GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Interpretable Binary and Multiclass Prediction Models for Insolvencies and Credit Ratings

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität Göttingen

im Promotionsprogramm Computer Science (PCS)
der Georg-August University School of Science (GAUSS)

vorgelegt von
Lennart Obermann
aus Göttingen

Göttingen
im April 2016

Betreuungsausschuss

Erster Betreuer	Prof. Dr. Stephan Waack, Institut für Informatik, Georg-August Universität Göttingen.
Zweiter Betreuer	Prof. Dr. Carsten Damm, Institut für Informatik, Georg-August Universität Göttingen.

Prüfungskommission

Referent:	Prof. Dr. Stephan Waack, Institut für Informatik, Georg-August Universität Göttingen.
Korreferent:	Prof. Dr. Carsten Damm, Institut für Informatik, Georg-August Universität Göttingen.
Weitere Mitglieder der Prüfungskommission:	Prof. Dr. Matthias Schumann, Instituts für Wirtschaftsinformatik, Georg-August Universität Göttingen.
	Prof. Dr. Burkhard Morgenstern, Institut für Mikrobiologie und Genetik, Georg-August Universität Göttingen.
	Prof. Dr. Jens Grabowski, Institut für Informatik, Georg-August Universität Göttingen.
	Prof. Dr. May, Institut für Informatik, Georg-August Universität Göttingen.
Tag der mündlichen Prüfung:	10. Mai 2016

Abstract

Insolvency prediction and credit rating are challenging tasks used to evaluate the creditworthiness of commercial enterprises based on qualitative and quantitative attributes. One way to approach these tasks is machine learning whereby prediction models are built from sample data. The advantage of machine learning is the automatization of the process obviating the need for human knowledge in most cases and thus, its high level of objectivity. Nevertheless, this approach does not claim to be perfect which is why it does not completely replace human knowledge. Since these models can be used as decision support for experts, interpretable models are desirable. Unfortunately, interpretable models are provided by only a few machine learners. Furthermore, some tasks in finance like credit rating often are multiclass problems. Multiclass classification is often achieved via meta-algorithms using multiple binary learners. However, most state-of-the-art meta-algorithms destroy the interpretability of binary models.

In this thesis, we study the performance of interpretable models compared to non-interpretable models in insolvency prediction and credit rating. We look at disjunctive normal forms and decision trees of thresholds of financial ratios as interpretable models. We use random forests, artificial neural networks, and support vector machines as non-interpretable models. Furthermore, we use our own developed machine learning algorithm Thresholder to build disjunctive normal forms and interpretable multiclass models.

For the task of insolvency prediction, we demonstrate that interpretable models are not inferior to non-interpretable black-box models. In a first case study, a real-life database with financial statements of 5152 enterprises is used to evaluate the performance for all models.

In a second case study focused on credit rating, we show that interpretable multiclass models are even superior to non-interpretable multiclass models. We evaluate their performances on three real-life data sets divided into three rating classes.

In these case studies, we compare different interpretable approaches concerning their model size and type of interpretability. We provide example models built on these real-life databases and an interpretation for them.

The results show that interpretable threshold-based models are appropriate for classification problems in finance. For these tasks they are not inferior to more sophisticated models like support vector machines. Our algorithm Thresholder builds the smallest models while its performance is comparable to the other interpretable models.

In our case study on credit rating, interpretable models perform better than for our case study on insolvency prediction. A possible explanation can be found in the nature of credit rating. In contrast to insolvencies, credit ratings are man-made. This implies that credit ratings are based on decisions by people thinking in interpretable rules, e.g., logical operations on thresholds. Thus, we assume that interpretable models match the problems and detect and represent these interpretable rules.

Zusammenfassung

Insolvenzprognosen und Ratings sind wichtige Aufgaben der Finanzbranche und dienen der Kreditwürdigkeitsprüfung von Unternehmen. Eine Möglichkeit dieses Aufgabenfeld anzugehen, ist maschinelles Lernen. Dabei werden Vorhersagemodelle aufgrund von Beispieldaten aufgestellt. Methoden aus diesem Bereich sind aufgrund Ihrer Automatisierbarkeit vorteilhaft. Dies macht menschliche Expertise in den meisten Fällen überflüssig und bietet dadurch einen höheren Grad an Objektivität. Allerdings sind auch diese Ansätze nicht perfekt und können deshalb menschliche Expertise nicht gänzlich ersetzen. Sie bieten sich aber als Entscheidungshilfen an und können als solche von Experten genutzt werden, weshalb interpretierbare Modelle wünschenswert sind. Leider bieten nur wenige Lernalgorithmen interpretierbare Modelle. Darüber hinaus sind einige Aufgaben wie z.B. Rating häufig Mehrklassenprobleme. Mehrklassenklassifikationen werden häufig durch Meta-Algorithmen erreicht, welche mehrere binäre Algorithmen trainieren. Die meisten der üblicherweise verwendeten Meta-Algorithmen eliminieren jedoch eine gegebenenfalls vorhandene Interpretierbarkeit.

In dieser Dissertation untersuchen wir die Vorhersagegenauigkeit von interpretierbaren Modellen im Vergleich zu nicht interpretierbaren Modellen für Insolvenzprognosen und Ratings. Wir verwenden disjunktive Normalformen und Entscheidungsbäume mit Schwellwerten von Finanzkennzahlen als interpretierbare Modelle. Als nicht interpretierbare Modelle werden Random Forests, künstliche Neuronale Netze und Support Vector Machines verwendet. Darüber hinaus haben wir einen eigenen Lernalgorithmus Thresholder entwickelt, welcher disjunktive Normalformen und interpretierbare Mehrklassenmodelle generiert.

Für die Aufgabe der Insolvenzprognose zeigen wir, dass interpretierbare Modelle den nicht interpretierbaren Modellen nicht unterlegen sind. Dazu wird in einer ersten Fallstudie eine in der Praxis verwendete Datenbank mit Jahresabschlüssen von 5152 Unternehmen verwendet, um die Vorhersagegenauigkeit aller oben genannter Modelle zu messen.

In einer zweiten Fallstudie zur Vorhersage von Ratings demonstrieren wir, dass interpretierbare Modelle den nicht interpretierbaren Modellen sogar überlegen sind. Die Vorhersagegenauigkeit aller Modelle wird anhand von drei in der Praxis verwendeten Datensätzen bestimmt, welche jeweils drei Ratingklassen aufweisen.

In den Fallstudien vergleichen wir verschiedene interpretierbare Ansätze bezüglich deren Modellgrößen und der Form der Interpretierbarkeit. Wir präsentieren exemplarische Modelle, welche auf den entsprechenden Datensätzen basieren und bieten dafür Interpretationsansätze an.

Unsere Ergebnisse zeigen, dass interpretierbare, schwellwertbasierte Modelle den Klassifikationsproblemen in der Finanzbranche angemessen sind. In diesem Bereich sind sie komplexeren Modellen, wie z.B. den Support Vector Machines, nicht unterlegen. Unser Al-

gorithmus Thresholder erzeugt die kleinsten Modelle während seine Vorhersagegenauigkeit vergleichbar mit den anderen interpretierbaren Modellen bleibt.

In unserer Fallstudie zu Rating liefern die interpretierbaren Modelle deutlich bessere Ergebnisse als bei der zur Insolvenzprognose (s. o.). Eine mögliche Erklärung dieser Ergebnisse bietet die Tatsache, dass Ratings im Gegensatz zu Insolvenzen menschengemacht sind. Das bedeutet, dass Ratings auf Entscheidungen von Menschen beruhen, welche in interpretierbaren Regeln, z.B. logischen Verknüpfungen von Schwellwerten, denken. Daher gehen wir davon aus, dass interpretierbare Modelle zu den Problemstellungen passen und diese interpretierbaren Regeln erkennen und abbilden.

Acknowledgements

I would like to thank several people who supported me with my thesis. First of all, I want to thank my supervisor Prof. Dr. Stephan Waack who introduced me to the exciting topic of machine learning during my masters program. He made it possible for me to continue my research in a PhD program and has always been available for discussion and good pieces of advice.

Moreover, my thanks goes to my co-advisor Prof. Dr. Carsten Damm for providing advice and discussions about my research topic. We had a nice collaboration on teaching several courses. Furthermore, I want to thank the thesis committee members Prof. Dr. Matthias Schumann, Prof. Dr. Burkhard Morgenstern, Prof. Dr. Jens Grabowski, and Prof. Dr. May for investing their valuable time.

I would like to thank the theoretical computer science and algorithmic methods group, especially Dr. Mehmet Gültas for proof-reading this thesis and having lots of discussions during common lunch breaks and well-spent coffee breaks. In addition, I had fruitful discussions with Dr. Steffen Herbold who provided me with pieces of advice on the machine learning topic. Furthermore, I would like to thank my former and current colleagues at the institute of computer science who provided an enjoyable environment.

I owe many thanks to my patient parents Detlef and Angela Obermann who supported me during my lengthy course of studies.

Last but definitely not least, I want to thank my girlfriend Linda Neumeyer for motivating and supporting me during the process of writing this thesis. Furthermore, she invested her very rare spare time proof-reading this thesis. I am very grateful for this.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ANN Artificial Neural Network

CNF Conjunctive Normal Form

DL Decision List

DNF Disjunctive Normal Form

DT Decision Tree

EBIT Earnings Before Interest and Taxes

FN False Negative

FP False Positive

HGB German commercial code called “Handelsgesetzbuch”

IDK I-Dont-Know (classification label)

IFRS International Financial Reporting Standards

InsO German insolvency statute called “Insolvenzordnung”

MCC Matthews Correlation Coefficient

MDA Multiple Discriminant Analysis

RF Random Forest

RIPPER Repeated Incremental Pruning to Produce Error Reduction

RS Rule Set

RST Rough Set Theory

SD Standard Deviation

SVM Support Vector Machine

TN True Negative

TP True Positive

USGAAP Generally Accepted Accounting Principles in the United States

1 Introduction

Insolvency prediction and credit rating are important tasks in finance. Their purpose is the evaluation of the economic situation of enterprises. Inaccurate predictions may cause huge financial losses. Besides more accurate results, it is desirable to achieve a higher automation of these processes as well. Therefore, the development of computer programs for these tasks is an important research topic in business informatics. A commonly used way is to apply machine learning techniques.

Machine learning is an algorithmic method for deriving models from training data. The aim is to generalize and not to memorize instances in order to evaluate unseen data. Thus, by swapping the learning sample and recomputing the classifier, economic changes or needs of different industry sectors can be captured without modifying program logic. Machine learners are implemented as computer programs. In general, computer programs have the advantage of objective decision-making on the basis of the provided data and are not influenced by human opinions. Despite the fact that machine learning methods using annual accounts offer an automated and rather objective way to achieve high prediction rates for this task, to a certain degree humans may also subjectively contribute to the development of computer programs.

For predictions, an accuracy of 100% can rarely be achieved which results in so called classification errors. Low error rates are tolerable up to a certain limit. Since this limit

strongly depends on its application, it is not possible to provide a precise value. A distinction is made between different error types, e.g., misclassifications on insolvent and solvent enterprises. This distinction is important because a misclassified insolvent enterprise results in a payment default, whereas a misclassified solvent enterprise only results in the loss of a possible client which is less crucial. Furthermore, there are naturally much more solvent than insolvent enterprises which results in so called imbalanced data.

In any case, machine learning models may be incorrect. Therefore, these models cannot completely replace expensive experts. Thus, our goal is to build objective models with a low prediction error as a helpful decision support for experts in credit rating. Therefore, we additionally focus on the interpretability of models. This requires a definition of interpretability in terms of machine learning models. Different positions can be found in the literature. A model is already called interpretable if the importance of features is derivable [62]. In addition, some researchers call models interpretable, if they consist entirely of interpretable rules, no matter how many there are [39]. We think that an interpretable model needs to be interpretable by human beings and therefore consists entirely of interpretable rules, but of a reasonable amount. Furthermore, these rules have to be connected by interpretable operations. The less rules there are in a model the more interpretable it becomes. The rules should have a structure of what a human being would think of: Boolean expressions with threshold indicators. In this work, a threshold indicator is the Boolean indicator function of a threshold of a financial ratio, e.g., "Cash flow $>$ 1.5 mil. €". The indicator returns 1, if the condition displayed is satisfied and 0 otherwise. Decision Trees (DTs), for example, are well-established models of machine learning and considered interpretable in the literature [8, 26, 35, 109]. The models consist of threshold indicators and can be very effectively interpreted. Moreover, taking pattern from Brodag [19], we have adopted and further developed Thresholder which outputs Disjunctive Normal Forms (DNFs) of upper and lower threshold indicators for financial ratios as Boolean variables.

There are binary tasks such as insolvency prediction and multiclass tasks like credit rating. In machine learning, the latter case is often reduced to several binary learning steps. This might change the structure and increase the complexity and size of the models, and therefore may destroy their interpretability. Thus, we further developed Thresholder to output interpretable multiclass models.

1.1. Scope of this Thesis

We want to find out whether it is possible to achieve accurate results which are interpretable as well. Therefore, we compare different interpretable models with frequently used non-interpretable models in two case studies for two different problems in finance. Furthermore, we use our own developed algorithm Thresholder for interpretable multiclass models.

The first case study [85] is about the binary classification problem of insolvency prediction. A big challenge can be seen in the imbalanced data set. Highly imbalanced data is

always a problem for machine learning [20, 93]. To tackle this problem and to benefit from the huge amount of solvent data sets, we use asymmetric bagging [102]. This results in a majority vote over downsampled data sets with a solvent to insolvent ratio of 1:1. Therefore, we are able to train the algorithms on a balanced data set without having to relinquish solvent data. We compare the two models DTs and DNFs, which we call white-box models, with some of the most frequently and recently used non-interpretable models. We denote a method gray-box, if an interpretable form is not directly available, but can be calculated from the model. In most cases, this calculated model is more complex than white-box models or less accurate than the original form. There are some approaches to explain the outcome of Artificial Neural Networks (ANNs) and Random Forests (RFs) which renders them gray-box models. Support Vector Machines (SVMs) are considered black-box models. Details and sources of this distinction can be found in Section 3.2.1.

Our main finding is that the two white-box models are not inferior to the more sophisticated gray and black-box models. We think that logical operations on threshold indicators for financial ratios are appropriate for insolvency prediction.

The motivation of our second case study [86] is to find out whether the results of the first case study can be transferred to the multiclass classification problem of credit rating. Furthermore, we want to show that this does not only work for a multiclass problem, but even better for a problem with man-made classifications, i.e., credit ratings. We compare Thresholder with another DNF and DT algorithm. We compare these three interpretable models again with RFs, ANNs, and SVMs.

This time, the interpretable models even outperform to the more sophisticated models. This can be explained by the nature of the problem. Insolvency is influenced by multiple economic factors. In contrast, credit rating is based on decisions by people thinking in interpretable ways. Thus, we assume that logical operations on threshold indicators for financial ratios are the best choice to reconstruct human decisions in credit rating.

1.2. Thesis Impact

We published the results of this thesis in the following peer-reviewed international journal articles.

- **Lennart Obermann** and Stephan Waack. Demonstrating non-inferiority of easy interpretable methods for insolvency prediction. *Expert Systems with Applications*, 42(23):9117 – 9128, 2015.

- **Lennart Obermann** and Stephan Waack. Interpretable multiclass models for corporate credit rating capable of expressing doubt. Submitted, 2016.

Furthermore, the following master's thesis and student projects were supervised during the work on this thesis.

- Automatisierte Rekonstruktion unvollständiger Datensätze am Beispiel der Insolvenzprognose, Christian Otto, Student project, 2012.
- Verbesserung der Vorhersagegenauigkeit bei Klassifikationsproblemen durch einen Ensemble-Lern-Ansatz mit Boosting am Beispiel der Insolvenzprognose, Master's thesis, Christian Otto, 2012.
- A PAC learning heuristic considering noise for insolvency prediction, Wiebke Tornow, Student project, 2014.

1.3. Thesis Structure

This thesis is mainly based on two case studies [85, 86] published by us. The description in the following chapters is based on these studies. Following the introduction, the theoretical background of this thesis is presented. The thesis continues with a motivation for our work and a description of earlier and recently used algorithms and models for problems in finance. Afterwards, we present our methods, evaluate them in two case studies, and discuss the results. The thesis is summarized in a brief conclusion. In more detail, the chapters are organized as follows.

- **Chapter 2 (Background)** explains the foundations of this thesis in two sections. The section on finance describes the background of the problem statement and the section on machine learning describes the tools used to solve this problem statement.
- **Chapter 3 (Interpretability of Models)** presents the motivation for the use of interpretable models and defines the term "interpretable models" in the context of this thesis. Afterwards, the models used in this work are categorized by this definition. Finally, interpretable model classes are compared.
- **Chapter 4 (Related Work)** describes the latest state of the art in the field of machine learning for financial problems. It starts with the history of machine learning algorithms generally used in finance, presents multiclass approaches, and finally, presents studies focusing on interpretable models.
- **Chapter 5 (Cascaded DNFs as Interpretable Multiclass Models)** introduces Thresholder, a DNF learning algorithm developed in the context of this thesis. At first, the binary version for learning DNFs and the multiclass version for learning cascaded DNFs are presented. Afterwards, we describe a method to express uncertainty of the classifier in an interpretable way and list the benefits of this approach.

- **Chapter 6 (Case Studies)** presents the two case studies this thesis is based on, their results, and a discussion. The first case study is a binary insolvency prediction. Thereby, we show that interpretable models are not inferior to the rest. The second case study is a multiclass credit rating. Here interpretable models perform even better than most of the rest.
- **Chapter 7 (Discussion)** discusses the results of the case studies as a whole and points out strengths and limitations of our approach.
- **Chapter 8 (Conclusion)** summarizes this thesis and its main contributions and provides an outlook on possible future work.

2 Background

This section introduces the foundations of this thesis. They are split into two sections. In the section on finance we explain important terms to understand the problem statement. The theory and the tools used to approach the problem are explained in the section on machine learning.

2.1. Finance

Since our case studies are based on German data, all concepts in this section are described according to the generally accepted accounting principles in Germany, the German commer-

cial code called "Handelsgesetzbuch" (HGB). Note that there are international differences concerning accounting principles, e.g., Generally Accepted Accounting Principles in the United States (USGAAP) or International Financial Reporting Standards (IFRS).

In this section, we describe the data obtained from enterprises (financial ratios, financial statements), solvency classifications of an enterprise (insolvency, credit rating) and the prediction process of these classifications (default prediction). Sources are the HGB[1] and the German insolvency statute called "Insolvenzordnung" (InsO)[2].

2.1.1. Financial Statements

Financial statements are annual reports of a business's financial position. They consist of balance sheets and income statements which are explained in the following sections. In Germany, each businessman or business has to create an annual financial statement. An exception are businesses with small revenue and annual net income. The main purposes of these reports are informing owners, managers, shareholders, and creditors about the current status as well as the assessment of tax and dividend. Financial statements have to be published by corporations (Kapitalgesellschaften), business partnerships (Personenhandelsgesellschaften), and businesses exceeding a certain size. There are some easements for micro, small, and medium-sized enterprises concerning accountability as well as disclosure. Some items can be excluded and some can be combined.

2.1.1.1. Balance Sheets

Business balance sheets list capital usage and capital origin at the end of a fiscal year. Capital usage mainly consists of assets. In sum the total assets are listed on the left side of the balance sheet. Capital origin mainly consists of equity and liabilities. Equity and liabilities in total are listed on the right side. Since these are two different views on the same capital, the sums of both sides are always identical. Table 2.1 shows an example of an shortened balance sheet.

2.1.1.2. Income Statements

Income statements list the business's revenues and expenses during a particular period, commonly a fiscal year. It distinguishes different kinds and sources of revenues and expenses. Sales revenue is the starting value and all others are added up in a particular order with certain interim results. Table 2.2 shows items of an income statement.

[1] Handelsgesetzbuch in der im Bundesgesetzblatt Teil III, Gliederungsnummer 4100-1, veröffentlichten bereinigten Fassung, das durch Artikel 1 des Gesetzes vom 28. Juli 2015 (BGBl. I S. 1400) geändert worden ist

[2] Insolvenzordnung vom 5. Oktober 1994 (BGBl. I S. 2866), die zuletzt durch Artikel 16 des Gesetzes vom 20. November 2015 (BGBl. I S. 2010) geändert worden ist

Assets (capital usage)	Equity and liabilities (capital origin)
A. Non-current assets (fixed assets)	**A**. Equity
Intangible assets	Capital stock
Fixed assets	Capital reserves
Financial assets	Revenue reserves
B. Current assets	Retained earnings
Inventories	Unappropriated net income
Receivables and other assets	Annual net income
Securities	**B**. Provisions
Cash and cash equivalents	**C**. Liabilities
C. Prepaid expenses	Current liabilities
	long-term liabilities
	Liabilities to banks
	Advance payments received on orders
	Liabilities to affiliated companies
	D. Deferred income
Total assets	Total equity and liabilities

Table 2.1.: Example of a shortened balance sheet according to HGB §266.

2.1.2. Financial Ratios

Financial ratios are numbers to evaluate enterprises. Their values are determined by formulas whose parameters are mostly taken from balance sheets and income statements. Often, several items merge into one financial ratio. Thus, financial ratios offer a more compact overview on balance sheets and income statements. Financial ratios are either absolute, mostly monetary, values in a certain currency or relative values between two or more financial ratios. They are used for comparisons of past values or other companies to determine a company's performance. Many financial ratios are correlated to each other to a certain degree. Therefore, one financial ratio might be good, despite the fact that another one is bad. For example, an enterprise might have a comparatively big number of machines (high amount of fixed assets). However, those might be bought on credit (high amount of liabilities). This example shows why it is not possible to evaluate enterprises on just one (or few) financial ratios. Another reason for using multiple financial ratios are the differences of importance and value between industries. For example, fixed assets are less important for the evaluation of banks rather than for construction businesses. In turn, a more precise outline of liabilities is an important factor for evaluating banks.

Financial ratios are grouped into different categories describing different financial aspects. The ones that are used in this work are briefly described below. Most calculation rules could directly derived from our database [110]. The rest can be found in other literature [19, 107].

Sales revenue
Changes in inventories and other own capitalized costs
Other own work capitalized
Other operating income
Material expenses
Cost of consumables and supplies and of goods purchased and held for resale
Cost of purchased services
= Total operating performance
Staff costs
Wages, salaries and emoluments
Social security contributions, retirement benefit expenses and assistance benefits
Amortisation of intangible assets and depreciation of property, plant and equipment
Other operating expenses
= Results from ordinary business activities
Income from investments
Income from other securities and from loans held as financial assets
Interest income
Depreciation of financial assets and of securities held as current assets
Interest expenses
Income tax expense
= Annual net income
Unappropriated net income carried forward from previous year
Retained earnings
Capital surplus
= Unappropriated net income

Table 2.2.: Income statement according to the total expenditure format in HGB §275.

2.1.2.1. Profitability Ratios

- Return on investment measures the efficiency of an investment by comparing the income with the capital invested. It is defined as

$$\text{return on investment} = \frac{\text{net income}}{\text{total assets}}.$$

- Return on equity measures the interest rate from shareholders equity, by comparing the income with the equity and is defined as

$$\text{return on equity} = \frac{\text{net income}}{\text{equity}}.$$

2.1.2.2. Operating Ratios

- Sales revenue is the business's income from its normal business activities, commonly consisting of selling goods and services. It is directly taken from the enterprise's income statement.

- (Annual) net income is a result of the income statement and reflects the profit after taxation. The definition is

$$\text{(annual) net income} = \text{total revenues} - \text{total expenses}.$$

- Profit margin is profit as a percentage of sales revenue

$$\text{profit margin} = \frac{\text{net income}}{\text{sales revenue}}.$$

- Cash flow measures the amount of cash flowing in and out of an enterprise. There are different definitions in the literature [79], e.g.,

$$\text{cash flow} = \text{revenue} + \text{non-cash expenses} - \text{non-cash revenue}$$

and

$$\text{cash flow} = \text{net income} + \text{depreciation} + \text{amortization} + \text{depletion}.$$

- Earnings Before Interest and Taxes (EBIT) is profit without interest and taxes

$$\text{EBIT} = \text{annual net income} + \text{income taxes} - \text{financial income} + \text{financial expenses}.$$

- Cash flow is set in proportion to current liabilities to determine the amount of liabilities which can be defrayed by income within a year

$$\text{cash flow / current liabilities} = \frac{\text{cash flow}}{\text{current liabilities}}.$$

2.1.2.3. Liquidity Ratios

- Current ratio measures the business's ability to pay debts within a year by comparing current assets to current liabilities. It is defined as

$$\text{current ratio} = \frac{\text{current assets}}{\text{current liabilities}}.$$

- Working capital is similar to current ratio. It measures the remaining capital when current liabilities are paid with current assets and is defined as

$$\text{working capital} = \text{current assets} - \text{current liabilities}.$$

2.1.2.4. Capital Structure Ratios

- Equity ratio is the amount of equity compared to the total capital

$$\text{equity ratio} = \frac{\text{equity}}{\text{total assets}}.$$

- Dynamic debt-to-equity ratio measures the duration of repayment of liabilities by dividing liabilities by cash flow

$$\text{dynamic debt-to-equity ratio} = \frac{\text{total liabilities}}{\text{cash flow}}.$$

- Working capital can be set in proportion to total assets to get a relative value instead of an absolute one

$$\text{working capital / total assets} = \frac{\text{working capital}}{\text{total assets}}.$$

- Debt-to-capital ratio is the amount of debt compared to total capital and is defined as

$$\text{debt-to-capital ratio} = \frac{\text{total liabilities}}{\text{total assets}}.$$

- Current maturity (Kurzfristige Fristenkongruenz) is an undocumented financial ratio in the DAFNE database [111] which seems to be based on the ratio of equity and fixed assets.

2.1.3. Insolvency

Insolvency means that debtors cannot pay their debts to their creditors. To satisfy both parties, insolvencies are treated according to the insolvency proceedings process defined in InsO. Insolvency proceedings are opened either by a creditor or a debtor. The purpose is either to liquidate the debtors assets and distribute the earnings which leads to bankruptcy, or to reach an arrangement for an insolvency plan to keep the enterprise running. For the latter case, creditors give up a part of their redemption and rely on a complete refund of the rest in the near future.

This work only focuses on corporate insolvency. Reasons for insolvency can be misplanning or mismanagement. However, external factors can strike a healthy enterprise as well, e.g., strong competitors, a changing market, bad press, or other unexpected causes. According to the Federal Statistical Office of Germany [98] the number of annual corporate insolvencies in Germany is about 20,000 to 30,000 for the last few years. The absolute and relative number of insolvencies per industry varies a lot. Construction and trade, for example, have had the most insolvencies in the last few years. The total corporate insolvency claims for 2014 were about 25 billion €.

2.1.4. Credit Rating

Creditworthiness of a debtor measures its solvency. Often, this is measured by the probability of the debtors to pay back their debts. These probabilities are assigned to an ordinal set of rating classes. The process and the outcome of this evaluation are called credit rating.

Measuring solvency of a client is very important for creditors, since insolvency results in payment defaults and may cause huge financial losses for the creditor. Therefore, creditors use credit ratings to evaluate the solvency of a client. Depending on the outcome, the credit is either granted or denied. Furthermore, in some countries like in the US and the UK, it is required to provide reasons for a declined credit [29]. The amount of interest may depend on the credit rating of the client as well.

Credit ratings can be applied to customers, enterprises, or national governments. This work focuses on corporate credit rating which applies to enterprises only. Credit ratings can be performed for internal purposes, e.g., banks rating their customers, or external purposes, e.g., performed by credit rating agencies which sell these information to their clients. The most important international rating agencies are Standard & Poor's, Moody's, and Fitch which cover most of the market. There are also smaller national rating agencies like the German Creditreform. This is where we obtained our data sets from. Rating classes are not standardized and may differ from agency to agency. However, the labels are similar and they differ only in small details. There are long term ratings, short term ratings, and so called traffic light reports. The latter two try to simplify the original rating by combining several classes. Table 2.3 compares the rating scales of Standard & Poor's to the ones of Creditreform which we use in our studies.

Ratings are basically based on quantitative and qualitative factors. Quantitative factors are expressed by financial ratios which can be obtained and calculated from annual accounts. Their acquisition and evaluation can be partly automated. Qualitative factors are naturally not expressed in numbers. Examples are quality of management, business strategy, organizational structure, payment experiences from creditors, or competitiveness of the enterprise. These factors are tediously collected by hand and evaluated by analysts.

Standard & Poor's		Creditreform			Meaning
Long Rating	Short Rating	Long Rating	Short Rating	Traffic light Report	
AAA		AAA	AAA		Best solvency, lowest risk of insolvency.
AA+ AA AA-	A-1+	AA+ AA AA-	AA		Very good solvency, very low risk of insolvency.
A+ A	A-1	A+ A	A	Good	Good solvency, low risk of insolvency.
A-	A-2	A-			
BBB+		BBB+			Strongly satisfying solvency, low to medium risk of insolvency.
BBB BBB-	A-3	BBB BBB-	BBB		
BB+ BB BB-	B	BB+ BB BB-	BB		Satisfying solvency, medium risk of insolvency.
B+ B B-		B+ B B-	B	Medium	Sufficient solvency, higher risk of insolvency.
CCC+ CCC CCC- CC C	C	CCC CC C	C	Bad	Barely sufficient solvency, high to very high risk of insolvency.
D	/	D	D		Insufficient solvency, insovency.

Table 2.3.: Standard & Poor's ratings compared to Creditreform ratings.

2.1.5. Default Prediction

Predicting insolvency, bankruptcy, business failure, or financial distress as well as credit rating are important tasks with a similar aim in finance. They are all used to predict the default risk of a debtor at different levels. All of the above tasks are binary classification problems except for credit rating which can be a multiclass problem as well. From the machine learning point of view, these are harder to handle than binary classification problems. Another difference is the origin of the classification labels. Credit ratings are man-made. Business failure and financial distress classifications depend on the definition of failure and distress. Insolvencies and bankruptcies are influenced by multiple economic factors and are more likely to affect enterprises in financial distress.

Default predicting is an important task since default cause huge financial losses. How-

ever, these predictions are not perfect and may contain errors. It is important to distinguish between different types of errors, e.g., errors on solvent and insolvent enterprises. The former results in denying a credit to a healthy enterprise and the loss of a possible client. The latter results in granting a credit to an unhealthy enterprise which might result in a payment default.

These predictions are mainly used by banks, but also by shareholders, suppliers, and auditors. Rating agencies and credit bureaus provide these predictions. They collect data from published financial statements and use financial ratios and qualitative data to calculate their prediction models. These prediction models are based on expert knowledge, statistical models, or sophisticated artificial intelligence models. These models can be interpretable or black-box models. Research on this topic is important because there is always room for improvement of the prediction rate.

