Machine Learning for Credit Risk Analytics

Doctoral Thesis

to acquire the academic degree of
DOCTOR RERUM POLITICARUM
(Doctor of Economics and Management Science)

submitted to

School of Business and Economics
Humboldt University of Berlin

by

Nikita Kozodoi, M.Sc.

President of Humboldt University of Berlin:
Prof. Dr. Peter Frensch (komm.)

Dean of the School of Business and Economics:
Prof. Dr. Daniel Klapper

Reviewers:
1. Prof. Dr. Stefan Lessmann
2. Prof. Dr. Nadja Klein

Date of Colloquium: 24.03.2022

Abstract

The rise of machine learning (ML) and the rapid digitization of the economy has substantially changed decision processes in the financial industry. Financial institutions increasingly rely on ML to support decision-making. Credit scoring is one of the prominent ML applications in finance. The task of credit scoring is to distinguish applicants who will pay back the loan or default. Financial institutions use ML to develop scoring models, also known as scorecards, to estimate a borrower's probability to default and automate the approval decisions.

This dissertation focuses on three major challenges associated with building ML-based scorecards in consumer credit scoring: (i) optimizing data acquisition and storage costs when dealing with high-dimensional data of loan applicants; (ii) addressing the adverse effects of sampling bias on training and evaluation of scoring models; (iii) measuring and ensuring the scorecard fairness while maintaining high profitability. The thesis offers a set of tools to remedy each of these challenges and improve decision-making practices in financial institutions. The proposed methodologies are empirically tested on real-world credit data.

The first challenge stems from a growing number of emerging data sources on loan applicants. Using more features tends to improve the scorecard accuracy. At the same time, data are often purchased from third parties, which incurs extra costs. Furthermore, companies are required to comply with regulations (e.g., the Basel Accords) that enforce comprehensible models. To address these conflicting goals, the thesis develops novel feature selection strategies that optimize multiple business-inspired objectives. We show that our propositions reduce data acquisition costs and improve the model profitability and interpretability.

Another major challenge in credit scoring is sample selection bias. Scoring models are trained on the data of previously granted credit applications with observed repayment behavior. This creates sampling bias: the training data offer a partial picture of the distribution of candidate borrowers, to which the model is applied when screening new applications. We show that this bias impedes the model performance and prohibits accurate model evaluation on historical data. The thesis suggests methods to address the adverse effects of sampling bias. The proposed methods partly recover the loss due to bias, provide more reliable estimates of the future scorecard performance and increase the resulting model profitability.

The third challenge considered in the thesis relates to the algorithmic fairness of credit scorecards. The literature on fair ML in credit scoring is scarce. The thesis addresses this gap and investigates fair ML practices in consumer credit scoring. We catalog suitable algorithmic options for incorporating fairness goals in the model development pipeline and empirically test different fairness processors in a profit-oriented credit scoring context. The empirical results clarify the profit-fairness trade-off in lending decisions and identify suitable options to implement fair credit scoring and measure the scorecard fairness.

Keywords: credit scoring, machine learning, feature selection, sampling bias, fairness

Zusammenfassung

Der Aufstieg des maschinellen Lernens (ML) und die rasante Digitalisierung der Wirtschaft haben die Entscheidungsprozesse in der Finanzbranche erheblich verändert. Finanzinstitute setzen zunehmend auf ML, um die Entscheidungsfindung zu unterstützen. Kreditscoring ist eine der wichtigsten ML-Anwendungen im Finanzbereich. Die Aufgabe von Kreditscoring ist die Unterscheidung ob ein Antragsteller einen Kredit zurückzahlen wird. Finanzinstitute verwenden ML, um Scoring-Modelle zu entwickeln, die auch als Scorekarten bekannt sind. Die Scorekarten schätzen die Ausfallwahrscheinlichkeit eines Kreditnehmers und automatisieren die Genehmigungsentscheidungen.

Diese Dissertation konzentriert sich auf drei große Herausforderungen, die mit dem Aufbau von ML-basierten Scorekarten für die Bewertung von Verbraucherkrediten verbunden sind: (i) Optimierung von Datenerfassungs- und -speicherkosten bei hochdimensionalen Daten von Kreditantragstellern; (ii) Bewältigung der negativen Auswirkungen von Stichprobenverzerrungen auf das Training und die Bewertung von Scorekarten; (iii) Messung und Sicherstellung der Fairness von Instrumenten bei gleichzeitig hoher Rentabilität. Die Arbeit bietet eine Reihe von Instrumenten, um jede dieser Herausforderungen zu lösen und die Entscheidungsfindung in Finanzinstituten zu verbessern. Die vorgeschlagenen Methoden werden empirisch an realen Kreditdaten getestet.

Die erste Herausforderung ergibt sich aus der wachsenden Zahl neuer Datenquellen über Kreditantragsteller. Die Verwendung von mehr Merkmalen verbessert in der Regel die Genauigkeit der Scorekarten. Gleichzeitig werden die Daten oft von Dritten gekauft, was zusätzliche Kosten verursacht. Außerdem müssen die Unternehmen Vorschriften einhalten, die interpretierbare Modelle vorschreiben (z.B. Basler Akkord). Um diesen Zielkonflikten zu begegnen, werden in dieser Arbeit neuartige Strategien zur Merkmalsauswahl entwickelt, die mehrere unternehmensbezogene Zielfunktionen optimieren. Wir zeigen, dass unsere Vorschläge die Kosten der Datenerfassung senken und die Rentabilität und Interpretierbarkeit des Modells verbessern.

Eine weitere große Herausforderung des Kreditscorings ist die Verzerrung der Stichprobenauswahl. Scoring-Modelle werden auf der Grundlage von Daten früherer Kreditanträge mit beobachtetem Rückzahlungsverhalten trainiert. Dies führt zu einer Verzerrung der Stichprobe: die Trainingsdaten bieten ein unvollständiges Bild der Verteilung der Kreditnehmer, auf die das Modell beim Screening neuer Anträge angewendet wird. Wir zeigen, dass diese Verzerrung die Leistung des Modells beeinträchtigt und eine genaue Bewertung des Modells anhand historischer Daten unmöglich macht. In dieser Arbeit werden Methoden vorgeschlagen, um die negativen Auswirkungen der Stichprobenverzerrung zu beseitigen. Die vorgeschlagenen Methoden gleichen den durch die Verzerrung verursachten Verlust teilweise aus, liefern zuverlässigere Schätzungen der künftigen Scorekarte-Leistung und erhöhen die resultierende Modellrentabilität.

Die dritte in dieser Arbeit betrachtete Herausforderung bezieht sich auf die algorithmische Fairness der Kredit-Scorekarten. Die Literatur über faire ML in Kreditscoring ist spärlich. Diese Arbeit befasst sich mit dieser Lücke und untersucht faire ML-Praktiken in Kreditscoring. Wir katalogisieren geeignete algorithmische Optionen für die Einbeziehung von Fairness-Zielen in die Modellentwicklungspipeline und testen empirisch verschiedene Fairness-Prozessoren in einem gewinnorientierten Kreditscoring-Kontext. Die empirischen Ergebnisse verdeutlichen den Kompromiss zwischen Gewinn und Fairness bei Kreditentscheidungen und identifizieren geeignete Optionen zur Implementierung von fairem Kreditscoring und zur Messung der Fairness der Scorekarten.

Schlagworte: Kreditscoring, maschinelles Lernen, Merkmalsauswahl, Stichprobenverzerrung, Fairness

Acknowledgments

I wish to express my deepest gratitude to my supervisor, Prof. Dr. Stefan Lessmann. His continuous support and trust, invaluable advice and great guidance has helped me a lot on every stage of my doctoral path. I am most grateful to Stefan for always finding the time to talk, carefully listening to my ideas, reviewing my manuscript drafts and providing great feedback. This has lead to many hours of interesting discussions on countless occasions. I highly appreciate the amount of effort Stefan invested in each of the research projects we have been working on together. I also greatly thank Prof. Dr Nadja Klein for her insightful feedback on one of my research papers and for taking over the role as my second reviewer.

I am very grateful to Dr. Johannes Haupt and Dr. Annika Baumann. Together with Prof. Dr. Stefan Lessmann, they taught excellent Data Science courses at the Chair of Information Systems at the Humboldt University of Berlin. They introduced me to the exciting field of Machine Learning and inspired me to pursue PhD. I feel happy to be able to make a small and humble contribution towards helping to maintain and further improve these courses offered by the chair.

I want to express my gratitude to everyone who has been working in the Data Science team at Monedo, especially Konstantinos Papakonstantinou, Luis Moreira-Matias, Alamgir Morteza and Yiannis Gatsoulis. Collaboration with these people is what made my PhD possible. Together with them and other colleagues at Monedo we had a lot of exciting discussions and shared many ideas, which greatly contributed to my research and made my regular trips to the company's office in Hamburg so interesting. I appreciate the opportunity to work on some of the most challenging business problems of Monedo.

I am indebted to all my coauthors who contributed to my research projects. Apart from Prof. Dr. Stefan Lessmann and my colleagues at Monedo, I wish to thank Prof. Dr. Bart Baesens, Panos Katsas and Johannes Jacob.

I am grateful to my colleagues and fellow PhD students, including Dr. Alona Zharova, Elizaveta Zinovyeva, Alisa Kim, Marius Sterling, Daniel Jacob, Björn Bokelmann, Dr. Victor Medina and Georg Velev. It was great to be around these smart people working on different projects, always being curious to discuss research over lunch and share their experience. I also would like to thank Anna-Lena Bujarek for her support in day-to-day activities and Wiebke Peters, Elias Baumann and Adam Watkins for their help with many tasks.

My deepest thanks go to my parents, Andrei and Neli, for supporting my decisions. Most of all, I am endlessly grateful to my wonderful partner Margarita for her full support, patience and love. She has been reducing my bias and variance, improving my non-statistical significance and optimizing any utility function that matters to me.

Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

ABC	Artificial bee colony
ABR	Average *bad* rate among accepts
AI	Artificial intelligence
AgMOPSO	Archive-guided multi-objective PSO
AUC	Area under the ROC curve
BASL	Bias-aware self-learning
BS	Brier score
CRAN	The Comprehensive R Archive Network
CV	Cross-validation
DR	Doubly robust
EAD	Exposure at default
EMP	Expected maximum profit
EU	European Union
FNR	False negative rate
FPR	False positive rate
GA	Genetic algorithm
HCA	Hard cutoff augmentation
HV	Hypervolume
IND	Independence
KLIEP	Kullback-Leibler importance estimation procedure
KNN	K nearest neighbors
LGD	Loss given default
LR	Logistic regression
L1	L1-regularized linear regression
LSIF	Least-squares importance fitting
MAIC	Modified Akaike information criterion
MAR	Missing at random
MCAR	Missing completely at random
ML	Machine learning
MMD	Maximum mean discrepancy
MNAR	Missing not at random
NSGA	Non-dominated sorting based algorithm
ONVG	Overall non-dominated vector generation
PAUC	Partial AUC

PAKDD	Pacific-Asia Conference on Knowledge Discovery and Data Mining
PD	Probability of default
PI	Prejudice index
PPV	Positive predictive value
PSO	Particle swarm optimization
RAM	Random-access memory
ROC	Receiver operating characteristic curve
ROI	Return on investment
RMSE	Root mean squared error
RF	Random forest
RP	R-Precision
SBS	Sequential backward selection
SEPA	Strength Pareto evolutionary algorithm
SSL	Shallow self-learning
SF	Sufficiency
SFS	Sequential forward selection
SP	Separation
SPC	Spacing
SPR	Maximum spread
SVM	Support vector machine
TSC	Two-set coverage
UK	The United Kingdom
US	The United States
XGB	Extreme gradient boosting

Chapter 1

Introduction

The recent rise of machine learning (ML) and the rapid digitization of the economy has substantially changed decision processes in many domains, including the financial industry. Financial institutions increasingly rely on ML and artificial intelligence (AI) to support resource allocation decisions, inform risk management and automate operational decision-making. One of the prominent finance areas heavily affected by recent developments in ML and AI is credit scoring.

Credit scoring refers to the task of determining the creditworthiness of an individual or a company applying for credit. Leveraging the available data on potential borrowers, financial institutions use ML to guide loan approval decisions and risk management [10, 24]. To distinguish between defaulters and non-defaulters, financial institutions develop and deploy data-driven binary scoring models, also known as scorecards. The scorecards are usually based on supervised ML classification algorithms that predict the probability of default (PD) – an applicant's willingness and ability to repay debt in a defined time period [1]. The scorecard predictions (i.e., the credit scores) serve as a proxy for the applicant's creditworthiness and determine the loan approval decisions.

Focusing on consumer credit scoring, this dissertation speaks to recent challenges at the interface of financial decision-making and ML. The retail credit sector is of considerable economic importance. In 2021, the total outstanding amount of consumer credit in the US exceeded \$4,361 billion[1]. An increasing number of traditional banks are starting to use data-driven scorecards that have played a major role in the approval of this amount of credit. Furthermore, financial technology companies (FinTechs) that heavily rely on a data-driven business model and the automation of loan approval have substantially increased their market share from 22.4% in 2015 to 49.4% in 2019[2]. These trends indicate that the prevalence of ML-based credit scoring is expected to increase even further.

The availability of data on potential borrowers and recent ML advancements, including novel classification methods, facilitate the widespread use of ML for credit scorecards [18]. Automation of loan approval decisions through data-driven algorithms creates opportunities for increasing the scorecard accuracy. At the same time, reliance on ML at a large scale creates novel challenges that can affect the financial institution's profitability and have an adverse impact on the accuracy and fairness of the estimated risk scores, deteriorating the

[1]Source: The Federal Reserve (2021) Statistical Release on Consumer Credit, `https://www.federalreserve.gov/releases/g19/current`.

[2]Source: Experian (2019) Fintech vs. Traditional FIs: Trends in Unsecured Personal Installment Loans, `https://go.experian.com/IM-20-EM-AA-FintechTrendseBook?cmpid=fintech-trends-eBook-press-release`.

quality of the resulting loan approval decisions. The risk scores produced by scoring models increasingly determine access to finance, which plays a crucial role in economic inequality [27] and extends the potential impact of their quality beyond the accuracy of individual loan approval decisions. Further applications of conceptually similar scoring models for other behavioral scoring tasks such as estimating state transitions in mortgage portfolios [22] and corporate lending [16] further emphasize the growing concerns about some challenges arising from the use of ML for the scorecard development.

This dissertation examines practices underneath scorecard construction in consumer credit scoring and identifies three major challenges associated with building ML-based scorecards: (i) optimizing the data acquisition and storage costs when dealing with high-dimensional data of loan applicants; (ii) addressing the adverse effects of sampling bias on training and evaluation of scoring models; (iii) measuring and ensuring the fairness of scoring models while maintaining high profitability. The thesis overviews the adverse consequences and develops novel methodologies to address each of these three challenges. We empirically test the proposed remedies on real-world consumer credit scoring data sets. The thesis addresses its goals through the six essays summarized in Figure 1. We start with three chapters focusing on the data costs, follow with two chapters on sampling bias mitigation and conclude with a chapter on algorithmic fairness.

Chapters 2 – 4 consider the task of optimizing the data acquisition and storage costs. Due to the unprecedented availability of data on potential credit applicants and growing

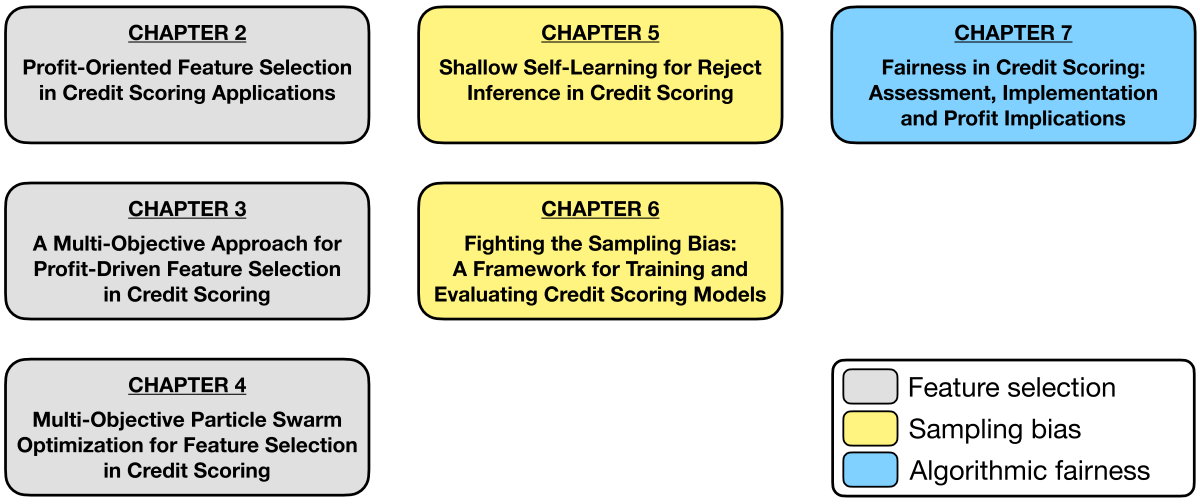

Figure 1. Thesis Structure

The figure depicts the structure of six essays constituting the thesis. Chapters 2 – 4 consider the problem of data acquisition and storage costs and address this problem by developing profit-driven feature selection frameworks. Chapters 5 – 6 focus on the problem of sampling bias and propose techniques for mitigating its adverse effects on the training and evaluation of scoring models. Chapter 7 considers the problem of fair credit scoring, catalogs and benchmarks suitable fair machine learning techniques, and investigates the profit-fairness trade-off.

access of financial institutions to newly emerging, non-traditional data sources, the collected customer data tend to be high-dimensional [6, 23]. Relying on data-driven scoring models requires financial institutions to face the costs of gathering and storing these large amounts of data on customer behavior. Features describing the customers are often purchased from third-party providers, which incurs additional data acquisition costs [20]. At the same time, companies are required to comply with regulations (i.e., the Basel Accords and IFRS 9) that enforce comprehensible scoring models. Chapters 2 – 4 suggest novel feature selection strategies to address the problems associated with the increasing data dimensionality. By removing irrelevant and redundant features, feature selection can reduce data acquisition and storage costs, improve the scorecard profitability and interpretability.

The standard feature selection techniques established in the ML literature use statistical criteria such as correlation or mutual information as a criterion for identifying a suitable subset of features [14]. In contrast, recent credit scoring literature criticizes a widespread practice of using standard performance measures such as the area under the receiver operating characteristic curve (AUC) for evaluating scoring models [15] and call for profit-based performance indicators [13, 25]. Chapter 2 makes the first step towards profit-oriented feature selection in credit scoring. The chapter extends the use of profit measures to the feature selection stage and develops a simple wrapper-based feature selection framework that uses the Expected Maximum Profit measure (EMP) as a fitness function [25]. An empirical study comprising multiple data sets demonstrates that the proposed framework identifies variable subsets that yield a higher expected profit compared to standard feature selection methods. The standard practice of using statistical measures such as AUC leads to scorecards with lower profitability, which calls for implementing the profit maximization as one of the objectives on different stages of the model development. These results stress the importance of using the business-inspired metrics for feature selection and serve as a basis for the work presented in the following two chapters.

Feature selection is usually considered as a single-objective task [14]. However, financial institutions may have multiple business-related goals that can be incorporated in the ML model development pipeline [20]. Recent studies in other domains beyond credit scoring have also demonstrated the importance of accounting for multiple objectives on the feature selection stage [7]. Thus, Chapters 3 and 4 treat feature selection as a multi-objective optimization task. In addition to maximizing the model profitability, financial institutions strive to reduce the number of features used to score prospective customers, as public discourse and regulatory requirements call for comprehensible credit scoring models. The conflicting nature of these two objectives motivates us to formulate feature selection as a multi-objective optimization problem with two fitness functions.

Chapter 3 contributes to credit scoring research in two ways. Building on the results presented in Chapter 2, it proposes a novel feature selection method that selects features in a profit-maximizing manner using the EMP as one of the two objectives. The second

objective is the cardinality of the feature set used in the scoring model, which serves as an indicator of model comprehensibility and data-related costs: minimizing the number of features reduces costs on data acquisition and storage and makes the model more comprehensible [21]. To simultaneously address both objectives, we employ a multi-objective non-dominated sorting-based genetic algorithm (NSGA-II) [11] with two fitness functions. The proposed method generates a frontier of non-dominated solutions representing a trade-off between two conflicting objectives. The resulting frontier serves as a tool to find a trade-off in model comprehensibility and profitability. By comparing the non-dominated solutions on the frontier, risk managers can select a suitable subset of features depending on the business context.

Extensive empirical experiments on ten real-world consumer credit scoring data sets indicate that the proposed framework identifies feature subsets that yield the same or higher expected profit using fewer features than single-objective feature selection methods on most data sets. Depending on a base classifier, solutions selected by the NSGA-II are not dominated by any of the considered single-objective benchmarks in 90% to 100% of cases. The results imply that previous studies that have ignored the two objectives of feature selection in credit scoring may have missed promising solutions identified using the suggested framework.

Chapter 4 builds on the previous two chapters by further extending the work on multi-objective feature selection. The usage of genetic algorithms such as NSGA-II has been recently challenged by the proposal of particle swarm optimization (PSO) techniques that demonstrate a superior performance [29, 30]. The chapter adopts a state-of-the-art external archive-guided PSO algorithm proposed by Zhu et al. [31] to perform the feature search in the credit scoring setup. A common practice of purchasing data in groups of features implies that a cost is charged once for a group of features, and there is no additional cost for each of the individual features. This practice reduces the correlation between the number of features and their acquisition cost, which provides an opportunity for multi-criteria optimization. Therefore, the chapter considers data acquisition costs as a distinct third objective. The number of features serves as a proxy for model comprehensibility and interpretability, whereas feature costs indicate the data acquisition costs faced by a financial institution. Therefore, we perform feature selection using three fitness functions reflecting relevant credit scoring objectives: the number of features, data acquisition costs, and model performance.

The performance of the proposed PSO framework is assessed on nine real-world credit scoring data sets. The results suggest that the developed PSO method is a highly competitive multi-objective feature selection framework, as indicated by multiple standard quality criteria for multi-objective optimization such as hypervolume, spread, and other metrics. Compared to other evolutionary algorithms, the proposed framework more effectively explores regions of the search space associated with a high model performance while also substantially reducing the number of features and the data acquisition costs compared to a model using all features.

Chapters 5 – 6 focus on the problem of sample selection bias. Credit scoring models are trained on the data of previously granted credit applications, where the borrowers' repayment behavior has already been observed. The binary labels indicating whether the applicant has repaid the loan are only available for the previously accepted applications. The data on rejected applicants that were denied credit do not enter the modeling pipeline and are ignored during model development. This implies that the training data offer only a partial picture of the distribution of candidate borrowers to which the model is applied when screening new credit applications [4]. The labels of rejected clients are either missing at random (MAR) or not at random (MNAR), which leads to sampling bias [19]. The sampling bias negatively affects two key elements of the scorecard development pipeline: model evaluation and model training. Reject inference comprises techniques to overcome sampling bias through assigning labels to rejected cases. Chapters 5 – 6 suggest novel reject inference strategies to address the adverse effects of sampling bias on credit scorecards.

Chapter 5 illustrates the sampling bias problem in credit scoring and takes the first step towards mitigating its adverse effects. Training a scorecard on a biased sample results in a performance loss when the model is applied to screen new applications. Previous research has tested some bias correction techniques, including the Heckman model, mixture models, and different data augmentation techniques, and concluded that gains from reject inference are little or non-existent [2, 8]. At the same time, only a few studies express performance gains in terms of profitability [e.g., 8] or have access to a proper representative evaluation sample to measure gains from reject inference [e.g., 3]. Chapter 5 introduces a novel self-learning based bias correction framework aimed at mitigating the impact of sampling bias on the scorecard performance. The framework can be applied to any machine learning algorithm to improve model training under sampling bias.

During the process of updating or replacing a scoring model, a financial institution typically compares the performance of the existing model to the performance of one or more challenger models. Reliable evaluation is important for model selection. Traditional performance measures such as AUC require application labels, which are not available for rejected clients. Assessing a scorecard on a sample of accepts filtered by the previous scorecards may provide a misleading, overoptimistic performance estimate [4]. As a result, the performance of the model that is eventually selected does not meet the expectations raised during model validation. Moreover, the bias in the performance estimates can vary a lot across models, which can lead to selecting a model with inferior performance. Focusing on the model selection application, Chapter 5 introduces a new evaluation measure denoted as the *kickout* metric. Our measure leverages domain knowledge to avoid an artificial labeling of rejected cases during evaluation and facilitates more reliable scorecard selection.

Experiments on a unique real-world credit scoring data set confirm the superiority of the suggested self-learning framework over previous bias correction strategies. The data set includes a rarely available sample of applications that were randomly accepted without

scoring. This sample represents the operating conditions of a scorecard and allows us to uncover the true merit of our propositions. We also find strong evidence in favor of the proposed evaluation measure providing a more accurate ranking of the scoring models, which improves the model selection and raises the performance of the eventual scoring model.

Chapter 6 builds on the positive results demonstrated in Chapter 5 and substantially extends the analysis in multiple distinct ways. First, the chapter introduces a novel Bayesian evaluation framework that addresses the impact of sampling bias on model evaluation. The *kickout* metric suggested in Chapter 5 improves model selection but does not allow assessing the expected model performance directly. Accurate model evaluation is important for judging the model's business value and informing long-term planning and risk assessment decisions. The Bayesian framework addresses this goal by allowing a risk manager to calculate an arbitrary performance measure on a representative sample from the borrowers' population that includes accepts and rejects. Drawing on prior knowledge, our framework avoids dependence on the actual labels of rejects and facilitates accurate scorecard evaluation under sampling bias.

Second, Chapter 6 introduces multiple modifications to the self-learning based bias correction algorithm proposed in Chapter 5. The extended framework is denoted as Bias-aware self-learning (BASL). Taking a closer look at the trade-off between bias reduction and scorecard accuracy, we aim at ensuring that the training data are only augmented with a few rejects, for which the labeling model is confident, and the data distribution is not too different from accepts. By doing so, we reduce sampling bias while keeping the error propagation sufficiently low. The extensive empirical analysis demonstrates that the implemented modifications raise the performance of the resulting scorecard.

The sampling bias correction methods suggested in Chapter 6 are tested on synthetic and real-world data. First, we set up a controllable simulated environment where the labels of rejects are known. We develop a synthetic data generation algorithm that mimics the real-world loan approval cycle supported by a scoring model. Using our simulation environment, we illustrate the sampling bias and its adverse impact on the scorecard training and evaluation. The simulation study also allows us to investigate the boundary conditions that influence the magnitude of the loss due to bias and the performance gains from our propositions. Second, using the same unique high-dimensional microloan data set introduced in Chapter 5, we compare the proposed methods to a rich set of the established bias correction benchmarks from different domains. Empirical results confirm the superiority of our propositions over previous work in terms of predictive performance and profitability.

Chapter 7 focuses on another crucial aspect of ML-based credit scoring. The rise of algorithmic decision-making has spawned much research on fair ML. The algorithmic fairness is commonly considered through the lens of differences in model predictions for various groups of individuals distinguished by a certain attribute such as gender, religious denomination, or ethnic group [5]. Yet, the literature on the fairness of the scorecard-based loan approval

decisions remains scarce. The chapter addresses this gap with three contributions. First, we revisit statistical fairness criteria established in the fair ML literature and examine their adequacy for credit scoring. We find that multiple fairness criteria can be approximately satisfied at once and recommend separation as a proper criterion for measuring the scorecard fairness. Separation acknowledges the imbalanced misclassification costs, which are instrumental to the lending business.

The fair ML literature has developed a variety of fairness processors to incorporate fairness goals in the model development pipeline. The complexity between these processors varies considerably, from simply relabeling the predictions [e.g., 17] to using deep learning to built a discrimination-free classifier [e.g., 28]. Chapter 7 systematically catalogs established fairness processors and benchmarks them in a profit-oriented credit scoring setup using seven real-world data sets. We find that selecting an appropriate fairness processor depends on the implementation feasibility and preferences of a decision-maker regarding the conflicting objectives of profit and fairness. Post-processing methods are the easiest to implement but improve fairness at a high monetary cost. In-processors perform best in finding the profit-fairness trade-off but require replacing a currently used scoring model with a new algorithm, which might require regulatory approval and is associated with considerable efforts.

While investigating the profit-fairness trade-off, we find that achieving perfect fairness is costly, but reducing discrimination to a reasonable extent is possible without sacrificing too much profit. These results support the current anti-discrimination regulation that allows unfairness to exist up to a certain limited extent. The analysis of fairness processors from the perspective of the Pareto frontiers offers decision-makers a tool to analyze the profit-fairness trade-off specific to their context and identify modeling techniques that reduce discrimination to a required level at the smallest monetary cost.

Each of the three challenges identified in the thesis represents a distinct and highly relevant problem for credit scoring researchers and practitioners. The methodologies proposed in the thesis can be used to tackle these challenges on a standalone basis. The feature selection techniques proposed in Chapters 2 – 4 offer a suitable framework for incorporating multiple business-driven objectives in the feature selection stage in order to account for the conflicting objectives of reducing the data acquisition and storage costs, improving the scorecard performance and profitability, and ensuring its comprehensibility and interpretability. The bias correction methods proposed in Chapters 5 – 6 help to mitigate the sampling bias arising from the use of scoring models and improve model training and evaluation by taking advantage of the data of rejected applicants. Finally, Chapter 7 identifies suitable methods to measure and implement the fairness goals in the scorecard development pipeline and investigates the profit-fairness trade-off. Combined, the six essays constituting the thesis offer a set of tools that can improve decision-making practices in financial institutions, increasing the resulting profit and improving the overall quality of the loan approval decisions.

Bibliography

[1] Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society, 54*(6), 627–635.

[2] Banasik, J., Crook, J. (2005). Credit scoring, augmentation and lean models. *Journal of the Operational Research Society 56*(9), 1072–1081.

[3] Banasik, J., Crook, J. (2007) Reject inference, augmentation, and sample selection. *European Journal of Operational Research 183*(3), 1582–1594.

[4] Banasik, J., Crook, J., Thomas, L. (2003). Sample selection bias in credit scoring models. *Journal of the Operational Research Society 54*(8), 822–832.

[5] Barocas, S., Hardt, M., Narayanan, A. (2019). *Fairness and Machine Learning*. fairmlbook.org.

[6] Biatat, V.A.D., Crook, J., Calabrese, R., Hamid, M. (2021). Enhancing credit scoring with alternative data. *Expert Systems with Applications*, 163.

[7] Bidgoli, A.A., Ebrahimpour-Komleh, H., Rahnamayan, S. (2019). A many-objective feature selection algorithm for multi-label classification based on computational complexity of features. *Proc. 2019 14th International Conference on Computer Science & Education (ICCSE)*, 85–91.

[8] Chen, G.G., Astebro T. (2001). The economic value of reject inference in credit scoring. *Proc. 7th Credit Scoring and Credit Control Conference*, 309–321.

[9] Crook J., Banasik J. (2004). Does reject inference really improve the performance of application scoring models? *Journal of Banking & Finance 28*(4), 857–874.

[10] Crook, J., Edelman, D., Thomas, L. (2007). Recent developments in consumer credit risk assessment. *European Journal of Operational Research, 183*(3), 1447–1465.

[11] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

[12] Feelders, A.J. (2000). Credit scoring and reject inference with mixture models. *Intelligent Systems in Accounting, Finance and Management Decision 9*(1), 1–8.

[13] Finlay, S. (2010). Credit scoring for profitability objectives. *European Journal of Operational Research, 202*(2), 528–537.

[14] Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A. (2008). *Feature extraction: Foundations and applications.* Springer.

[15] Hand, D. J. (2005). Good practice in retail credit scorecard assessment. *Journal of the Operational Research Society, 56*(9), 1109–1117.

[16] Hilscher, J., Wilson, M. (2016). Credit ratings and credit risk: Is one measure enough? *Management Science 63*(10), 3414–3437.

[17] Kamiran, F., Karim, A., Zhang, X. (2012). Decision theory for discrimination-aware classification. *Proc. International Conference on Data Mining*, 924–929.

[18] Lessmann, S., Baesens, B., Seow, H. V., Thomas, L. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

[19] Little, R.J., Rubin, D.B. (2019). *Statistical analysis with missing data.* John Wiley & Sons.

[20] Maldonado, S., Pérez, J., Bravo, C. (2017). Cost-based feature selection for support vector machines: An application in credit scoring. *European Journal of Operational Research, 261*(2), 656–665.

[21] Poursabzi-Sangdeh, F., Goldstein, D. G., Hofman, J. M., Vaughan, J. W., Wallach, H. (2017). Manipulating and measuring model interpretability. *Proc. NIPS 2017 Transparent and Interpretable Machine Learning in Safety Critical Environments Workshop.*

[22] Sadhwani, A., Giesecke, K., Sirignano, J. (2020). Deep learning for mortgage risk. *Journal of Financial Econometrics, 19*(2), 313–368.

[23] Sirignano, J., Giesecke, K. (2019). Risk analysis for large pools of loans. *Management Science, 65*(1), 107–121.

[24] Thomas, L., Edelman, D., Crook, J. (2002). *Credit Scoring and its Applications.* Philadelphia: SIAM.

[25] Verbraken, T., Bravo, C., Weber, R., Baesens, B. (2014). Development and application of consumer credit scoring models using profit-based classification measures. *European Journal of Operational Research, 238*(2), 505–513.

[26] Verbraken, T., Verbeke, W., Baesens, B. (2013). A novel profit maximizing metric for measuring classification performance of customer churn prediction models. *IEEE Transactions on Knowledge and Data Engineering, 25*(5), 961–973.

[27] Wei, Y., Yildirim, P., Van den Bulte, C., Dellarocas, C. (2016). Credit scoring with social network data. *Marketing Science 35*(2), 234–258.

[28] Zhang, B. H., Lemoine, B., Mitchell, M. (2018). Mitigating unwanted biases with adversarial learning. *Proc. AAAI/ACM Conference on AI, Ethics, and Society*, 335–340.

[29] Zhang, Y., Gong, D.wW, Cheng, J. (2015). Multi-objective particle swarm optimization approach for cost-based feature selection in classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 14*(1), 64–75.

[30] Zhang, Y., Gong, D.W., Sun, X.Y., Guo, Y.N. (2017). A PSO-based multi-objective multi-label feature selection method in classification. *Scientific Reports, 7*(1), 1–12.

[31] Zhu, Q., Lin, Q., Chen, W., Wong, K.C., Coello, C.A.C., Li, J., Chen, J., Zhang, J. (2017). An external archive-guided multiobjective particle swarm optimization algorithm. *IEEE Transactions on Cybernetics, 47*(9), 2794–2808.

Chapter 2

Profit-Oriented Feature Selection in Credit Scoring Applications

Publication

Kozodoi, N., Lessmann, S., Baesens, B., & Papakonstantinou, K. (2019). Profit-Oriented Feature Selection in Credit Scoring Applications. In Operations Research Proceedings 2018 (pp. 59-65). Springer, Cham.

Abstract

In credit scoring, feature selection aims at removing irrelevant data to improve the performance of the scorecard and its interpretability. Standard feature selection techniques are based on statistical criteria such as correlation. Recent studies suggest that using profit-based indicators for model evaluation may improve the quality of scoring models for businesses. We extend the use of profit measures to feature selection and develop a wrapper-based framework that uses the Expected Maximum Profit measure (EMP) as a fitness function. Experiments on multiple credit scoring data sets provide evidence that EMP-maximizing feature selection helps to develop scorecards that yield a higher expected profit compared to conventional feature selection strategies.

2.1 Introduction

One of the most important tasks in credit risk analytics is to decide upon loan provisioning. Binary scoring systems are widely deployed to support decision-making and predict applicants willingness and ability to repay debt. Financial institutions face costs of gathering and storing large amounts of data on customer behavior used to score applicants. In addition, companies need to comply with regulation that enforces comprehensible models. Feature selection aims at solving this problem by removing irrelevant data, which can reduce costs and improve the scorecard performance and interpretability.

Recent literature criticized a widespread practice of using standard performance measures such as area under the receiver operating characteristic curve (AUC) for evaluating scoring models [6]. Relying on profit-based indicators may improve scorecard profitability [4, 14]. This finding stresses the importance of using value-oriented feature selection strategies that identify the optimal subset of variables in a profit-maximizing manner. The goal of this paper is to introduce the profit maximization framework to the feature selection stage to facilitate the business-driven model development.

We develop a wrapper-based feature selection framework that uses the Expected Maximum Profit measure (EMP) as a fitness function. EMP has been previously used in credit scoring for model evaluation [14]. The advantage of the proposed approach is that it searches for variable subsets that optimize the business-inspired profitability indicator. To validate the effectiveness of our method, we conduct an empirical experiment on multiple consumer credit scoring data sets.

The remainder of this paper is organized as follows. Section 2 reviews the related literature on profit-driven credit scoring and feature selection methods. Section 3 describes our experimental setup, whereas Section 4 presents empirical results. In Section 5, we discuss the main conclusions of our study.

2.2 Related Literature

2.2.1 Profit-Oriented Credit Scoring

The credit scoring literature has proposed several profit measures to improve the quality of scorecards. Serranco-Clinca et al. use the internal rate of return based on the loan interest [12]. Finlay proposes estimating a contribution of each applicant to the profit of the financial institution [4]. Both these measures imply replacing binary default indicator by a continuous target variable and therefore transform a classification problem into the regression task.

Recently, Verbraken and colleagues developed the EMP measure [14]. EMP is based on the costs of the incorrect classification of *bad* loans (defaults) and benefits of the correct prediction of *good* ones (repayers). It can be computed as:

$$\text{EMP} = \int_0^1 \Big[B \cdot \pi_0 F_0(t) - C \cdot \pi_1 F_1(t)\Big] f(B) d(B), \tag{2.2.1}$$

where B is the expected loss in case of default and C is the return on the investment, π_i are prior probabilities of *good* and *bad* loans, and $F_i(t)$ are predicted cumulative fractions of class i based on cutoff t. The return on investment is assumed to be constant, whereas the expected loss is a stochastic variable based on the loss given default and exposure at default (see Verbraken et al. [14] for details).

EMP can be interpreted as the incremental profit from deciding on credit applications using a scorecard compared to a baseline scenario where credits are granted without screening. In this paper, we use the EMP criterion to measure the scorecard profitability.

2.2.2 Feature Selection

Feature selection methods split into filters, wrappers and embedded methods [5]. Filters rank and select features based on some general data characteristics. Popular measures in-

clude correlation, information gain and others [11]. Filters are fast and efficient but they were shown to perform poorly compared to wrappers and embedded methods [5]. Embedded methods conduct feature selection simultaneously with the model training. One of the popular approaches is recursive feature selection using the SVM framework [11]. The drawback of embedded methods is that they can only be applied within a specific model.

Wrappers go through different feature subsets and select the optimal subset based on the model performance. Since evaluating all possible feature combinations is computationally expensive, research has suggested heuristic search strategies. Popular approaches are sequential forward selection (SFS) and sequential backward selection (SBS) [5]. SFS starts with an empty model and iteratively adds features, selecting the one which brings the largest performance gain, whereas SBS starts with a full set of features and eliminates those contributing the least to the model performance. The search is continued until there is no further improvement. Another strategy relies on evolutionary algorithms such as genetic algorithms (GA) [15]. GAs operate on a population of individuals, where each individual represents a model with binary genes indicating inclusion of specific features. At each generation, a new population is created by selecting individuals according to their fitness (model performance), recombining them together and undergoing mutation. The individual with the highest fitness is selected after running multiple generations.

The literature on profit-oriented credit scoring focuses on model selection and parameter estimation but does not consider the feature selection stage. Existing studies on value-driven feature selection focus on feature costs. Some researchers suggest using a budget constraint that limits the maximal cost of the selected features [10]. Another approach is to use cost-adjusted ranking criteria when applying filter methods [3].

To the best of our knowledge, research on value-driven feature selection in credit scoring is currently limited to the embedded regularization framework for SVM [8, 9]. Recent benchmarking studies in credit scoring have shown that SVM performs poorly in comparison with other classifiers [7]. Given these results, developing a profit-driven feature selection approach that is not limited to SVM contributes to the literature. In this paper, we focus on wrappers due to their flexibility and better performance compared to filters.

2.3 Experimental Setup

2.3.1 Data Sets

The empirical evaluations are based on ten retail credit scoring data sets. Data sets *australian* and *german* stem from the UCI Machine Learning Repository[1]. The data sets *pakdd*, *lendingclub* and *gmsc* were provided by different financial insitutions for the data mining

[1]Source: `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`, `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`

Table 2.3.1. Credit Scoring Data Sets

Data set	Sample size	No. features	Default rate
australian	690	42	0.4449
german	1,000	61	0.3000
thomas	1,225	28	0.2637
bene1	3,123	83	0.3333
hmeq	5,960	20	0.1995
bene2	7,190	28	0.3000
uk	30,000	51	0.0400
lendingclub	43,344	206	0.1351
pakdd	50,000	373	0.2608
gmsc	150,000	68	0.0668

competitions on PAKDD[2] and Kaggle[3]. Datasets *bene1*, *bene2* and *uk* were collected from financial institutions in the Benelux and UK [2]. The *thomas* data set is provided by [13], whereas *hmeq* was collected by [1].

Each of the data sets has a unique set of features describing the loan applicant (e.g., gender, income) and loan characteristics (e.g., amount, duration). Some data sets also include information on previous loans of the applicant. The target variable is a binary indicator whether the customer has repaid the loan or not. Table 2.3.1 summarizes the main characteristics of the data sets.

2.3.2 Modeling Framework

The modeling pipeline consists of several stages. First, each data set is preprocessed in the same way. We impute missing values with means for continuous features and with modes for categorical features. Next, we encode categorical variables with $k-1$ dummies, where k is the number of unique categories.

The data sets are randomly partitioned into two subsets: training (70% cases) and holdout data (30%). On the training set, we use 5-fold cross-validation to perform feature selection. Then, we use the whole training set to train classification models with the identified feature subsets and evaluate results on the holdout data. The partitioning is repeated 10 times on each data set.

On a feature selection stage, we use three wrappers: SFS, SBS and GA. The parameters of GA were selected based on the grid search on a subset of training data: number of generations and number of individuals were set to 200. For each of the feature selection algorithms, we use two performance measures as objective functions. Relying on EMP as a fitness function is a central element of our approach, whereas using AUC serves as a benchmark for the

[2]Source: `https://www.kdnuggets.com/2010/03/f-pakdd-2010-data-mining-competition.html`
[3]Source: `https://www.lendingclub.com`, `https://kaggle.com/c/givemesomecredit`

standard feature selection techniques. Logistic regression is used as a base classifier.

2.4 Empirical Results

The performance of different methods is compared by the mean model ranks in terms of AUC and EMP. To compute the ranks, we order all algorithms by AUC and EMP values within each modeling trial on each of the data sets, and average the model positions. The results are depicted in Figure 2.4.1.

Results suggest that EMP-based wrappers identify feature subsets that yield a higher EMP on the holdout data, whereas using AUC as objective leads to models with a higher AUC. For all three wrappers, EMP optimization during feature selection generalizes to a higher expected profit on the new data. Therefore, our results emphasize the importance of selecting the appropriate fitness function in the early stages of scorecard development. If the goal of the scoring model is to maximize the expected profit, this measure should also be used as an objective for feature selection. Relying on AUC as one of the standard performance measures results in a suboptimal scorecard with a lower EMP.

We also observe differences in terms of the model complexity. For the sequential methods, EMP-driven wrappers reach the stopping criteria earlier, resulting in a lower average number of the selected features for SFS (14 compared to 25) and a higher number of features for SBS (89 compared to 81). These results provide evidence that using profit-driven fitness function may also lead to a faster convergence of the feature selection algorithms, which is important

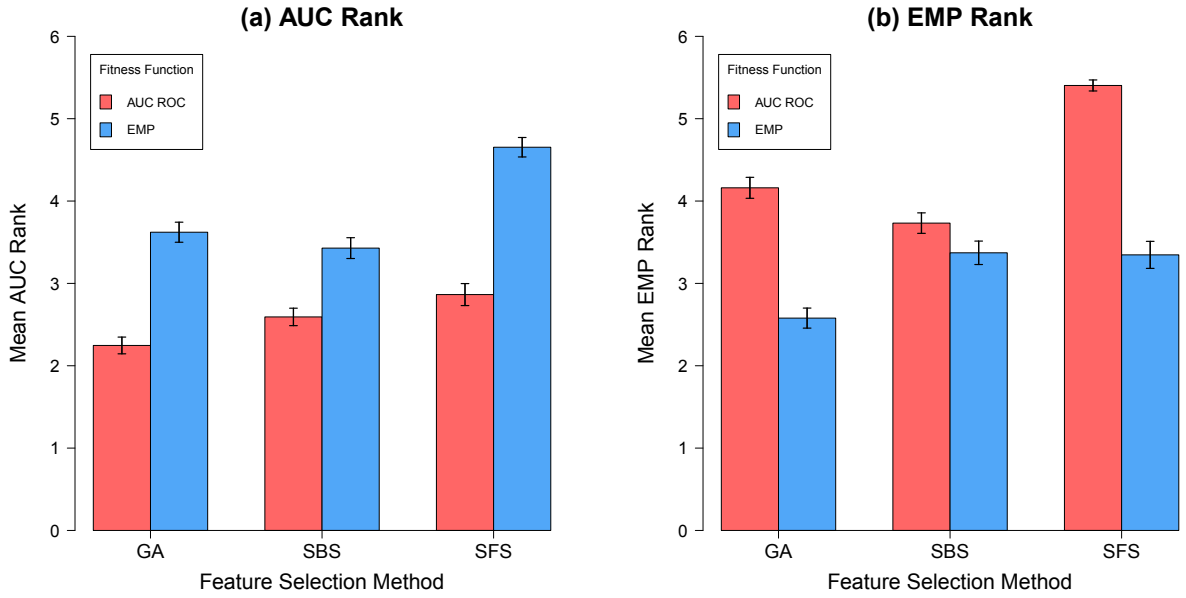

Figure 2.4.1. Mean Ranks of Feature Selection Methods

The figure displays mean ranks in terms of AUC (left) and EMP (right) aggregated across 10 trials on 10 data sets. Whiskers indicate the intervals of one standard error around the mean. Abbreviations: GA = genetic algorithm, SBS = sequential backward selection, SFS = sequential forward selection.

for reducing the computational time.

2.5 Conclusion

This paper presents a profit-driven framework for feature selection in credit scoring. We use the recently developed EMP measure as a fitness function for wrapper-based feature selection. The effectiveness of our approach is evaluated on ten real-world retail credit scoring data sets.

Empirical results indicate that the proposed profit-maximizing feature selection framework identifies variable subsets that yield a higher expected profit compared to methods based on standard performance measures. These results stress the importance of using the business-inspired metrics for feature selection. Relying on a standard practice of using statistical measures such as AUC may lead to scorecards with a lower profitability, which motivates implementing the profit maximization on different stages of the model development.

Future research could pursue several directions. For practitioners, it would be important to extend the profit-driven framework to other stages of model development. A benchmarking study with a rich set of EMP-based wrappers would help identifying the optimal search strategy for profit-driven feature selection. Another direction would be to use the developed approach in other business applications such as customer churn.

Bibliography

[1] Baesens, B., Roesch, D., Scheule, H. (2016). *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS.* John Wiley & Sons.

[2] Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society, 54*(6), 627–635.

[3] Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A. (2015). Recent advances and emerging challenges of feature selection in the context of big data. *Knowledge-Based Systems, 86*, 33–45.

[4] Finlay, S. (2010). Credit scoring for profitability objectives. *European Journal of Operational Research, 202*(2), 528–537.

[5] Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. A. (2006). *Feature Extraction: Foundations and Applications.* Springer.

[6] Hand, D. J. (2005). Good practice in retail credit scorecard assessment. *Journal of the Operational Research Society, 56*(9), 1109–1117.

[7] Lessmann, S., Baesens, B., Seow, H. V., Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

[8] Maldonado, S., Bravo, C., Lopez, J., Perez, J. (2017). Integrated framework for profit-based feature selection and SVM classification in credit scoring. *Decision Support Systems, 104*, 113–121.

[9] Maldonado, S., Pérez, J., Bravo, C. (2017). Cost-based feature selection for support vector machines: An application in credit scoring. *European Journal of Operational Research, 261*(2), 656–665.

[10] Min, F., Hu, Q., Zhu, W. (2014). Feature selection with test cost constraint. *International Journal of Approximate Reasoning, 55*(1), 167–179.

[11] Chandrashekar, G., Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering, 40*(1), 16–28.

[12] Serrano-Cinca, C., Gutiérrez-Nieto, B. (2016). The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending. *Decision Support Systems, 8*, 113–122.

[13] Thomas, L. C., Edelman, D. B., Crook, J. N. (2002). *Credit Scoring and its Applications.* Philadelphia: SIAM.

[14] Verbraken, T., Bravo, C., Weber, R., Baesens, B. (2014). Development and application of consumer credit scoring models using profit-based classification measures. *European Journal of Operational Research, 238*(2), 505–513.

[15] Yang, J., Honavar, V. (1998). Feature subset selection using a genetic algorithm. *Proc. Feature Extraction, Construction and Selection*, 117–136.

Chapter 3

A Multi-Objective Approach for Profit-Driven Feature Selection in Credit Scoring

Publication

Kozodoi, N., Lessmann, S., Papakonstantinou, K., Gatsoulis, Y., & Baesens, B. (2019). A multi-objective approach for profit-driven feature selection in credit scoring. Decision support systems, 120, 106-117.

Abstract

In credit scoring, feature selection aims at removing irrelevant data to improve the performance of the scorecard and its interpretability. Standard techniques treat feature selection as a single-objective task and rely on statistical criteria such as correlation. Recent studies suggest that using profit-based indicators may improve the quality of scoring models for businesses. We extend the use of profit measures to feature selection and develop a multi-objective wrapper framework based on the NSGA-II genetic algorithm with two fitness functions: the Expected Maximum Profit (EMP) and the number of features. Experiments on multiple credit scoring data sets demonstrate that the proposed approach develops scorecards that can yield a higher expected profit using fewer features than conventional feature selection strategies.

3.1 Introduction

Credit scoring refers to the use of statistical models that guide managerial decisions in the retail credit sector [12]. This sector has gained a considerable economic value: in 2017, consumer credit outstandings reached €1,195 billion in EU[1]. In the US, the total outstanding consumer credit amount exceeded $3,831 billion[2]. At the same time, the delinquency rate on consumer loans by commercial banks experienced a growth of more than 11% since 2015[3]. The rise of default rates emphasizes the importance of accurately deciding upon loan provisioning, which is a task of credit scoring. To distinguish defaulters and non-defaulters, financial institutions deploy binary scoring models (i.e., scorecards) that predict the probability of default (PD) – an applicant's willingness and ability to repay debt [38].

[1]Source: https://www.ca-consumerfinance.com/en/Espaces/Press-corner/Panorama-du-credit-a-la-consommation-en-Europe/Overview-of-consumer-credit-in-Europe-in-2016-Strong-growth-in-the-European-consumer-credit-market

[2]Source: https://www.federalreserve.gov/releases/g19/current/

[3]Source: https://fred.stlouisfed.org/series/DRCLACBS

Data-driven models, which are used to score applicants, require financial institutions to face costs of gathering and storing large amounts of data on customer behavior. At the same time, companies are required to comply with regulations (i.e., the Basel Accords and IFRS 9) that enforce comprehensible scoring models. By removing irrelevant and redundant features, feature selection can reduce costs and improve the model performance and its comprehensibility (interpretability).

Feature selection can be considered as a multi-objective problem with conflicting goals. In credit scoring, these goals are: increasing the model profitability, reducing the data acquisition costs and improving the interpretability of the model. Yet, most existing approaches in machine learning literature treat feature selection as a single-objective task [5, 11, 44].

Standard feature selection techniques use statistical criteria to identify the optimal subset of features. Recent credit scoring literature criticized a widespread practice of using standard performance measures such as area under the receiver operating characteristic curve (AUC) for evaluating scoring models [20] and call for profit-based performance indicators [15, 39]. This finding stresses the importance of using value-oriented feature selection strategies that identify the optimal subset of features in a profit-maximizing manner.

The goal of this paper is to design a feature selection framework for credit scoring that overcomes some of the drawbacks of traditional feature selection techniques. The proposed method selects features in a profit-maximizing manner rather than relying on statistical measures and addresses both profitability and comprehensibility with multi-criteria optimization. We use the recently developed Expected Maximum Profit (EMP) measure to evaluate the model profitability [39]. Previous research has applied EMP for model selection but did not consider profit maximization at the feature selection stage. We also use the number of features as an indicator of model comprehensibility and data-related costs: minimizing the number of features reduces costs on data acquisition and storage and makes the model more comprehensible [31]. To simultaneously address both objectives, we employ a multi-objective feature selection framework based on the non-dominated sorting-based genetic algorithm (NSGA-II) [13] with two fitness functions: EMP and the number of features. The proposed method generates a frontier of non-dominated solutions, which represents a trade-off between two objectives and can, therefore, aid decision-makers in selecting a suitable solution. To validate the effectiveness of our approach, we conduct empirical experiments on ten real-world credit scoring data sets.

The contribution of this paper is three-fold. First, we introduce a profit-centric feature selection framework by using the EMP measure as a fitness function, thereby extending the use of EMP to feature selection. Second, we employ a multi-objective feature selection framework based on the NSGA-II algorithm. To the best of our knowledge, the specific combination of multi-objective feature selection based on scorecard profitability and parsimony using NSGA-II is originally proposed here and extends previous work in the credit scoring literature. Third, we provide empirical evidence that the proposed multi-objective feature

selection technique identifies feature subsets that deliver the same or higher expected profit using fewer features than conventional feature selection strategies.

The remainder of this paper is organized as follows. Section 3.2 reviews related literature on feature selection methods and describes previous work on profit-driven credit scoring. In Section 3.3, we present and explain the proposed multi-objective feature selection framework. Section 3.4 describes our experimental setup and presents the empirical results. In Section 5, we discuss the main conclusions of our study.

