9-27-2023

Anomaly Detection in Large Datasets: A Case Study in Loan Defaults

Rayhaan Pirani
University of Windsor

Recommended Citation

Pirani, Rayhaan, "Anomaly Detection in Large Datasets: A Case Study in Loan Defaults" (2023). *Electronic Theses and Dissertations*. 9253.
https://scholar.uwindsor.ca/etd/9253

Anomaly Detection in Large Datasets: A Case Study in Loan Defaults

By

Rayhaan Pirani

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2023

Anomaly Detection in Large Datasets: A Case Study in Loan Defaults

by

Rayhaan Pirani

APPROVED BY:

August 24, 2023

DECLARATION OF CO-AUTHORSHIP AND PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this thesis incorporates material that is the result of joint research, as follows:

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes an original paper that has been previously accepted in a peer-reviewed conference, as follows:

Publication title/full citation	**Publication Status**
R. Pirani and Z. Kobti. A Novel System Architecture for Anomaly Detection for Loan Defaults. In: Distributed Computing and Artificial Intelligence, 20th International Conference on Distributed Computing and Artificial Intelligence, DCAI 2023. Springer Nature Switzerland AG.	Published (Appears in the proceedings of the DCAI conference, 12-14 July 2023)

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Given the rise in loan defaults, especially after the COVID-19 pandemic, it is necessary to predict if customers might default on a loan for risk management. This thesis proposes an early warning system architecture using anomaly detection based on the unbalanced nature of loan default data in the real world. Most customers do not default on their loans; only a tiny percentage do, resulting in an unbalanced dataset. We aim to evaluate potential anomaly detection methods for their suitability in handling unbalanced datasets. We conduct a comparative study on different anomaly detection approaches on four balanced and unbalanced datasets.

We compare five of each supervised, unsupervised, and semi-supervised anomaly detection approaches. The supervised algorithms compared are logistic regression, stochastic gradient descent (SGD), XGBoost, LightGBM, and CatBoost classification methods. The unsupervised anomaly detection methods are isolation forest, angle-based outlier detection (ABOD), outlier detection using empirical cumulative distribution function (ECOD), copula-based outlier detection (COPOD), and deep one-class classifier with autoencoder (DeepSVDD). The semi-supervised anomaly detection methods are improving supervised outlier detection with unsupervised representation learning (XGBOD), feature encoding with autoencoders for weakly-supervised anomaly detection (FeaWAD), deep semi-supervised anomaly detection (DeepSAD), progressive image deraining networks (PReNet), and deep anomaly detection with deviation networks (DevNet).

We compare them using standard evaluation metrics such as accuracy, precision, recall, F1 score, training and prediction time, and area under the receiver operating characteristic (ROC) curve. The results show that anomaly detection methods perform significantly better on unbalanced loan default data and are more suitable for real-world applications. The results also show that supervised methods work better for balanced datasets, and for peer-to-peer lending datasets, boosting approaches are expected to perform well.

DEDICATION

I dedicate this thesis to the memory of my beloved late father, Murad Ali, whose unwavering encouragement and support have been instrumental in shaping my aspirations and driving me toward pursuing my dreams.

To my dear mother, Zohra, I am forever grateful for instilling in me the values of resilience and perseverance. Your unwavering dedication and constant reminder to advocate for me have inspired and strengthened me.

I extend my heartfelt dedication to my paternal aunt, Habiba, whose nurturing presence and unconditional love felt like one from a second mother. Your unwavering support and care have been a source of immense comfort and encouragement. I am forever grateful for your presence in my life.

To my younger brother, Safzan, you have been a constant source of love, acceptance, and understanding. Your unwavering support and unconditional love have given me the courage to overcome challenges and pursue my academic goals. I am privileged to have you as a cherished sibling and confidant.

Finally, I dedicate this thesis to my dear friends, who have become my chosen family on this journey of life. Your love, support, and companionship have been invaluable throughout my academic endeavours. The countless moments of encouragement, laughter, and shared experiences have shaped me into who I am today. Thank you for standing by me and being essential to my life.

I honour and dedicate this thesis to all these incredible individuals who have played significant roles in shaping my path, offering unwavering support, and enriching my life with love and guidance. Their influence and presence have propelled me forward and made this academic achievement possible.

ACKNOWLEDGEMENTS

I would first acknowledge the invaluable support of my supervisor Dr. Ziad Kobti during my master's journey. I cannot thank him enough for providing ongoing input, direction and encouragement for my thesis paper and his help correcting my presentation and texts, ultimately leading to my acceptance of my work at a reputed conference. Thanks to him, I have learned much during this process and become much more confident in my abilities.

I would also recognize the contributions of my thesis committee members, Dr. Alioune Ngom and Dr. Bharat Maheshwari, for their consistent guidance, understanding and support during the entire process of writing and presenting my thesis work. I am incredibly grateful for the time and attention they have invested in me to ensure my success.

I acknowledge the knowledge my university course professors Dr. Dan Wu, Dr. Robin Gras, Dr. Alioune Ngom, and Dr. Hossein Fani, have imparted to me that helped me shape my projects, papers, and thesis. I thank Mrs. Christine Weisener, the graduate secretary, and Mrs. Melissa Robinet, the program secretary of the School of Computer Science, for their continuous support and assistance throughout my program.

I also thank my classmate Ehsan Ur Rahman Mohammed for co-authoring my first paper publication and being a long-term course project partner. I also thank my other classmates, Surajsinh Prakashchandra Parmar and Ahmed Shafeek Abouhassan, for collaborating with me and delivering high-quality projects during my learning journey at the university.

Finally, I acknowledge my supervisors, Mr. Rick Harrop and Mr. Wagner Ksenhuk and their teams at my previous co-op employment for giving me a chance to be a part of their work, for their guidance and support, and for being a significant part of my education.

TABLE OF CONTENTS

DECLARATION OF CO-AUTHORSHIP AND PREVIOUS PUBLICATION **iii**

ABSTRACT **v**

DEDICATION **vi**

ACKNOWLEDGEMENTS **vii**

LIST OF TABLES **x**

LIST OF FIGURES **xi**

LIST OF ABBREVIATIONS **xii**

1 Introduction **1**
- 1.1 Background . . . 2
  - 1.1.1 Anomaly Detection . . . 2
  - 1.1.2 Early Warning System . . . 3
- 1.2 Problem Definition . . . 5
- 1.3 Thesis Motivation . . . 6
- 1.4 Thesis Statement . . . 7
- 1.5 Thesis Contribution . . . 8
- 1.6 Thesis Organization . . . 10

2 Related Works **11**
- 2.1 Existing Relevant Literature . . . 11
- 2.2 Comparative Summary . . . 13

3 Proposed Approach **16**
- 3.1 Introduction . . . 16
- 3.2 Experimental Setup . . . 17
- 3.3 Early Warning System Architecture . . . 18

4 Experimental Evaluation **22**
- 4.1 Libraries and Tools . . . 22
- 4.2 System Configuration . . . 23
- 4.3 Algorithms . . . 23
  - 4.3.1 Supervised Binary Classification Methods . . . 24
  - 4.3.2 Unsupervised Methods for Anomalous Data Distribution . . . 29
  - 4.3.3 Semi-supervised Methods with Efficient Label Usage . . . 33
- 4.4 Datasets . . . 39
  - 4.4.1 Bondora Peer-to-Peer Lending Data . . . 40

4.4.2 L&T Vehicle Loan Default Dataset 41
4.4.3 LendingClub Data 42
4.4.4 Deloitte-MachineHack Default Prediction Data 43
4.5 Hyperparameter Tuning 46
4.6 Evaluation Metrics 54
4.6.1 Accuracy 55
4.6.2 Precision 55
4.6.3 Recall 56
4.6.4 F1 Score 56
4.6.5 Area Under Curve 56
4.6.6 Training and Prediction Times 57

5 Results, Discussion, and Analysis 58
5.1 Experiment Results and Discussion 58
5.1.1 Bondora Peer-to-Peer Lending Data 58
5.1.2 L&T Vehicle Loan Default Dataset 61
5.1.3 LendingClub Data 64
5.1.4 Deloitte-MachineHack Default Prediction Data 65
5.2 Analysis of Results and Considerations 69
5.2.1 Balanced versus Unbalanced Datasets 69
5.2.2 Peer-to-Peer versus Other Loans 70
5.2.3 Time Considerations 70
5.2.4 Availability of Class Label in the Data 71

6 Conclusion and Future Work 73
6.1 Conclusion 73
6.2 Limitations and Future Work 74

REFERENCES 76

VITA AUCTORIS 82

LIST OF TABLES

2.2.1 Summary of contributions of related work 13
2.2.2 Contributions and drawbacks of various approaches 14

4.4.1 Summary of the structure of datasets 45
4.5.1 Randomized hyperparameter grid search space for supervised algorithms 50
4.5.2 Randomized hyperparameter grid search space for unsupervised and semi-supervised algorithms 51
4.5.3 Optimal hyperparameter sets across datasets for supervised approaches 52
4.5.4 Optimal hyperparameter sets across datasets for unsupervised approaches 53
4.5.5 Optimal hyperparameter sets across datasets for semi-supervised approaches 53

5.1.1 Evaluation results for all algorithms on the Bondora P2P dataset . . 59
5.1.2 Low and high threshold evaluation results for all algorithms on the Bondora P2P dataset 60
5.1.3 Evaluation results for all algorithms on the L&T Vehicle Loan dataset 62
5.1.4 Low and high threshold evaluation results for all algorithms on the L&T Vehicle Loan dataset 63
5.1.5 Evaluation results for all algorithms on the LendingClub dataset . . . 64
5.1.6 Low and high threshold evaluation results for all algorithms on the LendingClub dataset 66
5.1.7 Evaluation results for all algorithms on the Deloitte-MachineHack dataset 67
5.1.8 Low and high threshold evaluation results for all algorithms on the Deloitte-MachineHack dataset 68