2.2. Machine Learning

Machine learning is the derivation of generalized hypotheses from data to predict properties in yet unseen data. This means for the learned models to describe the data in general and not to memorize specific samples. The aim is to predict unseen data using sample data. Machine learning has been successfully applied to numerous fields (e.g., DNA-Sequencing in bioinformatics [73], image recognition systems [96], and evaluating enterprises (see Chapter 4)). A general definition of machine learning is given by Mitchell [80, p. 2]:

> "A computer programm is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

In this thesis, the task T is the evaluation of enterprises using financial ratios. The performance P is the occurring error and the experience E is derived from the financial statements of the enterprises. Thus, this definition states that the error of an evaluation can be lowered by increasing the amount of data.

Typically, machine learning is classified into three main categories [32].

- In supervised learning, the data consists of pairs of a feature vector and a label. An algorithm trains a function on labeled data, which calculates the label for unlabeled data. Examples are insolvency prediction or handwriting recognition.

- In unsupervised learning or clustering, the data only consists of features without labels. Algorithms have to find a structure on their own and divide the data into different classes. Examples are identifying communities of people in social networks or partitioning customers into market segments.

- In reinforced learning, the algorithm learns to achieve a certain goal in a dynamic environment by getting rewards and punishments. Examples are playing a game against an opponent or driving a car.

This work is solely about supervised learning. Labels are determined by insolvency status or credit rating class. The data set is randomly split into a training and test data set. The training data set is used to build the models and the test data set is used to evaluate their performance.

In the following, we explain the theoretical principles of machine learning. We describe models, algorithms, and performance measures used in this thesis for binary and multiclass learning. The introduction of machine learning is based on several textbooks [4, 13, 32, 53, 65].

2.2.1. Definitions

This section describes the theoretical principles of machine learning necessary to understand the following sections of this chapter.

2.2.1.1. Learning Universe

The learning universe $\mathfrak{U}$ consists of an n-dimensional input space $\mathfrak{X} \subset \mathbb{R}^n$ which describes all possible input data and an output space $\mathfrak{Y} \subset \mathbb{N}_0$, $|\mathfrak{Y}| = l$ with all possible labels and, optionally a label for uncertainty [4] which we call IDK-classification. In the latter case, the label value l represents IDK-assignments. The learning universe is defined as the Cartesian product of input and output space $\mathfrak{U} := \mathfrak{X} \times \mathfrak{Y}$.

2.2.1.2. Training Data

The training data is a subset of the learning universe. It should be an independent and identically distributed random sample

$$\mathbf{U} := \big(U_1, U_2, ...U_s\big) = \Big(\big(\mathbf{X}_1, Y_1\big), \big(\mathbf{X}_2, Y_2\big), ...\big(\mathbf{X}_s, Y_s\big)\Big)$$

with s instances $U = (\mathbf{X}, Y) \in \mathfrak{U}$ consisting of a feature vector $\mathbf{X} \in \mathfrak{X}$ and a label $Y \in \mathfrak{Y}$. Note that IDK-classifications do not appear in the training data and therefore $P(Y = l) = 0$.

2.2.1.3. Concept Class and Hypothesis Class

The concept class $\mathcal{C}$ and the hypothesis class (or model class) $\mathcal{H}$, both describe sets of functions, which map the input space to the output space $h : \mathfrak{X} \to \mathfrak{Y}$. Each function provides a classification rule for the instances. It is the task of a learning algorithm A, using the training data, to select a hypothesis $h = A(\mathbf{U}) \in \mathcal{H}$, which approximates Y as accurate as possible.

2.2.1.4. Noise

Unfortunately, predicting the observed label Y is not the ultimate goal. In theory, there is an unknown function called target concept which calculates the true label of an instance $f(\mathbf{X})$, e.g., its solvency status or credit rating. However, in practical applications, the target concept can not necessarily be found, because of insufficiencies of the data. These can be wrong or missing features or wrong labels. This circumstance is modeled by noise. To match reality, it is assumed that the target concept is influenced by random noise ε which may invert the classification observed $Y = f(\mathbf{X}) + \varepsilon$. It would be desirable to approximate f instead of Y. However, since both f and the parameters of ε are unknown, it is only possible to approximate Y.

2.2.1.5. Loss Function

The degree of approximation of h to Y is measured by a loss function. A loss function $\lambda(c_i, c_j)$ assigns a numerical loss value for the true label c_i and the predicted label c_j. If there is no misclassification, the function returns 0. For a misclassification, the returned value depends on the function. Probably, the most common loss function is the $0/1$ loss function

$$\lambda_{0/1}(c_i, c_j) = \begin{cases} 0 & \text{if } c_i = c_j \\ 1 & \text{otherwise} \end{cases}.$$

An example for another loss function is the quadratic loss function

$$\lambda^{(2)}(c_i, c_j) = (c_i - c_j)^2.$$

Sometimes it is desirable for the decision of the classifier to express doubt. In case of doubt, the classifier assigns a special label to this instance. Later, these instances can receive a special treatment, e.g., classification by hand. We call this additional label IDK-classification. A $0/1$ loss function utilizing this IDK-classification uses a value τ to penalize assignments of this class

$$\lambda_\tau(c_i, c_j) = \begin{cases} 0 & \text{if } c_i = c_j \\ \tau & \text{if } c_j = l \\ 1 & \text{otherwise} \end{cases}.$$

Choosing $\tau = 0$, an IDK-classification is always assigned. Choosing $\tau = 1$, an IDK-classification is never assigned. Therefore the parameter should be chosen that $0 < \tau < 1$. A sensible value of τ is the probability of guessing the wrong label.

2.2.1.6. Classification Error

With a loss function, the classification error of a set of instances can be calculated. Using $\lambda_{0/1}$ as loss function, the error of a hypothesis h is the probability of a wrong classification

of a randomly drawn instance U using its feature vector $\mathbf{X}$, i.e., the deviation of h from label Y:

$$\operatorname{err} h := \mathbb{P}\big(h(\mathbf{X}) \neq Y\big).$$

The classification error for two classes can be divided into α-error and β-error:

$$\operatorname{err}_\alpha h := \mathbb{P}\big(h(\mathbf{X}) = 1 \mid Y = 0\big),$$

$$\operatorname{err}_\beta h := \mathbb{P}\big(h(\mathbf{X}) = 0 \mid Y = 1\big).$$

In practical applications, it is impossible to calculate the true error of a hypothesis, because the distribution of the sample is unknown and there is only a finite amount of samples. Instead, the empirical error or empirical risk can be calculated, which is the error observed in a certain sample, e.g., for $\lambda_{0/1}$

$$\operatorname{err}_{\text{emp}} h = \frac{1}{s}\sum_{i=1}^{s} \lambda_{0/1}\big(h(\mathbf{X}_i), Y_i\big).$$

Thus, error values provided by case studies are always empirical error values. It is obvious that larger training samples usually reduce classification error values, because they contain more empirical information to approximate the true distribution.

2.2.1.7. Bias-Variance Trade-off

The classification error can further be investigated. Let Z be an estimator of constant $z \in \mathbb{R}$. It can be shown [4] that

$$\mathbb{E}(Z - z)^2 = \mathbb{E}\big((Z - \mathbb{E}Z)^2\big) + (\mathbb{E}Z - z)^2. \tag{2.2.1}$$

A fixed instantiation of $\mathbf{U}$ leads to a constant hypothesis h. This trick allows for the usage of Equation 2.2.1 to divide the expected square error of h into two parts

$$\mathbb{E}\big((Y - h(\mathbf{X}))^2 \mid \mathbf{X}\big) = \underbrace{\mathbb{E}\big((Y - \mathbb{E}(Y|\mathbf{X})^2 \mid \mathbf{X}\big)}_{noise} + \underbrace{(\mathbb{E}(Y|\mathbf{X}) - h(\mathbf{X}))^2}_{squared\ error}. \tag{2.2.2}$$

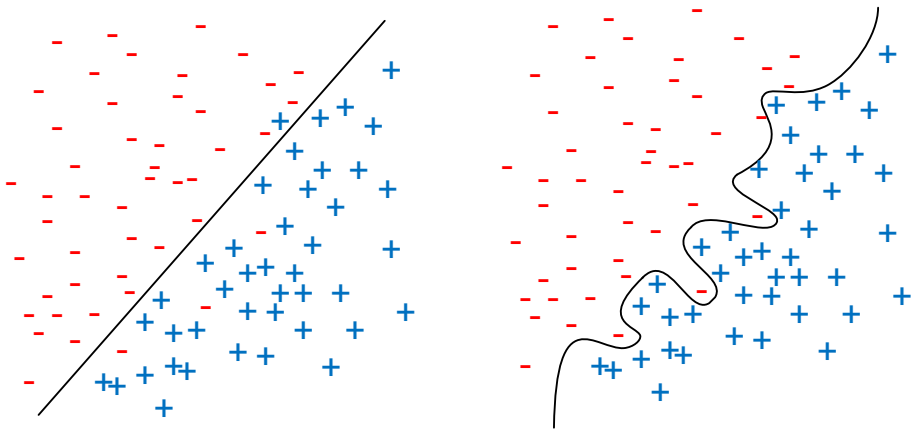
It can be seen that the noise part of Equation 2.2.2 does not depend on h and therefore not on $\mathbf{U}$. Thus, there is no potential to reduce it. The bias part depends on the deviation between Y and h and therefore on $\mathbf{U}$. For some data h may be a good approximation to Y and for some it may be a bad one.

However, averaging over all possible data sets $\mathbf{U}$ of size s and using the same trick from

above leads to Equation 2.2.1.7

$$\mathbb{E}_{\mathbf{U}}\big((h(\mathbf{X})-\mathbb{E}(Y|\mathbf{X}))^2 \mid \mathbf{X}\big) = \underbrace{\mathbb{E}_{\mathbf{U}}\big((h(\mathbf{X})-\mathbb{E}_{\mathbf{U}}(h(\mathbf{X})))^2\big)}_{variance} + \underbrace{(\mathbb{E}_{\mathbf{U}}(h(\mathbf{X}))-\mathbb{E}(Y|\mathbf{X}))^2}_{bias}.$$

Now, these two parts of the formula are called bias and variance. Instead of the squared error, bias measures the deviation to Y regardless of varying samples. Variance measures how much the classifier fluctuates for varying samples. For example, the very simple classifier $h = 1$ would have no variance and a big bias. In contrast, a complicated classifier that perfectly fits the data would have no bias but a big variance because a change in the data would strongly change the classifier as well. These two phenomena are called underfitting and overfitting and are displayed in Figure 2.1. It is challenging to find the best trade-off between bias and variance. Typically, a training sample is used for building the classifier and a separate testing sample is used for evaluating the performance. If the training error is much lower than the test error, there is strong evidence for overfitting. A possible solution is changing the generalization parameters of the algorithm which control the complexity of the model.



(a) Underfitting

(b) Overfitting

Figure 2.1.: Examples for underfitting and overfitting.

2.2.1.8. Cross-validation

Typically, machine learning models are trained on a training set and tested on a test set. These subsamples are created by randomly splitting the original data set. Common proportions are using 0.67% of the original data as training set and the remaining 0.33% as test set [50].

However, if the original data set is small, the training set might be too small to build a model. This results in a big variance. Furthermore, the test set might be too small to obtain a reliable performance estimation. A solution for this problem is the statistical model validation technique cross-validation. Using this technique, the size of the training set is typically chosen to be much more than 0.67% of the original data. This enables building models with a decreased variance. In contrast, the size of the test set becomes smaller which renders the evaluation less reliable. Therefore, the evaluation is performed multiple times; in η-fold cross-validation, the data is split in η equal sized partitions. A classifier is evaluated on each partition after it is trained on the remaining data. The performance is calculated by taking the mean of all η evaluations. Typical values of η are 10 or more [50].

2.2.2. Binary Learning

For binary classification problems the label of the data is 0 or 1 (sometimes also named -1 and +1). Binary learning is applicable to many problem statements. Furthermore, it is a simpler special case of multiclass classification and researched in more detail. Many learning algorithms are only capable of binary classification.

2.2.2.1. Models and Learning Algorithms Used

This section describes the binary learning algorithms and the models which are calculated by the former. Binary learning algorithms take a training data set as input and train a classifier. This classifier can output a classification prediction for unknown data. Some algorithms are also capable of returning probability estimates for each class which is useful for multiclass learning (see Section 2.2.3).

Disjunctive Normal Forms, Conjunctive Normal Forms, Rule Sets, and Decision Lists
DNFs are disjunctions of conjunctions of Boolean variables. In contrast, Conjunctive Normal Forms (CNFs) are conjunctions of disjunctions of Boolean variables. DNFs are subsets of Rule Sets (RSs) which are also called Decision Lists (DLs) or decision tables. They consist of a number of ordered rules. If a rule is fulfilled it determines the classification. If this is not the case the decision is passed to the next rule. There is a default classification after the last rule. The rules are most commonly conjunctions of thresholds.

DNFs are special cases of RSs. The rules are always conjunctions and if fulfilled they always yield a positive classification. The default case is always a negative classification. In the literature, DNFs are called rulesets as well and the containing monomials are called rules. This is likewise correct, but less precise as well.

A popular DNF learning algorithm is the Repeated Incremental Pruning to Produce Error Reduction (RIPPER) heuristic. It is an improvement to IREP [44] and was introduced by William Cohen [25]. RIPPER is a greedy heuristic, which grows monomials, prunes them, and then adds them to a DNF. To achieve this, the training set is randomly partitioned into

a growing set and a pruning set. After that, one monomial at a time is calculated, using the growing set. The selection of literals is based on the metric *precision - false discovery rate*. After a monomial is calculated, it is pruned using the pruning set and accuracy as the performance measure. The pruned monomial is added to the DNF. Instances covered by the monomial are deleted. The heuristic stops, if all positive instances are covered or the description length of the DNF is more than a certain parameter larger than the smallest description length of the monomials obtained so far. The DNF is post-processed in an optimization phase, which optimizes the monomials step by step by creating a replacement and a revision of the monomial. The replacement is created by growing and then pruning a new rule, where pruning minimizes the error of the entire DNF. The revision is created the same way, but starts with the original rule instead of an empty rule. A decision is made by the minimum description length (MDL) heuristic [92], whether the original monomial should or should not be exchanged by the replacement or the revision.

Important parameters are the number of folds which are used for pruning and the generalization parameter which determines the minimum number of samples in a rule.

In the context of this thesis, we have designed a DNF learning algorithm as well, which offers some additional features. It is described in Section 5.1.1.

Decision Trees Usually, a DT is a binary tree consisting of nodes with thresholds. The nodes split the input space. The leaves determine the classification.

C4.5 is a widely used algorithm to build DTs developed by Ross Quinlan [90]. It is an improvement of the ID3 algorithm [89]. Both of them split the data according to information gain. After the calculation of a tree, C4.5 uses pruning to remove branches and thus, reducing the size of the model. If properly applied, this reduces the variance and in turn slightly increases the bias.

Important generalization parameters are the confidence factor C which controls the amount of pruning and the minimum number of instances in a leaf M.

Random Forests RFs were developed by Breiman [18]. A good introduction can be found in Hastie et al. [53]. RFs are a combination of DTs and bagging (Section 2.2.2.2) algorithms. The model is a collection of DTs. The output of this model is a majority vote of these trees. By using multiple trees, RFs reduce the variance of the learner.

The RF-algorithm grows I independent DTs on randomly drawn bootstrap samples of the training data. The tree growing process differs from stand-alone DT algorithms; for each node the threshold is chosen only from a subset of features. Each tree has a maximum depth of d. In contrast to the C4.5 DT algorithm there is no pruning because variance already decreases through the usage of multiple trees.

Artificial Neural Networks ANNs are non-linear statistical models. Inspired by biological neural networks in the nervous system of humans and animals, they consist of a number of

connected neurons which can activate each other to generate an output.

There are different types of ANNs. However, in this thesis we use and describe only RBF-Networks which are amongst the most popular ANNs. RBF-Networks [12, 53] consist of an input layer, an output layer, and a hidden layer with radial basis functions (RBF) as activation functions. The output of this network is a function of the input vector $\mathbf{X}$

$$h(\mathbf{X}) = \sum_{j=1}^{B} w_j \phi_j(\mathbf{X}) + w_0$$

with Gaussian basis functions

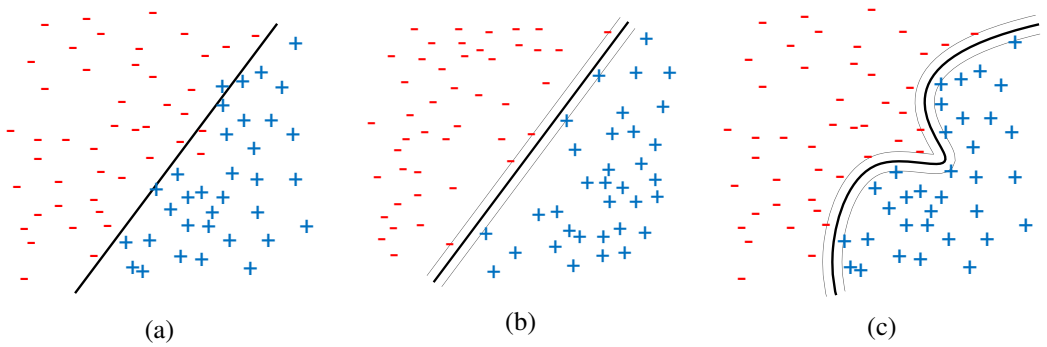
$$\phi_j(\mathbf{X}) = \exp\left(-\frac{||\mathbf{X} - \mu_j||^2}{2\sigma_j^2}\right),$$

the number of RBF-functions B, their weight w_j, a bias w_0, the center of the RBF-functions μ_j, and their width σ_j. There are two important parameters we can use to control the performance of our network, i.e., the number of RBF-functions (nodes in the hidden layer) B and the minimum width W of each RBF-function.

Support Vector Machines The basic SVM [53] is a linear classifier which maximizes the margin between two classes. This margin may be soft and violations can be punished by a cost factor C to control the model complexity. However, most real-life-data cannot be separated by linear classifiers. Nevertheless, the classifier can perform a non-linear classification by using what is called the kernel trick, which maps the input space to a high-dimensional feature space. A linear classifier in this high-dimensional space can be non-linear, if mapped back to the low-dimensional space. The mapping depends on the kernel used. In this thesis, we use

- the linear kernel $K(\mathbf{X}_i, \mathbf{X}_j) = \mathbf{X}_i \cdot \mathbf{X}_j$,
- the polynomial kernel $K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j)^d$, and
- the RBF-Kernel $K(\mathbf{X}_i, \mathbf{X}_j) = \exp\left(-\gamma \cdot ||\mathbf{X}_i - \mathbf{X}_j||^2\right)$.

A linear kernel yields only a linear classifier and does not benefit from the kernel trick. The polynomial kernel parameter d is the degree of the polynomial. The RBF-Kernel γ defines how far the influence of a single training instance reaches. Figure 2.2 shows an example of data which is not linearly separable. Transforming the data with a kernel makes it linearly separable.



(a) (b) (c)

Figure 2.2.: Non-linearly separable data (a) transformed to linearly separable data (b) using a kernel. For the original data, the SVM classifier now looks non-linear (c).

2.2.2.2. Ensemble Learning

Ensemble learners are meta algorithms which use several learning algorithms to build a better classifier. In this work we use bagging which reduces the variance of a learner and boosting which reduces bias.

Bagging Bagging is a meta-algorithm developed by Breiman [17]. A short introduction can be found in Hastie et al. [53, p. 282f]. Bagging increases the stability by reducing the variance and, therefore, decreases the error of a classifier. The algorithm iterates for a number of B bagging iterations. In each iteration it draws a bootstrap sample of the data and trains a classifier. After the last iteration, the final model is the majority vote of all classifiers. Since a mean classifier is built on different data sets, it is less data dependent than a single classifier, thus, reducing variance. Bagging is a generalized form of RFs.

There are different approaches to draw the bootstrap samples. They can be drawn by oversampling, undersampling, with replacement, or without replacement to name but a few [46].

In this work, a special case of undersampling, either on the positive or the negative classified data is used. We call it asymmetric bootstrapping. The bagging method using asymmetric bootstrapping is called asymmetric bagging and was already used by Tao et al. [102].

Boosting Hypothesis boosting is a technique based on the idea by Kearns [64]. It can be used to decrease the bias and the training error of a weak classifier without overfitting and hence, to decrease the prediction error of a learning algorithm. Freund et al. [41] developed the boosting algorithm AdaBoost. A short introduction can be found in Hastie et al. [53, page 337ff]. The algorithm trains a classifier, increases the weight of wrongly classified instances and trains a new classifier, according to the changed weights. This is repeated for B boosting iterations. After the last iteration, the final model is a linear combination of all

classifiers. The linear combination makes simple classifiers more descriptive and therefore might decrease the bias. Obviously, classifiers that are already complex might not profit from this technique.

Algorithm 1 shows this method in detail with weights $w = w_1 \dots w_s$. The labels of the instances are expected to be -1 or 1.

Algorithm 1: AdaBoost

Input : Training sample $\mathbf{U} = (U_1, U_2, \dots, U_s)$ of size s, learning algorithm A
Output: Boosted classifier h

$w_i = \frac{1}{s}, i = 1 \dots s;$
for $b = 1 \dots B$ **do**
  $h_b = A(\mathbf{U}, w);$
  $\text{err}_b = \frac{\sum_{i=1}^{s} w_i \text{err}\, h_b(\mathbf{X}_i)}{\sum_{i=1}^{s} w_i};$
  $\alpha_b = \ln \frac{1-\text{err}_b}{\text{err}_b};$
  $w_i = w_i \cdot e^{\alpha_b \cdot \text{err}\, h_b(\mathbf{X}_i)}, i = 1 \dots s;$
end
$h = \text{sign}\left(\sum_{b=1}^{B} \alpha_b \cdot h_b\right);$

2.2.2.3. Performance Evaluation

The performance of a classifier depends on the amount of correctly classified and misclassified instances of each label. The results are stored in a so called confusion matrix (see Table 2.4) where all actual (P) and predicted (P') positive labels as well as all actual (N) and predicted (N') negative labels are noted. Correctly classified positive instances are called True Positives (TPs) and correctly classified negative instances are called True Negatives (TNs). In return, wrongly classified positive instances are called False Negatives (FNs) and wrongly classified negative instances are called False Positives (FPs).

		Predicted	
		P'	N'
Actual	P	TP	FN (Type II Error)
	N	FP (Type I Error)	TN

Table 2.4.: Confusion matrix.

Since there are two classes which might be of different importance and of different number, it is not possible to put performance in just one number. There are different well

established performance measures using the $0/1$ loss function with values derived from the confusion matrix.

- Accuracy, also known as overall success rate, and error rate both measure the overall success of a classifier. They are defined as

$$\text{accuracy} = \frac{TP+TN}{P+N}$$

and

$$\text{error rate} = \frac{FP+FN}{P+N} = 1 - \text{accuracy}.$$

- α/β-error rate (type I/II error rate) measures the error only for positively/negatively labeled data and are defined as

$$\alpha\text{-error rate} = \frac{FP}{N}$$

and

$$\beta\text{-error rate} = \frac{FN}{P}.$$

- TP/TN rate represents the success only for positively/negatively labeled data. The TP rate is also called sensitivity or recall and the TN rate is called specificity. The definitions are

$$\text{TP rate} = \frac{TP}{P} = 1 - \beta\text{-error rate}$$

and

$$\text{TN rate} = \frac{TN}{N} = 1 - \alpha\text{-error rate}.$$

- Positive/negative prediction value shows the success for positively/negatively classified data classified by the classifier. The positive prediction value is also known as precision. False discovery rate shows the failure for positively classified data. These three performance measures are defined as

$$\text{positive prediction value} = \frac{TP}{P'},$$

$$\text{negative prediction value} = \frac{TN}{N'},$$

and

$$\text{false discovery rate} = \frac{FP}{P'}.$$

- Matthews Correlation Coefficient (MCC) and F-Score are appropriate measures for imbalanced data. They yield a value of 1 in case of no misclassification and a value of 0 in case of trivial models where all instances are classified as the same class. Inverting the classifications yields an inverted MCC. The definitions are

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{P \cdot N \cdot P' \cdot N'}}$$

and

$$\text{F-Score} = 2 \cdot \frac{Precision \cdot recall}{precision + recall}.$$

The confusion matrix for IDK-classifications has an additional column counting the predicted IDK-classifications as seen in Table 2.5.

		Predicted		
		P'	N'	
Actual	P	TP	FN (Type II Error)	IDK
	N	FP (Type I Error)	TN	

Table 2.5.: Confusion matrix for IDK-classifications.

- Using IDK-classifications, accuracy and error rate are calculated as follows. α/β-error rates and TP/TP rates are calculated in a similar way, namely

$$\text{accuracy}_\tau = \frac{TP + TN + (1-\tau) \cdot IDK}{P + N}$$

and

$$\text{error rate}_\tau = \frac{FP + FN + \tau \cdot IDK}{P + N} = 1 - \text{accuracy}_\tau.$$

2.2.3. Multiclass Learning

Multiclass classification problems are characterized by data having more than two different label values. This complicates both the training process and the performance evaluation, because not all binary performance measures are applicable.

2.2.3.1. Learning Algorithms Used

Many learning algorithms naturally support only binary classification, like SVMs. However, there are meta-algorithms which turn binary learning algorithms to multiclass classifiers

by using multiple binary learning algorithms. A short introduction can be found in [13, p. 182ff]. This section describes how learning algorithms achieve multiclass classification and a selection of popular multiclass meta-algorithms used in this thesis.

All-at-once Some classifiers naturally support training multiple classes at once. Tree-based models like DTs and RFs can assign arbitrary label values in their leaves. DLs and RSs can assign arbitrary label values as well when a rule is fulfilled. Since DNFs are Boolean expressions, they naturally support only binary classification. ANNs classify multiple classes by using multiple output nodes with a probability for each label. SVMs cannot handle multiclass learning problems naturally.

One-vs-one This method combines binary classifiers to multiclass classifiers and therefore naturally allows binary classifiers like SVMs to be used for multiclass problems [42]. There are classifiers trained for each pair of labels resulting in $\frac{l(l-1)}{2}$ classifiers where l is the number of labels. For an ordinal multiclass problem, each label c_i is trained against each label $c_j > c_i$. The predicted label is a majority vote of the $\frac{l(l-1)}{2}$ classifiers. In case of a tie, a decision must be made, e.g., by using the smaller class index.

One-vs-rest Like the one-vs-one method, this method trains multiple binary classifiers [68]. There is one classifier trained for each label. Each label $c_i = 1 \dots l$ is trained against the rest. Again, the final decision is a majority vote of all classifiers. If label c_i wins against the rest, it gets a vote.

Compared to one-vs-one, an advantage of this method is the smaller runtime which is linear in the number of labels. On the contrary, the binary training is more expensive, because the training set consists of the whole data set. A disadvantage is the imbalance between the data of label c_i and the data of the merged rest in the training set where in general the latter is probably much bigger. Furthermore, it is problematic that for this method a tie is much more likely than for the one-vs-one method, because a label can only get either one or no vote. This problem can be solved by using probability estimates for each class as a vote.

One-vs-next One-vs-next and one-vs-followers (explained below) are both methods for ordinal multiclass problems. Originally, Kwon et al. [70] proposed this method for ANNs. Later, this method was adopted for other methods like SVMs [67].

The idea is to train $l-1$ classifiers to differentiate between label c_i and c_{i-1} for $c_i = l \dots 1$. If the first classifier decides for the higher class, then this will be the final classification. However, if the classifier decides for the lower class, the next classifier will be evaluated. This is repeated until either the higher classification is chosen or the last classifier is evaluated with a final decision. Advantages of this method are fewer classifiers and balanced data

sets. Furthermore, this method preserves interpretability of interpretable binary classifiers by simply cascading them.

One-vs-followers Like the one-vs-next approach, this method works on ordinal multiclass problems. The difference is what label c_i is compared to. It is not only compared to c_{i-1}, but to all labels $c_j = i-1 \ldots 0$. This leads to the same amount of classifiers. Similarly to the one-vs-rest approach it leads to imbalanced training data sets. This method also maintains interpretability since it uses the same hypothesis class as the one-vs-next method.

2.2.3.2. Performance Evaluation

In multiclass learning problems there are classes C_1 to C_n instead of the two positive and negative classes P and N used for binary learning. Therefore, the confusion matrix looks different. As can be seen in Table 2.6, T_i denotes the number of correctly classified samples of class i. In turn, F_i denotes the number of incorrectly classified samples by summing up all wrong assignments for class i.

		Predicted			
		C'_1	C'_2	$\ldots$	C'_n
Actual	C_1	T_1	$F_{1,2}$	$\ddots$	$F_{1,n}$
	C_2	$F_{2,1}$	T_2	$\ddots$	$F_{2,n}$
	$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\vdots$
	C_n	$F_{3,1}$	$F_{3,2}$	$\ldots$	T_n

$$\text{with } F_i = \sum_j F_{i,j} \text{ and } n = \sum_i C_i$$

Table 2.6.: Multiclass confusion matrix.

Since the confusion matrix is different, formulae of performance measures have to be adjusted. According to Labatut et al. [71] they can be computed as follows.

- Accuracy and error rate are calculated similarly for both binary and multiclass classification, namely

$$\text{accuracy} = \frac{\sum_i T_i}{n}$$

and

$$\text{error rate} = \frac{\sum_i F_i}{n} = 1 - \text{accuracy}$$

- For multiclass learning, there are error rates for each class defined as

$$\text{error rate}_i = \frac{F_i}{C_i}.$$

- The TP rate is defined for each class as well, namely

$$\text{TP rate}_i = \frac{T_i}{C_i} = 1 - \text{error rate}_i.$$

Performance can also be measured by F-Score and Jaccard Coefficient [71]. Since these are rarely used in practical applications for finance problems, these are left out and only the common ones are presented.

Table 2.7 depicts the confusion matrix for multiclass IDK-classifications.

		Predicted				
		C'_1	C'_2	$\dots$	C'_n	
Actual	C_1	T_1	$F_{1,2}$	$\ddots$	$F_{1,n}$	*IDK*
	C_2	$F_{2,1}$	T_2	$\ddots$	$F_{2,n}$	
	$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\vdots$	
	C_n	$F_{3,1}$	$F_{3,2}$	$\dots$	T_n	

with $F_i = \sum_j F_{i,j}$ and $n = \sum_i C_i$

Table 2.7.: Multiclass confusion matrix for IDK-classifications.