3.2 Theoretical Background

3.2.1 Feature Selection

Feature selection is a dimensionality reduction technique that aims at selecting a subset of features from the input data by removing irrelevant, redundant or noisy features while maintaining the model performance [16]. Feature selection methods split into three groups: filters, wrappers and embedded methods [17].

Filters perform feature selection based on some general data characteristics before training the model. On the first stage, all features are ranked according to a certain criterion that describes the relevance of a particular feature. Popular measures include feature-to-target correlation [5], mutual information [11], Fisher score [8] and others. In the second stage, a certain percentage of the top-ranked features is selected, whereas features with lower importance are dropped from the model. Compared to other feature selection strategies, filters are fast and efficient. However, they were shown to perform poorly in benchmark studies [17].

Wrappers are algorithms that iteratively process different feature subsets and select the optimal subset based on the model performance. Since evaluating all possible feature combinations is computationally expensive, research has suggested multiple heuristic search strategies. Popular approaches are sequential forward selection (SFS) and sequential backward selection (SBS) [17]. SFS starts with an empty model and iteratively adds features, selecting the one which brings the largest performance gain, whereas SBS starts with a full set of features and eliminates those contributing the least to the model performance. The search is continued until there is no further improvement. Another strategy relies on evolutionary algorithms such as genetic algorithms (GA), particle swarm optimization (PSO) and others [45]. GAs operate on a population of individuals, where each individual represents a model with binary genes indicating the inclusion of specific features. At each generation, a new population is created by selecting individuals according to their fitness (model performance), recombining them together and undergoing mutation. The model with the highest fitness is selected after running the algorithm for multiple generations.

Embedded methods conduct feature selection simultaneously with the model training. One of the popular approaches is L1-regularized regression that performs feature selection

by assigning zero coefficients to irrelevant features in the process of the model development [41]. The main drawback of embedded methods is that they can only be applied within a specific model class.

Most existing feature selection techniques consider feature selection as a single-objective task. However, conflicting goals of feature selection (optimizing the model performance and minimizing the number of selected features) suggest that it can be treated as a multi-objective optimization problem. The literature on multi-objective feature selection is limited compared to the research on conventional single-objective techniques. Nevertheless, there exists a number of attempts to employ the multi-criteria optimization frameworks.

One of the approaches to perform multi-criteria feature selection is to convert a problem into a single-objective task by aggregating the weighted objectives into a single fitness function. For instance, Bolón-Canedo and colleagues propose adding a new term to the evaluation function of well-known filter methods such as correlation-based feature selection, Minimal-Redundancy-Maximal-Relevance and RelieF [6, 7]. The new term represents a number of features or their cost, which ensures that two objectives are included in the fitness function. A major downside of this approach is the requirement to explicitly assign weights to objectives, which is a challenging task given uncertainty and different scales of the objectives.

Another approach to account for multiple objectives is to consider a single-objective optimization problem with a budget constraint. In some studies, researchers suggest minimizing the number of features given that a certain level of performance is achieved [3, 32], whereas others optimize predictive performance under the budget constraint for the cost of included features [28]. Both these directions require setting a specific threshold to introduce a budget constraint, either for the model performance or for the number of used features. Therefore, the application of this approach is problematic in cases with no hard budget constraints.

A more promising strategy is to consider objectives separately and look for a set of non-dominated solutions that are optimal in terms of multiple objectives instead of focusing on a single solution. The set of non-dominated points is also known as the Pareto efficient frontier and represents points, for which one can not improve on one objective without decreasing the other. Literature proposed multi-objective modifications of the well-known evolutionary algorithms such as GA and PSO that rely on multiple fitness functions to perform a search of the non-dominated solutions. Emmanouilidis et al. used a two-objective genetic algorithm to perform feature selection that minimizes the number of features and optimizes the error rate or RMSE for classification and regression on different data sets [14]. More recent studies use modified versions of multi-objective genetic algorithms including the Strength Pareto Evolutionary Algorithm (SEPA-II) and the Non-Dominated Sorting Genetic Algorithm (NSGA-II) [35, 18] to perform feature selection with the same objectives. Research has also suggested using other evolutionary algorithms such as PSO [43] and Artificial Bee Colony (ABC) [19].

The first attempt to perform profit-driven feature selection has been applied in customer churn [24] within the embedded framework for holdout support vector machines (HOSVM), where the authors use multiple metrics for customer churn to select features. The authors also extended their approach to credit scoring [26] by introducing the L-infinity norm as a group penalty function to perform cost-based feature selection while training the SVM classifier. In [25], they also use the EMP measure to tune SVM parameters in a profit-maximizing manner. The studies conclude that the developed framework outperforms conventional feature selection techniques in terms of profit.

The approach proposed in this paper differs from the frameworks suggested in [25, 26] in two important dimensions. First, the latter balance three objectives: Euclidean norm minimization, group penalization for feature selection and hinge loss minimization. This way, the techniques in [25, 26] do not provide a Pareto frontier with non-dominated solutions in terms of the considered objectives. Producing a corresponding frontier of non-dominated solutions with respect to the trade-off between scorecard profitability and parsimony is a goal of this study. Insights into this trade-off will help risk analysts to make informed decisions how many variables to use for a scorecard, which, for example in the case where variables are purchased from external entities such as credit bureaus, has wider reaching benefits related to the costs of data acquisition. Second, the approaches proposed in [25, 26] qualify as embedded feature selection frameworks that can only be applied within an SVM classifier. Recent benchmarking studies in credit scoring suggest that alternative classifiers and tree-based ensemble methods in particular might perform better than SVMs in consumer credit scoring [23]. Given these results, developing a model-agnostic feature selection approach that can be used with any classifier and that facilitates optimizing both profitability and model comprehensibility contributes to the literature.

3.2.2 Profit-Oriented Credit Scoring

The credit scoring task is commonly expressed as a classification problem, where a predictive model learns to differentiate between *bad* risks (defaulters) and *good* risks (repayers). Traditional machine learning algorithms are designed to optimize statistical measures such as mean squared error. In recent years, credit scoring literature proposed different strategies to introduce the profit maximization to the scorecard development. One approach is to modify the target variable to reflect profitability. For instance, Serrano-Cinca et al. suggest using the internal rate of return based on the loan interest [33]. Finlay proposes estimating a contribution of each applicant to the profit of the financial institution [15]. Both these measures imply replacing a binary default indicator by a continuous target variable and therefore transform a classification problem into a regression task.

Another approach toward profit scoring is based on using profit-related performance measures for model selection. Recently, Verbraken and colleagues suggested the Expected

Table 3.2.1. Confusion Matrix with Costs

Actual Label	Predicted Label: Bad risk	Predicted Label: Good risk
Bad risk	$\pi_0 F_0(t)$ benefit: B	$\pi_0(1 - F_0(t))$ cost: 0
Good risk	$\pi_1 F_1(t)$ cost: C	$\pi_1(1 - F_1(t))$ cost: 0

Maximum Profit (EMP) measure [39]. The calculation of EMP is based on costs and benefits that arise as a result of the actions the company undertakes. To illustrate the calculation process, we follow their notation and label defaulters as class 0 and non-defaulters as class 1. The scorecard assigns a score to each applicant that expresses the probability of default. Applicants are then considered as *bad* risks and rejected if the estimated credit score exceeds a cutoff value t. Table 3.2.1 provides a confusion matrix with the corresponding class probabilities, where π_i are prior probabilities of *good* and *bad* loans, and $F_i(t)$ are predicted cumulative density functions of the scores of class i.

The EMP measure assumes that in the basic scenario no scoring mechanism is implemented and therefore all loans are granted. Hence, if an applicant is predicted as a *good* risk, no additional costs or benefits are observed. In contrast, if an applicant is predicted to be a defaulter, the company faces cost C in case of an incorrect prediction and gets benefit B from an accurate prediction. The methodology to calculate parameters B and C was developed by [9].

Parameter B is the benefit from correctly identifying a *bad* risk. By not providing a loan to a defaulter, the company saves money that would be lost in case of issuing the loan. This amount is the expected loss in case of default:

$$B = \frac{\text{LGD} \cdot \text{EAD}}{A}, \tag{3.2.1}$$

where LGD refers to the loss given default, EAD is the exposure at default, and A is the principal of the loan [27]. Since recovery rates for defaulted loans vary heavily [34], B is considered as a random variable, which can take values between 0 and 1. The following probability distribution is assumed:

- $B = 0$ with probability p_0 (a customer repays the entire loan)
- $B = 1$ with probability p_1 (a customer defaults on the entire loan)
- B follows a uniform distribution in (0, 1) with $F(B) = 1 - p_0 - p_1$

Parameter C is the cost of the incorrect classification of *good* risks. By rejecting a *good* customer, the company looses money that could be earned as return on investment:

$$C = \text{ROI} = \frac{I}{A}, \tag{3.2.2}$$

where I is the total interest. Verbraken et al. [39] treat parameter C as constant and that we follow their approach in this paper. Given these parameters, the EMP measure can be computed as:

$$\text{EMP} = \int_0^1 \Big[B \cdot \pi_0 F_0(t) - C \cdot \pi_1 F_1(t)\Big] f(B) d(B) \tag{3.2.3}$$

EMP can be interpreted as the incremental profit from deciding on credit applications using a scorecard compared to a baseline scenario where credits are granted without screening. In this paper, we use EMP to measure the profitability of the scorecard. Furthermore, we rely on the EMP measure as one of the optimization objectives to enable profit-driven feature selection.

The literature on profit-oriented credit scoring focuses on model selection and parameter estimation but does not pay sufficient attention to the feature selection stage. Current research on profit-driven feature selection in credit scoring is limited to the embedded regularization framework for SVMs [25, 26] described above. This paper proposes a model-agnostic profit-driven feature selection approach that optimizes both profitability and model comprehensibility.

3.3 Proposed Profit-Driven Feature Selection Approach

We treat feature selection as a multi-objective problem with two goals: a) maximizing the performance of the scorecard; b) minimizing the number of used features used by the model. We propose a wrapper method based on the binary multi-objective nondominated sorting based genetic algorithm (NSGA-II) with two fitness functions: EMP and number of features. The suggested approach addresses two issues with traditional feature selection techniques in credit scoring: it relies on a profit-driven indicator rather than statistical performance measures and addresses both profitability and model comprehensibility by employing multi-objective optimization.

NSGA-II is a multi-objective evolutionary algorithm developed by [13] to address disadvantages of the previous version of NSGA [36]. NSGA-II is designed to solve multi-objective optimization problems by finding a set of non-dominated solutions which form the efficient Pareto frontier. Experiments on different test problems have shown that NSGA-II is able to maintain a better spread of solutions and convergence compared to some other multi-objective optimizers [13].

NSGA-II consists of three main stages: fast non-dominated sorting, diversity preservation and population update. First, the initial population of n individuals is generated with

random gene values. In the case of feature selection, each individual represents a set of features included in the predictive model. We code a population of individuals with a set of binary genes with each gene representing the inclusion of a certain feature in the scorecard.

Second, we compute fitness values for the considered objective functions. For each individual, we construct a scoring model with a different set of features, which is defined by the gene values of these individuals. We evaluate the performance of the scorecard in terms of EMP and store EMP and the number of selected features as two fitness values.

On the next stage, the population goes through genetic operators: selection, crossover and mutation. The selection is performed with a binary tournament method based on the crowded comparison operator. First, we sort the population by a non-domination rank – the number of individuals dominated by a given solution in terms of the considered objective functions. Next, individuals with the same non-domination ranks are sorted by their crowding distance – the average distance of two solutions on either side of this individual along each of the objectives. Next, one-point crossover is applied to the remaining population. Gene values of the child are computed as a weighted average of the gene values of the parents. In a binary NSGA-II, which is the focus of this paper, a one-point crossover operator simply copies parents' genes if they are the same and randomly chooses a binary value for the conflicting genes. Finally, each gene of the child is flipped with a mutation probability m. These operations are performed until the size of the offspring population reaches n.

After applying all genetic operations, both parents and children are merged into the new population of size $2n$ to ensure elitism. The population is again sorted according to the non-domination and crowding distance. After the sorting is complete, only the top n individuals are selected to proceed to the next stage. This approach helps the algorithm to construct a uniformly spread-out Pareto-optimal frontier by eliminating solutions that are either dominated or located in the crowded regions of the frontier.

The NSGA-II algorithm was previously used for feature selection in fields not related to credit risk. The fitness functions considered in the literature are the number of features and statistical performance measures such as error rate or mean squared error [18, 30, 35]. In credit risk, NSGA-II has only been applied to a bank-loan portfolio selection problem [29], where the algorithm is used to optimize portfolio return and risk. In this paper, we rely on the NSGA-II algorithm to perform multi-objective feature selection for credit scoring. The central novelty of our approach is the use of a profit measure as one of the fitness functions within a multi-objective feature selection framework.

3.4 Experimental Results

3.4.1 Data Description

The empirical evaluations are based on ten retail credit scoring data sets coming from different sources. Data sets *australian* and *german* stem from the UCI Machine Learning

Table 3.4.1. Credit Scoring Data Sets

Data set	Sample size	No. features	Default rate
australian	690	42	0.4449
german	1,000	61	0.3000
thomas	1,225	28	0.2637
bene1	3,123	83	0.3333
hmeq	5,960	20	0.1995
bene2	7,190	28	0.3000
uk	30,000	51	0.0400
lending club	43,344	206	0.1351
pakdd	50,000	373	0.2608
gmsc	150,000	68	0.0668

Repository[4]. The data sets *pakdd*, *lendingclub* and *gmsc* were provided by different financial insitutions for the data mining competitions on PAKDD[5] and Kaggle[6]. Data sets *bene1*, *bene2* and *uk* were collected from financial institutions in the Benelux and UK [1]. The *thomas* data set is provided by [37]. Finally, *hmeq* is a data set on home equity loans collected by [2].

Each of the data sets has a unique set of features describing the loan applicant (e.g., gender, income) and loan characteristics (e.g., amount, duration). Some data sets also include information on previous loans of the applicant. The target variable is a binary indicator of whether the customer has repaid the loan or not. Table 3.4.1 summarizes the main characteristics of the data sets.

As suggested by Table 3.4.1, most of the data sets are imbalanced: default rate fluctuates between 4% and 44%. The sample size and number of features varies significantly across the data sets, which suggests that we use a heterogeneous data library for further analysis.

3.4.2 Experimental Setup

Our modeling pipeline consists of several stages. First, each data set is pre-processed in the same way. We impute missing values with means for continuous features and with most frequent values for categorical features. Next, we encode all categorical features with $k-1$ dummies, where k is the number of unique categories.

After preprocessing, the data sets are randomly partitioned into two subsets: training sample (70% cases) and holdout sample (30%). On the training set, we use 4-fold cross-validation to perform feature selection. Next, we use the whole training set to train scorecards

[4]Source: `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`, `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`

[5]Source: `https://www.kdnuggets.com/2010/03/f-pakdd-2010-data-mining-competition.html`

[6]Source: `https://www.lendingclub.com`, `https://kaggle.com/c/givemesomecredit`

with the identified feature subsets and evaluate their performance on the holdout data.

We use three base classifiers: extreme gradient boosting, logistic regression and L1-regularized logistic regression. This allows us to check the robustness of feature selection techniques across different predictive algorithms and see whether internal feature selection in models such as L1 regression diminishes the value of the proposed wrapper approach.

Before performing feature selection, we use a subset of the training data to tune meta-parameters of the base classifiers. For each of the considered classification algorithms, we perform a learning curve analysis to select a suitable sample size by gradually increasing the percentage of the training sample until the model performance in terms of EMP stops improving. Next, we use the corresponding subset to perform parameter tuning using grid search [4]. The full parameter grid is presented in Table 3.4.2.

The key meta-parameters of NSGA-II (number of generations and population size) were selected based on the experiments on the subset of training data. We compared three specifications (50×50, 100×100 and 200×200) in terms of maximal EMP. Based on these results, the number of generations and the population size were set to 200. After identifying suitable meta-parameter values, we perform feature selection with the suggested multi-objective framework.

As described in Section 3, the EMP measure depends on two parameters, which need to be specified in order to calculate EMP on a scorecard level. These parameters are the expected loss in case of default and return on investment. For data sets where this information is not available or cannot be deprived from available meta-data, we follow the empirical findings of [39] and assume that the loss given default follows a bimodal distribution with point masses $p_0 = 0.55$ for no loss and $p_1 = 0.1$ for full loss; we also follow [39] in assuming a constant return on investment of 0.2664. The selected values correspond to the default values provided in the R package for EMP estimation available at CRAN [10].

To evaluate the performance of the proposed algorithm, we compare it to five traditional feature selection strategies: SFS, SBS, LASSO, single-objective GA and single-objective binary PSO [42]. To ensure a fair comparison, we set the number of generations and number of individuals for the simple GA to the same values as for the NSGA-II, which results in

Table 3.4.2. Meta-Parameter Grid

Method	Parameter	Candidate values
LR	–	–
L1	cost	2^{-10}, $2^{-9.5}$, 2^{-9}, ... , 2^{10}
XGB	number of trees	10, 25, 50, 100, 250, 500, 1000, 2500
	learning rate	0.01, 0.03, 0.05
	maximum tree depth	1, 3, 5

Note: LR = logistic regression, L1 = L1-regularized LR, XGB = extreme gradient boosting.

the same total number of models trained within the algorithm. We also use a scorecard that relies on a full set of features as a benchmark. All five single-objective benchmarks use the EMP measure as a fitness function. We only consider wrapper methods as benchmarks because of their superior performance compared to other feature selection strategies [17].

Compared to other single-objective feature selection methods considered in the paper, the advantage of SFS and SBS is that they can also provide a Pareto frontier based on their path to the final solution. On each iteration, we save the best-performing variable subset and evaluate it on the holdout sample, thereby obtaining a set of non-dominated solutions.

3.4.3 Empirical Results

In this section, we start with the experimental results where logistic regression is used as a base model for all techniques and then focus on the aggregated results across different classifiers. Logistic regression is still widely used in practice [22, 37] despite that other algorithms have been shown to predict credit risk more accurately [23]. Detailed results for the other base classifiers including extreme gradient boosting and L1-regularized regression are given in Figures 3.6.1 and 3.6.2 in the Appendix.

Before moving to the empirical results, consider the example Pareto frontier depicted in Figure 3.4.1. Here, the task is to minimize objective I while maximizing objective II. The frontier is represented by points A to E, whereas points F, G and H are external solutions (benchmarks). Point H is dominated by points A to D on the Pareto frontier because they perform better in two objectives. Points F and G demonstrate better performance in terms of objective II compared to the best solution from the Pareto frontier (point E). However, there is a crucial difference between these points. Solution G does not dominate any points on the frontier – it achieves better performance in objective II only by deteriorating on objective

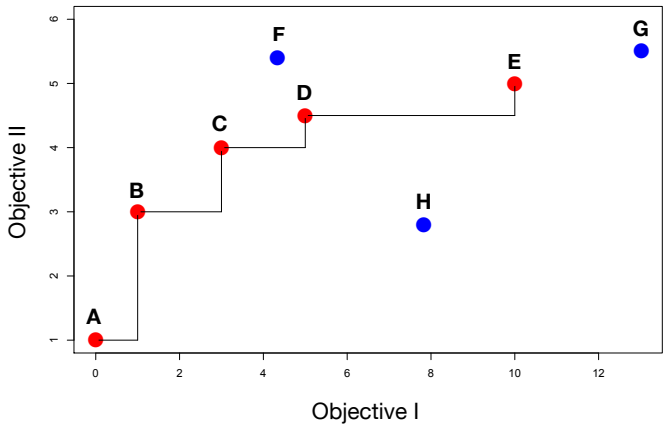

Figure 3.4.1. Example Multi-Objective Optimization

The example task is to minimize objective I while maximizing objective II. Points A – E represent solutions on the efficient frontier, points G, F and H are external solutions. Compared to the frontier, H is dominated by points A to D, G is a non-dominated point, and F dominates solutions D and E on the frontier.

I. At the same time, point F achieves better performance in both objectives compared to points D and E on the frontier. Therefore, F dominates these solutions. It is important to distinguish domination (point F) and non-domination (point G) when comparing the performance of different feature selection techniques.

Figure 3.4.2 presents the graph matrix with the performance of the considered feature selection methods on all ten data sets. The Pareto frontier identified by the NSGA-II algorithm is depicted with red markers, whereas other points represent the single-objective benchmarks. GA, PSO and LASSO provide single solutions, whereas for SBS and SFS we depict the Pareto frontiers obtained during the feature selection. The black cross marks the baseline solution which is based on a full model without feature selection.

Results indicate that the size of the NSGA-II Pareto frontier varies across the data library from having just two solutions (*thomas* and *bene1*) to 20 feature subsets (*pakdd*). The small size of the Pareto frontier can be explained by two reasons: first, no candidate solutions with a larger number of features demonstrate better performance during cross-validation; second, some solutions become dominated when evaluating their quality on the holdout data and are therefore dropped from the frontier. Hence, NSGA-II frontiers are likely to contain fewer solutions on data sets with lower dimensionality and stronger differences in data distribution between the training and holdout samples.

Overall, the points on NSGA-II frontiers usually populate regions with a smaller number of features compared to benchmarks. Single-objective methods optimize predictive performance but do not account for the number of features. This does not motivate the algorithm to select smaller feature subsets. Nevertheless, sequential forward selection chooses fewer features compared to sequential backward elimination on all ten data sets.

We also note that frontiers produced by SFS are more stable compared to SBS-based frontiers as they have more solutions that remain non-dominated after reevaluating performance on the holdout sample. According to Figure 3.4.2, SFS frontiers contain more solutions than NSGA-II frontiers on 6 data sets. Nevertheless, most points on SFS frontiers are dominated by the results obtained by NSGA-II. Below, we extend the comparison by focusing on the best-performing solutions from the frontiers.

To evaluate the quality of the NSGA-II frontiers and compare them with single-objective benchmarks, we look at the performance of the considered feature selection methods in Table 3.4.3. To facilitate comparison, on each of the Pareto frontiers we select one solution that achieves the best performance in terms of EMP (the upper-right point). Then, we compare this solution with single-objective benchmarks in terms of EMP and the number of features.

As Table 3.4.3 suggests, the best-performing NSGA-II solution is based on fewer features compared to the solutions selected by single-objective techniques in 7 out of 10 cases and achieves the highest expected profit in 4 data sets. There is only one data set where one of the benchmarks identifies a solution which has both higher EMP and a lower complexity (SFS on *gmsc*). Performing feature selection using other base classifiers produces similar

results (see Appendix 3.6 for performance values).

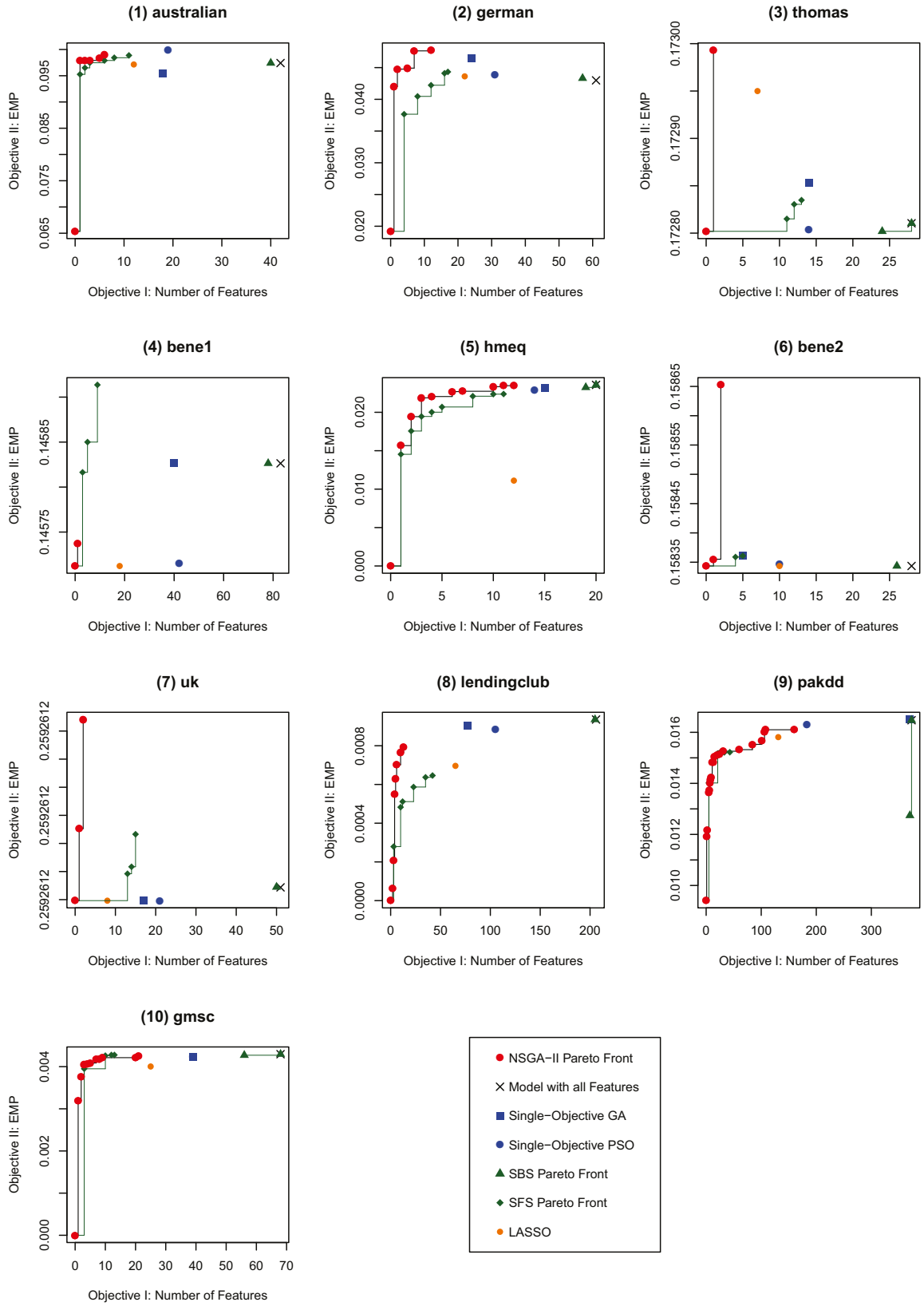

Figure 3.4.2. Performance of Feature Selection Methods: LR

Each diagram in the graph matrix depicts results on a single data set. The Pareto frontier produced by NSGA-II is depicted with red points. Green points represent non-dominated solutions from SFS and SBS; blue and yellow markers refer to other single-objective methods. LR is a base classifier.

Table 3.4.3. Performance of Feature Selection Methods: LR

Expected maximum profit							
Data	NSGA-II*	GA	PSO	SBS*	SFS*	LASSO	Full model
australian	0.0990	0.0953	0.0999	0.0974	0.0989	0.0972	0.0974
german	0.0477	0.0465	0.0439	0.0433	0.0443	0.0436	0.0430
thomas	0.1730	0.1729	0.1728	0.1728	0.1728	0.1729	0.1728
bene1	0.1457	0.1458	0.1457	0.1458	0.1458	0.1457	0.1458
hmeq	0.0235	0.0231	0.0229	0.0232	0.0224	0.0111	0.0236
bene2	0.1587	0.1584	0.1583	0.1583	0.1584	0.1583	0.1583
uk	0.2593	0.2593	0.2593	0.2593	0.2593	0.2593	0.2593
lending club	0.0008	0.0009	0.0009	0.0009	0.0006	0.0007	0.0009
pakdd	0.0161	0.0165	0.0163	0.0127	0.0152	0.0158	0.0165
gmsc	0.0042	0.0042	0.0042	0.0043	0.0043	0.0040	0.0043
Number of selected features							
australian	6	18	19	40	11	12	42
german	12	24	31	57	17	22	61
thomas	1	14	14	24	13	7	28
bene1	1	40	42	78	9	18	83
hmeq	12	15	14	19	11	12	20
bene2	2	5	10	26	5	10	28
uk	2	17	21	50	15	8	51
lending club	13	77	105	205	42	65	206
pakdd	160	370	183	370	43	131	373
gmsc	21	39	39	56	13	25	68

Results in this table use logistic regression as a base classifier. See Appendix 3.6 for results with the other models. EMP is rounded to four digits after the decimal point. Abbreviations: NSGA = non-dominated sorting based genetic algorithm, GA = genetic algorithm, PSO = particle swarm optimization, SBS = sequential backward selection, SFS = sequential forward selection, EMP = expected maximum profit, LR = logistic regression.

* Here, we consider a single solution on the Pareto frontier, which has the highest EMP and uses the maximal number of features.

To further extend the comparison, we define three metrics based on the notions discussed in example in Figure 3.4.1. Let S_1 be a share of data sets where all single-objective benchmarks are weakly dominated by points on the Pareto frontier resulting from the NSGA-II algorithm (e.g., point H). If satisfied, this condition indicates a clear advantage of the multi-objective feature selection over the benchmarks, since they can not achieve better performance in any of the objectives. Next, let S_2 indicate a share of data sets with a weaker condition: none of the benchmarks dominates the solution on the Pareto front. Here, benchmarks may either be dominated by the solutions on the frontier (e.g., point H) or achieve better EMP than solutions on the frontier, but only if they use more features (e.g., point

G). Finally, let S_3 be a share of data sets where one or more benchmarks dominates at least one solution on the frontier. This condition corresponds to point F from the aforementioned example and demonstrates an advantage of the single-objective benchmarks. We compute shares S_1, S_2 and S_3 separately for each base classifier. The results are given in Table 3.4.4.

According to Table 3.4.4, all single-objective benchmarks are dominated by the best point on the NSGA-II frontier on 40% of the data sets for LR, 50% of the cases for L1 and XGB. In other words, NSGA-II identifies a feature subset that simultaneously has a higher profitability and contains fewer features compared to the solutions identified by the conventional single-objective strategies on at least 40% of the data sets.

In most of the remaining cases, single-objective benchmarks can outperform the best multi-objective solution in terms of EMP only if they use more features. This is observed for five remaining data sets when using any of the considered base classifiers. In this case, solutions on the frontier identified by our method are still non-dominated by benchmarks and represent a trade-off between model comprehensibility and profitability in the regions where fewer features are used. Feature subsets selected by the single-objective benchmarks could serve as a possible extension of the frontier.

From the business perspective, solutions on the NSGA-II frontier may be more attractive for companies even if the scorecards are characterized by a lower profitability but based on a significantly smaller amount of data. For instance, NSGA-II achieves EMP of 0.0161 on *pakdd* data using 160 features, whereas single-objective GA identifies a subset of 370 features that obtains EMP of 0.0165. Here, relying on a multi-objective algorithm results in a 2% drop in EMP but also eliminates 57% of features. It is then the task of a risk analyst to decide whether a drop in profitability would be compensated by reducing the costs of collecting and storing the data on customer behavior.

Taking both objectives into account, solutions lying on the NSGA-II frontier are not dominated by any of the benchmarks in 90% to 100% cases depending on the base model. As we noted above, there is only one data set where one of the single-objective benchmarks identifies a feature subset that dominates some solutions on the NSGA-II based Pareto frontier. There is a single case (*gmsc* data with LR), where one of the single-objective methods dominates some solutions on the frontier. This indicates a good performance of the proposed multi-objective feature selection algorithm.

Another dimension of the algorithm comparison concerns the training times. In Table

Table 3.4.4. Aggregated Results

Base classifier	S_1	S_2	S_3
Logistic regression	40%	90%	10%
L1-regularized logistic regression	50%	100%	0%
Extreme gradient boosting	50%	100%	0%

6, we report the training times of all feature selection techniques considered in this study depending on the base classifier. The total training times are averaged across the ten credit scoring data sets. The experiments were performed on a machine with 4 cores at 3.4 GHz and 768 GB RAM.

As expected, training times of NSGA-II and single-objective GA are similar since they have the same total number of models trained within the algorithm. NSGA-II has slightly lower training times because it considers more feature subsets with lower cardinality while trying to minimize the number of features. PSO is characterized by a lower running time because of the faster convergence, whereas LASSO-based feature selection proves to be the fastest technique in our set as it only requires training a single L1 model to select features. Comparing SFS and SBS, we conclude that backward selection is preferable for L1-regularized regression due to a faster convergence but is substantially slower for LR and XGB, which require more time to train models with many features. SFS is faster than NSGA-II for all base classifiers, while SBS performs significantly slower for LR-based feature selection.

Overall, we can note that the strong advantage of LASSO in the efficiency is compensated by its poor performance in terms of profitability and number of features. The use of NSGA-II does not involve substantially larger training times compared to the techniques such as SBS or single-objective GA. Comparing NSGA-II and PSO, one can conclude that faster convergence of PSO comes at the cost of lower profitability and comprehensibility of the final scoring model. The same holds for SFS that usually fails to find more preferable feature subsets identified by NSGA-II due to the limitations of the greedy framework.

Table 3.4.5. Total Training Times

Feature selection method	Total training time (minutes)		
	LR	L1	XGB
NSGA-II	170.61	82.50	468.23
Single-objective GA	185.38	84.52	482.17
Single-objective PSO	107.98	57.94	270.63
SFS	65.42	61.69	115.96
SBS	295.06	14.02	425.62
LASSO	0.35	0.35	0.35

Abbreviations: NSGA = non-dominated sorting based genetic algorithm, GA = genetic algorithm, PSO = particle swarm optimization, SBS = sequential backward selection, SFS = sequential forward selection, LR = logistic regression, L1 = L1-regularized LR, XGB = extreme gradient boosting. The total training times are averaged across 10 data sets.

3.5 Conclusion

This paper introduces a multi-objective profit-driven framework for feature selection in credit scoring. We use the recently developed EMP measure and the number of features as two fitness functions for the wrapper-based feature selection to address both profitability and comprehensibility. Multi-objective optimization is performed with the genetic algorithm NSGA-II. We evaluate the effectiveness of our approach by running empirical experiments on ten real-world retail credit scoring data sets.

Empirical results indicate that the proposed multi-objective feature selection framework performs highly competitive compared to the conventional feature selection strategies. The developed approach identifies feature subsets that yield the same or higher expected profit using fewer features than single-objective benchmarks on at least half of the data sets. Depending on a base classifier, solutions selected by the NSGA-II are not dominated by any of the single-objective benchmarks in 90% to 100% of cases. The results imply that previous work in ignoring the two objectives of feature selection in credit scoring has missed promising solutions that can be identified using the suggested framework.

In addition to demonstrating a superior performance, the suggested multi-objective method serves as a tool to find a trade-off in two conflicting objectives: model comprehensibility and profitability. By comparing the non-dominated solutions on the frontier, risk managers can select a suitable subset of features depending on the business context.

Future research could pursue several directions. Recent literature suggested novel multi-criteria optimization methods that could replace the NSGA-II algorithm in the proposed profit-driven feature selection framework. Jimenez and colleagues proposed ENORA algorithm that demonstrates promising performance compared to NSGA-II [21]; Hancer and colleagues suggest multi-objective artificial bee colony optimization [19]; Zhang et al. apply multi-criteria particle swarm optimization to perform feature selection [46]. A systematic benchmark or corresponding solvers appears valuable to identify the most suitable multi-objective approach and clarify the degree to witch alternative approaches display different performance in a value-based feature selection context.

Another promising avenue would be to use the suggested approach to optimize a different set of relevant objectives. In particular, minimizing risk while maximizing profitability is crucial in the wider scope of financial risk management and could be considered in a credit portfolio management context. More generally, future research could consider adjusting or extending the set of objectives for the feature selection algorithm, or apply the algorithm for other tasks in a predictive modeling process.

Finally, the use of the developed feature selection approach could be extended to other business applications. One of the possible domains is customer churn. Verbraken and colleagues developed a similar EMP measure for customer churn models [40], which could serve as one of the objectives for the feature selection algorithm.

3.6 Appendix

The Appendix provides additional empirical results when using the L1-regularized logistic regression or extreme gradient boosting as a base classifier.

3.6.1 Empirical Results with L1 Model

Table 3.6.1 compares the performance of feature selection methods when using L1 as a base classifier. Figure 3.6.1 presents the graph matrix with the performance of the considered feature selection methods on all ten data sets.

Table 3.6.1. Performance of Feature Selection Methods: L1

Expected maximum profit							
Data	NSGA-II*	GA	PSO	SBS*	SFS *	LASSO	Full model
australian	0.1057	0.1014	0.1029	0.1031	0.1029	0.0852	0.0712
german	0.0371	0.0373	0.0397	0.0401	0.0357	0.0353	0.0224
thomas	0.1729	0.1728	0.1728	0.1728	0.1728	0.1728	0.1728
bene1	0.1463	0.1459	0.1459	0.1459	0.1460	0.1458	0.1457
hmeq	0.0186	0.0183	0.0183	0.0183	0.0186	0.0077	0.0077
bene2	0.1591	0.1591	0.1583	0.1583	0.1588	0.1583	0.1583
uk	0.2593	0.2593	0.2593	0.2593	0.2593	0.2593	0.2593
lending club	0.0001	0.0002	0.0001	0.0002	0.0001	0.0000	0.0000
pakdd	0.0159	0.0158	0.0153	0.0159	0.0150	0.0121	0.0120
gmsc	0.0044	0.0045	0.0045	0.0045	0.0044	0.0000	0.0000
Number of selected features							
australian	21	25	20	31	9	13	42
german	6	27	27	42	21	21	61
thomas	3	13	17	23	4	7	28
bene1	3	36	44	77	11	20	83
hmeq	3	14	13	17	4	13	20
bene2	3	3	14	21	4	11	28
uk	8	25	29	45	7	11	51
lending club	6	105	109	182	56	81	206
pakdd	229	207	191	371	40	126	373
gmsc	17	31	36	64	22	31	68

Results in this table use L1 as a base classifier. EMP is rounded to four digits after the decimal point. Abbreviations: NSGA = non-dominated sorting based genetic algorithm, GA = genetic algorithm, PSO = particle swarm optimization, SBS = sequential backward selection, SFS = sequential forward selection, EMP = expected maximum profit, L1 = L1-regularized logistic regression.
* Here, we consider a single solution on the Pareto frontier, which has the highest EMP and uses the maximal number of features.

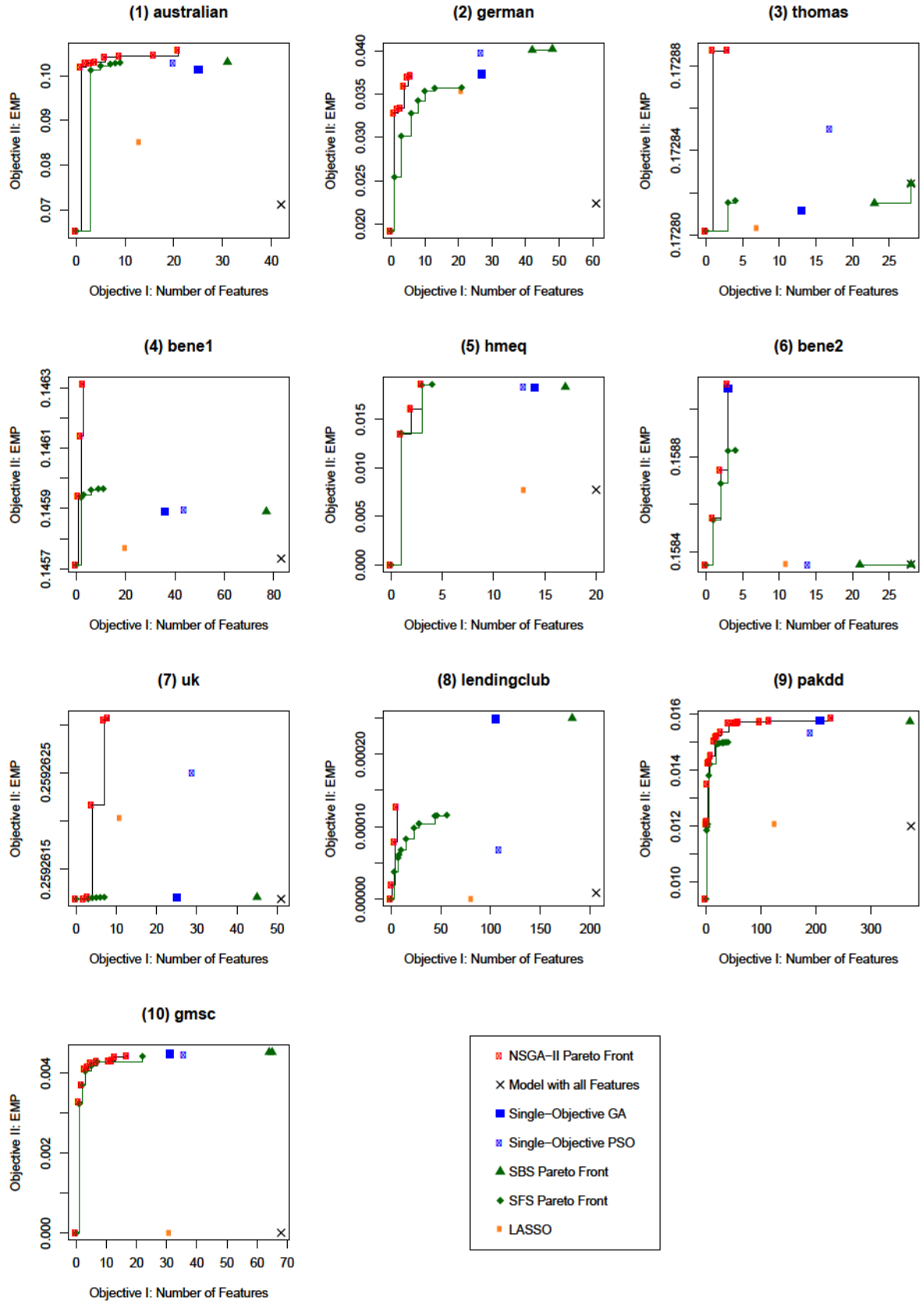

Figure 3.6.1. Performance of Feature Selection Methods: L1

Each diagram in the graph matrix depicts results on a single data set. The Pareto frontier produced by the NSGA-II algorithm is depicted with red points. Green points represent non-dominated solutions from SFS and SBS; blue and yellow markers refer to other single-objective benchmarks. L1 is used as a base classifier.

3.6.2 Empirical Results with XGB Model

Table 3.6.2 compares the performance of feature selection methods when using XGB as a base classifier. Figure 3.6.2 presents the graph matrix with the performance of the considered feature selection methods on all ten data sets.

Table 3.6.2. Performance of Feature Selection Methods: XGB

Expected maximum profit							
Data	NSGA-II*	GA	PSO	SBS*	SFS*	LASSO	Full model
australian	0.1060	0.1065	0.1046	0.1056	0.1060	0.1054	0.1055
german	0.0393	0.0391	0.0407	0.0411	0.0330	0.0327	0.0392
thomas	0.1731	0.1728	0.1728	0.1728	0.1728	0.1728	0.1728
bene1	0.1457	0.1457	0.1457	0.1457	0.1459	0.1459	0.1457
hmeq	0.0422	0.0418	0.0402	0.0415	0.0399	0.55	0.0418
bene2	0.1583	0.1583	0.1283	0.1583	0.1583	0.1583	0.1583
uk	0.2593	0.2593	0.2593	0.2593	0.2593	0.2593	0.2593
lending club	0.0008	0.0007	0.0007	0.0007	0.0008	0.0006	0.0009
pakdd	0.0168	0.0165	0.0163	0.0164	0.0157	0.0161	0.0166
gmsc	0.0046	0.0045	0.0045	0.0046	0.0044	0.0042	0.0045
Number of selected features							
australian	3	17	20	36	11	14	42
german	2	25	31	51	11	21	61
thomas	10	12	9	23	3	7	28
bene1	1	42	47	80	4	22	83
hmeq	19	15	13	19	12	12	20
bene2	1	10	28	28	3	11	28
uk	1	18	27	47	5	9	51
lending club	12	107	106	197	13	93	206
pakdd	203	180	195	366	14	124	373
gmsc	24	34	38	66	14	34	68

Results in this table use XGB as a base classifier. EMP is rounded to four digits after the decimal point. Abbreviations: NSGA = non-dominated sorting based genetic algorithm, GA = genetic algorithm, PSO = particle swarm optimization, SBS = sequential backward selection, SFS = sequential forward selection, EMP = expected maximum profit, XGB =extreme gradient boosting.
* Here, we consider a single solution on the Pareto frontier, which has the highest EMP and uses the maximal number of features.

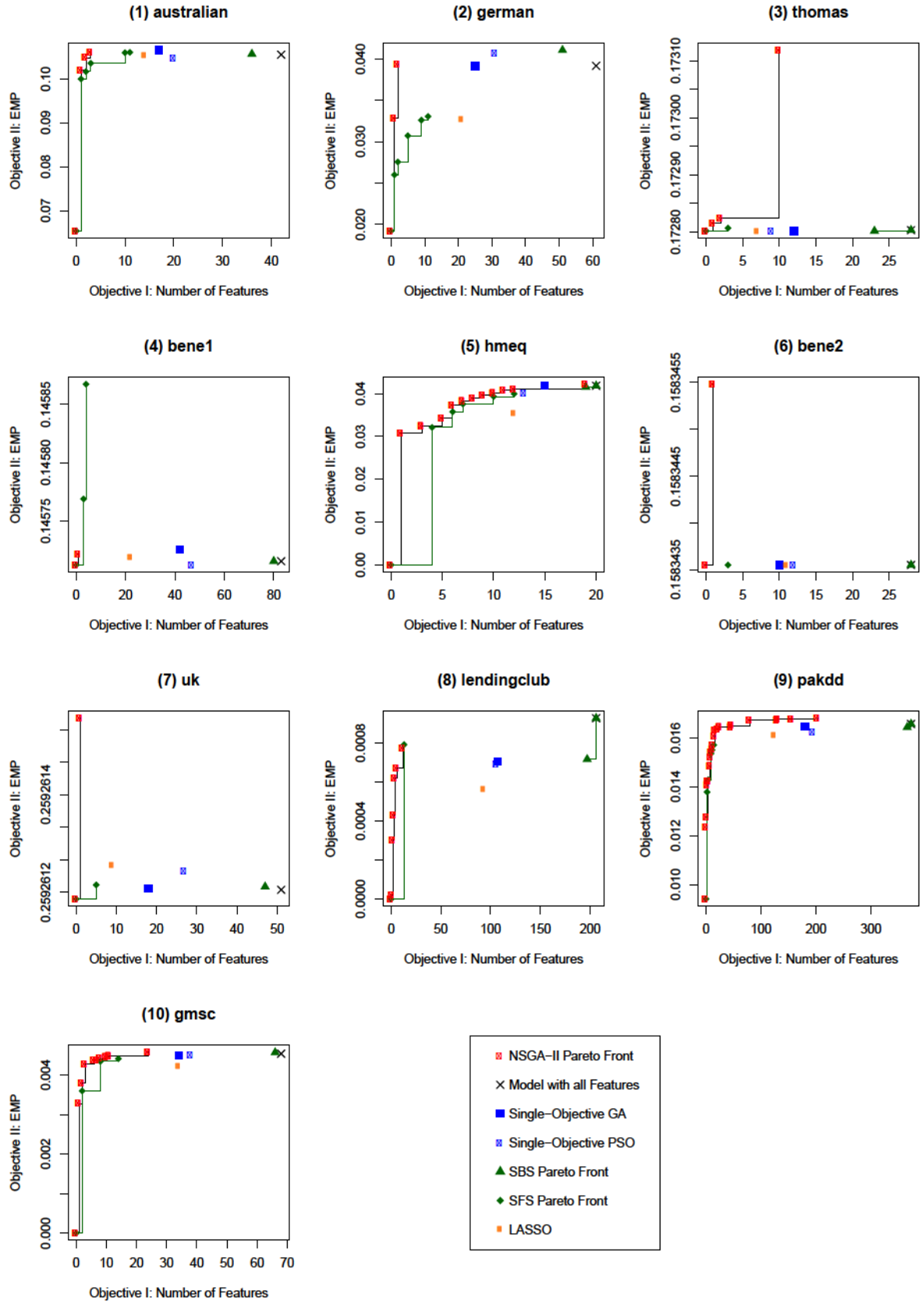

Figure 3.6.2. Performance of Feature Selection Methods: XGB

Each diagram in the graph matrix depicts results on a single data set. The Pareto frontier produced by the NSGA-II algorithm is depicted with red points. Green points represent non-dominated solutions from SFS and SBS; blue and yellow markers refer to other single-objective benchmarks. XGB is used as a base classifier.

Bibliography

[1] Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society, 54*(6), 627–635.

[2] Baesens, B., Roesch, D., Scheule, H. (2016). *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS*. John Wiley & Sons.

[3] Benítez-Peña, S., Blanquero, R., Carrizosa, E., Ramírez-Cobo, P. (2018). Cost-sensitive Feature Selection for Support Vector Machines. *Computers & Operations Research, 106*, 169–178.

[4] Bergstra, J. S., Bardenet, R., Bengio, Y., Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 2546–2554.

[5] Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A. (2013). A review of feature selection methods on synthetic data. *Knowledge and Information Systems, 34*(3), 483–519.

[6] Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A. (2015). Recent advances and emerging challenges of feature selection in the context of big data. *Knowledge-Based Systems, 86*, 33–45.

[7] Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A., Benítez, J. M., Herrera, F. (2014). A review of microarray datasets and applied feature selection methods. *Information Sciences, 282*, 111–135.

[8] Bonev, B., Escolano, F., Cazorla, M. (2008). Feature selection, mutual information, and the classification of high-dimensional patterns. *Pattern Analysis and Applications, 11*(3-4), 309–319.

[9] Bravo, C., Maldonado, S., Weber, R. (2013). Granting and managing loans for micro-entrepreneurs: New developments and practical experiences. *European Journal of Operational Research, 227*(2), 358–366.

[10] Bravo, C., Verbraken, T. (2014). *EMP: Expected maximum profit for credit scoring. R package version 1.0. URL <http://CRAN.R-project.org/package=EMP>*. Accessed 2018-09-01.

[11] Cang, S., Yu, H. (2012). Mutual information based input feature selection for classification problems. *Decision Support Systems, 54*(1), 691–698.

[12] Crook, J. N., Edelman, D. B., Thomas, L. C. (2007). Recent developments in consumer credit risk assessment. *European Journal of Operational Research, 183*(3), 1447–1465.

[13] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

[14] Emmanouilidis, C., Hunter, A., MacIntyre, J., Cox, C. (1999). Selecting features in neurofuzzy modelling by multiobjective genetic algorithms. *Proc. the 9th International Conference on Artificial Neural Networks*, 4387–4392.

[15] Finlay, S. (2010). Credit scoring for profitability objectives. *European Journal of Operational Research, 202*(2), 528–537.

[16] Guyon, I., Elisseeff, A. (2003). An introduction to feature and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

[17] Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. A. (2006). *Feature Extraction: Foundations and Applications*. Springer.

[18] Hamdani, T. M., Won, J. M., Alimi, A. M., Karray, F. (2007). Multi-objective feature selection with NSGA II. *Proc. International Conference on Adaptive and Natural Computing Algorithms*, 240–247.

[19] Hancer, E., Xue, B., Zhang, M., Karaboga, D., Akay, B. (2018). Pareto front feature selection based on artificial bee colony optimization. *Information Sciences, 422*, 462–479.

[20] Hand, D. J. (2005). Good practice in retail credit scorecard assessment. *Journal of the Operational Research Society, 56*(9), 1109–1117.

[21] Jimenez, F., Gómez-Skarmeta, A. F., Sánchez, G., Deb, K. (2002). An evolutionary algorithm for constrained multi-objective optimization. *Proc. the 2002 Congress on Evolutionary Computation*, 1133–1138.

[22] Jung, K. M., Thomas, L. C., So, M. C. (2015). When to rebuild or when to adjust scorecards. *Journal of the Operational Research Society, 66*(10), 1656–1668.

[23] Lessmann, S., Baesens, B., Seow, H. V., Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

[24] Maldonado, S., Flores, Á., Verbraken, T., Baesens, B., Weber, R. (2015). Profit-based feature selection using support vector machines – General framework and an application for customer retention. *Applied Soft Computing*, 35, 740–748.

[25] Maldonado, S., Bravo, C., Lopez, J., Pérez, J. (2017). Integrated framework for profit-based feature selection and SVM classification in credit scoring. *Decision Support Systems, 104*, 113–121.

[26] Maldonado, S., Pérez, J., Bravo, C. (2017). Cost-based feature selection for support vector machines: An application in credit scoring. *European Journal of Operational Research, 261*(2), 656–665.

[27] Mays, E., Lynas, N. (2004). *Credit scoring for risk managers: The handbook for lenders.* Ohio: Thomson/South-Western.

[28] Min, F., Hu, Q., Zhu, W. (2014). Feature selection with test cost constraint. *International Journal of Approximate Reasoning, 55*(1), 167–179.

[29] Mukerjee, A., Biswas, R., Deb, K., Mathur, A. P. (2002). Multi–objective evolutionary algorithms for the risk–return trade–off in bank loan management. *International Transactions in operational research, 9*(5), 583–597.

[30] Oliveira, L. S., Sabourin, R., Bortolozzi, F., Suen, C. Y. (2002). Feature selection using multi-objective genetic algorithms for handwritten digit recognition. *Proc. the 16th International Conference on Pattern Recognition*, 240–247.

[31] Poursabzi-Sangdeh, F., Goldstein, D. G., Hofman, J. M., Vaughan, J. W., Wallach, H. (2017). Manipulating and measuring model interpretability. *Proc. NIPS 2017 Transparent and Interpretable Machine Learning in Safety Critical Environments Workshop.*.

[32] Saeedi, R., Schimert, B., Ghasemzadeh, H. (2014). Cost-sensitive feature selection for on-body sensor localization. *Proc. of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 833–842.

[33] Serrano-Cinca, C., Gutiérrez-Nieto, B. (2016). The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending. *Decision Support Systems, 89*, 113–122.

[34] Somers, M., Whittaker, J. (2007). Quantile regression for modelling distributions of profit and loss. *European Journal of Operational Research, 183*(3), 1477–1487.