LIST OF FIGURES

1.1.1 Example of anomalies in two-dimensional data 3

3.2.1 Proposed experimental setup for evaluation search 18

3.3.1 Proposed early warning system architecture 21

5.2.1 Comparison of precision and recall for supervised approaches across all datasets 69

5.2.2 Average training and prediction time comparison for all algorithms on the LendingClub dataset 71

LIST OF ABBREVIATIONS

AD	Anomaly Detection
OD	Outlier Detection
EWS	Early Warning System
LR	Logistic Regression
SGD	Stochastic Gradient Descen
XGBoost	Extreme Gradient Boosting
XGB	Extreme Gradient Boosting
LightGBM	Light Gradient Boosting Machine
LGBM	Light Gradient Boosting Machine
CatBoost	Categorical Boosting
CB	Categorical Boosting
IF	Isolation Forest
ABOD	Angle-Based Outlier Detection
ECOD	Empirical-Cumulative-distribution-based Outlier Detection
COPOD	Copula-Based Outlier Detection
DeepSVDD	Deep Support Vector Data Description
XGBOD	eXtreme Gradient Boosting for Outlier Detection
FeaWAD	Feature Encoding with AutoEncoders for Weakly-Supervised Anomaly Detection
DeepSAD	Deep Semi-Supervised Anomaly Detection
PReNet	Pairwise Relation prediction Network

DevNet	Deviation Networks
P2P	Peer-to-Peer
AUC	Area Under Curve
ROC	Receiver Operating Characteristic
Train. time	Training time
Pred. time	Prediction time
TP	True Positives
FN	False Negatives
FP	False Positives
TN	True Negatives
TPR	True Positive Rate
FPR	False Positive Rate

CHAPTER 1

Introduction

Recently, there has been a spike in loan defaults, especially after the COVID-19 pandemic. Fitch Ratings, a finance company providing credit ratings and research for global capital markets, increased the US institutional leveraged loan default rate forecast to 2.0%-3.0% for 2023 and 3.0%-4.0% for 2024 [1]. Nigmonov and Shams [2], in their study to understand how the COVID-19 pandemic has affected lending markets, objectively demonstrate an increase in loan default levels after the pandemic. The probability of default shown in this study before the pandemic is 0.056 and had increased to 0.079 in the post-pandemic period. Canada's largest banks have also allocated a significantly large budget in anticipation of more loan defaults [3]. For Q1 2022, the big six banks in Canada set aside $373 million; for Q1 2023, the allocation was almost $2.5 billion, an increase of over 6.5 times.

To mitigate such loan defaults, it is crucial to have an early warning system to warn banks of potential loan defaults, allowing them to act before the default occurs. An early warning system is a set of capabilities designed to generate and disperse timely and meaningful warning information to enable individuals and organizations to prepare and act appropriately and immediately to reduce harm or loss [4].

In finance, such a system can help financial institutions receive alerts if it is determined that a customer might default on a loan. The institutions can use these alerts to take action to mitigate any potential loan default. Such actions may include offering customers lower interest rates, different payment plans, and something as simple as checking in with the customer to determine if anything can be done to mitigate potential delinquency.

1.1 Background

Loan defaults appear in low probability in the datasets [1, 2]. Consequently, the resulting unbalanced dataset presents challenges when using predictive methods, resulting in low accuracy on default prediction. We propose an early warning system architecture to detect loan defaults by identifying an optimal anomaly detection method for the dataset under study. We test the system on four different datasets to compare various approaches to determine the most optimal approach for this class of problems.

1.1.1 Anomaly Detection

Anomaly detection (AD), sometimes also called outlier detection (OD), is a class of solutions to problems involving identifying data patterns that deviate from expected or normal behaviour. These deviations are commonly referred to as anomalies or outliers. Anomaly detection is crucial in fraud detection, cybersecurity, fault detection, and military surveillance.

Chandola et al. (2008) [5] in their survey paper provide a comprehensive survey of the field of anomaly detection, presenting an overview of different techniques, approaches, and challenges associated with detecting anomalies in various data domains. Anomaly detection, also known as outlier detection, identifies rare and abnormal instances in a dataset that deviate significantly from the norm or expected behaviour. *Anomalies* are observations that do not conform to the majority of the data and can indicate critical events, errors, or fraudulent activities.

Their paper demonstrates an example of anomalies as shown in Fig. 1.1.1. The points in the region N_1 and N_2 represent normal data. Anomalies or outliers are represented by the points o_1, o_2, and all the points in the outlier region O_3.

Simple anomaly detection techniques, such as forming a decision boundary to detect them, may have significant challenges. The boundary between normal and

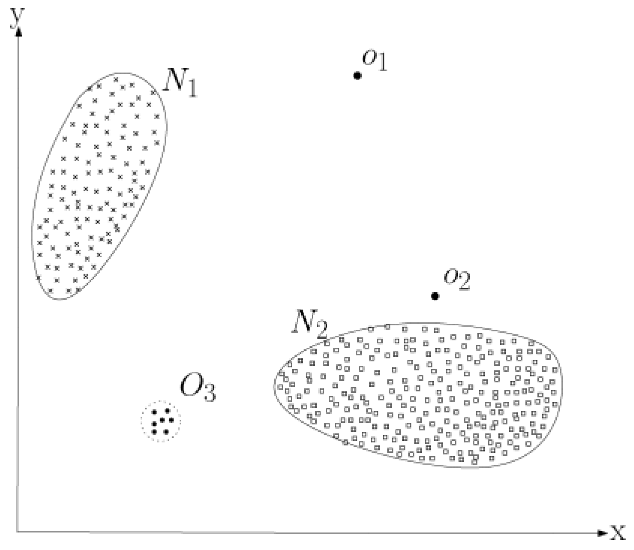

Fig. 1.1.1: Example of anomalies in two-dimensional data

anomalous behaviour may need to be clarified. Defining normal behaviour could be made more difficult by malicious actors who intentionally mask anomalies as normal behaviour. Besides, normal behaviour evolves over time, making establishing a representative notion of normality challenging. Furthermore, different domains have distinct interpretations of anomalies, further complicating the adaptation of techniques across domains. The availability of labelled data for training and validation is often limited, and distinguishing noise from actual anomalies can be complex.

1.1.2 Early Warning System

According to Chaves and Cola (2017) [4], an early warning system (EWS) is a proactive and timely notification mechanism designed to mitigate potential risks and minimize the impact of hazardous events by providing adequate warnings to the public. It is a vital tool for alerting individuals, communities, and relevant stakeholders about imminent threats or emergencies, enabling them to make informed decisions and take appropriate actions to safeguard their well-being.

An early warning system exhibits several key characteristics and requirements:

Timeliness: A fundamental aspect of an early warning system is the prompt delivery of warnings to ensure individuals receive alerts with sufficient lead time to respond effectively. The system should facilitate the rapid dissemination of information to allow for timely decision-making and actions.

Effectiveness: The effectiveness of an early warning system lies in its ability to provide accurate and reliable warnings. It should deliver relevant, actionable, tailored information about the threat or hazard. Warnings must convey the severity and urgency of the situation clearly to encourage appropriate responses.

Inclusiveness: An inclusive early warning system should consider diverse demographics, languages, and communication preferences to ensure accessibility for all individuals and communities. Employing multiple communication channels and formats enhances the system's reach and effectiveness, enabling a broad audience to receive warnings.

Scalability: The system's design should accommodate various scales of emergencies, from localized incidents to regional or national disasters. It should be flexible to handle many warnings, adapt to evolving needs, and expand coverage as the situation unfolds.

Interoperability: Collaboration and interoperability among relevant agencies, organizations, and stakeholders are vital for an effective early warning system. The system should seamlessly integrate information sources, facilitate data sharing, and promote coordination among various entities to enhance overall effectiveness and response capabilities.

Public Awareness: Public awareness and understanding are crucial elements of an early warning system. It should incorporate educational campaigns and training initiatives to familiarize individuals with the warning process, appropriate responses, and the significance of heeding the alerts. Ensuring the public is well-informed empowers them to take proactive measures and enhances the system's effectiveness.

Apart from the financial applications that are discussed in this thesis, early warning systems encompass a broad spectrum of hazards and risks, including natural disasters (e.g., hurricanes, floods, earthquakes), technological emergencies (e.g., industrial accidents, chemical spills), public health emergencies (e.g., disease outbreaks), and security threats (e.g., terrorism). Early warning systems play a pivotal role in these scenarios by providing critical information, instructions, and updates to enable individuals and communities to prepare, respond, and mitigate the impact of these events.

Early warning systems for loan defaults use indicators like declining profitability, increasing debt ratios, or other financial stress signals to provide warnings months in advance. However, there is no fixed time that an early warning system can guarantee universally. An early warning system's effectiveness and lead time are contingent upon the specific application, the quality of data, the model used, and external factors. The primary goal is to provide stakeholders enough time to act, even if that window is not consistently the same for every prediction. In essence, there is inherent uncertainty in predicting future events, and giving a fixed lead time for all situations is challenging.

1.2 Problem Definition

We have a set of loan datasets D and a set of supervised, unsupervised, and semi-supervised anomaly detection approaches A' along with risk threshold values in the set T.

The first subproblem is to generate a clean set of datasets D' from D by preprocessing and feature selection.

The next subproblem is to execute each algorithm in A' over all the datasets in D' and obtain the evaluation metrics, including precision, recall, F1-score, the area under the ROC curve, training time, and prediction time.

The evaluation should be done for different threshold values in T.

Based on the analysis of the evaluation metrics, we would calculate the optimal algorithm A over all datasets. The optimal algorithm A would then be used as the core detection algorithm in the proposed early warning system to warn against potential loan defaults.

1.3 Thesis Motivation

As we will see in Chapter 2, the current research needs specific crucial contexts to evaluate anomaly detection approaches more elaborately for designing an early warning system. The following are the drawbacks of current research.

1. Most current research only tests methods against one type of loan dataset. Banks offer various kinds of loans, and testing against multiple types of loans is essential to evaluate a method holistically.