- Using IDK-classifications, accuracy and error rate are calculated as follows

$$\text{accuracy}_\tau = \frac{\sum_i T_i + (1-\tau)\cdot IDK}{n}$$

and

$$\text{error rate}_\tau = \frac{\sum_i F_i + \tau \cdot IDK}{n} = 1 - \text{accuracy}_\tau.$$

3 Interpretability of Models

Machine learning models can be categorized into interpretable or non-interpretable. This distinction depends on whether it is comprehensible how classifications are obtained. In this section we define the term and motivate the usage of interpretable models and IDK-classifications. Section 4.3.1 in the next chapter deals with interpretable models and attempts to make black-box models interpretable.

3.1. Why Do We Need Interpretable Models and IDK-classifications?

The question may arise why models need to be interpretable and why not simply apply black-box models with a high accuracy. Florez-Lopez et al. [39] listed three important benefits from interpretable models.

- Interpretable models allow to justify the decision of a refused credit [51, 103] which is actually a legal obligation in some countries, e.g., in the UK and the US [29].

- Managers are less likely to refuse to use a model which they understand [100]
- Models that are understood can be combined with expert knowledge to obtain a more powerful model [38].

Sometimes the prediction for some instances may be of a high uncertainty. In some applications, it would be helpful to have an IDK-classification for these instances rather than guessing a label. These IDK-classified instances can be treated in a more sophisticated or expensive process. In insolvency prediction, for example, if an enterprise is labeled solvent with a high uncertainty, the possibility to lose the money of a granted credit to this enterprise can be reduced by manually reevaluating this enterprise again by experts.

Technically, IDK-classifications are simply an additional class label which does not occur in the training set. Therefore, each multiclass model can represent IDK-classifications. there are several challenges when dealing with IDK-classifications, like the training process where none of these labels are observed, finding a reasonable amount of IDK-classifications, and using a decent error measure when evaluating the performance.

3.2. Definition of Interpretability of Models and Model Classes

In the literature, the definitions for interpretability in terms of machine learning are different. A model is called interpretable if the importance of features is derivable [62] or if it consists entirely of interpretable rules, no matter how many there are [39]. However, the latter applies for RFs as well which are not considered interpretable in the literature [78]. Thus, we suggest in our recently published studies [85, 86] that interpretable models need to be interpretable by human beings and therefore should

- consist entirely of human-interpretable rules and
- consist of a reasonable amount of rules.

The less rules there are in a model the more interpretable it becomes. The rules should have a structure of what a human being would think of, e.g., Boolean expressions with threshold indicators. In this work, a threshold indicator is the Boolean indicator function of a threshold of a financial ratio, e.g., "Cash flow $>$ 1.5 mil. €". The indicator returns 1, if the condition displayed is satisfied and 0 otherwise. Models using threshold indicators are, e.g., DNFs and DTs.

In this thesis we specify this definition by two additional items stating that interpretable models

- consist of human-interpretable operations connecting these rules and
- do not consist of to many different types of operations.

This supplement ensures that interpretable rules are not connected by to many different types of connections. Although, conjunctions, disjunctions, cascades, trees, and majority votes are interpretable. In our opinion, combining multiple of these operations destroys the interpretability of the model for humans.

We are aware that this definition is kind of vague. However, we cannot give an exact list of rules or an exact number of rules or different operations which we consider interpretable. This is for the reason that there might be interpretable rules which we have not thought of, the number of rules is to some extend a matter of taste, and the number of different types of operations depends on their type and connection. For example, a tree with majority votes in its nodes is probably considered less interpretable by most people than a cascade of disjunctions of conjunctions.

In the following sections, we apply our definition to the models studied in this thesis. We show whether models provide interpretable results. Furthermore, we show whether meta-algorithms preserve the interpretability of interpretable binary models.

3.2.1. Binary Models

This section uses the definition above to categorize binary models into interpretable (white-box or glass-box) and non-interpretable (black-box) models. Gray-box is another term used in the literature to describe a partial interpretable model. However, according to our definition gray-box means non-interpretable. Unfortunately, the second part of our definition, a restricted model size, is normally not provided by model classes. Therefore, the model size has to be restricted by the parameters of the algorithm. The categories are summarized in Table 3.1.

Model	Provides/preserves interpretability	Box
DNFs	✓	□
DTs	✓	□
CNFs	✓	□
DLs	✓	□
Scoring tables	✓	□
Decision diagrams	✓	□
RFs	✗	■
ANNs	✗	■
SVMs	✗	■
Bagging	✗	■/■
Boosting	✗	■/■

Table 3.1.: Categorization of binary machine learning models concerning their interpretability.

3.2.1.1. Disjunctive Normal Forms

A DNF model predicting insolvency is shown below. If the Boolean formula is fulfilled, the enterprise is labeled solvent, otherwise insolvent.

$$\begin{aligned}\text{Solvent} = \quad & \big((\text{Cash flow} > 1.5 \text{ mil. €}) \ \wedge \ (\text{RoI} > 8\%)\big) \\ \vee \ & \big((\text{Cash flow} > 1 \text{ mil. €}) \ \wedge \ (\text{RoI} > 10\%)\big)\end{aligned}$$

DNF models are considered interpretable [91, 97] and analysts or managers can understand how classifications come about. In this example we see that a lower financial ratio (Cash flow: 1.5 mil. € $\rightarrow$ 1 mil. €) can be compensated by a higher one (RoI: 8% $\rightarrow$ 10%). The most important financial ratios and thresholds for classification can easily be detected by their number of appearance. This can be used to obtain a selection of most important ratios as well.

3.2.1.2. Decision Trees

Figure 3.1 shows an exemplarily DT. Starting at the root node, if the condition in a node is fulfilled the right edge, otherwise the left edge is used to get to the next node or leaf. The leaves determine the solvency status of the enterprise. Since this exemplarily DT is a direct transformation of the DNF model above, it can be interpreted the same way. In the literature, DTs are categorized as interpretable models as well [8, 26, 35, 109].

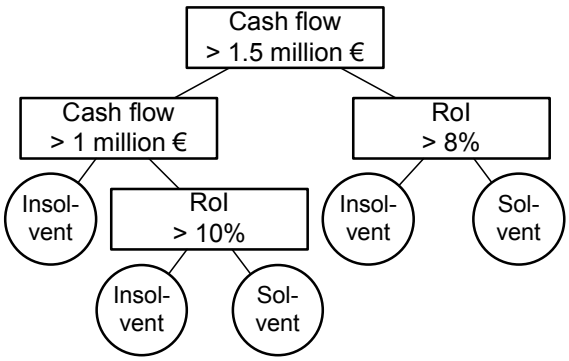


Figure 3.1.: An example DT model.

3.2.1.3. Other Interpretable Models

In addition to DNFs and DTs, there are other interpretable models as well. However, they are either similar or less popular compared to the previously presented DNF and DT models. Thus, they were not evaluated in our case studies.

- CNFs are Boolean conjunctions of disjunctions.
- DLs are a sequence of if-then-else rules where the if-conditions are Boolean formulas and the then-statements are assignments of labels.
- Scoring tables are lists of unordered pairs of conditions and points. The points of the conditions that are fulfilled are added up to a score. If the score is above a certain threshold, the corresponding classification is assigned.
- Decision diagrams are directed graphs with a starting node. Conditions in the nodes determine the next node. Nodes with no outgoing edges determine the classification.

All of these models consist of interpretable rules whose number can be restricted. Therefore, they are called interpretable in the literature [39, 81, 103].

3.2.1.4. Random Forests

RFs consist of interpretable rules. However, there are some issues that make RFs less interpretable than DTs. The main issue is the multiplicity of trees. The first study on RFs uses hundreds or thousands of trees [18]. In our case studies, just ten trees suffice to obtain the best result. However, even this amount of rules prevents easy interpretation. Another issue is the bigger size of each single tree because there is no pruning. Thus, for real-life applications the size of a RF is too big to be interpretable. The idea of restricting the number of trees and the size of each tree even further to obtain an interpretable model can be discarded for the following reasons. This model is not intended to grow only a small interpretable amount of trees, the performance may drop significantly, and one could use one DT in the first place.

However, there are some approaches that can help interpreting the model, e.g., by determining the importance of features by the number of their occurrence, or by visualizing the ensembles as graph [78]. And after all, the model consists of many big, but still individually interpretable trees. Therefore, in the literature, they are sometimes categorized as gray-box [88] instead of black-box models [26].

3.2.1.5. Artificial Neural Networks

The original structure of an ANN is not easily interpretable by a human, especially, when there are multiple hidden layers with multiple nodes. However, there are some approaches which try to extract interpretable rules from these networks. These approaches help in understanding the outcome of ANNs.

However, such approaches have some drawbacks, e.g., simplification of the original model which results in a reduced accuracy. On the other hand very big models keep the accuracy of the original model, but gain less interpretability. For details on approaches to interpret ANNs see Section 4.3.1. Since there is that much much work on interpreting

ANNs [27, 40, 59, 74, 105], they should rather be considered as gray-box models instead of black-box models [26, 53].

3.2.1.6. Support Vector Machines

SVMs are prime examples of black-box models. Hyper planes in high dimensional spaces are not imaginable for the human brain. However, there are attempts to open the black-box as well. Nevertheless, research in this topic is less far advanced compared to research for ANN interpretation. Therefore, SVMs are still considered black-boxes [26, 84]. For details on work on interpreting SVMs see Section 4.3.1.

3.2.1.7. Bagging and Boosting

Since bagging and boosting are meta-algorithms it can not be determined whether they build interpretable models. The categorization depends on the algorithm used. Therefore, it can only be determined whether these meta-algorithms preserve interpretability. The hypothesis classes of bagging and boosting using DTs are similar to the hypothesis class of RFs. They are majority votes of DTs. However, in the case of boosting, these votes are additionally weighted. Nevertheless, we denote them non-interpretable like RFs. Since it is not discussed in the literature, we do not assign the label gray-box or black-box.

3.2.2. Multiclass Models

Multiclass models based on an interpretable binary model are not necessarily interpretable as well. This holds for some multiclass meta-algorithms, e.g., one-vs-one and one-vs-rest. In turn, multiclass models based on a non-interpretable binary model are also non-interpretable. Therefore, we skip the discussion of multiclass models based on binary non-interpretable models in this section. Since we are not using them in our studies, we skip the "other interpretable models" from Section 3.2.1.3 as well. The categories concerning interpretability of multiclass models are summarized in Table 3.2.

Model	Provides/preserves interpretability
DTs	✓
Cascaded DNFs	✓
One-vs-one	✗
One-vs-rest	✗
One-vs-next	✓
One-vs-followers	✓

Table 3.2.: Categorization of multiclass machine learning models concerning their interpretability.

3.2.2.1. Decision Trees

DTs are naturally interpretable multiclass classifier. Their leaves can assign arbitrary class labels while their structure remains the same as in the binary case. However, their size might increase linearly with the number of class labels compared to the size of their binary models. Thus, the multiclass version of a DT can still be considered interpretable. The C4.5 DT algorithm uses this form for multiclass classification.

3.2.2.2. Cascaded DNFs

Interpretable DNF multiclass classification can be obtained by cascading number of labels $l-1$ binary DNF classifiers which classify one class with one DNF consecutively. This cascade of DNFs can be seen as a DL of DNFs. Algorithm 2 shows how such a classification is achieved. Since this model is basically an interpretable DL which grows only linearly in size with the number of class labels, it can be considered an interpretable multiclass model.

Algorithm 2: Cascaded DNF classifier

Input : DNFs $DNF_1,\ldots,DNF_{l-1}$, label assignments $c_{DNF_1},\ldots,c_{DNF_l}$, unknown instance $\mathbf{X}$
Output: Predicted class of $\mathbf{X}$

if $DNF_1(\mathbf{X})$ **then return** c_{DNF_1};
else if $DNF_2(\mathbf{X})$ **then return** c_{DNF_2};
$\vdots$
else if $DNF_{l-1}(\mathbf{X})$ **then return** $c_{DNF_{l-1}}$;
else return c_{DNF_l};

RIPPER uses this model for multiclass classification. To build such a classifier, the algorithm orders the classes of the data set ascending by their size $c_1,\ldots,c_l$ and trains $DNF_1,\ldots,DNF_{l-1}$ using labels $c_1,\ldots,c_{l-1}$ as positive data and labels $\bigcup(c_i, i>1),\ldots,\bigcup(c_i, i>l-1)$ as negative data. The training leads to a cascaded DNF as described in Algorithm 2. This method is basically the one-vs-followers approach with a different class ordering.

3.2.2.3. Models based on the One-vs-one and One-vs-rest Meta-algorithm

These two meta-models are majority votes of binary models like RFs. Therefore, it is hard to understand the outcome of a model. Both models increase in size with the number of class labels. The one-vs-one method's size even increases quadratically. Thus, we classify them non-interpretable independent of the binary model used.

3.2.2.4. Models based on the One-vs-next and One-vs-followers Meta-algorithm

These two approaches train cascaded classifiers like the cascaded DNFs above. The model size grows linearly in size with the number of class labels as well. We consider cascades of models more comprehensible than majority votes. To understand the outcome, cascades have to be evaluated successively and majority votes have to be evaluated all at once. The latter case renders it much more difficult for a human to keep track of the whole model. Thus, the one-vs-next and the one-vs-followers method can be categorized interpretable if the base classifier is interpretable as well.

According to this definition, cascades of DTs should be interpretable as well. However, multiclass DTs are naturally interpretable. Restricting them to binary outputs and cascading them would actually make the model less interpretable than a single DT. Therefore, we only use single multiclass DTs instead of cascaded DTs in this work.

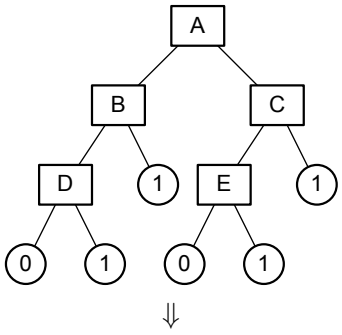
3.3. Comparison of Interpretable Model Classes

All interpretable models presented in this thesis are similar at least from a theoretical point of view. Many of them can be transformed into each other. From a practical point of view, the differences of the models may in fact matter, but more in a matter of taste. For example, one analyst prefers graph-based models where another one prefers Boolean formulas. In the following, a brief description between the relationship of these model classes is given.

- DNFs and CNFs can be transformed into each other using De Morgan's laws and inverting the output classification.
- DNFs are a subset of DLs where each if-condition is a monomial of the DNF and each then-statement returns 1 except for the default case which returns 0.
- Scoring tables can be expressed as DNFs by forming a disjunction of all conjunctions of all combinations of conditions fulfilled with the sum of their points bigger than the threshold. Obviously, this leads to a very big model.
- Decision diagrams are a generalization of DTs. They can be transformed into each other where the size of the DT is at least as big and in many cases even bigger than the size of the decision diagram.

Since DTs and DNFs are actually evaluated in our case studies, their relationship is explained in more detail. The pure hypothesis classes are, in fact, the same because it is possible to transform a DNF into a DT and vice versa. However, in practical applications both classes are restricted by parameters which on the other hand prevents a transformation. Performing a transformation, they might significantly change in size (number of comparisons) as seen in Figure 3.2 and 3.3. Additionally, both figures show that DTs are smaller for recurring comparisons and DNFs are smaller for different comparisons. Thus, they can

be interpreted differently and DNFs offer an alternative view on the interpretation of the data.



$$\left(\overline{A} \wedge B\right) \vee \left(\overline{A} \wedge \overline{B} \wedge D\right) \vee \left(A \wedge C\right) \vee \left(A \wedge \overline{C} \wedge E\right)$$

Figure 3.2.: This figure shows an example of how a DT can be transformed into a DNF considering all paths leading to positive labeled leaves. The conjunction of all nodes in a path represents a monomial and the disjunction of all monomials represents the DNF.

Both methods have generalization parameters to adjust the tradeoff between interpretability and accuracy as well as underfitting and overfitting by changing the model size. Using the C4.5 algorithm, DTs can be adjusted indirectly by changing the amount of pruning or changing the minimum number of instances in a leaf. The RIPPER algorithm can indirectly adjust DNFs by changing the amount of pruning as well and by determining the minimum number of instances in a rule. Likewise, our Thresholder algorithm (see Section 5.1.1) allows to indirectly adjust the DNF model size by determining the amount of pruning. Additionally, there are three parameters to direct adjust DNFs. Two of them determine the maximum number of literals and monomials and the third one determines a desired model size to which the model is pruned down.

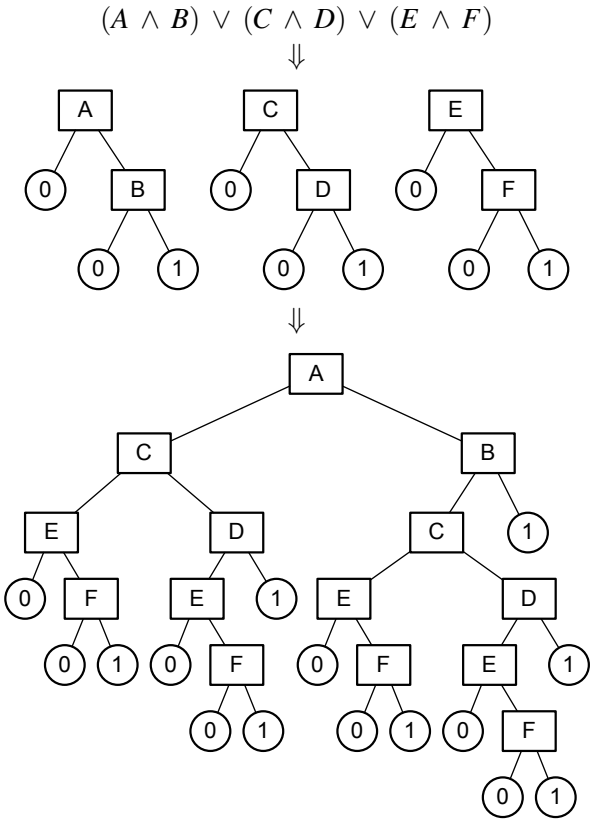


Figure 3.3.: This figure shows an example of the transformation of a DNF into a DT. Each monomial represents a single DT and the outcome is the disjunction of these DTs. They can be merged into a single tree by starting with one tree and inserting the other trees one after another at each leaf labeled 0.

4 Related Work

There are several binary classification problems in finance, e.g., predicting bankruptcy, insolvency, business failure, or financial distress. Credit rating or bond rating with more than two classes are typical multiclass classification problems. However, there are many studies which examine these problems for only two classes.

Most studies are solely based on data obtained from annual accounts. Despite the fact that few studies also consider qualitative factors [7], this paper focuses on quantitative data. A common problem is acquiring useful data sets since annual accounts of enterprises have to be collected from different sources, declarations of insolvency are only published for a limited amount of time, and rating classes which are not publicly available are determined by credit rating agencies. Therefore, many studies suffer from small and different data sets as well. Thus, their absolute results are not directly comparable. A second problem of most data sets are big imbalances. Naturally, there are less insolvent or low rated enterprises for a given time period than solvent ones. Inhomogeneities are a third problem. Predictions for a mixture of enterprises of different sizes, of different industries, and with annual accounts

from different years are more difficult than they are for homogeneous data sets. Many studies used a downsampling technique for the data set by pairing instances of each class to balance the classification ratio of the data set. They matched every instance of the smaller class with to of the bigger class, e.g., by industry or asset size. In some studies the data sets are restricted to some features (e.g., enterprise size) to obtain homogeneous data sets.

In the following, we provide an overview of the most used techniques and the size and the structure of the data sets used for evaluation. This chapter is divided into three parts. It is about general statistical and machine learning methods, methods for multiclass problems, and interpretable models in finance.

4.1. Machine Learning in Finance in General

The following is an overview about prior studies on binary financial problems using statistical and machine learning approaches. It shows which methods are used and that in most studies at least one of the data problems stated above is present.

An early study on business failure was published in 1966 by Beaver [11]. The model used was univariate meaning that classifications are obtained by only one variable (financial ratio). The author worked with 158 instances, paired by industry and asset size.

A few years later (linear) Multiple Discriminant Analysis (MDAs) were used more frequently. This method allows for the use of more than one variable and finds a linear combination of them for prediction. Altman [5] used this to predict corporate bankruptcy with 66 instances, paired by industry and asset size. Other studies followed the same approach [14, 72].

In the seventies, the logit model gained popularity. In this model, data is fitted to a logistic function which predicts the probability of the occurrence of an event. Ohlson [87] worked with 105 bankrupt and 2058 non-bankrupt industrial enterprises to predict bankruptcy. Martin [76] studied bank failure with 23 failed and 5575 non-failed Federal Reserve member banks. Gentry et al. [48] worked with 66 instances paired by industry, asset size, and sales to classify bankrupt firms.

The probit model is similar to the logit model, but less popular. The main difference is that data is fitted to a cumulative standard normal distribution function instead of a logistic function. It was used by Zmijewski [118] to predict financial distress. 40 positive and 800 negative instances were used with the industry code less than 6000, according to standard industrial classification [106]. Another study using the probit model by Skogsvik [94] predicts business failure. 51 failed and 328 non-failed Swedish mining or manufacturing companies with more than 200 employees or at least 200 million SEK[3] assets were used.

DTs are concepts where decision conditions are placed as nodes in a tree and labels are assigned by its leaves. The recursive partitioning method was used by Frydman et al. [45] to create DT classifiers. They applied this method to train a classifier with 58 bankrupt and

[3] Swedish krona, 200 million SEK are about 21 million €

142 healthy companies. No restrictions in the choice of companies are mentioned. There are several other DT algorithms like ID3 or its extension C4.5. The latter was used by Fernandez et al. [37] for bankruptcy prediction on a data set of 29 failed and 37 non-failed banks. Fritz et al. [43] studied regression trees in which each leaf of a tree can contain a regression function. In their case study, they worked with 2580 good and 1019 bad German accounts with an annual turnover of between 5 and 50 million DM[4]. There are up to 51 accounts for one enterprise each covering an interval of one month. The author studied five different methods to classify these accounts. Feldman et al. [35] used the CART algorithm to build DTs for a mortgage default prediction. They evaluated their method with 3,035 mortgage contracts from Israel where about half of them were delinquent.

ANNs try to simulate a network between simple neurons which provide complex global information. They were used by Tam et al. [101] for failure prediction for a pairing of 59 failed and 59 non-failed Texas banks matched by asset size, number of branches, age, and charter status. For each bank there were financial statements for two time periods available. Wilson et al. [117] performed a bankruptcy prediction with 65 and 64 instances matched on industry and year. Charitou et al. [22] studied 51 pairs of industrial firms. Neves et al. [83] worked with a data set of 583 distressed companies of about 2800 French industrial companies, most of them of small to medium size with 35 to 400 employees.

SVMs use hyperplanes to classify data sets. They were used to study 86 bankrupt and 88 non-bankrupt small and medium Australian industrial firms by Fan et al. [33]. Härdle et al. [56] studied smooth SVMs to predict insolvency with 10,468 solvent and 811 insolvent German financial statements for estimation and testing. Each of these enterprises has statements for up to six different years. Accordingly, the amount of enterprises is much smaller than the amount of statements. Harris [52] used clustered SVMs to reduce the computational complexity. A German and a Barbadian credit scoring data set with 300 and 503 non-creditworthy and 700 and 21,117 creditworthy applicants were used. Danenas et al. [30] proposed an approach also using linear SVMs to save computational complexity in the credit risk domain. Their evaluation was based on 5527 risky and 15,961 non-risky entries.

A less widespread method in this field of research is Rough Set Theory (RST). Usually, continuous values of features are transformed into finite sets of low cardinality. Based on these sets, decision rules are created. Slowinski et al. [95] were the first to use this for bankruptcy prediction on a very small sample of 39 clients of a Greek investment bank. McKee et al. [77] did a bankruptcy prediction on a sample of 100 bankrupt and 100 non-bankrupt companies of different industries. Bose [16] distinguished 120 healthy and 120 unhealthy dot-com enterprises using the RST approach.

A heuristic for learning DNFs was used by Brodag [19] for insolvency prediction based on German and Austrian data of about 177 paired enterprises paired by industry.

In recent years, the focus changed from single learning algorithms to ensemble learning algorithms to improve the performance of the models. Boosting and bagging are among

[4]Deutsche Mark, 5 to 50 million DM are about 2.5 to 25 million €

the most popular representatives of this class of methods. As described in the foundations chapter, boosting improves the performance of a learning algorithm by reweighting instances which are hard to learn. Therefore, boosting primarily decreases the bias of the base learner. Bagging improves the performance of a learning algorithm by randomly drawing bootstrap samples from a sample and casting a majority vote over all trained classifiers. This primarily decreases the variance of the base learner. Wang et al. [114] introduced a feature selection for boosting and compared it with boosting and bagging. They worked with a data set of 112 failed and 128 non-failed companies from 1997 to 2001 and another one of 66 risk cases and 66 non-risk cases from 1970 to 1982. Nanni et al. [82] used a feature selection method in combination with bagging which outperforms the standard bagging approach. Abellán et al. [1] presented a new procedure to build DTs which outperforms the feature selection of Nanni et al. [82]. Both worked with three data sets containing non-bankrupt and bankrupt instances from Australia (307/383), Germany (700/300), and Japan (307/383). Kang et al. [63] used genetic algorithms to improve the output of bagging and boosting on a data set of a Korean bank with 600 bankrupt and 600 non-bankrupt manufacturing firms from 2002 to 2005. There are other studies on bankruptcy prediction using boosting as well [2, 3].

A detailed overview of the history of methods used on business data can be found in Balcaen et al. [9] and Dimitras et al. [31]. A general survey on bagging and other ensemble techniques in bankruptcy prediction can be found in Verikas et al. [112]. This section shows that most of the studies were based on very small or imbalanced data sets which is often a problem in insolvency prediction.

4.2. Machine Learning in Finance for Multiclass Problems

Most of the studies on multiclass credit rating propose SVMs or ANNs. However, interpretable DNFs or DTs are hardly ever considered.

Huang et al. [58] performed a credit rating comparing methods for improved accuracy of SVMs and improved interpretability through feature extraction for ANNs. However, they realized that only a slight performance improvement of SVMs was achieved. They worked with two data sets from the US and Taiwan with 265 and 74 instances in five different rating classes.

Instead of a credit rating a bond rating with six classes was done by Cao et al. [21] on a data set with 239 enterprises from the same country and industry code. They tested different multiclass methods for SVMs and compared these results with ANNs and logit models. A main finding was that few features are not only sufficient, but can even improve the accuracy. They achieved an accuracy gain of about 2%.

Hájek et al. [55] proposed different ANNs for municipal credit rating and compared them with classification trees and SVMs. Probabilistic neural networks obtained the best results. However, the interpretable classification trees achieved good results as well. They

used values from 169 US municipalities from 2003-2007 resulting in 766 instances divided into nine and four classes. It was concluded that only small lists of features determine the classification.

Kim et al. [67] performed a credit rating with four rating classes using 1295 enterprises from the manufacturing industry in Korea. They propose new multiclass methods for SVMs and compare them with other methods. Their method outperformed the rest, but only with an advantage of less than 1%.

Guo et al. [49] studied credit rating with four classes as well. The same Korean data from above [67] was used in addition to a data set from China. They used a support vector domain combined with a fuzzy clustering algorithm and compared it with different SVM multiclass methods. Their approach outperformed conventional multiclass methods with less than 2%.

A credit rating with 16 classes was performed by Kwon et al. [69]. Their data set contains 1480 companies and covers the period from 2002 to 2012 resulting in 21,321 instances. They used double ensemble approaches containing bagging and boosting to significantly improve DTs.

4.3. Interpretable Models

Up to recently, insolvency prediction focused on improving the accuracy of the models. The models became more complex and their accuracy was improved, as for instance in ensemble learning. Recently, there are more and more papers being published that focus on interpretable models.

4.3.1. Interpretable Models in General

There are general approaches that try to simplify non-interpretable models. They render models interpretable by extracting rules or pointing out feature importance. Some approaches combine interpretable models to a more accurate but bigger interpretable model.

Some approaches focus on extracting rules from less interpretable models like ANNs. To mention just a few, Craven et al. [27] built DTs from ANN predictions. Jang et al. [59] showed equivalence of RBF-Networks and fuzzy inference systems. Mantas et al. [74] extracted fuzzy rules from multilayer perceptrons. Feraud et al. [40] explained multilayer perceptrons by clustering the data and selecting important variables for each cluster. Johansson et al. [60] introduced a rule extraction method based on genetic programming, which can transform ANN models into different interpretable models. A graphical visualization to understand ANNs was applied by Tzeng et al. [105]. Some approaches try to simplify black-box models like SVMs. Martens et al. [75] adopted methods, already applied to ANNs, to build DTs from SVMs. Barbella et al. [10] showed interpretations for single data points. Su et al. [99] extracted rules from SVMs. All of these extracted rules do only represent an approximation of the original model and therefore lose accuracy.

Some researchers combine simple rules like Florez-Lopez et al. [39]. They merged multiple DTs which leads to a higher accuracy, but additionally to more rules than in a single DT and therefore to less interpretability. Kainulainen et al. [62] introduced a method which calculates ensembles of linear models. This method is comparable to SVMs regarding the accuracy, but more interpretable because of an embedded feature selection which points out the important ones. For all methods above there is a tradeoff between accuracy and interpretability. Models gain accuracy by getting bigger and thus lose interpretability. Vice versa, non-interpretable models lose accuracy by becoming interpretable. Although, the loss in accuracy often is very small.

4.3.2. Interpretable Models in Finance

In finance, most studies concentrate on improving the prediction accuracy. There are few studies dealing with interpretability of models, especially for multiclass problems. One of these studies was performed by Kim et al. [66] on a small data set with 228 enterprises to predict six class bond rating. They compared DTs, ANNs, MDAs, and logit models. ANNs performed much better than the rest. However, this study dates back to 1993, the data set is very small, and DT algorithms have evolved a lot since then. As mentioned above, Huang et al. [58] improved interpretability in credit rating through feature extraction for ANNs. Mues et al. [81] extracted rules and DTs from ANNs to build interpretable decision diagrams which are smaller than DTs. They did a binary credit-risk prediction on a German credit data set publicly available at UCI repository[5] and two data sets from major Benelux finance institutions. There are no details provided about the latter data sets. Tomczak et al. [103] created scoring tables from ANNs (restricted Boltzmann machines) to build binary credit scoring models. They worked with four data sets from different countries with 471 to 150,000 instances and partially huge imbalances. De Bock et al. [15] studied the problem of customer churn prediction. They applied interpretable ensemble classifiers based upon generalized additive models. The classifier yields interpretability by pointing out importance of features and confidence bands and average trends of the entire range of a feature. The data is obtained from six real-life churn prediction projects of European companies which consist of 3827 to 43,305 instances.