[35] Soto, A. J., Cecchini, R. L., Vazquez, G. E., Ponzoni, I. (2009). Multi-objective feature selection in QSAR using a machine learning approach. *QSAR & Combinatorial Science, 28*(11-12), 1509–1523.

[36] Srinivas, N., Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation, 2*(3), 221–248.

[37] Thomas, L. C., Edelman, D. B., Crook, J. N. (2002) *Credit Scoring and its Applications.* Philadelphia: SIAM.

[38] Tsaih, R., Liu, Y. J., Liu, W., Lien, Y. L. (2004). Credit scoring system for small business loans. *Decision Support Systems, 38*(1), 91–99.

[39] Verbraken, T., Bravo, C., Weber, R., Baesens, B. (2014). Development and application of consumer credit scoring models using profit-based classification measures. *European Journal of Operational Research, 238*(2), 505–513.

[40] Verbraken, T., Verbeke, W., Baesens, B. (2013). A novel profit maximizing metric for measuring classification performance of customer churn prediction models. *IEEE Transactions on Knowledge and Data Engineering, 25*(5), 961–973.

[41] Vidaurre, D., Bielza, C., Larrañaga, P. (2013). A survey of L1 regression. *International Statistical Review, 81*(3), 361–387.

[42] Vieira, S. M., Mendonça, L. F., Farinha, G. J., Sousa, J. M. (2013). Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. *Applied Soft Computing, 13*(8), 3494–3504.

[43] Xue, B., Zhang, M., Browne, W. N. (2013). Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics, 43*(6), 1656–1671.

[44] Xue, B., Zhang, M., Browne, W. N., Yao, X. (2016). A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation, 20*(4), 606–626.

[45] Yang, J., Honavar, V. (1998). Feature subset selection using a genetic algorithm. *Proc. Feature Extraction, Construction and Selection*, 117–136.

[46] Zhang, Y., Gong, D. W., Cheng, J. (2017). Multi-objective particle swarm optimization approach for cost-based feature selection in classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 14*(1), 64–75.

Chapter 4

Multi-Objective Particle Swarm Optimization for Feature Selection in Credit Scoring

Publication
Kozodoi, N., & Lessmann, S. (2020). Multi-objective Particle Swarm Optimization for Feature Selection in Credit Scoring. In Workshop on Mining Data for Financial Applications (pp. 68-76). Springer, Cham.

Abstract
Credit scoring refers to the use of statistical models to support loan approval decisions. An ever-increasing availability of data on potential borrowers emphasizes the importance of feature selection for scoring models. Traditionally, feature selection has been viewed as a single-objective task. Recent research demonstrates the effectiveness of multi-objective approaches. We propose a novel multi-objective feature selection framework for credit scoring that extends previous work by taking into account data acquisition costs and employing a state-of-the-art particle swarm optimization algorithm. Our framework optimizes three fitness functions: the number of features, data acquisition costs and the AUC. Experiments on nine credit scoring data sets demonstrate a highly competitive performance of the proposed framework.

4.1 Introduction

Financial institutions use credit scoring models to support loan approval decisions [10]. Due to the unprecedented availability of data on potential credit applicants and growing access of financial institutions to new data sources, the data used to train scoring models tend to be high-dimensional [6].

Feature selection aims at removing irrelevant features to improve the model performance, which is traditionally considered as a single-objective task [7]. In credit scoring, feature selection can be treated as a multi-objective problem with multiple goals. In addition to optimizing model performance, companies strive to reduce the number of features as public discourse and regulatory requirements are calling for comprehensible credit scoring models [9]. Furthermore, financial institutions often purchase data from external providers such as credit bureaus and banks in groups of features. This creates a need for a separate account for the data acquisition costs [13]. The conflicting nature of these objectives motivates us to consider feature selection as a multi-objective optimization problem.

Recent research has demonstrated the effectiveness of multi-objective feature selection in credit scoring using genetic algorithms (GA) [9]. However, previous studies have not considered a simultaneous optimization of the number and the cost of features. Moreover, techniques based on particle swarm optimization (PSO) have been recently shown to outperform GAs in other domains [15].

This paper proposes a multi-objective feature selection framework for credit scoring and makes two contributions. First, our framework addresses three distinct objectives: (i) the number of features, (ii) the cost of features and (iii) the model performance. Optimizing both the number and the cost of features is crucial as purchasing data from multiple sources introduces grouping that decorrelates the two objectives. Increasing the number of data providers incurs higher costs, whereas adding individual features does not affect costs if the new features are purchased in a group with the already included features. Second, we adjust a state-of-the-art PSO algorithm denoted as AgMOPSO to the feature selection context to improve the performance of the search algorithm.

4.2 Related Work

Standard techniques consider feature selection as a single-objective task [7]. Recent studies have shown the importance of accounting for multiple objectives such as model performance and cardinality of the feature set [3, 9]. A simple approach to account for multiple objectives is to aggregate them into one objective or introduce optimization constraints to a single-objective task. This requires additional information such as weights of objectives or budget conditions. In contrast, a truly multi-objective approach produces a set of non-dominated solutions – a Pareto frontier – where improving one objective is impossible without worsening at least one of the others. Provided with such a frontier, a decision-maker can examine the trade-off between the objectives depending on the context.

Previous research on multi-objective feature selection has employed evolutionary algorithms such as GA and PSO. Most studies use a non-dominated sorting genetic algorithm NSGA-II, which is a well-known algorithm for multi-objective optimization [8]. NSGA-III was proposed as a successor of NSGA-II to handle challenges of many-objective optimization [3]. The usage of GAs has also been recently challenged by the proposal of PSO techniques that demonstrate a superior performance [15, 16]. Research outside of the feature selection domain has also suggested other optimization techniques such as Gaussian processes [4].

Credit scoring is characterized by multiple conflicting objectives, such as model performance and comprehensibility [9, 13]. Nevertheless, research on multi-objective feature selection in credit scoring has been scarce. Maldonado et al. use support vector machines (SVM) to optimize performance while minimizing feature costs as a regularization penalty [12, 13]. Both methods are embedded in SVMs and output a single solution instead of a Pareto frontier. Kozodoi et al. argue for a model-agnostic multi-objective approach [9].

Their framework uses NSGA-II and is limited to two objectives, assuming the number of features to be indicative of both model comprehensibility and data acquisition costs.

We extend the previous work on feature selection in credit scoring by adapting a state-of-the-art PSO algorithm to perform the feature search and considering feature costs as a distinct objective. A common practice of purchasing data in groups of features reduces the correlation between the objectives and provides an opportunity for multi-criteria optimization. The number of features serves as a proxy for model comprehensibility and interpretability, whereas feature costs indicate the data acquisition costs faced by a financial institution.

4.3 Proposed Framework

We propose a multi-objective feature selection framework based on the external archive-guided MOPSO algorithm (AgMOPSO), which demonstrates superior performance compared to GAs in various optimization tasks [17]. PSO is a meta-heuristic method that solves an optimization problem by evolving a population of candidate solutions (i.e., particles) that iteratively navigate through the search space. AgMOPSO encompasses two stages: initialization and feature search.

The algorithm initializes with a random swarm of n particles. Each particle is represented by a real-valued vector of length k, where k is the number of features. The particle values are restricted to $[0, 1]$ and indicate the probability of a feature being selected. The initialization is followed by an iterative process of guiding the swarm towards new solutions using the immune-based and PSO-based search and evolving an archive that stores the non-dominated solutions.

The immune-based search produces new particles by applying genetic operators such as cloning, crossover and mutation to the existing particles [17]. The PSO-based search creates new particles by updating the values of each particle using a decomposition-based approach. The particle position in the k-dimensional feature space is adjusted by moving them towards the swarm leaders, i.e., particles that perform best in each of the three objectives [1].

After each search round, we evaluate solutions. We train a model that includes features corresponding to the rounded particle values and evaluate three fitness functions: (i) the number of features, (ii) feature acquisition costs and (iii) the model performance in terms of the area under the ROC curve (AUC). Based on the evaluated fitness, we store non-dominated solutions representing different feature subsets in the archive. Until the maximum size of the archive is reached, all non-dominated solutions are added to the archive. Once the archive is full, we calculate the crowding degree of new particles that indicates the density of surrounding solutions. If the new solution displays a better crowding degree than at least one archive solution, it replaces the most crowded solution in the archive.

4.4 Experimental Setup

4.4.1 Data

Table 4.4.1 displays the data sets used in the experiments. All data sets come from a retail credit scoring context. Data sets *australian* and *german* are part of the UCI Machine Learning Repository[1]. Data sets *thomas* and *hmeq* are provided by [14] and [2]; *paipaidai* is collected from [11]. Data sets *pakdd*, *lendingclub* and *gmsc* are provided for data mining competitions on PAKDD[2] and Kaggle[3] platforms.

Each data set contains a binary target variable indicating whether a customer has repaid a loan and a set of features describing characteristics of the applicant, the loan and, in some cases, the applicant's previous loans. As illustrated in Table 4.4.1, the sample size and the number of features vary across the data sets, which allows us to test our feature selection framework in different conditions.

4.4.2 Setup

We consider a multi-criteria feature selection problem with three objectives: (i) the number of selected features, (ii) feature acquisition costs, (iii) the AUC. Each of the nine data sets is randomly partitioned into training (70%) and holdout sets (30%). We perform feature selection with AgMOPSO within four-fold cross-validation on the training set. The performance of the selected feature subsets is evaluated on the holdout set. To ensure robustness, the performance is aggregated over 20 modeling trials with different random data partitioning.

Since the data on feature acquisition costs are not available in all considered data sets, we simulate costs similar to [15]. The cost of each feature is drawn from a Uniform distribution in the interval $[0, 1]$. To simulate feature groups, we introduce a cost-based grouping for categorical features. Each categorical feature is transformed with dummy encoding. Next, we assign acquisition costs to dummy features: if one dummy variable stemming from a specific categorical feature is selected, other dummies related to the same feature can be included at no additional cost.

NSGA-II and NSGA-III with the same three objectives as AgMOPSO serve as benchmarks. The meta-parameters of the algorithms are tuned using grid search on a subset of training data. To ensure a fair comparison, the population size and the number of generations for NSGA-II and NSGA-III are set to the same values as for the AgMOPSO. We also use a full model with all features as a benchmark. L2-regularized logistic regression serves as a base classifier.

[1]Source: `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`, `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`

[2]Source: `https://www.kdnuggets.com/2010/03/f-pakdd-2010-data-mining-competition.html`

[3]Source: `https://www.lendingclub.com`, `https://kaggle.com/c/givemesomecredit`

Table 4.4.1. Credit Scoring Data Sets

Data set	Sample size	No. features	Default rate
australian	690	42	.44
german	1,000	61	.30
thomas	1,125	28	.26
hmeq	5,960	20	.20
cashbus	15,000	1,308	.10
lendingclub	43,344	206	.07
pakdd2010	50,000	373	.26
paipaidai	60,000	1,934	.07
gmsc	150,000	68	.07

We use five evaluation metrics common in multi-objective optimization to reflect different characteristics of the evolved solution frontiers: (i) hypervolume (HV) is an overall performance metric that indicates the objective space covered by the solutions; (ii) overall non-dominated vector generation (ONVG) is the number of distinct non-dominated solutions; (iii) two-set coverage (TSC) reflects the convergence of the frontier; (iv) spacing (SPC) considers how evenly the solutions are distributed; (v) maximum spread (SPR) accounts for the solution spread. The calculation of the SPC and the SPR requires knowledge of the true Pareto frontier, which is difficult to estimate due to the high dimensionality of the feature space. We combine the non-dominated solutions across the three algorithms and 20 trials to form an adequate approximation of the true frontier.

We also compare the full model with the single best-performing solutions of the evolutionary algorithms that achieve the highest AUC compared to the other solutions on the evolved Pareto frontiers. The single solutions are compared in the three considered objectives.

4.5 Results

Table 4.5.1 provides the experimental results. For each data set, we rank algorithms in the five multi-objective optimization metrics and report the mean ranks across the 20 trials. We also report the mean AUC, data acquisition costs and the number of features of the single solutions with the highest AUC.

Overall, AgMOPSO outperforms the GA-based benchmarks in three performance metrics, achieving the lowest average rank in the ONVG, the TSC and the HV. According to the Nemenyi test [5], differences in algorithm ranks are significant at a 5% level. The superior performance of AgMOPSO is mainly attributed to a higher cardinality and a better convergence of the evolved frontier compared to NSGA-II and NSGA-III. This is indicated by the best performance of AgMOPSO in the ONVG and the TSC on seven out of nine data sets.

In terms of the diversity of the evolved frontier, AgMOPSO does not outperform the

Table 4.5.1. Comparing Performance of Feature Selection Methods

Data set	Algorithm	ONVG	TSC	SPC	SPR	HV	AUC	FC	NF
australian	AgMOPSO	**1.70**	**1.80**	2.25	**1.45**	**1.80**	**.9184**	1.96	9.35
	NSGA-II	2.05	1.93	2.00	2.10	2.15	.9170	1.95	8.15
	NSGA-III	2.10	2.05	**1.75**	2.15	2.05	.9162	**1.49**	**6.90**
	Full model			–			.6751	6.81	42
german	AgMOPSO	**1.20**	1.87	2.55	**1.35**	1.95	.7823	3.78	15.70
	NSGA-II	1.75	**1.82**	2.15	1.50	**1.40**	**.7824**	3.24	13.65
	NSGA-III	2.90	2.07	**1.30**	2.70	2.65	.7723	**1.95**	**9.05**
	Full model			–			.5806	9.97	61
thomas	AgMOPSO	1.65	**1.75**	2.05	**1.75**	1.95	.6368	1.47	4.50
	NSGA-II	1.70	**1.75**	2.05	1.80	**1.80**	.6363	1.36	3.70
	NSGA-III	**1.55**	**1.75**	**1.50**	1.95	1.85	.6364	**1.00**	**3.30**
	Full model			–			**.6375**	6.81	28
hmeq	AgMOPSO	**1.20**	**1.77**	2.45	**1.45**	**1.45**	**.7660**	3.23	9.45
	NSGA-II	1.95	1.88	2.15	1.75	2.25	.7651	3.14	9.10
	NSGA-III	2.60	2.02	**1.40**	2.60	2.30	.7604	**2.32**	**5.85**
	Full model			–			.5730	5.76	20
cashbus	AgMOPSO	**1.85**	**1.72**	1.70	2.25	2.30	.6450	**3.52**	**10.55**
	NSGA-II	1.95	2.00	2.90	**1.05**	2.30	.6426	14.64	44.95
	NSGA-III	2.15	2.08	**1.40**	2.70	**1.40**	**.6606**	8.22	27.50
	Full model			–			.5233	321.10	1308
lendingclub	AgMOPSO	**1.60**	1.95	1.90	1.70	2.05	.6169	1.96	20.25
	NSGA-II	1.80	2.12	2.30	**1.50**	2.60	.6149	2.08	18.60
	NSGA-III	2.40	**1.75**	**1.80**	2.25	**1.35**	**.6176**	**1.81**	**16.75**
	Full model			–			.5725	6.24	206
pakdd2010	AgMOPSO	**1.45**	**1.75**	2.15	2.05	**1.00**	**.6254**	8.10	115.55
	NSGA-II	1.95	2.28	2.75	**1.00**	2.00	.6252	9.47	252.05
	NSGA-III	2.55	1.95	**1.10**	2.95	3.00	.6220	**7.65**	**69.65**
	Full model			–			.5748	14.58	373
paipaidai	AgMOPSO	2.85	**1.80**	1.70	2.50	1.85	.6639	5.84	**36.15**
	NSGA-II	1.60	2.03	2.85	**1.00**	**1.40**	.6727	8.88	245.40
	NSGA-III	**1.50**	1.92	**1.45**	2.40	2.75	**.6802**	**5.82**	76.35
	Full model			–			.4956	57.89	1934
gmsc	AgMOPSO	**1.00**	**1.72**	2.45	**1.50**	**1.35**	**.8603**	4.72	25.50
	NSGA-II	1.95	1.90	2.20	1.70	1.90	.8602	4.71	23.15
	NSGA-III	3.00	2.30	**1.35**	2.65	2.75	.8556	**3.52**	**15.75**
	Full model			–			.6437	4.85	68

Abbreviations: ONVG = overall non-dominated vector generation, TSC = two-set coverage, SPC = spacing, SPR = maximum spread, HV = hypervolume, AUC = area under the ROC curve; FC = feature acquisition costs, NF = number of selected features.

benchmarks. In the feature selection application, diversity of solutions is of secondary importance as we are mainly interested in models from a subspace that covers the best-performing models. A more relevant indicator and an apparent strength of AgMOPSO compared to the competing algorithms is its better convergence to the true frontier, which indicates that it misses a smaller number of non-dominated solutions in the relevant search space.

The mean correlation between the number of features and data acquisition costs across the non-dominated solutions is .7455. This emphasizes the importance of a separate optimization of the two feature-based objectives.

Due to a large number of noisy features, the performance of the full model is suboptimal. Comparing the full model to the solutions of multi-objective algorithms with the highest AUC, we see that all three evolutionary algorithms identify a feature subset that achieves a higher AUC, lower costs and a smaller number of features on eight data sets. On average, AgMOPSO reduces data acquisition costs and the number of selected features by 58.77% and 76.20%, respectively. AgMOPSO also outperforms GA-based benchmarks in terms of the AUC on five data sets. At the same time, solutions of NSGA-III entail lower costs and a smaller number of features compared to AgMOPSO, which comes at the cost of a lower average AUC. These results indicate that AgMOPSO more effectively explores regions of the search space associated with a higher AUC.

Figure 4.5.1 illustrates the Pareto frontiers outputted by the three feature selection algorithms on one of the trials on *gmsc*. The figure demonstrates the ability of AgMOPSO to identify a larger number of non-dominated solutions in the search subspace with a high AUC compared to the GA-based benchmarks.

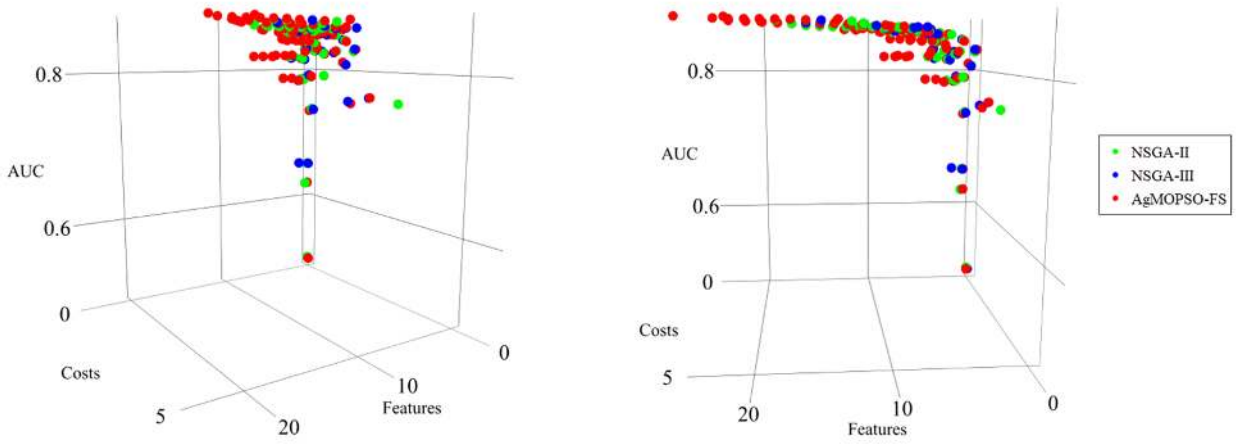

Figure 4.5.1. Pareto Frontiers for GMSC

The figure depicts the set of non-dominated solutions in the three-dimensional objective space from two angles for the *gmsc* data set.

4.6 Discussion

This paper proposes a multi-objective framework for feature selection in credit scoring using the AgMOPSO algorithm. We perform feature selection using three fitness functions reflecting relevant credit scoring objectives: the number of features, data acquisition costs, and model performance. The performance of our framework is assessed on nine real-world credit scoring data sets.

The results suggest that AgMOPSO is a highly competitive multi-objective feature selection framework, as indicated by standard quality criteria for multi-objective optimization. Compared to other evolutionary algorithms, AgMOPSO more effectively explores regions of the search space associated with a high model performance, while also substantially reducing the number of features and the data acquisition costs compared to a model using all features.

In future studies, we plan to conduct a more in-depth analysis of AgMOPOSO. It would be interesting to compare results with the solutions evolved by two-objective feature selection algorithms that ignore data acquisition costs. Analysis of the impact of correlation between the objectives on the algorithm performance could also shed more light on conditions in which the number and the cost of features should be considered as separate objectives. In addition, computing the running times and the number of generations before convergence would contribute a new angle to compare feature selection algorithms.

AgMOPOSO has a wide set of meta-parameters, which poses an opportunity for a systematic sensitivity analysis that could provide deeper insights into the appropriate values. For instance, the diversity of the evolved solutions could be improved by adjusting the crossover and mutation operations within the search. Using different base learners could help evaluate gains given a model with a built-in feature selection mechanism (e.g., L1-regularized logistic regression).

Our multi-objective feature selection framework could be extended to other application areas, such as fraud detection or churn prediction. For both of these applications, customer data are typically gathered from different sources and therefore provides opportunities for group-based cost optimization.

Bibliography

[1] Al Moubayed, N., Petrovski, A., McCall, J. (2014). D2MOPSO: MOPSO based on decomposition and dominance with archiving using crowding distance in objective and solution spaces. Evolutionary Computation 22(1), 47–77.

[2] Baesens, B., Roesch, D., Scheule, H. (2016). *Credit risk analytics: Measurement techniques, applications, and examples in SAS.* John Wiley & Sons.

[3] Bidgoli, A.A., Ebrahimpour-Komleh, H., Rahnamayan, S. (2019). A many-objective

feature selection algorithm for multi-label classification based on computational complexity of features. *Proc. 2019 14th International Conference on Computer Science & Education (ICCSE),* 85–91. IEEE.

[4] Bradford, E., Schweidtmann, A.M., Lapkin, A. (2018). Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm. *Journal of Global Optimization, 71*(2), 407–438.

[5] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research, 7*, 1–30.

[6] Gambacorta, L., Huang, Y., Qiu, H., Wang, J. (2019). How do machine learning and non-traditional data affect credit scoring? New evidence from a Chinese fintech firm, Working paper, Bank for International Settlements

[7] Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A. (2008). *Feature extraction: Foundations and applications.* Springer.

[8] Hamdani, T.M., Won, J.M., Alimi, A.M., Karray, F. (2007). Multi-objective feature selection with NSGA II. *Proc. International Conference on Adaptive and Natural Computing Algorithms, 240–247.* Springer.

[9] Kozodoi, N., Lessmann, S., Papakonstantinou, K., Gatsoulis, Y., Baesens, B. (2019). A multi-objective approach for profit-driven feature selection in credit scoring. *Decision Support Systems, 120*, 106–117.

[10] Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

[11] Li, H., Zhang, Y., Zhang, N. (2017). Evaluating the well-qualified borrowers from PaiPaiDai. *Procedia Computer Science, 122*, 775–779.

[12] Maldonado, S., Flores, Á., Verbraken, T., Baesens, B., Weber, R. (2015). Profit-based feature selection using support vector machines – general framework and an application for customer retention. *Applied Soft Computing, 35*, 740–748.

[13] Maldonado, S., Pérez, J., Bravo, C. (2017). Cost-based feature selection for support vector machines: An application in credit scoring. *European Journal of Operational Research, 261*(2), 656–665.

[14] Thomas, L., Crook, J., Edelman, D. (2017). *Credit scoring and its applications.* SIAM.

[15] Zhang, Y., Gong, D.wW, Cheng, J. (2015). Multi-objective particle swarm optimization approach for cost-based feature selection in classification. *Proc. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 14*(1), 64–75.

[16] Zhang, Y., Gong, D.W., Sun, X.Y., Guo, Y.N. (2017). A PSO-based multi-objective multi-label feature selection method in classification. *Scientific Reports, 7*(1), 1–12.

[17] Zhu, Q., Lin, Q., Chen, W., Wong, K.C., Coello, C.A.C., Li, J., Chen, J., Zhang, J. (2017). An external archive-guided multiobjective particle swarm optimization algorithm. *IEEE Transactions on Cybernetics, 47*(9), 2794–2808.

Chapter 5

Shallow Self-Learning for Reject Inference in Credit Scoring

Publication

Kozodoi, N., Katsas, P., Lessmann, S., Moreira-Matias, L., & Papakonstantinou, K. (2019). Shallow self-learning for reject inference in credit scoring. In European Conference on Machine Learning and PKDD 2019 Proceedings (pp. 516-532). Springer, Cham.

Abstract

Credit scoring models support loan approval decisions in the financial services industry. Lenders train these models on data from previously granted credit applications, where the borrowers' repayment behavior has been observed. This approach creates sample bias. The scoring model is trained on accepted cases only. Applying the model to screen applications from the population of all borrowers degrades its performance. Reject inference comprises techniques to overcome sampling bias through assigning labels to rejected cases. This paper makes two contributions. First, we propose a self-learning framework for reject inference. The framework is geared toward real-world credit scoring requirements through considering distinct training regimes for labeling and model training. Second, we introduce a new measure to assess the effectiveness of reject inference strategies. Our measure leverages domain knowledge to avoid artificial labeling of rejected cases during evaluation. We demonstrate this approach to offer a robust and operational assessment of reject inference. Experiments on a real-world credit scoring data set confirm the superiority of the suggested self-learning framework over previous reject inference strategies. We also find strong evidence in favor of the proposed evaluation measure assessing reject inference strategies more reliably, raising the performance of the eventual scoring model.

5.1 Introduction

Financial institutions use supervised learning to guide lending decisions. The resulting credit scoring models, also called scorecards, predict the probability of default (PD) – an applicant's willingness and ability to repay debt [31]. Loan approval decisions are made based on whether the scorecard predicts an applicant to be a repaying borrower (*good* risk) or a likely defaulter (*bad* risk).

Scoring models are trained on data of accepted applicants. Their repayment behavior has been observed, which provides the labels for supervised learning. Inevitably, the sample

of accepted clients (accepts) differs from the overall population of credit applicants. Accepts have passed the screening of the lender's scorecard, whereas the population also includes clients who have been denied credit by that scorecard (rejects) as well as customers who have not applied for credit. As a result, scoring models suffer from sample bias. Training a classifier only on data from accepts deteriorates the accuracy of PD predictions when the scorecard is out into production for screening incoming credit applications [28].

Reject inference refers to techniques that remedy sampling bias through inferring labels for rejects. Previous research has suggested several approaches including naive strategies (e.g., label all rejects as *bad*) and model-based techniques [28]. However, empirical evidence concerning the value of reject inference and the efficacy of labeling strategies is scarce. Several studies use incomplete data, which only contain accepted cases [e.g., 5, 11], do not have a labeled unbiased sample with both accepts and rejects [e.g., 7] or use synthetic data [e.g., 16]. In addition, the data sets employed in prior studies are usually low-dimensional [e.g., 21], which is not representative of the real-world credit scoring data used today [33]. Previous work is also geared toward linear models and support vector machines (SVM) [1, 19, 21]. Yet, there is much evidence that other algorithms (e.g., tree-based ensembles) outperform these methods in credit scoring [18, 34].

The contribution of this paper is two-fold. First, we introduce a novel self-learning framework for reject inference in credit scoring. Our framework includes two different probabilistic classifiers for the training and labeling stages. The training stage benefits from using a strong learner such as gradient boosting. However, we suggest using a shallow (i.e. weaker) learner for the labeling stage and show that it achieves higher calibration with respect to the true PD [23]. As a result, we maximize the precision of our model on the extreme quantiles of its output and minimize the noise introduced on newly labeled rejects.

Second, we introduce a novel measure (denoted as *kickout*) to assess reject inference methods in a reliable and operational manner. Aiming at labeling rejects to raise the scorecard performance, the acid test of a reject inference strategy involves comparing a scorecard without correction for sample bias to a model that has undergone reject-inference based correction on data from an unbiased sample of clients including both accepts and rejects with actual labels for both groups of clients. Such a sample would represent the operating conditions of a scorecard and thus uncover the true merit of reject inference [11]. However, obtaining such a sample is very costly as it requires a financial institution to lend money to a random sample of applicants including high-risk cases that would normally be denied credit. Drawing on domain knowledge, the proposed *kickout* measure avoids dependence on the actual labels of rejects and, as we establish through empirical experimentation, assesses the merit of a reject inference method more accurately than previous evaluation approaches. The data set used in this paper includes an unbiased sample containing both accepts and rejects, giving us a unique opportunity to evaluate a scorecard in its operating conditions.

The paper is organized as follows. Section 2 reviews related literature on reject inference.

Section 3 revisits the reject inference problem, presents our self-learning framework and introduces the kickout measure. Section 4 describes our experimental setup and reports empirical results. Section 5 concludes the paper.

5.2 Literature Review

The credit scoring literature has suggested different model-based and model-free approaches to infer labels of rejected cases. Some model-free techniques rely on external information such as expert knowledge to manually label rejects [22]. Another approach is to label all rejected cases as *bad* risks [28], assuming that the default ratio among the rejects is sufficiently high. One other strategy is to obtain labels by relying on external performance indicators such as credit bureau scores or an applicant's outcome on a previous loan [2, 28].

Model-based reject inference techniques rely on a scoring model to infer labels for rejects. Table 5.2.1 depicts corresponding techniques, where we sketch the labeling strategy used in a study together with the base classifier that was used for scorecard development. Table 5.2.1 reveals that most reject inference techniques have been tested with linear models such as logistic and probit regression.

The literature distinguishes several approaches toward model-based reject inference such as augmentation, extrapolation, bivariate models and others [19]. Extrapolation refers to a set of techniques that use the initial scoring model trained on the accepts to label the rejected cases. For instance, hard cutoff augmentation labels rejects by comparing their model-estimated PDs to a predefined threshold [28]. Parceling introduces a random component, separating the rejected cases into segments based on the range (e.g., percentile) of PDs. Instead of assigning labels based on the individual scores of rejects, they are labeled randomly within the identified segments based on the expected default rate for each score range. A drawback of such techniques is their reliance on the performance of the initial scoring model when applied to rejects.

Augmentation (or re-weighting) is based on the fact that applicants with a certain distribution of features appear in the training data disproportionately due to a non-random sample selection [11]. Re-weighting refers to the techniques that train an additional model that separates accepts and rejects and predicts the probability of acceptance. These probabilities are then used to compute sampling weights for a scoring model.

Some scholars suggest using a two-stage bivariate probit model or two-stage logistic regression to perform reject inference [6]. A bivariate model incorporates the Heckman's correction to account for a sample bias within the model, estimating both acceptance and default probability. These models assume linear effects within the logistic or probit regression framework.

Empirical studies have shown little evidence that reject inference techniques described above improve scorecard's performance [3, 9, 11, 32]. Recently suggested alternatives rely

on semi-supervised learning. For example, Maldonado et al. have shown that self-learning with SVM outperforms well-known reject inference techniques such as ignoring rejects or labeling all rejects as *bad* risks [21]. Their work is continued by Li et al. [19], who propose a semi-supervised SVM that uses a non-linear kernel to train a scoring model.

We follow recent studies and cast the reject inference problem in a semi-supervised learning framework. Our approach to solve the problem is a variation of self-learning adapted to a credit scoring context by extending the work of Maldonado et al. [21].

Table 5.2.1. Model-Based Reject Inference Methods

Reference	Reject inference technique	Base model
Reichert et al. [24]	LDA-based	LDA
Joanes [16]	Reclassification	LR
Hand et al. [15]	Ratio prediction	–
Hand et al. [15]	Rebalancing model	–
Feelders [12]	Mixture modeling	LR, QDA
Banasik et al. [6]	Augmentation	LR, Probit
Smith et al. [29]	Bayesian network	Bayesian
Crook et al. [11]	Reweigthing	LR
Verstraeten et al. [32]	Augmentation	LR
Banasik et al. [3]	Augmentation	LR
Fogarty [13]	Multiple imputation	LR
Montrichard [22]	Fuzzy augmentation	LR
Banasik et al. [4]	Augmentation	LR, Probit
Banasik et al. [4]	Bivariate probit	Probit
Kim et al. [17]	Bivariate probit	–
Banasik et al. [5]	Augmentation	Survival
Maldonado et al. [21]	Self-training	SVM
Maldonado et al. [21]	Co-training	SVM
Maldonado et al. [21]	Semi-supervised SVM	SVM
Chen et al. [10]	Bound and collapse	Bayesian
Bücker et al. [7]	Reweighting	LR
Siddiqi [28]	Define as bad	–
Siddiqi [28]	Soft cutoff augmentation	–
Siddiqi [28]	Hard cutoff augmentation	–
Siddiqi [28]	Parceling	–
Siddiqi [28]	Nearest neighbors	–
Anderson et al. [1]	Mixture modeling	LR
Li et al. [19]	Semi-supervised SVM	SVM

Abbreviations: LR = logistic regression, LDA = linear discriminant analysis, QDA = quadratic discriminant analysis, SVM = support vector machine.

5.3 Methodology

5.3.1 Self-Learning for Reject Inference

In reject inference, we are given a set of n examples $x_1, ..., x_n \in \mathbb{R}^k$, where k is the number of features. Set X consists of l accepted clients $x_1^a, ..., x_l^a \in X^a$ with corresponding labels $y_1^a, ..., y_l^a \in \{good, bad\}$ and m rejected examples $x_1^r, ..., x_m^r \in X^r$, whose labels are unknown. To overcome sampling bias and eventually raise scorecard accuracy, reject inference aims at assigning labels $y_1^r, ..., y_m^r$ to the rejected examples, which allows using the combined data for training a scoring model.

Standard self-learning starts with training a classifier $f(x)$ on the labeled examples $x_1^a, ..., x_l^a$ and using it to predict the unlabeled examples $x_1^r, ..., x_m^r$. Next, the subset of unlabeled examples $X^* \subset X^r$ with the most confident predictions is selected such that $f(x_i^* \in X^*) > \alpha$ or $f(x_i^* \in X^*) < 1 - \alpha$, where α is a probability threshold corresponding to a specified percentile of $f(x_i^* \in X^r)$. The selected rejects are labeled in accordance with the classifier's predictions. Cases obtained within this process are removed from X^r and appended to X^a to form a new labeled sample X_1^a. Finally, the classifier is retrained on X_1^a and used to score the remaining cases in X^r. The procedure is repeated until all cases from X^r are assigned labels or until certain stopping criteria are fulfilled [25].

Self-learning assumes that labeled and unlabeled examples in X follow the same distribution [25]. In a credit scoring context, X^a and X^r come from two different distributions because the scoring model employed by the financial institution separates accepts and rejects based on their feature values. The difference in distributions has negative consequences for self-learning: since the initial model is trained on a sample that is not fully representative of the unlabeled data, predictions of this model for the rejects are less reliable. The error is propagated through the iterative self-learning framework, which deteriorates the performance of the final model due to the incorrectly assigned labels.

In this section, we describe a novel shallow self-training framework for reject inference that is geared toward reducing the negative effects of sample bias. The proposed framework consists of three stages: filtering, labeling and model training. We summarize the algorithm steps in Algorithm 2.

Within the proposed framework, we suggest to filter and drop some rejected cases before assigning them with labels. The goal of the filtering stage is two-fold. Firstly, we strive to remove rejected cases that come from the most different part of distribution compared to the accepts. Removing these cases would reduce the risk of error propagation, since predictions of the model trained on the accepts become less reliable as the distribution of cases to be predicted becomes more different from the one observed on the training data. Secondly, we remove rejects that are most similar to the accepted cases. Labeling such cases would potentially provide little new information for a scorecard and might even harm performance

due to introducing noise. Therefore, the filtering stage is aimed at removing the cases that could have a negative impact of the scorecard performance.

The filtering is performed with isolation forest, which is a novelty detection method that estimates the normality of a specific observation by computing the number of splits required to isolate it from the rest of the data [20]. We train isolation forest on all accepts in X^a and use it to evaluate the similarity of the rejects in X^r. Next, we remove rejects that are found to be the most and least similar to the accepts by dropping cases within the top β_t and bottom β_b percentiles of the similarity scores. Algorithm 1 describes the filtering stage.

1 train isolation forest classifier $g(x)$ using all data in X^a;
2 use $g(x)$ to evaluate similarity scores of all unlabeled examples in X^r;
3 select a subset $X^* \subset X^r$ such that $g(x_i^* \in X^*) \in [\beta_b, \beta_t]$, where β_b and β_t are values of pre-defined percentiles of $g(x_j^r \in X^r)$, $j = 1, ..., m$.

Algorithm 1: Isolation Forest for Filtering Rejected Examples

After filtering, we use self-learning with distinct labeling and training regimes to perform reject inference. While the scoring model is based on a tree-based algorithm (gradient boosting), we propose using a weak learner for labeling rejects because of its ability to produce better-calibrated predictions [23]. In this paper, we rely on L1-regularized logistic regression (L1) to label rejects.

Logistic regression is a parametric learner which assumes a Gaussian distribution of the data. Because of this assumption, predicted probabilities can be output directly by the sigmoid function. In contrast, XGB is a non-parametric learner which has more degrees of freedom and a higher potential for inductive bias reduction. Predicted scores produced by XGB are not well calibrated [23]. Consider the example score distributions of L1 and extreme gradient boosting (XGB) depicted in Figure 5.3.1. Here, adding regularization to logistic regression is important as we are dealing with high-dimensional data with noisy features. Compared to L1, the range of the output probabilities of XGB is wider.

Within the proposed framework, we require the labeling model to produce well-calibrated probabilities as we limit the number of selected rejects based on the predicted PD values. Furthermore, by using different base models for application scoring and reject inference, we strive to reduce bias and error propagation. Hence, using a weak learner for reject inference is more promising.

An important aspect of our framework is to account for a higher default rate among the rejects [21]. Recall that X is partitioned into accepts and rejects based on a scoring model that is currently employed by a financial institution. Assuming that the scoring model in place performs better than a random loan allocation, we expect that the default rate among rejects is higher than among accepts. To address that difference, we introduce the imbalance parameter θ into our self-learning framework. On each labeling iteration, we only select the

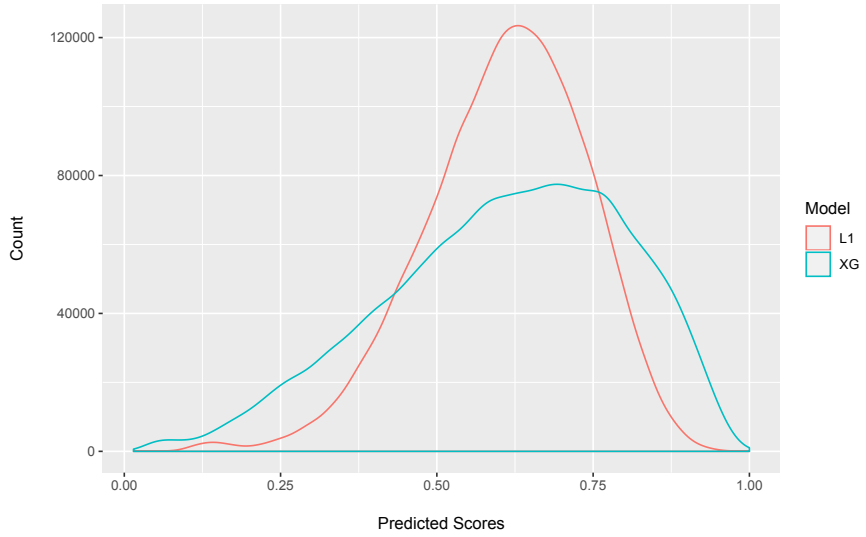

Figure 5.3.1. Predicted Score Densities

The figure compares the distributions of the scores predicted by two scoring models: L1-regularized logistic regression (red) and extreme gradient boosting (blue). Both models use the same data.

top $\alpha\%$ of the *good* loans and top $\alpha\theta\%$ of the *bad* loans among rejects for labeling. Keeping only the top-ranked instances ensures that we append rejects with high confidence in the assigned labels, reducing the potential amount of noise. By setting $\theta > 1$ we append more *bad* cases to the training data, accounting for the imbalance. Parameter θ can be optimized at the meta-parameter tuning stage.

filter rejected cases in X^r with isolation forest (see Algorithm 1);
set $X^* = X^r$;
while $X^* \neq \emptyset$ **do**
    train L1 classifier $f(x)$ with penalty parameter λ on all data in X^a;
    use $f(x)$ to predict PD for all unlabeled examples in X^*;
    if $c_b = \{\}$ *and* $c_g = \{\}$ **then**
        derive c_g: $P(f(x_i^* \in X^*) < c_g) = \alpha$, α is a percentile threshold;
        derive c_b: $P(f(x_i^* \in X^*) > c_b) = \alpha\theta$, θ is the imbalance parameter;
    end
    select $X^* \subset X^r$ such that $f(x_i^* \in X^*) < c_g$ or $f(x_i^* \in X^*) > c_b$;
    remove examples in X^* from X^r and append them to X^a;
end
train a scoring model $s(x)$ using XGB classifier on all cases in X^a.

Algorithm 2: Shallow Self-Learning for Reject Inference

Different variants of self-learning consider different ways to choose the most confident cases for labeling: either selecting top and bottom percentiles of the probability distribution

or selecting cases based on a pre-defined probability threshold [8]. We suggest using the combined approach: on the first iteration, we compute the corresponding score values c_g and c_b for the selected $\alpha\%$ and $\alpha\theta\%$ probability percentiles. Since the labeling model is geared toward providing well-calibrated probabilities, we fix the absolute values c_g and c_b as thresholds for the subsequent iterations. By doing that, we reduce the risk of error propagation on further iterations. The absence of rejected cases with predicted scores above the fixed thresholds serves as a stopping criterion.

5.3.2 Proposed Evaluation Measure

Performance evaluation is an important part of selecting a suitable reject inference technique. In practice, accurate evaluation of reject inference is challenging. The true labels of rejects are unknown, which prohibits estimating the accuracy directly. Therefore, prior research evaluates the performance of a given technique by comparing the performance of the scorecard before and after appending the labeled rejects to the training data [3, 6, 19]. The major downside of this approach is that the performance of a scorecard is not evaluated on a representative sample, which should include both accepts and rejects. Since labels of rejects are unknown, the literature suggests to evaluate models on a holdout sample drawn from the accepts which exhibits sample bias [e.g., 21]. Very few empirical studies have access to the data on both accepts and rejects for evaluation [11].

Model selection based on the performance on accepts might lead to selecting a suboptimal model. Let us illustrate that by comparing the performance of different scoring models validated on the accepts (4-fold stratified cross-validation) and on the unbiased sample consisting of both accepts and rejects. We train a set of scoring models with different meta-parameter values and evaluate their performance in terms of the area under the receiver operating characteristic curve (AUC) [26]. Here, XGB is used as a base classifier. Figure 5.3.2 depicts the results.

The rank correlation between AUC values is just 0.0132. Due to the distribution differences between the accepted and rejected cases, the model's performance on the accepted applicants becomes a poor criterion for model selection. This result suggests that there is a need to develop an alternative measure for comparing and evaluating the scoring models in the presence of sample bias.

Without access to an unbiased sample that contains data on a representative set of applicants, the literature suggests performing the evaluation by using synthetic data [16], emulating rejected cases by artificially moving the acceptance threshold [21] or using other criteria based on the applicants' feature values [9]. In this paper, we suggest using *kickout* – a novel evaluation measure based on the known data. We argue that developing such a measure is a valuable contribution since obtaining an unbiased data sample for performance evaluation is costly.

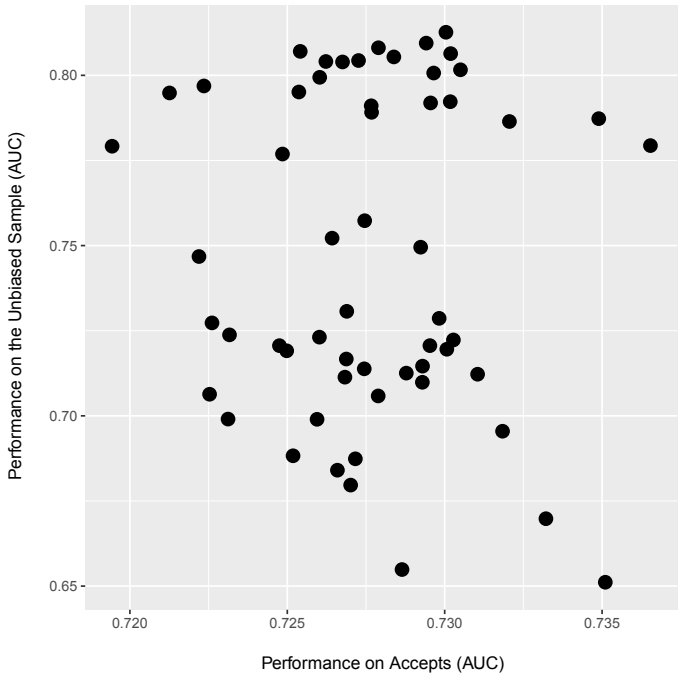

Figure 5.3.2. Comparing AUC on Accepts and the Unbiased Sample

The figure plots the AUC values obtained on the data from the accepted cases (using 4-fold stratified cross-validation) and the unbiased holdout sample. The dots indicate scoring models with different meta-parameter values.

The key idea of *kickout* is to compare a set of applications accepted by a scoring model before and after reject inference. Recall that we have data on the previously accepted X^a and rejected applicants X^r. Here, we partition X^a into two subsets: X^a_{train} and $X^a_{holdout}$. Let $s_1(x)$ be a scoring model trained on X^a_{train}. We use $s_1(x)$ to score cases from $X^a_{holdout}$ and select a pool of customers $A_1 \subset X^a_{holdout}$ that would be accepted by the model using the acceptance rate μ. Thus, A_1 contains the (simulated) accepted applications before reject inference.

The rejected cases in X^r are also split into two subsets: X^r_{train} and $X^r_{holdout}$. The former is labeled with a reject inference technique and appended to the X^a_{train}. Rejected cases in $X^r_{holdout}$ are appended to $X^a_{holdout}$, which now contains labeled accepts and unlabeled rejects, simulating the production-stage environment. Next, we train a new scoring model $s_2(x)$ on the expanded training sample X^a_{train} and use it to score and select customers in $X^a_{holdout}$ using the same acceptance rate μ. Since both training and holdout samples have changed, model $s_2(x)$ would accept a different pool of customers A_2. Analyzing the differences between A_1 and A_2, we can identify the kicked-out cases – applications that were included in A_1 but do not appear in A_2.

We define the *kickout* metric as follows:

$$kickout = \frac{\frac{\mathrm{K}_B}{\mathrm{p}(B)} - \frac{\mathrm{K}_G}{1-\mathrm{p}(B)}}{\frac{\mathrm{S}_B}{\mathrm{p}(B)}}, \qquad kickout \in [-1, 1] \tag{5.3.1}$$

where K_B is the number of *bad* cases kicked out from the set of accepted cases after performing reject inference, K_G is the number of kicked-out *good* cases, S_B is the number of *bad* cases selected by the original model, and $\mathrm{P}(B)$ is the share of *bad* cases in A_1. The *kickout* metric ranges from -1 (all *good* cases and no *bad* cases are kicked out) to 1 (all *bad* cases and no *good* cases are kicked out). We normalize the metric by the share of *bad* cases to reflect the difficulty of kicking out a *bad* customer. Positive values of *kickout* signal a positive impact of reject inference, with higher values indicating a better performance.

It is important to note that *kickout* does not require knowing the actual labels of the rejected cases that replace previously accepted cases. Instead, the metric focuses on the kicked-out applications. Replacing a *bad* loan with a rejected case may have two possible outcomes. If the newly selected rejected case is also *bad*, we are indifferent between the old and the new scoring model. If the rejected case is *good*, the scoring model improves. Therefore, kicking out a *bad* case has a positive expected value. In contrast, kicking out a *good* case has a negative expected value: we are indifferent between the old and the new scoring model if the new rejected case is *good*, whereas scorecard performance deteriorates if the rejected case is *bad*. Hence, a good reject inference technique should change a scorecard such that it starts to kick out more *bad* and less *good* customers.

The proposed measure relies on two assumptions. First, we assume that all *bad* loans and all *good* loans have the same expected value: that is, replacing one *bad* case with another *bad* case does not have any effect on the model's performance. Given the stable interest rates that determine the return on investment at fixed terms [31] and an uncertain relationship between a loan amount and its PD, we argue that this assumption is reasonable in a credit scoring context. Second, we assume that the *bad* ratio among rejected cases is higher compared to the accepted applications. As we detailed above, this assumption holds if the employed scoring model performs better than random.

5.4 Experimental Results

5.4.1 Data Description

The empirical experiments are based on a real-world credit scoring data set on consumer micro-loans provided by Kreditech, a Germany-based financial institution. Although the data are not available publicly, it provides a unique opportunity to study reject inference on a high-dimensional data set which includes an unbiased sample with customers who have been granted a loan without scoring.

The data set contains 2,410 features describing the applicants, their behavior and loan characteristics. The target variable is a binary indicator of whether the customer has repaid the loan. The data consist of 59,593 loan applications, out of which 39,579 were accepted and 18,047 were rejected. The target variable is only observed for the accepts, whereas the

Table 5.4.1. Data Summary

Characteristic	Accepts	Rejects	Unbiased
Number of cases	39,579	18,047	1,967
Number of features	2,410	2,410	2,410
Default rate	0.39	unknown	0.66

repayment status of rejects is unknown. Table 5.4.1 summarizes the main characteristics of the data set.

The unbiased sample contains 1,967 customers accepted without scoring. The sample, therefore, includes cases that would normally be rejected by a scorecard. This makes it representative of the through-the-door population of customers who apply for a loan. As noted in Table 5.4.1, the default rate in the unbiased sample is 1.7 times higher than on the accepted cases. The unbiased sample allows us to evaluate the performance gains from reject inference on the sample representative of the production environment.

5.4.2 Experimental Setup

To evaluate the effectiveness of our propositions, we perform two experiments. Experiment 1 benchmarks the proposed self-learning framework against conventional reject inference techniques and standard self-learning. In the second experiment, we illustrate the effectiveness of the new *kickout* measure for model selection. Below, we describe the modeling pipeline for these experiments.

We partition the data into three subsets: accepts, rejects and the unbiased holdout sample. Next, we use 4-fold stratified cross-validation on accepts to perform reject inference. On each iteration, the training folds are used to develop a reject inference technique that is used to infer labels of the rejects. Next, labeled rejects are appended to the training folds, providing a new sample to train a scoring model. Finally, a scoring model after reject inference is evaluated on the remaining fold and on the holdout sample. To ensure robustness, we evaluate performance on 50 bootstraped samples of the holdout set. Performance metrics of the reject inference techniques are then averaged over 4×50 obtained values.

We use XGB classifier as a scoring model in both experiments. Meta-parameters of XGB are tuned once on a small subset of training data using grid search. Within the experiments, we employ early stopping with 100 rounds while setting the maximum number of trees to 10,000 to fine-tune the model for each fold.

In Experiment I, we compare the suggested self-learning framework to the following benchmarks: ignore rejects, label all rejects as *bad* risks, hard cutoff augmentation, parceling, cross-validation-based voting and standard self-learning. Here, cross-validation-based voting is an adaption of a label noise correction method suggested by [30]. It refers to an extension of hard cutoff augmentation that employs a homogeneous ensemble of classifiers based on

different training folds instead of a single scoring model to label the rejects. The labels are only assigned to the cases for which all individual models agree on the label.

We test multiple versions of each reject inference technique with different meta-parameter values using grid search. For shallow self-learning, penalty λ of the labeling model is tuned and optimized once on the first labeling iteration. Table 5.4.2 provides the candidate values of meta-parameters.

For performance evaluation, we use three metrics that capture different dimensions of the predictive performance: AUC, Brier Score (BS) and R-Precision (RP). We use AUC as a well-known indicator of the discriminating ability of a model. In contrast, BS measures the calibration of the predicted default probabilities. Last, we use RP as it better reflects the business context. The financial institution that provided data for this study decides on a loan allocation by approving a certain percentage of the least risky customers. RP measures performance only for cases which will indeed be accepted. In our experiments, we compute RP in the top 30% of the applications with the lowest predicted PDs.

In Experiment II, we compare different variants of self-learning using grid search within the cross-validation framework described above. Apart from the three selected performance measures, we also evaluate reject inference in terms of the proposed *kickout* measure. The goal of this experiment is to compare model rankings based on three evaluation strategies: performance on the accepts, performance on the unbiased sample and performance in terms of *kickout*.

Table 5.4.2. Reject Inference Techniques: Parameter Grid

Technique	Parameter	Candidate values
Label all as *bad*	–	–
Hard cutoff augmentation	probability threshold	0.3, 0.4, 0.5
Parceling	multiplier	1, 2, 3
	no. batches	10
CV-based voting	probability threshold	0.3
	no. folds	2, 5, 10
Regular self-learning	labeled percentage α	0.01, 0.02, 0.03
	max no. iterations	5
Shallow self-learning	filtered percentage β_b	0, 0.02
	filtered percentage β_t	1, 0.98
	penalty parameter λ	$2^{-8}, 2^{-7.5}, ..., 2^{8}$
	labeled percentage α	0.01, 0.02, 0.03
	imbalance parameter θ	1, 2
	max no. iterations	5

5.4.3 Empirical Results

Experiment I: Assessing the Shallow Self-Learning

Table 5.4.3 summarizes the performance of the reject inference techniques on the accepted cases and on the unbiased sample. Recall that the latter serves as a proxy for the production-stage environment for a scoring model, whereas performance on accepts refers to a conventional approach toward evaluation in credit scoring. According to the results, not all methods improve on the benchmark of ignoring rejects: only three out of six techniques achieve higher AUC and lower BS on the unbiased sample, and only one has a higher RP.

Labeling rejects as *bad* performs better than disregarding reject inference on the accepts but does substantially worse on the unbiased sample. In contrast, parceling is outperformed by all other techniques on the accepts but has higher AUC on the unbiased sample. These results support the argument that performance on accepts might be a poor indicator of the production-stage performance.