2. There needs to be more focus on the speed and performance of the algorithm. According to the CIBC Annual Report 2022 [6], CIBC has around 13 million clients; according to RBC Annual Report 2022 [7], they have 17 million. Banks usually have many clients and hence a large amount of data. Therefore, any predictive system must be fast and scalable.

3. Current research focuses on predicting loan defaults but needs to focus specifically on the probabilistic nature of such predictions. An early warning system should have the flexibility to adjust the probabilistic threshold of the predictions depending on the financial institution's risk appetite. Some institutions may be more risk-averse and want alerts even if the possibility of default is less, and vice versa. An excellent early warning system should allow for such flexibility.

4. Most current loan default prediction methods use a classification algorithm to predict defaults. However, real-world data shows that most loan customers do not default. A loan default can thus be treated as a deviation from the norm or an anomaly and not a class of a customer type. Anomaly detection methods can use real-world data directly instead of processing data to balance the number of default and non-default customers, saving much preprocessing and retraining time.

5. It is necessary to compare different approaches and different kinds of approaches to solve a problem. In most papers in the context of loan default prediction,

comparisons are made with only a few approaches and a few evaluation metrics. The evaluation needs to be more comprehensive.

Considering the above factors, the motivation of this thesis is to evaluate different approaches in different contexts and suggests an early warning system architecture to warn against potential loan defaults using anomaly detection.

1.4 Thesis Statement

This thesis aims to develop a fast and scalable machine learning approach for accurately classifying customers who may default. The research proposes a comprehensive investigation of various supervised, unsupervised, and semi-supervised anomaly detection methods using multiple loan default datasets.

The primary objective is to determine the best approach among these anomaly detection techniques to build an ideal system with high accuracy and low prediction times. It would also exhibit a high true-positive rate and a low false-negative rate which complement each other. Maximizing the number of correct default predictions achieves a high true-positive rate. In contrast, a low false-negative rate is achieved by minimizing the number of incorrect non-default predictions.

The system proposed in this research focuses on detecting and reporting warning signs indicative of potential defaults rather than predicting actual defaults. By serving as an early warning system, it can provide crucial insights for risk assessment and intervention strategies.

We aim to develop a system that can work with unbalanced datasets directly. We do not balance an unbalanced dataset synthetically. Balancing data synthetically creates very similar data points, which may result in overfitting. Synthetic data points may result in loss of information and not capture the data distribution. The data might also result in misleading results as the model may perform poorly on real-world datasets, even if it did on synthetically balanced data. Besides, considering that the size of banking data is usually large, using such techniques to balance data would result in an increase in the size of the data, which would require more computational resources and memory.

Given the nature of an early warning system, the prediction threshold for this classification model would be set lower than a model designed to predict certain defaults. This adjustment acknowledges the objective of identifying potential defaults rather than solely predicting definitive occurrences.

To accomplish these objectives, the research will encompass preprocessing and feature engineering techniques to enhance data quality and relevance. The investigation will cover a range of supervised, unsupervised, and semi-supervised anomaly detection algorithms, considering their applicability and effectiveness in detecting potential default cases.

The evaluation and validation of the developed model will involve rigorous performance metrics, including accuracy, precision, recall, and F1-score. Additionally, prediction times will be considered to ensure real-time feasibility for practical implementation.

The outcome of this thesis will be a robust machine learning model capable of efficiently and accurately classifying customers at risk of potential default, providing valuable insights for financial institutions and aiding in proactive decision-making. The research contributes to risk management, loan assessment, and early intervention strategies within the financial domain.

1.5 Thesis Contribution

The primary contribution of this thesis is to demonstrate the effectiveness of anomaly detection as an approach for developing an early warning system to mitigate potential loan defaults on large unbalanced datasets, thereby minimizing financial losses. By harnessing the power of anomaly detection, we aim to achieve desirable characteristics in an early warning system, including versatility, scalability, real-time responsiveness, probabilistic nature, and suitability for handling unbalanced data.

To substantiate our hypothesis, we made the following contributions.

Architecture Development

We designed and developed a comprehensive architecture for an early warning system tailored to identify potential loan defaults. This architecture incorporates state-of-the-art anomaly detection techniques and addresses the challenges posed by large unbalanced datasets. The system's design emphasizes the desirable characteristics, ensuring versatility, scalability, real-time responsiveness, probabilistic decision-making, and the ability to handle unbalanced data effectively.

Experimentation and Evaluation

Using the developed architecture as a foundation, we conducted extensive experiments on multiple datasets representative of loan default scenarios. Various anomaly detection approaches were implemented and evaluated based on relevant metrics, such as accuracy, precision, recall, F1 score, and computational efficiency. These experiments shed light on the performance and effectiveness of different approaches in detecting early warning signs of loan defaults.

Evaluation Search

Building upon the experimentation results, we perform an in-depth evaluation search to identify the most effective approaches in speed and performance. By comparing and analyzing the outcomes of the implemented anomaly detection techniques, we conclude their efficacy and identify the approaches that exhibit the most favourable balance between detection accuracy and computational efficiency.

The results of this study provide valuable insights into the effectiveness of anomaly detection in developing early warning systems for potential loan defaults. Moreover, the findings and methodologies can be extrapolated to other domains, such as power outage alerts, where early detection of abnormal events is crucial for proactive intervention. By contributing to developing effective models for detecting early warning signs, this research aims to enhance risk management strategies and facilitate timely decision-making in the financial sector and beyond.

1.6 Thesis Organization

The thesis is organized as described by the following outline.

Chapter 2 discusses previous literature on solving loan defaults using anomaly detection and early warning systems by comparing state-of-the-art methods and understanding where the current research needs to improve.

In Chapter 3, we describe our proposed early warning system using anomaly detection and the methodology used to assess different anomaly detection approaches to find the most suitable one for this class of problem.

In Chapter 4, we go through the experimental setup, evaluation metrics, and hyperparameter tuning to compare various approaches for the early warning system.

Chapter 5 is an overview of the results obtained from our experimental evaluation and a discussion of the findings conducted on four different loan default benchmark datasets across various supervised, unsupervised, and semi-supervised anomaly detection methods.

Finally, Chapter 6 concludes the research by summarizing the insights from the results, discussing any limitations, and providing direction for future work based on the evaluation.

CHAPTER 2

Related Works

Existing literature for loan default prediction is extensive and primarily focused on balanced datasets. Most research focuses on resampling the training data to balance it and then using the resampled balanced data to train supervised classification methods. Research on anomaly detection and early warning systems on large unbalanced datasets is less extensive than ideal. In this chapter, we shall review related research contributions for both anomaly detection and early warning systems. We shall then compare them in the context of loan default prediction on large unbalanced datasets and state their contributions and limitations. We then describe the general drawbacks of current research in this context and state the motivation of this thesis.

2.1 Existing Relevant Literature

In 2018, Qiu, Tu et al. [8] built an early warning system using anomaly detection to detect problems in users' power consumption patterns. The paper provides an application for anomaly detection in the context of early warning systems. However, the researchers only evaluate computation time and the area under the ROC curve to evaluate the metrics for only the Anomaly Detection Algorithm Based on Log Analy-SIS (ADLA) method. This paper also does not focus on large datasets or loan defaults.

Mukherjee and Badr [9] in 2022 compare four unsupervised anomaly detection methods on a large and realistic P2P loan dataset, overcoming some limitations of the previous research. They use precision and recall as evaluation metrics. This paper is one of the few to use anomaly detection in the context of loan risk evaluation. However, the paper does not evaluate more evaluation metrics, such as accuracy, the area under the ROC curve, and running time. It only evaluates the approaches on one dataset and, therefore, only on one type of loan (P2P loan). It also evaluates it in the context of binary classification alone.

In the same year, Rao, Liu, et al. [10] implemented a novel approach by proposing a PSO-XGBoost model to predict loan defaults. Unlike the previous research, this paper compares existing methods using multiple metrics, including execution time. However, they only evaluate the method against one type of loan (automobile loans), compare only with three other methods, and resample the training data to balance the number of defaulters instead of using the data as is.

In 2023, Zhu, Ding, et al. [11] improved over the PSO-XGBoost model. They proposed a novel state-of-the-art approach using CNNs for feature selection and LightGBM for prediction and demonstrated higher prediction performance. However, the evaluation is performed only on one dataset and for one loan class. Also, like the previous research, the dataset is resampled to avoid imbalance instead of directly using it. The results are also binary classes and not probabilistic.

The above two papers resampled the data and converted the training data to a balanced dataset from an unbalanced dataset. Song, Wang, et al. [12] overcame this limitation by developing a novel rating-specific and multi-objective ensemble method to classify imbalanced class distribution in the case of predicting loan defaults. The methodology also focuses on maximizing sensitivity to correctly classify the minority class, i.e., customers who default on loans. However, the method does not compare execution times, only evaluates one dataset type (P2P loans) and only deals with binary classification.

2.2 Comparative Summary

From the review of existing related work, we discover that the idea of loan default prediction and credit risk monitoring is well-researched. We also observe different results demonstrated on different datasets. In addition, we observe that most approaches view loan default prediction as a classification problem with two equal classes, and the goal is to predict one class over the other. However, in designing an early warning system, because most customers in real-time data do not default, there is a need to view loan defaults as an anomaly detection problem rather than a classification problem.

Table 2.2.1: Summary of contributions of related work

Authors	**Year**	**Type**	**Contributions**
Qiu, Tu, et al.	2018	Anomaly detection	Compares performance, uses anomaly detection for an early warning system
Mukherjee and Badr	2022	Unsupervised Learning	Uses anomaly detection for loan default prediction
Rao, Liu, et al.	2022	PSO-XGBoost	Compares performance, large amount of metrics
Zhu, Ding, et al.	2023	CNN and LightGBM	Recent, novel, high accuracy, reasonable dataset size
Song, Wang, et al.	2023	Ensemble learning	Recent, novel, focuses on sensitivity, large amount of metrics and comparisons

Table 2.2.1 summarizes the contributions of the discussed relevant literature. It mentions the author, the year of publication, the type of approach used, and the relevant contributions to warning against potential loan defaults using anomaly detection.