For binary problems in finance, research has shown that already existing interpretable models are not necessarily worse than more complex models. Jones et al. [61] concluded that simpler and more interpretable classifiers like logit, probit, and MDAs performed comparatively well to ANNs and SVMs. Their task was to predict rating changes with 2891 good ratings and 2162 bad ratings from 1983 to 2013 of enterprises from the US. Vir'ag et al. [113] studied RST for bankruptcy prediction. They showed that this interpretable model is competitive to ANNs and SVMs. Their database contains 78 bankrupt and 78 non-bankrupt enterprises with 16 financial ratios.

[5] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

On a final note, David Hand [51] argues that "the apparent superiority of more sophisticated methods may be something of an illusion". He showed that for many cases the marginal gain of sophisticated and ensemble models is small compared to simple models. Section 4.2 shows similar observations for most multiclass credit rating studies. Nevertheless, we appreciate the work on sophisticated methods because classification performance is still a more important factor for a classifier than interpretability.

4.3.3. Our Work on Interpretable Models in Finance

In our first case study [85], we follow the trend of examining interpretable models for insolvency prediction. We take two of the most interpretable model classes, namely DTs and DNFs, and restrict the model size. Thus, the resulting models stay interpretable. We compare them with the most common methods, namely ANNs and SVMs. RFs are used to represent combined interpretable models using thresholds. A huge database with financial statements of 246 insolvent and a selection of 4916 solvent companies is used. The selection is a matching of 20 solvent enterprises per one insolvent enterprise, if available of the same industry and size[6].

In our second case study [86], we examine interpretable multiclass models for a three-class credit rating. We consider the interpretable model classes of DTs and DNFs using different learning algorithms to build the models. The model size is restricted to obtain small and interpretable models. Again, we compare them with ANNs and SVMs. RFs are used to represent combined interpretable models using thresholds. We compare the multiclass methods used by Guo et al. [49] and an ensemble method representatively for the work of Kwon et al. [69]. Three data sets with financial statements of 1256 trading, 1361 construction, and 1066 financial enterprises are used for a three-class credit rating.

[6]Enterprise size is determined by internal criteria of our database.

5 Cascaded DNFs as Interpretable Multiclass Models

In this chapter, we present the Thresholder heuristic which was developed and implemented within the scope of this thesis. The implementation can be found on the website of our research group.[7] This algorithm builds DNFs for binary classification problems and cascaded DNFs for multiclass classification problems. Following the description of the algorithm, a small summary of its benefits is given.

5.1. Thresholder Heuristic

This section describes the Thresholder algorithm. The binary Thresholder learning algorithm for learning DNFs is described as well as the multiclass version training cascaded DNFs. At last, we present a general method to obtain IDK-classifications and apply it to our interpretable multiclass model.

[7] `http://filepool.informatik.uni-goettingen.de/publication/tcs/2016/thresholder.zip`

5.1.1. Learning DNFs

A DNF is a disjunction of monomials m with a conjunction of literals l

$$\begin{aligned} DNF &= m_1 \vee m_2 \vee \ldots \vee m_r \\ &= (l_{1,1} \wedge l_{1,2} \wedge \ldots \wedge l_{1,p}) \vee (l_{2,1} \wedge l_{2,2} \wedge \ldots \wedge l_{2,p}) \vee \ldots \vee (l_{r,1} \wedge l_{r,2} \wedge \ldots \wedge l_{r,p}), \end{aligned}$$

with a maximum number of monomials r and a maximum number of literals per monomial p. In our work, these literals are threshold indicators. If this formula is fulfilled, the instance will be classified positive, otherwise negative.

The Thresholder heuristic calculates a DNF of threshold indicators as hypothesis h. Ideally, the hypothesis h chosen should minimize the empirical risk [108, p. 20f]. According to their different importance, an error on a positive instance $(\mathbf{X},1) \in \mathbf{U}$ counts for $\omega \in \mathbb{R}^+$, whereas an error on a negative instance $(\mathbf{X},0) \in \mathbf{U}$ counts for 1. The value ω provides us with the possibility to implement weights between the α-error on the negative instances and the β-error on the positive instances. Unfortunately, empirical risk minimization in the case of the hypothesis class of DNFs is NP-hard because this is already the case for the subclass of monomials [36]. Thus, heuristics come into play. Modifications of the well-known standard greedy algorithm for the set cover problem [24] are a good choice as proposed by Brodag [19]. Each monomial's threshold is calculated step by step. This is achieved by considering each feature value of the instances as a possible upper and lower threshold candidate and selecting the best one. After a threshold is calculated, the algorithm uses only the positively classified instances for the next threshold in this monomial because adding further conjunctions so far only affects positively classified data. If all p thresholds are calculated or there is no further benefit in adding thresholds, the algorithm builds the next monomial only using the negatively classified instances for the next monomials. Adding disjunctions only affects yet negatively labeled data.

In more detail, assume that we have already computed

$$h_{i-1} := m_1 \vee m_2 \vee \ldots \vee m_{i-1}.$$

Let

$$\mathbf{U}_i := \mathbf{U} \setminus U \in \mathbf{U} \mid \forall U : h_{i-1}(\mathbf{X}) = 1.$$

Furthermore, assume that

$$m_{i,j-1} := l_{i,1} \wedge l_{i,2} \wedge \ldots \wedge l_{i,j-1}$$

as prefix of m_i has just been computed. To compute the j-th literal of m_i, let

$$\mathbf{U}_{i,j} := \mathbf{U}_i \setminus U \in \mathbf{U}_i \mid \forall U : m_{i,j-1}(\mathbf{X}) = 0.$$

The candidate threshold indicators for $l_{i,j}$ are the set of all indicator functions $\mathbf{1}_{\{x_k>t\}}$ or $\mathbf{1}_{\{x_k\leq t\}}$, where $k = 1,2,\ldots,n$, and t is any midpoint between two consecutive data points for feature $x_k \in \mathbf{X}$. This maximizes the margin between the two instances resulting in a better generalization. The candidate threshold indicator l is chosen that maximizes the advantage. The advantage of a literal l on sample $\mathbf{U}$ in turn is derived to be

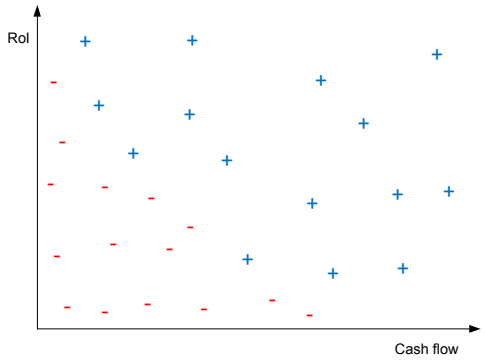
$$adv(l,\mathbf{U}_{i,j}) = \underbrace{\frac{\#\{(\mathbf{X},0)\in\mathbf{U}_{i,j}|l(\mathbf{X})=0\}}{\#\{(\mathbf{X},0)\in\mathbf{U}_{i,j}\}}}_{\text{gain (TN rate)}} - \omega\cdot\underbrace{\frac{\#\{(\mathbf{X},1)\in\mathbf{U}_{i,j}|l(\mathbf{X})=0\}}{\#\{(\mathbf{X},1)\in\mathbf{U}_{i,j}\}}}_{\text{loss (β-error rate)}}.$$

Figure 5.1 visualizes this process for the example model in Section 3.2.1.1.

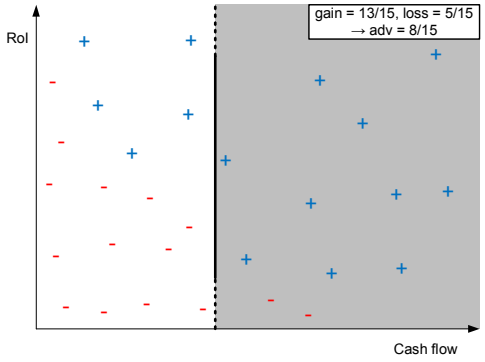
For dimension n and sample size s, the complexity is $\mathcal{O}\left(r\cdot p\cdot n\cdot s^2\right)$, as there are two nested loops over a maximum number of $r\cdot p$ thresholds for which $n\cdot s$ possible upper and lower thresholds are evaluated. The evaluation itself costs s steps for counting how many instances are above or below this threshold. Thus, the heuristic runs in quadratic time with regard to the sample size.

The greedy approach of this heuristic allows a reasonable run time of the program. However, greedy algorithms do generally not find an optimal solution. A bruteforce approach could find an optimal solution, but runs in polynomial time $\mathcal{O}\left((n\cdot s)^{2r\cdot p}\right)$. As a compromise, we propose a semi-greedy heuristic between those two approaches which yields better results and increases computation time only by a linear factor. For each monomial in the DNF, we calculate n monomial candidates $m_1, m_2, \ldots, m_n$ simultaneously, with the first threshold indicator of dimension $1,2,\ldots,n$. The following threshold indicators of each monomial candidate are calculated greedy as before. The best of these n monomial candidates is added as monomial to the DNF. This decision is made according to the classification error of the DNF using this monomial candidate. Afterwards, the next monomial is calculated in the same way. This procedure increases the complexity for each monomial as well as for the whole DNF only by factor n.

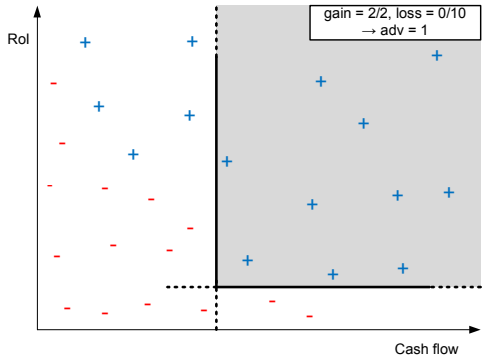
To further improve the heuristic, we use post-pruning which is adapted from the C4.5 DT algorithm. We build a bigger DNF than intended and afterwards prune some threshold indicators to obtain a different DNF. Like the improvement above, this technique should compensate for the greediness of the approach. The process of building the DNF involves adding one literal after another. In contrast, the pruning technique deletes multiple literals at any position in the DNF. Our pruning technique has three parameters pruning complexity p_c, pruning error p_e, and pruning size p_s and works as follows: remove the set of $1\ldots p_c$ threshold indicators or the monomial, whichever worsens the error of the model at least. Repeat this until the error worsens by at most p_e. Depending on the value of p_e, this might increase the training error slightly, but decreases overfitting and therefore might decrease the generalization error. The third parameter p_s restricts the maximum size of the model



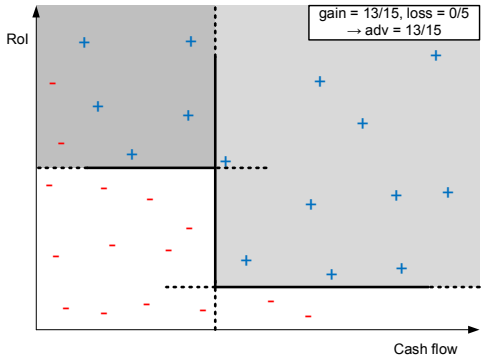
(a) 1. An example data set.



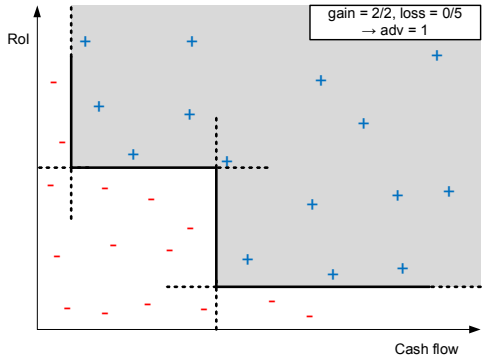
(b) 2. Use all instances for the first monomial.



(c) 3. Use only positively classified instances (dark gray area in (a)) for the second threshold.



(d) 4. Use all negatively classified instances (white area in (b)) for the next monomial.



(e) 5. Use only positively classified instances (dark gray area in (c)) for the second threshold of the second monomial.

Figure 5.1.: An example model calculated by the threshold heuristic for two literals and monomials with $\omega = 1$

measured by the number of threshold indicators. Pruning will not stop until model size is equal or below p_s. This parameter controls the degree of interpretability. Especially for small values of p_s, it is important to prune the DNF to exactly size $p_s + p_c$ and, afterwards, to prune the remaining p_c literals all at once. This enhances the performance by minimizing the influence of the greediness of the pruning algorithm. For reducing the generalization error, these pruning parameters should be selected on a separate data set. The complexity increases at most by factor $r \cdot p \cdot \sum_{k=1}^{p_c} \binom{n}{k}$. Since p_c highly affects the complexity we only recommend using small values.

The pseudo-code of the algorithm using the semi-greedy approach and the pruning technique is given in Algorithm 3.

The threshold heuristic provides two ways of reducing the α-error, although this comes at the cost of a higher β-error. Firstly, ω penalizes misclassification of negative instances, i.e., insolvent enterprises. Secondly, by adapting the parameters r and p, the maximum number of monomials and literals is restricted. Increasing r or decreasing p will reduce the α-error at the expense of the β-error because each monomial offers a chance of a positive classification. The more monomials there are, the higher the probability of a positive classification. The opposite applies to the literals. Each literal limits the chance of a positive classification of its monomial.

Between the first and the second case study, we applied a few improvements for the pruning part of the Thresholder algorithm. They are not displayed in Algorithm 3 for reasons of clarity and comprehensibility, but briefly described in the following. The option to prune a monomial as a whole was added. Additionally, we added some runtime performance improvements such as dynamically lowering p_c for bigger model sizes. As the model size shrinks through pruning, p_c increases until it obtains its adjusted value. Since this parameter exponentially increases computation time with regard to the model size, this performance improvement now allows for using higher values of p_c.

There are several generalization parameters which allow for the output models to be adjusted to one's needs. The maximum number of literals p and monomials r is adjustable. Furthermore, the pruning parameters p_e and p_s allow to output models of a certain size. Smaller models increase the interpretability, but might increase the training error as well. However, a small increase in the training error might increase the generalization of the heuristic. To find the right amount of pruning, we recommend using a separate cross-validation loop for parameter selection.

5.1.2. Learning Cascaded DNFs

Due to its Boolean structure, a DNF is only capable of handling binary problems. To handle multiclass problems with l different labels, at least $l-1$ DNFs are needed. The easiest ways to achieve this model structure are the one-vs-next and one-vs-followers approaches which directly yield a cascade of DNFs. As mentioned before, RIPPER uses the one-vs-followers approach with labels ordered ascending by their occurrence in the data set. The one-vs-one

Algorithm 3: Thresholder heuristic

Input : Training sample $\mathbf{U}$, maximum number of monomials r and literals p, pruning parameters p_c, p_e, and p_s

Output: DNF h

```
/* building                                                    */
```

$\mathbf{U}_0 := \mathbf{U}$;
for $i = 1$ **to** r **do**
  for $k = 1$ **to** n **do**
    $m_k \bigcup \arg\max_l adv(l \mid \text{in dimension } k, \mathbf{U}_0)$;
    $\mathbf{U}_k := \mathbf{U}_k^0 := \mathbf{U}_0 \setminus U \in \mathbf{U}_0 \mid m_k(\mathbf{X}) = 0$;
    for $j = 2$ **to** p **do**
      $m_k \bigcup \arg\max_l adv(l, \mathbf{U}_k)$;
      $\mathbf{U}_k := \mathbf{U}_k^0 \setminus U \in \mathbf{U}_k^0 \mid m_k(\mathbf{X}) = 0$;
    end
  end
  $q := \arg\min_k(\text{err}(h \bigcup m_k))$;
  $h := h \bigcup m_q$;
  $\mathbf{U}_0 := \mathbf{U}_q \setminus U \in \mathbf{U}_q \mid h(\mathbf{X}) = 1$;
end

```
/* pruning                                                     */
```

$prune := true$;
repeat
  if $|h| > p_s + p_c$ **then**
    $L := \{\text{all sets of literals of size at most } p_c\}$;
    $l_{min} := \arg\min_{l \in L}(\text{err}(h \setminus l))$;
    $h := h \setminus l_{min}$;
  else if $|h| > p_s$ **then**
    $L := \{\text{all sets of literals of size } |h| - p_s\}$;
    $l_{min} := \arg\min_{l \in L}(\text{err}(h \setminus l))$;
    $h := h \setminus l_{min}$;
  else
    $prune := false$;
    $L := \{\text{all sets of literals of size at most } p_c\}$;
    $l_{min} := \arg\min_{l \in L}(\text{err}(h \setminus l))$;
    if $err(h \setminus l_{min}) - err(h) < p_e$ **then**
      $h := h \setminus l_{min}$;
      $prune := true$
  end
until $prune$;

and the one-vs-rest approaches naturally do not yield interpretable forms since they consist of majority votes of DNFs.

For our Thresholder algorithm we use all multiclass strategies from Section 2.2.3.1 and convert them into an interpretable form. As mentioned above, the one-vs-next and one-vs-followers approaches already yield an interpretable form since they are trained according to Algorithm 2 with an ordering of class labels. The one-vs-one and one-vs-rest approaches are more difficult because the resulting majority votes consist of interpretable parts, but the whole classifiers are not interpretable, like RFs. In our approach, we solve this problem by transforming them into cascaded DNFs. We achieve this by using Boolean algebra to calculate conjunctions and negations of DNFs. Before we explain this algorithm in detail, we show how applying these two operations on DNFs will maintain the DNFs every time.

5.1.2.1. Conjunctions of DNFs

The conjunction of two DNFs can be transformed back into a single DNF using the distributive property and the associative property of Boolean algebra as seen below. The disjunction of two DNFs is automatically a disjunction of all monomials and therefore a new DNF.

$$DNF_1 \vee DNF_2 = (m_l \vee \ldots \vee m_r) \vee (m'_1 \vee \ldots \vee m'_2) = DNF_{1\vee 2}$$

The conjunction of a DNF and a literal is a new DNF with a conjunction of this literal with each monomial.

$$\begin{aligned} l \wedge DNF_1 &= l \wedge (m_1 \vee \ldots \vee m_r) = (l \wedge m_1) \vee \ldots \vee (l \wedge m_r) \\ &= (l \wedge l_{1,1} \wedge \ldots \wedge l_{1,p}) \vee \ldots \vee (l \wedge l_{r,1} \wedge \ldots \wedge l_{r,p}) = DNF_{l\wedge 1} \end{aligned}$$

Using the two formulae above, the conjunction of two DNFs can be transformed into a single DNF.

$$\begin{aligned} DNF_1 \wedge DNF_2 &= (m_1 \vee \ldots \vee m_r) \wedge DNF_2 = (m_1 \wedge DNF_2) \vee \ldots \vee (m_r \wedge DNF_2) \\ &= ((l_{1,1} \wedge \ldots \wedge l_{1,p}) \wedge DNF_2) \vee \ldots \vee ((l_{r,1} \wedge \ldots \wedge l_{r,p}) \wedge DNF_2) \\ &= (l_{1,1} \wedge \ldots \wedge (l_{1,p} \wedge DNF_2)) \vee \ldots \vee (l_{r,1} \wedge \ldots \wedge (l_{r,p} \wedge DNF_2)) = DNF_{1\wedge 2} \end{aligned}$$

5.1.2.2. Negations of DNFs

The negation of a DNF can be calculated using De Morgan's and distributive laws as seen below. Threshold indicators can be negated by inverting the relational operator, i.e., changing "$>$" to "$\leq$" and vice versa. Using this, the negation of a monomial can be calculated.

$$\overline{m_1} = \overline{(l_{1,1} \wedge \ldots \wedge l_{1,p})} = (\overline{l_{1,1}} \vee \ldots \vee \overline{l_{1,p}})$$

The conjunction of two negated monomials can be transformed into a DNF by using every combination of one literal per monomial as a new monomial.

$$\begin{aligned}\overline{m_1} \wedge \overline{m_2} &= (\overline{l_{1,1}} \vee \ldots \vee \overline{l_{1,p}}) \wedge (\overline{l_{2,1}} \vee \ldots \vee \overline{l_{2,p}}) \\ &= (\overline{l_{1,1}} \wedge \overline{l_{2,1}}) \vee \ldots \vee (\overline{l_{1,1}} \wedge \overline{l_{2,p}}) \vee \ldots \vee (\overline{l_{1,p}} \wedge \overline{l_{2,1}}) \vee \ldots \vee (\overline{l_{1,p}} \wedge \overline{l_{2,p}})\end{aligned}$$

And finally, the negation of a DNF, which is a CNF, can be transformed into a DNF as well using the formulae above.

$$\overline{DNF_1} = \overline{m_1 \vee \ldots \vee m_r} = \overline{m_1} \wedge \ldots \wedge \overline{m_r} = DNF_{\overline{1}}$$

Calculating the conjunction or negation of a DNF exponentially increases the amount of threshold indicators, a problem which will be addressed later in Section 5.1.2.5.

5.1.2.3. Interpretable One-Vs-One Classifiers

Since the data sets of our case study have three classes, we explain our algorithm only for the three class case for reasons of simplicity. However, it can easily be extended to n classes.

We denote DNF_{ivj} the binary classifier which is trained with label j as positive and label i as negative data. Taking an instance of the data set as parameter, it returns true for label j and false for label i. The majority vote of the three one-vs-one classifiers DNF_{0v1}, DNF_{0v2}, and DNF_{1v2} votes for label 2 only if DNF_{0v2} and DNF_{1v2} both return true. It votes for label 1 only if DNF_{0v1} and $DNF_{2v1} = \overline{DNF_{1v2}}$ both return true. It votes for label 0 if DNF_{0v1} and DNF_{0v2} both return false. Otherwise, there is a tie. In this case we assign label 0 as well.

Firstly, we train DNF_{0v1}, DNF_{0v2}, and DNF_{1v2} similar to the normal one-vs-one approach. Afterwards, we build two cascaded DNFs using the conjunctions and negations of DNFs representing the majority votes as seen above. An alternative method is to directly train the negated DNF from the data instead of calculating it. This requires one additional training step. Both methods are shown in Algorithm 4.

5.1.2.4. Interpretable One-Vs-Rest Classifiers

We denote $DNF_{(i,j)vk}$ the binary classifier which is trained with label i and j as negative and label k as positive data. Taking an instance of the data set as parameter, it returns true for label k and false for label i or j. The interpretable one-vs-rest classifier returns a positive classification for class k only if $DNF_{(i,j)vk}$ returns true, $DNF_{(k,i)vj}$ returns false, and $DNF_{(j,k)vi}$ returns false. Like the interpretable one-vs-one method, this method exists in an indirect and direct version. Algorithm 5 shows this procedure in detail.

Algorithm 4: Indirect and direct method for interpretable one-vs-one DNF classifiers.

Input : Training sample $\mathbf{U}$, unknown instance $\mathbf{X}$
Output: Predicted class of $\mathbf{X}$

train DNF_{0v1} on $\mathbf{U}$;
train DNF_{0v2} on $\mathbf{U}$;
train DNF_{1v2} on $\mathbf{U}$;
$\underbrace{DNF_{2v1} := \overline{DNF_{1v2}}}_{\text{indirect method}}$ OR $\underbrace{\text{train } DNF_{2v1} \text{ on } \mathbf{U}}_{\text{direct method}}$;
if $DNF_{0v2}(\mathbf{X}) \wedge DNF_{1v2}(\mathbf{X})$ **then**
  return 2
else if $DNF_{0v1}(\mathbf{X}) \wedge DNF_{2v1}(\mathbf{X})$ **then**
  return 1
else
  return 0
end

Algorithm 5: Indirect and direct method for interpretable one-vs-rest DNF classifiers.

Input : Training sample $\mathbf{U}$, unknown instance $\mathbf{X}$
Output: Predicted class of $\mathbf{X}$

train $DNF_{(1,2)v0}$ on $\mathbf{U}$;
train $DNF_{(0,2)v1}$ on $\mathbf{U}$;
train $DNF_{(0,1)v2}$ on $\mathbf{U}$;
$DNF_{0v(1,2)} := \overline{DNF_{(1,2)v0}}$ OR train $DNF_{0v(1,2)}$ on $\mathbf{U}$;
$DNF_{1v(0,2)} := \overline{DNF_{(0,2)v1}}$ OR train $DNF_{1v(0,2)}$ on $\mathbf{U}$;
$\underbrace{DNF_{2v(0,1)} := \overline{DNF_{(0,1)v2}}}_{\text{indirect method}}$ OR $\underbrace{\text{train } DNF_{2v(0,1)} \text{ on } \mathbf{U}}_{\text{direct method}}$;
if $DNF_{(0,1)v2}(\mathbf{X}) \wedge DNF_{0v(1,2)}(\mathbf{X}) \wedge DNF_{1v(0,2)}(\mathbf{X})$ **then**
  return 2
else if $DNF_{(0,2)v1}(\mathbf{X}) \wedge DNF_{0v(1,2)}(\mathbf{X}) \wedge DNF_{2v(0,1)}(\mathbf{X})$ **then**
  return 1
else
  return 0
end

5.1.2.5. Simplification and Pruning of Cascaded DNFs

As mentioned above, the conjunction and negation operations for DNFs increase the size of the resulting cascaded DNFs exponentially, a problem which does not exist for directly calculated models from the one-vs-next and one-vs-followers approaches. However, in these bigger DNFs, many rules are redundant and can be pruned for simplification.

If a monomial contains multiple threshold indicators of the same dimension and orientation, the less restrictive ones can be discarded without changing the logic of the Boolean formula. In this example, the first threshold indicator can be discarded

$$\begin{aligned}\text{Solvent} = \quad & \big((\text{Cash flow} > 1.45 \text{ mil. €}) \wedge (\text{Cash flow} > 1.5 \text{ mil. €}) \wedge (\text{RoI} > 9.5\%)\big) \\ \vee \ & \ldots.\end{aligned}$$

Whole monomials can be discarded as well; consider the example of the conjunction of two monomials m_1 and m_2, where m_1 is less restrictive than m_2 in every threshold

$$\begin{aligned}\text{Solvent} = \quad & \big((\text{Cash flow} > 1.45 \text{ mil. €}) \wedge (\text{RoI} > 9.5\%)\big) \\ \vee \ & \big((\text{Cash flow} > 1.5 \text{ mil. €}) \wedge (\text{RoI} > 10\%)\big) \\ \vee \ & \ldots.\end{aligned}$$

Then, m_2 can be pruned and m_1 already represents the conjunction.

Using these conversions, which do not touch the outcome of the formulae, the size of the cascaded DNF shrinks significantly. Nevertheless, the model size might become clearly bigger than directly calculated models since there might be monomials which are not exactly as restrictive but only almost as restrictive as other monomials

$$\begin{aligned}\text{Solvent} = \quad & \big((\text{Cash flow} > 1.45 \text{ mil. €}) \wedge (\text{RoI} > 9.5\%)\big) \\ \vee \ & \big((\text{Cash flow} > 1.4 \text{ mil. €}) \wedge (\text{RoI} > 10\%)\big) \\ \vee \ & \ldots.\end{aligned}$$

Pruning them would change the logic of the formula, but in practice this might only affect very few instances indicating the usage of post pruning. Therefore, we apply the same pruning algorithm for both a single DNF and the cascaded DNF classifier.

After all, we have six interpretable multiclass methods implemented for our Thresholder algorithm, namely

- one-vs-next,
- one-vs-followers,
- one-vs-one(-indirect),
- one-vs-one-direct,
- one-vs-rest(-indirect), and
- one-vs-rest-direct.

All of these methods are trained with an ascending and descending order of the class labels in a three-fold-cross-validation of the training data to chose the better order. The all-at-once method of our Thresholder trains all of the six methods above in a three-fold-cross-validation and chooses the best one. When there is a tie, it chooses the one with the smaller model size.

5.1.3. IDK-Classification for Cascaded DNFs

IDK-labels are assigned by classifiers, but cannot be observed in training data sets. We propose to assign IDK-classifications when a tie of the one-vs-one and one-vs-rest multiclass method occurs. Therefore, all classifiers using these methods can implement IDK-classification.

However, we want to go one step further and develop interpretable models using IDK-classifications. In the previous section, we presented interpretable one-vs-one and one-vs-rest multiclass methods for Thresholder. We assign the label 0 in case of a tie. However, if we add an l-th DNF to our cascade, we can distinguish between label 0 and a tie which we can assign an IDK-classification. Algorithm 6 shows this procedure for the one-vs-one method. IDK-classification using the one-vs-rest method works similar.

The all-at-once method of Thresholder using IDK-classifications uses only the four multiclass methods which support IDK-classifications, i.e., the two versions of one-vs-one and one-vs-rest.

Compared to the basic one-vs-one method, the model size is increased by one additional DNF and the training involves negating or training two additional DNFs. As before, the model size can be controlled via pruning. Furthermore, using different values of τ, pruning can control the amount of IDK-assignments. Decreasing τ should result in an increase of IDK-classifications and an increase of the classification error.

Algorithm 6: Indirect and direct method for interpretable one-vs-one DNF classifiers using IDK-classifications.

Input : Training sample **U**, unknown instance **X**
Output: Predicted class of **X**

train DNF_{0v1} on **U**;
train DNF_{0v2} on **U**;
train DNF_{1v2} on **U**;
$DNF_{2v1} := \overline{DNF_{1v2}}$ OR train DNF_{2v1} on **U**;
$DNF_{1v0} := \overline{DNF_{0v1}}$ OR train DNF_{1v0} on **U**;
$\underbrace{DNF_{2v0} := \overline{DNF_{0v2}}}_{\text{indirect method}}$ OR $\underbrace{\text{train } DNF_{2v0} \text{ on } \mathbf{U}}_{\text{direct method}}$;
if $DNF_{0v2}(X) \wedge DNF_{1v2}(X)$ **then**
  return 2
else if $DNF_{0v1}(\mathbf{X}) \wedge DNF_{2v1}(\mathbf{X})$ **then**
  return 1
else if $DNF_{1v0}(\mathbf{X}) \wedge DNF_{2v0}(\mathbf{X})$ **then**
  return 0
else
  return 3 (IDK)
end

5.2. Advantages of Thresholder

In the following, benefits and advantages of Thresholder over the similar RIPPER algorithm and other algorithms are summarized. Thresholder

- builds interpretable binary and multiclass models,
- allows better adjustments of model size and interpretability compared to RIPPER,
- implements different ways of building cascaded DNFs, which allows to chose the best one for a certain task compared to RIPPER which only provides one way to build cascaded DNFs,
- builds interpretable models using IDK-classifications, and
- allows to control the balance between number of IDK-classifications and the classification error rate.

6 Case Studies

This section presents the two case studies of this thesis. The first case study compares interpretable with non-interpretable models for insolvency prediction. The second one compares different algorithms to build interpretable and non-interpretable models for a multi-class credit rating. The former focuses on comparing different model classes and demonstrates non-inferiority of interpretable models in a binary setting. The latter goes into more

detail about comparing interpretable models among each other. Furthermore, it introduces a new interpretable multiclass method and a technique to build interpretable models using doubt in their decisions.

