of the environment and libraries used. ONNX format is based on so-called *computational graphs*, used by many deep and machine learning libraries to represent the algorithms' logic. However, it provides consistency between different approaches (for example, some libraries use static graphs, while others use dynamic graphs). In other words, ONNX guarantees that the model will always behave in the same way for a given data set and results obtained will be independent of both modeling and production environment (McDonald, 2019).

ONNX format comes in two variants: the base ONNX for deep neural networks and ONNX-ML dedicated to classic machine learning algorithms. Moreover, `ONNXruntime` library is also available, allowing for deployment of the model in the production environment in a fully compatible manner. Currently, the following languages are supported: Python, C#, C++, C, Java and Ruby. In addition, this library provides support for scoring models with both: CPUs (Central Processing Unit – a standard processor) and GPUs (Graphics Processing Unit which provides higher computing performance). It also supports various accelerators offered for these architectures (for example NVIDIA CUDA for GPUs).

The only drawback is the young age of the format (McDonald, 2019). It was created in 2017 and although the documentation is constantly expanding, it is still limited. Only selected libraries are directly supported as well. However, many independent packages are created to provide full compatibility of ONNX format with popular tools. It is ,therefore, a format with the potential of becoming a new standard for exporting models.

9.2.4 Environment-dependent formats

Apart from formats created with interoperability in mind, there are ones specific to the environment in which the model was built.

In Python, there is a *pickle* format used to export various types of objects. It enables a so-called *serialization* of objects (i.e. instances of individual classes) – transforming them into a bitstream file which can be saved to a disk or sent to another computer or process. In particular, an object containing the implementation of a ready-made predictive model can be a subject to serialization (in Python, the term *pickling* is adopted from the format name). Moreover, it is possible to include any custom data transformations inside the pickled model (Baatout, 2017).

In R ,there are two formats for saving objects: `.RDS` and `.RDA`. Both allow saving and exporting created models but there is a slight difference between them. After loading, the RDA format restores the exact copy of the exported object in the working environment, including its name, which is not a convenient solution in case of model deployment. On the contrary, the RDS format enables a full serialization of models. Therefore, instead of an exact copy of the object, its representation is exported which after loading can be assigned to any variable.

Apart from formats typical for a given programming language, there are also formats supported by specific tools or libraries. Most popular machine learning libraries have their own formats for saving ready models (such as SavedModel for `TensorFlow` or POJO and MOJO for `h2o`) (Kervizic, 2019).

The disadvantage of using environment-dependent formats is that the production environment must be compatible with the environment used to create the model. For example, a model exported by using pickle format can only be deployed in a production environment with available Python installation. However, there is a solution (discussed in the next section) which provides a workaround of this disadvantage in a relatively simple way – deploying the model as a microservice (Baatout, 2017).

9.3 Model deployment

Once the model is built and exported to one of the formats described in Section 9.2, it is time to deploy it in a production environment. This section outlines the methods for implementing the model and discusses the impact of various approaches to model scoring and retraining with regards to presented deployment methods.

9.3.1 Methods of model deployment

Depending on the method used to export the model, implementing it may require different solutions and workload (Koen, 2019).

Code porting requires a manual implementation of the model in a production environment by an engineering team, which can be a lengthy process, even if the implementation of the model in the appropriate language will be provided by a data science team. This is due to the fact that following items need to be handled: data used by the model, delivered in an appropriate format, adequate processes initializing model scoring and proper handling of the model's outputs (Baatout, 2017).

If the model has been exported to an interoperable format (e.g. PMML or ONNX) which is also supported by the production environment, deployment of the model usually boils down to loading it into the environment and integrating it properly via built-in tools which, apart from time savings, provides also a better optimization in terms of scoring performance and is less error-prone (McDonald, 2019).

When the format to which the model has been exported requires the reconstruction of a strictly defined environment, the model can be made available as a microservice – a self-encapsulated application which can be treated as a black box with regards to the production system, as presented in Figure 9.2.

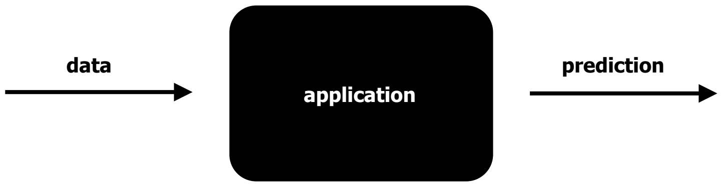


Figure 9.2: Communicating with the model as a microservice
Source: own work based on the Danka (2020)

The production system does not know how the model operates internally. Microservice offers a strictly defined way of communication – an API (Application Programming Interface) which enables communication between the model and the production system. Through the API, the system is able to send the data to the model and receive the predictions from it. Usually, inputs and outputs are provided in a structured form – a JSON or YAML file (Danka, 2020). A JSON input to the API could look like this:

```
{
"data":
[[0.00632,18,2.31,0,0.538,6.575,65.2,1,296,15.3,396.9,4.98]]
}
```

while the response obtained from the model (containing the forecast) would similarly be in a form as below.

```
{
"data":[25.813999999999993]
}
```

Since the production system is only responsible for providing and receiving the data in an appropriate format, all steps required to produce a model's prediction should happen inside of the application (microservice). These steps consist of (as presented in Figure 9.3): processing the input data provided by the production environment, making predictions with the model built-in within the application, processing the outputs of the model and providing these outputs in an understandable format back to the production environment.

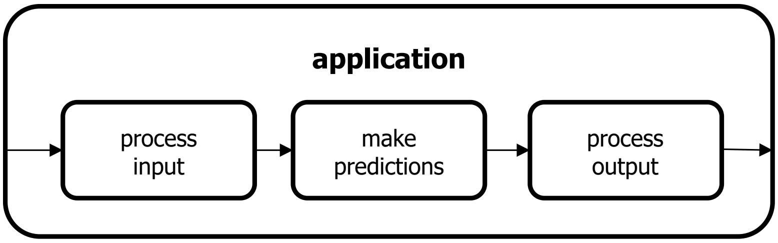


Figure 9.3: The insides of the model as a microservice application
Source: own work based on the Danka (2020)

In order to make the model available as a microservice, it needs to be provided with a properly configured environment. This can be achieved by using the open-source Docker software. It allows for creating special containers which encapsulate fully-working environments, configured for the model (for example, a Docker container can have the appropriate version of Python language installed, as well as the libraries utilized by the model, such as scikit-learn or TensorFlow). Such an environment is fully isolated from the production system and can communicate with it only through API. Docker containers can be easily run on servers, regardless of their operating system and configuration which makes the Docker a very versatile solution (Elfouly, 2019). Thanks to this, the model can be easily transferred between different types of platforms on which it would work. It can be deployed both locally and utilizing the bank's servers or even in the cloud. What is more, it is also possible to use distributed computing platforms (such as Spark or Kubernetes) which provide an easy and fast way of scaling the model (Nazrul, 2018).

9.3.2 Model scoring and retraining in terms of model deployment

The way the model should be deployed depends a great deal on how it will be trained and scored (Fletcher, 2019). As already mentioned in Chapter 8, there are three main approaches with regards to model (re-)training: one-off, batch and real-time. When it comes to model scoring, there are batch and real-time scoring.