Table 2.2.2 compares the approaches discussed so far with each other and our approach. The following are the descriptions of each column in the table.

- **Evaluation metrics:** This column refers to the research using a wide range of metrics to evaluate the algorithm's efficacy.

Table 2.2.2: Contributions and drawbacks of various approaches

Type	**Evaluation metrics**	**Performance metrics**	**Multiple datasets**	**Probabilistic**	**Comprehensive comparison**
Anomaly detection	×	✓	×	×	×
Unsupervised learning	×	×	×	×	×
PSO-XGBoost	✓	✓	×	×	×
CNN and LightGBM	✓	✓	×	×	×
Ensemble learning	✓	×	×	×	✓
Proposed work	✓	✓	✓	✓	✓

- **Performance metrics:** It refers to the research evaluating the speed of the approach using metrics like training and prediction time.

- **Multiple datasets:** This refers to the approach being evaluated against a wide range of different types of data and datasets instead of evaluating the approach on only one dataset or a small number of them.

- **Probabilistic:** If a methodology is probabilistic, the evaluation considers the method's adaptability to a change in the minimum risk threshold value. In constructing an early warning system, it is important that the system allows adjustment based on the institution's risk appetite using the system rather than simply returning a binary value suggesting that a particular data point is an anomaly.

- **Comprehensive comparison:** If an evaluation is comprehensively compared, the method is compared against many other methods. Comprehensive comparison of a method with others is crucial as it increases confidence in the efficacy of the method as compared to other methods.

As we have seen, this section highlights several limitations in existing research on warning against potential loan defaults using anomaly detection. Firstly, most studies only test methods on one type of loan dataset, neglecting the holistic evaluation of methods across various loan types. Additionally, there is a need for more emphasis on the speed and performance of algorithms, considering the vast amount of data financial institutions possess.

Moreover, current research primarily focuses on predicting loan defaults without addressing the probabilistic nature of such predictions. An effective early warning system should allow for flexibility in adjusting the probabilistic threshold to accommodate different risk appetites of financial institutions.

Furthermore, while most loan default prediction methods utilize classification algorithms, the real-world data demonstrate that loan defaults are deviations or anomalies rather than a distinct class of customers. Therefore, anomaly detection methods can leverage real-world data directly, avoiding the preprocessing and retraining efforts required to balance default and non-default customer data.

In terms of evaluation, the existing literature needs comprehensive comparisons across different approaches and evaluation metrics, limiting the understanding of the relative strengths and weaknesses of different methods.

Motivated by these limitations, this thesis aims to address the gaps by evaluating various approaches in different contexts and proposing an architecture for an early warning system utilizing anomaly detection to warn against potential loan defaults. We use a total of fifteen different supervised, unsupervised, and semi-supervised anomaly detection methods and compare them to select the best method for the proposed early warning system. We thus aim to provide valuable insights into developing an effective early warning system for loan defaults, considering the diverse characteristics of loan datasets and the probabilistic nature of predictions.

In this chapter, we discussed the current research on loan default prediction and anomaly detection using early warning systems. We summarized current research's shortcomings in developing an early warning system against loan defaults and how this thesis aims to overcome these shortcomings. The next chapter will dive into our proposed approach to this problem.

CHAPTER 3

Proposed Approach

3.1 Introduction

In Chapter 1, when introducing an early warning system, we described specific characteristics an early warning system should possess, such as timeliness, effectiveness, accessibility, scalability, interoperability, and education and training. In the context of constructing an early warning system against potential loan defaults, we define the following ideal characteristics such a system should have.

- **Versatility**: Banks offer various kinds of loans. An early warning system should work the same for predicting potentiality of defaults for any loan type.
- **Fast and scalable**: Banks deal with large customer data. An early warning system should be fast enough to handle large data and offer scalability to accommodate rapid data volume changes.
- **Function in real-time:** For every change or refresh in the data, the early warning system should update its predictions and be able to notify if a customer might default.
- **Probabilistic**: Every financial company has a unique risk appetite. The early warning system should function probabilistically based on the risk threshold of a financial institution rather than providing fixed predictions.

- **Work on imbalanced datasets**: Most people do not default on their loans, and default data would thus be imbalanced. The system should thus naturally work on such imbalanced data.

Consequently, this thesis aims to determine a machine learning approach that is fast and scalable to classify if a customer might default. This is proposed by evaluating multiple anomaly detection methods to decide the best approach on different loan default datasets. The ideal system would have high accuracy, low prediction times, a high true positive rate (the number of correct loan default predictions) and a low false negative rate (the number of incorrect loan non-default predictions). The proposed system architecture aims to detect and report warning signs and not actual defaults. Therefore, the prediction threshold for such a classification method would be lower than for a method that would predict certain defaults.

In this chapter, we shall first propose the experimental setup for evaluating the various anomaly detection algorithms to obtain the optimal approach for all loan types. Then we shall describe the high-level early warning system architecture where the optimal algorithm would be used.

3.2 Experimental Setup

The experimental setup design is shown in Fig. 3.2.1. A dataset is initially selected from our set of four loan default datasets D. Then the dataset is pre-processed, and only relevant and necessary features for predicting defaults are chosen. Then from the set of various anomaly detection approaches A', we choose a predictive algorithm. This algorithm is trained over a part of the dataset and tested over another unseen part. We train the algorithm on 80% of the dataset and use 20% of the data as our test set. The confidence scores, also known as probabilities of default, are obtained using this algorithm over the unseen test subset. Based on each risk threshold value in the set T, the predictions are generated and compared against the true predictions in the test dataset. We perform this process ten times for each algorithm and dataset. Evaluation metrics are then computed for all of these predictions. The mean of the

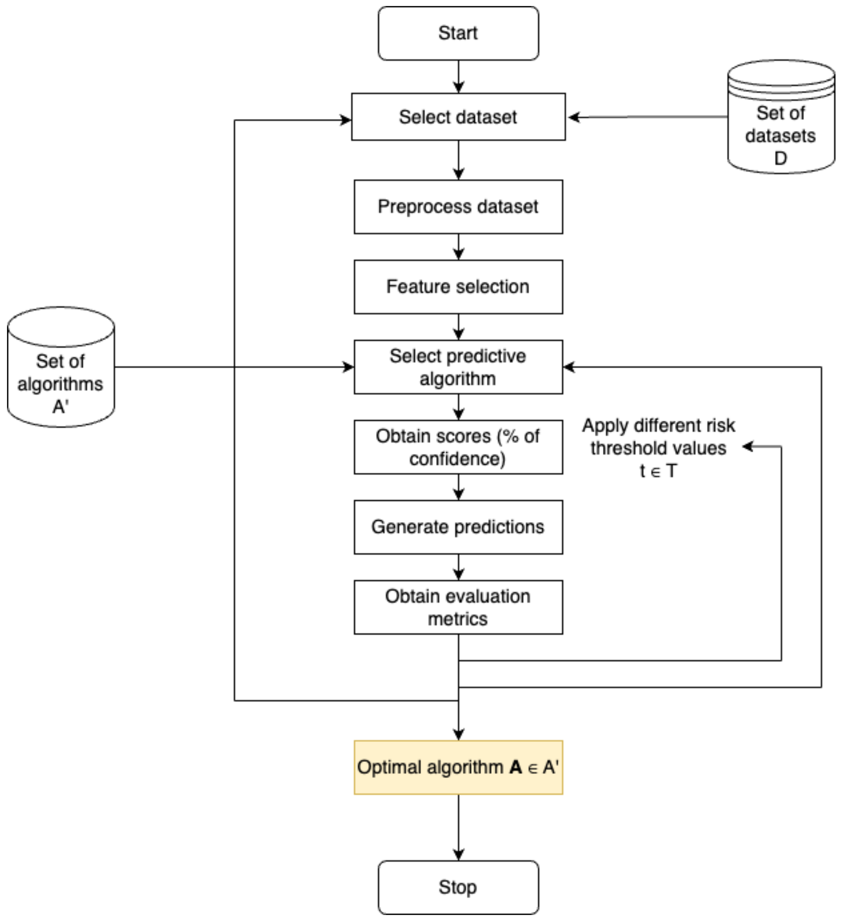

Fig. 3.2.1: Proposed experimental setup for evaluation search

evaluation metrics for each algorithm and dataset is used for comparison. Here, we do not use cross-validation in the interest of time; instead we simply split the data into training and test set for each run.

The procedure is repeated for all the algorithms in A' over all the datasets in D. Once all the evaluation metrics for each permutation of the experiment are obtained, they are analyzed to determine the optimal anomaly detection algorithm $A \in A'$ over all datasets. Algorithm 3.2.1 describes the proposed step-by-step experimental setup for evaluation.

3.3 Early Warning System Architecture

Let us consider a data store D' belonging to a financial institution, an optimal anomaly detection algorithm A obtained from the previous proposed experimental

Algorithm 3.2.1 Proposed experimental setup for evaluation

```
Input: Set of loan datasets $D$, set of anomaly detection approaches $A'$, set of risk
    threshold values $T$
Output: Optimal algorithm $A \in A'$

  Generate a clean set of data $D'$ by preprocessing and feature selection
  for each dataset in $D'$ do
      for each risk threshold value in $T$ do
          Execute each algorithm in $A'$
          For each algorithm in $A'$, note down the evaluation metrics
      end for
  end for
  Analyze the evaluation metrics
  Obtain the best algorithm $A \in A'$ based on evaluation metrics
```

setup for evaluating algorithms, and the financial institution's risk threshold value t. The data store D' contains loan data of the financial institution's customers. Ideally, this data are periodically updated with information such as the current debt, if the loan is ongoing or closed, and any attributes of the customers such as their income, housing situation, and financial status.