Regular self-learning outperforms ignoring rejects in terms of AUC and BS but does not improve in terms of RP. The proposed self-learning framework performs best in all three measures on the unbiased sample as well as on the accepted applicants. The best performance is achieved by a self-learning model that includes filtering of rejects ($\beta_b = 1 - \beta_t = 0.02$). Therefore, the suggested modifications help to adjust self-learning for the reject inference problem.

Performance gains appear to be modest, supporting the prior findings [15]. We check statistical significance of the differences using Friedman's rank sum test and Nemenyi pairwise test [14]. The Friedman's rank sum test rejects the null hypothesis that all reject inference techniques perform the same at 1% level for AUC ($\chi^2 = 419.82$), RP ($\chi^2 = 326.99$) and BS ($\chi^2 = 485.59$). Nemenyi test indicates that shallow self-learning performs significantly better than all competitors in terms of AUC and RP, whereas differences in BS between standard and shallow self-learning are not statistically significant at 5% level.

Even small differences might have a considerable effect on the costs of the financial

Table 5.4.3. Comparing Performance of Reject Inference Techniques

Method	Accepted cases			Unbiased sample		
	AUC	BS	RP	AUC	BS	RP
Ignore rejects	0.7297	0.1829	0.8436	0.8007	0.2092	0.7936
Label all as bad	0.7332	0.1816	0.8474	0.6797	0.2284	0.7253
Hard cutoff augmentation	0.7295	0.1770	0.8430	0.7994	0.2212	0.7751
Parceling	0.7277	0.1842	0.8430	0.8041	0.1941	0.7851
CV-based voting	0.7293	0.1804	0.8430	0.7167	0.2160	0.7510
Regular self-learning	0.7302	0.1758	0.8434	0.8063	0.1838	0.7929
Shallow self-learning	**0.7362**	**0.1736**	**0.8492**	**0.8070**	**0.1799**	**0.7996**

Table 5.4.4. Correlation between Evaluation Strategies

Evaluation strategy	(1)	(2)	(3)
(1) AUC on the accepted cases	1		
(2) AUC on the unbiased sample	−0.0009	1	
(3) The kickout metric	0.0336	0.4069	1

institution [27]. Comparing shallow self-learning to ignoring rejects, 0.006 increase in RP translates to 60 less defaulted loans for every 10,000 accepted clients. Considering the average personal loan size of $17,100 and interest rate of 10.36% observed in the US in Q1 2019[1], potential gains from reject inference could amount for up to $1.13 million depending on the recovery rates.

Experiment II: Evaluation Strategy for Model Selection

In the second experiment, we perform model selection on 28 variants of self-learning with different meta-parameter values. Table 5.4.4 displays the correlation between model ranks in terms of three evaluation measures: AUC on the accepts, AUC on the unbiased sample and the *kickout* measure.

The absolute value of rank correlations between the performance on the accepts and performance on the unbiased data does not exceed 0.01. In contrast, the rankings based on *kickout* are positively correlated with those on the unbiased sample ($r = 0.41$). Therefore, the common practice to assess reject inference strategies using the model's performance on the accepted cases provides misleading results as there is a very small correlation with the performance on the production stage. In contrast, comparing reject inference techniques using the proposed *kickout* measure is more promising.

Figure 5.4.1 illustrates the advantages of using *kickout* instead of the performance on the accepts for model selection. Red points indicate the predictive performance of a scoring model selected by the *kickout* measure, while green dots refer to the best-performing model on the accepts in terms of AUC, BS and RP. As before, we evaluate the selected scoring models on the unbiased sample.

As shown in Figure 5.4.1, using the *kickout* measure results in selecting a better model in terms of all three performance indicators. By relying on *kickout* instead of the performance on the accepts, we are able to identify a scorecard that has a better performance on the unbiased sample.

These results emphasize the importance of using a suitable evaluation strategy to assess the value of reject inference. Relying on conventional evaluation measures such as AUC that are estimated on the accepted cases would result in selecting a suboptimal scoring model in terms of its production-stage performance. Our experiments show that *kickout* proves to be

[1]Source: https://www.supermoney.com/studies/personal-loans-industry-study/

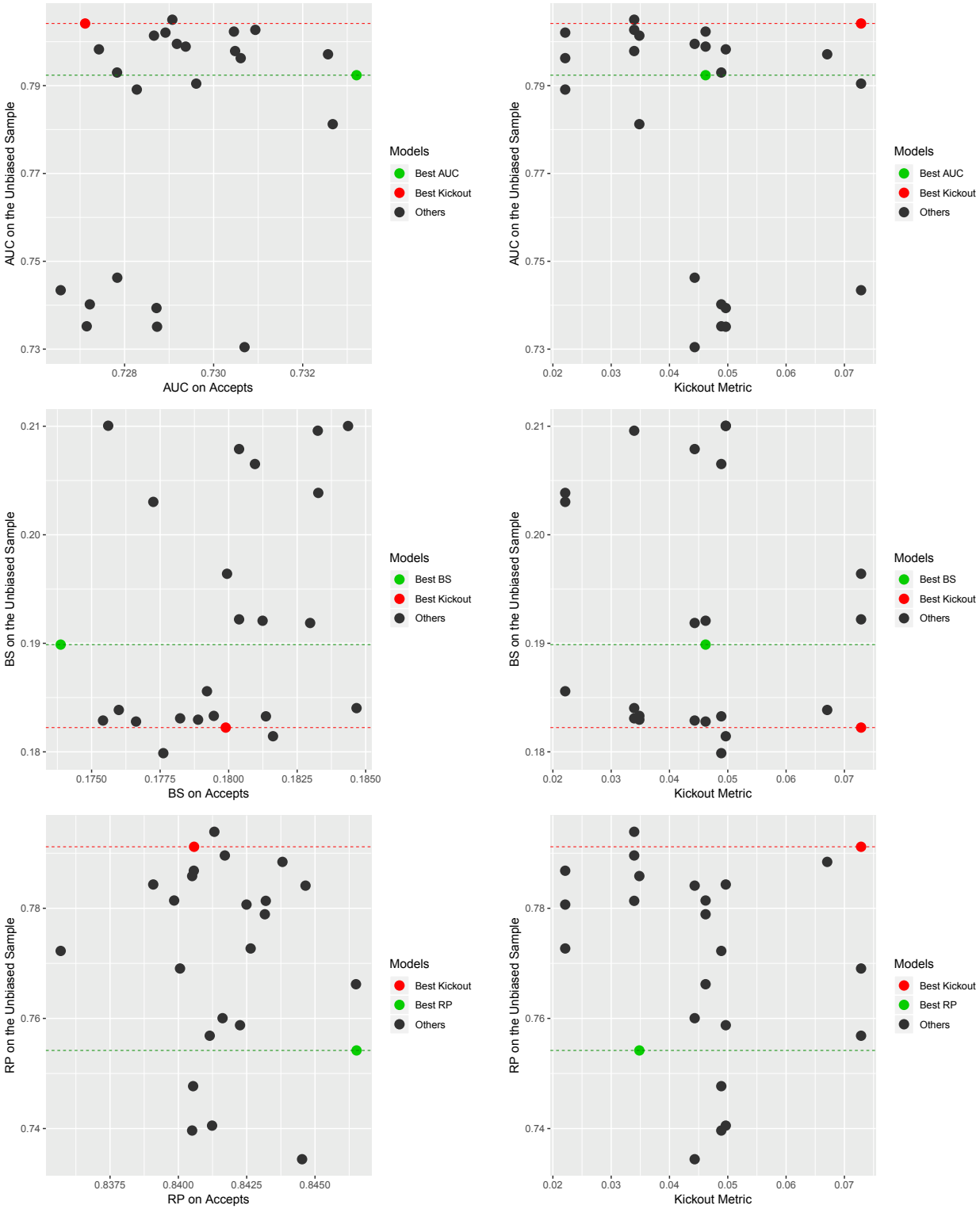

Figure 5.4.1. Model Selection Results

The two upper diagrams compare results based on AUC on the accepts (green) and on the kickout metric (red). The two diagrams in the center compare results based on Brier Score on the accepts (green) and kickout (red). The two lower diagrams refer to R-Precision on the accepts (green) and kickout (red).

a suitable measure for doing model selection. According to the results, the *kickout* measure identifies a better scoring model in the absence of an unbiased sample, which is particularly useful for practitioners.

5.5 Conclusion

This paper suggests a self-learning framework with distinct training and labeling regimes for reject inference in credit scoring and develops a novel evaluation measure for model selection. We evaluate the effectiveness of our approach by running empirical experiments on a high-dimensional real-world credit scoring data set with unique properties.

Empirical results indicate that the proposed self-learning framework outperforms regular self-learning and conventional reject inference techniques in terms of three performance measures. These results indicate that the modifications suggested here help to adjust self-learning to the reject inference problem.

We also develop a novel evaluation measure to perform model selection for reject inference techniques. We show that the standard practice of selecting models (or meta-parameters) based on their performance on the accepted cases may lead to choosing a model with a suboptimal predictive performance at the production stage. Compared to the standard approach, the proposed *kickout* measure exhibits a higher correlation with the performance on the unbiased sample and allows to identify a scoring model with better performance.

Our results imply that future research on reject inference should not rely on the model's performance on the accepted cases to judge the value of a certain reject inference technique. The *kickout* measure proves to be a good alternative for practitioners who often do not have access to an unbiased sample that contains both accepted and rejected applications.

Bibliography

[1] Anderson, B., Hardin, J.M. (2013). Modified logistic regression using the EM algorithm for reject inference. *International Journal of Data Analysis Techniques and Strategies 5*(4), 359–373.

[2] Ash, D., Meester, S. (2002). Best practices in reject inference. *Presentation at Credit Risk Modeling and Decision Conference. Wharton Financial Institutions Center, Philadelphia, May.*

[3] Banasik, J., Crook, J. (2005). Credit scoring, augmentation and lean models. *Journal of the Operational Research Society, 56*(9), 1072–1081.

[4] Banasik, J., Crook, J. (2007). Reject inference, augmentation, and sample selection. *European Journal of Operational Research, 183*(3), 1582–1594.

[5] Banasik, J., Crook, J. (2010). Reject inference in survival analysis by augmentation. *Journal of the Operational Research Society, 61*(3), 473–485.

[6] Banasik, J., Crook, J., Thomas, L. (2003). Sample selection bias in credit scoring models. *Journal of the Operational Research Society, 54*(8), 822–832.

[7] Bücker, M., van Kampen, M., Krämer, W. (2013). Reject inference in consumer credit scoring with nonignorable missing data. *Journal of Banking & Finance, 37*(3), 1040–1045.

[8] Chapelle, O., Schölkopf, B., Zien, A. (2006). *Semi-Supervised Learning.* MIT Press.

[9] Chen, G.G., Astebro, T. (2001). The economic value of reject inference in credit scoring. *Proc. 7th Credit Scoring and Credit Control Conference*, 309–321.

[10] Chen, G.G., Åstebro, T. (2012). Bound and collapse bayesian reject inference for credit scoring. *Journal of the Operational Research Society, 63*(10), 1374–1387.

[11] Crook, J., Banasik, J. (2004). Does reject inference really improve the performance of application scoring models? *Journal of Banking & Finance, 28*(4), 857–874.

[12] Feelders, A. (2000). Credit scoring and reject inference with mixture models. *Intelligent Systems in Accounting, Finance & Management, 9*(1), 1–8.

[13] Fogarty, D.J. (2006). Multiple imputation as a missing data approach to reject inference on consumer credit scoring. *Interstat, 41*, 1–41.

[14] García, S., Fernández, A., Luengo, J., Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences, 180*(10), 2044–2064.

[15] Hand, D.J., Henley, W.E. (1993). Can reject inference ever work? IMA Journal of *Management Mathematics, 5*(1), 45–55.

[16] Joanes, D.N. (1993). Reject inference applied to logistic regression for credit scoring. *IMA Journal of Management Mathematics, 5*(1), 35–43.

[17] Kim, Y., Sohn, S. (2007). Technology scoring model considering rejected applicants and effect of reject inference. *Journal of the Operational Research Society, 58*(10), 1341–1347.

[18] Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

[19] Li, Z., Tian, Y., Li, K., Zhou, F., Yang, W. (2017). Reject inference in credit scoring using semi-supervised support vector machines. *Expert Systems with Applications, 74*, 105–114.

[20] Liu, F.T., Ting, K.M., Zhou, Z.H. (2008). Isolation forest. *Proc. 2008 Eighth IEEE International Conference on Data Mining*, 413–422.

[21] Maldonado, S., Paredes, G. (2010). A semi-supervised approach for reject inference in credit scoring using SVMs. *Proc. Industrial Conference on Data Mining*, 558–571.

[22] Montrichard, D. (2007). Reject inference methodologies in credit risk modeling. *Proc. the South-East SAS Users Group.*

[23] Niculescu-Mizil, A., Caruana, R. (2005). Obtaining calibrated probabilities from boosting. *Proc. UAI*, 413.

[24] Reichert, A.K., Cho, C.C., Wagner, G.M. (1993). An examination of the conceptual issues involved in developing credit-scoring models. *Journal of Business & Economic Statistics, 1*(2), 101–114.

[25] Rosenberg, C., Hebert, M., Schneiderman, H. (2005). Semi-supervised self-training of object detection models. *Proc. 2005 Seventh IEEE Workshops on Applications of Computer Vision*, 29–36.

[26] Rosset, S. (2004). Model selection via the AUC. *Proc. the 21st International Conference on Machine Learning*, 89.

[27] Schebesch, K.B., Stecking, R. (2008). Using multiple SVMs models for unbalanced credit scoring data sets. *Proc. Data Analysis, Machine Learning and Applications*, 515–522.

[28] Siddiqi, N. (2012). *Credit risk scorecards: Developing and implementing intelligent credit scoring.* John Wiley & Sons.

[29] Smith, A., Elkan, C. (2004). A Bayesian network framework for reject inference. *Proc. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 286–295.

[30] Verbaeten, S., Van Assche, A. (2003). Ensemble methods for noise elimination in classification problems. *Proc. International Workshop on Multiple Classifier Systems*, 317–325.

[31] Verbraken, T., Bravo, C., Weber, R., Baesens, B. (2014). Development and application of consumer credit scoring models using profit-based classification measures. *European Journal of Operational Research, 238*(2), 505–513.

[32] Verstraeten, G., Van den Poel, D. (2005). The impact of sample bias on consumer credit scoring performance and profitability. *Journal of the Operational Research Society, 56*(8), 981–992.

[33] Wang, D., Zhang, Z., Bai, R., Mao, Y.: A hybrid system with filter approach and multiple population genetic algorithm for feature selection in credit scoring. *Journal of Computational and Applied Mathematics, 329*, 307–321.

[34] Wang, G., Hao, J.x., Ma, J., Huang, L.h. (2012). Empirical evaluation of ensemble learning for credit scoring. *Proc. Machine Learning: Concepts, Methodologies, Tools and Applications*, 1108–1127.

Chapter 6

Fighting the Sampling Bias: A Framework for Training and Evaluating Credit Scoring Models

Publication
Kozodoi, N., Lessmann, S., Alamgir, M., Gatsoulis, Y., Moreira-Matias, L., & Papakonstantinou, K. (2021). Fighting the Sampling Bias: A Framework for Training and Evaluating Credit Scoring Models. Submitted to Management Science (2nd review round).

Abstract
Scoring models support decision-making in financial institutions. Their estimation relies on the data of previously accepted applicants with known repayment behavior. This creates sampling bias: the training data offers a partial picture of the distribution of candidate borrowers to which the model is applied when screening new applications. The paper makes two contributions to address the adverse effect of sampling bias on model evaluation and training. First, we propose a Bayesian evaluation framework that extends standard evaluation metrics to the biased setting and provides a reliable estimate of future scorecard performance. To improve training, we develop Bias-aware self-learning – a reject inference framework that augments the biased training data by inferring labels for selected rejected applications. Extensive experiments on synthetic and real-world data confirm the superiority of our propositions over previous bias correction methods in terms of predictive performance and profitability and identify boundary conditions affecting their performance.

6.1 Introduction

The rise of big data and AI impacts management practices and decision processes in the financial industry. Financial institutions use scoring models to support resource allocation, inform risk management, and automate operational decision processes. A scoring model predicts the future state of a variable based on observational data. Credit scorecards are a prominent example. Estimating a borrower's probability to default, they support loan approval decisions and loss provisioning [11]. Generally speaking, the value of a scoring model depends on its ability to generate accurate predictions when processing new data not seen during model development [40]. We examine the practices underneath scorecard construction and argue that these create sampling bias, which diminishes the quality of scorecard-based decisions.

Application scorecards, which estimate an applicant's repayment ability, illustrate the

problem. To obtain the data required for scorecard estimation, a financial institution labels previous loan applications with a known outcome according to whether a debt was repaid or a default event occurred. We refer to the corresponding applications as *good* or *bad* risks. Class labels are observed for previously granted applications. Inevitably, the sample of accepted clients differs from the overall population of applicants, which includes applicants the scorecard would reject. Lacking the labels of rejected clients creates a missing data problem. Approving applications using a scorecard implies that application labels of rejected clients are either missing at random (MAR) or not at random (MNAR), which leads to sampling bias [62]. The bias impedes model training and evaluation. Training a scorecard on data from a biased sample may deteriorate the accuracy of its predictions when the model is used to screen new applications. Evaluating a model on a biased sample provides a misleading estimate of its actual performance.

The prevalence of scorecard-based decisions warrants concern about the sampling bias. In 2021, the total outstanding amount of consumer credit in the US exceeded \$4,325 billion[1]. Scorecards played a major role in the approval of this amount of credit. Given a trend to attain financing via financial technology companies (FinTechs), we expect the importance of scoring models to increase even further. Many FinTechs rely on a data-driven business model and the automation of loan approval. Thus, risk scores produced by scoring models increasingly determine access to finance, which plays a crucial role in economic inequality [101] and extends the impact of sampling bias beyond the accuracy of individual approval decisions. Applications of conceptually similar models to inform, for example, corporate lending [43] and the management of mortgage portfolios [82], corroborate this view. The availability of labeled data is crucial to supervised machine learning (ML), making sampling bias a serious concern in an increasingly data- and model-driven economy.

The goal of the paper is to shed light on the severity of sampling bias and develop strategies to mitigate its adverse effect on the two key steps of an ML pipeline, training and evaluation. Our first contribution is a new evaluation framework for scorecard assessment. Traditional performance measures such as, e.g., the area under a receiver operating characteristics curve (AUC), require labeled data. The labels are not available for rejected clients. Assessing a scorecard on accepts provides a misleading performance estimate. Reliable model validation is important for judging the model's business value, informing long-term planning and risk assessment decisions as well as performing the model selection. We propose a Bayesian evaluation framework that allows calculating an arbitrary performance measure on a representative sample from the borrowers' population that includes rejects. Drawing on prior knowledge, our framework avoids dependence on the actual labels of rejects and facilitates accurate evaluation under sampling bias.

Second, we introduce bias-aware self-learning (BASL) – a reject inference framework

[1]Source: The Federal Reserve (2021) Statistical Release on Consumer Credit, `https://www.federalreserve.gov/releases/g19/current`.

that mitigates the impact of sampling bias on scorecard performance. BASL augments the training data by labeling selected rejected cases and comprises procedures to address the high uncertainty associated with label estimation. For example, we establish the importance of involving learning algorithms with different characteristics – strong and weak learners – and propose a filtering stage to restrict the labeling to a suitable subset of rejected applications. The BASL framework extends our previous work on reject inference [55].

We test our propositions on synthetic and real-world data. First, we set up a controllable synthetic environment in which the labels of rejects are known and develop a data generation algorithm that mimics the loan approval cycle supported by a scoring model. The simulation study illustrates sampling bias and its adverse impact on the scorecard training and evaluation. It also allows us to investigate boundary conditions that influence the magnitude of the loss due to bias and the performance gains from our propositions. Second, we compare the proposed methods to established bias correction benchmarks on a real-world high-dimensional microloan data set. The data set includes a sample of applications that were randomly accepted without scoring. This sample represents the operating conditions of a scorecard and uncovers the true merit of bias correction [26]. The unbiased sample allows us to evaluate the performance of the proposed methods properly and measure performance gains in monetary terms.

It is worth noting that each of the two contributions of the paper can be used on a standalone basis. The first contribution ensures that scorecards are evaluated in a suitable way when sampling bias is present. The second contribution represents a reject inference framework that supports any supervised ML algorithm and can improve its performance under sampling bias. The two contributions combined constitute a holistic approach to sampling bias mitigation in credit scoring.

6.2 Theoretical Background

This section formalizes the sampling bias problem in credit scoring in relation with the missingness mechanisms. Let $X \in \mathbb{R}^k$ denote a loan applicant. The matrix of the applicants' attributes is denoted as $\mathbf{X} = (X_1, ..., X_n)^\top$, and $\mathbf{y} = (y_1, ..., y_n)^\top$ is a random vector of binary labels, indicating if the applicant repays the loan ($y = 0$) or defaults ($y = 1$). Suppose $\mathbf{X}$ and $\mathbf{y}$ have marginal distributions denoted as $\mathbf{P}_X$ and $\mathbf{P}_Y$ and a joint distribution $\mathbf{P}_{XY} = \mathbf{P}(y|X)$. Given a set of independent and identically distributed applications $D = \{(\mathbf{X}, \mathbf{y})\}$ with $(\mathbf{X}, \mathbf{y}) \sim \mathbf{P}_{XY}$, a financial institution uses a scorecard $f(X)$ that approximates $\mathbf{P}(y = 1|X)$ to split D into two subsets: accepts D^a and rejects D^r, $D = D^a \sqcup D^r$. The repayment behavior is eventually observed for applicants in D^a, while the labels of rejects remain unknown. In other words, D exhibits missingness with respect to $\mathbf{y}$.

The missingness mechanism has implications for credit scorecards. Let $\mathbf{a} \in \{0, 1\}$ denote a binary variable indicating if the applicant's repayment outcome is observed ($a = 1$) or

missing ($a = 0$), which corresponds to whether the applicant was accepted. Labels are missing completely at random (MCAR) if $\mathbf{P}(a|X,y) = \mathbf{P}(a)$, implying that missingness is not related to the data and no bias correction is needed. A finite-sample bias, which may occur due to limited sample size, can be reduced by collecting more data [6]. In credit scoring, MCAR occurs only if a bank accepts applications at random, which is unrealistic and does not warrant further consideration.

Filtering accepts using a scorecard causes D^a to have different empirical distributions compared to $\mathbf{P}_{XY}$, $\mathbf{P}_X$ and $\mathbf{P}_Y$ and creates sampling bias. We face MAR if $\mathbf{P}(a|X,y) = \mathbf{P}(a|X)$, which implies that the label missingness does not depend on the repayment status and is driven by the applicants' attributes $\mathbf{X}$. This occurs if a financial institution does not use any external information apart from $\mathbf{X}$ to make acceptance decisions (e.g., always relies on predictions of the same scorecard). Under MAR, posterior probability models such as logistic regression (LR) trained on a biased sample produce unbiased estimates and do not require bias correction [8]. However, the performance of certain classifiers may deteriorate. This concerns tree-based models that split the training data based on the observed feature values and, therefore, fail to extrapolate on new examples that lie outside of the previously observed feature ranges [71]. In credit scoring, tree-based classifiers such as random forest (RF) or extreme gradient boosting (XGB) were shown to outperform other benchmarks [e.g., 41, 57]. Using such models for scorecard development emphasizes the need for sampling bias correction in the MAR setting.

The MNAR setting is more challenging and implies that missingness depends on $\mathbf{y}$ due to unobserved factors that can not be explained through the attributes $\mathbf{X}$. Formally, the data exhibits MNAR if $\mathbf{P}(a|X,y) \neq \mathbf{P}(a|X)$. In practice, it is difficult to distinguish MNAR and MAR since the unobserved factors might not be accessible. In credit scoring, one of the main drivers of MNAR is manual overwriting of the scorecard predictions based on attributes not included in $\mathbf{X}$, which ties missingness to the factors unknown to the model $f(X)$. For instance, applicants with a County Court Judgment may be manually rejected by a decision-maker even if the scorecard prediction is positive [8]. MNAR can also occur when some of the features in $\mathbf{X}$ included in a previous scorecard can no longer be used by a financial institution (e.g., due to new data privacy regulations or changes in data providers). MNAR leads to biased model parameters [42, 62], which harms the performance of a model trained on a biased sample. The bias correction under MNAR is needed irrespective of the base classifier.

Apart from impacting model training, sampling bias adversely affects model evaluation under both MAR and MNAR. A validation subset H^a drawn from the labeled set D^a is not representative of D if the labels do not exhibit MCAR. As a result, evaluating $f(X)$ on a subset of previously accepted applicants will provide misleading performance estimates with regards to the actual performance of $f(X)$ on new loan applications drawn from $\mathbf{P}_{XY}$. In credit scoring, D^a contains applications predicted as least risky, which usually leads to

overoptimistic performance estimates when using accepts-based evaluation [9].

We formalize the research goal of this paper as follows: given a set of labeled accepts D^a and unlabeled rejects D^r, whereby labels are not MCAR, we strive to: (i) infer a function $f(X)$ that approximates $\mathbf{P}(y = 1|X)$ and generalizes well over applications from $\mathbf{P}_{XY}$ and (ii) estimate the predictive performance of $f(X)$ over applications from $\mathbf{P}_{XY}$. The task aims at improving the scorecard performance and the accuracy of estimates of scorecard performance, respectively. Exploiting the information in D^r can help to reduce the impact of sampling bias in both tasks.

6.3 Related Work

Sampling bias has received much attention in the literature. Prior work considers missing data problems when some examples are not observed due to non-random sampling [e.g., 25]. A related concept is domain adaptation, or data set shift, which studies differences in the training and test distributions due to, for example, a shift in the marginal feature distributions [88]. Model training and evaluation on biased incomplete data is also considered in the literature on off-policy learning and evaluation [31, 4]. This section reviews prominent bias correction methods and empirical studies on sampling bias in credit scoring. A survey of bias correction techniques is available in Table 6.9.1 in Appendix 6.9.1.

6.3.1 Training under Sampling Bias

Representation change is a family of bias correction methods applied in the data prepossessing stage before training a corrected model. Such methods assume MAR and use a mapping function Φ to project features into a new representational space $\mathbf{Z}$, $\Phi : \mathbf{X} \rightarrow \mathbf{Z}$, such that the training data distribution over $\mathbf{Z}$ is less biased and $\Phi(X)$ retains as much information about X as possible. A suitable representation is found by maximizing a distribution similarity measure such as the distance between the distribution moments in a kernel space [15] or during feature selection and/or transformation [e.g., 23, 80].

A recent feature transformation approach trains a deep autoencoder with a mismatch penalty and extracts the corrected data representation from the bottleneck layer [3]. Using such transformation harms model comprehensibility, whereas regulatory compliance requires financial institutions to ensure comprehensible scoring models [5].

Model-based bias correction methods modify a learning algorithm to account for the bias. In his pioneering work, Heckman [42] proposed a two-stage least-squares model for the MNAR setup. The Heckman model simultaneously estimates two equations: the outcome and the sample selection process, which allows to eliminate bias in the estimated model parameters and yield consistent estimates. Building on the linear Heckman model, Meng and Schmidt [77] developed a bivariate probit model with non-random sample selection for

setups where the outcome variable is binary. Their model represents a theoretically sound approach for the credit scoring setup under assumptions of MNAR and normally distributed residuals in the estimated equations.

Another research stream considers mixture models for bias correction [e.g., 32]. Mixture models operate under the MAR assumption and treat the data as drawn from a mixture of two distributions: training and population. Learning from the labeled training sample and unlabeled sample from the population, such models infer labels of new examples using the conditional expectation-maximization algorithm for maximum likelihood estimation.

The main disadvantage of model-based methods is that they are embedded in a learning algorithm, which requires a specific classifier. Previous work has mostly focused on linear and parametric models with particular assumptions. Yet, there is evidence that other non-parametric algorithms such as XGB demonstrate better performance in credit scoring [e.g., 41].

Reweighting is another method that rebalances the training loss towards representative examples. Weights of the training examples, also known as importance weights or propensity scores, can be computed as a ratio of the two distribution densities: $w(X) = p_D(X)/p_{D^a}(X)$. High values of $w(X)$ indicate that X is more likely drawn from $\mathbf{P}_{XY}$ and is, therefore, more important for training. Prior work suggests numerous techniques for importance weight estimation. For example, a model-based method estimates weights by fitting a classifier $c(X)$ on D using a binary sample indicator s as a label, where $s(X) = 1$ if $X \in D^a$ and 0 otherwise. Kernel Mean Matching [45] estimates density ratios by matching distributions in kernel space. Another idea is to use cluster-based empirical frequencies by splitting the data into clusters and computing weights as a ratio of test and training examples within clusters [25]. The importance weights can then be used during scorecard training using, for example, weighted least squares.

Since reweighting only relies on attributes in $\mathbf{X}$, it assumes MAR and can not correct for MNAR. However, reweighting can still be helpful under MNAR as it may reduce error in estimating a model from the training sample [8]. Another limitation of reweighting is that it faces difficulties in high-dimensional feature spaces where weight estimates exhibit high variance [100]. Last, a reweighted training set still consists of previously accepted clients and misses certain distribution regions populated by rejects only.

The credit scoring literature has also explored the idea of data augmentation – expanding the training sample by labeling and appending examples from D^r. The augmented sample covers a wider distribution region, which reduces sampling bias. Prior work suggests different approaches that use a model trained over D^a to label rejects. A classic example is hard cutoff augmentation (HCA), which labels rejects by comparing their scores predicted with the accepts-based model to a predefined threshold. Under sampling bias, reliance on the accepts-based model may increase the risk of error propagation when labeling rejects. Extrapolating predictions of the accepts-based scorecard on rejects is, therefore, a valid technique for

posterior probability classifiers under MAR but suffers from the omitted variable bias under MNAR [9].

Parceling aims to improve upon HCA by considering rejects as riskier than accepts. Parceling splits rejects into segments based on the predicted score range and labels rejects within each range proportional to the assumed probability of default in that range. The probabilities can then be altered by a decision-maker compared to the ones observed within the same score range on D^a. This implies that parceling can work in MNAR settings if the decision-maker is able to correctly specify the change in the default probabilities across the considered groups of applicants.

This paper introduces BASL – a reject inference framework that builds on self-learning based data augmentation and incorporates important extensions to account for the presence of sampling bias. The framework is model-agnostic and includes distinct regimes for labeling rejects and training a resulting scorecard. This allows us to reduce the risk of error propagation during labeling rejects and employ a classifier with high discriminative power for screening new applications.

6.3.2 Evaluation under Sampling Bias

The difference in data distributions also affects model evaluation. An estimate of model performance derived from a biased sample may not generalize to unseen data, which causes standard model evaluation strategies such as cross-validation to fail [91].

To address the evaluation problem, prior work proposes generalization error measures, whose asymptotic unbiasedness is maintained under sampling bias. This includes the modified Akaike information criterion [MAIC, 86] and the generalization error measure suggested by Sugiyama et al. [92]. Both measures rely on density ratio estimation to compensate for the distribution differences between the training and the test data and can thus be less accurate in high-dimensional feature spaces. Measures like the MAIC are also limited to parametric learners.

Evaluation under sampling bias is also studied in off-policy evaluation research, which focuses on the evaluating of a policy (e.g., a classifier) in a contextual bandit setting with incomplete historical data [87]. In this setup, a policy reward depends on the action of a decision-maker and is only partially observed. A prominent policy evaluation method is importance-weighted validation, which reweights the reward towards more representative examples in the evaluation set using importance weights [91]. In a binary classification setting, policy reward corresponds to an assessment of the scoring model performance using some evaluation metric.

Reweighting produces biased estimates if the past policy is modeled incorrectly [31]. In our setting, this implies that the attributes in $\mathbf{X}$ do not explain previous acceptance decisions accurately, and the data exhibits MNAR. For such cases, Dudik et al. [31] recommend doubly

robust (DR) estimators, which combine estimating importance weights with predicting policy reward (i.e., classifier loss). DR produces unbiased estimates if at least one of the modeled equations is correct. However, using DR in credit scoring is difficult. The contextual bandit setting considers a set of actions to decide on a case and assumes that we observe a reward for one of those actions. DR can then impute the reward for other actions. In credit scoring, however, we do not observe a reward for rejected clients, which complicates the imputation of reward substantially. Also, measuring reward as classifier loss limits DR to performance measures calculated on the level of an individual loan. This prohibits using DR with rank-based metrics such as the area under the ROC curve (AUC), which are established in credit scoring [e.g. 82].

This paper introduces a Bayesian evaluation framework that remedies the adverse impact of sampling bias on model evaluation and provides a more reliable estimate of model performance. The framework is metric-agnostic and allows evaluating any scoring model on a data sample with labeled accepts and unlabeled rejects. The framework leverages prior knowledge of the label distribution among rejects and uses Monte-Carlo sampling to optimize calculations.

6.3.3 Applications in Credit Scoring

Sampling bias has gained considerable attention in credit scoring. Previous research has mostly focused on the impact of sampling bias on scorecard training and tested some bias correction techniques including the Heckman model [e.g., 9], data augmentation techniques such as HCA [e.g., 26], and mixture models [e.g., 32]. A commonly used reweighting approach is banded weights, a cluster-based method that uses the bands of predicted probabilities of default to form clusters [7]. Recent studies also explore semi-supervised learning methods such as self-learning and semi-supervised SVMs [e.g., 59]. Several studies conclude that gains from reject inference are little or non-existent [7, 21]. At the same time, only a few studies express performance gains in terms of profitability [e.g., 21] or use a proper representative sample to measure gains from bias correction [e.g., 8]. Table 6.9.2 in the Appendix provides a detailed overview of empirical studies on reject inference.

The problem of evaluation under sampling bias has received less attention in the credit scoring literature. This can be attributed to limited data availability. Analyzing the impact of bias on evaluation requires a representative holdout sample that includes labeled applicants rejected by a scorecard. Seven out of 25 studies presented in Table 6.9.2 have (partial) access to the labels of some of the real rejects. However, two of these studies focus on corporate credit scoring, and the remaining five rely on just two distinct consumer credit scoring data sets. Using such proprietary data, Banasik et al. [9] illustrate the discrepancies between the accuracy estimates obtained on a sample from accepts and a representative holdout sample and use the latter to judge the value of reject inference. To the best of our knowledge,

previous work has not considered techniques that aim to correct the impact of sampling bias on model evaluation in the absence of such a sample.

Another limitation of empirical studies on reject inference is that the employed data sets are usually low-dimensional (see Table 6.9.2). While traditional banks still rely on parsimonious scorecards, this is not typical for FinTechs, which operate with large amounts of high-dimensional data from different sources [89]. Recent studies also indicate that financial institutions increasingly rely on alternative data such as applicants' digital footprints, e-mail activity and others [e.g. 12]. This trend emphasizes the importance of coping with high-dimensional data in reject inference.

This paper aims to address limitations of the prior work on sampling bias in credit scoring by employing a high-dimensional FinTech data set, evaluating performance on a representative sample from the borrowers' population, and examining the business impact of reject inference.

6.4 Bayesian Evaluation Framework

Estimating the performance of a scorecard before applying the model to screen new loan applications is a crucial task for decision-makers. An accurate estimate of the future model performance is necessary for judging the business value of the model, informing long-term planning and risk assessment decisions, and selecting a model variant expected to achieve better performance. Evaluating a model on a biased validation set of previously accepted applicants produces biased performance estimates. As a result, the actual scorecard performance in production does not match the expectations raised in the model evaluation stage. This section introduces the Bayesian evaluation framework that aims at mitigating the adverse effect of sampling bias on performance evaluation.

6.4.1 Evaluation Framework

Recall that we are given a population of loan applicants $D = D^a \sqcup D^r$, where D^a is accepts and D^r is rejects. Let $f(X)$ denote the scoring model to be evaluated. To infer its true ability to assess the creditworthiness of new applicants, $f(X)$ has to be evaluated on a representative holdout set denoted as $H, H \subset D$ [9]. Calculating standard evaluation measures would require knowledge of labels for all cases in H. However, in practice, only the labels in $H^a = H \cap D^a$ are known. The labels of rejects in $H^r = H \cap D^r$ are not available. A common approach, called accepts-based evaluation hereinafter, is to assess $f(X)$ based on H^a. Empirical results in Section 6.7 illustrate the inappropriateness of this approach and emphasize the need for improvement.

The proposed Bayesian framework extends standard performance metrics by incorporating rejects and available information on their label distribution. Assume $f(X)$ is evaluated

input : model $f(X)$, evaluation set H consisting of labeled accepts $H^a = \{(\mathbf{X}^a, \mathbf{y}^a)\}$ and unlabeled rejects $H^r = \{\mathbf{X}^r\}$, prior $\mathbf{P}(\mathbf{y}^r|\mathbf{X}^r)$, evaluation metric $\mathrm{M}(f, H, \tau)$, meta-parameters j_{max}, ε
output: Bayesian evaluation metric $\mathrm{BM}(f, H, \tau)$

$j = 0$; $\Delta = \varepsilon$; $E = \{\}$; // initialization
while $(j \leq j_{max})$ *and* $(\Delta \geq \varepsilon)$ **do**
  $j = j + 1$
  $\mathbf{y}^r = \mathrm{binomial}(1, \mathbf{P}(\mathbf{y}^r|\mathbf{X}^r))$; // generate labels of rejects
  $H_j = \{(\mathbf{X}^a, \mathbf{y}^a)\} \cup \{(\mathbf{X}^r, \mathbf{y}^r)\}$; // construct evaluation sample
  $E_j = \frac{1}{j} \sum_{i=1}^{j} \mathrm{M}(f, H_i, \tau)$; // evaluate f(X)
  $\Delta = E_j - E_{j-1}$; // check metric convergence
end
return : $\mathrm{BM}(f, H, \tau) = E_j$

Algorithm 3: Bayesian Evaluation Framework

on H using an arbitrary evaluation metric $\mathrm{M}(f, H, \tau)$, where τ is a vector of metric meta-parameters (e.g., classification cut-off). Algorithm 3 computes the Bayesian extension of the metric M denoted as $\mathrm{BM}(f, H, \tau)$. Since the labels of examples in H^r are unknown, we choose a prior of the label distribution among rejects $\mathbf{P}(\mathbf{y}^r, \mathbf{X}^r)$ and assign random pseudo-labels accordingly. This allows us to evaluate $f(X)$ on a representative sample consisting of labeled accepts and pseudo-labeled rejects.

Within the Bayesian evaluation framework, we employ Monte-Carlo sampling to optimize computation. Each unknown label is drawn from a binomial distribution with the probability set to the prior for that rejected example. The Bayesian extension of the metric is then computed by averaging the metric values across multiple label realizations. The sampling iterations are terminated once the incremental change of the average value does not exceed a convergence threshold ε.

As commonly observed in Bayesian estimation, the accuracy of the estimated metric depends on the choice of the prior. Instead of using the less informative and difficult to estimate class prior $\mathbf{P}(\mathbf{y}^r)$, we recommend leveraging the attributes of the cases in D^r denoted by $\mathbf{X}^r$. This enables estimating the prior $\mathbf{P}(\mathbf{y}^r|\mathbf{X}^r)$ by rescoring rejects with a model that has been used to support loan approval decisions in the past or use the original scores used in those decisions if they are available. The prior governs the sampling of class labels of rejects. In Section 6.7.1, we show that the scores of the past model should display high calibration. It is also essential that the past model has been trained on the historical data that is not part of the evaluation set H.

Note that a model trained on accepts to compute the prior also suffers sampling bias. The proposed evaluation framework stands on the trade-off of two components: the benefit from evaluating a model on a larger sample more representative of the population and the noise in the simulated labels of rejects. As we establish through empirical experimentation and illustrate in Section 6.7.1, gains from extending the evaluation sample outweigh losses

from the noise in the prior, which facilitates a good performance of the Bayesian framework.

6.4.2 Applications to Performance Metrics

One of the key advantages of the proposed evaluation framework is its support of arbitrary performance metrics. When evaluating credit scorecards, it is beneficial to take into account not only the operating conditions, such as the class priors and the misclassification costs, but also the financial institution's strategy, such as risk minimization, growth, or profit maximization. This ensures that the institution's objectives can be met by mapping the evaluation output to the company's key performance indicators and that all relevant information is used. The model-agnostic nature of the proposed framework allows it to benefit any institution irrespective of the chosen metric.

The AUC and the Brier Score (BS) are widely used evaluation measures in credit scoring that do not have any meta-parameters. The AUC is an established indicator of the discriminatory ability of a scorecard. Calculating the mean squared error between model-estimated predictions and a zero-one coded default indicator, the BS measures the degree to which the model predictions are well-calibrated. Under sampling bias, estimates of the AUC and BS on a sample from D^a will be misleading as D^a only represents a limited region of the target data distribution. A Bayesian extension of each of these two metrics can be computed on a holdout set H using Algorithm 3.

In credit scoring, accepting a *bad* applicant incurs higher costs than rejecting a *good* applicant. The Partial AUC (PAUC) summarizes the ROC curve on a limited range of thresholds and facilitates accounting for asymmetric error costs [99]. A financial institution can measure the PAUC to evaluate the ranking ability of a model in the area of the ROC curve with a low false negative rate (i.e., FNR $\in [0, \xi]$). The upper bound ξ is a meta-parameter of the metric and limits the range of thresholds to ensure that the acceptance rate is sufficiently low such that the target FNR is not exceeded. Similar to the AUC, we can estimate the Bayesian PAUC on H using Algorithm 3. In this paper, we compute the PAUC in the area of the ROC curve with FNR $\in [0, .2]$.

Assuming that a financial institution's objective is to maximize the acceptance rate while keeping credit losses below a certain threshold or to minimize credit losses while approving a specific percentage α of loan applications, a suitable evaluation metric is the *bad* rate among accepts (ABR), where accepts are the top $\alpha\%$ applications with the lowest estimated probabilities of default. Computing the ABR on H^a leads to an over-optimistic performance estimate since H^a already represents the top $\alpha\%$ applications from the population. The Bayesian ABR computed on H using the proposed evaluation framework provides a more reliable estimate of the *bad* rate among accepts. In this paper, we integrate the ABR over acceptance between 20% and 40%, which is a historical acceptance range on the real-world lending data set used in the paper.

6.5 Bias-Aware Self-Learning Framework

Training a scorecard on a biased sample results in a performance loss when the model is applied to screen new applications from D. This section introduces a reject inference framework aimed at mitigating the impact of sampling bias on training. We start by revisiting traditional self-learning that serves as a base for BASL and extend it to a setup where the data exhibits sampling bias.

6.5.1 Traditional Self-Learning

Self-learning is an incremental semi-supervised learning framework [58]. Given a labeled set D^a and an unlabeled set D^r, self-learning trains a supervised model over D^a. Next, the trained model is used to score examples in D^r. Examples where model predictions exceed the specified confidence thresholds are assigned the corresponding labels and appended to D^a. The classifier is then retrained on the augmented labeled sample to score the remaining unlabeled data. The procedure is repeated until a stopping criterion is met.

Sampling bias impedes the effectiveness of self-learning. First, labeling rejects based on the confident predictions of a model trained on accepts may be misleading. Since rejects come from a different distribution region, the accepts-based model can produce overconfident predictions that become less reliable as the difference between the two samples increases. The prediction errors are propagated through consecutive labeling iterations, impairing the resulting performance. The accuracy of the assigned labels is further threatened when using a strong learner, which may be prone to overfitting the biased training sample. Using the same confidence thresholds for labeling *good* and *bad* rejects also results in preserving the class ratio in the augmented labeled sample, whereas the *bad* ratio on a representative sample of loan applicants is expected to be higher than on accepts. Finally, employing commonly used stopping criteria based on the absence of examples with confident predictions may lead to exceeding the suitable number of labeling iterations, which risks overfitting the sample of accepts and can strengthen the error propagation due to the bias.

6.5.2 Bias-Aware Self-Learning

The proposed BASL framework addresses the limitations of traditional self-learning and extends it to a setup where the data is missing not completely at random. The extensions include: (i) introducing a filtering stage before labeling; (ii) implementing modifications to the labeling stage and training regime; (iii) introducing stopping criteria to handle sampling bias. The BASL framework is visualized in Figure 6.5.1. The pseudo-code is provided in Algorithm 6 in Appendix 6.9.2.

Note that BASL does not aim to solve the fundamental extrapolation problem by completely eliminating bias in the training data. This is not feasible as the repayment behavior

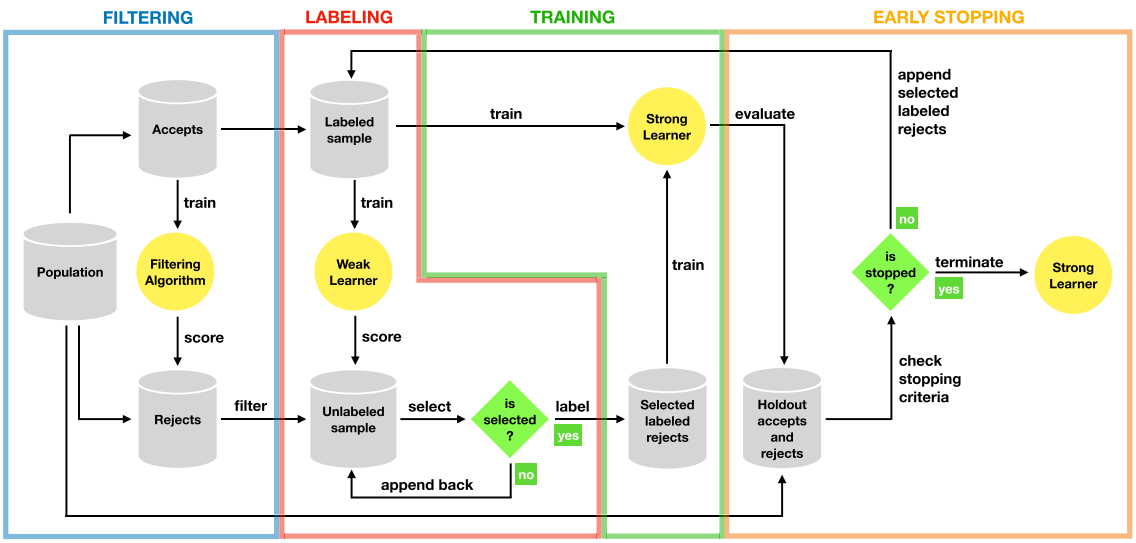

Figure 6.5.1. Bias-Aware Self-Learning Framework

Population is partitioned into labeled accepts, unlabeled rejects and a holdout set containing accepts and rejects. Filtering algorithm filters rejects before labeling; we use isolation forest trained on accepts. Weak learner assigns labels to rejects; we use LR. Strong learner screens new loan applications; we use XGB. Selection criteria: (i) random sample of rejects with a rate ρ; (ii) scores of weak learner exceed confidence thresholds based on percentile thresholds γ and imbalance multiplier θ. Stopping criteria: (i) reaching maximum no. iterations j_{max}, (ii) performance of strong learner measured with the Bayesian framework is not improving; (iii) set of selected labeled rejects is empty.

of rejects from a too distant distribution region is hard to estimate. Therefore, we ensure that the training data is only augmented with a few rejects, for which the labeling model is confident and the data distribution is not too different from accepts. By doing so, we can not expect the augmented sample to be free of sampling bias. Instead, we aim at reducing the bias while keeping the error propagation sufficiently low. The trade-off between bias reduction and scorecard accuracy is a crucial element of the BASL framework. Appendix 14 further investigates this trade-off by constructing bias-accuracy Pareto frontiers with BASL variants that label different subsets of rejects.

Filtering Stage

In the first stage, we filter rejects in D^r. The goal of the filtering is two-fold. First, we aim at removing rejects coming from the most different part of distribution compared to D^a. Removing such examples reduces the risk of error propagation since predictions of a model trained over accepts become less reliable as the distribution of examples to be labeled shifts further from the one observed during training. Second, we aim at removing rejects that are most similar to accepts. Labeling such examples would potentially provide little new information for a scorecard and might even harm performance due to introducing noise.

We estimate the similarity between rejects and previously accepted applications by scoring rejects in D^r with a novelty detection algorithm trained over D^a. To remove the rejects

most and least similar to accepts, we drop examples within the top β_u and bottom β_l percentiles of the predicted similarity scores. The threshold values $\beta = (\beta_u, \beta_l)$ act as meta-parameters of the filtering algorithm, which we implement using isolation forest, a scalable tree-based novelty detection algorithm suitable for high-dimensional feature spaces [65].

Labeling Stage

After filtering, we iteratively label selected rejects. We employ distinct regimes for labeling rejects and training the resulting scorecard and suggest scoring rejects using a learner with different inductive bias compared to the one employed for scorecard construction. The labeling algorithm should provide well-calibrated predictions to select the appropriate confidence thresholds. Another desideratum of the labeling algorithm is that it should be less prone to overfitting the biased training sample. Using different algorithms for reject inference and scoring new applications also reduces the risk of amplifying the bias of the base classifier.

We use L1-regularized LR as a weak learner for labeling rejects. The L1 penalty is introduced when working with high-dimensional data with noisy features. LR is a parametric learner that outputs probabilistic predictions. As we show in Appendix 6.9.2, predictions of LR are better calibrated and take extreme values less frequently compared to a strong non-parametric learner such as XGB. Another advantage of LR over tree-based models such as XGB is its ability to extrapolate outside of the feature value ranges observed on accepts [71], which is crucial since rejected applications are coming from a different distribution region.

On each labeling iteration, we randomly sample ρm examples from the available set of m rejects. Sampling aims at preventing overfitting by examining different regions of the distribution of rejects. Assuming that the currently deployed scorecard performs better than random, we expect the *bad* rate in D^r to be higher than that in D^a. To address this, we introduce the imbalance parameter θ. We only label examples in the bottom γ percentile and the top $\gamma\theta$ percentile of the distribution of scores predicted by the weak learner. This ensures that we select rejects with high confidence in the assigned labels and append more *bad* examples than *good* ones by setting $\theta > 1$. The latter helps to increase the *bad* rate in the training sample to approximate the population distribution. The selected labeled rejects are removed from D^r and appended to D^a. After the first iteration, we fix the absolute values of the confidence thresholds and use them on the following iterations.

Training Stage

At the end of each labeling iteration, we train a scoring model on the augmented labeled sample D^a containing accepts and selected labeled rejects. The augmented sample covers a wider range of the feature space compared to the original sample of accepted applications. This helps to reduce the adverse effect of sampling bias on the trained model. The training stage benefits from using a strong base learner to develop a scorecard with high discriminative power to screen new applications. We use XGB as a base classifier for the resulting scorecard.

Early Stopping

The number of labeling iterations is controlled by the stopping criteria. We use the Bayesian evaluation framework proposed in Section 6.4 to track the performance of the corrected scorecard across the labeling iterations. At the end of each iteration, we evaluate the scorecard on a holdout sample containing labeled accepts and unlabeled rejects. Evaluating a model with the Bayesian framework is important as it allows to account for the impact of sampling bias on evaluation. If the model performance does not improve, we stop labeling at this iteration and use the best-performing model as a resulting scorecard. We also specify the maximum number of labeling iterations j_{max} and terminate the BASL algorithm if there are no more rejects in D^r for which predictions exceed the specified confidence thresholds.

6.6 Experimental Setup

This section describes the data used in the paper and outlines the experimental setup. First, we use a controlled simulation environment to illustrate sampling bias, demonstrate gains from our propositions, and investigate boundary conditions affecting their performance. Next, we test our methods and quantify their business impact on a real-world high-dimensional microloan data set.

6.6.1 Synthetic Data

We generate synthetic loan applications using two multivariate mixtures of Gaussian distributions:

$$\begin{cases} \mathbf{X}^g \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^g, \boldsymbol{\Sigma}_c^g) \\ \mathbf{X}^b \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^b) \end{cases} \tag{6.6.1}$$

where $\mathbf{X}^g$ and $\mathbf{X}^b$ are feature matrices of *good* and *bad* applications, and δ_c, $\boldsymbol{\mu}_c$, and $\boldsymbol{\Sigma}_c$ are the weight, mean vector and covariance matrix of the c-th mixture component. The distribution parameters control the difference between the two applicant groups.

Mimicking the scorecard-based loan approval process, which leads to sampling bias, we introduce a simulation framework called the acceptance loop. We assume a financial institution approves loan applications using a scoring model $f_a(X)$ that predicts $\mathbf{P}(y = 1|X)$. The institution accepts the applicant X if $f_a(X) \leq \tau$, where τ is a probability threshold. Suppose $D_j = \{(\mathbf{X}, \mathbf{y})\}$ is the batch j of independent and identically distributed applications with $(\mathbf{X}, \mathbf{y}) \sim \mathbf{P}_{XY}$ where $\mathbf{y}$ is unknown at the time of application. Acceptance decisions partition D_j into $D_j^a = \{X_i \in \mathbf{X} | f_a(X_i) \leq \tau\}$ and $D_j^r = \{X_i \in \mathbf{X} | f_a(X_i) > \tau\}$ for accepts and rejects. Once the labels in D_j^a are available, the scoring model is updated by incorporating all labeled applications $D^a = \bigcup_{j=1}^{J} D_j^a$ during training and applied on new incoming applications, where J is the total number of batches. Over time, D^a grows in size with a bias towards accepts.

We run the acceptance loop for 500 iterations. On each iteration, we generate a new batch of applications using the same distribution parameters and train a scoring model $f_a(X)$ over D^a to split them into accepts and rejects. We also draw a representative holdout sample from $\mathbf{P}_{XY}$ denoted as H. The sample H is used to evaluate the performance of scorecards and bias correction methods on unseen data representative of the borrowers' population. A detailed description of the simulation framework and synthetic data generation is provided in Appendix 6.9.3.

Full control over the data generating process facilitates sensitivity analysis to clarify how the loss due to bias and gains from our propositions develop with changes in the environment and uncover boundary conditions. For example, Section 6.2 has discussed missingness mechanisms and how they impact the loss due to bias. Hence, the sensitivity analysis comprises a gradual transition from an MAR to an MNAR process. Other factors influencing the effectiveness of BASL include the strength of the sampling bias, the class imbalance ratio, and the complexity of the classification task. Similarly, the Bayesian framework depends on the validation set of labeled accepts and unlabeled rejects and the quality of the class prior for the labels of rejects. The sensitivity analysis proposes measures for these factors and examines their impact on our propositions.