In one-off training, the model is prepared by a data scientist and then deployed in a production environment where it continues to work until its performance deteriorates enough so that it is called upon back to the data science team for refreshing or a total rebuild (Kervizic, 2019). This is a traditional approach used in credit scoring (Sousa et al., 2016).

In batch training, the initial model can be trained by a data science team (allowing it to take advantage of their expert and domain knowledge) but then the model weights could be adjusted automatically, as new data becomes available at some fixed intervals of time (batches).

In real-time training, the model is learning constantly, as soon as new data becomes available. Such an approach is quite rarely used, especially in highly regulated areas such as credit scoring. Nevertheless, batch approaches are gaining in popularity (Sousa et al., 2016).

The more frequently the model is being updated, the more flexible solution to model deployment would be desired. For infrequent one-off model training, porting the code manually could be acceptable – and switching to more agile methods could not be worth the costs associated with it. However, if the utilized model starts to change more frequently, it becomes reasonable to switch towards more universal approaches, such as using some standardized formats or microservices (Kervizic, 2019).

When it comes to model scoring, choosing a deployment method is a bit simpler. In real-time scoring, the model's forecast should be produced on demand, whenever some trigger occurs (e.g. a client submits an online form for an overdraft). In batch prediction, the model is run in predefined moments. Predictions can be produced for the whole population (e.g. for pre-approved cash loans) or only customers for whom some change in data (e.g. they started using a new type of banking product) or a trigger event (e.g. credit application was filed) appeared since previous batch run (Kervizic, 2019).

Since real-time environments are complex, manual porting of the code in such production systems is not advised (Fletcher, 2019). Most of such environments work best with models served as microservices which are easily scalable if the system load increases. For batch scoring, every presented method can be used, with all their advantages and disadvantages.

9.3.3 Choosing model deployment method

There is a plethora of model export formats and multiple deployment methods. A natural question would be – which one is the best and which ones to use? While there is no universal answer, there are some factors worth considering while choosing the deployment method. Methods presented in Subsection 9.3.1 are discussed below.

Code porting. Implementing the model manually in a production system is rarely the desired solution. Such an approach might be apt for one-off projects, however once a more flexible infrastructure is available, even such projects can be implemented quicker and in a more reliable way (Kervizic, 2019). Moreover, in credit scoring, models are constantly monitored and improved if their performance deteriorates, therefore truly one-off projects occur rarely and being able to deploy models quickly becomes desirable (Sousa et al., 2016).

Built-in integration tools. Many production systems provide built-in solutions for deploying machine learning models. This is a convenient solution since it allows using tools already at hand and simplifies the integration process which is then partially automated. However, such systems often require that the model is exported to a particular format (such as PMML (Levitan, 2018) or ONNX (McDonald, 2019)) which may differ between the systems. This can be problematic when switching to another system – then redevelopment of scoring models might be required as well. Moreover, these formats usually handle only basic data transformations, so there still might be some feature engineering which would need to be implemented by developers separately. Using built-in integration tools and some standardized export formats should be a sufficient solution if utilized algorithms and data transformations are standard (Kervizic, 2019).

Model as a microservice. This is the most versatile solution. It requires the data scientists to follow stricter software development practices, however offers a great deal of flexibility. Not only can it handle *any* algorithm and data transformation but also allows the data science team to use any tools they require and are most comfortable with (Kervizic, 2019). Additionally, it is compatible with most production environments and engineers are well accustomed to dealing with microservices served as Docker containers (Husain, 2017). Models encapsulated as microservices can also be utilized regardless of production solution – its vendor or a way of distribution (on internal servers, in the cloud or with distributed computing platforms).

9.4 Good practices in MLOps

There are some good practices stemming from software development which are industry standards and could be also useful for simplifying model deployment and managing model's lifecycle (Elfouly, 2019). This section covers the most important ones: ensuring reproducibility, version control, proper testing and various deployment approaches.

9.4.1 Reproducibility

It is a good practice to write the code used to build the model so that it is reproducible from the start (meaning all the steps can be easily reproduced in another environment or using another computer) (Samiullah, 2019). In particular, it should be ensured that model inputs and outputs are always in the same format. Special attention should also be paid to used metadata, configuration files, dependencies, formats, etc. For example, changing the coding method of a binary explanatory variable (change of the reference class), formatting of dates from DD/MM/YYYY to MM/DD/YYYY or measurement units from unity to thousands can all have a substantial impact on the way the model works.

9.4.2 Version control

Version control is an important factor in building and managing the model. Developers are successfully using code versioning systems such as Git. However, in the data science world, in addition to the code containing implementation of the model itself, the data, based on which the model works, are also a key element of the design (Bitzer, 2019). It can dynamically change – through adding new observations or using various transformations to obtain new variables, all of which can affect the final model (e.g. variables used or optimal parameter values). In particular, if the production data deviates from the data based on which the model was trained, the production model may not work at all. Therefore, in addition to managing the version of the code, it is necessary to manage the data as well – in a way that connects it to the code. In other words, being able to assign the model to the appropriate version of the code and the data based on which it was created is crucial (Vázquez, 2019).

Another useful functionality is being able to easily monitor data science experiments in a way which allows comparing the quality of the model with regards to its specification (multiple runs of different versions of the code, based on different data sets and with different model parameter values). This way, the data scientist would always be able to track which approaches have proved to be fruitful and which not. What is more, it makes data science team coordination easier – each team member is able to track which models have already been tested which leads to less work duplication (Vázquez, 2019).

Correct version control can be also crucial with regard to legal compliance. Proper model documentation and logging is often required by the regulator (e.g. see Polish Financial Supervision Authority (2013, 2015)). Being able to strictly determine by which version of the model a credit scoring for a given client was obtained might also be required with regard to the customers' *right to explanation* of the model's score, mentioned by the General

Data Protection Regulation (GDPR) (European Parliament, 2016; Goodman & Flaxman, 2016).

9.4.3 Model testing

Code testing is another good practice derived from software development which should be introduced to the model deployment. Once the model is already built and implemented, it is a good idea to create *unit tests* which will be responsible for checking whether the model, as a whole, works correctly (Samiullah, 2019). A unit test is the smallest and simplest form of software testing, employed to assess a separable functionality of the tested software – the model, in this case (Sculley et al., 2015). Examples of what unit tests for models should cover are presented below.

It is worth to design simple tests checking if the data provided to the model is in the appropriate format (e.g. whether it contains all the required variables) and whether it takes appropriate values (e.g. whether the client's age is a positive value, within a reasonable range). Model outputs should also be covered by a separate set of tests (e.g. verifying whether scoring probabilities are from range $[0, 1]$) (Böhm, 2019).

Classical methods of model *performance monitoring* can also be used to validate the technical correctness of the model's implementation. For example, a rapid decrease in the model's predictive quality may result from incorrect loading of the data. Discrepancies between the model's performance in training and production environment may also point to other technical problems, such as utilizing other versions of libraries or tools (which can greatly affect the way the model works) (Samiullah, 2019).

Another type of testing involves preventing bugs from sneaking back into the model. It is a good idea to create a separate test for each historically known issue which would ensure that it does not occur again in the future. This approach is called *regression testing* and is designed to easily detect problems already known but which may be accidentally reintroduced into the model. By having such tests, recurring problems are easy and cheap to diagnose and repair, see (Sculley et al., 2015).