After every refresh, the data are extracted from the data store D', and all data unchanged from the previous refresh is eliminated. If we run the process for the first time, there is no elimination step, and all the data are considered in the next step. Preprocessing is performed on the data to make it suitable to be passed onto the optimal algorithm. Any loans already paid off or defaulted are removed as they are no longer needed for risk assessment. These data are kept aside and stored in D' for future retraining of the model based on the optimal algorithm A. We remove any unnecessary or irrelevant features through a feature selection process.

Once the data are ready, it is used on the previously trained model of the optimal algorithm A. The model returns the probabilities of default. Based on the current risk threshold value of the financial institution t, the probabilities are used to generate a risk assessment report R containing early warnings for potential future loan defaults, which is the output of this process. The report R is shared with the stakeholders and is also stored in the institution's data store D' for future reference. Based on this output, the stakeholders may take appropriate action to mitigate the risk of defaults.

Once enough new paid-off or default data are available, we combine them with the previous data. Any outdated data (for example, loans older than 5 years) are eliminated. The resulting dataset is used as a training dataset to retrain the model with the optimal algorithm A. The existing model is replaced with the retrained model. These retraining steps are periodically performed when enough new data is available for training. This ensures that the model captures new patterns of loan defaults and the intricacies of the current lending market to increase its efficacy in raising early warnings for newer patterns of defaults that did not exist previously.

Algorithm 3.3.1 Proposed early warning system architecture

```
Input: Financial institution's data store $D'$, optimal anomaly detection algorithm
  $A \in A'$, financial risk threshold value $t \in T$
Output: Risk assessment report $R$

  After every data refresh, obtain unchanged data from $D'$
  Perform preprocessing and feature selection on $D'$
  Apply $A$ onto the preprocessed data using $t$
  Generate $R$ with potentially defaulting customers and share with stakeholders
  Store report in $D'$
  for every set time period do
      From preprocessed data, obtain paid-off loans and known defaults
      Combine with previous data in the data storage and filter out outdated data
      Retrain the optimal predictive algorithm A on this data
      Update existing model with the retrained model
  end for
```

The implementation of our proposed approach can be found in the linked GitHub repository.[1]

This chapter described an experimental setup for evaluating various anomaly detection approaches to obtain an optimal algorithm. Next, we introduced an early warning system architecture that would use this optimal algorithm to warn against potential loan defaults. The next chapter will focus on this approach's experimental setup and evaluation.

[1]`https://github.com/RayhaanPirani/AnomalyDetectionEarlyWarningLoanDefaults`

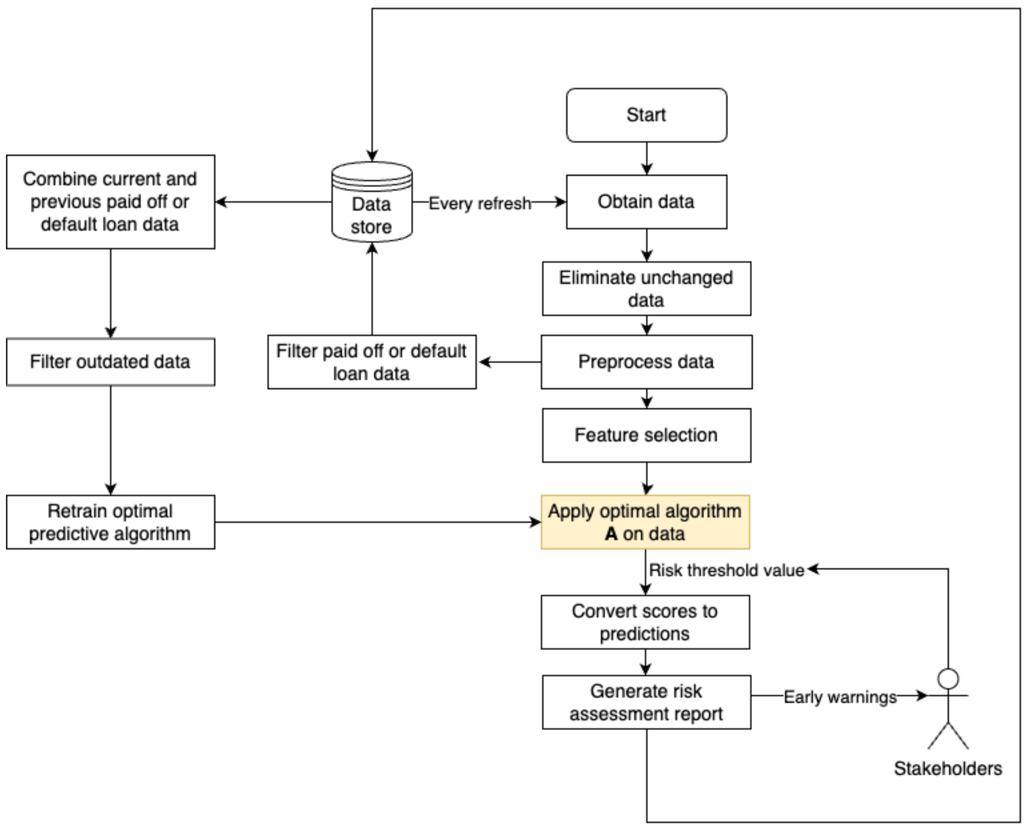
Start
Obtain data
Data store
Every refresh
Combine current and previous paid off or default loan data
Eliminate unchanged data
Filter outdated data
Filter paid off or default loan data
Preprocess data
Feature selection
Retrain optimal predictive algorithm
Apply optimal algorithm **A** on data
Risk threshold value
Convert scores to predictions
Generate risk assessment report
Early warnings
Stakeholders

Fig. 3.3.1: Proposed early warning system architecture

CHAPTER 4

Experimental Evaluation

In this chapter, we describe the libraries and tools used for experimentation along with the system configuration, the different algorithms evaluated, datasets used and the reasoning behind their usage, hyperparameter tuning of the algorithms, and the evaluation metrics considered to compare the performance of the approaches.

4.1 Libraries and Tools

We performed the experiments using Python v3.9.7 [13] on Microsoft Windows 11. For some supervised anomaly-detection algorithms, dataset split, hyperparameter tuning, and calculating metrics, we used the library Scikit-learn [14]. For the supervised boosting methods, we use their independent libraries (XGBoost [15], LightGBM [16], and CatBoost [17]). For the unsupervised and a few semi-supervised anomaly detection algorithms, the library PyOD [18] was used. For the remainder of the semi-supervised anomaly detection algorithms, we used the DeepOD [19] library. PyOD and DeepOD use the PyTorch [20], Keras [21] and TensorFlow [22] libraries for deep learning tasks. The libraries NumPy [23] and Pandas [24] were used for data processing. Matplotlib [25] was used for plotting charts. The following section lists the details of the library versions.

- NumPy v1.22.3
- Pandas v1.4.3

- Matplotlib v3.5.2
- Scikit-learn v1.2.2
- XGBoost v1.6.1
- LightGBM v3.3.2
- CatBoost v1.0.6
- PyOD v1.0.2
- DeepOD v0.3.0
- Keras v2.8.0
- TensorFlow v2.8.0

4.2 System Configuration

We use the following system configuration to evaluate the efficacy and performance of the anomaly detection models.

- **Operating System:** 64-bit Microsoft Windows 11
- **CPU:** 2.90GHz Intel Core i7-10700
- **Memory:** 3200 MHz 16 GB DDR4 RAM
- **GPU:** 4GB NVIDIA GeForce RTX 3060

4.3 Algorithms

Based on the ADBench Anomaly Detection Benchmark results by Han, et al. [26], we evaluate the best performing fifteen different algorithms in our experiment; five supervised methods, unsupervised, and semi-supervised anomaly detection algorithms. We discuss all the algorithms used in this section.

4.3.1 Supervised Binary Classification Methods

Supervised methods treat anomaly detection as binary classification, utilizing existing classifiers like logistic regression, stochastic gradient descent, and boosting. However, making use of ground truth labels can risk missing unknown anomalies, and specialized supervised anomaly detection algorithms are limited. Supervised methods are primarily used for detecting known anomalies. We use the supervised machine learning methods from the scikit-learn library and individual boosting libraries for this experiment, described as follows.

Logistic Regression (LR) Classification

The core idea behind logistic regression, as initially discussed by McCullagh and Nelder in Generalized Linear Models (1983) [27], is to link the linear predictor (a linear combination of predictor variables) to the probability of the binary outcome using a logistic function, also known as the sigmoid function. This function maps the linear predictor to a value between 0 and 1, representing the probability of the binary outcome. The logistic function ensures that the predicted probabilities remain within this range, accommodating the binary nature of the response variable.

Estimating the parameters of the logistic regression model involves applying maximum likelihood estimation (MLE) techniques. The MLE approach determines the values of the model parameters for the maximization of the likelihood of observing the given binary outcomes based on the predictor variables. The model's parameters represent each predictor variable's effect on the binary outcome's log-odds.

In logistic regression, interpreting the estimated coefficients is essential for understanding the relationship between the predictor variables and the binary outcome. These coefficients quantify the impact of each predictor variable on the log-odds of the outcome. By exponentiating the coefficients, one can interpret them as odds ratios, representing the multiplicative change in the odds of the outcome for a one-unit increase in the corresponding predictor variable, with other variables constant.

Assumptions underlying logistic regression include

- The independence of observations.
- The linearity between the predictors and the log-odds of the outcome (which can be assessed through various techniques, such as diagnostic plots).
- The absence of multicollinearity among the predictor variables.

Stochastic Gradient Descent Classification

Stochastic Gradient Descent (SGD) is a widely used optimization algorithm that has found application in various machine learning tasks, including classification. The concept can be traced back to the paper "Stochastic Estimation of the Maximum of a Regression Function" by J. Kiefer and J. Wolfowitz (1952) [28].

SGD classification is a variant of gradient descent optimization that addresses computational efficiency and scalability concerns when dealing with large datasets. It aims to find the optimal parameters of a classification model by iteratively updating them based on a randomly selected subset of training examples, known as mini-batches. This stochastic sampling strategy allows for efficient computations compared to the traditional gradient descent, which requires processing the entire training dataset at each iteration.