For both studies the underlying data sets, features, and preprocessing techniques used are described, as well as the execution of the experiments and their results and discussion.

6.1. Insolvency Prediction

Insolvency prediction is a binary prediction problem in finance. Most challenges emerge from the data sets. In our case study and in many others (see Chapter 4), inhomogeneous data and imbalanced classification labels are a big challenge. Furthermore, the latter leads to the problem of finding a proper error measure. There is an important difference between errors on negative (insolvent) and positive (solvent) data, named α-errors and β-errors. Therefore, we choose the error measure of the classifiers on grounds of the following two facts.

- The α-error is more important than the β-error. A misclassified insolvent enterprise results in a payment default, whereas a misclassified solvent enterprise only results in the loss of a possible client.
- Naturally there are much more solvent than insolvent enterprises. In our data sets, this results in an imbalance ratio of about 1:1000. If we had taken the standard empirical risk as error measure, the α-error would have been of minor influence.

Thus, we use the mean of the α-error and β-error rate instead. A more detailed explanation can be found in Section 6.1.4.1.

6.1.1. Data

As basis for this case study the DAFNE database by Verband der Vereine Creditreform e.V., a group offering business information and debt collection services, was used [110]. It contains about 250,000 financial statements of mostly German and some Austrian enterprises. From these financial statements we calculate financial ratios which are used as features for the learning algorithms.

However, the data of most statements is incomplete, i.e., not all financial ratios are available. Thus, many instances were excluded from the case study or repaired using a replacement strategy.

Another problem is that there are only very few insolvent enterprises in this database. There are usable financial statements of about 250 insolvent enterprises which is only about 0.1% of the whole data. However, in machine learning it is important to have balanced data sets to prevent trivial models which would classify all enterprises as solvent. Although the error of this model would only be 0.1%, it would not perform its duty, i.e., actually

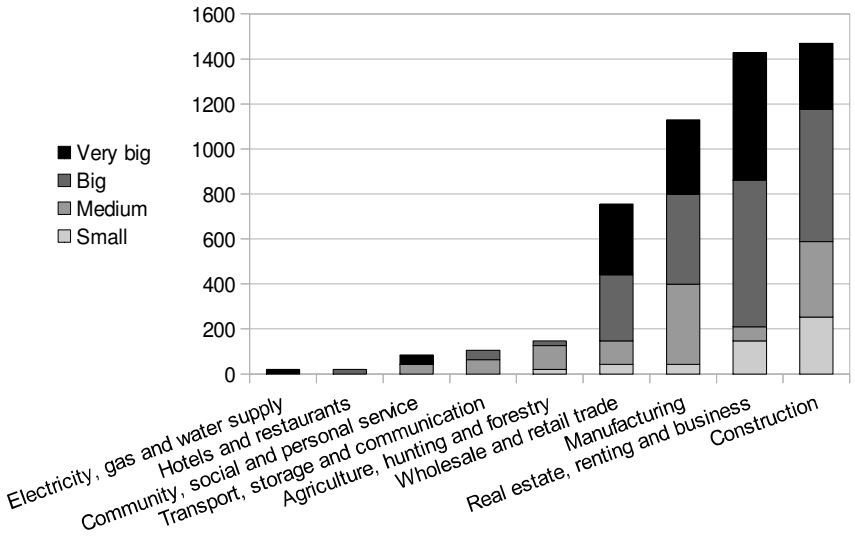


Figure 6.1.: The amount of enterprises for each industry and size as they appear in our database.

predict insolvencies. There are some ways to prevent this, e.g., by adjusting the weight of the smaller class [23, 54]. We tried this but for such a big imbalance it did not work, especially, for ANNs and SVMs. For ANNs, a weight of 1 for the negative data always yielded positive classifications. With the increase of this value the classification changed only slightly, but at some point completely tilted to the opposite side and classified all data as negative. SVMs always achieved balanced but very poor error rates. DNFs and RFs performed better. However, these improved results were not satisfying as well.

To solve this problem, we used asymmetric bagging with undersampling only on the solvent data and with a matching of enterprise size and industry. In the following, we refer to enterprises which are of the same size and industry as a group. Enterprise size is an attribute in DAFNE. Its calculation is not documented, but it has the advantage that it is available for each enterprise. Hence, there are no features needed that might be missing for some enterprises[8]. Industry classes are determined by WZ 2003, a measure of the German Federal Statistical Office [34]. The database provides data for 36 different industry classes and four different enterprise sizes. The distribution of these groups is shown in Figure 6.1 with detailed numbers in Table 6.1. For reasons of clarity, only nine super classes of 36 industry classes are shown.

[8] Among others, the number of employees is a feature commonly used for determining the size of an enterprise. Since this feature often is missing in DAFNE, we could not have grouped these enterprises and would have had to discard them.

Industry	Small	Medium	Big	Very big	Total
Electricity, gas and water supply	0	0	0	21	21
Hotels and restaurants	0	0	21	0	21
Community, social and personal service	0	42	0	42	84
Transport, storage and communication	0	63	42	0	105
Agriculture, hunting and forestry	21	105	21	0	147
Wholesale and retail trade	42	105	294	315	756
Manufacturing	42	357	399	332	1130
Real estate, renting and business	147	63	651	567	1428
Construction	252	336	588	294	1470
All industries					5162

Table 6.1.: The number of enterprises for each industry and size as they appear in our database.

6.1.2. Financial Ratios

In this case study, we used 18 financial ratios as features. This is an adaptation of the ratio system defined by Uthoff [107] which has been further developed by Brodag [19]. We denote the features directly available in DAFNE as base features. They are used to calculate these 18 financial ratios.

Since many of these financial ratios are similar, we performed a correlation analysis to reduce them. The Pearson product-moment correlation coefficient was calculated pairwise for each financial ratio. These results are shown in Table 6.2. Based on this table, we clustered the financial ratios. We iterated through all financial ratios row by row. If this financial ratio was not in a cluster, a new cluster was created. It consisted of this financial ratio and all other financial ratios, which were not already assigned to a cluster, with a correlation coefficient of above 0.4. These values are in bold in Table 6.2. If this financial ratio was already assigned to a cluster, it was skipped. This way, we obtained nine clusters of similar ratios, from which we have choose one representative. For the rest of this study we only used these representatives as features. Thus, we have reduced the dimension from 18 to nine financial ratios. The financial ratios and clusters are shown in Table 6.3. Tests have shown that this dimension reduction does not significantly affect the accuracy but drastically decreases the run time of the algorithms.

We experimented with different threshold values for the correlation coefficients. The value 0.3 already assigns eight financial ratios to the first cluster which is almost half of the ratios. Compared to 0.4, the value 0.5 would additionally divide a cluster of size two into two single ratios. Therefore, we think that 0.4 is an appropriate threshold yielding an appropriate amount of clusters with an appropriate size.

		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	Net income / total assets	**0.74**	0.00	0.00	0.21	0.21	0.00	0.03	**0.56**	0.00	0.37	0.39	0.20	0.03	0.01	0.21	0.03	**0.51**
2	Cash flow		0.00	0.00	0.08	0.08	0.00	0.02	**0.57**	0.00	0.31	0.77	0.12	0.02	0.01	0.08	0.03	**0.54**
3	Current ratio			0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.02	0.00
4	Return on equity				0.00	0.00	**0.82**	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
5	Return on assets					**0.99**	0.00	0.04	0.00	0.00	0.18	0.03	**0.52**	0.04	0.01	**0.68**	0.01	0.01
6	Return on investment						0.00	0.06	0.00	0.00	0.18	0.03	**0.52**	0.06	0.02	**0.69**	0.02	0.01
7	Debt-to-equity ratio							0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
8	Equity ratio								0.01	0.01	0.01	0.01	0.07	**1.00**	0.06	0.08	0.01	0.01
9	Working capital									0.00	0.18	0.35	0.00	0.01	0.08	0.00	0.01	**0.89**
10	Dynamic debt-to-equity ratio										0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
11	EBIT											**0.68**	0.17	0.01	0.00	0.19	0.02	0.17
12	EBITDA												0.06	0.01	0.01	0.04	0.02	0.31
13	Cash flow / total liabilities													0.07	0.01	**0.46**	0.00	0.00
14	Debt ratio														0.06	0.08	0.01	0.01
15	Working capital / total assets															0.03	0.00	0.01
16	Net income / total assets																0.02	0.00
17	Cash flow / current liabilities																	0.01
18	Net working capital																	

Table 6.2.: Correlation coefficients for all pairs of the 18 financial ratios used in this case study. High correlations, where ratios are thrown together into a cluster, are bold printed.

Net income
Cash flow
Working capital
Net working capital
Current ratio
Return on equity (RoE)
Debt-to-equity ratio
Return on assets
Return on investment (RoI)
Net income / total assets
Cash flow / total liabilities
Equity ratio
Debt ratio
Dynamic debt-to-equity ratio (Dyn. D/E)
EBIT
EBITDA
Working capital / total assets (WC / TA)
Cash flow / current liabilities (Cash flow / CL)

Table 6.3.: All 18 financial ratios from Brodag [19] reduced to nine clusters with representatives in bold.

6.1.3. Replacement and Selection

This section describes the preprocessing of our data. At first, insolvent instances missing more than 50% of the base features were discarded, thus, 246 instances were left. Due to the big amount of solvent data, we are in the postion to choose only solvent data with no missing base features. Subsequently, the enterprises were grouped by industry and size as described in Section 6.1.1 and depicted in Figure 6.2. Twenty or less solvent statements, depending on the availability, were randomly added from the same group for each insolvent statement. There are 4,916 out of 4,920 solvent data sets available, which totals to 5,162 instances together with the insolvent ones.

For the incomplete insolvent instances, the missing base feature values were replaced by the median of the base feature of their group. This replacement methodology has already been successfully used [19, 104]. Afterwards, the financial ratios were calculated from the base features. The whole replacement and selection process for a group is visualized in Figure 6.3.

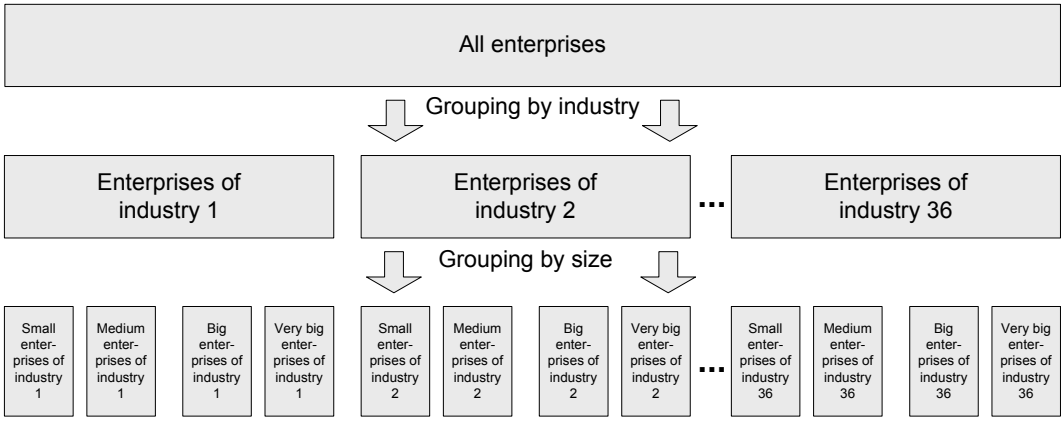
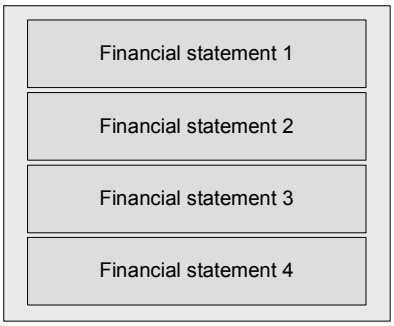
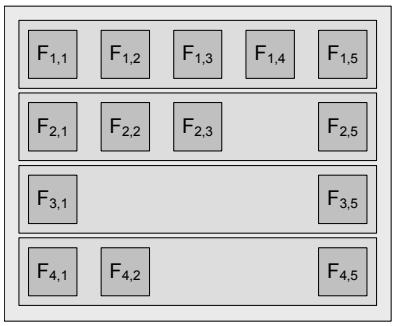


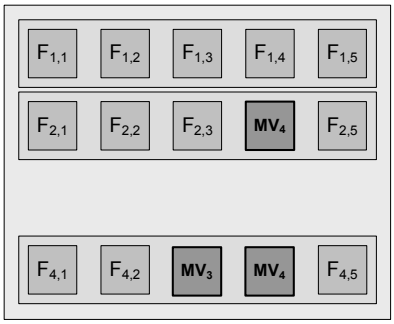
Figure 6.2.: Grouping of enterprises by industry and size.



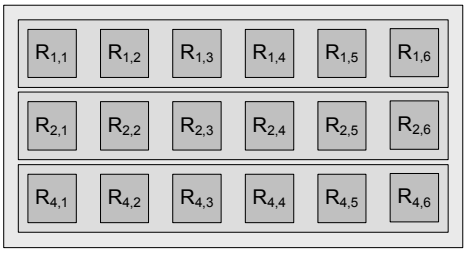
(a) 1. Financial statements of insolvent enterprises.



(b) 2. Base features (F) may be missing.



(c) 3. Replacement of missing base features with the median value (MV) of the according group or deletion of the whole financial statement.



(d) 4. Calculation of financial ratios from the replaced base features.

Figure 6.3.: Example of the treatment of instances with missing values. Replacements with the median value of this base feature are marked MV.

6.1.4. Experiments

In this section, we describe the experimental setting of our first case study. The experiments were performed for the models

- Decision Trees (DTs) using C4.5,
- Disjunctive Normal Forms (DNFs) using Thresholder,
- Random Forests (RFs),
- Artificial Neural Networks (ANNs) using RBF-networks, and
- Support Vector Machines (SVMs) using the RBF-kernel

and the meta-algorithm

- Asymmetric bagging.

Details are provided in the following sections.

6.1.4.1. Performance Measure

It is a difficult task to find one reasonable performance measure for ranking imbalanced data sets where one class is more important than the other, as in our case study. Due to our selection, we have 20 times as many solvent instances. However, they are less important than the insolvent instances. Due to this imbalance, taking accuracy as the performance measure would lead to nearly the same value as the TP rate. Therefore, accuracy or error rate are useless measures in this scenario. However, MCC and F-score consider imbalances, but not the weighting of the different classes. Furthermore, they favor the less important solvent data [54].

Altman et al. [6] introduced a measure called ZETA-score which weighs α- and β-error rates according to probability of occurrence and costs. Since this paper is from 1977 and does not provide parameters that are up-to-date, we cannot apply this measure. However, Altman et al. stated, and his data gives evidence, that the two pairs of parameters more or less neutralize each other[9]. Thus, we just take the mean of both error rates

$$\overline{\text{err}} = \frac{\alpha\text{-err} + \beta\text{-err}}{2}.$$

Since insolvent enterprises are the negative instances and solvent enterprises are the positive ones, α-errors are the errors on insolvent data and β-errors are the errors on solvent data.

[9] High insolvency costs (0.7) and low insolvency occurrence (0.02) in contrast to low opportunity costs (0.02) for solvent enterprises with a high occurrence (0.98) nearly neutralize each other.

In order to show whether there are significant differences between the mean error rates of the models, we did 20 repetitions of the experiments. Then, we applied two commonly used statistical tests; Welch's t test [115] and the Wilcoxon signed-rank test [116] for pairwise error comparisons. We used the two-tailed test which means testing the null hypothesis, so that the two population means are equal. It was rejected at a significance value of 0.01. We tested the significance of the differences of mean errors between all models.

6.1.4.2. Training Process

We examined each model using asymmetric bagging with 100 bagging iterations. In each bagging iteration we used a 10-fold cross-validation for estimating the performance. The cross validation was used on the imbalanced data set. The balancing through the asymmetric bagging method was applied on each training set of the cross-validation. Thus, the test set of the cross-validation remains imbalanced for the performance measure.

6.1.4.3. Evaluation

All models listed in Section 6.1.4 are trained using asymmetric bagging. The idea behind this is to have a balanced data set and to make use of the excess of solvent data. Based on this idea, we take the insolvent data and then train multiple classifiers with different, randomly drawn solvent data using the asymmetric bootstrapping method. Asymmetric bagging merges them to a single classifier. This allows for the training on a balanced data set while still using the information of all[10] solvent data. While the results may fluctuate for a low number of bagging iterations, they converge with a high number of iterations.

A second evaluation was performed to measure the impact of the asymmetric bagging approach. As mentioned above, using imbalanced data sets yielded bad results. Therefore, we trained our models only using asymmetric bootstrapping without using a majority vote and compared it with asymmetric bagging.

Each algorithm has different parameters, whose values determine its performance. Previous tests have shown that standard parameters are not always the best choice. Therefore, we identified important parameters where we used different values. The remaining parameters use standard values. Based on some experiments, we broke the important parameters down into two per algorithm and three values per parameter which results in a set of nine parameter value combinations. For Thresholder, only one important parameter with four different values is needed. To select parameter value combinations from these sets, a 10-fold cross-validation loop was used. The parameter sets of the white-box algorithms are chosen in a way that the models have an average size of about 14 threshold indicators or decision nodes. Thus, we can guarantee that the size of the model is small and therefore stays interpretable. The parameter sets used are listed in Table 6.4.

[10] The probability that each solvent enterprise will be drawn at some point converges with the number of random draws.

Model	Parameters and values used
DNF	Max. number of literals $p = 4$ Max. number of clauses $r = 4$ Prune at most $p_c = 2$ literals at once Prune only if error worsens by at most $p_e = \{0; 0.005; 0.01; 0.02\}$
DT	Confidence factor C used for pruning and min. number of instances per leaf M $(C,M) \in \{(0.0005,3);(0.001,3);(0.005,3);(0.001,6);$ $(0.005,6);(0.01,6);(0.005,12);(0.01,12);(0.05,12)\}$
RF	Number of trees $I = \{4;7;10\}$ Max. depth of the trees $d = \{2;3;4\}$
ANN	Number of RBF-functions $B = \{2;10;20\}$ Min. width of RBF-functions $W = \{0.01;0.1;1\}$
SVM	γ in RBF-function $G = \{0.01;0.1;1\}$ Complexity parameter $C = \{10;100;1000\}$ Normalize data

Table 6.4.: The parameter sets used to build the models. For each algorithm all parameter combinations of these values are used.

6.1.5. Results

We performed the experiments as described above. The corresponding results are described below. In tables and figures, the abbreviation err refers to the error rate.

6.1.5.1. Comparison of the Performance of Interpretable and Non-interpretable Models

Table 6.5 shows the results of all models. First of all, it needs to be mentioned that there are still very few fluctuations on the error rate caused by the random process of drawing the training sample. Thus, we performed a significance test as described in Section 6.1.4.1. According to the test, all differences are significant. A graphical representation of the deviation of the mean error can be found in Figure 6.4.

Concerning the error rates, we are interested in how well white-box models compare to other models. It can be seen that DTs and DNFs perform best. DTs have a smaller β-error rate, in turn DNFs have a smaller α-error rate. They are followed by the gray-box models. SVMs are the worst performing models in this study.

6.1.5.2. Performance of Asymmetric Bagging

The results of the second evaluation comparing the use of asymmetric bagging to training without bagging are shown in Table 6.6 and visualized in Figure 6.5. An improvement of the error rate for each algorithm except for RFs can be observed.

Model		$\overline{\text{err}} \pm$ SD	α-err	β-err	TP FP	FN TN
DT	□	24.2 ± 0.0033	34.4	14.0	4228 85	688 161
DNF	□	24.8 ± 0.0038	32.5	17.2	4070 80	846 166
RF	■	26.1 ± 0.0022	39.4	12.7	4292 97	624 149
ANN	■	27.0 ± 0.0023	50.0	3.9	4724 123	192 123
SVM	■	29.1 ± 0.0027	42.6	15.5	4154 105	762 141

Table 6.5.: Ranking of all models determined by the cross-validation error rates in percent. $\overline{\text{Err}}$ is the mean value of α- and β-error rate. The square shows the interpretability of the model as white, gray, or black-box.

Model	No bagging			Asymmetric bagging		
	$\overline{\text{err}}$	α-err	β-err	$\overline{\text{err}}$	α-err	β-err
DT	26.4	33.5	19.3	24.2	34.4	14.0
DNF	28.1	33.7	22.5	24.8	32.4	17.2
RF	26.1	37.2	14.9	26.1	39.4	12.7
ANN	28.4	51.4	5.4	27.0	50.0	3.9
SVM	31.9	45.7	18.1	29.1	42.6	15.5

Table 6.6.: Comparison of the cross-validation error rates in percent using asymmetric bagging with the error rates using no bagging. $\overline{\text{Err}}$ is the mean value of α- and β-error rate.

6.1.6. Discussion

This section discusses the results and is divided into a comparison of interpretable models with the non-interpretable ones, an evaluation of the benefit of asymmetric bagging, and a comparison of the interpretability of DTs and DNFs.

6.1.6.1. Comparison of the Performance of Interpretable and Non-interpretable Models

In our case study the white-box models perform best, followed by gray-box and black-box models. Furthermore, the model size of the white-box models is limited to ensure a good interpretability. This is a desirable result because interpretability and a low error rate are the most important factors. However, the results are empirical, therefore, we carefully interpret the result as non-inferiority of interpretable models. An interesting outcome is that RFs, which have a lot more rules than DTs and DNFs, actually perform worse.

Moreover, Table 6.5 shows the Standard Deviation (SD) of error rates over 20 repetitions. It can be seen that the SD for the white-box models is a little bit bigger than for the other

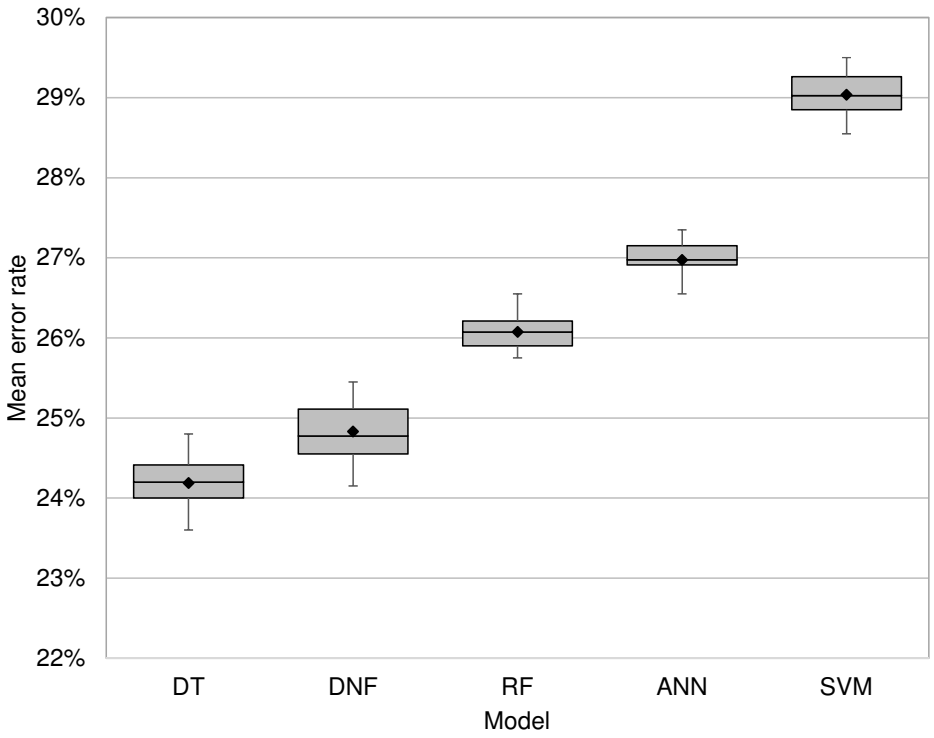


Figure 6.4.: Boxplot for the error rates and variation of all five models.

models in this case study. Nevertheless, the SD for all models is quite low, which explains why Welch's t test and the Wilcoxon signed-rank test both find statistical differences for all results.

Concerning a comparison to other approaches building interpretable models from originally not interpretable models, black-box rule extraction, white-box rule combination, or variable extraction either lose accuracy or do not provide a comparable level of interpretability. The models to which these approaches can be applied, already perform worse and are less interpretable. Applying these approaches will either decrease the performance further or yield a lower interpretability with an already lower performance.

Our results match the findings of other researchers [61, 113] (see Section 4.3.2) who also figured out that interpretable models are not inferior in finance classification problems. We could verify these findings on a large sample with two of the most interpretable models and an interpretable model size. Thus, we can conclude that interpretable models are useful for insolvency prediction and similar tasks in finance. A possible explanation can be found in the small amount of required features. Financial ratios are correlated somehow, so that there is no need for a bigger number. Most studies from Chapter 4 use only 15 or less financial ratios. It seems that few simple rules suffice for insolvency prediction. However, black-box

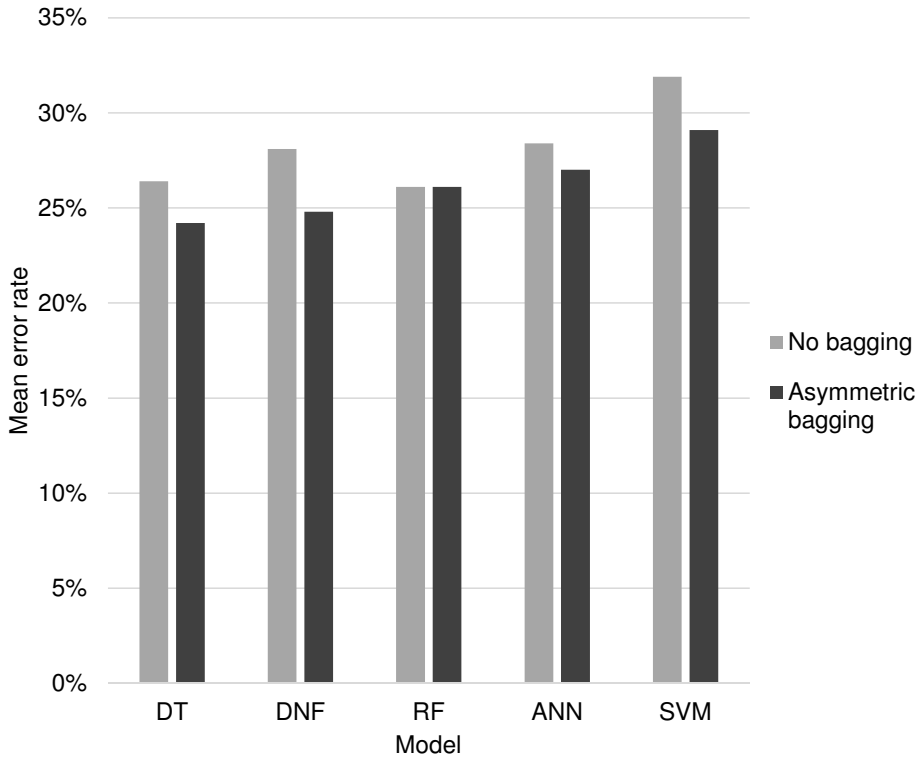


Figure 6.5.: Comparison of the mean error rates using asymmetric bagging with the mean error rates using no bagging.

algorithms like SVMs have a big advantage classifying high dimensional data [28].

6.1.6.2. Performance of Asymmetric Bagging

We successfully used asymmetric bagging to tackle the class imbalance problem. As mentioned above, directly learning this imbalanced data set worked out quite badly, especially, with ANNs and SVMs. Using asymmetric bootstrap samples, we solved the problem of classifying almost the whole data as positive. Moreover, asymmetric bagging further improved the overall error rate for each learning algorithm by about 1% to 3%. An exception are RFs which are not improved, probably because they already consist of bagged trees.

All β-error rates are much lower than their corresponding α-error rates. Asymmetric bagging further decreases all β-error rates while the α-error rates even increase in some cases.

Unfortunately, bagging does not preserve the interpretability of the interpretable models. Therefore, the overall classifier is non-interpretable. However, asymmetric bagging is primarily used to compensate for the imbalanced data set. We expect similar results for a

$$
\begin{aligned}
&\text{Solvent} = \\
&\quad \big((\text{Equity ratio} > 1{,}529.5\%) \wedge (\text{Dyn. D/E} \leq 649{,}000\%) \wedge \\
&\quad\ (\text{EBIT} > 9{,}490.5€) \quad\ \wedge (\text{WC / TA} > 0.031\%)\big) \\
&\vee \big((\text{Equity ratio} > 272.5\%) \quad \wedge (\text{Dyn. D/E} \leq 4{,}556{,}000\%) \wedge \\
&\quad\ (\text{EBIT} > 68{,}993.5€) \quad\ \wedge (\text{RoE} > -6.315\%)\big)
\end{aligned}
$$

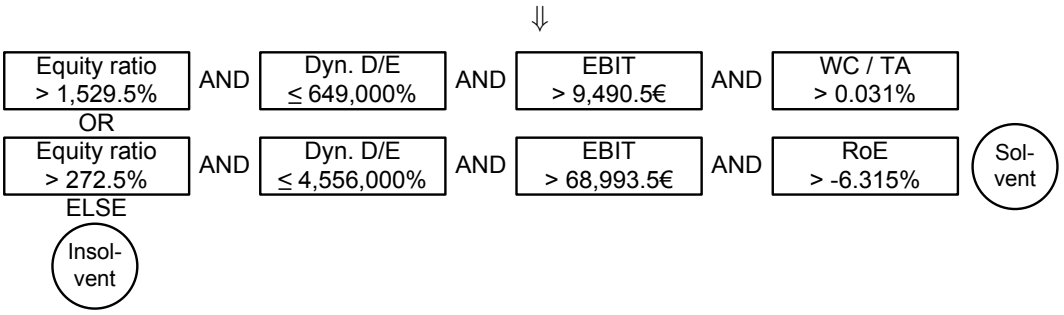


Figure 6.6.: Example model of a DNF in its mathematical form and a clearer representation for analysts. Abbreviations are taken from Table 6.3.

balanced data set without bagging. On this data set, the ranking of the models stays unchanged when using no bagging except for the RFs which take the lead.

6.1.6.3. Interpretability of DTs Compared to DNFs

Since DTs perform just as good as DNFs, the question arises which one should be used. To answer this question, we compare their interpretability with two example models. They were built during the experiments of this case study and are displayed in Figure 6.6 and 6.7.

The DNF output by Thresholder shows tighter thresholds for "Equity ratio" and "Dyn. D/E" in the first monomial which have to be fulfilled for a solvent enterprise. If these thresholds are less tight in the second monomial, we see that it can be compensated by a better "EBIT" value. Its focus is on the combination of financial ratios.