It is also worth to test the model's performance with regards to the speed of its execution – this is a *benchmark testing*. Some model may provide high-quality predictions but might be virtually useless business-wise due to very long response times. When the model is scored in batches (and usually outside business hours) this might be of lesser importance. However, for real-time scoring (e.g. when deciding whether to offer an overdraft for a client who wants to transfer some money via online banking but lacks enough funds), waiting time of several minutes might be unacceptable (Samiullah, 2019, Sculley et al., 2015).

9.4.4 Deployment approaches

When the model being implemented is supposed to replace the existing one, it is worth checking (preferably in a safe way) whether the new model really offers better forecast quality than the previous one and whether it integrates correctly with the production environment (Gonfalonieri, 2019).

The first solution is to implement the model in a so-called *shadow mode* (Sridharan, 2017). It involves launching a new model in a production environment identical to the ones used in the previous version of the model but without using the results of the tested model in the business process – outputs of the tested model are only saved and can then be used to analyze how the model works. This approach ensures no real unexpected consequences in a production environment, even if the model proves to be flawed. The drawback is that it requires doubling the computing resources available for a given process, which is not always possible or cost-effective.

An approach that requires fewer resources is *canary testing* (Sculley et al., 2015). In a canary test, the new model is launched only for a small subset of the customer population. This way, the model can be tested in reality, while minimizing the risk of failure. If the model is satisfactory, it can be gradually launched for an increasing proportion of customers or made available to the entire population. Customers subject to canary testing can be selected in different ways: randomly, based on their geolocation or other features of interest. This allows testing the model for specific target groups (Sridharan, 2017).

It is also possible to combine both approaches – the new model can be launched only for a part of the population in the shadow mode. Sometimes, however, it is necessary to use the model in real conditions, e.g. when it is a part of a business process with which the customer has a direct contact. If the model is used in real-time, deploying it only for a small part of the population allows one to verify whether it will work efficiently under real traffic load (Samiullah, 2019).

9.5 Tools useful for MLOps

Currently, there are already algorithms, tools and libraries for machine learning which can be recognized as mature (Dong, 2017). However, the MLOps domain is still relatively young and thriving (Gallatin, 2019) and there are no solutions which can be considered as a standard (Dong, 2017; Gallatin, 2019). Luckily, there are tools which are gaining popularity, develop rapidly and are supported by major players in the field of machine learning (Marranc, 2018).

9.5.1 Data Version Control

The first solution is DVC (Data Version Control)[3]. Initially, this open source project focused only on managing both model code and related data sets conveniently. Currently, it supports:

- version control of machine learning projects (models, data and intermediate files),
- managing the results of experiments (tracking the results in a unified format),
- model deployment and sharing (tools that allow to easily share a model with others and send it to a production environment).

This tool is based on the popular Git version control system and is language- and framework-independent. Additionally, it supports various data storage formats: locally, on an internal server and in the cloud.

[3]https://dvc.org/

9.5.2 MLflow

MLflow[4] is an open source tool used for comprehensive machine learning project management. It consists of three components:

- **MLflow Tracking** – serves as a system for version control of the model and data, as well as for managing experiments by monitoring and comparing parameters and metrics of individual models.
- **MLflow Projects** – used to share ML projects in a form that allows them to be recreated by other users.
- **MLflow Models** – provides tools for deployment of various types of models, with support for many environments and production formats.

Additionally, this tool works with any programming language and library. It also has built-in integration tools for the most popular solutions:

- languages used in data science: Python and R,
- libraries utilized to build models: scikit-learn, PyTorch, Keras, TensorFlow, H2O, XGBoost and LightGBM,
- different export formats and model deployment methods: ONNX, Docker containers with Python REST API and objects for integration with cloud services: Microsoft Azure ML, Amazon SageMaker or Apache Spark UDF.

Furthermore, if a language or library are not directly supported by MLflow, it provides tools for adding custom integrations as needed.

MLflow's biggest advantage are so-called *flavors*. MLFlow makes it possible to save models in many flavors at once – e.g. as a Python or R function, in ONNX format and one native for scikit-learn and XGBoost libraries. Thanks to this, people preferring different environments can work together on the same model – for example, a gradient boosting model could have

[4]https://mlflow.org/

been built in the past by a person using XGBoost library, while the data scientist who is currently responsible for updating model parameters, prefers the scikit-learn library. By using MLflow they can easily integrate these two environments.

9.5.3 Kubeflow

Kubeflow[5] is a tool designed to enhance and simplify the implementation of machine learning models on Kubernetes which is a *container orchestrator* – it ensures the automation of container creation and enables simple scaling of processes based on them. For example, if a given Docker container encapsulating the model is not able to handle all queries submitted by the production system, the Kubernetes environment will automatically start another copy and redirect excessive traffic to it.

Therefore, Kubeflow can be considered as a *glue* connecting Kubernetes environment with individual libraries and tools utilized for model design and export (Bisong, 2019).

Moreover, Kubeflow is constantly further developed to provide additional functionalities, such as version control of entire ML projects, experiment monitoring and also model export. Some of these features are still provided in a beta phase only (at the time of writing – March 2020) but in the future it will be possible to use Kubeflow either for comprehensive management of machine learning projects (such as MLflow) or just for Kubernetes support (which is already available).

9.5.4 Commercial solutions

There are also commercial solutions which offer a comprehensive model management throughout its entire life cycle – from model building, through its implementation and continuous monitoring.

Cloud-based products are the most popular, with Amazon SageMaker, Microsoft Azure Machine Learning, Google Cloud AI Platform or Databricks as market leaders (Marranc, 2018). However, due to security reasons,

[5]https://www.kubeflow.org/

the environments used in banks for credit scoring are usually isolated from the Internet, so using cloud-based solutions is not possible then. There are also solutions that can be used inside own infrastructure, e.g. Dataiku, Algorithmia or Microstrategy. Most database system providers also offer partially or fully automated modules for model implementation and management (although they are often limited only to specific formats and functionalities).

Most commercial products use or at least provide support for the previously mentioned open source tools, so it is possible to use both, depending on the organization's needs.

Conclusions

The title and content of this monograph are directly related to three elements: 1) the business problem (i.e. creditworthiness assessment), 2) the analytical instrument (i.e. modern machine learning methods) and 3) the method enabling the integration of the previous two (i.e. interpretability framework for machine learning methods – XAI). However, the combination of these three elements is not yet sufficient to call the use of machine learning methods in credit decision responsible. The approach to responsible credit scoring requires a thorough understanding of the business context of data sources for credit scoring (Chapter 2) and the sensitivity of individual methods to data artifacts (Chapter 5), too. Practical applications of the techniques described in this book require also the highest standards in the model validation process (Chapter 6), as well as an appropriate technological architecture and implementation of these models. The technical robustness of these algorithms is a multidimensional issue – only a mature approach to managing this process will ensure an appropriate level of transparency and resilience.