The process of SGD classification involves the following steps:

1. **Initialization:** The model's parameters, such as coefficients or weights, are initialized with random values.
2. **Iterative Update:** A mini-batch of training examples is randomly selected at each iteration. The model makes predictions using the current parameter values and computes the error or loss between the predicted and actual labels.
3. **Parameter Update:** The model updates its parameters by computing the loss function's gradient with respect to the parameters using the mini-batch. The gradient indicates the direction of the steepest descent in the parameter space. To minimize the loss, the parameters are then adjusted in the opposite direction of the gradient.

4. **Convergence:** The iterations continue until a convergence criterion is met, such as reaching a predefined number of iterations or achieving a desired level of loss reduction.

The advantages of SGD classification lie in its ability to handle large-scale datasets, computational efficiency, and potential to avoid local optima. However, the convergence of SGD might be slower than full gradient descent due to the random sampling of mini-batches.

Extreme Gradient Boosting (XGBoost) Classification

XGBoost, an optimized gradient boosting algorithm, is widely utilized as a machine learning classifier. This algorithm was introduced in the paper "XGBoost: A Scalable Tree Boosting System" by Tianqi Chen and Carlos Guestrin (2016) [15].

XGBoost classification is based on the gradient boosting framework, which aims to construct a robust predictive model by combining an ensemble of weak learners, often decision trees. The XGBoost algorithm enhances the gradient boosting approach by employing innovative techniques that optimize performance and efficiency.

The core concept of XGBoost classification involves iteratively training decision trees to correct the errors made by the previous trees in the ensemble. Each subsequent tree is built by minimizing a specific loss function, such as logistic loss, for binary classification. The trees are trained sequentially, with each tree adding further improvements to the overall ensemble's predictive power.

The distinctive characteristics and techniques employed in XGBoost classification, as described in the referenced paper, include:

1. **Regularization:** XGBoost incorporates regularization techniques such as L1 and L2 regularization to prevent overfitting and enhance generalization performance. This regularization helps control the complexity of individual trees and the overall model.

2. **Gradient-based Optimization:** XGBoost employs a novel technique called "gradient-based optimization" to efficiently optimize the objective function by

computing gradients and using second-order approximations, resulting in faster convergence and improved performance.

3. **Parallel Processing:** XGBoost supports parallel processing by utilizing multiple threads to build trees simultaneously, enabling efficient computation and reducing training time.

4. **Tree Pruning:** The algorithm applies tree pruning techniques to prevent overfitting and enhance model generalization. Pruning removes unnecessary branches and nodes from the trees, reducing complexity while maintaining predictive accuracy.

5. **Handling Missing Values:** XGBoost incorporates a mechanism to handle missing values in the data by making informed decisions during the tree construction.

Light Gradient Boosting Machine (LightGBM) Classification

LightGBM is a widely used gradient boosting framework in machine learning tasks like classification. The fundamental understanding of LightGBM classification can be derived from the paper "LightGBM: A Highly Efficient Gradient Boosting Decision Tree" by Guolin Ke, et al. (2017) [16]. Zhu, Ding, et al. (2023) [11] used a version of this approach with a convolutional neural network layer, as discussed in Chapter 2.

LightGBM classification builds upon the gradient boosting algorithm and introduces several optimizations to improve efficiency and performance. It leverages a novel technique called Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to achieve high accuracy with reduced computational cost.

The concept of LightGBM classification involves the following key aspects:

1. **Gradient Boosting:** LightGBM utilizes the gradient boosting framework, where an ensemble of decision trees is constructed sequentially. Each subsequent tree corrects the errors made by the previous trees to improve the overall predictive power of the model.

2. **Gradient-based One-Side Sampling (GOSS):** GOSS is a technique employed by LightGBM to optimize the training process. It samples a subset of instances based on their gradients, prioritizing instances with larger gradients for higher contribution to the learning process. This approach reduces the number of instances used for training while preserving the informative samples, leading to enhanced computational efficiency.

3. **Exclusive Feature Bundling (EFB):** EFB is a feature engineering technique introduced by LightGBM to enhance efficiency further. It bundles mutually exclusive features, reducing the number of feature categories and decreasing the memory footprint during training.

4. **Additional Optimizations:** LightGBM incorporates additional optimizations such as histogram-based computing, leaf-wise tree growth, and a unique way of handling categorical features. These optimizations contribute to faster training and prediction times, making LightGBM well-suited for large-scale datasets.

Categorical Boosting (CatBoost) Classification

CatBoost classification was introduced by Prokhorenkova, et al. (2019) [17]. It builds upon the gradient boosting algorithm and introduces innovative techniques to handle categorical features effectively. It addresses the challenges posed by high-cardinality categorical variables by employing a combination of ordered boosting and novel gradient-based strategies.

The core concept of CatBoost classification encompasses the following key aspects:

1. **Gradient Boosting:** CatBoost utilizes the gradient boosting framework, which involves sequentially constructing an ensemble of decision trees. Each subsequent tree corrects the errors made by the previous trees, improving the overall predictive power of the model.

2. **Handling Categorical Features:** CatBoost provides efficient solutions for handling categorical features, which are common in many real-world datasets.

It employs an algorithmic approach called Ordered Boosting, which transforms the categorical features into numerical values using an ordering algorithm. This preserves the category information while enabling the gradient-based optimization process.

3. **New Gradient-Based Strategies:** CatBoost introduces novel gradient-based strategies to improve model accuracy and generalization. It incorporates gradient calculation corrections and a novel algorithm for optimizing the learning rate during the training process. These techniques contribute to improved performance and stability of the model.

4. **Advanced Regularization:** CatBoost employs advanced regularization techniques such as L2 regularization, feature combinations, and per-feature permutations to prevent overfitting and enhance model generalization.

4.3.2 Unsupervised Methods for Anomalous Data Distribution

Unsupervised methods assume different data distributions, such as anomalies in low-density regions. These methods have been proposed over the years and can be categorized into shallow and deep (neural network) methods. Shallow methods offer better interpretability, while deep methods handle large, high-dimensional data more effectively. The performance of these algorithms depends on the agreement between the data and the assumptions of the algorithms. All of the unsupervised anomaly detection methods in our experiments come from the PyOD library. We describe them as follows.

Isolation Forest (IF) Anomaly Detection

Isolation Forest developed by Liu, et al. (2008) [29] is based on the principle that anomalies are data points that are few in number and different from the majority of the normal data points. The concept of Isolation Forest unsupervised anomaly detection involves the following key aspects:

1. **Anomaly Isolation:** The algorithm constructs isolation trees by recursively partitioning the data. It randomly selects a feature and splits the data along a random value within the feature's range. By repeating this process recursively, the algorithm creates a collection of isolation trees.

2. **Path Length for Scoring:** The anomaly score of a data point is determined by the average path length in the isolation trees required to isolate that point. Anomalies are expected to have shorter average path lengths as they are easier to isolate due to their distinctive nature.

3. **Random Splitting:** The random selection of features and random splitting values within those features allows the algorithm to isolate anomalies efficiently, as anomalies are expected to require fewer splits to become isolated. This randomness contributes to the algorithm's efficiency and effectiveness in detecting anomalies.

4. **Ensemble-based:** Isolation Forest utilizes an ensemble of isolation trees to enhance the overall detection performance. Anomalies are expected to have shorter average path lengths across multiple trees. In comparison, regular data points will likely have longer average path lengths. The ensemble-based approach helps reduce false positives and improves the robustness of the anomaly detection process.

Angle-Based Outlier Detection (ABOD)

Angle-Based Outlier Detection (ABOD) created by Kriegel, et al. (2008) [30] identifies outliers by evaluating the angles formed between data points within a dataset. The concept of ABOD encompasses the following key aspects:

1. **Angle Calculation:** ABOD computes the angles between a data point and all possible pairs of other data points. These angles are measured using the

geometric properties of the dataset and provide insights into the relationships and distributions of the data.

2. **Outlier Score:** The outlier score for each data point is determined based on the variations in the angles it forms with other data points. Outliers exhibit more significant deviations in angle values than most data points, reflecting their dissimilarity and potential anomalous nature.

3. **Handles High-Dimensional Data:** ABOD is specifically designed to address the challenges of outlier detection in high-dimensional data. Traditional distance-based methods often struggle with the curse of dimensionality, making angle-based approaches more effective in capturing the structural information and variations within the dataset.

4. **Geometric:** ABOD is an unsupervised algorithm and thus does not require prior knowledge or labelled data to detect outliers. It relies solely on the geometric properties and patterns observed within the dataset to identify potential anomalies.

Empirical-Cumulative-distribution-based Outlier Detection (ECOD)

Empirical-Cumulative-distribution-based Outlier Detection (ECOD), presented by Li, et al. (2022) [31], identifies outliers by leveraging the empirical cumulative distribution function (ECDF), which characterizes the distribution of data points. The concept of ECOD encompasses the following key aspects:

1. **Empirical Cumulative Distribution Function:** ECOD constructs the ECDF by ordering the data points and calculating the cumulative probability distribution. The ECDF provides information about the data distribution and enables the detection of outliers based on their deviation from the expected distribution.

2. **Outlier Score Calculation:** ECOD calculates an outlier score for each data point based on its distance to the ECDF curve. Data points that exhibit more

considerable distances from the ECDF curve are considered potential outliers, as they deviate significantly from the expected distribution.

3. **Efficiency Considerations:** ECOD emphasizes computational efficiency to handle large-scale datasets. It achieves this by adopting efficient algorithms for constructing the ECDF, such as an interval-based algorithm that avoids excessive sorting operations. This allows ECOD to handle datasets with millions of data points efficiently.

Copula-Based Outlier Detection (COPOD)

Copula-Based Outlier Detection (COPOD), discussed by Li, et al. (2020) [32], identifies outliers by modelling the dependence relationships between variables using copula functions. The concept of COPOD encompasses the following key aspects:

1. **Copula Functions:** Copula functions are employed to describe the multivariate dependence structure of the data. Copulas capture the correlation between variables, allowing for a more accurate assessment of outliers based on the joint behaviour of variables.