The DT first checks if "Equity ratio", "Cash flow / CL", and "EBIT" are good enough for a solvent enterprise and then branches according to "WC / TA". It focuses on the importance of financial ratios ranking them starting at the root. Notice that there are some decisions which at first sight do not appear plausible, marked with an *. Sometimes, they occur in the DNFs as well. The problem might either be the small data set, where artifacts have too much impact on the algorithm or complex circumstances which cannot be understood at first sight.

It can be seen that these models are both quite easy to read. However, the interpretations are different. Thus, an analyst should consider both models to achieve the best decision support. An additional advantage of these models and their algorithms is the ability to adjust

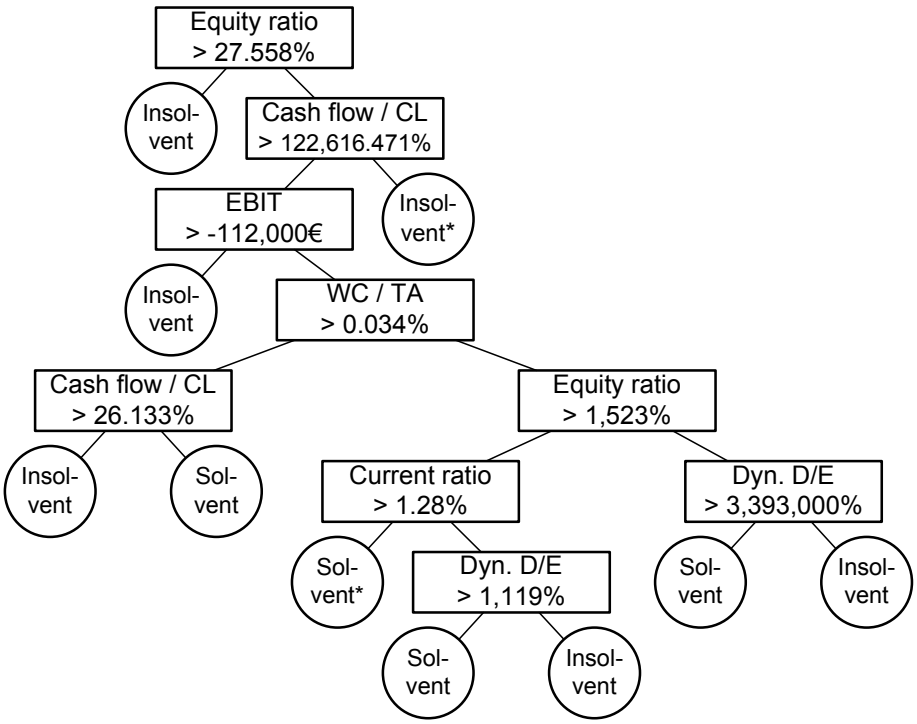


Figure 6.7.: Example model of a DT. Abbreviations are taken from Table 6.3.

the tradeoff between interpretability and accuracy by adjusting generalization parameters. Thus, analysts could obtain yet simpler and smaller models with a loss of accuracy.

6.2. Credit Rating

Like insolvency prediction, credit rating evaluates the solvency of enterprises. However, for this task, the labeling of the data is man-made.

Overall, our second case study faces different challenges. Our data sets for this study are more homogeneous and balanced. Therefore, the problem of dealing with imbalances and finding an adequate error measure fades into the background. The new challenges are predicting multiple classes and obtaining interpretable classifiers for this task. Furthermore, we experiment with interpretable IDK-classifications. This study compares different interpretable models in more detail than the previous study.

Our research went on between these two studies meaning that Thresholder evolved a bit since the first study (see Section 5.1.1). The resulting runtime improvements enable us to chose more computationally complex parameter settings.

6.2.1. Data

This case study is based on a newer version of the DAFNE database [111]. In our previous study on insolvency prediction [85], we worked with a much older version of this database with a big number of inhomogeneous enterprises of different industries, sizes, and years of annual accounts. This time we obtained more homogeneous data. Thus, we were able to work with a random selection of three separate data sets roughly containing 1000 to 1500 enterprises of the industries wholesale and retail trade, construction, and finance. Each data set contains mostly German and only very big enterprises[11] with annual accounts from 2013 divided into three rating classes. The highest rating matches Standard & Poor's *AAA* to *BB*+ rating classes, the medium rating matches *BB* to *B*, and the lowest rating matches *B*− to *D*. The idea was to find 500 enterprises for each industry and rating class, but since enterprises with the lowest rating are comparatively rare, this requirement could not be met. Details about the actual numbers can be found in Table 6.7.

Data set	Low	Medium	High	Total
Wholesale and retail trade	256	500	500	1256
Construction	361	500	500	1361
Finance	62	500	504	1066

Table 6.7.: Number of enterprises of each rating class as they appear in our data sets

6.2.2. Financial Ratios

The features of the data sets are directly taken or calculated from annual accounts, i.e., balance sheets and income statements. Many of these features contain missing values since only a few financial ratios are required to be published in an annual financial statement. We discarded all features that were not at least 90% complete. All features which were not used in our previous study and which are just single values from balance sheets or income statements were discarded as well. Performing this feature selection we ended up with nine financial ratios as shown in Table 6.8.

6.2.3. Replacement and Selection

Since these data sets are much more complete than in the previous case study, no instances had to be discarded. Furthermore, there is only a small imbalance between the classes. Therefore, no data balancing techniques were necessary.

However, missing values had to be replaced with some numerical values. Experiments with different replacement strategies have shown that missing values provide rating information. Enterprises with a low rating tend to have more missing values. Therefore, our

[11] Enterprise size is a feature in the database and its calculation is undocumented.

Revenue
Net income
Profit margin
Capital-debt ratio
Equity ratio
Cash flow
Current maturities
Return on equity (RoE)
Return on investment (RoI)

Table 6.8.: The nine financial ratios used in this study.

replacement strategy for missing values is using the value zero instead of mean, median or other estimators. This value isolates the information and can easily be recognized in the resulting model. Our data sets are randomly drawn subsets of bigger data sets of the Creditreform. Thus, the distribution of missing values should represent the distribution of the missing values of the bigger data sets. Since this distribution is not altered, missing values are a legitimate discrimination criterion. Note that in the previous case study, the data selection of solvent enterprises was partially based on missing values. Using information of missing values in the previous study would have been cheating.

6.2.4. Experiments

This section describes the experimental setting of this case study. The experiments were performed using the models

- Thresholder DNFs,
- RIPPER DNFs,
- C4.5 DTs,
- RBF-networks (ANNs),
- RFs,
- Linear SVMs (L-SVMs),
- RBF-kernel SVMs (R-SVMs), and
- Polynomial-kernel SVMs (P-SVMs),

the multiclass meta-algorithms

- One-vs-next,

- One-vs-followers,
- One-vs-one, and
- One-vs-rest,

and the meta-algorithm

- Boosting.

Details are provided in the following sections.

6.2.4.1. Performance Measure

Since we do not have big imbalances in our data anymore, we have chosen the error rate as a performance measure. According to the related literature this is a common measure for multiclass problems in finance.

For the error rate in case of IDK-classifications, a value of τ had to be chosen. We have chosen $\tau = 0.67$, i.e., the probability of guessing the wrong label in the three class case as discussed in Section 2.2.1.5.

Since the data is split randomly, repetitions slightly change results similar to the previous study. Therefore, we performed 20 repetitions as well and took the mean value of these results. Then we tested the statistical significance between the different methods using Welch's t test [115] and the Wilcoxon signed-rank test [116] for pairwise error and model size comparisons. We denoted results as significantly different, if the p-value was below 0.01.

6.2.4.2. Model Size Measure

There are different measures for the model size of DNFs and DTs. The size of a DT can be measured by the size of the tree which is the number of nodes (threshold indicators) and the number of leaves (output values). This measure is used in WEKA as well [50]. Possible measures for the DNF size are the number of rules as used in WEKA [50] and the total number of threshold indicators [47][12]. Since the first measure only counts the number of monomials and does not take the size of the monomials into account at all, we consider the second measure to be more appropriate to quantify model size and interpretability. However, simply using the latter measure for cascaded DNFs would result in an unfair advantage over DTs. Hence, we have to add the number of output values which is the number of single DNFs $+1$.

[12] Technically they use the number of rules multiplied by the mean rule size.

6.2.4.3. Training Process

The algorithms from Section 2.2.2.1 were evaluated using the standard procedure which splits the data randomly into a 67% training and 33% test split. Since standard parameters are not always the best choice, a three-fold cross-validation was applied to the training set for the parameter selection of all algorithms.

6.2.4.4. Evaluation

We evaluated all learning algorithms of Section 2.2.2.1 in combination with all multiclass methods from Section 2.2.3.1. The one-vs-one and one-vs-rest methods were applied in two different ways. Thresholder used the indirect and direct method (Sections 5.1.2.3 and 5.1.2.4). All other algorithms used the normal method with majority votes and a modified method with probability estimates as votes. To test whether the learning algorithms can be further improved by ensemble learning, we exemplarily tested the one-vs-one method with boosted classifiers using AdaBoost with 10 boosting iterations.

We experimented with different parameter values for the algorithms to find an ideal and fair setting for each of them to represent their performance. This resulted in sets of parameter values where the final setting is selected using a three-fold-cross-validation. For details of the parameter sets see Table 6.9. Parameters of interpretable models were chosen only based on their performance regardless of model size.

In a second experiment we tried different generalization parameters for all interpretable models of the all-at-once multiclass method. We started with the parameter settings of the previous experiments and changed the values stepwise in a way that the model size of the last parameter combination was five or lower. This is the smallest model size which allows for a separation of three classes. For details see Table 6.10. That way, we can evaluate a model's performance compared to its size. Lowering the model size is desirable, because a lower model size increases interpretability.

A third experiment was performed to study the behaviour of IDK-classifications. We tested all algorithms in combination with all multiclass methods supporting IDK-classification, i.e., one-vs-one and one-vs-rest. Additionally, we evaluated Thresholder using different values of τ. Although we tried different values of τ in the training process, τ is always fixed to 0.67 when testing the model. This allows for a proper evaluation. Since we are more interested in the performance using IDK-classifications and model size is of second rank, we use the parameters from Table 6.9.

6.2.5. Results

The experiments were performed as described above. Welch's t test and the Wilcoxon signed-rank test both yielded almost always the same results for our experiments. If they disagreed, result were denoted as significantly different.

Algorithm	Parameters and values used
Thresholder	Max. number of literals $p = 5$ Max. number of clauses $r = 5$ Max. number of thresholds in a single DNF $s_{single} = 7$ Prune at most $p_c = 4$ literals at once Prune only if error worsens by at most $p_e = 0.001$ Max. number of thresholds in the cascaded DNF $s = \{2;3;4;6;8;10\}$
C4.5	Confidence factor used for pruning $C = \{0.005;0.01;0.02\}$ Min. number of instances per leaf $M = \{10;20;50\}$
RIPPER	Number of folds for growing/pruning $F = \{2;3;4\}$ Min. number of instances per rule $N = \{1;2;4\}$ Number of optimization runs $O = 2$
RF	Number of trees $I = \{4;6;10\}$ Max. depth of the trees $d = \{2;5;10\}$
ANN	Number of RBF-functions $B = \{10;20;30\}$ Min. width of RBF-functions $W = \{1;10;100\}$
L-SVM	Complexity parameter $C = \{100;500;1000;5000;10000\}$ Normalize data
P-SVM	Exponent in polynomial function $E = \{2;3;4\}$ Complexity parameter $C = \{10;100;1000\}$ Normalize data
R-SVM	γ in RBF-function $G = \{0.01;0.1;1\}$ Complexity parameter $C = \{10;100;1000\}$ Normalize data

Table 6.9.: Parameter sets used for learning algorithms. For each algorithm all parameter combinations of these values are used.

Parameter set	Thresholder	C4.5	RIPPER
P_1	$s = \{2;3;4;6;8;10\}$	$M = \{10;20;50\}$	$N = \{1;2;4\}$
P_2	$s = \{2;3;4;6;8\}$	$M = 20$	$N = 5$
P_3	$s = \{2;3;4;6\}$	$M = 30$	$N = 10$
P_4	$s = \{2;3;4\}$	$M = 40$	$N = 20$
P_5	$s = \{2;3\}$	$M = 50$	$N = 50$
P_6	$s = \{2\}$	$M = 60$	$N = 100$

Table 6.10.: The six different generalization parameter sets used for evaluating the correlation between classification error and model size.

In the following tables, the abbreviation err refers to the error rate and size to the model size.

6.2.5.1. Performance of All Models and Methods

Table 6.11 shows the error rates for all three data sets, learning algorithms, and multiclass methods. Figure 6.8 visualizes the mean error rates for the three data sets and shows that all algorithms for interpretable models and RFs perform similarly well and are much better than other algorithms. The visualization of the error rates for each single data set can be found in Figure A.1 - A.3. All best performing algorithms are threshold-based. Different multiclass methods do only affect the performance of the non-threshold-based algorithms. Boosting does not significantly increase performance. In fact, in most cases it tends to overfit and slightly decreases it.

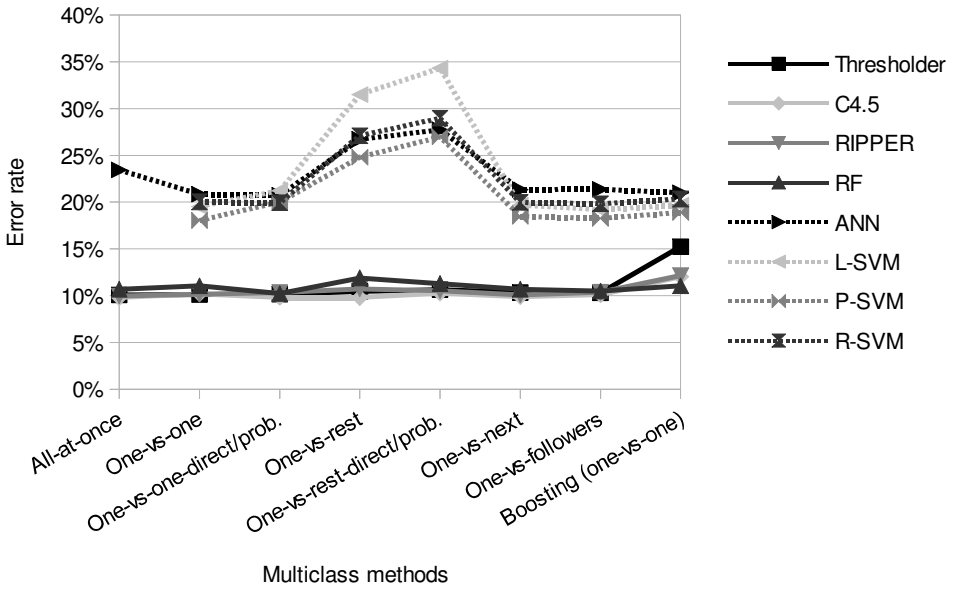


Figure 6.8.: Mean error rates of learning algorithms and multiclass methods for all data sets.

6.2.5.2. Performance of Interpretable Models Compared Among Themselves

Multiclass methods may break the interpretability of the model. Therefore, Table 6.12 shows the error rates, their significances, and model sizes only of interpretable multiclass models. Figure 6.9 visualizes the results of the interpretable methods by plotting the mean error rates and model sizes for the three data sets. The visualization of the error rates and models sizes for each single data set can be found in Figure A.4 - A.6. As mentioned above, the performances are comparable, while the model sizes change. Since we focused only on performance when choosing parameters for the algorithms, it might not be fair to compare these model sizes at this point. Therefore, we performed an additional experiment described in the following section.

Algorithm	All -at- once	One -vs- one	One -vs- one direct/prob.	One -vs- rest	One -vs- rest direct/prob.	One -vs- next	One -vs- followers	Boosting (one -vs- one)
Thresholder	4.8	5.0	5.1	5.0	5.2	4.6	4.9	5.8
C4.5	4.8	5.2	5.5	5.2	5.0	5.4	5.4	5.3
RIPPER	4.7	4.2^{+}	4.2^{+}	5.1	5.0	4.7	4.5	4.9
RF	4.9	4.9	4.2^{+}	4.9	4.8	5.0	4.6	5.1
ANN*	19.2	11.6	12.0	16.1	18.2	11.5	13.4	11.7
L-SVM*		7.9	10.5	20.3	28.4	8.2	7.5	8.2
P-SVM*		8.1	13.1	18.1	22.6	8.4	8.1	9.6
R-SVM*		13.0	12.9	22.4	25.4	12.0	13.0	11.9

(a) Trade data set.

Algorithm	All -at- once	One -vs- one	One -vs- one direct/prob.	One -vs- rest	One -vs- rest direct/prob.	One -vs- next	One -vs- followers	Boosting (one -vs- one)
Thresholder	8.8	8.8	8.4	9.4	9.2	9.2	9.0	11.5
C4.5	8.6	8.9	8.2^{+}	8.2^{+}	8.8	8.4	8.3	10.4
RIPPER	8.4	8.7	8.9	9.9	9.7	8.5	8.7	11.1
RF	8.8	10.0	9.1	10.6	10.3	9.5	9.5	9.6
ANN*	18.5	18.2	18.0	27.9	28.5	19.1	17.8	18.1
L-SVM*		13.8	16.1	37.1	33.4	14.3	14.3	14.5
P-SVM*		15.4	17.8	25.5	26.4	16.9	16.5	17.2
R-SVM*		16.2	17.0	24.5	26.2	15.6	15.5	16.5

(b) Construction data set.

Algorithm	All -at- once	One -vs- one	One -vs- one direct/prob.	One -vs- rest	One -vs- rest direct/prob.	One -vs- next	One -vs- followers	Boosting (one -vs- one)
Thresholder	16.6	16.6	17.0	17.0	17.5	17.2	17.0	28.4
C4.5	16.0	16.5	15.8^{+}	16.0	17.0	15.9	16.6	20.4
RIPPER	16.8	17.6	17.9	17.2	16.9	17.0	17.9	20.4
RF*	18.4	18.3	17.4	20.2	18.8	17.6	17.4	18.5
ANN*	32.6	32.7	32.2	36.1	36.5	33.4	32.9	33.2
L-SVM*		36.9	37.1	37.1	41.1	36.5	35.8	36.3
P-SVM*		30.6	29.1	30.8	31.9	30.1	30.2	29.9
R-SVM*		30.9	29.9	34.4	35.3	32.3	31.0	32.6

(c) Finance data set.

Table 6.11.: Error rates of learning algorithms and multiclass methods in percent. There are marks, if the best result for an algorithm (*) is significantly worse than the overall best result ($^{+}$).

Algorithm		Trade		Construction		Finance	
		err	size	err	size	err	size
Thresholder	All-at-once	4.8	7.6	8.8	8.8	16.6	5.9
	One-vs-one	5.0	7.4	8.8	8.4	16.6	5.1
	One-vs-one-direct	5.1	7.4	8.4^{+}	7.9	17.0	6.6
	One-vs-rest	5.0	8.7	9.4^{*}	8.0	17.0	5.6
	One-vs-rest-direct	5.2	8.0	9.2	8.0	17.5	5.9
	One-vs-next	4.6	8.6	9.2	8.9	17.2	6.4
	One-vs-followers	4.9	8.2	9.0	8.7	17.0	6.4
RIPPER	All-at-once	4.7	8.9	8.4^{+}	9.9	16.8	9.3
	One-vs-next	4.7	9.1	8.5	8.9	17.0	7.8
	One-vs-followers	4.5^{+}	8.0	8.7	9.7	17.9^{*}	8.0
C4.5	All-at-once	4.8	7.6	8.6	9.0	16.0^{+}	6.8

Table 6.12.: Comparison of error rate in percent and model size of all interpretable multiclass models in this study. Results with an * are significantly worse than the best interpretable result (marked with a +) for this data set.

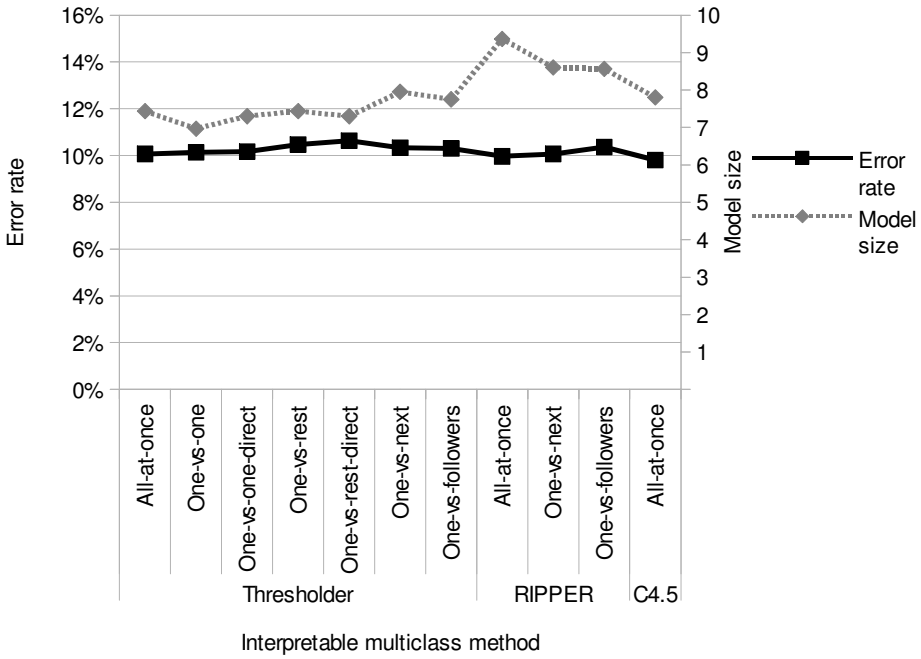


Figure 6.9.: Comparison of mean error rate and mean model size for all data sets of all interpretable multiclass models in this study.

6.2.5.3. Model Sizes of Interpretable Models

To obtain the required model size for a comparable performance, we examined performance and model size for different parameters. Figure 6.10 and Table 6.13 show the connection of performance and model size when the model size is decreased. They denote statistical significances between error rates and model sizes as well.

Generalization	Thresholder		C4.5		RIPPER	
parameter set	err	size	err	size	err	size
P_1	4.8	7.6	4.8	7.6	4.7	8.9
P_2	4.6	7.8	5.3^*	5.8	4.6	8.5
P_3	4.9	7.3	5.4^*	5.0	4.7	8.0
P_4	4.9	6.7^+	5.4^*	5.0	5.9^*	5.4
P_5	$5.3*$	5.4	5.4^*	5.0	5.9^*	5.4
P_6	5.5^*	5.0	5.4^*	5.0	5.7^*	5.1

(a) Trade data set.

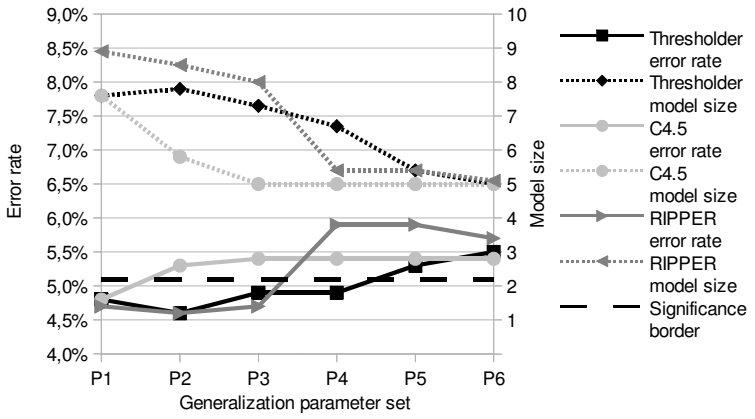
Generalization	Thresholder		C4.5		RIPPER	
parameter set	err	size	err	size	err	size
P_1	8.8	8.8	8.6	9.0	8.4	9.9
P_2	8.7	8.0	8.9	8.8	8.3	9.4
P_3	8.6	7.4	10.5^*	7.2	8.4	9.1
P_4	9.0	6.9^+	11.0^*	5.0	10.4^*	7.1
P_5	10.9^*	5.3	11.0^*	5.0	10.2^*	5.0
P_6	10.8^*	5.0	11.0^*	5.0	10.2^*	5.0

(b) Construction data set.

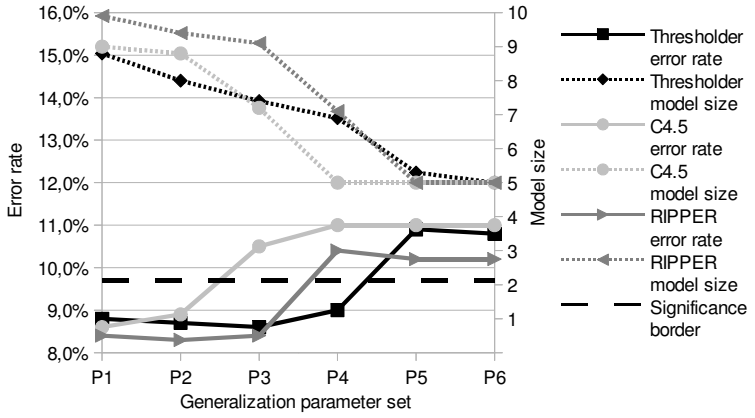
Generalization	Thresholder		C4.5		RIPPER	
parameter set	err	size	err	size	err	size
P_1	16.6	5.9	16.0	6.8	16.8	9.3
P_2	16.6	5.6	16.5	6.0	16.7	9.5
P_3	16.9	5.3	16.5	5.0	17.0	8.9
P_4	16.8	5.1	16.5	5.0	15.9	5.4
P_5	16.9	4.5	16.5	5.0	17.5	4.3
P_6	16.9	4.0	16.9	5.0	17.9	3.7^+

(c) Finance data set.

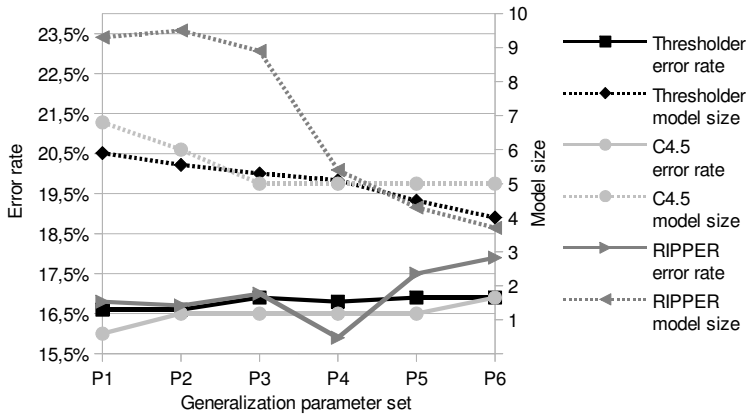
Table 6.13.: Error rate in percent and model size for different generalization parameters. Results with an * are significantly worse than the best interpretable result (marked with a $^+$ in Table 6.12) for this data set. Horizontal lines separate significant from non-significant results. All smallest model sizes which are not significantly different from each other with an error not significantly worse than the best error for this data set are marked with a $^+$.



(a) Trade data set.



(b) Construction data set.



(c) Finance data set. Since there are no results which are significantly worse than the best result, no border is plotted.

Figure 6.10.: Error rate and model size for different generalization parameters. A border is plotted between the results which are not significantly worse and the results which are significantly worse than the best result.

6.2.5.4. Performance of IDK-classifiers

Table 6.14 shows the results obtained using IDK-classifications. Results which were significantly worse than the best interpretable result (marked with a $^+$ in Table 6.12) or with less than one assigned IDK-label were intentionally left out, because we could not derive benefit from them. However, the whole results can be found in Tables A.1 - A.3. The tables show the error rate, the number of IDK-assignments, and die proportion of IDK-assignments per misclassification.

6.2.6. Discussion

This section discusses the results of the second case study and is divided into a comparison of all models and methods, all interpretable models, model sizes, and IDK-classifiers.

6.2.6.1. Performance of All Models and Methods

Our case study shows that interpretable algorithms perform best. Moreover, all of them have a similar classification error. RFs perform slightly worse only for the finance data set. All other non-interpretable algorithms perform significantly worse. Thus, the best performing algorithms are all threshold-based. There is quite a big gap between the error rates of the threshold-based algorithms and the rest, as can be seen in Figure 6.8. The figure shows that the mean gap over the three data sets is almost always about 10% or higher for each multiclass method which means about twice as many misclassifications. Despite the fact that this figure only plots the mean values for all data sets, it reflects the relative results of each data set as well. Different multiclass methods do not influence the performance of threshold-based algorithms to a great extend. Nevertheless, for ANNs and SVMs, there is a big performance drop when using the one-vs-rest method which was observed in other studies [49, 57, 67] as well. The remaining multiclass methods only show marginal differences among each other. For these data sets, using methods with probability estimates is almost always worse than using methods with simple votes. When ignoring the badly performing one-vs-rest methods, SVMs perform better than ANNs as observed by Kim et al. [67]. The simple L-SVMs perform slightly better than the more sophisticated P-SVMs and R-SVMs on the trade and construction data set. The ensemble learning method boosting slightly increases the performance only of some of the bad performing non-interpretable algorithms. However, the best performances are achieved by the threshold-based algorithms and boosting does not increase it. This and the fact that they all perform similarly well leads to the conclusion that the remaining error is noise which can only be eliminated using additional information.

We were surprised at the bad performance of the popular SVMs and ANNs. Therefore, we experimented with different parameters and different kernels in this study. Furthermore, we tried a feature selection, but to no avail. There are very few studies comparing these methods with threshold-based models in the field of multiclass credit rating. Furthermore,

Learning algorithm	Multiclass method	τ	#IDK	#IDK / #err	$\text{err}_{\tau=0.67}$
Thresholder	All-at-once	0.67	2.8	15	4.8
		0.5	2.9	15	5.0
		0.3	4.3	23	5.2
	One-vs-one	0.3	2.7	14	5.0
	One-vs-one-direct	0.67	1.7	8	5.0
		0.5	1.9	10	4.9
		0.3	3.5	18	5.2
	One-vs-rest-direct	0.3	2.7	14	5.1
RF	One-vs-rest		5.0	30	4.8

(a) Trade data set.

Learning algorithm	Multiclass method	τ	#IDK	#IDK / #err	$\text{err}_{\tau=0.67}$
Thresholder	All-at-once	0.67	3.2	8	9.3
		0.5	5.4	15	8.9
	One-vs-one-direct	0.67	2.4	6	9.1
		0.5	3.8	10	9.1
		0.3	5.6	14	9.5
	One-vs-rest	0.67	3.6	9	9.5
		0.5	4.3	11	9.5
	One-vs-rest-direct	0.67	4.9	12	9.5
		0.5	4.2	10	9.6
C4.5	One-vs-rest		2.3	6	8.3

(b) Construction data set.