As a part of this monograph, we presented the particular issues related credit scoring, which have to be taken into account w hen building both machine learning models as well as in traditional approaches. The fact that we raised them simultaneously when describing modern techniques is related to the fact that they are generally much more resistant to these issues than the classical approaches currently utilized by banks. Additionally the techniques described in this monograph that enable the automation of handling of common data quality related issues (e.g. missing data). In classic credit scoring, the burden of handling such problems is transferred to the data analysts who, through multiple repeated steps (unfortunately also exposed to the human perception error), try to best fix problems with data in the final model. The techniques described in this book make it possible to, at least partially, automate this process, which may undoubtedly translates into an increase in the efficiency of model building, without sacrificing its quality.

The second aspect which, in our opinion, puts the machine learning models discussed in this monograph in a favorable light is the fact that they are much more flexible as to the form of functional relations between features and target variable. They are able to automatically identify non-linear relationships and interactions between features that may difficult to discover for humans. Specialists in the field of classical econometrics, who point out that classical models can also be "taught" non-linear relationships (e.g. by carrying out non-linear transformations on explanatory variables), are of course right. Even very complex relationships (i.e. interactions between variables, nonlinearities) can be modeled with simple techniques. However, this process is time consuming and requires personnel which is highly qualified in both quantitative methods and the details of the business problem. This is why analytical departments of many institutions are extensive (sometimes a single analyst is responsible for maintaining the definition of only a few variables). Machine learning techniques can be successfully used as tools supporting experts in discovering these dependencies, even if the final decision model is built using classical modeling techniques. Machine learning models should not be perceived as a threat to the classic form of data modeling but rather as a set of modern tools supporting the analyst in discovering the relationships that affect credit scoring. Obtaining high quality parameters of these models is not an ultimate goal in itself as above all, the objective is to prepare a robust analytical tool that supports the decision making process.

Moreover, the modern approach is not something significantly different from classical credit scoring modeling. Of course, the form of the model as well as the way of interpreting parameters change, but the process itself remains similar. This situation is best reflected in the issues discussed in Chapter 2 or Chapter 6. Note that, in fact, in these two chapters describing the foundations of credit scoring (i.e. data and validation), the processes look very similar, and often they are the same! In terms of these chapters, we intentionally tried to describe the techniques so that they would be applicable both to classical credit scoring and to scoring based on modern analytical techniques. We are aware that XAI techniques constitute an additional layer – but wouldn't classical methods require a similar layer if nonlinearity and interactions between variables were included in their definitions?

However, transformation related to the use of advanced analytical methods is not an instant switch to different methods and solutions. Our recommendation in this regard is the responsible and gradual introduction of machine learning solutions – also because the interpretability of these models may not be straightforward in terms of credit scoring. These techniques can initially be used as supporting models, i.e. analysis of their specification and parameterization could help identify previously unknown relationships between variables (or the identification of defects in existing models). Such solutions, at least at the beginning, will not be directly related to the business process, but they will enable indirect verification of the results obtained using classical methods. It is this gradual familiarization with new analytical solutions that will allow banks to get to know them well, gain skills and expertise in the field on the way and more importantly, to gain confidence of the environment not only in terms of their applicability but also their adequacy.

The aspect of collective perception of these methods is also important – as long as these methods are called "black-box" techniques, both business practice and regulatory bodies may remain skeptical about their use. However, we would like to draw your attention to the thread that we already mentioned at the very beginning of the monograph in Chapter 1 – credit scoring is derived from pragmatism and empiricism, both of them speak for the start of applying machine learning methods in credit scoring.

The predictions about the future use of machine learning methods, not only in banking, indicate mostly their further development and advancement. Even today, these techniques are widely used in many areas of human activity, such as medicine, biotechnology, art, marketing, or technology. The amount and diversity of data that we collect every day nowadays speaks for the need to automate analytical processes and utilizing machine learning is an inherent part of that.

We hope that the collection of methods and reflections discussed in the monograph will help readers develop disruptive extensions of their scoring approaches.

Editors

Bibliography

Aas, K., Jullum, M., & Løland, A. (2019). Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *arXiv preprint arXiv:1903.10464*.

Amdahl, G. M. (1967). Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, In *Proceedings of the april 18-20, 1967, spring joint computer conference*, Atlantic City, New Jersey, Association for Computing Machinery. https://doi.org/10.1145/1465482.1465560

Anderson, R. (2007). *The credit scoring toolkit: Theory and practice for retail credit risk management and decision automation*. Oxford University Press.

Apley, D. W., & Zhu, J. (2016). Visualizing the effects of predictor variables in black box supervised learning models.

Arya, V., Bellamy, R. K., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q. V., Luss, R., Mojsilović, A., Et al. (2019). One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*.

Azevedo, A. I. R. L., & Santos, M. F. (2008). KDD, SEMMA and CRISP-DM: a parallel overview. *IADS-DM*.

Baatout, A. (2017). Putting machine learning in production [Retrieved 7 March 2020].

Bank, E. C. (2017). ECB Guide on materiality assessment (EGMA).

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Et al. (2019). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *arXiv*, arXiv–1910.

Basel Committee on Banking Supervision, Bank For International Settlements. (2005a). *International convergence of capital measurement and capital standards* (tech. rep.) [dostęp: 2012.08.30]. Basel Committee on Banking Supervision, Bank For International Settlements. dostęp: 2012.08.30.

Basel Committee on Banking Supervision, Bank For International Settlements. (2005b). Studies on the validation of internal rating systems. *Working Paper*.

Basel Committee on Banking Supervision, Bank For International Settlements. (2005c). *Studies on validation of internal rating systems, working paper no. 14* (tech. rep.). Basel Committee on Banking Supervision, Bank For International Settlements.

Bech, M., & Gyrd-Hansen, D. (2005). Effects coding in discrete choice experiments. *Health economics*, *14*(10), 1079–1083.

Belsley, D. A., Kuh, E., & Welsch, R. E. (2005). *Regression diagnostics: Identifying influential data and sources of collinearity* (Vol. 571). John Wiley & Sons.

Benjamin, N., Cathcart, A., & Ryan, K. (2006). Low default portfolios: A proposal for conservative estimation of default probabilities. *Financial Services Authority*.

Biecek, P. (2018). DALEX: explainers for complex predictive models in R. *The Journal of Machine Learning Research*, *19*(1), 3245–3249.

Biecek, P., & Burzykowski, T. (2021). *Explanatory Model Analysis: Explore, Explain, and Examine Predictive Models* (1st). Chapman; Hall/CRC. https://pbiecek.github.io/ema/

BIS. (2020). *Basel Committee on Banking Supervision*. https://www.bis.org/list/bcbs_all/sdt_1/index.htm

Bisong, E. (2019). Kubeflow for poets [Retrieved 7 March 2020].

Bitzer, M. (2019). How to use data version control (DVC) in a machine learning project [Retrieved 7 March 2020].

Böhm, T. (2019). How to use Test Driven Development in a Data Science Workflow [Retrieved 7 March 2020].

Bolton, C. Et al. (2010). *Logistic regression and its application in credit scoring* (Doctoral dissertation). University of Pretoria.

Braun, M. (2017). What is hardcore data science – in practice? [Retrieved 7 March 2020].

Breiman, L. (1996). Some properties of splitting criteria. *Machine Learning*, *24*(1), 41–47.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Bretscher, O. (1997). *Linear algebra with applications*. Prentice Hall Eaglewood Cliffs, NJ.