6.6.2 Real Data

The real-world credit scoring data set is provided by a FinTech called Monedo and constitutes consumer microloans issued to customers in Spain. The data includes 2,410 features characterizing the loan applicants. The target variable is a binary indicator of whether the customer has timely repaid the loan (*good*) or experienced delinquency of at least three consecutive months (*bad*). The data consist of 59,593 loan applications, out of which 39,579 were accepted and 18,047 were rejected. The target variable is only observed for accepts, whereas the repayment outcome of rejected clients is unknown. Table 6.6.1 summarizes the real-world data set.

Apart from accepts and rejects, we also have access to a labeled unbiased holdout sample with 1,967 customers who have been granted credit without scoring. The sample, therefore, includes examples that would normally be rejected by a scorecard and represents the through-the-door population of customers who apply for a loan. This unbiased sample allows us to evaluate the performance gains from our propositions under the true operating conditions of

Table 6.6.1. Real Data Summary

Characteristic	Accepts	Rejects	Holdout
Number of applications	39,579	18,047	1,967
Number of features	2,410	2,410	2,410
Percentage of *bad* applications	39%	Unknown	66%

Monedo.

Table 6.6.1 shows that the *bad* rate in the holdout sample is 1.7 times higher than among accepts, which hints at the presence of sampling bias. Appendix 6.9.4 provides additional analysis confirming that the data do not exhibit MCAR and illustrating sampling bias and its adverse effect on the scorecard parameters, training and evaluation. The results indicate the potential of bias correction.

6.6.3 Experiments

The empirical evaluation focuses on two research questions. Experiment I tests whether the Bayesian framework provides a more reliable estimate of the scorecard performance on unseen data compared to other evaluation strategies. Experiment II focuses on training under sampling bias and tests whether the BASL framework outperforms conventional bias correction methods.

Experiment I compares evaluation strategies in a performance prediction setup. We split accepts into training and validation sets and apply evaluation strategies to a scorecard trained on the training data. Each strategy provides an estimate of the scorecard performance on a holdout sample representative of the borrowers' population. Ignoring sampling bias and evaluating on accepts is a naive benchmark. DR and reweighting act as off-policy evaluation benchmarks. Differences between the off-policy evaluation setup and our study prohibit the direct application of DR. Appendix 6.9.6 details our implementation of an adjusted DR estimator that supports credit scoring. The Bayesian framework evaluates the scorecard on a merged validation set of accepts and unlabeled rejects. To produce a prior on the labels of rejects, we score them with the XGB-based scorecard trained on accepts and calibrate the scores using LR. We judge the performance of an evaluation strategy by calculating the RMSE between the model performance estimates produced by that strategy over the experimental trials and the actual scorecard performance on the holdout sample.

In Experiment II, we correct the training set of accepts with one of the bias correction methods. The scoring model is trained over the corrected sample and evaluated on a representative holdout sample. We compare BASL to established techniques from different families of bias correction methods. Ignoring rejects serves as a baseline. Labeling rejects as *bad* and bureau score based labeling are simple augmentation techniques popular in credit scoring. HCA and parceling represent the model-based augmentation methods. The Heckman model is another benchmark suited for MNAR and established in the credit scoring literature. We also implement reweighting with cluster-based weights. The bias-removing autoencoder serves as a representation change benchmark.

The simulation study allows us to dynamically conduct the experiments within the acceptance loop and aggregate the results over 100 simulation trials. Knowledge of the actual labels of synthetic rejects also allows us to implement an oracle model $f_o(X)$ trained on

$D^a \cup D^r$. The oracle represents a scorecard that does not suffer from sampling bias and indicates an upper performance bound. The real data is static and does not support dynamic evaluation and an oracle scorecard. To improve the robustness of the results obtained on the real data, we aggregate performance over 100 values coming from 4 cross-validation folds times 25 bootstrap samples of the holdout sample. We use XGB as a base classifier in experiments on both data sets. Further details on the data partitioning and meta-parameter values of bias correction methods are provided in Appendix 6.9.5.

6.7 Results

6.7.1 Synthetic Data

This section presents empirical results on synthetic data. We start by illustrating sampling bias and gains from our propositions in the MAR setup. Next, we perform sensitivity analysis to investigate boundary conditions affecting the performance of BASL and the Bayesian evaluation framework. Last, we compare gains from our propositions and benchmarks depending on the missingness type.

Results in the MAR Setup

Figure 6.7.1 illustrates sampling bias and its adverse effects on the scorecard behavior, training and evaluation. Panel (a) compares the distribution densities of one of the synthetic features in D^a, D^r and H. The results indicate differences in the distribution of x_1 in D^a and H. The values of x_1 in $(-3, 1)$ are not observed among accepts in D^a, although the density peak in the unbiased set H is located within this interval. This confirms that the training data of previously accepted clients are not representative of the population of loan applicants.

Bias in the training data affects the scorecard behavior. We use a non-parametric XGB-based scorecard, which prohibits a direct inspection of the model parameters to illustrate the bias in the classifier. Regressing applicant features on the predictions of an XGB scorecard using linear regression, we obtain a surrogate model that approximates the way in which XGB translates feature values into predictions. Panel (b) compares the coefficients of the surrogate models corresponding to the three XGB scorecards: (i) biased model $f_a(X)$ trained over D^a; (ii) oracle model $f_o(X)$ trained over $D^a \cup D^r$; (iii) model $f_c(X)$ corrected by labeling rejects with BASL. The results indicate that sampling bias affects the coefficients of surrogate scorecards and causes them to diverge from the oracle values. BASL partly recovers this difference, bringing the coefficients closer to the oracle. The bias in model parameters translates into a difference in the scores predicted by the scorecards. As illustrated in panel (c), f_a provides more optimistic scores compared to f_o, whereas the distribution of scores produced by f_c is more in line with that of the unbiased model.

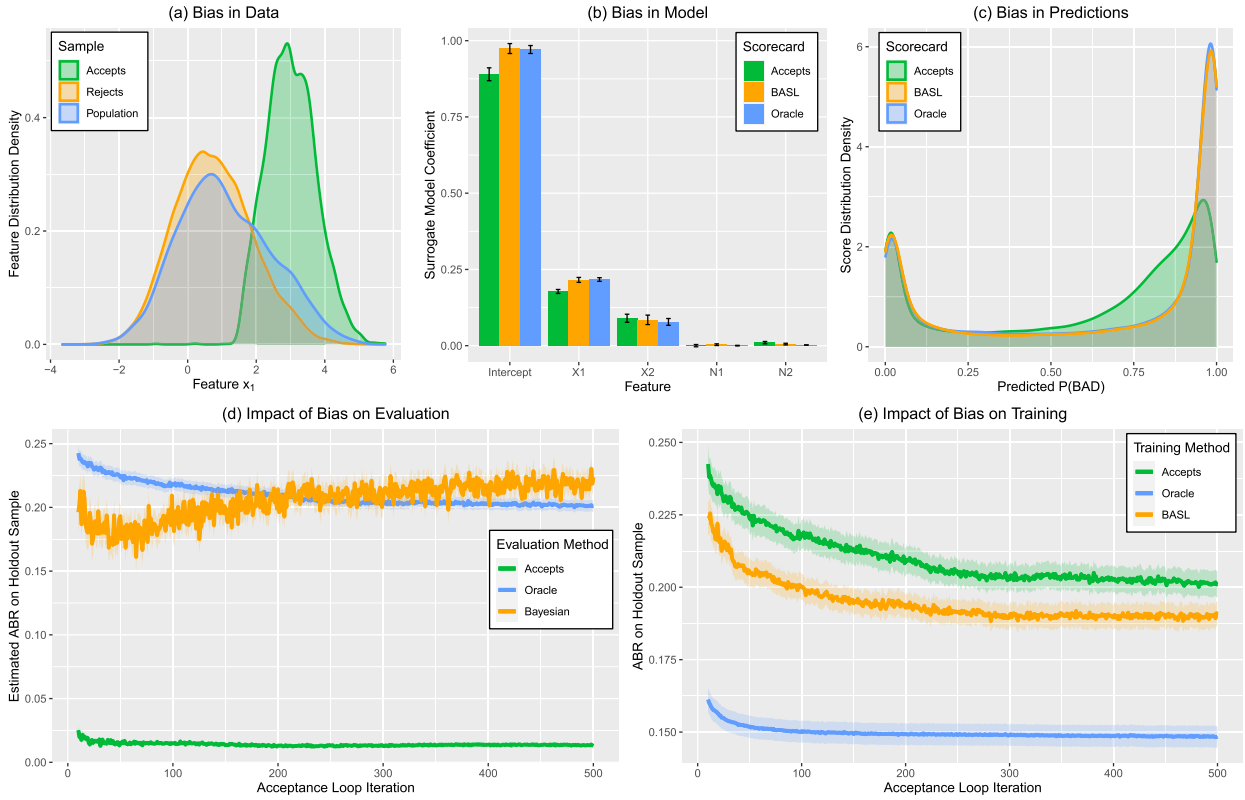

Figure 6.7.1. Loss due to Sampling Bias and Gains from Our Propositions

The figure illustrates sampling bias, its adverse effect on the scorecard behavior, training and evaluation, and gains from our propositions within the acceptance loop on synthetic data. Panel (a) demonstrates bias in the distribution of one of the features. Panel (b) visualizes bias in scorecards by comparing the coefficients of the corresponding linear surrogate models. Panel (c) shows bias in scorecard predictions for new loan applications in the holdout sample. Panels (d) and (e) depict the impact of bias on scorecard training and evaluation in terms of the ABR. The meta-parameters of the data generation process and the bias correction methods are provided in Appendix 6.9.5.

Panel (d) of Figure 6.7.1 depicts the results of Experiment I that analyzes the impact of sampling bias on the scorecard evaluation. It compares the ABR of f_a on the representative holdout sample H (labeled as oracle performance) and the estimated ABR of f_a using accepts-based evaluation or Bayesian evaluation. Evaluating f_a on a sample from D^a provides an overoptimistic estimate of around .0140. However, when applied to new applications from H, the scorecard demonstrates the ABR of around .2095. This gap illustrates that the accepts-based performance estimate is misleading due to sampling bias. The Bayesian framework provides a more reliable estimate of the ABR, reducing the mean RMSE between the actual and predicted ABR values from .2010 to .0929.

Panel (e) illustrates the results of Experiment II, depicting the effect of sampling bias on the scorecard performance. It compares the ABR of f_a, f_o and f_c. The performance of f_a gradually improves over acceptance loop iterations due to the increasing training sample size. The ABR of f_o remains stable after 100 iterations and is consistently lower than that of f_a. The difference between f_o and f_a captures the loss due to sampling bias. The average

ABR loss across 100 simulation trials is .0598. The model corrected with BASL consistently outperforms f_a starting from the first iteration of the acceptance loop. Note that f_c still suffers from the bias and does not reach the ABR of f_o. However, BASL consistently recovers about 25% of the ABR loss compared to f_a.

Similar performance gains from BASL and the Bayesian framework in Experiment I and II are observed in other evaluation metrics, the AUC, PAUC and BS, and reported in Appendix 6.9.3. Following the recommendations of Demšar [29], we conclude that the observed gains are statistically significant. Friedman's rank sum tests reject the null hypothesis that our propositions perform the same as ignoring rejects in both experiments (p-values are lower than 2.2×10^{-16}). Pairwise Nemenyi post-hoc tests indicate that our propositions significantly outperform the accept-based training and evaluation at a 5% significance level for each of the four metrics.

Sensitivity Analysis

This section examines factors that determine the effectiveness of our propositions to understand when their use is recommended and identify boundary conditions.

Figure 6.7.2 depicts loss due to bias and gains from BASL across different simulations. Panel (a) examines the influence of the magnitude of sampling bias in the training data. The bias magnitude is controlled by the acceptance rate, which varies across financial products and markets. Lower acceptance results in a greater difference between D^a and D. This raises the loss due to sampling bias and the effectiveness of BASL to mitigate this loss. We find BASL to improve scorecards for acceptance rates below 20%, whereby this result originates from assuming a 30% *good* rate in the borrowers' population. Increasing the *good* rate would lead to a slower decay of the loss due to bias and performance gains from BASL. Still, the analysis implies that BASL is more helpful for financial products with low acceptance rates,

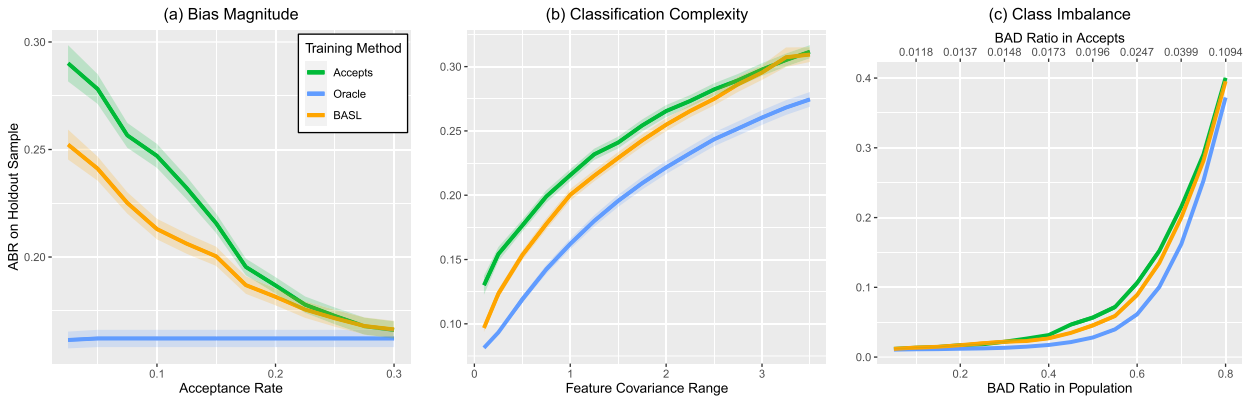

Figure 6.7.2. Sensitivity Analysis: Bias-Aware Self-Learning

The figure illustrates the impact of sampling bias on the model training and benefits from reject inference with BASL as a function of three parameters: (i) bias magnitude controlled by the acceptance rate; (ii) classification task complexity controlled by the feature covariance range; (iii) class imbalance controlled by the *bad* rate in population.

such as, e.g., installment loans for prime customers. The results also agree with Crook and Banasik [26], who find a negative relationship between the acceptance rate and performance gains from reweighting-based bias correction.

Panel (b) studies the classification complexity and depicts the development of scorecard performance as a function of the feature covariance range. The elements of the feature covariance matrix are drawn randomly. A wider range of possible covariance values increases the classification complexity because loan applications of different classes tend to overlap more frequently in the feature space. The loss due to sampling bias is consistently present across the considered complexity range. At the same time, performance gains from BASL are higher in environments with a lower classification complexity and gradually diminish in more complex environments. This is explained by the fact that the pseudo-labels assigned to rejects are more accurate when class separation is easier. The ability to distinguish *good* and *bad* applicants is, therefore, an important factor affecting the potential usefulness of reject inference. In practice, observed default rates can shed light on the complexity of the classification task associated with scoring applications for a financial product.

Panel (c) investigates the impact of class imbalance, which we control by the proportion of *bad* applications in the population. The results suggest that any *bad* rate in the population translates into imbalance among accepts since the data is filtered by a scorecard. The loss due to bias shrinks when class imbalance becomes too strong. This is observed because the ABR metric only focuses on the least risky applicants, which are mostly *good* due to high imbalance. BASL provides the largest gains at moderate imbalance between 2% and 5% among accepts. This imbalance level is sufficiently high so that an accepts-based model is not exposed to enough *bad* risks but is not too severe to prohibit learning from the scarce number of *bad* applications.

Turning attention to the Bayesian evaluation framework, panel (a) of Figure 6.7.3 examines the effect of the acceptance rate on scorecard evaluation. To isolate this effect, we assume a perfect prior when calculating the Bayesian extension of the ABR. Under this assumption, the Bayesian framework estimates scorecard performance accurately across all acceptance rates. Similar to BASL, potential gains from Bayesian evaluation are higher at lower acceptance, as the inconsistency between the performance on accepts versus that on a representative sample becomes stronger.

Calculating the Bayesian extension requires a validation sample of labeled accepts and unlabeled rejects. Panel (b) studies how the quality of this sample affects evaluation. We assess sample quality using the maximum mean discrepancy metric [MMD, 15], which measures the similarity of the feature distribution in the validation set and the unbiased holdout set. The results reinforce accept-based evaluation to underestimate error rates substantially. To predict scorecard performance accurately, the Bayesian framework requires validation data that matches the target distribution in the holdout set. To ensure this, the validation sample should include accepts and rejects from the same time period and match the accept/reject

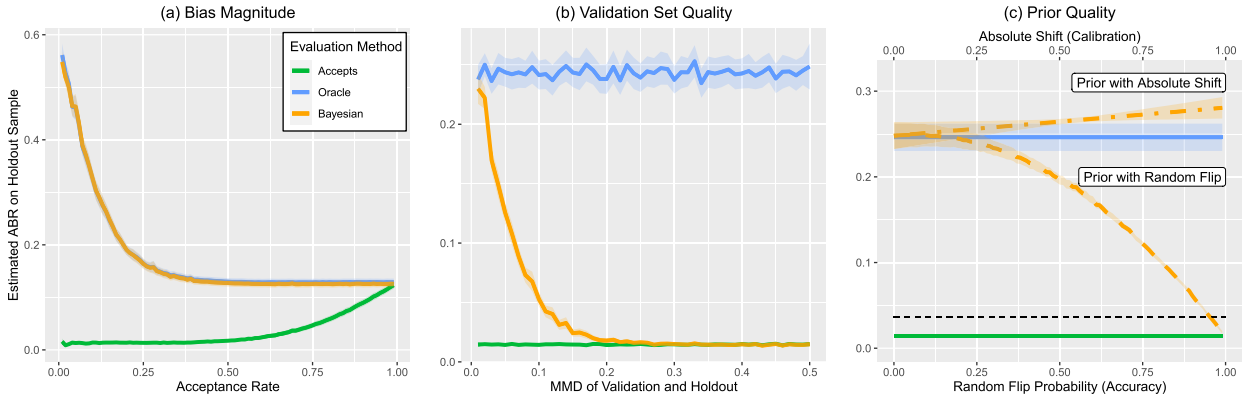

Figure 6.7.3. Sensitivity Analysis: Bayesian Evaluation

The figure illustrates the impact of sampling bias on scorecard evaluation and benefits from the Bayesian framework as a function of three parameters: (i) bias magnitude controlled by the acceptance rate; (ii) validation set quality controlled by the accept/reject rate and measured by the MMD metric; (iii) prior quality controlled by injecting noise in form of a random label flip or level shift. A dashed black line on panel (c) indicates the ABR predicted by the Bayesian framework when using empirical *bad* rate on accepts as a prior for labeling rejects.

ratio in the holdout sample.

Panel (c) focuses on a key ingredient of the Bayesian framework, the class prior for the labels of rejects. Reducing the quality of the estimate of the prior decreases the accuracy of Bayesian evaluation. The quality reduction can stem from diluting the accuracy or the calibration of the prior, which we simulate by injecting random flips or level shifts. The Bayesian framework provides more accurate performance estimates than accepts-based evaluation even when the prior contains a lot of noise. This implies that using imperfect scores (e.g., from a currently used scorecard) as a prior for the labels of rejects still produces a better result than ignoring rejects during evaluation.

Missingness Type

We obtain the previous results on the data generated according to an MAR process, where all applicants' features are revealed to the scorecard. Next, we introduce MNAR by generating data with three explanatory features and using only the first two in the scorecard, creating the omitted variable bias. On each iteration of the acceptance loop, we overwrite a certain percentage of the scorecard-based acceptance decisions by replacing applicants with the lowest values of the hidden feature. This allows us to gradually increase the strength of the MNAR process in the synthetic data. Figure 6.7.4 illustrates the loss due to sampling bias and the performance of our propositions depending on the overwriting percentage. We also implement two established bias correction benchmarks. BASL is compared to the Heckman model, which is developed for MNAR settings, whereas the Bayesian framework is compared to the DR evaluation.

We observe adverse effects of sampling bias on the training and evaluation of scorecards at any level of overwriting. This confirms that bias correction is required regardless of whether

the data exhibits MAR or MNAR. In both extremes with no or full manual overwriting, loss in the ABR exceeds .05 during training. The impact on evaluation is even stronger, with a mean difference of .21 between the ABR estimated on accepts and the actual ABR on holdout applications.

Our propositions consistently outperform accepts-based training and evaluation. Panel (a) of Figure 6.7.4 reveals the poor performance of the Heckman model under MAR. We explain this result by a high correlation between the two variables considered in the model: outcome (i.e., whether the applicant is *bad*) and selection (i.e., whether the applicant was accepted). Previous studies also noted a poor Heckman performance in the presence of strong target-missingness correlation [21]. In our context, the correlation is high when scorecards are accurate. Traditional banks that focus on prime customers often obtain low default rates, which indicates the high accuracy of their scorecards. Increasing the overwriting percentage reduces the target-missingness correlation and strengthens the magnitude of the MNAR process, facilitating a superior performance of the Heckman model. In our simulations, Heckman outperforms BASL once more than 20% of the applications undergo overwriting. Such a large degree of manual intervention is not typical for financial institutions that increasingly rely on the automation of approval decisions. More generally, since the type of missingness is difficult to identify in practice, consistent superiority of BASL over the accepts-based benchmark suggests that institutions adopting BASL will not run the risk of impeding scorecard performance; which they might when using the Heckman model.

Considering model evaluation, we observe the accuracy of the ABR estimates from the Bayesian framework to decrease in the MNAR setting. However, they are much closer to the true scorecard performance than estimates coming from accepts-based evaluation and DR across all setups. DR improves over ignoring rejects but still provides ABR estimates that are, on average, more than .10 off the actual model performance. This large gap can

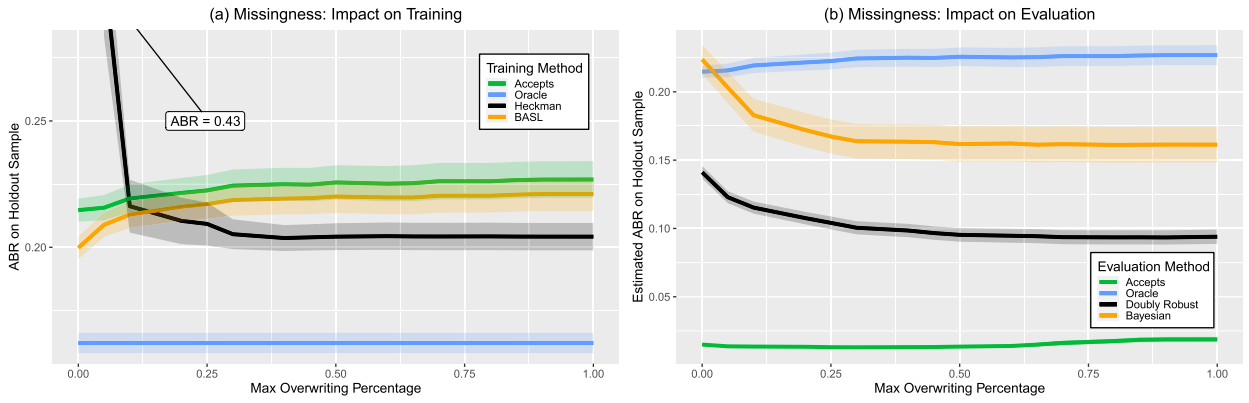

Figure 6.7.4. Sensitivity Analysis: Missingness Type

The figure illustrates the relationship between overwriting and the impact of sampling bias on scorecard training and evaluation, benefits from our propositions (BASL and Bayesian evaluation) and benchmarks (Heckman model and doubly robust evaluation). The vertical axis of panel (a) is truncated at ABR = 0.40 for visualization purposes.

be explained by the adjustments required to apply DR in credit scoring, which we discuss in Appendix 6.9.6.

6.7.2 Real Data

This section presents the results of Experiment I and II on the real-world data set and reports on a business impact analysis, in which we examine the monetary gains from our propositions.

Experiment I: Evaluation with the Bayesian Framework

Table 6.7.1 compares the accuracy of different model evaluation strategies. We compute the RMSE between the scorecard performance estimates produced by each evaluation strategy and the actual scorecard performance on the holdout sample representing the true borrower population.

In line with previous results from synthetic data, we observe relatively high RMSE values when ignoring rejects, which evidences the loss due to sampling bias. Overoptimistic estimates of scorecard performance from the accepts-based evaluation lead to sub-optimal decisions and fail to capture the scorecard's business value. Expecting a certain default rate upon scorecard deployment, a financial institution would face losses and potential liquidity problems when encountering a substantially higher default rate on new loan applications.

Weighted validation improves the accuracy of the scorecard performance estimates for two evaluation metrics. Overall, reweighting performs marginally worse than accepts-based evaluation, achieving an average rank of 2.49 compared to 2.46. At the same time, reweighting outperforms accepts-based evaluation in the metrics that account for asymmetric error costs, the PAUC and the ABR, which are of high importance for decision-makers. DR demonstrates a poor RMSE for the two supported metrics, the BS and ABR. This can be attributed to the high difficulty of reward prediction in a high-dimensional environment and the limitations of DR when applied to credit scoring. Poor performance in the BS and ABR

Table 6.7.1. Scorecard Evaluation: Performance of Bias Correction Methods

Evaluation method	AUC	BS	PAUC	ABR	Rank
Ignore rejects	.1234 (.0309)	.0306 (.0034)	.0983 (.0246)	.0356 (.0603)	2.46
Reweighting	.1277 (.0601)	.0348 (.0054)	.0826 (.3058)	.0315 (.0903)	2.49
Doubly robust	–	.0506 (.0050)	–	.1167 (.0216)	–
Bayesian evaluation	**.0111** (.1158)	**.0073** (.0213)	**.0351** (.0628)	**.0130** (.0331)	**1.06**

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate at 20-40% acceptance, rank = average rank across the four performance measures. Values indicate RMSE between the actual scorecard performance on the holdout sample and performance estimates obtained with a given evaluation method. Variance of the performance estimates $\times 10^{-5}$ in parentheses.

while lacking support for rank-based indicators such as the AUC and PAUC make DR an inappropriate evaluation method for the considered data set.

The Bayesian evaluation framework provides the most accurate estimates of the scorecard performance across all evaluation metrics and achieves an average rank of 1.06. This implies that Bayesian evaluation produces the most reliable predictions of scorecard performance on new loan applications, helping decision-makers to anticipate the accuracy of a scorecard and judge its value ex ante. Appendix 6.9.4 augments Table 6.7.1 with results from statistical testing. Pairwise Nemenyi post-hoc tests indicate that performance estimates obtained with the Bayesian framework are significantly better than those obtained with the benchmark strategies at a 5% level.

Experiment II: Reject Inference with BASL

Table 6.7.2 compares the performance of bias correction methods. We find some methods to perform worse than disregarding rejects. Only three approaches have a lower rank than ignoring rejects. Labeling rejects as *bad* performs worst. Given a historical acceptance rate of $20-40\%$ at Monedo, the underlying assumption of all rejects being *bad* risks is too strong for the used data set. The bias-removing autoencoder also performs poorly. As discussed in Appendix 6.9.6, due to a large number of features and a broad set of meta-parameters, the reconstruction error of the autoencoder remains high even after much tuning. This evidences the difficulty of using an autoencoder in high-dimensional settings.

The Heckman model improves on the previous benchmarks but performs worse than ignoring rejects. Hence, relying on this approach is also ineffective for the data considered here. The poor performance of Heckman can be attributed to two reasons. First, a parametric Heckman model faces difficulties in handling high-dimensional and noisy data. To address this, we consider model variants with reduced feature subsets. The best-performing variant

Table 6.7.2. Scorecard Training: Performance of Bias Correction Methods

Training method	AUC	BS	PAUC	ABR	Rank
Ignore rejects	.7984 (.0010)	.1819 (.0004)	.6919 (.0010)	.2388 (.0019)	3.58
Label all as bad	.6676 (.0014)	.2347 (.0006)	.6384 (.0010)	.3141 (.0022)	8.56
Bias-removing autoencoder	.7304 (.0011)	.2161 (.0004)	.6376 (.0019)	.3061 (.0036)	7.79
Heckman model	.7444 (.0011)	.2124 (.0006)	.6397 (.0010)	.3018 (.0013)	7.53
Bureau score based labels	.7978 (.0009)	.1860 (.0003)	.6783 (.0010)	.2514 (.0021)	4.97
Hard cutoff augmentation	.8033 (.0010)	.1830 (.0006)	.6790 (.0011)	.2458 (.0021)	4.25
Reweighting	.8040 (.0005)	.1840 (.0002)	.6961 (.0009)	.2346 (.0015)	3.45
Parceling	.8038 (.0011)	.1804 (.0004)	.6885 (.0011)	.2396 (.0019)	3.32
Bias-aware self-learning	**.8166** (.0007)	**.1761** (.0003)	**.7075** (.0011)	**.2211** (.0012)	**1.55**

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate at 20-40% acceptance, rank = average rank across the four measures. Standard errors in parentheses.

presented in Table 6.7.2 includes the 65 most important features, which we selected using permutation-based importance. Second, in line with the synthetic data results, the Heckman model performs poorly when the outcome and selection equations are highly correlated. The correlation increases with the accuracy of the previous scorecard. High correlation is also more typical for data exhibiting MAR. Although it is not feasible to reliably estimate the strength of the MNAR process on the real data, the poor performance of Heckman could imply that the missingness type is more geared towards MAR.

Considering established model-based augmentation techniques, HCA improves on ignoring rejects only in the AUC, whereas parceling performs better in two evaluation measures. The better performance of parceling can be explained by introducing randomness on the labeling stage, which helps this approach reduce the error propagation and achieve an overall rank of 3.32.

Reweighting outperforms other benchmarks in the AUC, PAUC and ABR. Despite the good performance in these measures, reweighting has a worse BS than ignoring rejects, indicating a poor calibration ability of the resulting scorecard. This translates to a marginally higher overall rank of reweighting compared to parceling. Appendix 6.9.6 discusses the performance of different reweighting variants in more detail, whereas Table 6.7.2 only includes the best-performing specification.

BASL performs the best in each performance indicator and achieves the lowest average rank of 1.55. Compared to reweighting, the closest competitor in the cost-sensitive metrics, the PAUC and ABR of the scorecard after bias correction with BASL increase by .0114 and .0135, respectively. Gains from BASL are statistically significant: Nemenyi post-hoc tests indicate that BASL significantly outperforms all benchmarks at a 5% level in the AUC, PAUC, and ABR. Appendix 6.9.4 provides auxiliary results from an ablation study, which examines incremental performance gains from different stages of BASL. The largest gains are attributed to the filtering stage.

Business Impact Analysis

To evaluate the business impact of our propositions, we estimate gains in monetary performance. This requires knowledge of key loan parameters. We consider two financial products with different properties: microloans and installment loans. Microloans are small-amount, short-term, often single-payment, high-interest loans. Installment loans have larger amounts, smaller interest, and are repaid over time by regular payments. The approval rates for microloans tend to be higher than those for installment loans.

We draw loan amounts and interest rates from Gaussian distributions with means set to the values observed in the US consumer loan market[12]. We compute i as the total interest and fees divided by the principal A. The loss given default (LGD) indicates the percentage

[2]Source: The Pew Charitable Trusts (2016) Payday Loan Facts, `https://www.pewtrusts.org/-/media/assets/2016/06/payday_loan_facts_and_the_cfpbs_impact.pdf`.

Table 6.7.3. Business Settings

Parameter	Notation	(a) Microloans	(b) Installment loans
Acceptance rate	α	$[.2, .4]$	$[.1, .2]$
Loan principal	A	\$375 (SD = \$100)	\$17,100 (SD = \$1,000)
Total interest rate	i	.1733 (SD = .01)	.1036 (SD = .01)
Loss given default	LGD	$[0, 1]$	$[0, 1]$

The table reports parameters of the business impact analysis. Principals and interest rates are drawn from Gaussians with reported means and standard deviations (in parentheses). The LGD is drawn from [0, 1] with a step of .01.

of A lost in case of default and varies between 0 and 1. Table 6.7.3 provides the parameter values for the two markets.

In the event of default occurring with a probability PD, a financial institution recovers $A \times (1+i) \times (1-\text{LGD})$. If there is no default, the expected revenue is $A \times (1+i)$. For each bias correction method, we approximate the loan-level PD by computing the ABR of this method within the specified acceptance range. We use the modeling pipeline of Section 6.6 to obtain 100 ABR estimates for each bias correction method. Given these 100 estimates and the values from Table 6.7.3, Equation 6.7.2 yields an estimate of the average profit per loan for every bias correction method:

$$\pi = \frac{1}{100} \sum_{j=1}^{100} \left[\text{PD}_j \times A \times (1+i) \times (1-\text{LGD}) + (1-\text{PD}_j) \times A \times (1+i) - A \right] \qquad (6.7.2)$$

We aggregate the average profit per loan over 10,000 trials, drawing A and i from the Gaussian distributions and varying the LGD from 0 to 1. By subtracting the profit of each bias correction method from the profit of a scorecard that ignores rejects, we compute the incremental profit compared to ignoring sampling bias. Finally, we compute the expected margin (i.e., the expected return per dollar issued) by dividing the incremental profit by the average loan amount. It is worth noting that the expected profit assumes that all applications are either *good* or *bad*. In reality, more outcomes are possible: e.g., customers can repay early or consolidate into a different loan.

Figure 6.7.5 illustrates the expected return as a function of the LGD. We focus on the two bias correction methods achieving the lowest ABR: BASL tuned with the Bayesian evaluation framework and reweighting. Ignoring sampling bias impacts the profit of a financial institution. On the microloan market, BASL increases the expected return per dollar issued by up to 2.07 percentage points compared to ignoring rejects and up to 1.58 percentage points compared to the best benchmark. For installment loans, monetary gains are up to 2.70 percentage points compared to ignoring rejects and 2.18 compared to reweighting. Assuming the loan amounts reported in Table 6.7.3, the incremental profit from correcting

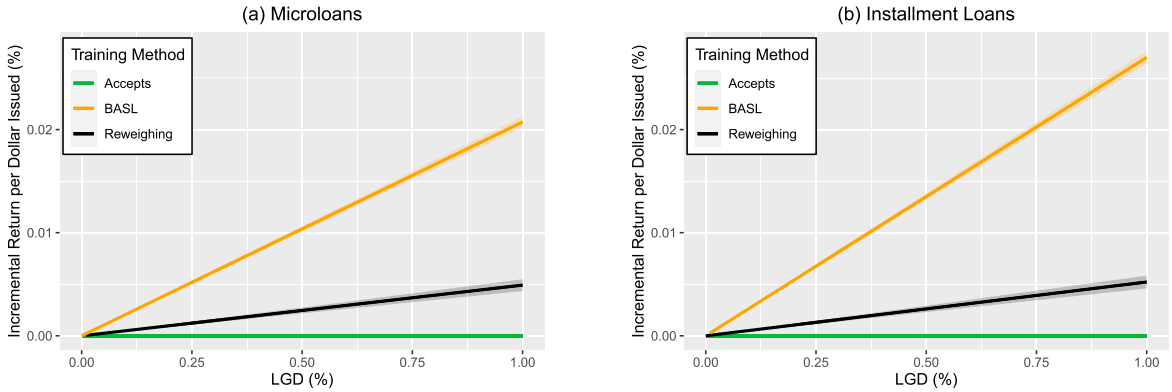

Figure 6.7.5. Monetary Gains

The figure illustrates the expected incremental return per dollar issued of BASL and reweighting relative to a scorecard that ignores sampling bias on two markets: (a) microloans and (b) installment loans.

sampling bias with our propositions is up to $7.78 per loan on the microloan market and up to $461.70 per loan on the market of installment loans.

6.8 Conclusion

Scoring models aid decision-making by predicting the outcomes of choice alternatives. Studying loan approval decisions in retail finance, the paper illustrates the adverse effects of sampling bias on training and evaluation of scoring models and addresses both with two propositions. The Bayesian evaluation framework leverages unbiased unlabeled data for estimating model performance reliably. The BASL framework mitigates the loss due to bias by training scorecards on a debiased sample, which comprises selected rejects with an inferred label. Using real-world lending data including an unbiased sample representative of the true borrower distribution, we confirm the effectiveness of BASL and the Bayesian framework, demonstrate their superiority over other bias correction methods, and find reject inference to increase profit. Results from a simulation study augment those findings by identifying boundary conditions that determine the effectiveness of our propositions.

The study has several implications for credit scoring. Regulatory frameworks such as the Basel Accord require banks to establish internal capital adequacy assessment procedures, which involve the validation of scorecards. We show that widely used evaluation practices are vulnerable to sampling bias and give misleading advice. The Bayesian framework anticipates the loss in accuracy due to bias and facilitates quantifying this effect prior to scorecard deployment. A more realistic and forward-looking performance evaluation helps to improve risk management practices in the industry and benefits the empirical credit scoring literature, which routinely assesses scoring models using biased data of accepted clients [e.g., 57].

The paper also shows that training on biased data decreases the accuracy and profitability of a scorecard. Financial institutions can use BASL in combination with any supervised

learning algorithm to reduce the loss in model performance. Doubt as to whether reject inference is worthwhile prevails in the literature [e.g., 21]. Reporting positive results from an unbiased evaluation sample, the paper speaks to this scepticism. Reject inference is a hard problem. Financial rewards will not be excessive. However, the specific engineering of BASL facilitates consistent and material gains in this study. Improvements of the magnitude observed here in a core business process may well be a deciding factor in highly competitive lending markets.

Exploiting the potential of reject inference and our propositions requires access to unbiased unlabeled data. Meeting this requirement in a credit context is nontrivial. Financial institutions need to store data on rejected applicants, which poses challenging questions related to privacy and consumer protection. Balancing the interests of lenders to gather more data for improving processes such as loan approval and the interests of consumers for protection against privacy infringement is a major challenge in the digital economy. Quantifying the value of a specific type of data in a specific business use case, the paper contributes a humble piece of empirical evidence to this societal debate, which may inform bank governance and regulatory authorities.

The increasing use of scoring models to derive predictions and recommendations from observational data in various fields warrants general concern about sampling bias. The growing literature on off-policy evaluation and learning echoes these concerns and provides approaches for the robust evaluation and learning of policies in a contextual bandit setup. To our best knowledge, corresponding methods have received minimal attention in credit scoring, where the outcomes or rewards associated with a reject decision are never observed. Based on a simulation study and experiments on real-world lending data, we find that BASL and the Bayesian framework outperform selected off-policy benchmarks. These results are specific to our data and experimental design, which reflect the characteristics of a credit scoring context. Hence, they evidence that our propositions deserve a place in data scientists' toolbox and can offer superior decision support in certain scenarios.

Performing sensitivity analysis and examining boundary conditions, the paper offers several criteria to anticipate the loss due to sampling bias in an application setting and the suitability of the proposed remedies. We find that the magnitude of the loss due to bias and the potential recovery from bias correction is higher in environments with low acceptance rates, moderate or high class imbalance and good class separation. Class separability is dependent on the available features and difficult to measure in real life. Class imbalance, on the other hand, is a known modeling challenge encountered in many scoring model applications [e.g., 88]. The last characteristic, termed low approval rate in a credit context, refers to the amount of labeled data that is available for model training and evaluation. Applications in which the acquisition of labels is costly or involves the allocation of a scarce resource display this characteristic.

The characteristics indicate when the loss due to bias is likely substantial. How to

address sampling bias is a different question. One way to mitigate bias involves gathering a representative evaluation and/or training sample by experimentation. Bias correction methods such as BASL and the Bayesian framework should be considered whenever a random allocation of resources is very costly, prohibited, or unethical, which can be the case in medical applications. A criterion to judge the suitability of the Bayesian framework is the observability of decision outcomes (or policy rewards). In credit scoring, the repayment status of a loan is observable only if the application was accepted. Off-policy evaluation methods require adjustments to support this peculiarity, which complicates their use and may harm their effectiveness. Hence, the Bayesian framework is especially suitable in scenarios where certain actions do not reveal rewards. The same consideration applies when measuring scoring model performance using indicators like the AUC, which cannot be calculated on the level of an individual case. For BASL, we observe relative advantages over alternatives like the Heckman model if the process that governs the relationship between outcomes and features and the labeling process (i.e., selection equation) are strongly correlated. Finally, the sensitivity analysis emphasizes the generality of the problem by confirming that sampling bias diminishes the accuracy of scoring model performance estimates independent of whether class labels are missing at random (MAR) or not at random (MNAR). Concerning model training, the status-quo in the credit scoring literature suggests that scorecards lose accuracy under MNAR, whereas posterior probability models like logistic regression do not require debiasing under MAR [8]. Our analysis extends this result by showing that tree-based models, which fail to extrapolate outside of the observed feature ranges, benefit from bias correction even in the MAR setting.

The discussion of environmental characteristics offers guidance when to worry about sampling bias and helps to identify scenarios that could benefit from our propositions. Consider the example of fraud detection, which involves processing a vast amount of transactions or insurance claims and generating model-based fraud scores. Pointing analysts to the most suspicious cases, the scores facilitate efficient utilization of fraud screening resources. Fraud labels are known for an often small subset of previously investigated cases, and the share of fraudulent cases is very low [97]. These characteristics mimic the low acceptance and high imbalance setting in our simulation and suggest that sampling bias might be a serious issue. Given an abundance of unlabeled data and noting that fraud labels (outcomes) remain unknown unless investigating a case, BASL and the Bayesian framework may have the potential to enhance fraud detection practices.

Other interesting examples come from medical settings. Being well aware of the risks of sampling bias, randomized trials and off-policy learning and evaluation are well established in the field. Exemplary use cases of scoring models include treatment allocation decisions. Outcomes relate to recipients' health or well-being and these can be observed independent from taking a specific action (e.g., do not depend on prescribing a treatment). However, scoring models also inform the allocation of transplants to patients on a waiting list by

predicting, e.g., post-transplant survival [17]. Here, an outcome is observed for the low percentage of candidates previously selected for transplant but never observed when rejecting a recipient. This causes class imbalance, creates sampling bias, and mimics the scenario studied in the paper, which proved challenging for off-policy evaluation methods. Gathering representative data through experiments is also not an option. Thus, the validation of the scores is a major problem in transplant allocation, which the Bayesian framework could address.

The examples underline the generality of the sampling bias problem and the vast space of applications for debiasing techniques in management and beyond. They also illustrate how use cases of scoring models in different fields share characteristics of the credit scoring context studied in this paper. Ignoring sampling bias affects the efficiency of resource allocation decisions and may have adverse implications for the people affected by those decisions. The two contributions proposed in the paper constitute a holistic approach to sampling bias mitigation and can be used together or on a standalone basis to raise decision quality and create value.

6.9 Appendix

6.9.1 Prior Work on Bias Correction

This appendix includes the literature tables that provide a comprehensive overview of bias correction methods suggested in different research streams and summarize previous empirical studies on reject inference in credit scoring. A detailed description of the prominent bias correction methods is provided in Section 6.3.

Bias Correction Methods

Table 6.9.1 overviews sampling bias correction methods suggested in the prior work. The bias correction methods suggested in different research streams are grouped into three families depending on the application stage: data preprocessing, model training and model evaluation. Data preprocessing methods encompass representation change techniques that transform input data before modeling to reduce the bias between the source and target distributions. Methods that correct sampling bias in the training stage split into two subgroups: model-based and reweighting techniques. Model-based techniques are embedded in a learning algorithm and account for the bias during model training by adjusting the optimization problem. Reweighting methods rebalance the loss function of a learning algorithm towards more representative data examples and can be applied during model training and model evaluation. Apart from reweighting, methods that correct bias in the evaluation stage include multiple evaluation metrics that approximate the generalization error under sampling bias and metric-agnostic evaluation frameworks.

Table 6.9.1. Sampling Bias Correction Methods

Reference	Method	Type	DP	TR	EV	MA	NT
Blitzer et al. [14]	Structural correspondence learning	RC	✓			✓	
Daumé III [27]	Supervised feature augmentation	RC	✓			✓	
Saenko et al. [83]	Supervised feature transformation	RC	✓			✓	
Gopalan et al. [38]	Sampling geodesic flow	RC	✓			✓	
Gong et al. [37]	Geodesic kernel flow	RC	✓			✓	
Caseiro et al. [20]	Unsupervised feature transformation	RC	✓			✓	
Saptal et al. [84]	Penalized feature selection	RC	✓			✓	
Pan et al. [80]	Transfer component analysis	RC	✓			✓	
Long et al. [68]	Transfer joint matching	RC	✓			✓	
Sun et al. [95]	Correlation alignment	RC	✓			✓	
Wang et al. [100]	Extreme dimension reduction	RC	✓			✓	
Atan et al. [3]	Bias-removing autoencoder	RC	✓			✓	
Heckman [42]	Heckman's model	MB		✓			✓
Meng et al. [77]	Heckman-style bivariate probit	MB		✓			✓
Lin et al. [60]	Modified SVM	MB		✓			✓
Daumé III et al. [28]	Maximum entropy genre adaptation	MB		✓			✓
Yang et al. [105]	Adapt-SVM	MB		✓			✓
Marlin et al. [73]	Multinomial mixture model	MB		✓			✓
Bickel et al. [13]	Kernel logistic regression	MB		✓			✓
Chen et al. [23]	Co-training for domain adaptation	MB, RC	✓	✓		✓	
Duan et al. [30]	Domain adaptation machine	MB		✓			✓
Long et al. [67]	Regularized least squares	MB		✓			✓
Liu et al. [64]	Robust bias-aware classifier	MB		✓			✓
Joachims et al. [48]	Modified ranking SVM	MB		✓			✓
Chen et al. [24]	Robust bias-aware regression	MB		✓			✓
Liu et al. [63]	Modified bias-aware classifier	MB		✓			✓
Kügelgen et al. [56]	Semi-generative model	MB		✓			✓
Rosenbaum et al. [81]	Model-based probabilities	RW		✓	✓	✓	✓
Shimodaira [86]	Distribution density ratios	RW		✓	✓	✓	✓
Zadrozny [106]	Selection probabilities are known	RW		✓	✓	✓	✓
Huang et al. [45]	Kernel mean matching	RW		✓	✓	✓	✓
Cortes et al. [25]	Cluster-based frequencies	RW		✓	✓	✓	✓
Sugiyama et al. [93]	Kullback-Leibler weights	RW		✓	✓	✓	✓
Kanamori et al. [50]	Least-squares importance fitting	RW		✓	✓	✓	✓
Loog [69]	Nearest-neighbor based weights	RW		✓	✓	✓	✓
Gong et al. [36]	Focusing on cases similar to test data	RW		✓	✓	✓	✓
Shimodaira [86]	Modified AIC	EM			✓		✓
Sugiyama et al. [94]	Subspace information criterion	EM			✓		✓
Sugiyama et al. [92]	Generalization error	EM			✓		✓
Sugiyama et al. [91]	Importance-weighted validation	EF			✓	✓	✓
Bruzzone et al. [18]	Circular evaluation strategy	EF			✓	✓	✓
This paper	**BASL and Bayesian evaluation**	**MB, EF**		✓	✓	✓	✓

Method types: RC = representation change, MB = model-based, RW = reweighting, EM = evaluation metric, EF = evaluation framework. Applications stages: DP = preprocessing, TR = training, EV = evaluation. Other abbreviations: MA = model-agnostic method, NT = does not involve input data transformation.

Table 6.9.2. Empirical Studies on Reject Inference in Credit Scoring

Reference	Implemented technique(s)	Training	Evaluation	Representative holdout	Profit gains	No. features
Joanes [49]	Reclassification	DA	–			3
Fogarty [33]	Multiple imputation	DA	–			10
Xia [103]	Outlier detection with isolation forest	DA	–			9
Liu et al. [66]	Ensembling classifiers and clusteres	MB	–			5, 23
Kang et al. [51]	Label spreading with oversampling	DA	–			22
Boyes et al. [16]	Heckman model variant (HM)	MB	–			42
Feelders [32]	Mixture modeling	MB	–			2
Chen et al. [21]	HM	MB	–	✓		24
Banasik et al. [9]	HM	MB	–	✓		30
Wu et al. [102]	HM	MB	–			2
Kim et al. [54]	HM	MB	–	✓		16
Chen et al. [22]	Bayesian model	MB	–		✓	40
Li et al. [59]	Semi-supervised SVM (S3VM)	MB	–			7
Marshall et al. [75]	HM	MB	–			18
Tian et al. [96]	Kernel-free fuzzy SVM	MB	–			7, 14
Xia et al. [104]	CPLE-LightGBM	MB	–			5, 17
Anderson [1]	Bayesian network	MB	–			7, 20
Kim et al. [53]	S3VM with label propagation	MB	–			17
Shen et al. [85]	Unsupervised transfer learning	MB	–			20
Banasik et al. [7]	Banded weights	RW	–	✓		30
Verstraeten et al. [98]	Resampling	RW	–	✓	✓	45
Bücker et al. [19]	Missing data based weights	RW	–			40
Crook et al. [26]	Banded weights, extrapolation	RW, DA	–	✓		30
Banasik et al. [8]	HM with banded weights	MB, RW	–	✓		30
Maldonado et al. [70]	Self-learning, S3VM	MB, DA	–			2, 20, 21
Anderson et al. [2]	HCA, Mixture modeling	DA, MB	–			12
Nguyen [78]	Parceling, HM, Banded weights	DA, MB, RW	–			9
Mancisidor et al. [72]	Bayesian model, self-learning, S3VM	DA, MB	–			7, 58
This paper	**BASL, Bayesian evaluation**	**DA**	**EF**	✓	✓	**2,410**

Abbreviations: DA = data augmentation, MB = model-based, RW = reweighting, EF = evaluation framework. "Representative holdout" indicates whether the study has access to a sample from the borrower's population for evaluation. "Profit gains" indicates whether gains are measured in terms of profit.

In addition to the method type and the application stage, Table 6.9.1 indicates two further characteristics of the bias correction methods: (i) whether the method is model-agnostic and (ii) whether it requires input data transformation. The advantage of model-agnostic methods is their flexibility with respect to the base classifier. Methods that rely on input data transformation require training a scoring model on latent features, which may harm the comprehensibility and explainability of the model.

Applications in Credit Scoring

Table 6.9.2 overviews empirical studies on bias correction in credit scoring. We group the studies by the type of the implemented bias correction technique(s) and distinguish the methods applied in the model training and model evaluation stage. The discussion of the findings in Table 6.9.2 is available in Section 6.3.3.

The empirical studies on reject inference are summarized across multiple dimensions, including: (i) whether the employed data set includes a holdout sample representative of the population, (ii) whether the performance gains are measured in profit and (iii) the number of features in the data set. The first dimension illustrates the potential for an accurate estimation of gains from bias correction, which requires labeled applications rejected by a scorecard. The second dimension shows if improvements from bias correction are measured in terms of the monetary gains. The third dimension indicates the data set dimensionality. The importance of handling the high-dimensional feature spaces is rising in light of a growing market share of FinTechs that operate with large amounts of data from different sources [89] and an increasing reliance of financial institutions on alternative data sources such as applicant's digital footprint, e-mail activity and others [e.g. 47].

6.9.2 Bias-Aware Self-Learning Framework

This appendix consists of two parts: (i) it provides the pseudo-code describing the BASL reject inference framework; (ii) it elaborates on the benefits of using a weak learner such as LR to label rejected applications.

Framework Pseudo-Code

BASL includes four stages: (i) filtering rejects, (ii) labeling rejects, (iii) training the scorecard, (iv) early stopping. Algorithm 4 provides the pseudo-code describing the filtering stage. Algorithm 5 describes the labeling stage. The complete BASL framework is summarized in Algorithm 6 and explained in Section 6.5.

Labeling Rejects with a Weak Learner

This section illustrates the importance of using a weak learner on the labeling stage of the BASL framework. Consider two scoring models that employ different base classifiers: (i)

input : accepts $\mathbf{X}^a$, rejects $\mathbf{X}^r$, meta-parameters $\beta = (\beta_l, \beta_u)$
output: filtered rejects $\mathbf{X}^r$
$g(X)$ = novelty detection algorithm trained over $\mathbf{X}^a$;
$\mathbf{s}^r = g(\mathbf{X}^r)$; `// predict similarity scores`
$\mathbf{X}^r = \{X_i \in \mathbf{X}^r | s_i^r \in [\beta_l, \beta_u]\}$, β_l and β_u are percentiles of $\mathbf{s}^r$; `// filter rejects`
return : $\mathbf{X}^r$

Algorithm 4: Bias-Aware Self-Learning: Filtering Stage

input : labeled accepts $D^a = \{(\mathbf{X}^a, \mathbf{y}^a)\}$, unlabeled rejects $D^r = \{\mathbf{X}^r\}$, meta-parameters ρ, γ, θ
output: selected labeled rejects $D^* = \{(\mathbf{X}^*, \mathbf{y}^*)\}$
$\mathbf{X}^*$ = sample($\mathbf{X}^r, \rho$), ρ is the sampling rate ; `// random sample of rejects`
$f(X)$ = weak learner trained over $\mathbf{D}^a$;
$\mathbf{s}^* = f(\mathbf{X}^*)$; `// score rejects with a weak learner`
derive c_g: $\mathbf{P}(\mathbf{s}^* < c_g) = \gamma$, γ is the percentile threshold ;
derive c_b: $\mathbf{P}(\mathbf{s}^* > c_b) = \gamma\theta$, θ is the imbalance parameter ;
$\mathbf{X}^* = \{X_i^* \in \mathbf{X}^* | s_i^* < c_g \text{ or } s_i^* > c_b\}$; `// select relevant examples`
$\mathbf{y}^* : y_i^* = 0$ if $s_i^* < c_g$ and $y_i^* = 1$ if $s_i^* > c_b$; `// assign labels`
return : $\{(\mathbf{X}^*, \mathbf{y}^*)\}$

Algorithm 5: Bias-Aware Self-Learning: Labeling Stage

input : labeled accepts $D^a = \{(\mathbf{X}^a, \mathbf{y}^a)\}$, unlabeled rejects $D^r = \{\mathbf{X}^r\}$, holdout set $H = \{\mathbf{X}^h\}$; meta-parameters: filtering (β), labeling (τ, γ, θ), stopping criteria (j_{max}, $\mathbf{P}(y^r|\mathbf{X}^r), \epsilon$)
output: corrected scoring model $f_c(X)$
$\mathbf{X}^r$ = filtering($\mathbf{X}^a, \mathbf{X}^r, \beta$) ; `// filtering stage`
$j = 0$; $V = \{\}$; $F = \{\}$; `// initialization`
while $(j \leq j_{max})$ *and* $(\mathbf{X}^r \neq \emptyset)$ *and* $(V_j \geq V_{j-1})$ **do**
  $j = j + 1$
  $D^* = \{(\mathbf{X}^*, \mathbf{y}^*)\}$ = labeling($D^a, D^r, \rho, \gamma, \theta$) ; `// labeling stage`
  $D^a = D^a \cup D^*$; `// append labeled rejects to the labeled sample`
  $\mathbf{X}^r = \mathbf{X}^r - \mathbf{X}^*$; `// remove labeled rejects from the unlabeled sample`
  $F_j = f(X)$ = strong learner trained over augmented D^a ; `// training stage`
  $V_j = \text{BM}(F_j, H, \mathbf{P}(y^r|X^r), \epsilon)$; `// evaluate using a Bayesian metric`
end
return $f_c(X) = F_{\arg\max(V)}$; `// return best-performing strong learner`

Algorithm 6: Bias-Aware Self-Learning Framework

$f_{XGB}(X)$ uses a strong non-parametric learner (XGB); (ii) $f_{LR}(X)$ uses a weak learner (L1-regularized LR). Using a real-world data set described in Section 6.6, we train both models on D^a and use them to score applications in the holdout sample H.