Learning algorithm	Multiclass method	τ	#IDK	#IDK / #err	$\text{err}_{\tau=0.67}$
Thresholder	All-at-once	0.67	10.9	19	18.0
		0.5	10.7	19	17.9
		0.3	8.6	16	17.4
	One-vs-one	0.67	4.6	8	17.4
		0.5	3.4	6	17.1
		0.3	4.2	7	16.6
	One-vs-one-direct	0.67	7.1	12	18.3
		0.5	9.3	16	18.0
	One-vs-rest	0.67	2.0	3	17.8
		0.5	6.6	11	18.4
		0.3	7.7	13	18.4
	One-vs-rest-direct	0.67	4.8	8	18.2
		0.5	8.3	14	18.0
		0.3	6.8	12	17.9
C4.5	One-vs-rest		5.0	10	15.7
RIPPER	One-vs-rest		14.1	27	17.3
RF	One-vs-one		1.1	2	18.2

(c) Finance data set.

Table 6.14.: Absolute number of IDK-assignments and error rate and number of IDK-assignments per misclassifications in percent of all learning algorithms and IDK-classifier capable multiclass methods. Results are left out, either if there are less than one IDK classified instances or the error rate is significantly worse than the error rate of the best interpretable method. For a full list see Tables A.1 - A.3.

the credit agencies' processes of determining credit ratings are unknown and differ from agency to agency. We suspect that Creditreform's credit rating is focused on thresholds of account data. Therefore, threshold-based models are more appropriate to reconstruct this credit rating using these data sets. Other approaches from Section 4.2 would probably increase the performance of SVMs and ANNs. However, the small performance gain of methods from these studies compared to the standard algorithms used in this case study, renders it unlikely to fill the performance gap between the former and the threshold-based algorithms. This applies at least to this problem and these data sets.

6.2.6.2. Performance of Interpretable Models Compared Among Themselves

In this case study, interpretable models outperform non-interpretable models and according to Figure 6.9 and Table 6.12, there are no significant differences between the error rates of the interpretable multiclass methods except for two cases. The Thresholder one-vs-rest method for the construction data set and the RIPPER one-vs-followers method for the finance data set perform significantly worse than the best results for these data sets. On the contrary, model sizes differ from each other. The smallest model size for each data set is always achieved by Thresholder.

6.2.6.3. Model Sizes of Interpretable Models

The parameter selection for the previous comparisons is solely based on the classification error and does not consider model size, because we wanted to compare interpretable and non-interpretable algorithms. Therefore, it would be unfair to choose the winner of the similarly performing models solely based on the model size of these experiments. As described above, we did another experiment using different generalization parameters to compare classification error and model size for the all-at-once method of the Thresholder, RIPPER, and C4.5 algorithm. Figure 6.10 and Table 6.13 show that increasing the generalization also increases the classification error and lowers the model size at the same time. We consider the smallest model sizes that do not perform significantly worse than the overall best interpretable result of this data set. This is done for each algorithm and data set. In the following, we will refer to these smallest model sizes as the model size of an algorithm. Thus, we can compare the similarly performing algorithms based on their model size.

Furthermore, we did a significance analysis for the model sizes of each algorithm to determine models which are significantly bigger than the smallest model. For the trade data set, Thresholder and C4.5 models are significantly smaller than RIPPER models. Thresholder yields the significantly smallest models for the construction data set as well. For the finance data set the situation is different due to the small amount of low-rated enterprises. Using small models, these enterprises are ignored by the learners which results in model sizes below five. Nevertheless, these model sizes could be achieved without getting significantly worse. For the finance data set, RIPPER models are significantly smaller than

Thresholder models. However, the advantage is small and only significant because of the constant model sizes of Thresholder and C4.5 over all repetitions of the experiments. Nevertheless, the mean model size of Thresholder (5.87) is more than one threshold smaller than the sizes of C4.5 (7.13) and RIPPER (6.93). Although the proportions of these numbers might seem small, it has to be taken into consideration that the number of thresholds in these models is much smaller (3.21, 4.13, and 4.33) resulting in differences of about 25%. A reason for the small thresholder models is probably the combination of several multiclass methods and choosing the best ones with the smallest model size. Another reason could be the generalization parameter which directly allows to control the model size.

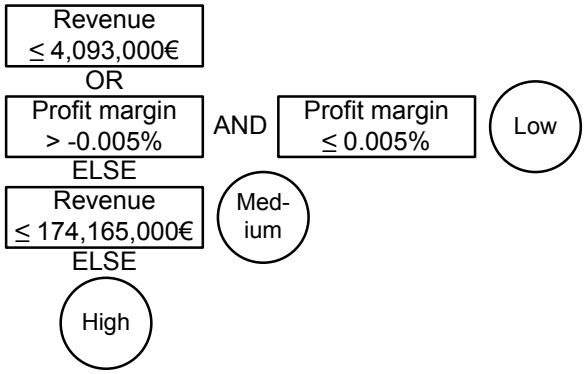
Figures 6.11 - 6.13 show example models for all interpretable all-at-once algorithms for each data set. For each algorithm and data set we picked a single model out of the 20 repetitions of the parameter set with the biggest generalization that did not perform significantly worse. We took the models whose model sizes were closest to the mean model size for this algorithm, data set, and parameter set. It can be seen that these models are small and only contain a small number of different financial ratios. The finding that few features are sufficient to solve financial problems was shown before [21, 55]. Investigating these models shows that revenue is the most important financial ratio to classify trade and construction enterprises. Despite the fact that revenue is of no importance for financial enterprises, missing values (replaced by zero-values) are indeed a very important discrimination criterion. The worse the enterprise's rating the more missing values seem to appear in their annual accounts. The models catch them by using an upper threshold of or slightly above zero and a second lower threshold of or slightly below zero. All models displayed are of an interpretable structure and of an interpretable size. The size of the Thresholder DNFs is lower than or equal to the size of RIPPER DNFs and DTs. Albeit, DT interpretability is of a different kind.

6.2.6.4. Performance of IDK-classifiers

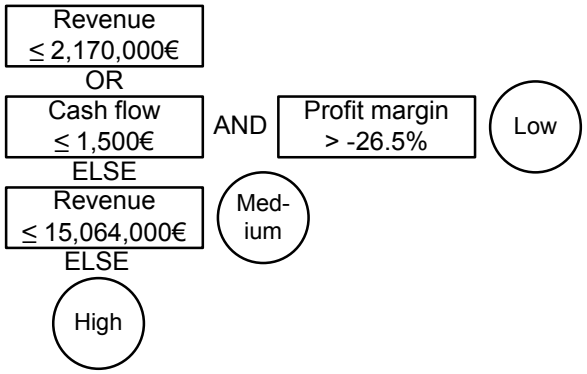
Roughly a third of the 34 IDK-classifier results assign at least one IDK label and are not significantly worse than the best interpretable result. There are less for the trade (9) and construction data sets (10) and more for the finance data set (17). As expected, increasing the IDK class weight τ leads to a higher amount of IDK-class assignments. In return it leads to higher error rates. However, there are results assigning a higher number of uncertain predictions whose errors are not significantly worse than the error of the best interpretable result.

As mentioned above, the one-vs-rest approach is more likely to produces a tie than the other methods, resulting in more IDK-classifications. However, this does not hold for Thresholder where all methods perform similarly. An explanation for this could be the pruning of the cascaded DNF which might prune the decisions leading to a tie.

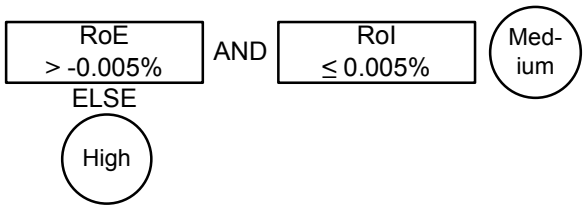
There is only a maximum number of five IDK-classifications in the trade and construction data set and 14 in the finance data set which seems to be a very low number compared



(a) Trade data set model.

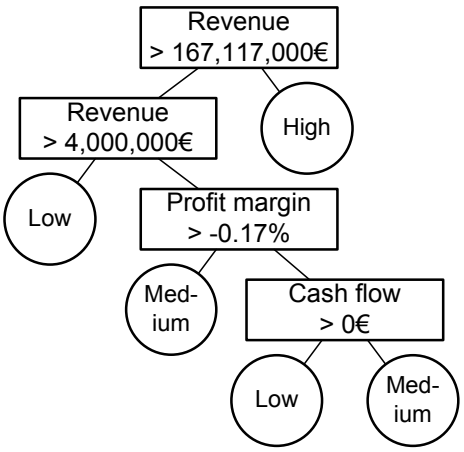


(b) Construction data set model.

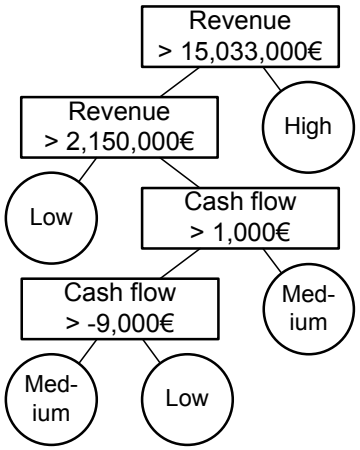


(c) Finance data set model.

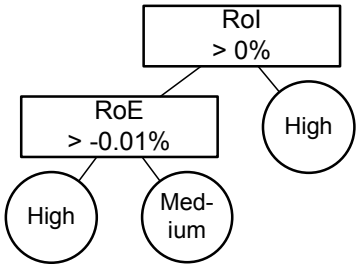
Figure 6.11.: Example models for Thresholder DNFs built on the three data sets.



(a) Trade data set model.

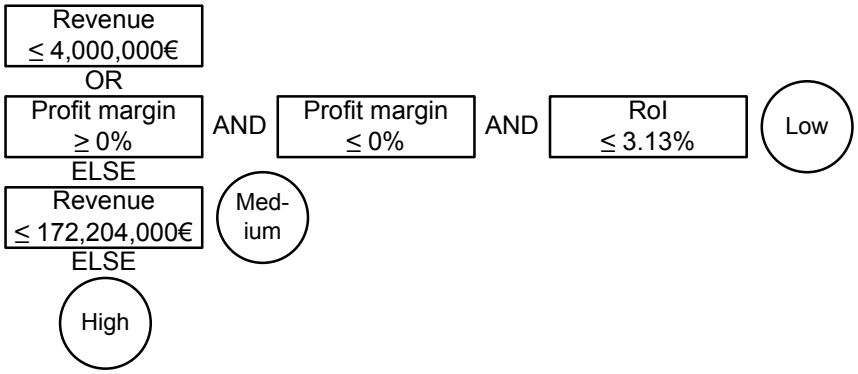


(b) Construction data set model.

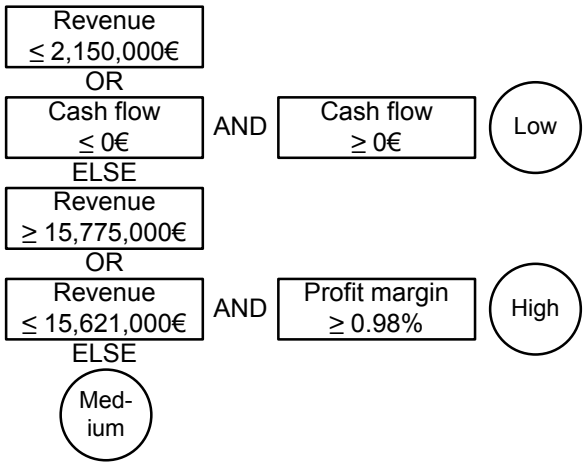


(c) Finance data set model.

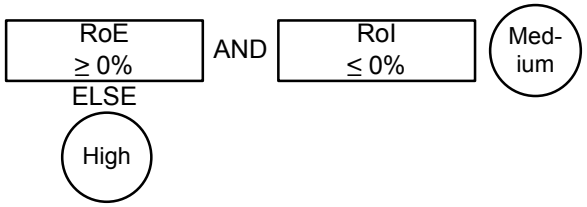
Figure 6.12.: Example models for C4.5 DTs built on the three data sets.



(a) Trade data set model.



(b) Construction data set model.

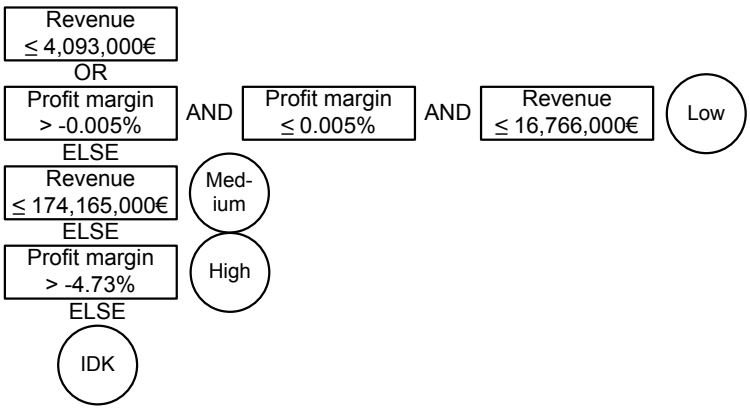


(c) Finance data set model.

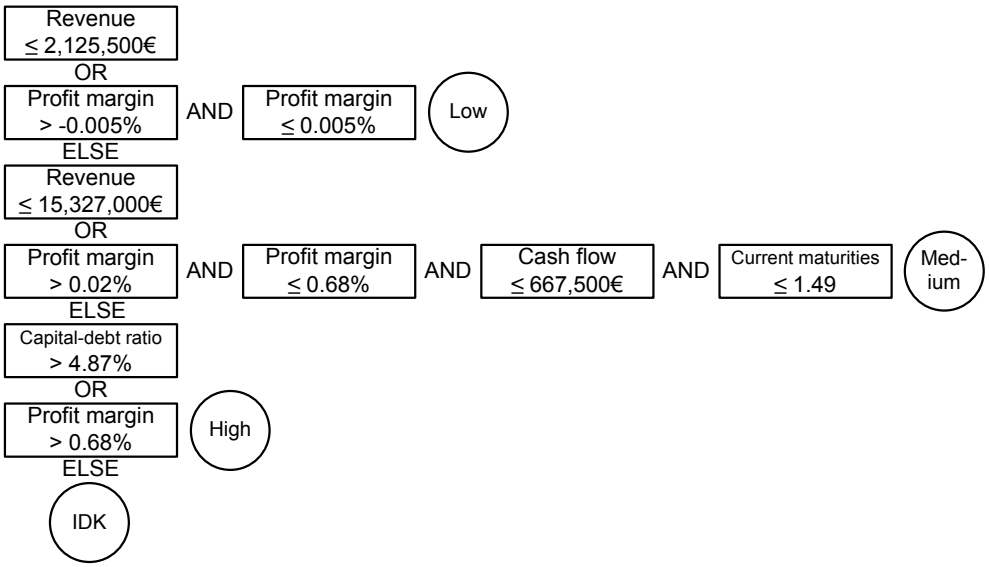
Figure 6.13.: Example models for RIPPER DNFs built on the three data sets.

to the size of a data set. Nevertheless, the purpose of this study is not to find as many IDK-instances as possible. Correct predictions are still the most important observations. However, it is desirable to turn as many misclassifications as possible into IDK-classifications. Consider that these absolute numbers of IDK-classifications are only observed on the test data set which is 33% of the whole data. Comparing their amount with the number of misclassifications yields rates of 15% to 30% IDK-classifications per misclassification without worsening the accuracy significantly.

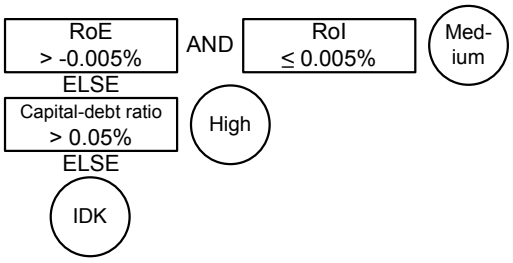
While performing not significantly different, RFs yield the most IDK-classifications for the trade data set, Thresholder for the construction data set, and RIPPER for the finance data set. However, only Thresholder yields interpretable models to explain these IDK-assignments. Figure 6.14 shows example DNFs built by Thresholder using IDK-classification. For each data set we selected the setting which yields the most IDK-assignments without performing significantly worse than the best interpretable result. The model whose number of IDK-assignments is closest to the mean number of IDK-assignments for this data set and settings is displayed. The trade and finance models are similar to those in Figure 6.11, but with one additional threshold which discriminates the IDK-label from the rest. The model built on the construction data set is much bigger. An explanation can be found in the missing optimization of the model size as done in the experiment above. However, we think that these models are well suited for decision support with their additional information about doubt.



(a) Trade data set model.



(b) Construction data set model.



(c) Finance data set model.

Figure 6.14.: Example models for Thresholder DNFs using IDK-classifications built on the three data sets.

7 Discussion

In this chapter, we discuss the results of this thesis as a whole and provide an overview of the research contributions and limitations of our work.

Even though the results are empirical, we conclude that interpretable models are well suited for classification problems in finance. Those performed slightly better than non-interpretable models in the first case study on insolvency prediction. In the second case study on credit rating interpretable models even outperformed non-interpretable models reducing the misclassifications by about 50%. The results can be explained by the way classification is achieved. Insolvency happens when an enterprise cannot pay its debts which is caused by several financial factors in contrast to credit ratings which are at least partially man-made classifications. We conclude that man-made classifications are based on few thresholds, and therefore, can be detected by threshold-based algorithms. ANNs and SVMs might build models which are too complex for these simple rules and lead to an overfitting of the data. The fact that less sophisticated L-SVMs outperform P-SVMs and R-SVMs on two data sets confirms this assumption. Furthermore, boosting which reduces bias by building more complex models does not increase the performance.

Since all threshold-based methods perform similarly, we expect the remaining error to be noise which cannot be explained by the data. In fact, Creditreform gave information that annual accounts only influence their rating by 20%. Instead, qualitative factors like payment experiences from the past are more important.

7.1. Contributions

The research contributions of this work are manifold and described in the following.

Showing Non-inferiority of Interpretable Models for Two Problems in Finance We showed non-inferiority of interpretable models for two different problems in finance. An explanation for these surprising results is provided and can be found in the nature of the problems.

Focusing on Interpretability in Detail Most studies in this field of research focus on improving the accuracy of prediction models. Some of them focus on interpretability as well. However, we focus mainly on interpretability and thus examine them in more detail. We do not just apply interpretable models, we also compare both the interpretability of their model structures and the model sizes of different learning algorithms.

Using Asymmetric Bagging to Solve the Class Imbalance Problem Imbalances are a big problem for finance problems where unusual cases are predicted. For example, insolvencies are not the normal case and therefore less common. We propose a successfully tested method to increase the accuracy for imbalanced data problems.

Building Adjustable DNFs Using Thresholder We present an algorithm to build DNFs which are more adjustable than the ones built by other algorithms. The generalization parameters are more comprehensible. It is possible to directly adjust the amount and structure of thresholds used in this model in various ways. As a result, decision makers can build models according to their purpose.

Building Cascaded DNFs on the Basis of Standard Multiclass Methods Using Thresholder Cascaded DNFs as interpretable multiclass models are already used by RIPPER. However, our approach calculates them in several different ways on the basis of well-known multiclass meta-algorithms inheriting their pros and cons. For practical applications, a particular method can be chosen or they can be used all together and Thresholder chooses the best one. Furthermore, the comprehensible generalization parameters can be applied to cascaded DNFs. Nevertheless, this approach performs as well as other algorithms building interpretable models while its model size is smaller.

Assigning IDK-classifications We provide a general approach for assigning IDK-classifications for multiclass meta-algorithms which use a voting. These IDK-labels can be used as a marker for doubtful classifications. These marks allow for a selective application of more expensive classification methods, e.g., classification by hand. However, simple methods can still be applied, e.g., assigning the most critical label or randomly choosing a

label. In two thirds of the cases, the performance either significantly drops or the number of IDK-assignments is vanishingly low. However, the remaining results are not significantly worse with up to 30% of IDK-classifications per misclassification. Therefore, it can be considered a useful decision support in cases where the performance is not negatively affected.

Interpretable Assignments of IDK-classifications Using Thresholder Last but not least, we combine the two preceding contributions and present an approach to build interpretable models assigning IDK-classifications. Furthermore, the balance between accuracy and number of IDK-assignments can be controlled by a parameter.

7.2. Limitations

In this section we will explain the limitations of this thesis.

Comparision with Other Interpretable Models and Algorithms We only evaluated interpretable DT and DNF models. For each of them we only used one algorithm. Additionally we used our approach. Nevertheless, there are other interpretable models and algorithms building interpretable models as can be seen in Section 3.2.1.3. Instead of testing additional approaches we decided to use only the most popular methods (see Section 3.2.1.3) due to the similar nature of most of the approaches (see Section 3.3). Using them would not change the finding that DTs and DNFs are not inferior to non-interpretable models.

Furthermore, it is unlikely that the performance could be improved much further as C4.5 is an established algorithm. As mentioned at the beginning of this section, we do not think that much better results could be achieved because all threshold-based models perform equally well. However, different models would yield a different kind of interpretation. We consider the evaluation of different interpretations future work. Nevertheless, Thresholder has benefits that we have not discovered in related literature, i.e., comprehensible model size restriction and interpretable IDK-classifications.

Comparision with Explanatory Approaches for Non-interpretable Models As stated in Section 4.3.1, there are some approaches that try to simplify non-interpretable models, e.g., by rule extraction. We consider a comparison of their performance and interpretability future work.

Comparision with Improvements of SVMs, ANNs, and Ensemble Methods In our case studies, we compare our work only with standard SVMs, ANNs, and AdaBoost. However, there are a lot of improvements of these algorithms. Section 4.1 and 4.2 only list some applied to finance problems. The application of these approaches is difficult as there are no implementations publicly available for most cases. Since also the data sets used for

evaluation are mostly not publicly available, these algorithms have to be reimplemented for a proper comparison on one's own data. However, this enormous effort discourages most researchers. Therefore, most studies in finance including this one compare their approaches with standard algorithms only. In Section 6.2.6.1, we justify our approach with the big performance gap between interpretable and non-interpretable methods and the other case studies' small performance gain.

Results are Empirical The results of our case study do not state that interpretable models are always non-inferior to more sophisticated models. The results are empirical and apply only to our data sets. However, we expect similar results for the same problem statements since we performed a proper evaluation and used multiple real-life data sets. Nevertheless, results may vary, especially for the credit rating, because different rating agencies apply different rating techniques. Thus, different algorithms could be more appropriate for credit data from different agencies.

Doubtlessly, the results are not generally transferable to other problems. For most applications SVMs and ANNs still yield the best results. However, they perform worse for insolvency and for credit rating whereas the bad results for insolvency prediction are more surprising than for credit rating.

Runtime of Our Approaches A drawback concerning Thresholder and asymmetric bagging maybe the runtime. The post-pruning of Thresholder is more costly than the pruning of RIPPER which takes place directly after the calculation of each monomial. However, limiting the model size via pruning works only this way. Thresholder's technique of combining all multiclass methods and choosing the best one is a runtime expensive method as well. Despite these two costly features Thresholder's runtime is shorter than the runtime of the SVMs in our studies. When runtime is important, these two features can be left out without worsening the prediction performance much.

8 Conclusion

This chapter summarizes the thesis and provides a brief outlook on future work.

8.1. Summary

Computer-aided rating and insolvency prediction have become a powerful tool. While its accuracy is not 100% reliable, it is a very helpful decision support.

As stated in Section 7.1, the research contributions of this thesis are manifold. In two case studies on insolvency prediction and credit rating, we showed that the simple DNF and DT models are not inferior or even perform better than non-interpretable models like SVMs and ANNs. As a consequence the best results are actually created in a way that is easy to interpret. We showed the theoretical differences between these two hypothesis classes. They can be transformed into each other, but with a loss of interpretability. We provided example models for both of them built on real-life data and offer an interpretative approach. We concluded that none of them is superior concerning interpretability. Thus, the preferable method is a matter of taste. We presented a new interpretable multiclass method to learn DNFs by adopting several well known multiclass methods. The comprehensible generalization parameters of our method are advantages as they allow for a direct adjustment of model size and structure. The classification error of our method is similar to the other interpretable methods, but further experiments show smaller model sizes with similar error rates. Born

from necessity, another small research contribution is the finding that asymmetric bagging helps to tackle the problem of imbalanced data for most learning algorithms.

The practical implication of our research is that interpretable models with thresholds are well suited for some classification problems in finance and should be used. Since DNFs and DTs perform similarly well and are of a different structure, we recommend using both models as decision support. If the size of DTs or DNFs is too big for easy interpretation, both algorithms provide parameters to decrease the model size. We prefer using our Thresholder algorithm over RIPPER for DNF models because it builds smaller models while yielding a comparable classification error. Furthermore, Thresholder provides parameters which are more comprehensible to adjust the amount of interpretability, e.g., by determining a maximum model size instead of determining a minimum number of instances in a rule. In practice, these models can be used by managers to justify decisions. Once they understand the model, they will hopefully be less reluctant to use it. Also, they might be able to combine statistical models with expert knowledge [39].

8.2. Outlook

Our results show that future work on computer-aided rating is a promising research topic. Interpretable models are quite underrated. Since they are not inferior to other models for solving important problems in finance, there should probably be more effort put in increasing the performance of algorithms building interpretable models.

In the future, we would like to implement the generalization of our multiclass method to work for more than three classes and evaluate it accordingly. These models will get much bigger due to logical operations on more DNFs. Experiments will show whether the resulting cascaded DNFs can be pruned down to an interpretable size.

We would like to identify additional problem statements where interpretable models are non-inferior to interpretable models. We suggest using sophisticated models only when it is necessary. However, to determine this necessity it is important to understand which problems are solvable by interpretable models.

As mentioned in Section 7.2, we want to evaluate different models which yield a different kind of interpretation, e.g., scoring tables or decision diagrams. Moreover, explanatory approaches for non-interpretable models deserve to be studied in more detail concerning the interpretability-performance trade-off.

A Supplementary Results

This appendix provides additional visualizations and results of this thesis. They do not necessarily provide new findings, but supplement some parts which we felt were too detailed. Therefore, they were left out in the main text.

A.1. Credit Rating

This section presents the supplementary results of the second case study on credit rating from Section 6.2.

A.1.1. Performance of All Models and Methods

Figure A.1 - A.3 visualize the results of each data set from Table 6.11. In the case study, only the mean over all data sets was visualized.

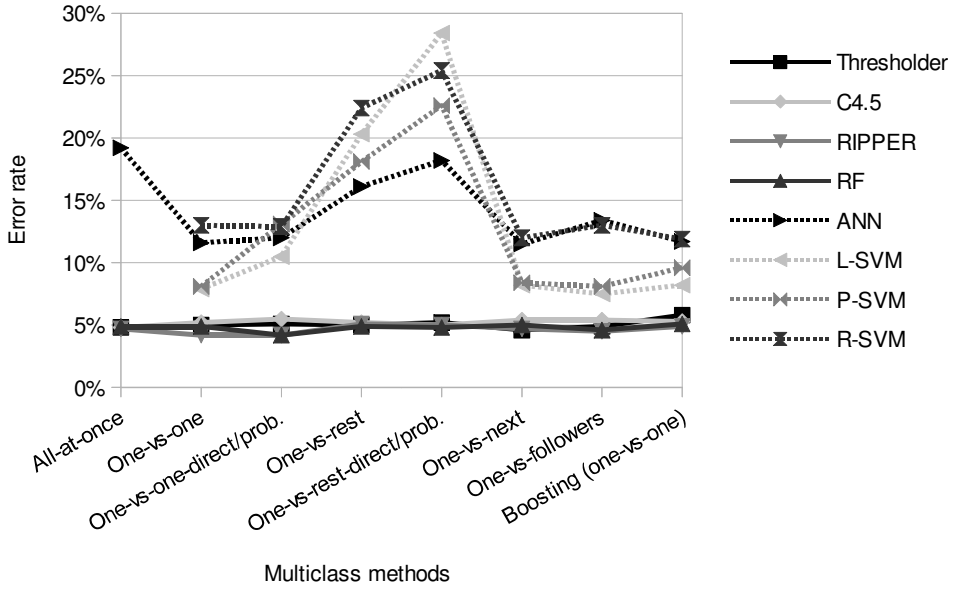


Figure A.1.: Mean error values of learning algorithms and multiclass methods for the trade data set.

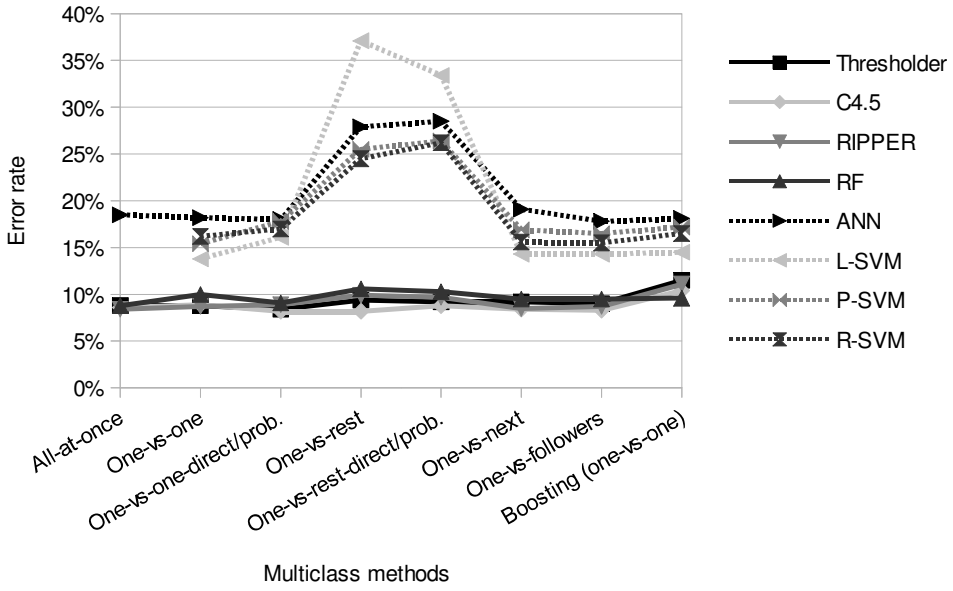


Figure A.2.: Mean error values of learning algorithms and multiclass methods for the construction data set.