2. **Scoring:** COPOD determines the outlyingness of each data point based on the copula-based score. This score represents the deviation of the data point from the expected dependence structure modelled by the copula function. Higher scores indicate a higher likelihood of the data point being an outlier.

3. **Multivariate:** COPOD extends outlier detection to multivariate datasets, simultaneously considering the interactions and dependencies among multiple variables. By incorporating copula functions, COPOD can capture complex dependence patterns and detect outliers that may not be apparent when considering variables in isolation.

Deep Support Vector Data Description (DeepSVDD)

Deep Support Vector Data Description (DeepSVDD) by Ruff, et al. (2018) [33], introduced in their paper titled "Deep One-Class Classification", identifies anomalies by learning a compact representation of normal data unsupervised using deep autoencoders. The concept of DeepSVDD encompasses the following key aspects:

1. **Deep AutoEncoders:** DeepSVDD employs deep autoencoders, a type of neural network architecture, to learn a low-dimensional representation of normal data. The AutoEncoder is trained to reconstruct the input data, aiming to capture normal instances' essential features and patterns.

2. **One-Class Classification:** DeepSVDD formulates the anomaly detection task as a one-class classification problem. It seeks to identify data points that deviate significantly from the learned representation of normal data. It defines anomalies as instances outside the normal data manifold in the learned feature space.

3. **Hypersphere-based Decision Boundary:** DeepSVDD defines a hypersphere in the learned feature space that encloses the normal data instances. Anomalies are considered data points that fall outside this hypersphere, indicating their dissimilarity to most normal instances.

4. **Deep Embedding Optimization:** DeepSVDD optimizes the deep autoencoder network by minimizing the reconstruction error between the input and the reconstructed output. This process aims to learn a compact and discriminative representation of normal data while suppressing the reconstruction of anomalies.

4.3.3 Semi-supervised Methods with Efficient Label Usage

Semi-supervised methods leverage partial labels to improve detection performance while detecting unseen types of anomalies. Some studies focus on using partially labelled data and leveraging unlabeled data for representation learning. For example, some semi-supervised models may be trained on normal samples and detect anomalies that deviate from these representations. In most of the algorithms used in our study, semi-supervision refers to incomplete label learning in weak supervision. Except for XGBOD, which is from the PyOD library, all the other semi-supervised anomaly detection methods in our experiments are deep-learning based. They are a part of the DeepOD library.

eXtreme Gradient Boosting for Outlier Detection (XGBOD)

Semi-Supervised XGBOD, proposed and demonstrated by Zhao and Hryniewicki (2019) [34], leverages XGBoost, a powerful gradient boosting algorithm, in a semi-supervised setting to enhance outlier detection. It combines the power of eXtreme Gradient Boosting (XGBoost) and unsupervised representation learning for anomaly detection. The concept of Semi-Supervised XGBOD encompasses the following key aspects:

1. **Outlier Detection:** XGBoost is an ensemble learning algorithm that combines multiple weak learners (decision trees) to create a robust model. In the context of outlier detection, XGBoost is adapted to identify anomalies based on their deviations from most normal instances.

2. **Unsupervised Representation Learning:** Semi-Supervised XGBOD incorporates unsupervised representation learning to capture the underlying structure of the data. The algorithm learns a compact and informative representation that can effectively capture normal data patterns by training an unsupervised autoencoder on the normal instances.

3. **Semi-Supervised Framework:** Semi-Supervised XGBOD combines the unsupervised representation learning with a supervised XGBoost model trained on labelled data. This combination allows for leveraging labelled and unlabeled data to improve the accuracy of outlier detection. The unsupervised representation learning aids in capturing the intrinsic characteristics of normal data,

while the supervised XGBoost model utilizes the labelled data to distinguish between normal and anomalous instances.

4. **Boosting and Ensemble Learning:** XGBoost employs boosting, a technique that iteratively builds a robust model by combining multiple weak models. Each weak model focuses on different aspects of the data, and their predictions are combined to make a final decision. This ensemble learning approach enhances the outlier detection capability of XGBoost.

Feature Encoding with AutoEncoders for Weakly-Supervised Anomaly Detection (FeaWAD)

The concept of Feature Encoding with AutoEncoders for Weakly-Supervised Anomaly Detection (FeaWAD) was introduced by Zhou, et al. (2021) [35]. This algorithm utilizes AutoEncoders for feature encoding and applies weakly-supervised learning to perform anomaly detection. It encompasses the following key aspects:

1. **AutoEncoders:** AutoEncoders are neural network architectures used for unsupervised learning. They consist of an encoder and a decoder, which learn to encode and reconstruct the input data. In anomaly detection, AutoEncoders encode the input features into a lower-dimensional representation.

2. **Feature Encoding:** The algorithm utilizes autoencoders to perform feature encoding, mapping the high-dimensional input features into a lower-dimensional latent space. This process aims to capture the essential characteristics of normal instances while preserving the distinctive patterns of anomalies.

3. **Weakly-Supervised Anomaly Detection:** The algorithm incorporates weak supervision, leveraging partial labels or prior knowledge about a subset of anomalies during the training phase. Combining the encoded features with weak supervision allows the algorithm to distinguish between normal and anomalous instances.

4. **Reconstruction Error:** The reconstruction error, measured as the discrepancy between the original input and its reconstructed version, indicates anomalousness. Larger reconstruction errors indicate instances that deviate significantly from the learned representation of normal data, suggesting the presence of anomalies.

Deep Semi-Supervised Anomaly Detection (DeepSAD)

Deep Semi-Supervised Anomaly Detection (DeepSAD) was proposed by Ruff, et al. (2019) [36]. It combines the power of deep learning with semi-supervised learning techniques for anomaly detection. It is characterized by the following attributes:

1. **Deep Learning:** Deep Semi-Supervised Anomaly Detection utilizes deep learning techniques, specifically deep neural networks, to learn representations and detect anomalies. Deep neural networks comprise multiple layers of interconnected artificial neurons, allowing them to learn complex patterns and representations from the input data.

2. **Semi-Supervised Learning:** The algorithm leverages labelled and unlabeled data for training. The availability of labelled data allows the algorithm to learn representations of normal instances, while the unlabeled data helps capture the underlying data distribution. The algorithm can effectively detect anomalies by combining labelled and unlabeled data.

3. **Autoencoder Architecture:** Deep Semi-Supervised Anomaly Detection employs an autoencoder architecture consisting of an encoder and a decoder. The encoder maps the input data into a lower-dimensional latent space. At the same time, the decoder reconstructs the input from the latent representation. By training the autoencoder on normal instances, the algorithm learns to reconstruct them accurately, enabling it to detect anomalies based on the reconstruction errors.

4. **Anomaly Scoring:** Deep Semi-Supervised Anomaly Detection computes anomaly

scores based on the difference between the input and its reconstructed output. Larger reconstruction errors indicate instances that deviate significantly from the normal data distribution, suggesting the presence of anomalies.

Pairwise Relation prediction Network (PReNet)

As introduced by Pang, et al. (2023) [37], the Pairwise Relation prediction Network (PReNet) algorithm addresses the challenge of weakly-supervised anomaly detection. It aims to detect both seen and unseen anomalies by learning pairwise relation features and anomaly scores by predicting relations between randomly sampled training instances.

PReNet leverages a deep learning approach, specifically a relation neural network, to predict the pairwise relations of instances. The relations can be anomaly-anomaly, anomaly-unlabeled, or unlabeled-unlabeled. By jointly learning these pairwise relations, PReNet captures discriminative patterns of anomalies in relation to anomalies, anomalies in relation to normal instances, and normal instances in relation to each other. This approach allows PReNet to detect seen and unseen abnormalities that fit the learned abnormal patterns or deviate from normal patterns.

During inference, PReNet considers a test instance an anomaly if it aligns well with the first two types of pairs or deviates from the last pair type when paired with a random training instance. By unifying the relation prediction and anomaly scoring, PReNet assigns higher anomaly scores to instance pairs that contain anomalies compared to other pairs. The algorithm also augments the training anomaly data by generating large-scale anomaly-informed surrogate class labels, enhancing training effectiveness with limited labelled data.

The contributions of this work include

- Addressing the problem of weakly-supervised anomaly detection.
- Proposing the PReNet approach for learning diverse pairwise relation features.
- Designing a detection model based on a relation neural network.

- Demonstrating the robustness of PReNet in leveraging unlabeled data while being tolerant to anomaly contamination.

Anomaly Detection using Deviation Networks (DevNet)

The Deviation Networks (DevNet) algorithm, developed by Guansong Pang, et al. (2019) [38], addresses certain limitations of existing deep learning methods for anomaly detection. While previous approaches focus on learning new feature representations for downstream anomaly detection, DevNet proposes an end-to-end learning framework that directly optimizes anomaly scores, leading to more data-efficient learning and improved anomaly scoring.

The motivation behind DevNet stems from the challenges faced by existing deep anomaly detection methods, which often rely on unsupervised learning due to the need for large-scale labelled anomaly data. This limitation prevents the effective utilization of prior knowledge, such as a few labelled anomalies, which can be valuable in real-world anomaly detection applications. DevNet leverages a novel anomaly detection framework incorporating a neural deviation learning approach to overcome these challenges.

In the proposed framework, instead of focusing on representation learning, DevNet learns anomaly scores directly. Given the original data as inputs, the framework employs a neural anomaly score learner to assign anomaly scores to training data objects. A reference score is defined based on the mean anomaly scores of normal data objects, utilizing a prior probability to guide the learning process. The deviation loss, a loss function defined within the framework, enforces statistically significant deviations of the anomaly scores of anomalies from those of normal data objects in the upper tail.

The instantiation of the framework results in the DevNet method. DevNet utilizes multiple to dozens of labelled anomalies, which account for a small percentage of the training data objects and anomalies per dataset, along with a Gaussian prior. It performs direct optimization of anomaly scores using a Z-Score-based deviation loss. This approach enables DevNet to efficiently learn anomaly scores while accommodating anomalies with different anomalous behaviours. Notably, the Z-Score-based deviation loss also facilitates the production of easily interpretable anomaly scores, distinguishing DevNet from many existing methods.