Figure 6.9.1 compares the distribution densities of predictions of f_{XGB} and f_{LR}. LR produces probabilistic predictions that follow a distribution with a high kurtosis and a density peak around .40. In contrast, XGB scores follow a distribution with heavier tails. The range of the scores of f_{XGB} is wider than that of f_{LR}, and the predictions are close to extreme values of 0 or 1 more frequently. This indicates a poor calibration of boosting predictions, which is also confirmed by the previous studies [79].

Within the BASL framework, we only label rejects for which the labeling model has high confidence in the predicted labels, focusing on the examples from the tails of the score distribution. This requires a labeling algorithm that is able to accurately assign labels for examples on the extremes of the score distribution and has a good calibration ability to facilitate setting the appropriate confidence thresholds based on the predicted probabilities of default. As indicated in Figure 6.9.1, a weak learner such as LR is a good fit. A weak learner is also less prone to overfitting the biased training sample, which could increase the risk of error propagation during labeling. Furthermore, as explained in Section 6.5, a regression-type algorithm such as LR is able to extrapolate outside of the feature range observed in the training data, which is not the case for tree-based learners such as XGB. Therefore, we use LR on the labeling stage of BASL.

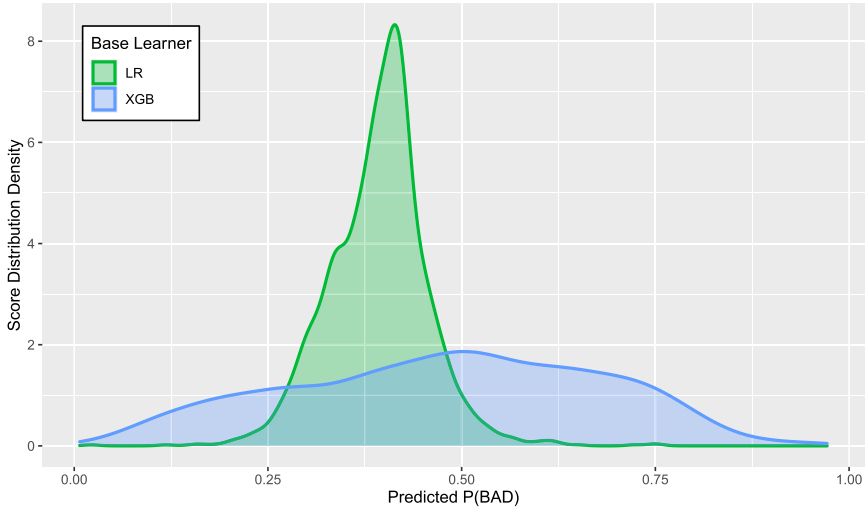

Figure 6.9.1. Prediction Density Comparison

The figure compares distribution densities of the scores predicted by two scoring models using different base classifiers: (i) L1-regularized logistic regression f_{LR} (green); (ii) extreme gradient boosting f_{XGB} (blue).

6.9.3 Extended Results on Synthetic Data

This appendix provides methodological details of the simulation framework introduced in Section 6.4 and additional empirical results that extend the results reported in Section 6.7.1. The simulation study illustrates sampling bias, its adverse effect on the behavior, training and evaluation of scoring models and gains from our two propositions: the Bayesian evaluation framework and BASL. The parameters of the data generation process and the acceptance loop used in the simulation are provided in Appendix 6.9.5.

Simulation Framework

The simulation framework is summarized in Algorithm 8. The framework consists of two stages: the initialization and the acceptance loop. In the initialization stage, we generate synthetic data including two classes of borrowers from a mixture of Gaussian distributions using Algorithm 7. A similar approach to generate synthetic loan applications using Gaussian distributions has been used in the prior work [e.g., 76, 70]. Let our synthetic examples $\mathbf{X}^g = (X_1^g, ..., X_n^g)^\top$ and $\mathbf{X}^b = (X_1^b, ..., X_m^b)^\top$ representing *good* and *bad* loan applications be generated as follows:

$$\begin{cases} \mathbf{X}^g \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^g, \boldsymbol{\Sigma}_c^g) \\ \mathbf{X}^b \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^b) \end{cases} \tag{6.9.3}$$

where δ_c is the weight of the c-th Gaussian function, $\sum_{c=1}^{C} \delta_c = 1$, and $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are the mean vector and the covariance matrix of the c-th Gaussian. The elements of $\boldsymbol{\Sigma}_c^i$ are drawn from a uniform distribution $\mathcal{U}(0, 1)$. We also append two noisy features with the same mean and variance for both classes: $x_\varepsilon \sim \mathcal{N}(0, 1)$.

Suppose the random binary vector $\mathbf{y} = \mathbf{y}^g \cup \mathbf{y}^b$ is a label indicating if an applicant is a *good* ($y = 0$) or *bad* risk ($y = 1$). The difference between the applicant classes is controlled by the parameters of the underlying distributions. Assuming a *bad* rate of b, we generate $n_b = nb$ *bad* examples and $n_g = n(1 - b)$ *good* examples and construct a first batch of the loan applications $D^* = \{(\mathbf{X}^*, \mathbf{y}^*)\}$ with $(\mathbf{X}^*, \mathbf{y}^*) \sim \mathbf{P}_{XY}$. We also generate a holdout set of h examples denoted as $H = \{(\mathbf{X}^h, \mathbf{y}^h)\}$ using the same parameters as for the initial population. H acts as a representative set that does not suffer from sampling bias. We use H for performance evaluation.

The second stage of the framework – the acceptance loop – simulates the dynamic acceptance process, where loan applications arrive in batches over certain periods of time (e.g., every working day). Assume that $D^* = \{\mathbf{X}^*\}$ is the first batch of n applicants a financial institution encounters when entering a new market. Since no repayment data have been collected so far, a company might rely on a simple business rule to filter applications. An example would be to rank applications in D^* by their credit bureau scores denoted as x_v. In our simulation, x_v refers to a feature with the largest difference in mean values between *good* and *bad* applicants and represents a powerful attribute, such as a bureau score, that can be

used to perform a rule-based application ranking. Assuming the target acceptance rate of α, the financial institution grants a loan to αn applicants with the highest bureau scores, forming a set of accepts $D^a = \{X_i \in \mathbf{X}^* | x_{i,v} \geq \tau\}$, and reject $(1-\alpha)n$ remaining applicants, forming a set of rejects $D^r = \{X_i \in \mathbf{X}^* | x_{i,v} \leq \tau\}$, where τ is the $(1-\alpha)$-th percentile of x_v with respect to D^*. Eventually, the repayment status of applicants in D^a is observed, providing the corresponding labels $\mathbf{y}^a$. The labeled set $D^a = \{(\mathbf{X}^a, \mathbf{y}^a)\}$ can now be used to train a scoring model $f_a(X)$ to support the acceptance decisions for the incoming loan applications.

On each iteration of the acceptance loop, $f_a(X)$ is trained over the available set of accepts in D^a. In addition to $f_a(X)$, we also train an oracle model $f_o(X)$ over the union of accepts

input : distribution parameters $\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C$, sample size n, *bad* ratio b
output: labeled set of examples $D = \{(\mathbf{X}, \mathbf{y})\}$

$n_b = bn$; $n_g = n - n_b$; // compute class-specific sample sizes
$\mathbf{X}^g \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^g, \boldsymbol{\Sigma}_c^g)$; // generate n_g good applications
$\mathbf{X}^b \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^b)$; // generate n_b bad applications
$\mathbf{y}^g = \vec{0}$; $\mathbf{y}^b = \vec{1}$; // define applications' labels
$D = \{(\mathbf{X}^g, \mathbf{y}^g) \cup (\mathbf{X}^b, \mathbf{y}^b)\}$; // construct a data set
return : D

Algorithm 7: Synthetic Data Generation

input : distribution parameters $\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C$, sample sizes n, h, *bad* ratio b, acceptance rate α, number of iterations j_{max}, feature indicator v
output: labeled accepts D^a, labeled rejects D^r, labeled holdout set H

$D^* = \{(\mathbf{X}^*, \mathbf{y}^*)\} = \text{generate}(\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C, b, n)$; // generate data using Algorithm C.1
$H = \{(\mathbf{X}^s, \mathbf{y}^s)\} = \text{generate}(\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C, b, h)$; // generate holdout set
$\tau = (1-\alpha)$-th percentile of x_v with respect to D^* ; // simple business rule
$D^a = \{(X_i^*, y_i^*) | x_{i,v} \geq \tau\}$; // accept αn applications
$D^r = \{(X_i^*, y_i^*) | x_{i,v} < \tau\}$; // reject $(1-\alpha)n$ applications
for $j \in \{1, 2, ..., j_{max}\}$ **do**
  $f_a(X) =$ accepts-based model trained over D^a ;
  $f_o(X) =$ oracle model trained over $D^a \cup D^r$;
  $D_j = \{(\mathbf{X}, \mathbf{y})\} = \text{generate}(\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C, b, n)$; // batch of new applications
  $\tau = \alpha$-th percentile of $f_a(D_j)$; // compute acceptance threshold
  $D_j^a = \{(X_i, y_i) | f_a(X_i) \leq \tau\}$; // accept αn applications
  $D_j^r = \{(X_i, y_i) | f_a(X_i) > \tau\}$; // reject $(1-\alpha)n$ applications
  $D^a = \bigcup_{i=1}^{j} D_i^a$; $D^r = \bigcup_{i=1}^{j} D_i^r$; // append accepts and rejects
end
return : D^a, D^r, H

Algorithm 8: Simulation Framework

and rejects $D^a \cup D^r$. The model f_o represents an upper performance bound as it uses representative data that is not available in practice and does not suffer from sampling bias. We use both f_a and f_o to score examples in H and evaluate their performance. Next, we generate a batch of n new applicants $D_j = \{(\mathbf{X}, \mathbf{y})\}$ using the same distribution parameters as for the initial population (i.e., assuming the absence of population drift) and predict the scores of the new applicants in using f_a. Based on the model predictions, we accept αn examples with the lowest predicted scores and reject the remaining $(1 - \alpha)n$ applications. The newly rejected examples are appended to D^r, whereas the newly accepted examples with their labels are appended to D^a. The augmented set of accepts is used to retrain f_a on the next iteration of the acceptance loop.

To illustrate performance gains from BASL, we apply it on each iteration of the acceptance loop. We train a scoring model $f_c(X)$ that has undergone bias correction by applying BASL to the available set of rejects D^r and augmenting the training set D^a with the selected labeled rejects. On each iteration, we train f_c on the augmented data and score examples in H. This allows us to track gains from BASL compared to f_a.

Gains from the Bayesian evaluation framework are demonstrated by comparing the actual performance of f_a on a representative holdout sample H (labeled as oracle performance) and the predicted performance of f_a estimated with different evaluation methods on each iteration of the acceptance loop. First, f_a is evaluated on a validation subset drawn from the available set of accepts D^a. Second, we evaluate f_a on a validation set that consists of the labeled accepts in D^a and pseudo-labeled rejects in D^r using the Bayesian evaluation framework. This allows us to quantify the gap between the actual and predicted performance of the scorecard and measure the recovery of this gap by using the Bayesian framework for performance evaluation.

Experiment I

This section provides the extended results of Experiment I on synthetic data in the MAR setup. Table 6.9.3 compares the performance of accepts-based evaluation and the Bayesian evaluation framework. The table quantifies the difference between the actual scorecard performance on a representative holdout set and the predicted scorecard performance estimated with one of the two evaluation strategies. We measure bias, variance and RMSE of the performance estimates using four evaluation metrics: the AUC, BS, PAUC and ABR. The results are aggregated across 100 simulation trials $\times$ 500 acceptance loop iterations.

According to Table 6.9.3, performance estimates provided by the Bayesian evaluation framework have a lower bias than those obtained within the accepts-based evaluation. Despite accepts-based estimates demonstrating a lower variance in two evaluation metrics, the BS and ABR, RMSE values between the actual and predicted scorecard performance clearly indicate the advantage of using the Bayesian framework for scorecard evaluation. In all considered evaluation metrics, the Bayesian framework is able to provide a better estimate

Table 6.9.3. Experiment I Results on Synthetic Data

Evaluation method	Metric	Bias	Variance	RMSE
Accepts-based evaluation	AUC	.1923	.0461	.2205
Bayesian framework		.0910	.0001	.1000
Accepts-based evaluation	BS	.0748	.0006	.0828
Bayesian framework		.0038	.0009	.0566
Accepts-based evaluation	PAUC	.2683	.0401	.2803
Bayesian framework		.1102	.0002	.1187
Accepts-based evaluation	ABR	.1956	.0004	.2010
Bayesian framework		.0039	.0040	.0929

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance, RMSE = root mean squared error.

of the scorecard performance on unseen cases from the borrowers' population.

Experiment II

Table 6.9.4 presents the extended results of Experiment II on synthetic data in the MAR setup. The table provides the average loss due to sampling bias using five metrics. First, we use four scorecard performance metrics, the AUC, BS, PAUC and ABR, to measure the performance deterioration. The loss due to bias is measured as a difference between the performance of the oracle model trained on the union of accepts and rejects and that of the accepts-based model trained on accepts only. Second, we measure the loss in the MMD metric, which represents the magnitude of sampling bias in the labeled training data. The MMD is calculated between the training data of accepts and the representative holdout sample. The gains from reject inference with BASL are measured as a percentage from the corresponding loss due to sampling bias in each metric. The results are averaged across 100 simulation trials $\times$ 500 acceptance loop iterations.

The results suggest that the loss due to sampling bias is observed in all considered performance metrics. BASL consistently recovers between 22% and 36% of the loss. The largest performance gains are observed in the AUC and the BS, which represent the metrics that disregard error costs and are measured on the full set of credit applicants. The gains in the two cost-sensitive metrics measured on the subset of applications deemed as least risky, the PAUC and the ABR, are smaller but still exceed 22%. This suggests that gains from reject inference are observed through both type I and type II error reduction.

Interestingly, the results in the MMD metric indicate that augmenting the training data of accepts with rejects labeled by BASL improves the MMD by just 3.74%. This implies that the training data still exhibits a strong sampling bias. At the same time, using that data to train a corrected scoring model recovers more than 22% of the loss due to bias and

Table 6.9.4. Experiment II Results on Synthetic Data

Metric	Loss due to sampling bias	Gain from BASL
AUC	.0591	35.72%
BS	.0432	29.29%
PAUC	.0535	22.42%
ABR	.0598	24.82%
MMD	.5737	3.74%

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance, MMD = maximum mean discrepancy.

scorecard performance, respectively. This result emphasizes that it is enough to label only a portion of rejected cases that help to improve the predictive performance and is further supported by the results of the accuracy-bias trade-off analysis provided in Appendix 14. Increasing the number of labeled rejects allows to further improve the MMD, but does not lead to better scorecard performance due to the noise in the assigned labels. The trade-off between introducing too much noise in the labels and gains from having more representative training data is, therefore, a crucial part of BASL.

Bias-Accuracy Trade-Off

This section investigates the trade-off between sampling bias in the data used for scorecard development and scorecard accuracy. Using synthetic data, we compare multiple variants of BASL and reject inference techniques that perform data augmentation (i.e., label rejected applications and append them to the training data). The results demonstrate the importance of limiting the number of labeled rejects to obtain the best performance and illustrate the relationship between performance maximization and bias mitigation.

The analysis is performed on the synthetic data, which we describe in detail in Appendix 6.9.3. After running the acceptance loop, we apply different reject inference techniques to augment the biased training data of accepts and measure the accuracy and bias of each technique. First, we evaluate the performance of each reject inference method using the four performance metrics considered in the paper: the area under the ROC curve (AUC), the partial AUC (PAUC), the Brier score (BS) and the average *bad* rate among accepts (ABR). Second, we evaluate the magnitude of sampling bias in the augmented training data after reject inference. Here, we use the maximum mean discrepancy metric [MMD, 15], which measures the feature distribution similarity between the augmented training data and the holdout sample.

We implement different variants of BASL, varying the meta-parameter values such that a different subset of rejects is selected during the labeling iterations of the framework. This allows us to consider BASL variants that arrive at different points in the accuracy-bias

space. Labeling more rejects facilitates reducing sampling bias, as the distribution mismatch between the training data and the holdout sample diminishes when adding more rejected applications. On the other hand, noise in the pseudo-labels assigned to the appended rejects harms the performance of the resulting scorecard. To study the trade-off between these conflicting dimensions, we construct Pareto frontiers that contain the non-dominated BASL solutions in the bias-accuracy space.

It is important to emphasize that our approach to measuring bias after reject inference is only suitable for data augmentation methods that label rejects and expand the training data. Some reject inference methods (e.g., the Heckman model) do not explicitly label rejects. Therefore, apart from BASL, our experiment includes four data augmentation benchmarks: ignoring rejects, labeling all rejects as *bad* risks, hard cutoff augmentation (HCA) and parceling. In addition to the Pareto frontiers with non-dominated BASL variants, we also depict some dominated BASL solutions with a high MMD to sketch the bias-accuracy relationship when labeling fewer rejects. For that purpose, we split the MMD interval between the non-dominated BASL variant with the highest MMD and ignoring rejects into equal bins and display the best-performing BASL variant within each of the bins. Figure 6.9.2 demonstrates the results.

As depicted in Figure 6.9.2, ignoring rejects leads to the strongest sampling bias in the training data since it only includes accepts, exhibiting MMD of around .60. The data augmentation benchmarks – labeling rejects as *bad*, HCA and parceling – completely eliminate sampling bias and reduce the MMD to around 0. Such a low MMD is achieved by labeling all rejects, which provides training data that represents the borrower population. However, a high reduction in the MMD does not necessarily improve the performance of the corrected scorecard. Except for the AUC, where all benchmarks improve on ignoring rejects, only some of the three data augmentation techniques outperform the scorecard that ignores rejects. This can be explained by the noise in the pseudo-labels of rejects that results from labeling all rejects, including those that are very different from the accepts.

The BASL framework includes multiple steps to restrict the labeling to selected rejects and attend to the distribution similarity between accepts and rejects and the model's confidence in the assigned label. Limiting the number of labeled rejects substantially decreases the gain in MMD. The BASL variants lying on the Pareto frontiers label between 3% and 42% of the rejects after multiple labeling iterations. This allows decreasing the MMD to some value in the range between .40 and .15, indicating that the training data still exhibits sampling bias. We obtain the best performance from scorecards that make use of only a small part of the labeled rejects (around 3% for the BS and 9% for the other evaluation metrics). The best dominated BASL variants lying outside of the Pareto frontiers further reduce the number of labeled rejects to between 1% and 3%. This harms the performance compared to the best solutions on the frontiers but still allows outperforming the considered data augmentation benchmarks.

Overall, the results indicate that there is a trade-off between reducing sampling bias and improving scorecard performance. This trade-off depends on the quality of the labels assigned to the rejected applications. Naturally, correctly labeling all rejects and appending them to the training data would maximize both the performance and the distribution similarly. In practice, predicted labels of rejects are noisy, which makes labeling too many rejects harm scorecard performance. At the same time, labeling too few rejects does not allow to fully realize the potential of reject inference, as demonstrated by the performance of the dominated BASL scorecards. This bias-accuracy relationship forces a decision-maker to settle for a trade-off. In our paper, we focus on the model accuracy as the ultimate

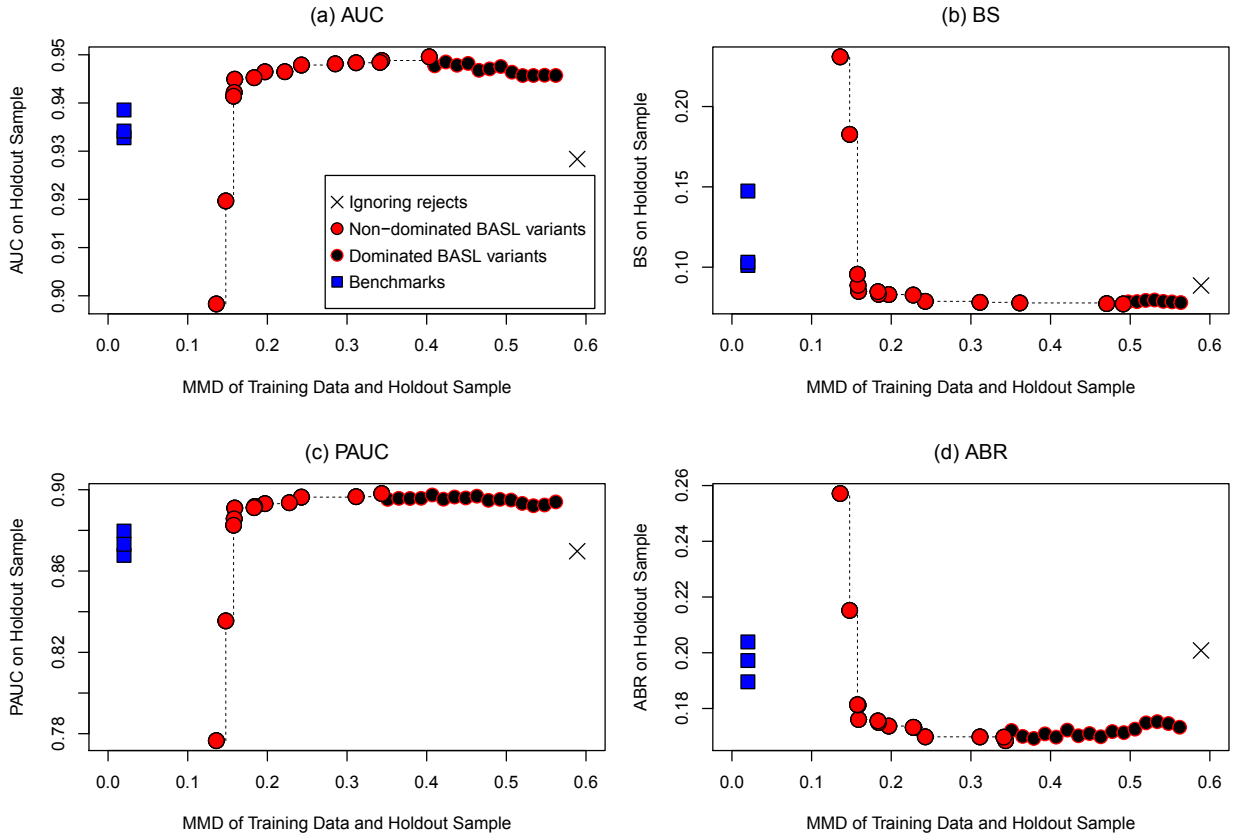

Figure 6.9.2. Bias-Accuracy Trade-Off of Reject Inference Techniques

The figure illustrates the trade-off between the scorecard accuracy and sampling bias when performing reject inference with data augmentation techniques. The vertical axes measure the scorecard performance in one of the four metrics: area under the ROC curve (AUC), Brier score (BS), partial AUC on FNR $\in [0, .2]$ (PAUC), average *bad* rate among accepts at 20-40% acceptance (ABR). The horizontal axes quantify sampling bias in the (augmented) training data by calculating the mean maximum discrepancy (MMD) between the training data and the representative holdout sample. The black cross refers to ignoring rejects. The blue squares depict data augmentation benchmarks that label all rejects: label all as *bad*, hard cutoff augmentation and parceling. The red points depict non-dominated BASL variants with different meta-parameter values. The non-dominated BASL variants label between 3% and 42% of rejects. The black points refer to the best dominated BASL variants that label between 1% and 3% of rejects.

goal of bias correction and tune the meta-parameters of BASL to optimize the scorecard performance.

6.9.4 Extended Results on Real Data

This appendix provides additional empirical results on the real-world credit scoring data set. First, we demonstrate the presence of sampling bias in the sample of accepted applications and illustrate its impact on scorecard behavior, training and evaluation. Second, we provide extended results of Experiment I and II.

Sampling Bias Illustration

Due to the high dimensionality of the real-world data, we focus on a subset of important features to illustrate sampling bias. First, we remove features that exhibit high multicollinearity by applying correlation-based filtering (Spearman or Pearson correlation above .95), which reduces the number of features from 2,410 to 1,549. Second, we train an XGB-based scorecard to produce estimates of the feature permutation importance. We rank features by their importance and use the most important features in the following analysis.

Figure 6.9.3 illustrates sampling bias and its adverse effects on credit scorecards. Panel (a) compares the distribution densities of the most important feature denoted as x_1 on D^a, D^r and H. The results show clear differences in the feature distribution. This observation is supported by the results of three statistical tests over the top-ten most important features at the 1% significance level. Little's mean-based test rejects the null hypothesis that the labels are missing completely at random, indicating the presence of sampling bias [61]. The Probit-based Lagrange Multiplier test reaches the same conclusion and rejects the null hypothesis of the absence of non-random sample selection [74]. Finally, the kernel MMD test indicates the difference in the feature distribution between the accepts and the holdout sample [39]. Overall, the results indicate that the data exhibits sampling bias in previously accepted applications.

Bias in the training data affects the scorecard behavior. Panel (b) compares the coefficients of the top-five most important features of two exemplary scorecards: (i) biased model $f_a(X)$ trained over D^a; (ii) oracle model $f_o(X)$ trained over H. Both scorecards use LR as a base classifier. The results indicate that sampling bias in accepts affects coefficients of the trained scorecard, causing them to diverge from the oracle values. The differences are observed in coefficient sizes (e.g., for x_1) as well as in their signs (e.g., for x_4). The bias in the model parameters translates into a difference in the scores predicted by the scorecards illustrated in panel (c). The accepts-based model f_a provides more optimistic scores compared to f_o.

Panel (d) depicts the impact of sampling bias on the scorecard evaluation in four performance metrics. It compares the actual AUC, BS, PAUC and ABR of f_a on the representative holdout sample H (labeled as oracle performance) and the estimated performance of f_a ob-

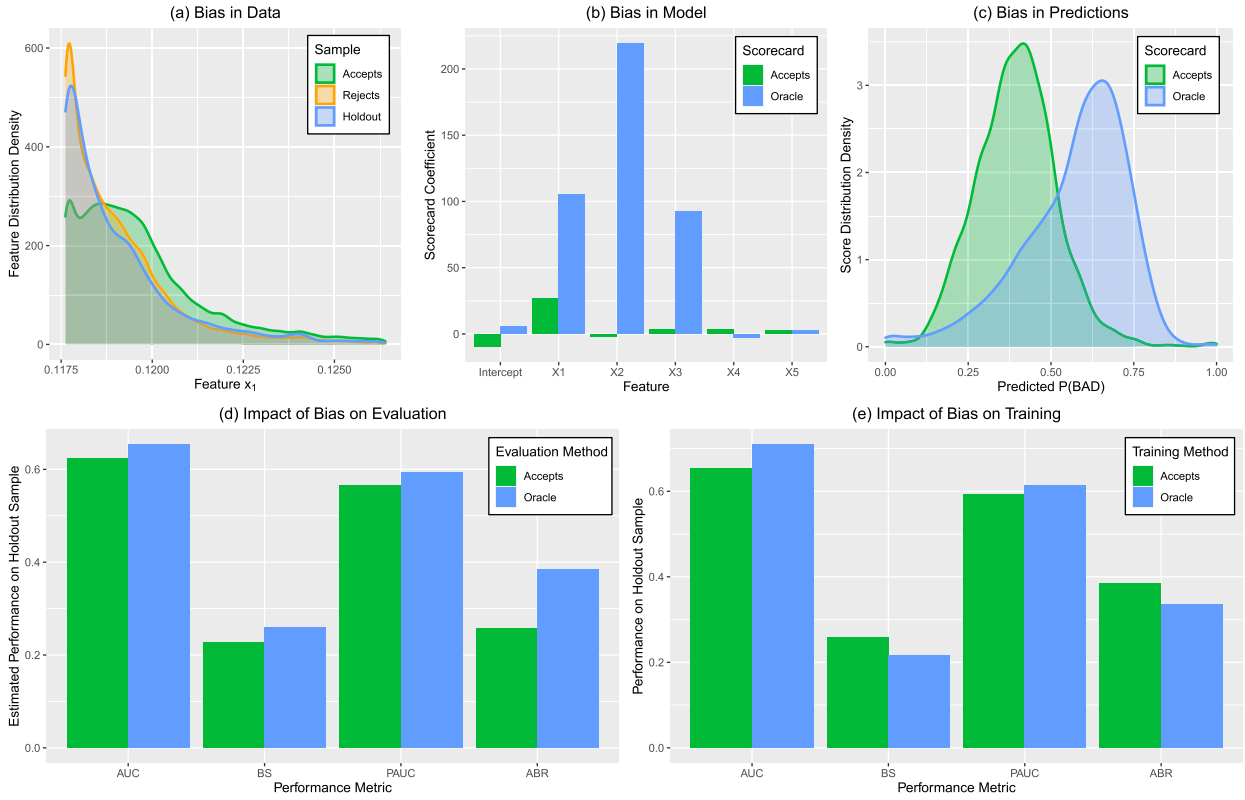

Figure 6.9.3. Sampling Bias Illustration on Real Data

The figure illustrates sampling bias and its adverse effect on the scorecard behavior, training and evaluation on real credit scoring data set. Panel (a) demonstrates the bias in the distribution of one of the features. Panel (b) visualizes the bias in scorecards by comparing their coefficients for the top-five features. Panel (c) shows the bias in scorecard predictions for new loan applications. Panels (d) and (e) depict the impact of bias on scorecard training and evaluation in four performance metrics: the AUC, BS, PAUC and ABR.

tained when evaluating it on a subset of accepts. Performance values are aggregated using leave-one-out validation on the holdout sample. The results suggest that evaluating f_a on accepts provides a misleading estimate in all four performance metrics. This implies that the scorecard evaluation on real data is affected by sampling bias and requires correction. Similarly, panel (e) illustrates the adverse effect of sampling bias on the scorecard training. It compares the AUC, BS, PAUC and ABR of f_a and f_o. In each of the four metrics, the performance of the scorecard deteriorates when training it on a biased set of accepted applications, indicating that the bias also affects model training.

Overall, the results on real data demonstrated in Figure 6.9.3 are in line with the results on synthetic data presented in Figure 6.7.1 in Section 6.7. The sampling bias originates from the fact that the training data of previously accepted applicants is not representative of the borrowers' population. The bias in the data translates to the bias in model parameters and predictions and negatively affects the training and evaluation of scorecards. Bias correction methods could help to improve both of these dimensions.

Experiment I

This section provides the results of the statistical tests performed in Experiment I on real data. To check the statistical significance of the results presented in Table 6.7.1 in Section 6.7, we perform a Friedman's non-parametric rank sum test for performance differences [35]. The null hypothesis of the test is that all evaluation methods have similar performance. The null hypothesis is rejected for all performance measures with p-values below 2.2×10^{-16}. Given that the Friedman test indicates differences in the predictive performance, we proceed with post-hoc tests of pairwise differences between the evaluation methods.

We also use a Nemenyi post-hoc pairwise test, which compares the differences between the average ranks of two methods to the critical difference value determined by the significance level [29]. Figure 6.9.4 depicts the rank differences between the evaluation methods based on the Nemenyi test results. The bold segments connect evaluation techniques for which the rank differences in a given evaluation measure are not statistically significant at a 5% level. The results suggest that the Bayesian evaluation framework outperforms both accepts-based evaluation and importance reweighting.

Experiment II

This section provides the results of the statistical significance tests performed in Experiment II om real data and the ablation study that investigates performance gains from different stages of the BASL framework. Similar to Experiment I, we check the significance of the performance gains presented in Table 6.7.2 in Section 6.7. The null hypothesis of the Friedman test that all bias correction methods have similar performance is rejected for all four performance measures with p-values of each test statistic below 2.2×10^{-16}.

Figure 6.9.5 depicts the rank differences calculated for the pairwise Nemenyi post-hoc

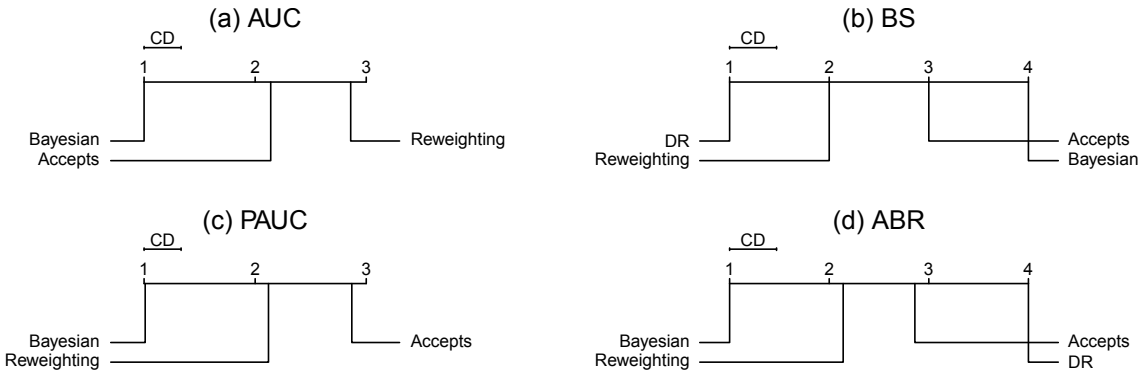

Figure 6.9.4. Experiment I: Critical Difference Plots for Nemenyi Tests

The figure depicts rank differences between evaluation methods. The bold segments connect methods for which the differences are not statistically significant at the 5% level according to the Nemenyi post-hoc pairwise test. Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance.

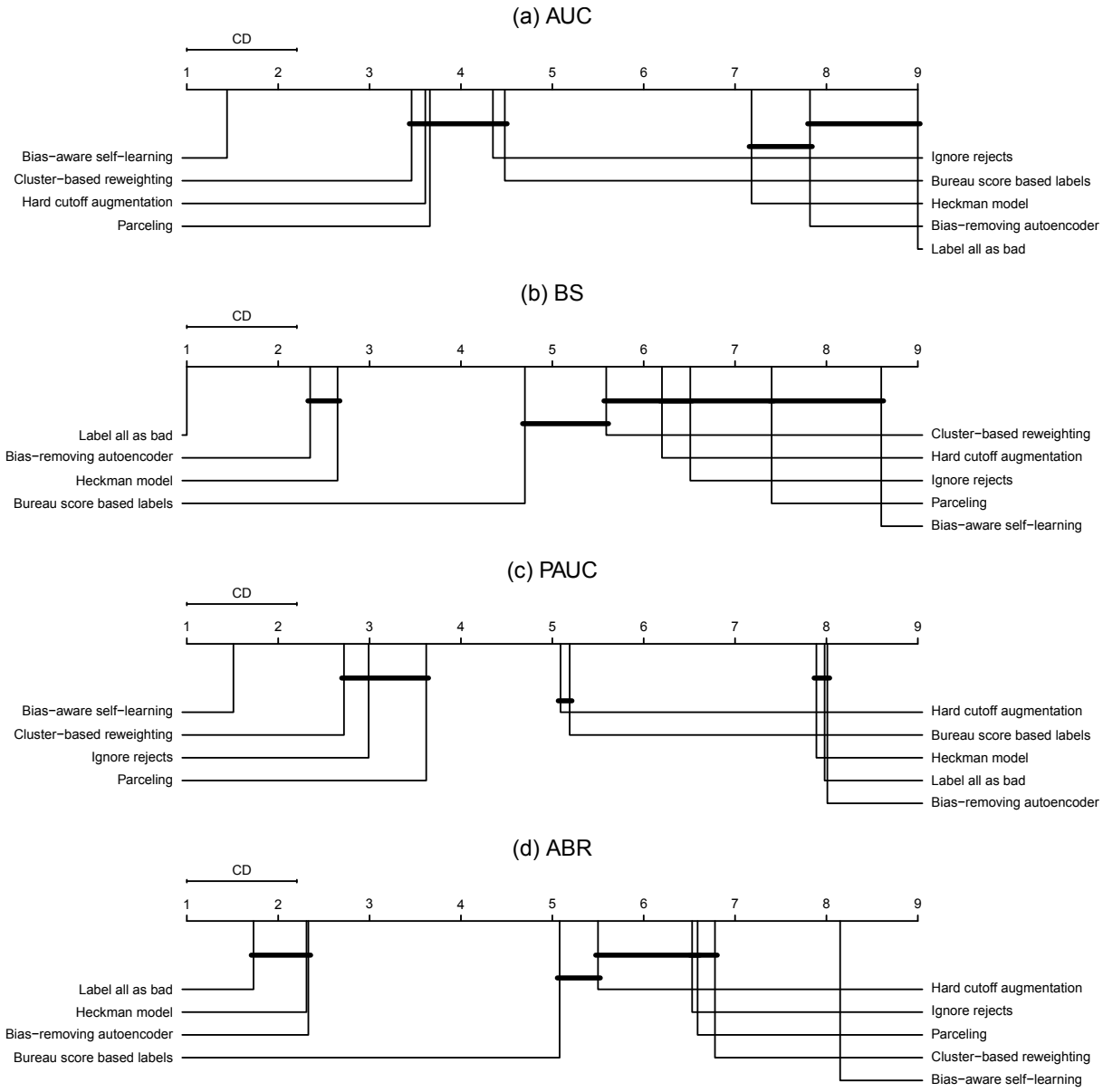

Figure 6.9.5. Experiment II: Critical Difference Plots for Nemenyi Tests

The figure depicts rank differences between evaluation methods. The bold segments connect methods for which the differences are not statistically significant at the 5% level according to the pairwise Nemenyi post-hoc test. Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance.

tests. As indicated in the figure, BASL outperforms all bias correction benchmarks at a 5% significance level in the AUC, PAUC and ABR. BASL also achieves the best BS, but the BS improvement over the closest competitor, parceling, is not significant at 5% level. In many cases, multiple of the other bias correction benchmarks perform similarly to or worse than ignoring rejects. Parceling and cluster-based reweighting are two methods that tend to come closer to BASL than the other benchmarks in terms of the mean ranks.

Table 6.9.5 provides results of the ablation study of BASL. The table displays incremental

Table 6.9.5. Ablation Study: Gains from Different BASL Steps

Framework extension	AUC	BS	PAUC	ABR	Rank
Traditional self-learning	.8059 (.0010)	.1804 (.0004)	.6868 (.0011)	.2387 (.0020)	4.80
Filter rejects using isolation forest	.8054 (.0011)	.1790 (.0004)	.6981 (.0013)	.2312 (.0022)	3.93
Label rejects with a weak learner	.8134 (.0006)	.1774 (.0002)	.6992 (.0009)	.2294 (.0011)	3.60
Introduce the imbalance multiplier	.8133 (.0006)	.1796 (.0002)	.7026 (.0010)	.2238 (.0012)	3.48
Sampling rejects at each iteration	.8157 (.0006)	.1765 (.0002)	.7035 (.0010)	.2254 (.0013)	2.85
Bayesian metric for early stopping	**.8166** (.0007)	**.1761** (.0003)	**.7075** (.0011)	**.2211** (.0012)	**2.34**

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance, rank = average rank across the four evaluation measures.

performance gains from different algorithm steps, starting from traditional self-learning and incorporating the proposed extensions. The extensions make different contributions to the overall performance of BASL.

Overall, incorporating different extensions on top of the traditional self-learning framework improves the the model performance, increasing the PAUC from .6868 to .7075 and the ABR from .2387 to .2211. The largest performance gains in the cost-sensitive metrics are attributed to introducing the filtering stage, which improves the overall rank from 4.80 to 3.93. Gains from implementing the early-stopping mechanism using the Bayesian evaluation framework are observed in all four evaluation metrics, which emphasizes the important role of using a bias-corrected evaluation metric when performing the model selection.

6.9.5 Meta-Parameters of Data Generation and Bias Correction Methods

This appendix provides meta-parameter values of the base classifiers and bias correction methods. We also provide parameters of the data generation process and the acceptance loop used in the simulation study.

Synthetic Data

The data generation process and the acceptance loop have multiple important meta-parameters. Concerning the data generation, we assume the number of mixture components $C = 2$ and set the distribution parameters as follows: $\boldsymbol{\mu}_1^g = (0, 0), \boldsymbol{\mu}_1^b = (2, 1), \boldsymbol{\mu}_2^g = \boldsymbol{\mu}_1^g + \vec{1}$ and $\boldsymbol{\mu}_2^b = \boldsymbol{\mu}_1^b + \vec{1}$. The elements of $\boldsymbol{\Sigma}_c^i$ are drawn from a uniform distribution $\mathcal{U}(0, \sigma_{max})$. We run the acceptance loop for 500 iterations, assuming $n = 100$ and $h = 3{,}000$. In the MAR setup considered in Section 6.7.1, we set $\alpha = .15$, $b = .70$ and $\sigma_{max} = 1$. In the sensitivity analysis presented in Section 6.7.1, we vary α, β and σ_{max} to investigate the boundary conditions affecting the performance of our propositions. In the MNAR setup considered in Section 6.7.1, we assume $\boldsymbol{\mu}_1^g = (0, 0, 0), \boldsymbol{\mu}_1^b = (2, 1, 0.5), \boldsymbol{\mu}_2^g = \boldsymbol{\mu}_1^g + \vec{1}$ and $\boldsymbol{\mu}_2^b = \boldsymbol{\mu}_1^b + \vec{1}$, hiding the feature with the smallest mean difference from the scorecard and using it for overwriting the scorecard

Table 6.9.6. Meta-Parameters of Base Classifiers

Meta-parameter	Candidate values	Selected values (synthetic data)	Selected values (real data)
Maximum number of trees	100, 1,000, 10,000	100	10,000
Early stopping rounds	100, no early stopping	no early stopping	100
Maximum depth	$1, 3, 5$	3	3
Learning rate	$.1, .3$	.1	.1
Bagging ratio	$.8, .9, 1$	.8	.8
Feature ratio	$.8, .9, 1$	.8	.8

predictions. XGB is used as a base classifier for all scoring models. The meta-parameters of XGB on synthetic data are provided in Table 6.9.6.

Concerning the BASL framework, we set the filtering thresholds β to $(.05, 1)$. In the labeling stage, we set $\theta = 2, \rho = .8, \gamma = .01$ and $j_{max} = 3$. We use LR to label the rejected applications and use the Bayesian evaluation framework for early stopping the labeling iterations. To perform the Bayesian evaluation, we set the convergence threshold ϵ to 10^{-6} and specify the number of Monte-Carlo simulations between 10^2 and 10^4. The prior on the labels of rejects denoted as $\mathbf{P}(\mathbf{y}^r|\mathbf{X}^r)$ is obtained by predicting the scores of rejected cases using the accepts-based scoring model and calibrating them using LR.

Real Data

Table 6.9.6 provides the list of the candidate values and the selected values of the meta-parameters of the XGB classifier that is used as a base classifier for all bias correction methods considered in Experiment I and II on the real data. The meta-parameter values are optimized using grid search on a subset of training data.

Table 6.9.7 contains the bias correction methods considered in the empirical comparison, including both training and evaluation strategies. For each bias correction method, we provide a list of their meta-parameters, including a set of candidate values used for the meta-parameter tuning and the values selected after tuning. Two baseline bias correction strategies – ignoring rejects and labeling all rejects as *bad* risks – do not have any meta-parameters and are not included in the table.

6.9.6 Implementation of Benchmarks

This appendix provides further implementation details and additional empirical results for some variants of the bias correction benchmarks not included in the paper. The considered benchmarks include importance reweighting techniques, doubly robust evaluation and the bias-removing autoencoder.

Table 6.9.7. Meta-Parameters of Bias Correction Methods

Bias correction method	Meta-parameter	Candidate values	Selected values
Bias-removing autoencoder	Learning rate	.01	.01
	Number of training epochs	50, 100	100
	Batch size	50, 100	100
	Regularization parameter	$10^{-1}, 10^{-2}, ..., 10^{-5}$	10^{-5}
	Number of hidden layers	3	3
	Bottleneck layer size	$.8k$, k is no. features	$.8k$
Heckman model	Number of features	5, 10, ..., 100	65
	Functional form	probit, logit, linear	linear
Bureau score based labels	Rating of *good* risks	$\{AA\}, \{AA, A\}$	$\{AA, A\}$
	Rating of *bad* risks	$\{D\}, \{C, D\}$	$\{D\}$
Hard cutoff augmentation	Probability threshold	.3, .4, .5	.5
Reweighting	Truncation parameter	.01, .05	.05
	Weight scaling	yes, no	yes
	Density ratio estimation	cluster-based, LSIF, KLIEP	cluster-based
	Number of leaves in DT	100	100
Parceling	Multiplier	1, 2, 3	1
	Number of batches	10	10
Doubly robust evaluation	Truncation parameter	.01, .05	.05
	Weight scaling	yes, no	yes
	Density ratio estimation	cluster-based, LSIF, KLIEP	cluster-based
	Number of leaves in DT	100	100
	Reward prediction model	LR, RF	RF
	Reward prediction threshold	$.1, .2, ..., .9$	.2
Bias-aware self-learning	Filtering thresholds β	(0, 1), (.01, .99), (.01, 1)	(.01, 1)
	Sampling ratio ρ	1, .3	.3
	Labeled percentage γ	.01, .02, .03	.01
	Imbalance parameter θ	1, 2, 3	2
	Max number of iterations j_{max}	5	5
Bayesian evaluation	Max number of trials j_{min}	10^2, 10^3, 10^4	10^2
	Max number of trials j_{max}	10^6	10^6
	Convergence threshold ε	10^{-6}	10^{-6}
	Prior calibration	yes, no	yes

Reweighting

This section focuses on the reweighting techniques considered in this paper. Reweighting tackles sampling bias by estimating importance weights for training examples to rebalance the loss function of the trained algorithm towards examples that are more representative of the population. Given a biased training set D^a and a representative test set $H \subset D$, weights of training examples can be computed as a ratio of two distribution densities: $w(X) = p_H(X)/p_{D^a}(X)$. We focus on the two established families of reweighting techniques: density

ratio estimation and cluster-based methods. In addition, we propose and use an alternative weight estimation method that uses isolation forest to produce the importance weights.

We implement two prominent density ratio estimation methods: Kullback-Leibler Importance Estimation Procedure [KLIEP, 93] and Least Square Importance Fitting [LSIF, 50]. These techniques directly estimate the density ratio without explicit estimation of distribution densities $p_H(X)$ and $p_{D^a}(X)$. KLIEP estimates weights by minimizing the Kullback-Leibler divergence between $p_H(X)$ and $w(X)p_{D^a}(X)$. LSIF formulates a least-squares function fitting problem by modeling weights using a linear model: $w(X) = \sum_{l=1}^{b} \alpha_l \phi_l(X)$, where $\alpha = (\alpha_1, \alpha_2, ..., \alpha_b)$ are parameters to be learned from data, and $\{\phi_l(X)\}_{l=1}^{b}$ are basis functions such that $\phi_l(X) \leq 0$ for all $X \in D^a$ and for $l = 1, 2, ..., b$.

The cluster-based method estimates weights based on the empirical frequencies of data examples [25]. The data spits into n clusters, $C = \{C_i\}_{i=1}^{n}$. The example weights within each cluster are computed as a ratio of test and training examples in that cluster: $w(C_i) = |C_i \cup D^a|/|C_i \cup H|$. Following the suggestion of Cortes et al. [25], we use leaves of a fitted decision tree to form the clusters.

Finally, we estimate importance weights using isolation forest, which is a novelty detection algorithm that estimates the normality of each observation by computing the number of splits required to isolate it from the rest of the data [65]. We fit isolation forest $g(X)$ on the attributes of cases $\mathbf{X}^h$ in the representative sample H. Next, we use g to predict similarity scores for the training examples in D^a: $\mathbf{s}^a = g(\mathbf{X}^a \subset D^a)$. The predicted similarity scores can be used to judge the likelihood that a certain example comes from the population distribution. The obtained score vector $\mathbf{s}^a$ is then used as importance weights.

In the domain adaptation literature, importance weights are normally computed based on the comparison between a biased training sample and a representative target test sample. In the credit scoring context, we observe two samples: a labeled set of accepted clients D^a and an unlabeled set of rejected clients D^r. Both accepts and rejects are biased with respect to the general population of loan applicants D. As indicated in Section 6.6, this paper has access to a representative holdout sample consisting of loans that were granted to a random set of applicants without scoring. This sample can be used as a target sample for estimating the importance weights. However, a representative holdout sample is normally not available as it is very costly to obtain. In this case, a representative validation sample can be constructed by combining the applications that were accepted and rejected by a scoring model during the same time interval.

Before computing the importance weights, we drop features that have Spearman's or Pearson's pairwise correlation higher than .95 to reduce dimensionality and lower the potential noise in weight estimates. The resulting data set contains 1,549 features. Next, we perform 4-fold stratified cross-validation on the data of accepted applicants D^a, following the procedure described in Experiment II in Section 6.6. Within the cross-validation framework, we iteratively estimate the importance weights of the training examples among accepts and

rejected examples using one of the considered reweighting techniques.

For each reweighting method, we estimate importance weights in multiple distinct ways. For the density ratio estimation methods KLIEP and LSIF, we estimate density ratios on two sets of samples: (i) comparing the data of accepted applicants and a holdout sample; (ii) comparing the data of accepted applicants and a validation sample constructed from both accepts and rejects. The estimated ratios serve as training weights.

For the cluster-based method, we train two decision trees that split the data into clusters: (i) the first variant is trained over the accepted applicants; (ii) the second variant is trained over the holdout sample. Both decision trees are limited to 100 leaves to ensure that we have enough observations in each cluster. We assign each training example to a cluster in accordance with the leave of the decision tree in which this example falls. Next, we use each of the leaves to compute cluster-specific example weights in two ways: (i) as a ratio between the number of accepted examples and holdout examples in the cluster; (ii) as a ratio between the number of accepted examples and validation examples in the cluster. This gives us four cluster-based reweighting methods employing different clustering models and different weight estimation techniques.

Similarly, we use two variants of the isolation forest: (i) trained over the holdout sample; (ii) trained over the validation sample consisting of both accepts and rejects. Next, we use the trained models to produce similarity scores for the accepted examples. The similarity scores are then used as importance weights.

Before using the estimated importance weights for sampling bias correction, we truncate weights to reduce their variance [34]. The weights are truncated to the interval $[\alpha, \frac{1}{\alpha}]$, where α is tuned using grid search. We also normalize the obtained importance weights using min-max scaling.

The resulting weights are used for both scorecard training and evaluation. First, we use the importance weights for scorecard evaluation within Experiment I. The examples in the validation subset that contains accepts and rejects are used to calculate one of the four performance metrics used in the paper: the AUC, BS, PAUC and ABR. The BS and ABR are reweighted by multiplying the corresponding application-level errors by the importance weights. The weighted AUC is calculated using a technique suggested by [52, 44]. Second, within Experiment II, we use the importance weights for the scorecard development. The importance weights of accepts act as training weights when fitting the XGB-based scorecard on the data from the training folds. The meta-parameters of the base classifiers are provided in Table 6.9.6.

Table 6.9.8 provides the extended results of Experiment II in terms of the predictive performance of a corrected scorecard on the holdout sample. The results suggest that only some of the reweighting techniques improve the scorecard performance compared to a model with no weights. Overall, one of the cluster-based methods outperforms the benchmarks from the density ratio estimation techniques and isolation forest based reweighting. The

Table 6.9.8. Performance of Reweighting Techniques

Approach	Weights	Sample	AUC	BS	PAUC	ABR	Rank
No weights	–	–	.7984 (.0010)	.1819 (.0004)	.6919 (.0010)	.2388 (.0019)	5.21
Density ratio	KLIEP	H	.7930 (.0011)	.1874 (.0005)	.6815 (.0014)	.2543 (.0024)	8.52
	KLIEP	V	.7950 (.0011)	.1879 (.0005)	.6775 (.0013)	.2570 (.0023)	8.94
	LSIF	H	.8033 (.0007)	**.1827** (.0004)	.6916 (.0013)	.2372 (.0021)	4.54
	LSIF	V	.8027 (.0008)	.1837 (.0004)	.6936 (.0012)	.2365 (.0019)	4.58
Cluster-based	A/H	H	.7979 (.0010)	.1861 (.0006)	.6895 (.0014)	.2447 (.0024)	6.68
	A/V	H	.7988 (.0010)	.1863 (.0004)	.6848 (.0013)	.2473 (.0021)	7.20
	A/H	V	.8040 (.0008)	.1840 (.0004)	**.6961** (.0012)	**.2346** (.0022)	**4.22**
	A/V	V	.7955 (.0012)	.1844 (.0003)	.6884 (.0013)	.2407 (.0022)	6.35
Isolation forest	SS	V	**.8045** (.0009)	.1837 (.0004)	.6932 (.0013)	.2381 (.0021)	4.43
	SS	V	.8029 (.0009)	.1845 (.0003)	.6925 (.0013)	.2407 (.0021)	5.33

Weights: A/H = no. accepts divided by no. holdout examples, A/V = no. accepts divided by no. of validation examples, SS = similarity score. Sample: sample used to estimate weights, train isolation forest or clustering algorithm; H = holdout, V = validation. Performance measures: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance rate, rank = the average method rank across the four measures. Standard errors in parentheses.

best performance is achieved when the decision tree that forms the clusters is trained over accepts, whereas the weights are computed as a ratio between the number of accepts and holdout examples in each cluster. Using isolation forest to estimate weights achieves the second-best performance.