Brown, L. D., Cai, T. T., & DasGupta, A. (2001). Interval estimation for a binomial proportion. *Statistical science*, 101–117.

Bücker, M., Szepannek, G., Gosiewska, A., & Biecek, P. (2020). Transparency, Auditability and eXplainability of Machine Learning Models in Credit Scoring.

Bzdok, D., Altman, N., & Krzywinski, M. (2018). Points of significance: Statistics versus machine learning. Nature Publishing Group.

Caruana, R., Karampatziakis, N., & Yessenalina, A. (2008). An empirical evaluation of supervised learning in high dimensions, In *Proceedings of the 25th international conference on machine learning*.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms, In *Proceedings of the 23rd international conference on machine learning*.

Castro, J., Gómez, D., Molina, E., & Tejada, J. (2017). Improving polynomial estimation of the Shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research*, *82*, 180–188.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system, In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*.

Chollet, F. (2018). *Deep Learning with Python*. Manning.

Cohen, M. (2012). Cotton, Capital, and Ethnic Networks: Jewish Economic Growth in the Postbellum Gulf South. *American Jewish Archives Journal*, *64*, 112–36.

Cramer, J. S. (2002). The origins of logistic regression.

Daly, A., Dekker, T., & Hess, S. (2016). Dummy coding vs effects coding for categorical variables: Clarifications and extensions. *Journal of choice modelling*, *21*, 36–41.

Danka, T. (2020). How to properly ship and deploy your machine learning model [Retrieved 7 March 2020].

Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters, In *Osdi'04: Sixth symposium on operating system design and implementation*, San Francisco, CA.

Dong, C. (2017). The evolution of machine learning [Retrieved 7 March 2020].

Dresner Advisory Services. (2018). 2018 Big Data Analytics Market Study.

Duan, T., Avati, A., Ding, D. Y., Basu, S., Ng, A. Y., & Schuler, A. (2019). Ngboost: Natural gradient boosting for probabilistic prediction. *arXiv preprint arXiv:1910.03225*.

Dunteman, G. H., & Ho, M.-H. R. (2005). *An introduction to generalized linear models* (Vol. 145). Sage Publications.

ECB. (2020). *European Central Bank*. https://www.ecb.europa.eu/paym/pol/policies/html/index.en.html

Edward Miller, G. (1991). Asymptotic test statistics for coefficients of variation. *Communications in Statistics-Theory and Methods*, *20*(10), 3351–3363.

Elfouly, S. (2019). MLOps Done Right [Retrieved 7 March 2020].

Engelmann, B., Hayden, E., & Tasche, D. (2003). *Measuring the discriminative power of rating systems* (tech. rep.). Discussion Paper Series 2.

Engelmann, B., & Rauhmeier, R. (2006). *The Basel II risk parameters: estimation, validation, and stress testing*. Springer Science & Business Media.

European Banking Authority. (2017a). *Guidelines on PD estimation, LGD estimation and the treatment of defaulted exposures.*

European Banking Authority. (2017b). Guidelines on the application of the definition of default under Article 178 of Regulation (EU) No 575/2013. *EBA/GL/2016/07.*

European Banking Authority. (2020). Report on Big Data and advanced analytics. *EBA/REP/2020/01.*

European Central Bank. (2019). *Instructions for reporting the validation results of internal models.*

European Commission. (2019). *Ethics guidelines for trustworthy AI* (Report). European Commission.

European Parliament. (2016). General Data Protection Regulation [Retrieved 7 March 2020].

Everitt, B., & Skrondal, A. (2010). Standardized mortality rate (SMR). *The Cambridge Dictionary of Statistics*, *409.*

Fantazzini, D., & Figini, S. (2009). Random survival forests models for SME credit risk measurement. *Methodology and computing in applied probability*, *11*(1), 29–45.

Finance, Y. (2013). How FICO became 'the' credit score. Yahoo! https://finance.yahoo.com/news/fico-became-credit-score-100000037.html

Finastra. (2019). Cloud Banking - Innovation without Limits Report.

Finlay, S. (2012). *Credit Scoring, Response Modeling, and Insurance Rating: A Practical Guide to Forecasting Consumer Behavior*. Palgrave Macmillan UK.

Fisher, A., Rudin, C., & Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, *20*(177), 1–81.

Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, *222*(594-604), 309–368.

Fletcher, J. (2019). Putting Machine Learning Models into Production [Retrieved 7 March 2020].

Flynn, M. J. (n.d.). some computer organizations and their effectiveness. *IEEE Transactions on Computers.*

Fox, G., Otto, S., Lyzenga, G., & Rogstad, D. (1985). The Caltech concurrent computation program-Project description.

Fox, J., & Monette, G. (1992). Generalized collinearity diagnostics. *Journal of the American Statistical Association*, *87*(417), 178–183.

Frączkowski, K. (2003). Zarządzanie projektem informatycznym: Projekty w środowisku wirtualnym, czynniki sukcesu i niepowodzeń projektów. *Wrocław*, Oficyna Wydawnicza Politechniki Wrocławskiej.

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*(771-780), 1612.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, *38*(4), 367–378.

Furnival, G. M., & Wilson, R. W. (2000). Regressions by leaps and bounds. *Technometrics*, *42*(1), 69–79.

Gallatin, K. (2019). The Rise of the Term "MLOps" [Retrieved 7 March 2020].

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection, In *Advances in artificial intelligence – sbia 2004*, Berlin, Heidelberg, Springer Berlin Heidelberg.

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, *24*(1), 44–65.

Gonfalonieri, A. (2019). Why is machine learning deployment hard? [Retrieved 7 March 2020].

Goodman, B., & Flaxman, S. (2016). European union regulations on algorithmic decision-making and a "right to explanation".

Goodman, B., & Flaxman, S. (2017). European Union Regulations on Algorithmic Decision-Making and a Right to Explanation. *AI Magazine*, *38*, 50–57.

Google. (2016). *Google supercharges machine learning tasks with TPU custom chip*. Google. https://cloud.google.com/blog/products/gcp/google-supercharges-machine-learning-tasks-with-custom-chip

Google. (2017). *Build and train machine learning models on our new Google Cloud TPUs*. Google. https://blog.google/products/google-cloud/google-cloud-offer-tpus-machine-learning/

Gosiewska, A., & Biecek, P. (2020). Do Not Trust Additive Explanations.

Greene, W. H. (2003). *Econometric analysis*. Pearson Education India.

Greenwell, B. M. (2017). pdp: An R Package for Constructing Partial Dependence Plots. *R J.*, *9*(1), 421.

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2017). Correlation and variable importance in random forests. *Statistics and Computing*, *27*(3), 659–678.

Grossman, R., Bailey, S., Ramu, A., Malhi, B., Hallstrom, P., Pulleyn, I., & Qin, X. (1999). The Management and Mining of Multiple Predictive Models Using the Predictive Modeling Markup Language (PMML) [http://papers.rgrossman.com/journal-016.pdf].

Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., Tatham, R. L., Et al. (1998). *Multivariate data analysis* (Vol. 5). Prentice hall Upper Saddle River, NJ.