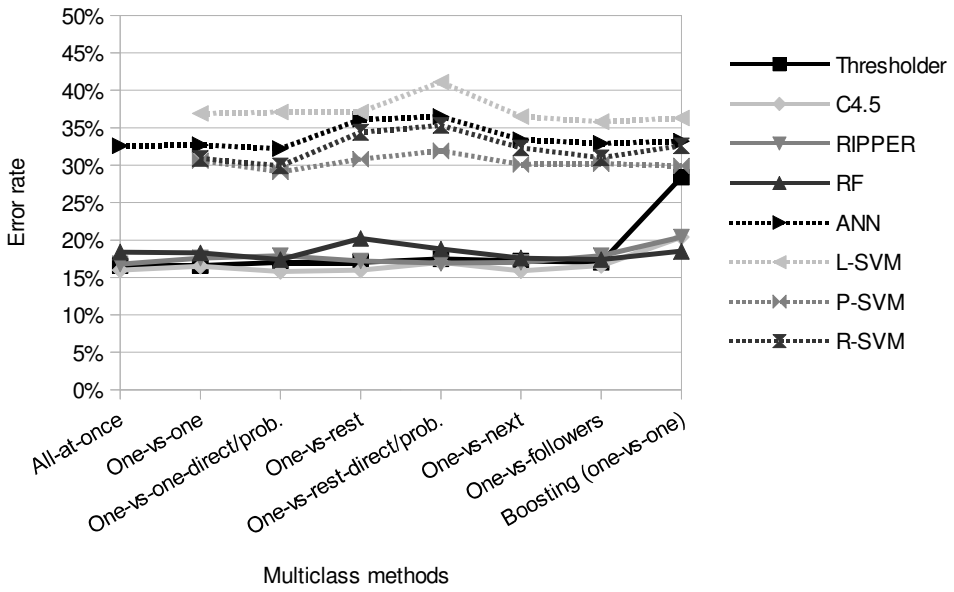


Figure A.3.: Mean error values of learning algorithms and multiclass methods for the finance data set.

A.1.2. Performance of Interpretable Models Compared Among Themselves

Figure A.4 - A.6 visualize the results of each data set from Table 6.12. In the case study, only the mean over all data sets was visualized.

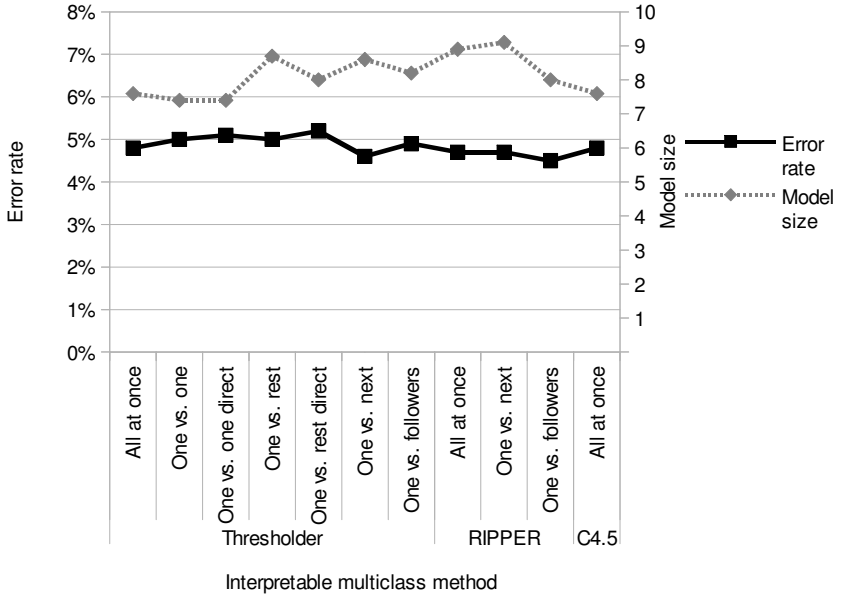


Figure A.4.: Comparison of error rate and model size for the trade data set of all interpretable multiclass models in this study.

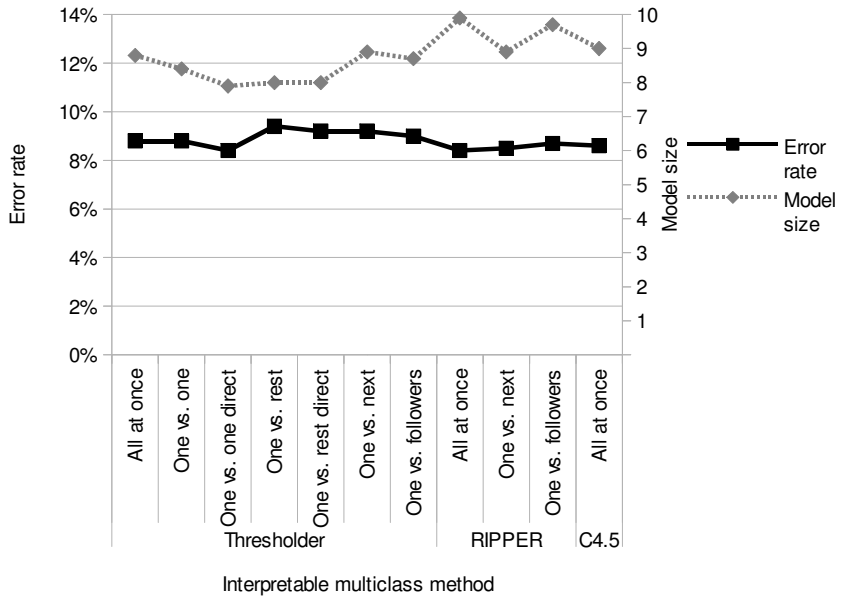


Figure A.5.: Comparison of error rate and model size for the construction data set of all interpretable multiclass models in this study.

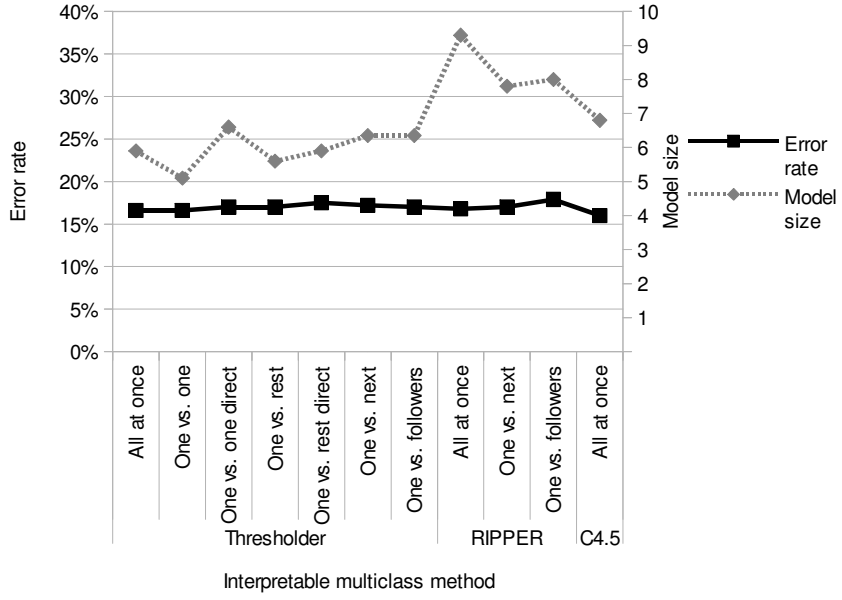


Figure A.6.: Comparison of error rate and model size for the finance data set of all interpretable multiclass models in this study.

A.1.3. Performance of IDK-classifiers

Table A.1 - A.3 show the full list of results from Table 6.12. In the case study, results which are significantly worse than the best interpretable result or with less than one assigned IDK-label were intentionally left out.

Learning algorithm	Multiclass method	τ	#IDK	#IDK / #err	$\text{err}_{\tau=0.67}$	Sig.
Thresholder	All-at-once	0.67	2.8	15	4.8	✓
		0.5	2.9	15	5.0	✓
		0.3	4.3	23	5.2	✓
		0.1	14.6	81	6.7	
	One-vs-one	0.67	0.6	3	4.7	
		0.5	0.7	4	4.7	
		0.3	2.7	14	5.0	✓
		0.1	15.3	78	7.2	
	One-vs-one-direct	0.67	1.7	8	5.0	✓
		0.5	1.9	10	4.9	✓
		0.3	3.5	18	5.2	✓
		0.1	11.9	67	6.2	
	One-vs-rest	0.67	0.4	2	5.1	
		0.5	0.6	3	5.2	
		0.3	4.0	19	5.8	
		0.1	14.8	75	7.1	
	One-vs-rest-direct	0.67	0.5	2	5.2	
		0.5	0.4	2	5.2	
		0.3	2.7	14	5.1	✓
		0.1	10.5	53	6.5	
C4.5	One-vs-one		0.1	0	5.2	
	One-vs-rest		2.6	13	5.3	
RIPPER	One-vs-one		0.1	0	4.2	
	One-vs-rest		11.7	76	5.6	
RF	One-vs-one		0.2	1	4.9	
	One-vs-rest		5.0	30	4.8	✓
ANN	One-vs-one		4,1	9	11,9	
	One-vs-rest		92,2	309	22,1	
L-SVM	One-vs-one		1,2	4	7,8	
	One-vs-rest		12,0	15	21,3	
P-SVM	One-vs-one		4,4	14	8,0	
	One-vs-rest		40,2	68	20,7	
R-SVM	One-vs-one		6,9	14	13,4	
	One-vs-rest		40,2	56	23,8	

Table A.1.: Trade data set: Absolute number of IDK-assignments and errors and number of IDK-assignments per misclassifications in percent of all learning algorithms and IDK-classifier capable multiclass methods. Methods with more than one IDK classified instance and an error rate not significantly worse than the error rate of the best interpretable method are checkmarked.

Learning algorithm	Multiclass method	τ	#IDK	#IDK / #err	$\text{err}_{\tau=0.67}$	Sig.
Thresholder	All-at-once	0.67	3.2	8	9.3	✓
		0.5	5.4	15	8.9	✓
		0.3	8.5	22	9.9	
		0.1	38.3	17	13.0	
	One-vs-one	0.67	5.1	12	9.8	
		0.5	6.4	16	9.9	
		0.3	9.3	23	10.3	
		0.1	30.9	78	13.4	
	One-vs-one-direct	0.67	2.4	6	9.1	✓
		0.5	3.8	10	9.1	✓
		0.3	5.6	14	9.5	✓
		0.1	38.0	105	13.7	
	One-vs-rest	0.67	3.6	9	9.5	✓
		0.5	4.3	11	9.5	✓
		0.3	9.0	24	9.7	
		0.1	38.5	106	13.8	
	One-vs-rest-direct	0.67	4.9	12	9.5	✓
		0.5	4.2	10	9.6	✓
		0.3	12.4	32	10.5	
		0.1	28.8	69	13.6	
C4.5	One-vs-one		0.3	1	8.9	
	One-vs-rest		2.3	6	8.3	✓
RIPPER	One-vs-one		0.7	2	8.7	
	One-vs-rest		13.4	39	9.6	
RF	One-vs-one		0.5	1	10.0	
	One-vs-rest		15.5	45	9.9	
ANN	One-vs-one		3,4	4	18,2	
	One-vs-rest		111,3	200	29,0	
L-SVM	One-vs-one		1,1	2	13,7	
	One-vs-rest		179,9	428	36,2	
P-SVM	One-vs-one		3,9	6	15,3	
	One-vs-rest		101,2	191	26,9	
R-SVM	One-vs-one		3,1	4	16,1	
	One-vs-rest		97,4	170	27,3	

Table A.2.: Construction data set: Absolute number of IDK-assignments and errors and number of IDK-assignments per misclassifications in percent of all learning algorithms and IDK-classifier capable multiclass methods. Methods with more than one IDK classified instance and an error rate not significantly worse than the error rate of the best interpretable method are checkmarked.

Learning algorithm	Multiclass method	τ	#IDK	#IDK / #err	$\text{err}_{\tau=0.67}$	Sig.
Thresholder	All-at-once	0.67	10.9	19	18.0	✓
		0.5	10.7	19	17.9	✓
		0.3	8.6	16	17.4	✓
		0.1	42.0	84	22.2	
	One-vs-one	0.67	4.6	8	17.4	✓
		0.5	3.4	6	17.1	✓
		0.3	4.2	7	16.6	✓
		0.1	42.2	77	23.5	
	One-vs-one-direct	0.67	7.1	12	18.3	✓
		0.5	9.3	16	18.0	✓
		0.3	13.6	24	18.9	
		0.1	49.8	92	24.8	
	One-vs-rest	0.67	2.0	3	17.8	✓
		0.5	6.6	11	18.4	✓
		0.3	7.7	13	18.4	✓
		0.1	42.8	78	23.8	
	One-vs-rest-direct	0.67	4.8	8	18.2	✓
		0.5	8.3	14	18.0	✓
		0.3	6.8	12	17.9	✓
		0.1	45.9	86	23.9	
C4.5	One-vs-one		0.6	1	16.5	
	One-vs-rest		5.0	10	15.7	✓
RIPPER	One-vs-one		0.7	1	17.5	
	One-vs-rest		14.1	27	17.3	✓
RF	One-vs-one		1.1	2	18.2	✓
	One-vs-rest		23.6	45	19.4	
ANN	One-vs-one		5,0	4	32,5	
	One-vs-rest		33,5	33	34,8	
L-SVM	One-vs-one		1,3	1	36,8	
	One-vs-rest		11,4	9	36,8	
P-SVM	One-vs-one		4,1	4	30,5	
	One-vs-rest		14,5	15	30,3	
R-SVM	One-vs-one		1,8	2	30,9	
	One-vs-rest		21,8	21	33,5	

Table A.3.: Finance data set: Absolute number of IDK-assignments and errors and number of IDK-assignments per misclassifications in percent of all learning algorithms and IDK-classifier capable multiclass methods. Methods with more than one IDK classified instance and an error rate not significantly worse than the error rate of the best interpretable method are checkmarked.

Bibliography

[1] Joaquín Abellán and Carlos J. Mantas. Improving experimental studies about ensembles of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, 41(8):3825 – 3830, 2014.

[2] Esteban Alfaro, Noelia García, Matías Gámez, and David Elizondo. Bankruptcy forecasting: An empirical comparison of adaboost and neural networks. *Decision Support Systems*, 45(1):110–122, 2008.

[3] Esteban Alfaro Cortés, Matías Gámez Martínez, and Noelia García Rubio. Multiclass corporate failure prediction by adaboost.m1. *International Advances in Economic Research*, 13(3):301–312, 2007.

[4] Ethem Alpaydin. *Introduction to Machine Learning*. Adaptive computation and machine learning. MIT Press, 2004.

[5] Edward I. Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *Journal of Finance*, 23(4):589–609, September 1968.

[6] Edward I. Altman, Robert G. Haldeman, and P. Narayanan. ZETA analysis a new model to identify bankruptcy risk of corporations. *Journal of Banking & Finance*, 1(1):29 – 54, 1977.

[7] Silvia Angilella and Sebastiano Mazzù. The financing of innovative smes: A multicriteria credit rating model. *European Journal of Operational Research*, 244(2):540 – 554, 2015.

[8] Chidanand Apté and Sholom Weiss. Data mining with decision trees and decision rules. *Future Generation Computer Systems*, 13(2-3):197–210, 1997. Data Mining.

[9] Sofie Balcaen and Hubert Ooghe. 35 years of studies on business failure: an overview of the classic statistical methodologies and their related problems. *The British Accounting Review*, 38(1):63 – 93, 2006.

[10] David Barbella, Sami Benzaid, Janara M. Christensen, Bret Jackson, X. Victor Qin, and David R. Musicant. Understanding support vector machine classifications via a recommender system-like approach. In *DMIN*, pages 305–311, 2009.

[11] William H. Beaver. Financial ratios as predictors of failure. *Journal of Accounting Research*, 4(3):71–111, 1966.

[12] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 2006.

[14] Marc Blum. Failing company discriminant analysis. *Journal of Accounting Research*, 12(1):1–25, 1974.

[15] Koen W. De Bock and Dirk Van den Poel. Reconciling performance and interpretability in customer churn prediction using ensemble learning based on generalized additive models. *Expert Systems with Applications*, 39(8):6816 – 6826, 2012.

[16] Indranil Bose. Deciding the financial health of dot-coms using rough sets. *Information & Management*, 43(7):835 – 846, 2006.

[17] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

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

[19] Thomas Brodag. *PAC-Lernen zur Insolvenzerkennung und Hotspot-Identifikation: Anwendung statistischer Modelle des algorithmischen Lernens auf betriebswirtschaftliche und bioinformatische Probleme der Praxis*. PhD thesis, Georg August Universität Göttingen, 2008.

[20] Iain Brown and Christophe Mues. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3):3446–3453, 2012.

[21] Lijuan Cao, Lim Kian Guan, and Zhang Jingqing. Bond rating using support vector machine. *Intelligent Data Analysis*, 10(3):285–296, May 2006.

[22] Andreas Charitou, Evi Neophytou, and Chris Charalambous. Predicting corporate failure: Empirical evidence for the UK. *European Accounting Review*, 13(3):465–497, 2004.

[23] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6, June 2004.

[24] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):pp. 233–235, 1979.

[25] William Cohen. Fast effective rule induction. In *Twelfth International Conference on Machine Learning*, 1995.

[26] Paulo Cortez and Mark J. Embrechts. Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, 225:1 – 17, 2013.

[27] Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8:24–30, 1996.

[28] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[29] Jonathan N. Crook, David B. Edelman, and Lyn C. Thomas. Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, 183(3):1447 – 1465, 2007.

[30] Paulius Danenas and Gintautas Garsva. Selection of support vector machines based classifiers for credit risk domain. *Expert Systems with Applications*, 42(6):3194–3204, 2015.

[31] Augustinos I. Dimitras, Stelios H. Zanakis, and Constantin Zopounidis. A survey of business failures with an emphasis on prediction methods and industrial applications. *European Journal of Operational Research*, 90(3):487–513, 1996.

[32] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2012.

[33] Alan Fan and Marimuthu Palaniswami. Selecting bankruptcy predictors using a support vector machine approach. In *Proceedings of the international joint conference on neural networks*, volume 6, pages 354–359, 2000.

[34] Federal Statistical Office of Germany. German Classification of Economic Activities, Edition 2003 (WZ 2003), 2003.

[35] David Feldman and Shulamith Gross. Mortgage default: Classification trees analysis. *The Journal of Real Estate Finance and Economics*, 30(4):369–396, 2005.

[36] Vitaly Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590, 2012.

[37] Eugenio Fernández and Ignacio Olmeda. Bankruptcy prediction with artificial neural networks. In José Mira and Francisco Sandoval, editors, *From Natural to Artificial Neural Computation*, volume 930 of *Lecture Notes in Computer Science*, pages 1142–1146. Springer Berlin Heidelberg, 1995.

[38] Steven Finlay. Multiple classifier architectures and their application to credit risk assessment. *European Journal of Operational Research*, 210(2):368 – 378, 2011.

[39] Raquel Florez-Lopez and Juan Manuel Ramon-Jeronimo. Enhancing accuracy and interpretability of ensemble strategies in credit risk assessment. a correlated-adjusted decision forest proposal. *Expert Systems with Applications*, 42(13):5737 – 5753, 2015.

[40] Raphael Féraud and Fabrice Clérot. A methodology to explain neural network classification. *Neural Networks*, 15(2):237 – 246, 2002.

[41] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[42] Jerome Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.

[43] Sebastian Fritz and Detlef Hosemann. Restructuring the credit process: Behaviour scoring for german corporates. *International Journal of Intelligent Systems in Accounting, Finance & Management*, 9:9–21, 2000.

[44] Johannes Fürnkranz and Gerhard Widmer. Incremental reduced error pruning. In *International Conference on Machine Learning*, 1994.

[45] Halina Frydman, Edward I. Altman, and Duen li Kao. Introducing recursive partitioning for financial classification: The case of financial distress. *The Journal of Finance*, 40(1):269–291, 1985.

[46] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, July 2012.

[47] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing*, 13(10):959–977, 2009.

[48] James A. Gentry, Paul Newbold, and David T. Whitford. Classifying bankrupt firms with funds flow components. *Journal of Accounting Research*, 23(1):146–160, 1985.

[49] Xuesong Guo, Zhengwei Zhu, and Jia Shi. A corporate credit rating model using support vector domain combined with fuzzy clustering algorithm. *Mathematical Problems in Engineering*, 2012, 2012.

[50] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, nov 2009.

[51] David J. Hand. Classifier technology and the illusion of progress. *Statistical science*, 21(1):1–14, 02 2006.

[52] Terry Harris. Credit scoring using the clustered support vector machine. *Expert Systems with Applications*, 42(2):741–750, 2015.

[53] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2nd edition, 2009.

[54] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[55] Petr Hájek. Municipal credit rating modelling by neural networks. *Decision Support Systems*, 51(1):108 – 118, 2011.

[56] Wolfgang Härdle, Yuh-Jye Lee, Dorothea Schäfer, and Yi-Ren Yeh. Variable selection and oversampling in the use of smooth support vector machines for predicting the default risk of companies. *Journal of Forecasting*, 28(6):512–534, 2009.

[57] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[58] Zan Huang, Hsinchun Chen, Chia-Jung Hsu, Wun-Hwa Chen, and Soushan Wu. Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decision Support Systems*, 37(4):543 – 558, 2004.

[59] Jyh-Shing R. Jang and C.-T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *Neural Networks, IEEE Transactions on*, 4(1):156–159, Jan 1993.

[60] Ulf Johansson, Rikard König, and Lars Niklasson. The truth is in there-rule extraction from opaque models using genetic programming. In *FLAIRS Conference*, pages 658–663, 2004.

[61] Stewart Jones, David Johnstone, and Roy Wilson. An empirical evaluation of the performance of binary classifiers in the prediction of credit ratings changes. *Journal of Banking & Finance*, 56(0):72 – 85, 2015.

[62] Laura Kainulainen, Yoan Miche, Emil Eirola, Qi Yu, Benoît Frénay, Eric Séverin, and Amaury Lendasse. Ensembles of local linear models for bankruptcy analysis and prediction. *Case Studies in Business, Industry and Government Statistics*, 4(2):116–133, 2014.

[63] Dae-Ki Kang and Myoung-Jong Kim. Performance enhancement of svm ensembles using genetic algorithms in bankruptcy prediction. In *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10)*, volume 2, pages V2154–V2158, Aug 2010.

[64] Michael Kearns. Thoughts on hypothesis boosting. Unpublished manuscript, December 1988.

[65] Michael J. Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA, 1994.

[66] Jun Woo Kim, H. Roland Weistroffer, and Richard T. Redmond. Expert systems for bond rating: a comparative analysis of statistical, rule-based and neural network systems. *Expert Systems*, 10(3):167–172, 1993.

[67] Kyoung-jae Kim and Hyunchul Ahn. A corporate credit rating model using multi-class support vector machines with an ordinal pairwise partitioning approach. *Computers & Operations Research*, 39(8):1800 – 1811, 2012. Special Issue: Advances of Operations Research in Service Industry.

[68] Ulrich H-G Kreßel. Pairwise classification and support vector machines. In *Advances in kernel methods: support vector learning*, pages 255–268. Cambridge MA: MIT Press, 1999.

[69] Jungeun Kwon, Keunho Choi, and Yongmoo Suh. Double ensemble approaches to predicting firms' credit rating. In *PACIS*, page 158, 2013.

[70] Young S. Kwon, Ingoo Han, and Kun Chang Lee. Ordinal pairwise partitioning (opp) approach to neural networks training in bond rating. *International journal of intelligent systems in accounting finance and management*, 6:23–40, 1997.

[71] Vincent Labatut and Hocine Cherifi. Accuracy Measures for the Comparison of Classifiers. In Al-Dahoud Ali, editor, *The 5th International Conference on Information Technology*, pages 1,5, amman, Jordan, May 2011. Al-Zaytoonah University of Jordan.

[72] Erkki K. Laitinen. Prediction of failure of a newly founded firm. *Journal of Business Venturing*, 7:323–340, 1992.

[73] Maxwell W. Libbrecht and William Stafford Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332, 2015.

[74] Carlos Javier Mantas, José Manuel Puche, and J. M. Mantas. Extraction of similarity based fuzzy rules from artificial neural networks. *International Journal of Approximate Reasoning*, 43(2):202 – 221, 2006.

[75] David Martens, Bart Baesens, Tony Van Gestel, and Jan Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466 – 1476, 2007.

[76] Daniel Martin. Early warning of bank failure: a logit regression approach. *Journal of Banking and Finance*, 1:249–276, 1977.

[77] Thomas E. McKee. Developing a bankruptcy prediction model via rough sets theory. *International Journal of Intelligent Systems in Accounting, Finance & Management*, 9(3):159–173, 2000.

[78] Nicolai Meinshausen. Partition maps. *Journal of Computational and Graphical Statistics*, 20(4):1007–1028, 2011.

[79] John Mills, Lynn Bible, and Richard Mason. Defining free cash flow. *The CPA Journal*, 72(1):36, 2002.

[80] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1991.

[81] Christophe Mues, Bart Baesens, Craig M. Files, and Jan Vanthienen. Decision diagrams in machine learning: an empirical study on real-life credit-risk data. *Expert Systems with Applications*, 27(2):257 – 264, 2004.

[82] Loris Nanni and Alessandra Lumini. An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, 36(2, Part 2):3028 – 3033, 2009.

[83] Joao Carvalho Neves and Armando Vieira. Improving bankruptcy prediction with hidden layer learning vector quantization. *European Accounting Review*, 15(2):253–271, 2006.

[84] Haydemar Núñez, Cecilio Angulo, and Andreu Català. Rule extraction from support vector machines. In *European Symposium on Artificial Neural Networks*, pages 107–112, 2002.

[85] Lennart Obermann and Stephan Waack. Demonstrating non-inferiority of easy interpretable methods for insolvency prediction. *Expert Systems with Applications*, 42(23):9117 – 9128, 2015.

[86] Lennart Obermann and Stephan Waack. Interpretable multiclass models for corporate credit rating capable of expressing doubt. Submitted, 2016.

[87] James A. Ohlson. Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research*, 18(1):109–131, 1980.

[88] Anantha M. Prasad, Louis R. Iverson, and Andy Liaw. Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. *Ecosystems*, 9(2):181–199, 2006.

[89] John Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[90] John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[91] Anil Rajput, Ramesh Prasad Aharwal, Meghna Dubey, SP Saxena, and Manmohan Raghuvanshi. J48 and jrip rules for e-governance data. *International Journal of Computer Science and Security (IJCSS)*, 5(2):201–207, 2011.

[92] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978.

[93] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. A comparative study of data sampling and cost sensitive learning. In *Data Mining Workshops, 2008. ICDMW '08. IEEE International Conference on*, pages 46–52, Dec 2008.

[94] Kenth Skogsvik. Current cost accounting ratios as predictors of business failure: The swedish case. *Journal of Business Finance & Accounting*, 17(1):137–160, 1990.

[95] Roman Slowinski and Constantin Zopounidis. Application of the rough set approach to evaluation of bankruptcy risk. *Intelligent Systems in Accounting, Finance and Management*, 4(1):27–41, 1995.

[96] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014.

[97] Cecilia Sönströd, Ulf Johansson, and Rikard König. Evolving accurate and comprehensible classification rules. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1436–1443, June 2011.

[98] Statistisches Bundesamt. Fachserie 2 Reihe 4.1, Insolvenzverfahren (Unternehmen und Arbeitsstätten). Wiesbaden : Statistisches Bundesamt, 12 2015.

[99] Chao-Ton Su and Yan-Cheng Chen. Rule extraction algorithm from support vector machines and its application to credit screening. *Soft Computing*, 16(4):645–658, 2011.

[100] Jie Sun, Hui Li, Qing-Hua Huang, and Kai-Yu He. Predicting financial distress and corporate failure: A review from the state-of-the-art definitions, modeling, sampling, and featuring approaches. *Knowledge-Based Systems*, 57:41 – 56, 2014.

[101] Kar Yan Tam and Melody Y. Kiang. Managerial applications of neural networks: The case of bank failure predictions. *Management Science*, 38(7):926–947, 1992.

[102] Dacheng Tao, Xiaoou Tang, Xuelong Li, and Xindong Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1088–1099, 2006.

[103] Jakub M. Tomczak and Maciej Zieba. Classification restricted boltzmann machine for comprehensible credit scoring model. *Expert Systems with Applications*, 42(4):1789 – 1796, 2015.

[104] Jon Tucker. Neural networks versus logistic regression in financial modelling: A methodological comparison. In *Proceedings of the 1996 World First Online Workshop on Soft Computing (WSC1)*, volume 1, pages 19–30, 1996.

[105] Fan-Yin Tzeng and Kwan-Liu Ma. Opening the black box - data driven visualization of neural networks. In *Visualization, 2005. VIS 05. IEEE*, pages 383–390, Oct 2005.

[106] U.S. Economic Classification Policy Committee (ECPC), Statistics Canada, and Mexico's Instituto Nacional de Estadistica y Geografia. Standard industrial classification, 1937.

[107] Carsten Uthoff. *Erfolgsoptimale Kreditwürdigkeitsprüfung auf der Basis von Jahresabschlüssen und Wirtschaftsauskünften mit Künstlichen Neuronalen Netzen*. M u. P-Schriftenreihe für Wissenschaft u. Forschung, 1997.

[108] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 2000.

[109] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

[110] Verband der Vereine Creditreform e.V. DAFNE Database. *http://en.creditreform.de/portfolio/marketing-services/target-group-analysis.html [Accessed 16 July 2015]*, 2008.

[111] Verband der Vereine Creditreform e.V. DAFNE Database. *http://en.creditreform.de/portfolio/marketing-services/target-group-analysis.html [Accessed 24 November 2015]*, 2015.

[112] Antanas Verikas, Zivile Kalsyte, Marija Bacauskiene, and Adas Gelzinis. Hybrid and ensemble-based soft computing techniques in bankruptcy prediction: a survey. *Soft Computing*, 14(9):995–1010, 2010.

[113] Miklós Virág and Tamás Nyitrai. Is there a trade-off between the predictive power and the interpretability of bankruptcy models? the case of the first hungarian bankruptcy prediction model. *Acta Oeconomica*, 64(4):419–440, 2014.

[114] Gang Wang, Jian Ma, and Shanlin Yang. An improved boosting based on feature selection for corporate bankruptcy prediction. *Expert Systems with Applications*, 41(5):2353 – 2361, 2014.

[115] Bernard L. Welch. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 1947.

[116] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1(6):80–83, 1945.

[117] Rick L. Wilson and Ramesh Sharda. Bankruptcy prediction using neural networks. *Decision Support Systems*, 11:545–557, 1994.

[118] Mark E. Zmijewski. Methodological issues related to the estimation of financial distress prediction models. *Journal of Accounting Research*, 22:59–82, 1984.