The superior performance of the cluster-based reweighting and isolation forest can be explained by the good scalability of tree-based methods in high-dimensional feature spaces. The density ratio estimation methods KLIEP and LSIF produce noisier estimates, which harms the resulting scorecard performance. The cluster-based reweighting demonstrates the best performance when we calculate the importance weights using a time-based validation set constructed of both accepts and rejects. Relying on such a sample is also easier in practice since a representative holdout set is costly to obtain.

Doubly Robust

This section provides additional methodological details on the implementation of the doubly robust off-policy evaluation method [DR, 31]. Due to the differences between the contextual bandit setting considered in the off-policy evaluation literature and the credit scoring setup considered in this paper, using DR for scorecard evaluation requires some adjustments, which we detail below.

In the off-policy evaluation literature, DR is used in a contextual bandit setting. A decision-maker chooses from a set of possible actions and evaluates a policy that determines the assignment of actions. The quality of a policy, or a classifier, is estimated on historical data. In practice, this data is incomplete as every subject has been assigned to exactly one

of the possible actions. The reward associated with that action was observed and is available in the data. The (counterfactual) reward corresponding to other actions cannot be observed. To address this, DR combines estimating importance weights, which account for sampling bias in the historical data, with predicting the policy reward for the missing actions. DR produces unbiased estimates if at least one of the two modeled equations is correct [90].

The off-policy evaluation setting resembles the credit scoring setup to some extent. Rewards in the form of repayment outcomes are observed for accepted applications. The credit scorecard acts as a policy that determines the assignment of actions (i.e., acceptance vs. rejection). However, a substantial difference between the off-policy evaluation setup and credit scoring concerns the availability of information on policy rewards. We can measure the policy reward by the classifier loss [31], which indicates the predictive performance of the scorecard. In the off-policy evaluation setup, a reward from one of the possible actions is available for each subject in the historical data. DR is then used to combine the observed rewards for actions with the observed outcome and predicted rewards for the remaining actions, where the outcomes are missing. In credit scoring, rewards are only observed for applications that have been accepted in the past (i.e., assigned to one specific action). No rewards are observed for applications assigned to other actions (i.e., rejected) as the financial institution never learns the repayment behavior of rejects. This implies that we need to predict the missing rewards for all rejects.

A second limitation of DR in a credit scoring context is associated with the measurement of reward as classifier loss. This measurement implies that the use of DR is feasible only if we can calculate the evaluation measure on the level of an individual loan. One exemplary loan-level measure is the BS, which assesses a scorecard by calculating the squared difference between the predicted score and a binary label. However, DR is unable to support non-loan level performance measures, including rank-based indicators. Rank-based indicators such as the AUC are widely used in the credit scoring literature [e.g. 57] and regulatory frameworks such as the Basel Capital Accord highlight their suitability to judge the discriminatory power of scoring systems [e.g. 10, 46]. Lacking support for corresponding performance measures constrains the applicability of DR for credit scoring.

In this paper, we implement DR on both synthetic and real-world data. The labeled data of accepted applications is partitioned into training and validation subsets. The training data is used for training the scorecard that is evaluated with DR. The validation subset provides loan applications used for the evaluation. As with the Bayesian evaluation framework, we append rejects to the validation subset to obtain a representative evaluation sample. The repayment outcomes in the validation set are only available for accepts.

As detailed above, DR includes two main components: calculating propensity scores and predicting missing rewards. The first step involves the estimation of propensity scores or importance weights. For this purpose, we use the same method as for the reweighting benchmarks. The comparison of multiple reweighting procedures is described in detail in

Appendix 6.9.6. In our experiments, cluster-based weights with weight clipping performs best and is used for the DR estimator. We calculate importance weights for both accepted and rejected applications in the validation subset and store them for the next steps of the DR framework.

The second step involves the calculation of policy rewards, which requires producing a vector of classifier losses for each of the applications in the validation set. To calculate rewards, we score applications in the validation subset with the scorecard evaluated by DR. Next, we compute the rewards for accepts. This procedure depends on the considered evaluation metric. For the BS, the reward is simply the squared difference between the risk score predicted by the scorecard and the actual 0-1 application label. The ABR metric only penalizes the type-II error (i.e., accepting a *bad* applicant). Therefore, for the ABR, we compute the policy reward as a binary variable that equals 1 if the application is predicted to be a *good* but is actually a *bad* risk, and 0 otherwise. Calculating the other two performance measures used in the paper – the AUC and PAUC – is not feasible on the application level, so we only use DR with the BS and the ABR.

Apart from rewards for accepted clients, we also require rewards for rejects. However, since the actual labels of rejects are unknown, we have to predict the rewards for such applications. For this purpose, we train a reward prediction model on the accepted cases from the validation subset and use it to predict rewards for rejects. Reward prediction is performed using a random forest (RF) regressor for the BS metric and using an RF classifier for the ABR. Both models process all applicant features to predict rewards. Since the ABR calculation requires binary rewards, we convert classifier scores into the class predictions using a specified threshold, which we tune to minimize the RMSE of the DR performance estimates.

The final step of the DR framework is calculating the estimate of the scorecard performance based on the computed rewards and importance weights. The actual rewards on accepts and predicted rewards on rejects are multiplied with the weights to correct for the sample selection bias. Next, we aggregate the resulting values across all applications in the validation subset. For the BS, this implies averaging the corrected squared differences over the applications. For the ABR, which has acceptance rate as a meta-parameter, we average the corrected binary error indicators over a certain percentage of applications predicted as least risky.

Bias-Removing Autoencoder

In this section, we take a closer look at the performance of the bias-removing autoencoder. The bias-removing autoencoder tackles sampling bias by finding a function that maps features into a new representational space $\mathbf{Z}$ (i.e., $\Phi : \mathbf{X} \rightarrow \mathbf{Z}$) such that distribution of the labeled training data over $\mathbf{Z}$ is less biased and $\Phi(X)$ retains as much information about X as possible. To analyze the performance of this bias correction method in more detail, we

compare the predictive performance of four different scoring models that use features coming from the data representations constructed by different autoencoder variants.

The first scoring model $f_a(X)$ serves as a baseline. The model f_a is trained over raw features over a biased sample of previously accepted clients. The next three scorecards are trained over latent features extracted from different autoencoders. The autoencoders are trained using different data samples. First, we train a standard deep stacked autoencoder $a_1(X)$ over $\mathbf{X}^a$ to reconstruct the features of accepted clients. We extract latent features from the bottleneck layer of a_1 and use them for training a new scoring model $f_{a_1}(X)$. The scoring model is, therefore, based on the data representation computed on a biased sample of applicants.

Second, we train the autoencoder $a_2(X)$ with the same architecture as a_1 but using a training sample constructed of both accepted and rejected applicants $\mathbf{X}^a \cup \mathbf{X}^r$. The scoring model $f_{a_2}(X)$ is trained over the latent features extracted from the bottleneck layer of a_2. The extracted features account for the patterns observed on both accepts and rejects, which should improve the performance of the scorecard.

Finally, we train the third autoencoder $a_3(X)$ on $\mathbf{X}^a \cup \mathbf{X}^r$. Compared to a_2, a_3 includes an additional regularization term that accounts for the distribution mismatch similar to Atan et al. [3]. The regularization term penalizes the mismatch between the distributions of latent features on accepted examples and examples in a validation sample consisting of accepts and rejects from the same time window. This helps the autoencoder to derive latent features that are distributed similarly on the two data samples. After training the autoencoder, we train a scoring model $f_{a_3}(X)$ on the extracted feature representation.

In all three cases, we use a stacked deep autoencoder architecture with three hidden layers. The number of neurons is set to $.9k$ on the first and the last hidden layer and $.8k$ on the bottleneck layer, where k is the number of features in the input data. To facilitate convergence, we preprocess the data before training the autoencoder. First, we drop features that have Spearman or Pearson pairwise correlation higher than .95, reducing the number of features to 1,549. Second, we remove outliers by truncating all features at .01 and .99 distribution percentiles. Third, we normalize feature values to lie within $[0, 1]$ interval. Other meta-parameters of the autoencoder are tuned using grid search; the list of candidate values is given in Table 6.9.7. All scoring models use XGB-based classifier as a base model; the meta-parameters are provided in Table 6.9.6.

As a mismatch penalty, we use the Maximum Mean Discrepancy [MMD, 15], which measures a distance between distribution means in a kernel space. The MMD is commonly used as a distribution mismatch measure in the domain adaptation literature [e.g. 80, 68]. The MMD is measured between the latent features of accepts and latent features of the validation sample. The validation sample refers to a time-based representative sample that contains both accepted and rejected clients. The autoencoders only use the features, ignoring the actual labels. Table 6.9.9 reports the results.

Table 6.9.9. Performance of Bias-Removing Autoencoder

Features	Sample	MMD	AUC	BS	PAUC	ABR	Rank
Raw	A	-	**.7984** (.0010)	**.1819** (.0004)	**.6919** (.0010)	**.2388** (.0019)	**1.01**
Latent	A	-	.7006 (.0034)	.2212 (.0008)	.6066 (.0026)	.3385 (.0023)	3.37
Latent	A + R	-	.7177 (.0020)	.2187 (.0004)	.6170 (.0013)	.3328 (.0024)	3.14
Latent	A + R	+	.7304 (.0011)	.2161 (.0004)	.6376 (.0019)	.3061 (.0036)	2.48

Features: features used to train a scorecard (either raw features or latent features extracted from the bottleneck layer of the autoencoder). Sample: training sample of the autoencoder; A = accepts, R = rejects. MMD: whether the MMD penalty is included in the autoencoder loss function. Performance measures: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance rate, rank = the average strategy rank across the four performance measures. Standard errors n in parentheses.

First, we compare latent features extracted from a_1 trained on accepts and latent features from a_2 trained on both client types. The results suggest that the latter set of features leads to a better predictive performance of the eventual scoring model. Furthermore, including the MMD penalty in the autoencoder loss function allows us to extract features that further improve the scorecard's performance. From this comparison, we can conclude that using data of rejected applications and penalizing the distribution discrepancies helps to find a feature representation that suffers less from sampling bias, which has a positive impact on the performance.

At the same time, comparing the performance of the scoring model f_a trained over the original features of accepts to the scoring model f_{a_1} trained over the latent features of the accepts-based autoencoder, we observe a sharp performance drop in all four evaluation measures. This indicates that the predictive power of the latent features constructed by the autoencoder a_1 is too low compared to that of the original features. The observed information loss is too large to be offset by the performance improvement from using rejects and adding a distribution mismatch regularizer. This can be explained by a high dimensionality of the feature space, which complicates the reconstruction task.

Bibliography

[1] Anderson, B. (2019). Using Bayesian networks to perform reject inference. *Expert Systems with Applications, 137*, 349–356.

[2] Anderson, B., Hardin, J.M. (2013). Modified logistic regression using the EM algorithm for reject inference. *International Journal of Data Analysis Techniques and Strategies, 5*(4), 359–373.

[3] Atan, O., Jordon, J., van der Schaar, M. (2018). Deep-treat: Learning optimal per-

sonalized treatments from observational data using neural networks. *Proc. 32nd AAAI Conference on Artificial Intelligence*.

[4] Athey, S., Wager, S. (2021). Policy learning with observational data. *Econometrica, 89*(1), 133–161.

[5] Baesens, B., Setiono, R., Mues, C., Vanthienen, J. (2003). Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science, 49*(3), 312–329.

[6] Ban, G.Y., Rudin, C. (2019). The big data newsvendor: Practical insights from machine learning. *Operations Research, 67*(1), 90–108.

[7] Banasik, J., Crook, J. (2005). Credit scoring, augmentation and lean models. *Journal of the Operational Research Society, 56*(9), 1072–1081.

[8] Banasik, J., Crook, J. (2007). Reject inference, augmentation, and sample selection. *European Journal of Operational Research, 183*(3), 1582–1594.

[9] Banasik, J., Crook, J., Thomas, L. (2003). Sample selection bias in credit scoring models. *Journal of the Operational Research Society, 54*(8), 822–832.

[10] Basel Committee on Banking Supervision (2005). Studies on the validation of internal rating systems. *BIS Working Paper Series* 14.

[11] Bhat, G., Ryan, S.G., Vyas, D. (2019). The implications of credit risk modeling for banks' loan loss provisions and loan-origination procyclicality. *Management Science, 65*(5), 2116–2141.

[12] Biatat, V.A.D., Crook, J., Calabrese, R., Hamid, M. (2021). Enhancing credit scoring with alternative data. *Expert Systems with Applications, 163*, 113766.

[13] Bickel, S., Brückner, M., Scheffer, T. (2009). Discriminative learning under covariate shift. *Journal of Machine Learning Research, 10*(9).

[14] Blitzer, J., McDonald, R., Pereira, F. (2006). Domain adaptation with structural correspondence learning. *Proc. 2006 Conference on Empirical Methods in Natural Language Processing*, 120–128.

[15] Borgwardt, K.M., Gretton, A., Rasch, M.J., Kriegel, H.P., Schölkopf, B., Smola, A.J. (2006) Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics, 22*(14), e49–e57.

[16] Boyes, W.J., Hoffman, D.L., Low, S.A. (1989). An econometric analysis of the bank credit scoring problem. *Journal of Econometrics, 40*(1), 3–14.

[17] Briceño, J., Cruz-Ramírez, M., Prieto, M., Navasa, M., De Urbina, J.O., Orti, R., Gómez-Bravo, M.Á., Otero, A., Varo, E., Tomé, S., et al. (2014). Use of artificial intelligence as an innovative donor-recipient matching model for liver transplantation: results from a multicenter spanish study. *Journal of Hepatology, 61*(5), 1020–1028.

[18] Bruzzone, L., Marconcini, M. (2010). Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(5), 770–787.

[19] Bücker, M., van Kampen, M., Krämer, W. (2012). Reject inference in consumer credit scoring with nonignorable missing data. *Journal of Banking & Finance, 37*(3), 1040–1045.

[20] Caseiro, R., Henriques, J.F., Martins, P., Batista, J. (2015). Beyond the shortest path: Unsupervised domain adaptation by sampling subspaces along the spline flow. *Proc. 28th IEEE Conference on Computer Vision and Pattern Recognition*, 3846–3854.

[21] Chen, G.G., Astebro, T. (2001). The economic value of reject inference in credit scoring. *Proc. 7th Credit Scoring and Credit Control Conference*, 309–321.

[22] Chen, G.G., Åstebro, T. (2012). Bound and Collapse Bayesian reject inference for credit scoring. *Journal of the Operational Research Society* 63(10), 1374–1387.

[23] Chen, M., Weinberger, K.Q., Blitzer, J. (2011). Co-training for domain adaptation. *Advances in Neural Information Processing Systems, 24*, 2456–2464.

[24] Chen, X., Monfort, M., Liu, A., Ziebart, B.D. (2016). Robust covariate shift regression. *Artificial Intelligence and Statistics*, 1270–1279.

[25] Cortes, C., Mohri, M., Riley, M., Rostamizadeh, A. (2008). Sample selection bias correction theory. *Proc. 19th International Conference on Algorithmic Learning Theory*, 38–53.

[26] Crook, J., Banasik, J. (2004). Does reject inference really improve the performance of application scoring models? *Journal of Banking & Finance* 28(4), 857–874.

[27] Daumé III, H (2009). Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

[28] Daumé III, H., Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research, 26*, 101–126.

[29] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

[30] Duan, L., Xu, D., Tsang, I.W.H. (2012). Domain adaptation from multiple sources: A domain-dependent regularization approach. *IEEE Transactions on Neural Networks and Learning Systems, 23*(3), 504–518.

[31] Dudík, M., Erhan, D., Langford, J., Li, L. (2014). Doubly robust policy evaluation and optimization. *Statistical Science, 29*(4), 485–511.

[32] Feelders, A.J. (2000). Credit scoring and reject inference with mixture models. *Intelligent Systems in Accounting, Finance and Management Decision* 9(1), 1–8.

[33] Fogarty, D.J. (2006). Multiple imputation as a missing data approach to reject inference on consumer credit scoring. *Interstat, 41*, 1–41.

[34] Freedman, D.A., Berk, R.A. (2008). Weighting regressions by propensity scores. *Evaluation Review, 32*(4), 392–409.

[35] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*(200), 675–701.

[36] Gong, B., Grauman, K., Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. *Proc. 30th International Conference on Machine Learning*, 222–230.

[37] Gong, B., Shi, Y., Sha, F., Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. *Proc. 25th IEEE Conference on Computer Vision and Pattern Recognition*, 2066–2073.

[38] Gopalan, R., Li, R., Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. *Proc. 13th International Conference on Computer Vision*, 999–1006.

[39] Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research* 13(1), 723–773.

[40] Gu, S., Kelly, B., Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies, 33*(5), 2223–2273.

[41] Gunnarsson, B.R., Vanden Broucke, S., Baesens, B., Óskarsdóttir M., Lemahieu W. (2021). Deep learning for credit scoring: Do or don't? *European Journal of Operational Research, 295*(1), 292–305.

[42] Heckman, J.J. (1979). Sample selection bias as a specification error. *Econometrica, 47*(1), 153–161.

[43] Hilscher, J., Wilson, M. (2016). Credit ratings and credit risk: Is one measure enough? *Management Science, 63*(10), 3414–3437.

[44] Hocking, T.D. (2020). *WeightedROC: Fast, Weighted ROC Curves. R package version 2020.1.31.* URL `https://CRAN.R-project.org/package=WeightedROC`. Accessed 2021-12-01.

[45] Huang J., Gretton, A., Borgwardt, K., Schölkopf, B., Smola, A. (2006). Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems, 19*, 601–608.

[46] Irwin, R.J., Irwin, T.C. (2012). Appraising credit ratings: Does the CAP fit better than the ROC? *IMF Working Paper* 12/122.

[47] Jagtiani, J., Lemieux, C. (2019). The roles of alternative data and machine learning in fintech lending: Evidence from the LendingClub consumer platform. *Financial Management, 48*(4), 1009–1029.

[48] Joachims, T., Swaminathan, A., Schnabel T. (2017). Unbiased learning-to-rank with biased feedback. *Proc. 10th ACM International Conference on Web Search and Data Mining*, 781–789.

[49] Joanes, D.N. (1993). Reject inference applied to logistic regression for credit scoring. *IMA Journal of Management Mathematics, 5*(1), 35–43.

[50] Kanamori, T., Hido, S., Sugiyama, M. (2009). A least-squares approach to direct importance estimation. *Journal of Machine Learning Research* 10(Jul), 1391–1445.

[51] Kang, Y., Jia, N., Cui, R., Deng, J. (2021). A graph-based semi-supervised reject inference framework considering imbalanced data distribution for consumer credit scoring. *Applied Soft Computing, 105*, 107259.

[52] Keilwagen, J., Grosse, I., Grau, J. (2014). Area under precision-recall curves for weighted and unweighted data. *PloS one, 9*(3), e92209.

[53] Kim, A., Cho, S.B. (2019). An ensemble semi-supervised learning method for predicting defaults in social lending. *Engineering Applications of Artificial Intelligence, 81*, 193–199.

[54] Kim, Y., Sohn, S.Y. (2007). Technology scoring model considering rejected applicants and effect of reject inference. *Journal of the Operational Research Society, 58*(10), 1341–1347.

[55] Kozodoi, N., Katsas, P., Lessmann, S., Moreira-Matias, L., Papakonstantinou, K. (2019). Shallow self-learning for reject inference in credit scoring. *Proc. European Conference on Machine learning and Knowledge Discovery in Databases*, 516–532.

[56] Kügelgen, J., Mey, A., Loog, M. (2019). Semi-generative modelling: Covariate-shift adaptation with cause and effect features. *Proc. 22nd International Conference on Artificial Intelligence and Statistics*, 1361–1369.

[57] Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

[58] Levatić, J., Ceci, M., Kocev, D., Džeroski, S. (2017). Self-training for multi-target regression with tree ensembles. *Knowledge-Based Systems* 123, 41–60.

[59] Li, Z., Tian, Y., Li, K., Zhou, F., Yang, W. (2017). Reject inference in credit scoring using semi-supervised support vector machines. *Expert Systems with Applications, 74*, 105–114.

[60] Lin, Y., Lee, Y., Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning, 46*(1-3), 191–202.

[61] Little, R.J. (1988). A test of missing completely at random for multivariate data with missing values. *Journal of the American Statistical Association* 83(404), 1198–1202.

[62] Little, R.J., Rubin, D.B. (2019). *Statistical analysis with missing data*. John Wiley & Sons.

[63] Liu, A., Fathony, R., Ziebart, B.D. (2017). Kernel robust bias-aware prediction under covariate shift. *arXiv preprint arXiv:1712.10050*.

[64] Liu, A., Ziebart, B. (2014). Robust classification under sample selection bias. *Advances in neural information processing systems, 27*, 37–45.

[65] Liu, F.T., Ting, K.M., Zhou, Z.H. (2008). Isolation Forest. *Proc. 8th IEEE International Conference on Data Mining*, 413–422.

[66] Liu, Y., Li, X., Zhang, Z. (2020). A new approach in reject inference of using ensemble learning based on global semi-supervised framework. *Future Generation Computer Systems, 109*, 382–391.

[67] Long, M., Wang, J., Ding, G., Pan, S.J., Yu, P.S. (2014). Adaptation regularization: A general framework for transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 26*(5), 1076–1089.

[68] Long, M., Wang, J., Ding, G., Sun, J., Yu, P.S. (2014). Transfer joint matching for unsupervised domain adaptation. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1410–1417.

[69] Loog, M. (2012). Nearest neighbor-based importance weighting. *Proc. 22nd IEEE International Workshop on Machine Learning for Signal Processing*, 1–6.

[70] Maldonado, S., Paredes, G. (2010). A semi-supervised approach for reject inference in credit scoring using SVMs. *Proc. 10th Industrial Conference on Data Mining*, 558–571.

[71] Malistov, A., Trushin, A. (2019). Gradient boosted trees with extrapolation. *Proc. 18th IEEE International Conference on Machine Learning and Applications*, 783–789.

[72] Mancisidor, R.A., Kampffmeyer, M., Aas, K., Jenssen, R. (2020). Deep generative models for reject inference in credit scoring. *Knowledge-Based Systems* 105758.

[73] Marlin, B.M., Zemel, R.S. (2009). Collaborative prediction and ranking with non-random missing data. *Proc. 3rd ACM Conference on Recommender Systems*, 5–12.

[74] Marra, G., Radice, R., Filippou, P. (2017). Regression spline bivariate probit models: a practical approach to testing for exogeneity. *Communications in Statistics-Simulation and Computation, 46*(3), 2283–2298.

[75] Marshall, A., Tang, L., Milne, A. (2010). Variable reduction, sample selection bias and bank retail credit scoring. *Journal of Empirical Finance* 17(3), 501–512.

[76] Martens, D., Baesens, B., van Gestel, T., Vanthienen, J. (2007). Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research, 183*(3), 1466–1476.

[77] Meng, C.L., Schmidt, P. (1985). On the cost of partial observability in the bivariate probit model. *International Economic Review*, 71–85.

[78] Nguyen, H.T. (2016). Reject inference in application scorecards: evidence from France, Working paper, Paris Nanterre University, Paris.

[79] Niculescu-Mizil, A., Caruana, R. (2005). Obtaining calibrated probabilities from boosting. *Proc. 21st Conference on Uncertainty in Artificial Intelligence*, 28–33.

[80] Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22(2), 199–210.

[81] Rosenbaum, P.R., Rubin, D.B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika, 70*(1), 41–55.

[82] Sadhwani, A., Giesecke, K., Sirignano, J. (2020). Deep learning for mortgage risk. *Journal of Financial Econometrics, 19*(2), 313–368.

[83] Saenko, K., Kulis, B., Fritz, M., Darrell, T. (2010). Adapting visual category models to new domains. *Proc. 11th European Conference on Computer Vision*, 213–226 (Springer).

[84] Satpal, S., Sarawagi, S. (2007). Domain adaptation of conditional probability models via feature subsetting. *Proc. 11th European Conference on Principles of Data Mining and Knowledge Discovery*, 224–235.

[85] Shen, F., Zhao, X., Kou, G. (2020). Three-stage reject inference learning framework for credit scoring using unsupervised transfer learning and three-way decision theory. *Decision Support Systems, 137*, 113366.

[86] Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference, 90*(2), 227–244.

[87] Simester, D., Timoshenko, A., Zoumpoulis, S.I. (2020). Efficiently evaluating targeting policies: Improving on champion vs. challenger experiments. *Management Science, 66*(8), 3412–3424.

[88] Simester, D., Timoshenko, A., Zoumpoulis, S.I. (2020). Targeting prospective customers: Robustness of machine-learning methods to typical data challenges. *Management Science, 66*(6), 2495–2522.

[89] Sirignano, J., Giesecke, K. (2019). Risk analysis for large pools of loans. *Management Science, 65*(1), 107–121.

[90] Su, Y., Dimakopoulou, M., Krishnamurthy, A., Dudík M. (2020). Doubly robust off-policy evaluation with shrinkage. *Proc. 37th International Conference on Machine Learning*, 9167–9176.

[91] Sugiyama, M., Krauledat, M., Müller, K.R. (2007). Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research, 8*, 985–1005.

[92] Sugiyama, M., Müller, K.R. (2006). Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions, 23*(4), 249–279.

[93] Sugiyama, M., Nakajima, S., Kashima, H., Von Buenau, P., Kawanabe, M. (2007). Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in Neural Information Processing Systems, 7*, 1433–1440.

[94] Sugiyama, M., Ogawa, H. (2001). Subspace information criterion for model selection. *Neural Computation, 13*(8), 1863–1889.

[95] Sun, B., Feng, J., Saenko, K. (2016). Return of frustratingly easy domain adaptation. *Proc 30th AAAI Conference on Artificial Intelligence.*

[96] Tian, Y., Yong, Z., Luo, J. (2018). A new approach for reject inference in credit scoring using kernel-free fuzzy quadratic surface support vector machines. *Applied Soft Computing, 73*, 96–105.

[97] Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., Baesens, B. (2017). Gotcha! Network-based fraud detection for social security fraud. *Management Science, 63*(9), 3090–3110.

[98] Verstraeten, G., Van den Poel, D. (2005). The impact of sample bias on consumer credit scoring performance and profitability. *Journal of the Operational Research Society, 56*, 981–992.

[99] Walter, S.D. (2005). The partial area under the summary ROC curve. *Statistics in Medicine, 24*(13), 2025–2040.

[100] Wang, F., Rudin, C. (2017). Extreme dimension reduction for handling covariate shift. *arXiv preprint arXiv:1711.10938.*

[101] Wei, Y., Yildirim P., Van den Bulte, C., Dellarocas, C. (2016). Credit scoring with social network data. *Marketing Science, 35*(2), 234–258.

[102] Wu, I.D., Hand, D.J. (2007). Handling selection bias when choosing actions in retail credit applications. *European Journal of Operational Research* 183(3), 1560–1568.

[103] Xia, Y. (2019). A novel reject inference model using outlier detection and gradient boosting technique in peer-to-peer lending. *IEEE Access* 7, 92893–92907.

[104] Xia, Y., Yang, X., Zhang, Y. (2018). A rejection inference technique based on contrastive pessimistic likelihood estimation for P2P lending. *Electronic Commerce Research and Applications, 30*, 111–124.

[105] Yang, J., Yan, R., Hauptmann, A.G. (2007). Adapting SVM classifiers to data with shifted distributions. *Proc. 7th IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, 69–76.

[106] Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. *Proc. 21st International Conference on Machine learning*, 903–910.

Chapter 7

Fairness in Credit Scoring: Assessment, Implementation and Profit Implications

Publication

Kozodoi, N., Jacob, J., & Lessmann, S. (2021). Fairness in Credit Scoring: Assessment, Implementation and Profit Implications. European Journal of Operational Research.

Abstract

The rise of algorithmic decision-making has spawned much research on fair machine learning (ML). Financial institutions use ML for building risk scorecards that support a range of credit-related decisions. Yet, the literature on fair ML in credit scoring is scarce. The paper makes three contributions. First, we revisit statistical fairness criteria and examine their adequacy for credit scoring. Second, we catalog algorithmic options for incorporating fairness goals in the ML model development pipeline. Last, we empirically compare different fairness processors in a profit-oriented credit scoring context using real-world data. The empirical results substantiate the evaluation of fairness measures, identify suitable options to implement fair credit scoring, and clarify the profit-fairness trade-off in lending decisions. We find that multiple fairness criteria can be approximately satisfied at once and recommend separation as a proper criterion for measuring the fairness of a scorecard. We also find fair in-processors to deliver a good balance between profit and fairness and show that algorithmic discrimination can be reduced to a reasonable level at a relatively low cost. The codes corresponding to the paper are available on GitHub.

7.1 Introduction

Financial institutions increasingly rely on machine learning (ML) to support decision-making [13]. The paper considers ML applications in the retail credit market, which is a large and economically important segment of the credit industry. For example, the total outstanding amount of retail credit in the US exceeded $4,161 billion in 2020[1]. ML-based scoring models, also called scorecards, have played a major role in the approval of the corresponding loans.

In 2016, the Executive Office of the President of the US published a report on algorithmic systems, opportunity, and civil rights [18], which highlights the dangers of automated decision-making to the detriment of historically disadvantaged groups. It emphasizes credit

[1]Source: `https://www.federalreserve.gov/releases/g19/current`

scoring as a critical sector with a large societal impact, calling practitioners for using the principle of “equal opportunity by design" across different demographic groups. Similar actions were taken by the EU when they supplemented their General Data Protection Regulation with a guideline that stresses the need for regular and systemic monitoring of the credit scoring sector [17]. The guidelines issued by the EU and the US evidence political concern that potential violations of anti-discrimination law in credit scoring might affect debt and wealth distributions and have undesired economic effects on the society [33].

A growing literature on fair ML echos these concerns and proposes a range of statistical fairness measures and approaches for their optimization. It is common practice to discuss algorithmic fairness through the lens of differences between groups of individuals. The groups emerge from one or multiple categorical attributes that are considered sensitive. Examples include gender, religious denomination or ethnic group. The goal of fair ML is then to ensure that model predictions meet statistical fairness criteria. Narayanan [37] distinguishes 21 such criteria, while Barocas et al. [2] show that most criteria can be derived from one of three main fairness measures: independence, separation, and sufficiency. Beyond quantifying fairness in model-based predictions, fairness criteria also serve as constraints or objectives in the optimization problem that underlines the training of an ML model. Approaches to adjust model training to optimize fairness criteria next to common indicators of model fit are known as fairness processors.

Surprisingly, the literature on fair ML and credit scoring share few touching points. As we detail in Section 7.3.1, only three studies [21, 24, 33] have considered the interface between the two disciplines. None of them focuses on operational decisions in the loan approval process and the potential trade-off between fairness and profit. Therefore, the goal of the paper is to i) provide a broad overview and systematization of recently developed fairness criteria and fairness processors, and to ii) empirically test their adequacy for credit scoring. While the fairness enhancing procedures that we consider are not new and have been developed in the fair ML literature, we suggest that our holistic and integrative perspective is useful to help risk analysts stay abreast of recent developments in that literature, judge their impact on credit scoring practices, and focus future research initiatives concerning fair credit scoring.

In pursuing its objective, the paper makes the following contributions: First, we revisit statistical criteria for measuring fairness and examine whether these criteria and their underlying understanding of distributional equality are appropriate for credit scoring. Given that different fairness criteria typically conflict with one another [10], our analysis is useful to inform the selection of a suitable fairness criterion (or set of criteria). Considering the relative costs of classification errors for banks and retail clients, we identify separation as a preferable criterion to appraise fairness in a lending context. More generally, our analysis may raise awareness for the risk of algorithmic discrimination in credit scoring, which, given the sparsity of prior work on the topic, may be seen as a valuable contribution to the credit risk community.

Second, we review and catalog state-of-the-art fairness processors across multiple important dimensions, including the target fairness criterion, the implementation method, and requirements for the classification problem. The catalog provides a systematic overview of fairness processors and clarifies whether and when these meet requirements associated with loan approval processes and the application context of credit scoring. The catalog also addresses the critique of Mitchell et al. [36], who demand a more uniform fairness terminology among scholars.

Last, we empirically compare a range of different fairness processors along several performance criteria using seven real-world credit scoring data sets. Unlike prior studies on fair ML, our analysis recognizes prediction performance indicators that are established in credit scoring and, importantly, the profitability of a scoring model. Furthermore, to extend the conceptual discussion on the suitability of the fairness criteria for credit scoring, we measure fairness not only with the criterion optimized by a processor but a range of different fairness criteria. The corresponding results provide original insights concerning the agreement among fairness criteria in credit scoring and their compatibility with profit. More specifically, our comparative analysis contributes to the empirical credit scoring literature by identifying fairness processors that best serve the interests and requirements of risk analysts and by elucidating the trade-off between profitability and fairness of a credit scoring system. A deeper understanding of this trade-off is crucial for managers and policy-makers to decide on the deployment of fairness enhancing procedures in financial institutions and regulatory directives to enforce certain levels of fairness, respectively.

7.2 Theoretical Background

This section covers relevant background on fair ML. We first examine methods to integrate fairness constraints into the model development pipeline and than review established fairness criteria. We focus on independence, separation and sufficiency because these criteria encompass a variety of other fairness concepts [2, 36]. Table 7.8.1 in the Appendix details how independence, separation and sufficiency have synonymously been referred to in the literature and how they relate to the other formulations of fairness.

7.2.1 Fairness Optimization in the Modeling Pipeline

Research on fair ML has recently emerged from the continuous integration of automated decision-making into important areas of social life and fairness concerns arising during this process [3]. Much fair ML literature focuses on classification settings in which an unprivileged demographic group experiences discrimination through a classification model [36]. Several attempts have been made to formalize the concept of fairness. Incorporating the corresponding fairness criteria in the ML pipeline facilitates measuring the degree to which class predictions discriminate against minorities [2].

Algorithmic interventions designed to implement statistical fairness constraints are denoted as fairness processors. A processor can alter different stages in the ML pipeline. The literature distinguishes three methods of intervention: pre-processing, in-processing and post-processing [4]. Their application generally depends on the conceptual and technical feasibility of a given prediction task. Figure 7.2.1 illustrates the fairness processors within an ML pipeline. We describe selected approaches from each group in Section 7.4.

Integrating a fairness processor into the pre-processing stage transforms the training data such that the input to a model is fair with respect to one or more sensitive features. Typically, fair pre-processing involves decorrelating the feature space with the sensitive attribute [e.g., 7]. Even though modifying the training data is sometimes not possible or practical, the advantage of fair pre-processing is that if fairness is ensured before ML model training, it will also be ensured during the next model development steps [2].

In-processing methods introduce auxiliary fairness constraints during ML model training. Then, training involves minimizing the empirical risk of the model while also optimizing a fairness criterion. In-processing renders a learned classifier (approximately) fair for the training data [45]. Optimizing fairness during training has the potential to generate the highest utility as the tuning process also considers the fairness constraint. At the same time, in-processors are typically developed for settings with specific requirements (e.g., supporting only a single sensitive attribute), which limits their generality [2]. Another disadvantage is that implementing a fair in-processor requires full access to the training process and the input data. This is especially problematic in heavily regulated domains such as credit scoring, where changes to a risk model might require regulatory approval and are generally associated with high costs.

After an ML model is trained, post-processing can be applied to adjust the learned

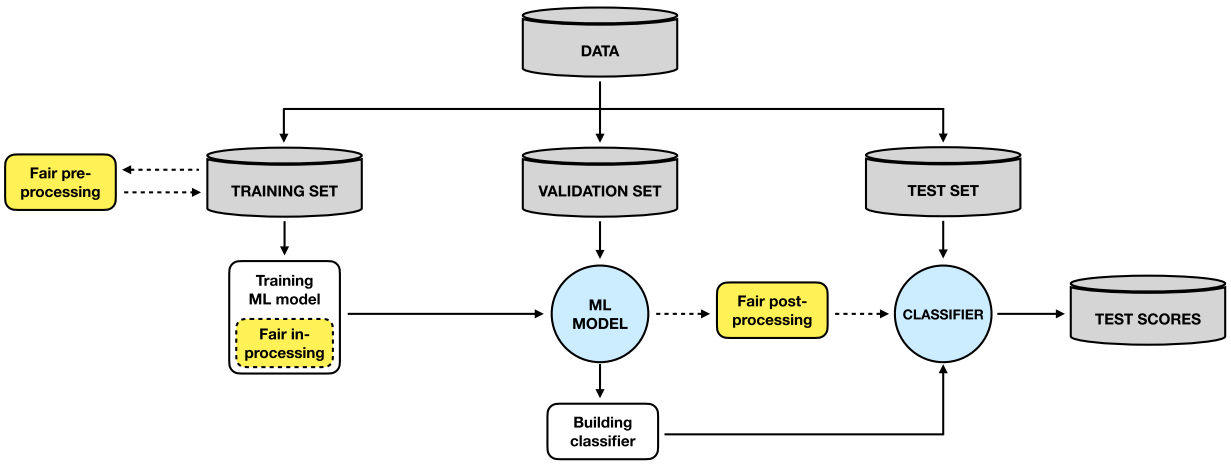

Figure 7.2.1. Fairness Integration in the ML Pipeline

The figure demonstrates three stages of fairness integration in the machine learning pipeline: in-processing, pre-processing and post-processing.

classifier or change its predictions according to the requirements of a particular fairness criterion [24]. The standard procedures include modifying the predicted scores or labels for specific observations. Unlike pre- or in-processing, post-processors need no information about the input data or the base model. This has the advantage that post-processors can be applied to any set of predictions. However, generality has a price. Post-processing is often less effective than alternative approaches and may substantially decrease classification accuracy [2].

7.2.2 Fairness Criteria

This subsection introduces three established fairness criteria from a credit scoring perspective. Consider a setting in which a financial institution uses data on previous customers to predict whether a loan applicant will default. Let $X \in \mathbb{R}^k$ denote the k features of a loan applicant and $y \in \{0, 1\}$ a random variable indicating if the applicant repays the loan ($y = 1$) or defaults ($y = 0$). The institution approves applications using a scoring model that predicts risk scores $s(X) = \mathbf{P}(y = 1|X)$. The score function can be turned into a classifier by accepting customers with scores above a cutoff τ. Let $x_a \in \{0, 1\}$ denote a protected attribute associated with certain characteristics of an applicant. For example, x_a could indicate whether she has a disability ($x_a = 1$) or not ($x_a = 0$). Clearly, the value of x_a must not impact the decision of the credit institution.

In the following, we consider a binary protected attribute to simplify the exposition. The discussed fairness criteria generalize to multinomial protected attributes (i.e., protected attributes with more than two unique values). Also, note that the fair ML literature often uses the terms protected attribute and sensitive attribute interchangeably. From a methodological perspective, it is less important whether the use of an attribute is socially undesirable or regulated by law. We use the term sensitive attribute throughout the paper while acknowledging that our example attribute disability is not only sensitive but protected. The groups created when splitting individuals by a sensitive attribute are referred to as sensitive groups.

Independence

The score $s(X)$ satisfies independence at a cutoff τ if the fraction of customers classified as good risks ($y = 1$) is the same in each sensitive group. Formally, this condition can be written as:

$$\mathbf{P}\left[s(X \mid x_a = 0) > \tau\right] = \mathbf{P}\left[s(X \mid x_a = 1) > \tau\right] \tag{7.2.1}$$

Equation (7.2.1) states that $s(X)$ is statistically independent of the sensitive attribute x_a [2]. Classifier predictions are not affected by the sensitive attribute, and the probability to be classified as a good risk is the same in both groups [40]. In the prior work, the independence condition is also known as demographic or statistical parity [10].

This strict constraint is usually not feasible for real-world applications like credit scoring,

as the resulting loss in model performance can make a business unsustainable. Therefore, it is a common practice in anti-discrimination law to allow the score and the sensitive attribute to share at least some mutual information and introduce a relaxation of the independence criterion [3]. The Equal Opportunity Credit Act has a regulation that is referred to as the "80 percent rule" [19]. The rule requires that $\mathbf{P}(s(X \mid x_a = 1) > \tau) \leq 0.8 \cdot \mathbf{P}(s(X \mid x_a = 0) > \tau)$, where $\{x_a = 0\}$ is the privileged group [30].

Following the relaxation of the independence condition suggested in the prior work [2], we measure independence using a metric denoted as IND, which we define as:

$$\text{IND} = |\mathbf{P}\left[s(X \mid x_a = 0) > \tau\right] - \mathbf{P}\left[s(X \mid x_a = 1) > \tau\right]| \tag{7.2.2}$$

A positive difference between the two terms implies that the group $\{x_a = 0\}$ is considered the privileged group and vice versa. The closer IND is to zero, the lower is the discrimination.

Separation

The separation condition, also known as the equalized odds condition, is satisfied if the classification based on the predicted score $s(X)$ and the cutoff τ is independent on x_a conditional on the true outcome y [2]. Formally, the score $s(X)$ satisfies separation at a cutoff τ if:

$$\begin{cases} \mathbf{P}\left[s(X \mid y = 0, x_a = 0) > \tau\right] = \mathbf{P}\left[s(X \mid y = 0, x_a = 1) > \tau\right] \\ \mathbf{P}\left[s(X \mid y = 1, x_a = 0) \leq \tau\right] = \mathbf{P}\left[s(X \mid y = 1, x_a = 1) \leq \tau\right] \end{cases} \tag{7.2.3}$$

The expression in the first line compares the false positive rate (FPR) across the sensitive groups, whereas the second line compares the false negative rate (FNR) per group. The separation criterion, therefore, requires that the FNR and the FPR are the same for the sensitive groups.

Separation acknowledges that x_a may be correlated with y (e.g., applicants with a disability might can a higher default rate). However, the criterion prohibits the use of x_a as a direct predictor for y. When the difference between group sizes is large, the criterion will punish models that perform well only on the majority group [24]. To measure the degree to which the separation condition is satisfied, we suggest using a criterion denoted as SP, which we define as:

$$\text{SP} = \frac{1}{2}\left|\left(\text{FPR}_{\{x_a=1\}} - \text{FPR}_{\{x_a=0\}}\right) + \left(\text{FNR}_{\{x_a=1\}} - \text{FNR}_{\{x_a=0\}}\right)\right| \tag{7.2.4}$$

SP calculates the average absolute difference between the group-wise FPR and FNR. A positive difference between each of the two group-wise error rates indicates that the $\{x_a = 0\}$ group has a lower misclassification rate and is, therefore, the privileged group. Perfect separation (i.e., SP = 0) is observed when the group-wise FPR and FNR are equal. Higher values of SP indicate stronger discrimination through a larger difference in model performance across the sensitive groups.

Sufficiency

The score $s(X)$ is sufficient at a cutoff τ if the likelihood that an individual belonging to a positive class is classified as positive is the same for both sensitive groups [2]. This implies that for all values of $s(X)$ the following condition holds:

$$\mathbf{P}(y = 1 \,|\, s(X) > \tau, x_a = 0) = \mathbf{P}(y = 1 \,|\, s(X) > \tau, x_a = 1) \tag{7.2.5}$$

Equation (7.2.5) requires that the positive predictive value (PPV) is the same for the sensitive groups [10]. This paper defines the sufficiency metric SF as the absolute difference between the group-wise PPV:

$$\text{SF} = \left|\text{PPV}_{\{x_a=0\}} - \text{PPV}_{\{x_a=1\}}\right| \tag{7.2.6}$$

A large difference between the group-wise PPV indicates inconsistent model performance across the sensitive groups. The closer SF is to zero, the higher is the achieved sufficiency.

7.3 Fairness and Credit Scoring

The section discusses the interplay between fair ML and credit scoring. We summarize previous work in the field and examine the adequacy of fairness criteria for credit scoring.

7.3.1 Prior Work on Fair Credit Scoring

Prior literature on fair ML for credit scoring is surprisingly sparse. To our best knowledge, only three studies address algorithmic discrimination in credit scoring, and their focus differs substantially from that of this study. A first study by [21] considers the credit market. The authors formalize the introduction of ML as a market intervention and examine the corresponding effect on interest rates in demographically different groups. [33] take a similar perspective. Referring to the sample-selection bias, which arises from training scorecards on previously accepted cases [1], they argue that selection bias leads to scorecards overestimating the creditworthiness of some groups of applicants and perpetuates existing unfairness. To remedy this effect, [33] call for mathematical constraints that optimize fairness as a long-term societal goal. However, the formulation of these constraints is still subject to further research. More generally, the long-term perspective of [21] and [33] emphasizes regulatory questions and is orthogonal to the focus on static fairness interventions, which prevails in the fair ML literature. These interventions address operational loan approval decisions and provide concrete approaches to remedy algorithmic bias.

Focusing on fairness interventions, a third study of [24] is related to this paper more closely. [24] propose the equalized odds fairness criterion and develop an algorithm that adjusts classifier predictions to raise fairness according to this criterion. The authors report

enhanced fairness compared to a maximum profit benchmark using a credit scoring example based on FICO scores. In comparison to the focal paper, [24] focus on the specific combination of one fairness processor and one fairness criterion. Their study does not examine the trade-off between profit and fairness and provides limited empirical evidence on how equalized odds compare to other fairness criteria or how fairness is best ensured in an ML pipeline.

In summary, the main distinction between the focal paper and previous studies on fairness in credit scoring is that we undertake a comprehensive empirical analysis of alternative fairness criteria and fairness processors, which optimize these criteria. Prior work fails to account for the breadth of approaches that have been proposed in the scope of fair ML. Also, no previous study examines the interplay between fairness criteria and processors. Therefore, we aim at consolidating different advancements in fair ML, discussing their suitability for credit scoring, and providing rich empirical results that clarify the degree to which fairness constraints affect the predictive ability of credit scorecards and the corresponding profit implications, and how the trade-off between fairness and profit develops across fairness criteria and processors. We hope that our results offer actionable insights on how to set and pursue fairness objectives in credit scoring.

7.3.2 Fairness Criteria for Credit Scoring

The choice of the fairness criterion has severe consequences for the social impact of lending decisions [33]. An unconstrained scoring model will take full advantage of the available (sensitive) information and discriminate between protected groups if this enhances predictive performance. The purpose of introducing fairness is, therefore, to adjust decision-making (i.e., scoring) practices for a better, discrimination-free outcome. According to the U.S. anti-discrimination law, for example, the demographic properties of a loan applicant should not influence lending decisions [16]. Arguably, the societal goal behind such law is an equal opportunity for financial well-being across demographically different groups. Achieving this goal in credit scoring is difficult as clients face unequal misclassification costs. Applicants that are denied a loan they could have repaid face the cost of a missed opportunity to enhance their social and economic position. However, if applicants receive a loan they cannot repay, they are confronted with financial debt and a long-term worsening of their financial situation as future access to financing will be more difficult. With these characteristics of credit scoring in mind, the following considerations elaborate on the extent to which independence, separation and sufficiency fulfill the goal of equal opportunity for financial well-being in society.

Forcing independence on a scoring model's results in the same rate of accepted customers within sensitive groups. The problem with this approach is that the ability to repay a loan can have a different distribution in each group [2]. If this is the case, but members of both

groups have the same probability of receiving a loan, one group will experience more actual defaults. For a client, the consequences of defaulting can be more severe than the opportunity costs associated with a rejected application. Typically, the historically unprivileged group has a higher rate of non-solvent customers. Handing out loans to such individuals might worsen their financial situation in the long term [24]. Instead of achieving fairness, this can lead to further perpetuating existing unfairness. The goal of better financial equality would not be met, and the financial gap in society could become even wider.

The separation criterion addresses this dilemma and acknowledges that a sensitive attribute might correlate with default rates. Requiring the same error rates between groups but allowing different positive classification rates, separation achieves a fair result that is closer to the reality of credit allocation decisions and more desirable from a customer's perspective. More precisely, separation accounts for different misclassification costs between groups. On the contrary, separation would be inadequate if credit scoring had a strictly preferred outcome for a customer, as is the case in domains like college admission [36]. Interestingly, the first formulation of the separation criterion in the context of ML by [24] is based on the example of the credit scoring domain and the limitations of the independence criterion to meet its requirements.

Sufficiency requires the ratio of true positive classifications over all positive classifications to be the same for the sensitive groups. This concept has two disadvantages for credit scoring. First, it allows for substantial discrimination in separation. For both groups, the proportion of correctly labeled non-default clients can be the same, satisfying sufficiency. In contrast, the likelihood of a potential non-default customer being classified as a bad risk can still differ between groups, violating the separation constraint. Second, most ML algorithms are designed to achieve sufficiency without integrating a fairness constraint if the model can predict the sensitive attribute from the other features [2]. In credit scoring, the question would, therefore, be if the current procedure for assessing a customer's default risk and the associated distribution of loans is fair. The literature suggests a negative answer to this question [21, 33, 24]. Hence, sufficiency appears less suitable for credit scoring.

Based on these considerations, the separation criterion appears most suitable to achieve a desirable form of fairness in credit scoring. Separation accounts for the imbalanced misclassification costs of the customer, and, as these imbalanced costs also exist for the financial institution, separation is also able to consider the interests of the loan market.

The considerations provided in this section suggest that the question of which fairness constraint is most adequate for credit scoring should be a part of a wider academic and societal debate. Such a democratic process should also acknowledge the importance of studying the long-term effects of implementing different fairness constraints to judge whether the societal goal of better financial equality between demographic groups can be achieved with specific interventions [33].

7.4 Methodology

This section systematically reviews and catalogs fairness processors suggested in the prior work across different dimensions and discusses their applicability in credit scoring. Using the constructed catalog, we select and describe eight fairness processors that are part of the empirical study.

7.4.1 Cataloging Fairness Processors

The fair ML literature has developed a variety of fairness processors to implement independence, separation and sufficiency constraints. The complexity between these processors varies considerably, from simply relabeling the prediction outcomes [e.g., 27] to complex deep learning approaches for training a discrimination-free classifier [e.g., 48]. Furthermore, some processors are limited to specific problem setups. This motivates us to develop a structured overview of fairness processors with respect to their characteristics and applicability. Specifically, we catalog existing fairness processors in Table 7.4.1 using six dimensions: (i) point of intervention into the ML pipeline; (ii) optimized fairness criterion; (iii) classification problem type supported by a processor (binary or multinomial); (iv) possible number of sensitive attributes (one or multiple); and (vi) supported types of sensitive attributes (binary or multinomial).

Three main conclusions emerge from Table 7.4.1. First, the majority of processors implement the independence criterion. This may come the other criteria being invented only recently (see Table 7.8.1 in Appendix 7.8.1 for comparison). Furthermore, independence allows implementation via pre-processing, which provides an additional point of intervention in the ML pipeline. In many scenarios, however, fairness through independence may not be a suitable choice. This calls for additional processors that implement the other two criteria.

Second, the choice of a suitable fairness processor is limited by the application and implementation context of a scorecard. The application context determines the type of target variable and sensitive attribute(s) to be handled by a processor. For instance, in a setup with multiple sensitive attributes optimizing separation is only possible via the adversarial debiasing or reject option classification. This is a severe limitation for credit scoring because financial institutions commonly face several protected attributes: the U.S. anti-discrimination law distinguishes nine bases that must not influence lending decisions, including race, color, religion and other customer attributes [16]. The implementation context can also limit possible points of intervention in the ML pipeline. Replacing a scorecard with a fair in-processor might require regulatory approval and incur additional costs. Post-processors are easier to implement since they are agnostic of the input data and the scorecard and only require access

Table 7.4.1. Fairness Processors

Fairness processor	Reference	Method	Criterion	MT	MS	MA	PE	This paper
Reweighting	Calders et al. [5]	PRE	IND					✓
Massaging	Calders et al. [5]	PRE	IND					
Classification without discrimination	Kamiran, Calders [26]	PRE	IND					
Discrimination discovery K-NN	Luong et al. [35]	PRE	IND	✓				
Fair representation learning	Zemel et al. [47]	PRE	IND	✓				
Disparate impact remover	Feldman et al. [19]	PRE	IND		✓	✓		✓
Variational fair autoencoder	Louizos et al. [34]	PRE	IND	✓	✓	✓		
Feature adjustment	Johndrow et al. [25]	PRE	IND	✓	✓	✓		
Discrimination-free pre-processing	Calmon et al. [7]	PRE	IND		✓	✓		
Prejudice remover regularizer	Kamishima et al. [28]	IN	IND	✓				✓
Fair accuracy maximizer	Zafar et al. [46]	IN	IND	✓	✓	✓		
Non-discriminatory Learner	Woodworth et al. [43]	IN	SP					
Adversarial debiasing	Zhang et al. [48]	IN	SP	✓	✓	✓		✓
Meta-fairness algorithm	Celis et al. [8]	IN	IND, SP, SF		✓	✓		✓
Group-wise Platt scaling	Platt [39], Barocas et al. [2]	POST	SF	✓	✓	✓		✓
Group-wise histogram binning	Zadrozny, Elkan [44]	POST	SF	✓	✓	✓		
Group-wise isotonic regression	Niculescu-Mizil, Caruana [38]	POST	SF	✓	✓	✓		
Fairness-aware classifier	Calders & Verwer [6]	POST	IND					
Reject option classification	Kamiran et al. [27]	POST	IND, SP		✓	✓		✓
Fairness constraint optimizer	Goh et al. [22]	POST	IND	✓	✓	✓		
Equalized odds processor	Hardt et al. [24]	POST	SP		✓		✓	✓
Calibrated equalized odds	Pleiss et al. [40]	POST	SP					

Abbreviations: IND = Independence, SP = separation, SF = sufficiency; PRE = pre-processor, IN = in-processor, POST = post-processor; MT = multinomial target, MS = multinomial sensitive attribute, MA = multiple sensitive attributes, PE = profit-driven evaluation.

to the predicted scores.