Hall, P. (2019). *An introduction to machine learning interpretability*. O'Reilly Media, Incorporated.

Hand, D. J. (2005). Good practice in retail credit scorecard assessment. *Journal of the Operational Research Society*, *56*(9), 1109–1117.

Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*(1), 29–36.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer Science & Business Media.

Heinze, G., & Schemper, M. (2002). A solution to the problem of separation in logistic regression. *Statistics in medicine*, *21*(16), 2409–2419.

Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. *The annals of statistics*, 1171–1220.

Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Vol. 398). John Wiley & Sons.

Hotz, H. (2006). A short introduction to game theory. *Retrieved on*, *21*, 2017.

Husain, H. (2017). How Docker Can Help You Become A More Effective Data Scientist [Retrieved 7 March 2020].

ITRS. (2015). International Technology Roadmap for Semiconductors.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Janc, A., & Kraska, M. (2001). *Credit-scoring: Nowoczesna metoda oceny zdolności kredytowej*. Biblioteka Menedżera i Bankowca Zarządzanie i Finanse.

Johnston, J., & DiNardo, J. (1997). *Econometric methods*. McGraw-Hill Companies, Inc.

Kamiński, B., & Zawisza, M. (2012). *Receptury w R: podręcznik dla ekonomistów*. Szkoła Główna Handlowa. Oficyna Wydawnicza.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *29*(2), 119–127.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree, In *Advances in neural information processing systems*.

Kennedy, P. (2003). *A guide to econometrics*. MIT press.

Kervizic, J. (2019). Overview of different approaches to deploying machine learning models in production [Retrieved 7 March 2020].

Kłosok, M., & Chlebus, M. (2020). Towards better understanding of complex machine learning models using Explainable Artificial Intelligence (XAI)-case of Credit Scoring modelling. *Faculty of Economic*

Sciences, University of Warsaw Working Papers. https://www.wne.uw.edu.pl/files/9815/9355/2268/WNE_WP324.pdf

Koen, S. (2019). Architecting a Machine Learning Pipeline.

Koronacki, J., & Ćwik, J. (2008). *Statystyczne systemy uczące się*. Akademicka Oficyna Wydawnicza EXIT.

Kupiec, P. (1995). Techniques for verifying the accuracy of risk measurement models. *The J. of Derivatives*, *3*(2).

Lehmann, E. L., & Romano, J. P. (2006). *Testing statistical hypotheses*. Springer Science & Business Media.

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), 124–136.

Lessmanna, S., Seowb, H., Baesenscd, B., & Thomasd, L. C. (2013). Benchmarking state-of-the-art classification algorithms for credit scoring: A ten-year update, In *Credit research centre, conference archive*.

Levitan, S. (2018). State of the Art Model Deployment.

Lewis, E. M. (1992). *An introduction to credit scoring*. Fair, Isaac; Company.

Liao, D., & Valliant, R. (2012). Condition indexes and variance decompositions for diagnosing collinearity in linear model analysis of survey data. *Survey Methodology*, *38*(2), 189–202.

Liebert, F. (2017). Zarządzanie projektami w przedsiębiorstwach branży IT – studium literaturowe. *Zeszyty Naukowe. Organizacja i Zarządzanie / Politechnika Śląska*, *z. 101*, 271–284.

Lin, J. (2017). The Lambda and the Kappa. *IEEE Internet Computing*, *21*, 60–66. https://doi.org/10.1109/MIC.2017.3481351

Louppe, G., Wehenkel, L., Sutera, A., & Geurts, P. (2013). Understanding variable importances in forests of randomized trees, In *Advances in neural information processing systems*.

Louzada, F., Ara, A., & Fernandes, G. B. (2016). Classification methods applied to credit scoring: Systematic review and overall comparison. *Surveys in Operations Research and Management Science*, *21*(2), 117–134.

Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S.-I. (2020). From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, *2*(1), 2522–5839.

Lundberg, S., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *ArXiv*.

M2cgen [Retrieved 7 March 2020]. (2019). Retrieved 7 March 2020. https://github.com/BayesWitnesses/m2cgen/blob/master/README.md.

Macskassy, S. A., Provost, F., & Rosset, S. (2005). ROC confidence bands: An empirical evaluation, In *Proceedings of the 22nd international conference on machine learning*.

Marranc, M. (2018). Building The AI Stack [Retrieved 7 March 2020].

Matuszyk, A. (2004). *Credit scoring: Metoda zarządzania ryzykiem kredytowym*. CeDeWu.

Mayes, E. (2004). Credit Scoring for Risk Managers. The Handbook for Landers. Thomson South-Western. Mason. Ohio. *Links*.

McCulloch, C. E., & Neuhaus, J. M. (2014). Generalized linear mixed models. *Wiley StatsRef: Statistics Reference Online*.

McDonald, J. (2019). ONNX – Made Easy [Retrieved 7 March 2020].

Mentch, L., & Hooker, G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*, *17*(1), 841–881.

Molnar, C. (2019). *Interpretable Machine Learning: A guide for making black box models explainable*.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, *38*(8).

Morawski, W. (2003). *Kronika kryzysów gospodarczych*. Wydawnictwo Trio.

Narendra, P. M., & Fukunaga, K. (1977). A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, (9), 917–922.

Nazrul, S. S. (2018). DevOps for Data Scientists: Taming the Unicorn [Retrieved 7 March 2020].

Nelson, B. D. (2001). Variable reduction for modeling using PROC VARCLUS, In *Proceedings of the twenty-sixth annual sas users group international conference, cary, nc, sas institute.*

Nielsen, D. (2016). *Tree boosting with xgboost-why does xgboost win every machine learning competition?* (Master's thesis). NTNU.

Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 2018). Determination press San Francisco, CA.

NVIDIA. (2006). *NVIDIA Unveils CUDA™-The GPU Computing Revolution Begins.* NVIDIA. https://www.nvidia.com/object/IO_37226.html

O'brien, R. M. (2007). A caution regarding rules of thumb for variance inflation factors. *Quality & quantity*, *41*(5), 673–690.

Oesterreichische Nationalbank. (2004). *Guidelines on credit risk management: Rating models and validation.* Oesterreichische Nationalbank.

Oesterreichische Nationalbank. (2006). Guidelines on Credit Risk Management: Credit Approval Process and Credit Risk Management. *Otto Wagner Platz*, *3*.

on Banking Supervision, B. C. (2009). Principles for sound stress testing practices and supervision. BIS Basel.

on Banking Supervision, B. C. (2018). Stress testing principles.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pluto, K., & Tasche, D. (2011). Estimating probabilities of default for low default portfolios, In *The basel ii risk parameters.* Springer.

PMML 4.4 - General Structure [Retrieved 7 March 2020]. (2019). Retrieved 7 March 2020. http://dmg.org/pmml/v4-4/GeneralStructure.html.

Polish Financial Supervision Authority. (2013). *Recommendation D on the mangement of the information technology and ICT environment security at banks.*

Polish Financial Supervision Authority. (2015). *Recommendation W on model risk management in banks.*

Potharst, R., & Feelders, A. J. (2002). Classification trees for problems with monotonicity constraints. *ACM SIGKDD Explorations Newsletter*, *4*(1), 1–10.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features, In *Advances in neural information processing systems.*

Pruska, K. (2009). Estimation of Variance of Logistic Regression Predictors for Small Areas.