Third, it is a standard procedure to embed the fairness processor into an accuracy-optimizing framework. The loss in predictive accuracy is commonly used as a performance measure to judge the cost of integrating a fairness constraint. In line with this framework, [20] conducted a comparative study to examine the achieved fairness and accuracy of four fairness processors. However, recent credit scoring literature criticizes the practice of using standard performance measures for evaluating scoring models and calls for profit-driven evaluation [42]. In such a setup, evaluation of fairness processors should be performed with a profit maximization objective instead of standard statistical performance measures such as accuracy.

To conclude, the catalog suggests that a comparative analysis of fairness processors under profit maximization is needed to clarify the "cost of fairness". We argue that the profitability aspect is underrepresented in the fair ML literature, while it is highly relevant for real-world applications. A better understanding of the (dis)agreement of profitability and different fairness criteria is also useful for policy making as it sheds some light on the thorny question of which criterion lending institutions should emphasize. Which fairness processor to use for optimizing the desired criterion is yet another question with high relevance for practice. Prior literature offers limited guidance due to assessing processors typically only in terms of the single criterion that this processor implements. Contributing toward answering these pressing questions is the overall goal of the paper.

7.4.2 Selected Fairness Processors

This subsection overviews eight fairness processors from the catalog presented in Table 7.4.1. The selection of processors covers all combinations of fairness interventions. Following the setup introduced in Section 7.2, we consider a credit scoring setup with a binary target variable $y \in \{0, 1\}$ and a binary sensitive attribute $x_a \in \{0, 1\}$ to introduce the processors. Some of the considered processors also generalize to multinomial target and sensitive attributes (see Table 7.4.1 for details).

Pre-Processors

Fairness pre-processors transform the input data to achieve fairness. Reweighting is a pre-processor that assigns weights to each observation in the training set based on the overall probabilities of the group-class combinations [5]. Thus, weights for observations with $(x_a = 1, y = 1)$ are greater than weights for observations with $(x_a = 0, y = 1)$ if members of the group $\{x_a = 1\}$ have a lower probability to belong to a positive class than those of the group $\{x_a = 0\}$:

$$W(X \mid x_a = 1,\, y = 1) = \frac{\mathbf{P}_{exp}(x_a = 1 \mid y = 1)}{\mathbf{P}_{obs}\,(x_a = 1 \mid y = 1)}\,, \tag{7.4.7}$$

where $\mathbf{P}_{exp}$ is the expected probability and $\mathbf{P}_{obs}$ is the observed probability. For instance,

assume that 90% of all individuals belong to the positive class and 20% percent belong to the group $\{x_a = 1\}$. Then, $\mathbf{P}_{exp}(x_a = 1 \mid y = 1) = 0.9 \cdot 0.2 = 0.18$. If, in fact, only 12% of all cases in $\{x_a = 1\}$ belong to the positive class, then $W(X \mid x_a = 1, y = 1) = \frac{0.18}{0.12} = 0.9$.

Based on the computed weights, a fair training set is resampled with replacement such that combinations with a higher weight reappear more often. This procedure helps to fulfill the independence criterion. A discrimination-free classifier can then be trained on the resampled data.

Another pre-processing technique is the disparate impact remover proposed by [19]. The intuition behind this processor is to ensure independence by prohibiting the possibility of predicting the sensitive attribute x_a with the other features in X and the outcome y. This is achieved by transforming X into $\overline{X}$ while preserving the rank of X within sensitive groups defined by x_a. By preserving the rank of X given x_a, the classification model $f(\overline{X})$ will still learn to choose higher-ranked credit applications over lower-ranked ones based on the other features.

The transformation is performed using an interpolation based on a quantile function and the cumulative distribution of $F : \mathbf{P}(X \mid x_a = a)$. This ensures that given the transformed $\overline{X}$ at some rank, the probability of drawing an observation given $x_a = a$ is the same as for the entire data set. Hence, x_a cannot be predicted with the other attributes, and the independence criterion is fulfilled. Since ensuring perfect independence can have a strong negative impact on a classifier utility, the transformation can be modified to only partially remove disparate impact. The meta-parameter $\lambda \in [0, 1]$ allows controlling the desired level of fairness-utility trade-off during transformation.

In-Processors

In-processors achieve fairness when building a classifier. One of such methods, prejudice remover, introduces a fairness-driven regularization term to the classification model [28]. Regularization is a standard statistical approach to penalize a model for some undesired behavior. This is typically done by adding a regularizer term to the loss function.

The fairness-driven regularization introduced by [28] is based on the prejudice index PI, which quantifies the degree of unfairness based on the independence criterion:

$$\mathrm{PI} = \sum_{(y,x_a)\in D} \mathbf{P}(y,x_a) \ln \frac{\mathbf{P}(y,x_a)}{\mathbf{P}(x_a)\mathbf{P}(y)}, \tag{7.4.8}$$

where $\mathbf{P}(y, x_a)$, $\mathbf{P}(y)$ and $\mathbf{P}(x_a)$ are empirical distributions of y and x_a over the sample D. PI measures the amount of mutual information between y and x_a. High values of PI indicate that a sensitive attribute x_a is a good predictor for y. The optimization problem extends to:

$$\min_f L\left[f\left(X\right), y\right] + \eta \mathrm{PI}\ , \tag{7.4.9}$$

where $L(\cdot)$ is the underlying loss function of the model $f(X)$, and η controls the importance of the term PI. In this study, we tune η to maximize the profitability of a scorecard. The regularization term ensures that the sensitive attribute x_a becomes less influential in the final prediction.

Adversarial debiasing is another in-processor that stacks two neural networks with contrary objectives on top of each other [48]. The first network (predictor) is trying to learn a function to predict y given X, while also minimizing the success of the second network. The second network (adversary) takes the output layer of the first model $\hat{y}$ and the true labels y as input and tries to predict the sensitive attribute x_a. Both models have objective-specific loss functions and weights that can be optimized using standard gradient-based optimization methods such as stochastic gradient descent or Adam [29].

The adversary is assumed to have weights U and loss function $L_A(\hat{x}_a,x_a)$. The weights U are updated according to the gradient $\nabla_U L_A$ to minimize L_A. The weights of the predictor denoted as W are modified based on a gradient that minimizes its loss function $L_P(\hat{y},y)$ but also maximizes the loss function of the adversary: $\nabla_W L_P\left(\hat{y},y\right) - \ \alpha\ \nabla_W L_A\left(\hat{x}_a,x_a\right)$, where α is a meta-parameter.

Since the adversary takes the output of the predictor $\hat{y}$ as input, the predictor aims to hold back any additional information about the sensitive attribute x_a in its output $\hat{y}$ as it would improve the adversary's loss. In other words, the predictor will try to deceive the adversary and not share any additional information in $\hat{y}$. As y is known to the adversary, the algorithm acknowledges that the sensitive attribute might correlate with y, and only unnecessary information will be avoided. Hence, the adversarially debiased model will converge towards the separation criterion.

The meta fair classification algorithm is yet another in-processor designed to achieve fairness according to one of the different fairness criteria. For a given criterion, [8] suggest using a corresponding group-wise fairness metric denoted as FM, where similar values of FM across sensitive groups indicate a higher level of fairness. Given a classifier $f(X)$ with a loss function $L\left(f\left(X\right),y\right)$, they add a fairness constraint to the loss optimization problem during training:

$$\min_f L\left(f\left(X\right),y\right) \quad \text{s.t.} \quad \frac{\min\left[\text{FM}(f(X \mid x_a=0)),\ \text{FM}(f(X \mid x_a=1))\right]}{\max\left[\text{FM}(f(X \mid x_a=0)),\ \text{FM}(f(X \mid x_a=1))\right]} \ \geq\ \sigma, \qquad (7.4.10)$$

where $\sigma \in [0,1]$ is a desired fairness bound. Higher values of the fraction in Equation 7.4.10 indicate a higher similarity of FM across sensitive groups, and $\sigma = 1$ implies perfect fairness.

For example, in case of sufficiency, FM is set to positive predictive value (PPV) such that $\text{FM}(f) = \text{PPV}(f) = \frac{\mathbf{P}(f=1 \mid x_a=a,y=1)}{\mathbf{P}(f=1 \mid x_a=a)}$. If the group $\{x_a = 1\}$ has a low PPV and the group $\{x_a = 0\}$ has a high PPV, the fraction in the optimization condition is close to zero. A high σ will, therefore, bound the classifier to a high degree of fairness. During training, the value

for σ can be tuned such that it maximizes profit while minimizing the loss in fairness, i.e., the loss in sufficiency.

Post-Processors

As a post-processing method, reject option classification is based on the output of a learned classifier [27]. In a credit scoring setup, the classifier output is a credit score that reflects the posterior probability to not default for each customer $s(X) = \mathbf{P}(\hat{y} = 1|X)$. The closer the score is to 1 or 0, the higher is the certainty with which the classifier assigns the corresponding labels, whereas a score close to 0.5 implies a high degree of uncertainty.

Reject option classification defines a critical region of high uncertainty and reassigns labels for customers that have predicted scores within this region, such that members of the unprivileged group receive a positive label ($y = 1$) and vice versa. Formally, the critical region is defined as:

$$\max\left[\mathbf{P}(\hat{y} = 1|X)\ ,\ 1 - \mathbf{P}(\hat{y} = 1|X)\right]\ \leq \theta\ , \tag{7.4.11}$$

where $0.5 < \theta < 1$. Given a set of predicted scores and the true outcomes, a suitable value of θ and the number of required posterior reclassifications can be tuned to optimize a fairness criterion (e.g., independence) within a specified interval restricted by the lower and the upper bound of the fairness metric denoted as $[\sigma_l, \sigma_u]$.

Equalized odds processor uses a different logic to post-process classifier predictions. It finds a cutoff value τ that optimizes the predictive performance while satisfying the separation criterion, i.e., ensuring the same false negative and false positive rate per group [24].

Consider the receiver operating characteristic (ROC) curves that depict the trade-off between true and false positive rates for two sensitive groups. In an unfair scenario, the group-wise ROC curves have different slopes, which implies that not all trade-offs are achievable in each group. In the accuracy optimization setting, the optimal cutoff that satisfies sufficiency lies at the intersection of group-wise ROC curves. When optimizing for profit, the misclassification costs are not the same for both error rates. Thus, the optimal cutoff could lie somewhere else. Given a loss function $L(\cdot)$, Hardt et al. [24] suggest to derive a suitable cutoff τ by optimizing the following objective:

$$\begin{aligned}\min \mathbf{P}\left(s(X|x_a = a, y = 0) \leq \tau\right) \cdot L(\hat{y} = 1, y = 0) + \\ \left[1 - \mathbf{P}\left(s(X|x_a = a, y = 1) > \tau\right)\right] \cdot L(\hat{y} = 0, y = 1)\end{aligned} \tag{7.4.12}$$

Platt scaling is a post-processing method that stems from the notion of calibration [39]. Calibration addresses the problem that some classification algorithms are not able to make a statement about the certainty of their prediction, i.e., the probability with which an instance belongs to a certain class. In credit scoring, the predicted score could be an indica-

tor of default risk but not the actual probability of default. A score $s(X)$ is calibrated if $\mathbf{P}(y = 1 \mid s(X) = \tau) = \tau$.

When extending the calibration condition to the group level, it becomes apparent that it implements the sufficiency criterion (see Barocas et al. [2] for proof):

$$\mathbf{P}[y = 1 \mid s(X) = \tau, x_a = 1] = \mathbf{P}[y = 1 \mid s(X) = \tau, x_a = 0] = \tau \tag{7.4.13}$$

To achieve calibration per group, Platt scaling is applied separately to each sensitive group. The method uses the output of a possibly uncalibrated score $s(X)$ as input for logistic regression fitted against the target variable y. Based on the loss function of the logistic regression, the result is a new calibrated score that represents the probability that an instance belongs to the positive class. Formally, Platt scaling minimizes the log-loss $-\mathbb{E}[ylog(\sigma) + (1 - y)\log(1 - \sigma)]$ by finding the optimal parameters a and b of the sigmoid function $\sigma = \frac{1}{1+\exp(aS+b)}$.

7.5 Experimental Setup

7.5.1 Data

The empirical experiment is based on seven credit scoring data sets. Data sets *german* and *taiwan* stem from the UCI Machine Learning Repository[2]. *Pakdd*, *gmsc* and *homecredit* were provided by different companies for the data mining competitions on PAKDD[3] and Kaggle[4]. *Bene* and *uk* were collected from financial institutions in the Benelux and UK [32].

Each data set has a unique set of features describing a loan applicant and loan characteristics. The target variable y is a binary indicator of whether the applicant has repaid the loan ($y = 1$) or not ($y = 0$). Each data set also contains a sensitive demographic attribute x_a indicating the applicant's age group. The Equal Credit Opportunity Act prohibits that demographic characteristics such as the applicants' age impact credit approval decisions. We distinguish two groups of applicants: $\{x_a = 1\}$ contains applications where the applicant's age is below ψ years, and $\{x_a = 0\}$ refers to the applications from customers older than ψ. We set $\psi = 25$, following the findings of [26], who used one of the consumer credit scoring data sets to discover that applicants from different age groups exhibit the greatest disparate impact (i.e., difference in $\mathbf{P}[y = 1 \mid x_a = a]$) at a threshold of 25 years. Table 7.5.1 summarizes the main characteristics of the data sets.

[2]Source: `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`, `https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients`

[3]Source: `https://www.kdnuggets.com/2010/03/f-pakdd-2010-data-mining-competition.html`

[4]Source: `https://kaggle.com/c/home-credit-default-risk`, `https://kaggle.com/c/givemesomecredit`

Table 7.5.1. Credit Scoring Data Sets

Data set	Sample size	No. features	Default rate	Sensitive group rate
german	1,000	61	.30	.19
bene	3,123	82	.33	.12
taiwan	23,531	76	.23	.14
uk	30,000	51	.04	.20
pakdd	50,000	185	.26	.11
gmsc	150,000	68	.07	.02
homecredit	307,511	92	.08	.04

7.5.2 Experimental Setup

On each data set, we implement the eight fairness processors introduced in Section 7.4, following the model development pipeline depicted in Figure 7.2.1[5]. First, we partition the data into training (60%) and test (40%) sets. We then perform five-fold cross-validation on the training set. Each of the five combinations of training folds is used to train a scoring model and implement fairness processors. An unconstrained scoring model (i.e., a model that does not include any fairness-optimizing procedures) serves as a benchmark and represents the profit maximization scenario. Next, we consider in-processors in the form of the prejudice remover, adversarial debiasing and the meta fair algorithm. Relying on an in-processor implies that the trained in-processor serves as a scorecard. This contrasts pre- and post-processors, in which the actual scorecard is still based on a conventional ML algorithm. We consider reweighting and the disparate impact remover to pre-process (i.e., transform) the training data before developing a scoring model. Reject option classification, the equalized odds processor and Platt scaling represent the post-processors in our study. To learn a post-processing model, we apply each of them to the validation fold predictions of the unconstrained scorecard.

Fairness pre- and post-processors, as well as an unconstrained scorecard, use four base classifiers: logistic regression, artificial neural network and the tree-based ensemble learners random forest and extreme gradient boosting (XGB). Using multiple base learners allows us to check the robustness of processors across different classifiers. The base learners are established in credit scoring [e.g., 32, 31], whereby XGB [9] is maybe less known in the community. We include XGB due to its reputation as a highly powerful learning algorithm in Kaggle competitions and its strong performance in a recent credit scoring study by [23], who find XGB outperforming challenging deep learning benchmarks. Meta-parameters of the base classifiers are tuned in a nested four-fold cross-validation on the training data. The meta-parameters of fairness processors are also tuned using grid search. The details on the

[5]The code reproducing the experiments is available at `https://github.com/kozodoi/Fair_Credit_Scoring`

Table 7.5.2. Cost Matrix for Profit Computation

Actual label	Predicted label: Bad risk	Predicted label: Good risk
Bad risk	$\pi_0 F_0(\tau)$ benefit: 0	$\pi_0(1 - F_0(\tau))$ cost: B
Good risk	$\pi_1 F_1(\tau)$ cost: C	$\pi_1(1 - F_1(\tau))$ benefit: C

meta-parameter values and the tuning procedure are provided in the Appendix.

Fairness processors and benchmarks are evaluated on the test set using multiple performance metrics. First, we measure the profitability of a scorecard by computing profit per EUR issued by a financial institution. To estimate profit, we start from the Expected Maximum Profit (EMP) criterion [42]. The EMP measures the incremental profit compared to a base scenario in which loan applications are accepted without screening. This often leads to a small magnitude of EMP differences across classifiers [31] and complicates the interpretation of the metric. To enable a more direct interpretation, we normalize misclassification costs such that the base scenario represents rejecting all applications.

Table 7.5.2 provides the confusion matrix of a scoring model, where π_i are prior probabilities of good and bad risks, and $F_i(\tau)$ are predicted cumulative density functions of the scores of class i given a cutoff value τ. If an applicant is predicted to be a good risk, a financial institution faces cost B in case of an incorrect prediction and earns C from an accurate prediction. In contrast, if an applicant is predicted to be a bad risk, a company faces an opportunity cost C in case of an incorrect prediction. Parameters B and C are defined according to Verbraken et al. [42].

The parameter B reflects the cost associated with misclassifying a bad risk. Providing credit to a defaulter, the company faces a loss; specifically, the expected loss in case of default:

$$B = \frac{\text{LGD} \cdot \text{EAD}}{A} \ , \tag{7.5.14}$$

where LGD refers to the loss given default, EAD is the exposure at default, and A is the principal. B varies between 0 and 1 and several distributions may arise [41]. We follow [42] and treat B as a random variable with probability distribution:

- $B = 0$ with probability p_0 (a customer repays the entire loan after default);
- $B = 1$ with probability p_1 (the bank loses the entire loan);
- B follows a uniform distribution in (0, 1) with $F(B) = 1 - p_0 - p_1$.

The parameter C reflects the opportunity cost or earned benefit associated with good risks. By accepting a good customer, the company earns the equivalent to the return on

investment denoted as ROI:

$$C = \text{ROI} = \frac{I}{A} \,, \tag{7.5.15}$$

where I is the total interest payments. Given these parameters, we compute profit as:

$$\text{Profit} = \int_0^1 \Big[C \cdot \big(\pi_1 (1 - F_1(\tau)) - \pi_1 F_1(\tau) \big) - B \cdot \pi_0 (1 - F_0(\tau)) \Big] f(B) d(B) \tag{7.5.16}$$

This paper follows the empirical findings of [42] and assumes a constant ROI of 0.2664 and the point masses $p_0 = 0.55$ for no loss and $p_1 = 0.1$ for full loss to compute B.

Apart from estimating the profitability of each fairness processor, we also compute the area under the ROC curve (AUC), which is a widely used indicator of the discriminatory ability of a scoring model. In addition, we evaluate fairness by measuring independence, separation and sufficiency. We aggregate the performance of pre- and post-processors over seven credit scoring data sets, five training fold combinations and four base classifiers, obtaining 140 performance estimates per processor. Since in-processors do not require a base classifier, their performance is aggregated over 35 values obtained from seven data sets and five training fold combinations.

7.6 Empirical Results

This section presents the empirical results. We first examine the correlation between the scorecard performance, profitability, and fairness. Next, we compare the performance of different fairness processors. Last, drawing on the findings that suggest a strong negative correlation between profit and fairness, we examine the profit-fairness trade-off to appraise the monetary cost of fairness.

7.6.1 Correlation Analysis

Table 7.6.1 depicts the mean Spearman correlation between the evaluation metrics. The correlation coefficients are computed on the performance estimates obtained from different variants of fairness processors and averaged over the seven credit scoring data sets. The results suggest that the AUC and profit often produce similar model rankings (correlation is 0.80). Still, there is some disagreement between the two measures, which indicates that optimizing profit is important to identify potentially more profitable scorecards. Therefore, we emphasize profit in the following.

Comparing profit and fairness, we observe a moderate negative correlation between independence, separation, and profitability[6]. As expected, integrating fairness constraints to

[6]Higher AUC and profit values indicate better performance, whereas lower values of independence, separation, and sufficiency indicate higher fairness. Therefore, we invert correlation signs between the two former

Table 7.6.1. Rank Correlation between Evaluation Metrics

Metric	AUC	Profit	IND	SP	SF
AUC	1				
Profit	0.8014	1			
IND	−0.4707	−0.3774	1		
SP	−0.3326	−0.2994	0.9477	1	
SF	0.3489	0.1636	−0.2156	−0.1311	1

Abbreviations: AUC = area under the ROC curve, IND = independence, SP = separation, SF = sufficiency.

reduce discrimination prevents a scorecard from taking full advantage of the available information, which decreases profit. At the same time, a weak positive correlation between sufficiency and profit suggests that optimizing profitability without implementing additional fairness constraints could also improve sufficiency. This result confirms the observation that most ML algorithms are designed to automatically achieve sufficiency and implies that directly optimizing sufficiency with a fairness processor is not essential.

A different conclusion emerges from examining the agreement of the other two fairness criteria. As indicated by Table 7.6.1, independence and separation have a strong positive correlation of 0.95. Optimizing either of these two criteria will, therefore, favor models that fulfill both independence and separation. In other words, reducing the mutual information between a sensitive attribute and model predictions also helps to align the parity of error rates across the sensitive groups. This is an interesting finding, given that the former constraint targeted by independence is stricter compared to the one targeted by separation. For a risk analyst, the observed result implies that it is ample to rely on a single fairness criterion. Since separation has a better ability to capture the cost asymmetry (see Section 7.3 for details), we conclude that optimizing and measuring the separation criterion is the most suitable way to integrate and evaluate the fairness of a credit scoring model.

7.6.2 Benchmarking Fairness Processors

Table 7.6.2 provides average performance gains from fairness processors compared to the unconstrained scoring model across the seven credit scoring data sets. A positive gain indicates a better performance of a processor relative to the unconstrained model in terms of a particular evaluation measure. Individual results for each of the data sets are provided in Appendix 7.8.3.

Table 7.6.2 confirms that using a processor to enhance fairness decreases profit compared to the unconstrained model. Results in terms of the AUC mirror this finding, whereby two processors show marginally higher AUC values than the unconstrained model. Table 7.6.2

performance metrics and the three fairness criteria to facilitate the consistent interpretation of the results.

Table 7.6.2. Average Gains from Fairness Processors Relative to the Unconstrained Model

Method	Fairness processor	AUC	Profit	IND	SP	SF
Pre-processing	Reweighting	-3.19%	-23.04%	66.00%	61.24%	-38.18%
	Disparate impact remover	**0.82**%	-10.60%	5.33%	4.50%	**-19.99**%
In-processing	Prejudice remover	0.37%	**-4.28**%	11.51%	9.41%	-202.36%
	Adversarial debiasing	-0.21%	-13.90%	9.38%	2.98%	-148.36%
	Meta fair algorithm	-2.98%	-7.25%	-7.49%	-20.88%	-108.17%
Post-processing	Reject option classification	-8.64%	-30.71%	**74.80**%	**74.55**%	-263.51%
	Equalized odds processor	-16.22%	-59.73%	25.83%	-11.08%	-407.82%
	Platt scaling	-0.45%	-26.98%	-85.28%	-108.45%	-85.02%
Average change across fairness processors		-3.81%	-22.06%	12.51%	1.53%	-159.18%

Abbreviations: AUC = area under the ROC curve, IND = independence, SP = separation, SF = sufficiency. Values represent percentage differences relative to an unconstrained model averaged over seven data sets × five folds × four base models; positive values indicate improvement.

also evidences that the unconstrained model suffers from discrimination. Six out of eight processors achieve better independence and five processors attain better separation. However, sufficiency is consistently higher in the unconstrained model, which confirms that this metric differs fundamentally from independence and separation. High agreement between the sufficiency and profit, expressed by strict dominance of the unconstrained model in Table 7.6.2, also indicates that the goal of profit maximization is compatible with maximizing sufficiency, which questions the fairness perspective that the latter embodies.

Considering individual processors, the reject option classification post-processor demonstrates the best fairness in independence and separation. This is achieved by sacrificing more than 30% profit compared to the unconstrained model. On the other hand, we observe the least profit decrease of less than 5% for the prejudice remover, which also attains a similar AUC as the unconstrained model. At the same time, the prejudice remover provides a smaller fairness improvement than other processors. These results emphasize the trade-off between profit and fairness.

Comparing processors within the implementation methods, we can identify promising techniques. Considering post-processors, the equalized odds processor is dominated by reject option classification in all evaluation measures. Platt scaling achieves higher profit and sufficiency than the latter but gives the by far worst results in independence and separation. In sum, Table 7.6.2 clearly identifies reject option classification as the most suitable post-processor. Concerning pre-processors, no clear result emerges. Reweighting achieves the best fairness but decreases profitability by 23%. The disparate impact remover retains a higher share of profit but offers substantially smaller improvements in independence and separation.

Among the in-processors, we observe the unconstrained model to dominate the meta fair algorithm, which displays negative results for all metrics of Table 7.6.2. Therefore, the meta

fair algorithm does not warrant further consideration. Comparing the prejudice remover to adversarial debiasing, we find the former to deliver better results in all metrics but sufficiency. Given reservations against the fairness concept of the sufficiency metric, the results of Table 7.6.2 suggest that the prejudice remover is the best performing in-processor.

The results of Table 7.6.2 have several implications. First, we identify two fairness processors, Platt scaling and the meta-fair algorithm, inadequate for credit scoring since they decrease profit and predictive performance while not improving fairness compared to the unconstrained model. Second, we find that the equalized odds processor is dominated by another post-processor in all considered evaluation metrics and should, therefore, be avoided.

The remaining processors arrive at different solutions in the space between sacrificing profit and reducing discrimination, leaving decision-makers with the difficult task to balance these conflicting goals according to their preferences, business requirements, and regulation. In general, in-processors offer more flexibility in prioritizing fairness or profit through meta-parameters. For example, the prejudice remover incorporates a regularizer to penalize fairness violations and exposes the weight of that penalty as a meta-parameter. However, the benefit of higher flexibility carries a cost. Compared to alternative options, in-processors replace existing scorecards and impact the scoring process the most. Post-processors largely retain an existing scoring pipeline, which simplifies their deployment. Pre-processors address fairness at the data level, which represents a more invasive change of the scoring process compared to post-processing but seems less difficult to implement than in-processing. Together with the results of Table 7.6.2, in which the best in-processor (i.e., the prejudice remover) finds a better trade-off between profit and fairness than the disparate impact remover while the best post-processor (i.e., reject option classification) increases fairness to a larger extent than reweighting, considerations related to the complexity of deploying fairness processors and revising loan approval processes suggest two options for addressing fairness in credit scoring. Decision-makers can choose between a flexible but invasive in-processor and a post-processor, which is easier to deploy but might substantially decrease profitability. Table 7.6.2 represents the corresponding options by the prejudice remover and reject option classification.

7.6.3 The Cost of Fairness

Previous results indicate that it is possible to improve fairness by sacrificing profit. Figure 7.6.1 provides a more detailed examination of the profit-fairness trade-off on each of the seven data sets using the concept of Pareto frontiers. The points on the frontiers refer to the test set performance of fairness processors trained with different base classifiers and on different combinations of the training folds. The frontiers only contain the non-dominated solutions, i.e., the points where it is impossible to improve on one objective (i.e., profit) without harming the other objective (i.e., fairness). Based on the previous results, we use

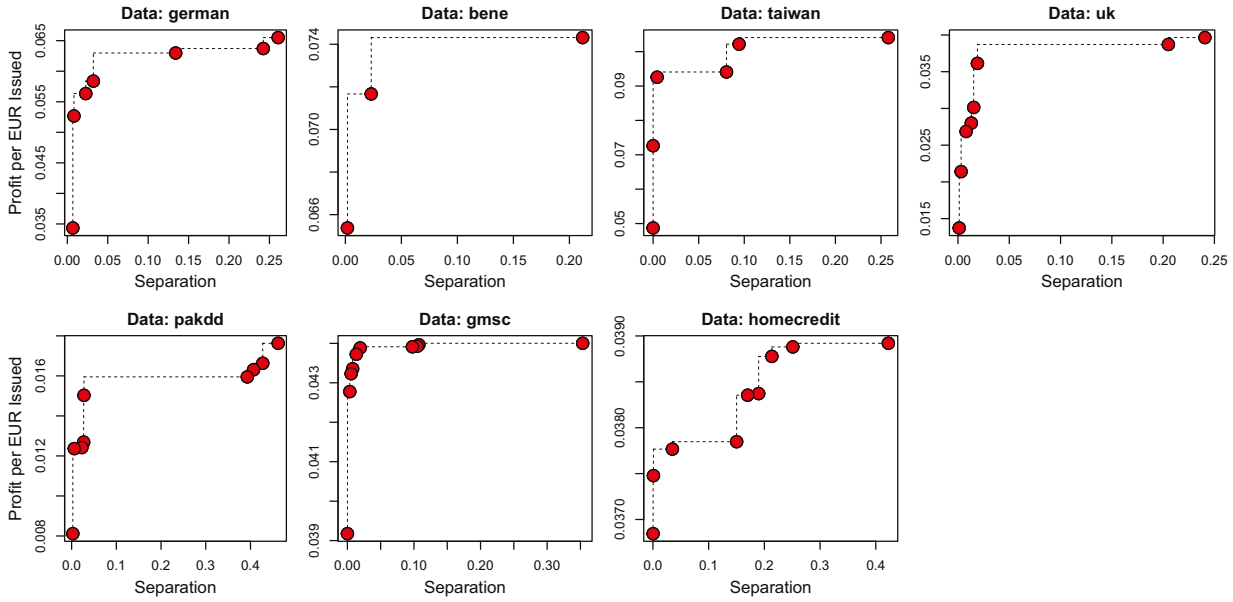

Figure 7.6.1. Profit-Fairness Trade-Off: Frontiers with Non-Dominated Solutions

the separation criterion to measure fairness.

Figure 7.6.1 reveals that discrimination can be substantially reduced at a relatively low cost. Recall that separation indicates the difference between the false positive and false negative rates across the sensitive groups. According to Figure 7.6.1, reducing the difference in error rates below 0.2 is possible while sacrificing less than €0.01 profit per EUR issued. Across the data sets, this translates to an average profit reduction of 4.91% compared to the most profitable scorecard with stronger discrimination. At the same time, completely eliminating unfairness is more costly: achieving separation of 0 is only possible when sacrificing more than 35% of the profit. However, since perfect fairness is not required by regulation, we conclude that a financial institution can reduce discrimination to a reasonable extent while maintaining a relatively high profit margin.

7.7 Conclusion

The paper sets out to consolidate recent advancements in fair ML from a credit scoring perspective. Cataloging approaches for quantifying fairness and the ML pipeline interventions for fairness maximization, we have examined the adequacy of these fairness measures and processors for credit scoring. To substantiate our conceptual analysis, we have undertaken a systematic empirical comparison of several fairness processors from different families to identify preferable approaches and clarify the degree to which increasing fairness in loan approval processes harms profitability.

The conceptual comparison of different fairness criteria reveals separation to be the most appropriate metric for credit scoring. Separation acknowledges the imbalanced misclassification costs, which are instrumental to the lending business. The presented catalog of fairness

processors offers practitioners a starting point for deciding which processors to consider for a given problem setting. The catalog also indicates that most processors have been evaluated based on their accuracy and that some relevant credit scoring scenarios are not well covered by the available processors. For example, in a setting with multiple sensitive attributes (e.g., race and religion), only two processors, adversarial debiasing and reject option classification, facilitate optimizing the separation criterion.

The empirical study benchmarks fairness processors in a profit-oriented credit scoring setup. Several implications emerge from the results. First, examining the agreement between the fairness criteria under study reveals that separation and independence are strongly correlated. While other empirical studies support this finding [20], it contradicts the intuition from theoretical considerations that fairness criteria are mutually exclusive [36]. We also find that sufficiency has a property to be achievable by any well-trained classifier that can predict the sensitive attribute from the other features [2]. This calls into question the overall suitability of sufficiency for credit scoring and further emphasizes separation as a proper criterion for measuring the fairness of credit scorecards.

Second, we find that the choice of an appropriate fairness processor depends on the implementation feasibility and preferences of a decision-maker regarding the conflicting objectives of profit and fairness. Post-processing methods such as reject option classification are the easiest to implement in production but improve fairness at a high monetary cost. In-processors such as the prejudice remover perform best in finding the profit-fairness trade-off and offer the most flexibility in calibrating the importance of the conflicting objectives. However, using in-processors requires replacing a deployed scoring model with a new algorithm, which might require regulatory approval and is associated with considerable efforts.

Third, while achieving perfect fairness is costly, we find that reducing discrimination to a reasonable extent is possible while maintaining a relatively high profit. These results support the current anti-discrimination regulation that allows unfairness to exist up to a certain limited extent. The analysis of fairness processors from the perspective of the Pareto frontiers offers decision-makers a tool to analyze the profit-fairness trade-off specific to their context and identify techniques that reduce discrimination to a required level at the smallest monetary cost.

Our study may also have implications for customer scoring models beyond the credit industry. Fairness concerns arise from the increasing use of ML to automate decisions in many domains, such as hiring [2], college admission [36] or criminal risk assessment [4]. The catalog of fairness processors and the results of their empirical analysis can aid these domains in identifying suitable techniques for integrating fairness in decision support systems. Future work on fair ML may also draw value from the empirical comparison in that it highlights effective approaches that set a benchmark for new fairness processors.

7.8 Appendix

7.8.1 Overview of Fairness Criteria

As shown in [2], the fairness criteria considered in the paper – independence, separation and sufficiency – comprise a number of other fairness criteria proposed in prior work. This appendix illustrates the relationship between the three considered criteria and related fairness formulations.

Table 7.8.1 reveals that the statistical formulation of fairness constraints originates from the field of psychological testing [14] and has been rediscovered for ML applications much later. The 19 fairness concepts presented in the table can be derived from independence, separation and sufficiency in the form of an equivalent or a relaxed condition. This underpins the relevance of the three fairness criteria and justifies our criteria selection in the focal paper.

It is important to note that all fairness criteria of Table 7.8.1 and the paper as a whole embody the idea of group-based fairness. Prior literature has introduced alternative fairness concepts including individual and counterfactual fairness. The former requires a classifier to produce similar outputs for similar individuals, whereas the latter implies that a classifier output remains the same when the sensitive attribute is changed to its counterfactual value.

Table 7.8.1. Fairness Criteria and their Relation to Independence, Separation, Sufficiency

Reference	Criterion	Closest relative	Relation
Darlington [14]	Darlington criterion (4)	Independence	Equivalent
Dwork et al. [15]	Statistical parity	Independence	Equivalent
Dwork et al. [15]	Group fairness	Independence	Equivalent
Dwork et al. [15]	Demographic parity	Independence	Equivalent
Corbett-Davies et al. [12]	Conditional statistical parity	Independence	Relaxation
Darlington [14]	Darlington criterion (3)	Separation	Relaxation
Hardt et al. [24]	Equal opportunity	Separation	Relaxation
Hardt et al. [24]	Equalized odds	Separation	Equivalent
Kleinberg et al. [30]	Balance for the negative class	Separation	Relaxation
Kleinberg et al. [30]	Balance for the positive class	Separation	Relaxation
Zafar et al. [45]	Avoiding disparate mistreatment	Separation	Equivalent
Chouldechova [10]	Predictive equality	Separation	Relaxation
Woodworth et al. [43]	Equalized correlations	Separation	Relaxation
Berk et al. [4]	Conditional procedure accuracy	Separation	Equivalent
Cleary [11]	Cleary model	Sufficiency	Equivalent
Darlington [14]	Darlington criterion (1), (2)	Sufficiency	Relaxation
Chouldechova [10]	Predictive parity	Sufficiency	Relaxation
Chouldechova [10]	Calibration within groups	Sufficiency	Equivalent
Berk et al. [4]	Conditional use accuracy	Sufficiency	Equivalent

7.8.2 Meta-Parameters of Base Models and Fairness Processors

This appendix provides meta-parameter values of the base classifiers and the fairness processors used in the empirical experiment. Table 7.8.3 depicts the candidate values of the meta-parameters of the four base classifiers used as a scoring model by fairness pre- and post-processors as well as by the unconstrained profit maximization benchmark. The meta-parameter values are optimized with grid search using the EMP as an objective. The meta-parameter tuning is performed separately on each combination of the training folds using a nested four-fold cross-validation.

Table 7.8.2 provides candidate values of the meta-parameters of fairness processors that are tuned within the higher-level cross-validation framework. We measure the EMP of fairness processors on each validation fold to select the appropriate meta-parameter values. The notation for processor meta-parameters and their explanation is available in Section 4.

7.8.3 Extended Empirical Results

This appendix provides additional results of the experiment presented in Section 6. Tables 7.8.4 – 7.8.9 compare the performance of fairness processors as well as an unconstrained scorecard on each of the seven credit scoring data sets in terms of the AUC, profit per EUR issued and fairness. Performance of pre- and post-processors is averaged over 25 values from five cross-validation folds × five base classifiers; performance of in-processors is aggregated over five training fold combinations.

Table 7.8.2. Meta-Parameters of Fairness Processors

Method	Fairness processor	Meta-parameter	Candidate values
Pre-processing	Reweighting	–	–
	Disparate impact remover	Repair level λ	$.5, .6, .7, .8, .9, 1$
In-processing	Prejudice remover	Fairness penalty η	$1, 5, 15, 30, 50, 100, 150$
	Meta fair algorithm	Fairness penalty τ	$.05, .10, .15, .20, .25, .30$
	Adversarial debiasing	Adversarial loss weight α	$.1, .01, .001$
		Number of epochs	50
		Batch size	128
Post-processing	Reject option classification	Fairness bound $[\sigma_l, \sigma_u]$	[-.1, .1], [-.2, .2], [-.3, .3]
		Number of thresholds	100
		Number of ROC margins	50
	Equalized odds processor	–	–
	Platt scaling	–	–

Table 7.8.3. Meta-Parameters of Base Classifiers

Base classifier	Meta-parameter	Candidate values
Logistic regression	–	–
Random forest	Number of trees	500
	Number of sampled features	$5, 10, 15$
Extreme gradient boosting	Number of trees	$100, 500, 1000$
	Maximum tree depth	$5, 10$
	Learning rate	0.1
	Ratio of sampled features	$0.5, 1$
	Ratio of sampled cases	$0.5, 1$
	Minimum child weight	$0.5, 1, 3$
Artificial neural network	Size	$5, 10, 15$
	Decay	$0.1, 0.5, 1, 1.5, 2$
	Maximum umber of iterations	1000

Table 7.8.4. Performance of Fairness Processors: German

Method	Fairness processor	AUC	Profit	AR	IND	SP	SF
Pre-processing	Reweighting	.7604	.0252	.6113	.2204	.1752	.1563
	Disparate impact remover	.8121	.0494	.6172	.2989	.1919	.1249
In-processing	Prejudice remover	.7933	.0463	.6112	.3200	.2091	.1655
	Adversarial debiasing	.7965	.0502	.6103	.2528	.1898	.1705
	Meta fair algorithm	.8074	.0467	.6158	.2262	.1117	.1555
Post-processing	Reject option classification	.7124	.0254	.5985	.1105	.0881	.2121
	Equalized odds processor	.6999	.0300	.5965	.0836	.1475	.2514
	Platt scaling	.8012	.0464	.6139	.4195	.3369	.1532
Unconstrained profit maximization		.8124	.0492	.6143	.3078	.1979	.1445

Abbreviations: PRE = pre-processor, IN = in-processor, POST = post-processor; AUC = area under the ROC curve, AR = acceptance rate, IND = independence, SP = separation, SF = sufficiency. Performance is averaged over five cross-validation folds × four base models.

Table 7.8.5. Performance of Fairness Processors: Bene

Method	Fairness processor	AUC	Profit	AR	IND	SP	SF
Pre-processing	Reweighting	.7469	.0524	.6108	.0934	.0777	.0487
	Disparate impact remover	.7875	.0638	.6138	.3622	.2832	.0694
In-processing	Prejudice remover	.7952	.0702	.6194	.3141	.2284	.1615
	Adversarial debiasing	.7813	.0670	.6118	.3250	.2450	.1393
	Meta fair algorithm	.7875	.0653	.6143	.3227	.2446	.0980
Post-processing	Reject option classification	.7082	.0501	.6037	.0726	.0711	.2339
	Equalized odds processor	.6677	.0491	.6039	.0396	.0844	.2485
	Platt scaling	.7880	.0659	.6158	.4469	.3684	.0681
Unconstrained profit maximization		.7896	.0659	.6152	.3540	.2743	.0825

Abbreviations: PRE = pre-processor, IN = in-processor, POST = post-processor; AUC = area under the ROC curve, AR = acceptance rate, IND = independence, SP = separation, SF = sufficiency. Performance is averaged over five cross-validation folds × four base models.

Table 7.8.6. Performance of Fairness Processors: Taiwan

Method	Fairness processor	AUC	Profit	AR	IND	SP	SF
Pre-processing	Reweighting	.7605	.0725	.5954	.0441	.0406	.0189
	Disparate impact remover	.7909	.0769	.5999	.1781	.1324	.0216
In-processing	Prejudice remover	.7867	.0882	.5992	.1170	.0876	.0311
	Adversarial debiasing	.7918	.0892	.5987	.2762	.2262	.0215
	Meta fair algorithm	.7893	.0880	.6009	.1212	.0854	.0197
Post-processing	Reject option classification	.7080	.0515	.5869	.0514	.0328	.0505
	Equalized odds processor	.6231	-.0081	.5797	.1902	.1915	.0618
	Platt scaling	.7565	.0294	.5960	.2738	.2187	.0159
Unconstrained profit maximization		.7532	.0643	.5956	.1211	.0872	.0278

Abbreviations: PRE = pre-processor, IN = in-processor, POST = post-processor; AUC = area under the ROC curve, AR = acceptance rate, IND = independence, SP = separation, SF = sufficiency. Performance is averaged over five cross-validation folds × four base models.

Table 7.8.7. Performance of Fairness Processors: UK

Method	Fairness processor	AUC	Profit	AR	IND	SP	SF
Pre-processing	Reweighting	.6786	.0165	.5544	.0807	.0396	.0056
	Disparate impact remover	.7174	.0131	.5543	.2926	.2051	.0113
In-processing	Prejudice remover	.7087	.0187	.5543	.3033	.2433	.0191
	Adversarial debiasing	.7092	.0181	.5543	.2614	.1622	.0181
	Meta fair algorithm	.5584	.0038	.5542	.3389	.4128	.0021
Post-processing	Reject option classification	.6523	.0181	.5542	.0621	.0222	.0162
	Equalized odds processor	.6206	.0189	.5542	.1783	.2042	.0225
	Platt scaling	.6986	.0244	.5543	.6839	.5329	.0200
Unconstrained profit maximization		.7129	.0180	.5543	.3111	.2141	.0141

Abbreviations: PRE = pre-processor, IN = in-processor, POST = post-processor; AUC = area under the ROC curve, AR = acceptance rate, IND = independence, SP = separation, SF = sufficiency. Performance is averaged over five cross-validation folds × four base models.

Table 7.8.8. Performance of Fairness Processors: PAKDD

Method	Fairness processor	AUC	Profit	AR	IND	SP	SF
Pre-processing	Reweighting	.5783	.0078	.5836	.0710	.0685	.0198
	Disparate impact remover	.6022	.0134	.5840	.4126	.3862	.0818
In-processing	Prejudice remover	.6003	.0134	.5839	.2829	.2506	.1171
	Adversarial debiasing	.5777	.0079	.5835	.1864	.1686	.0952
	Meta fair algorithm	.6027	.0136	.5839	.3383	.3070	.1138
Post-processing	Reject option classification	.5677	.0080	.5834	.0822	.0602	.1153
	Equalized odds processor	.5653	.0112	.5833	.0173	.0394	.1214
	Platt scaling	.6053	.0144	.5840	.6249	.5920	.1044
Unconstrained profit maximization		.6045	.0139	.5840	.4347	.4069	.0829

Abbreviations: PRE = pre-processor, IN = in-processor, POST = post-processor; AUC = area under the ROC curve, AR = acceptance rate, IND = independence, SP = separation, SF = sufficiency. Performance is averaged over five cross-validation folds × four base models.

Method	Fairness processor	AUC	Profit	AR	IND	SP	SF
Pre-processing	Reweighting	.8425	.0415	.5589	.0595	.0437	.0055
	Disparate impact remover	.8535	.0419	.5593	.1935	.1077	.0151
In-processing	Prejudice remover	.8553	.0437	.5593	.2454	.1445	.0085
	Adversarial debiasing	.8588	.0438	.5594	.1126	.0508	.0154
	Meta fair algorithm	.8261	.0418	.5593	.4318	.2766	.0182
Post-processing	Reject option classification	.7762	.0388	.5576	.0590	.0525	.0187
	Equalized odds processor	.5903	.0234	.5568	.3844	.2812	.0240
	Platt scaling	.8531	.0424	.5594	.5691	.3564	.0000
Unconstrained profit maximization		.8545	.0406	.5594	.2461	.1429	.0121

Abbreviations: PRE = pre-processor, IN = in-processor, POST = post-processor; AUC = area under the ROC curve, AR = acceptance rate, IND = independence, SP = separation, SF = sufficiency. Performance is averaged over five cross-validation folds × four base models.

Table 7.8.9. Performance of Fairness Processors: Homecredit

Method	Fairness processor	AUC	Profit	AR	IND	SP	SF
Pre-processing	Reweighting	.7275	.0353	.5589	.1225	.0958	.0056
	Disparate impact remover	.7392	.0361	.5590	.2784	.2010	.0167
In-processing	Prejudice remover	.7387	.0372	.5589	.2072	.1356	.0238
	Adversarial debiasing	.7379	.0371	.5589	.3170	.2464	.0169
	Meta fair algorithm	.7351	.0367	.5588	.3482	.2661	.0058
Post-processing	Reject option classification	.6785	.0350	.5585	.0506	.0207	.0290
	Equalized odds processor	.6190	.0260	.5583	.2481	.2482	.0426
	Platt scaling	.7406	.0371	.5590	.4884	.3643	.0140
Unconstrained profit maximization		.7411	.0367	.5590	.33044	.2435	.0130

Abbreviations: PRE = pre-processor, IN = in-processor, POST = post-processor; AUC = area under the ROC curve, AR = acceptance rate, IND = independence, SP = separation, SF = sufficiency. Performance is averaged over five cross-validation folds × four base models.

Bibliography

[1] Banasik, J., Crook, J. (2007). Reject inference, augmentation, and sample selection. *European Journal of Operational Research*, *183*, 1582–1594.

[2] Barocas, S., Hardt, M., Narayanan, A. (2019). *Fairness and Machine Learning*. fairmlbook.org.

[3] Barocas, S., Selbst, A. D. (2016). Big data's disparate impact. *California Law Review*, *104*, 671–732.

[4] Berk, R., Heidari, H., Jabbari, S., Kearns, M., Roth, A. (2021). Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, *50*, 3–44.

[5] Calders, T., Kamiran, F., Pechenizkiy, M. (2009). Building classifiers with independency constraints. *Proc. IEEE International Conference on Data Mining Workshops*, 13–18.

[6] Calders, T., Verwer, S. (2010). Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, *21*, 277–292.

[7] Calmon, F., Wei, D., Vinzamuri, B., Natesan Ramamurthy, K., Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. *Advances in Neural Information Processing Systems*, 3992–4001.

[8] Celis, L. E., Huang, L., Keswani, V., Vishnoi, N. K. (2019). Classification with fairness constraints: A meta-algorithm with provable guarantees. *Proc. Conference on Fairness, Accountability, and Transparency*, 319–328.

[9] Chen, T., Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

[10] Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, *5*, 153–163.

[11] Cleary, T. A. (1968). Test bias: Prediction of grades of negro and white students in integrated colleges. *Journal of Educational Measurement*, *5*, 115–124.

[12] Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., Huq, A. (2017). Algorithmic decision making and the cost of fairness. *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 797–806.

[13] Crook, J. N., Edelman, D. B., Thomas, L. C. (2007). Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, *183*, 1447–1465.

[14] Darlington, R. B. (1971). Another look at “cultural fairness". *Journal of Educational Measurement*, *8*, 71–82.

[15] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R. (2012). Fairness through awareness. *Proc. Innovations in Theoretical Computer Science Conference*, 214–226.

[16] Equal Credit Opportunity Act (1974). Art. 9 & 15 U.S. code §1691. URL: `https://www.law.cornell.edu/uscode/text/15/1691c`. Accessed 1 June 2021.

[17] European Commission (2017). Guidelines on data protection officers. URL: `https://ec.europa.eu/newsroom/article29/items/612048`. Accessed 1 June 2021.

[18] Executive Office of the President (2016). Big data: A report on algorithmic systems, opportunity,and civil rights. URL: `https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/2016_0504_data_discrimination.pdf`. Accessed 1 June 2021.

[19] Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., Venkatasubramanian, S. (2015). Certifying and removing disparate impact. *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 259–268.

[20] Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P., Roth, D. (2019). A comparative study of fairness-enhancing interventions in machine learning. *Proc. Conference on Fairness, Accountability, and Transparency*, 329–338.

[21] Fuster, A., Goldsmith-Pinkham, P., Ramadorai, T., Walther, A. (2017). *Predictably unequal? The effects of machine learning on credit markets*. Technical Report National Bureau of Economic Research.

[22] Goh, G., Cotter, A., Gupta, M., Friedlander, M. P. (2016). Satisfying real-world goals with dataset constraints. *Advances in Neural Information Processing Systems*, 2415–2423.

[23] Gunnarsson, B. R., Vanden Broucke, S., Baesens, B., Óskarsdóttir, M., Lemahieu, W. (2021). Deep learning for credit scoring: Do or don't? *European Journal of Operational Research*. DOI: 10.1016/j.ejor.2021.03.006

[24] Hardt, M., Price, E., Srebro, N. (2016). Equality of opportunity in supervised learning. *Advances in Neural Information Processing Systems*, 3315–3323.

[25] Johndrow, J. E., Lum, K. et al. (2019). An algorithm for removing sensitive information: application to race-independent recidivism prediction. *The Annals of Applied Statistics*, *13*, 189–220.

[26] Kamiran, F., Calders, T. (2009). Classifying without discriminating. *Proc. International Conference on Computer, Control and Communication*, 1–6.

[27] Kamiran, F., Karim, A., Zhang, X. (2012). Decision theory for discrimination-aware classification. *Proc. International Conference on Data Mining*, 924–929.

[28] Kamishima, T., Akaho, S., Asoh, H., Sakuma, J. (2012). Fairness-aware classifier with prejudice remover regularizer. *Proc. Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 35–50.

[29] Kingma, D. P., Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[30] Kleinberg, J., Mullainathan, S., Raghavan, M. (2017). Inherent trade-offs in the fair determination of risk scores. *Proc. 8th Innovations in Theoretical Computer Science Conference*, 43:1–43:23.

[31] Kozodoi, N., Lessmann, S., Papakonstantinou, K., Gatsoulis, Y., Baesens, B. (2019). A multi-objective approach for profit-driven feature selection in credit scoring. *Decision Support Systems*, *120*, 106–117.

[32] Lessmann, S., Baesens, B., Seow, H.-V., Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*, 124–136.

[33] Liu, L. T., Dean, S., Rolf, E., Simchowitz, M., Hardt, M. (2018). Delayed impact of fair machine learning. *Proc. International Conference on Machine Learning*, 3150–3158.

[34] Louizos, C., Swersky, K., Li, Y., Welling, M., Zemel, R. (2016). The variational fair autoencoder. *Proc. International Conference on Learning Representations*.

[35] Luong, B. T., Ruggieri, S., Turini, F. (2011). K-NN as an implementation of situation testing for discrimination discovery and prevention. *Proc. ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, 502–510.

[36] Mitchell, S., Potash, E., Barocas, S., D'Amour, A., Lum, K. (2021). Algorithmic fairness: Choices, assumptions, and definitions. *Annual Review of Statistics and Its Application*, *8*, 141–164.

[37] Narayanan, A. (2018). Translation tutorial: 21 fairness definitions and their politics. *Proc. Conference on Fairness, Accountability, and Transparency*.

[38] Niculescu-Mizil, A., Caruana, R. (2005). Predicting good probabilities with supervised learning. *Proc. International Conference on Machine Learning*, 625–632.

[39] Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, *10*, 61–74.

[40] Pleiss, G., Raghavan, M., Wu, F., Kleinberg, J., Weinberger, K. Q. (2017). On fairness and calibration. *Advances in Neural Information Processing Systems*, 5680–5689.

[41] Somers, M., Whittaker, J. (2007). Quantile regression for modelling distributions of profit and loss. *European Journal of Operational Research*, *183*, 1477–1487.

[42] Verbraken, T., Bravo, C., Weber, R., Baesens, B. (2014). Development and application of consumer credit scoring models using profit-based classification measures. *European Journal of Operational Research*, *238*, 505–513.

[43] Woodworth, B., Gunasekar, S., Ohannessian, M. I., Srebro, N. (2017). Learning non-discriminatory predictors. *Proc. Conference on Learning Theory* (pp. 1920–1953.

[44] Zadrozny, B., Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. *Proc. International Conference on Machine Learning*, 609–616.

[45] Zafar, M. B., Valera, I., Gomez Rodriguez, M., Gummadi, K. P. (2017). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. *Proc. International Conference on World Wide Web*, 1171–1180.

[46] Zafar, M. B., Valera, I., Gomez Rodriguez, M., Gummadi, K. P. (2017). Fairness constraints: Mechanisms for fair classification. *Proc. International Conference on Artificial Intelligence and Statistics*, 962–970.

[47] Zemel, R., Wu, Y., Swersky, K., Pitassi, T., Dwork, C. (2013). Learning fair representations. *Proc. International Conference on Machine Learning*, 325–333.

[48] Zhang, B. H., Lemoine, B., Mitchell, M. (2018). Mitigating unwanted biases with adversarial learning. *Proc. AAAI/ACM Conference on AI, Ethics, and Society*, 335–340.