Przanowski, K. (2014). *Credit Scoring w erze Big-Data.* Oficyna Wydawnicza SGH.

Przanowski, K. (2011). Banking retail consumer finance data generator-credit scoring data repository. *Finansowy Kwartalnik Internetowy e-Finanse.*

Quinlan, J. R. (2014). *C4. 5: programs for machine learning.* Elsevier.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016a). Why should I trust you? Explaining the predictions of any classifier, In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.*

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016b). Why Should I Trust You?: Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Roszkowski, M. (2015). Skalowalność systemów informatycznych w obszarze e-administracji. *Zeszyty Naukowe Uniwersytetu Szczecińskiego. Ekonomiczne Problemy Usług*, (nr 117 Wirtualizacja gospodarki - spojrzenie interdyscyplinarne), 603–621.

Roth, A. E. (1988). *The Shapley value: essays in honor of Lloyd S. Shapley.* Cambridge University Press.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*, 206–215.

Sallomi, P., & Lee, P. (2017). Machine learning: Things are getting intense, In *Deloitte technology, media and telecommunications predictions 2018*, Deloitte.

Samek, W., Wiegand, T., & Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Samiullah, C. (2019). How to deploy machine learning models [Retrieved 7 March 2020].

SAS. (2013). SAS/STAT 13.1 User's Guide: The LOGISTIC Procedure. SAS Publishing.

Scallan, G. (2013). Marginal Kolmogorov - Smirnov Analysis: Measuring Lack of Fit in Logistic Regression. *Credit Scoring Conference CRC, Edinburgh*. http://www.scoreplus.com/resources/reference/index.php

Scallan, G. (1999). Building Better Scorecards. *Autin*.

Scallan, G. (2011). Selecting characteristics and attributes in logistic regression, In *Credit scoring conference crc, edinburgh*.

Schwalbe, K. (2013). *Information Technology Project Management*. Cengage Learning.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems, In *Nips proceedings*.

Selvitella, A. (2017). The ubiquity of the simpson's paradox. *Journal of Statistical Distributions and Applications*, *4*(1), 1–16.

Shao, J., & Tu, D. (2012). *The jackknife and bootstrap*. Springer Science & Business Media.

Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing*, *5*(4), 13–22.

Siddiqi, N. (2012). *Credit risk scorecards: Developing and implementing intelligent credit scoring* (Vol. 3). John Wiley & Sons.

Simpson, E. H. (1951). The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, *13*(2), 238–241.

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods, In *Proceedings of the aaai/acm conference on ai, ethics, and society.*

Sousa, M. R., Gama, J., & Brandão, E. (2016). A new dynamic modeling framework for credit risk assessment [https://repositorio.inesctec.pt/bitstream/123456789/5314/1/P-00G-T4F.pdf]. *Expert Systems With Applications*, *45*, 341–351.

Sridharan, C. (2017). Monitoring in the time of cloud native [Retrieved 7 March 2020].

Stein, R. M. (2005). The relationship between default prediction and lending profits: Integrating ROC analysis and loan pricing. *Journal of Banking & Finance*, *29*(5), 1213–1236.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for Random Forests. *BMC Bioinformatics*, *9*(1), 307.

Strumbelj, E., & Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, *11*, 1–18.

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, *41*(3), 647–665.

Talagala, N. (2018). Why MLOps (and not just ML) is your Business' New Competitive Frontier [Retrieved 7 March 2020].

Tel, G. (2001). *Introduction to Distributed Algorithms* (2nd). USA, Cambridge University Press.

Thomas, L., Crook, J., & Edelman, D. (2017). *Credit scoring and its applications.* SIAM.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Tikhonov, A. N., & Arsenin, V. Y. (1977). Solutions of ill-posed problems. *New York*, 1–30.

Tologi, L., & Lengauer, T. (2011). Classification with correlated features: Unreliability of feature ranking and solutions. *Bioinformatics*, *27*(14), 1986–1994.

Varian, H. (2014). Machine Learning and Econometrics. *Slides package from talk at University of Washington.*

Vázquez, F. (2019). Data version control with DVC. What do the authors have to say? [Retrieved 7 March 2020].

Verstraeten, G., & Van den Poel, D. (2005). The impact of sample bias on consumer credit scoring performance and profitability. *Journal of the operational research society*, *56*(8), 981–992.

Welfe, A. (2018). *Ekonometria.* Polskie Wydawnictwo Ekonomiczne.

Zhao, Q., & Hastie, T. (2019). Causal interpretations of black-box models. *Journal of Business & Economic Statistics*, 1–10.

Zhao, X., Barber, S., Taylor, C. C., & Milan, Z. (2018). Classification tree methods for panel data using wavelet-transformed time series. *Computational Statistics & Data Analysis*, *127*, 204–216.

Zhou, Q., Chen, W., Song, S., Gardner, J. R., Weinberger, K. Q., & Chen, Y. (2015). A reduction of the elastic net to support vector machines with an application to gpu computing, In *Twenty-ninth aaai conference on artificial intelligence.*

List of Tables

List of Figures

Index

About the authors (alphabetical order)

Przemysław Biecek, Ph.D., D.Sc., professor at University of Warsaw and Warsaw University of Technology. Graduated in Software Engineering and Mathematical Statistics. His main research area is related with predictive modeling of large and complex data, model interpretability and data visualization. His main research project is DrWhy.AI i.e. collection of tools and methods for exploration, explanation and debugging of predictive models. Other research activities are focused on Evidence Based Machine Learning and Responsible Artificial Intelligence. Przemysław is an author of over 80 research articles and four books related to data analysis. He participated in research fellowships at UCDavis (USA), UHasselt (Belgium) or NTU (Singapoure). An R and Python enthusiast. Przemysław has over 15 years of experience in supporting R&D teams in high-tech companies such as Netezza, IBM, Samsung, Disney or iQor.

Kamil Cerazy, M.Sc., is a mathematics graduate of the Adam Mickiewicz University in Poznań. His professional career revolves around applications of mathematics in finance.Kamil has over six years of consulting experience in the area of financial risk, working internationally with top tier banks and corporates. He specializes in market risk but his broad interests include credit risk, climate risk and digital transformation.

Marcin Chlebus, Ph.D., is a head of Data Science programme at Faculty of Economic Sciences at University of Warsaw. In his academic live he focuses on building system for supporting business decisions using different predictive and segmentation methods (econometrics models, bagging and boosting algorithms, neural networks, clustering, homogenous & heterogeneous ensembling). His main field of research is risk modelling (credit & market risk), recently he focused on building best practices for using Explainable Artificial Intelligence (XAI) methods in decision making models. He is a member of COST action (FinAI – Fintech and Artificial Intelligence in Finance -Towards a transparent financial industry).

Bogumił Kamiński, Ph.D., (editor) is the head of the Decision Analysis and Support Unit, Institute of Econometricsat SGH Warsaw School of Economics and a professor at the Data Science Laboratory, Ryerson University, Canada. He specializes in the use of advanced methods of data analysis to support decision-making in enterprises. Bogumił is the author of over one hundred articles on the business applications of forecasting, optimization, and simulation methods. He actively promotes the use of the latest research results in economic practice. Prof. Kamiński has over twenty years of experience in implementing IT solutions in the largest Polish companies and institutions.

Daniel Kaszyński, M.Sc., (editor) is an assistant at the Decision Analysis and Support Unit, Institute of Econometrics at SGH Warsaw School of Economics. In his research and professional work, Daniel gets involved in the construction and validation of credit risk models, numerical optimization issues, as well as technological consulting. Daniel has over eight years of consulting experience in the area of financial risk and implementation of IT solutions.

Marta Kłosok, M.Sc., graduated from Faculty of Economic Sciences at University of Warsaw. She is passionate about developing data-driven solutions for variety of business and real-life problems. In professional work she implements advanced statistical and machine learning concepts. As a data scientist Marta devises models and algorithms, discovers patterns hidden in the data, reveal relevant characteristics about customers. In academic research she has been focused on concepts and tools to monitor and analyse complex machine learning models predictions.

Łukasz Kraiński, M.Sc., is a Ph.D. candidate at the Institute of Econometrics at SGH Warsaw School of Economics. In the scientific area, he works on the design and implementation of multi-agent simulations and algorithms on graph data. Professionally, Łukasz specializes in cloud technologies, data engineering, and DevOps practices. Łukasz holds multiple certificates of renowned IT service providers, incl. Google, Microsoft, or Oracle.

Maciej Kwiatkowski, M.Sc., is a credit risk modelling expert with 19 years of experience in over 10 countries, in retail and corporate lending area. He held managerial positions in group modelling and validation teams in Citigroup, KBC, and recently in Bank Pekao. He was the owner of group methodology of credit scoring. He also participated in development and implementation of group methodology for IFRS 9 and estimation of lifetime cost of risk for various products. In recent years he worked on implementation of Big Data solutions in credit scoring and assessment of credit capacity. Maciej holds an M.Sc in Applied Mathematics from University of Warsaw and M.Phil in Economic Theory and Econometrics from University of Cambridge.

Aleksander Nosarzewski, M.A., is a Ph.D. student at the Decision Analysis and Support Unit, Institute of Econometrics at SGH Warsaw School of Economics. In his research work, he focuses on games theory and text mining. Professionally, Aleksander has over three years of experience in Anti-Money Laundering modelling.

Łukasz Opiński, M.Sc., is an assistant at the Collegium of Economic Analysis at, SGH Warsaw School of Economics. In his research and professional work, Łukasz gets involved in multi-agent simulation and numerical optimization projects. Łukasz passed all three levels of the CFA Program.

Karol Przanowski, Ph.D., assistant professor, Statistical Methods & Business Analytics Unit, Institute of Statistics and Demography at SGH Warsaw School of Economics. Master degree in mathematics and Ph.D. in physics at University of Lodz. He is trying to cover all credit scoring topics by science research and practical solutions possible to implement in banking environment. Experienced in consumer finance portfolio analysis, data simulations and modelling various processes for that portfolio. He provides unique lecture entitled "Credit Scoring and macroprogramming in SAS". Author of books in Polish published by Oficyna Wydawnicza SGH: „Credit Scoring – studia przypadków modeli biznesowych" (Credit Scoring – model business case studies) and „Credit Scoring w erze Big Data" (Credit Scoring in Big Data era). Responsible in banks for: IRB and IFRS9 modelling, predictive model building, implementing and monitoring, automatic CRM processes, multichannel campaign management, automatic budget and planning processes.

Kinga Siuta, M.Sc., is an Advanced Analytics – Big Data graduate and a research assistant in Collegium of Economic Analysis of SGH Warsaw School of Economics. In her research, she focuses on complex network analysis, optimization algorithms, and multi-agent modeling. Her most recent projects include logistics network optimization for Polish Post and wind data analysis for the Polish Olympic sailing team. She is experienced in the academic teaching of optimization methods and data analysis techniques. Kinga also has broad experience in consulting. In her professional work, Kinga performs economic analyses for clients from various sectors of the economy, including financial institutions, develops applications, and interactive dashboards.

Tomasz Szapiro, Ph.D., (editor) the Decision Analysis and Support Unit, Institute of Econometrics at SGH Warsaw School of Economics. Adjunct Professor at Carlson School of Business at University of Minnesota. His current research is focused decision science methods for individual and group processes analysis and support. His gained business experience in the Supervisory Board of ING Bank Śląski and currently –- in the Supervisory Board of Aviva Investors Poland (he is the Head of Audit and Risk Committee). He authors numerous research articles and participated in international scientific projects. Recently he is actively involved in the National Council for Science and Higher Education, the Council of National Science Center and the Board of Conference of Rectors of Academic Schools in Poland.

Małgorzata Wrzosek, Ph.D., is an assistant professor at the Decision Analysis and Support Unit, Institute of Econometrics at the SGH Warsaw School of Economics. In her work, she uses data analysis as well as mathematical, statistical and data mining methods to support decision-making processes. She was professionally involved in building analytical models for companies from the financial and telecommunications sectors. She also has several years of experience in building and validating credit risk models.

Sebastian Zając, Ph.D., is theoretical physicist and assistant professor at the Statistical Methods & Business Analytics Unit at SGH Warsaw School of Economics. The focus of his research are Real-Time analytics, Quantum computing for machine learning, cloud computing and mathematical aspects of modeling in machine and deep learning.

CREDIT SCORING

IN CONTEXT OF INTERPRETABLE MACHINE LEARNING

THEORY AND PRACTICE

Daniel Kaszyński

is an assistant at the Decision Analysis and Support Unit, Institute of Econometrics at the Warsaw School of Economics. In his research and professional work, Daniel gets involved in the construction and validation of credit risk models, numerical optimization issues, as well as technological consulting. Daniel has over eight years of consulting experience in the area of financial risk and implementation of IT solutions.

Bogumił Kamiński

is the head of the Decision Analysis and Support Unit, Institute of Econometrics at the Warsaw School of Economics and a professor at the Data Science Laboratory, Ryerson University, Canada. He specializes in the use of advanced methods of data analysis to support decision-making in enterprises. Bogumił is the author of over one hundred articles on the business applications of forecasting, optimization, and simulation methods. He actively promotes the use of the latest research results in economic practice. Kamiński has over twenty years of experience in implementing IT solutions in the largest Polish companies and institutions.

Tomasz Szapiro

the Decision Analysis and Support Unit, Institute of Econometrics at the Warsaw School of Economics. Adjunct Professor at Carlson School of Business at the University of Minnesota. His current research is focused on decision science methods for individual and group processes analysis and support. He gained business experience in the Supervisory Board of ING Bank Śląski and currently – on the Supervisory Board of Aviva Investors Poland (he is the Head of Audit and Risk Committee). He authored numerous research articles and participated in international scientific projetcs. Recently, he is actively involved in the ational Council for Science and Higher Education, the Council of National Science Center and the Board of Conference of Rectors od Academic Schools in Poland.

ISBN 978-83-8030-424-6

9 788380 304246 >

OFICYNA WYDAWNICZA SGH
SZKOŁA GŁÓWNA HANDLOWA W WARSZAWIE
www.wydawnictwo.sgh.waw.pl