The contributions of this work include the introduction of a novel framework for end-to-end learning of anomaly scores, representing a departure from the indirect optimization approach. The framework enables the utilization of limited labelled anomaly data for achieving end-to-end anomaly score learning. The instantiation of the framework as DevNet provides a method that combines neural networks, Gaussian prior, and Z-Score-based deviation loss. DevNet achieves data-efficient and effective learning of anomaly scores, resulting in optimized and easily interpretable ones.

4.4 Datasets

In order to evaluate the different supervised methods and anomaly detection algorithms discussed in the previous section, we use four distinct datasets with different characteristics. This is because different loans have different characteristics, and our goal is to evaluate the various approaches to see if they work well under different loan types and conditions.

It is also important to note that banking data is highly inaccessible for academic research. Banking information is proprietary and usually protected with stringent security measures. Data sharing in the banking industry is heavily regulated, and privacy laws impose strict restrictions on releasing such data. Moreover, only some high-quality public datasets exist for training loan default prediction models.

The datasets used in this research were obtained from reputable sources, ensuring their reliability and credibility. Some of these datasets have been previously utilized in research studies, some even discussed in Chapter 2, and machine learning competitions, indicating their suitability for training and evaluation.

We describe the four datasets used in our experimentation as follows.

4.4.1 Bondora Peer-to-Peer Lending Data

The Bondora Peer-to-Peer Lending Data, developed by Manu Siddhartha (2020) [39], is an open dataset available on the IEEE Dataport platform. This dataset, obtained from a prominent European P2P (peer-to-peer) lending platform, provides a comprehensive collection of loans from 1st March 2009 to 27th January 2020. The dataset includes borrower demographics, financial information, and loan transaction details. P2P loans are typically uncollateralized, with lenders seeking higher returns to offset the inherent financial risk.

This dataset is also mostly preprocessed. Besides, it is also balanced, with 59% loan defaulters, denoted by 1, and the rest non-defaulters, denoted by 0. The dataset has 48 attributes and 77,394 records.

We performed the following preprocessing steps on the Bondora Peer-to-Peer Lending Dataset:

- We removed irrelevant columns (language, country, county, and city) since these attributes do not impact the loan default ability of customers.
- We converted all the binary columns to ones and zeroes from True and False.
- We converted all the date columns to the number of months from the date until the present date.
- We encoded the ordinal classes to scores (verification, education, rating, employment type, employment duration, and home ownership status).
- We encoded the nominal classes using one-hot encoding.
- We removed columns containing too many blanks.
- Any blank last payments were imputed with their first payment information. The reasoning is that if no previous payments were made except for the first payment, the first payment itself would be the latest.
- The remaining rows containing any blanks were removed (less than 10% of data).

After following the above steps, the resultant Bondora data had 85 attributes and 70,512 records left.

4.4.2 L&T Vehicle Loan Default Dataset

The L&T Vehicle Loan Default Dataset by L&T Finance is collected and available on the Kaggle platform (2019) [40]. This dataset is unbalanced and has almost 22% loan defaulters and the rest non-defaulters. It provides a perspective on automobile loans and secured loans. It is well-cited and used in many research papers, including the one by Rao, Liu, et al. [10] discussed in Chapter 2. The preprocessing steps we apply for this dataset in our experimentation are very similar to the ones in this paper.

It comprises 233,154 records and 41 variables, which include 40 independent variables for predicting loan default. These variables encompass loanee information (e.g., age, identity proof), loan details (e.g., disbursement information, loan-to-value ratio), and bureau data and history (e.g., bureau score, number of active accounts, credit history). The dataset also includes a dependent variable, "loan_default," which classifies borrowers into binary categories: 0 for non-defaulters and 1 for defaulters.

We performed the following preprocessing steps on the Bondora Peer-to-Peer Lending Dataset:

- We converted the date of birth column to the current age of the customer.
- We converted the loan disbursal date column to the number of months since the disbursal.
- We converted the average account age and credit history length from text format to the number of months.
- We encoded the categorical columns to scores (employment type and CNS score description)
- We created the following new quantitative columns from the existing columns: loan to asset ratio, total accounts, primary inactive accounts, secondary inactive

accounts, total inactive accounts, total active accounts, total current balance, total sanctioned amount, total disbursed amount, total installment amount, primary loan proportions, secondary loan proportions, and active to inactive account ratio.

- We removed the records with over three-quarters of the quantitative columns as zero to avoid bias towards missing and irrelevant data in our models.
- We removed irrelevant columns (unique ID, branch ID, supplier ID, manufacturer ID, current pincode ID, state ID, employee code ID, and mobile number available flag).

The resultant L&T Vehicle Loan Default data had 46 attributes and 120,165 records left after we performed the above preprocessing steps.

4.4.3 LendingClub Data

The LendingClub loan default dataset, available on Kaggle (2020) [41], is valuable for assessing the credit risk associated with peer-to-peer lending loans. LendingClub, headquartered in San Francisco, is the largest peer-to-peer lending platform globally and has registered its offerings with the Securities and Exchange Commission. With a reported origin of \$15.98 billion in loans through its platform by the end of 2015, this dataset provides an extensive and significant source of data for studying credit risk in P2P lending.

This dataset is the largest in the study, with 2,925,493 records and 141 attributes. About 1.58% of the records in this dataset refer to defaulted loans, and the rest are not defaults. Due to its sheer size, this dataset is ideal for stress and performance testing. It is also well-cited in research related to loan defaults. For example, Jadwal, et al. [42] use this dataset to demonstrate a novel oversampling algorithm called Spectral SMOTE for addressing class imbalance in P2P lending datasets like the LendingClub data.

We performed the following preprocessing steps on the LendingClub Dataset:

- We dropped irrelevant columns such as ID, URL, title, address, ZIP code, and payment plan.
- We dropped the grade column since the sub-grade column is a more granular version of the information Grade provides.
- We removed columns that have over 30% of their values as nulls to avoid bias towards missing and irrelevant data in our models.
- We converted the interest rate and revolving credit utilization textual percentage columns to numeric.
- We converted the term and employment length textual columns representing months and ranges of months to their respective numeric representations.
- We converted all the date columns to the number of months from the date until the present date.
- We encoded the ordinal classes to scores (sub-grade, employment title, verification status, home ownership status, loan status, purpose of loan, initial list status, application type, hardship flag, and debt settlement flag).

The resultant LendingClub data had 98 attributes and 1,501,168 records left after we performed the above preprocessing steps. Due to the size, for our experiments, we use a subset of this data that is 5% in size of the dataset selected by randomized stratified sampling.

4.4.4 Deloitte-MachineHack Default Prediction Data

The Deloitte-MachineHack Loan Default Prediction Data, available on Kaggle (2021) [43], is a dataset used in a hackathon organized by Deloitte and MachineHack in late 2021. The hackathon aimed to predict loan defaulters by analyzing various attributes like funded amount, location, and loan balance. Participants were required to have skills in handling big datasets, understanding underfitting vs. overfitting, and optimizing the “log_loss” metric to ensure good generalization on unseen data.

The training dataset consists of 67,463 rows and 35 columns. It provides a perspective on all general types of loans. About 9.25% of the records are loan defaults.

We performed the following preprocessing steps on the Deloitte-MachineHack Loan Default Prediction Dataset:

- We dropped irrelevant columns such as ID, batch enrolled, and payment plan.
- We encoded the ordinal classes to scores (grade, sub-grade, verification status, initial list status, and application type).
- We condensed the loan title column values by grouping multiple same or similar categories into generalized groups. For example, any loan title containing keywords related to a consolidation loan, like 'paydown', 'pay off', and 'debt relief', was condensed to a single category called 'consolidation'. The new categories obtained were 'Consolidation', 'Credit card', 'Home', 'Medical', 'Car', 'Personal', and 'Other'.
- We encoded the nominal classes using one-hot encoding.

The resultant Deloitte-MachineHack Loan Default data had 40 attributes and 67,463 records left after we performed the above preprocessing steps.

Table 4.4.1 summarizes the structure of the datasets before and after preprocessing. We observe that for all the datasets except Deloitte-MachineHack, the number of records after preprocessing decreased. This is primarily due to deleting rows with many empty or irrelevant values. We also noticed that the number of attributes for all the datasets except for LendingClub increased. This is due to the feature engineering of new columns and removing all irrelevant ones.

Table 4.4.1: Summary of the structure of datasets

Dataset	**Before preprocessing**		**After preprocessing**	
	Attributes	**Records**	**Attributes**	**Records**
Bondora Peer-to-Peer Lending	48	77,394	85	70,512
L&T Vehicle Loan	41	233,154	46	120,165
LendingClub	141	2,925,493	98	1,501,168
Deloitte-MachineHack	35	67,463	40	67,463

4.5 Hyperparameter Tuning

Hyperparameter tuning, sometimes also known as hyperparameter optimization, is a process of evaluating various hyperparameters for machine learning models to select an optimal set of them to maximize the model's performance [44]. Hyperparameters are parameters in a machine learning model that are external to the model and whose values cannot be estimated using the training data. In this section, we describe the methodology of hyperparameter tuning used in this thesis and discuss all the hyperparameters evaluated in our experiments.

We selected optimal hyperparameters for each of the four datasets and the fifteen methods evaluated. Hyperparameter tuning was performed by training a standard version of the algorithm and performing 5-fold randomized cross-validation [45] on the split dataset over different hyperparameter values. The hyperparameters in the randomized cross-validation that performed the best were selected for each combination of method and dataset. The AUC ROC metric (area under the ROC curve) was used to compare the performance of the hyperparameters when a model's internal scoring method was unavailable. These optimal sets of hyperparameters associated with each combination were used for performing the experiments.

The following is the list of all hyperparameters assessed in this thesis.

- **Penalty:** This is used in supervised algorithms like logistic regression and stochastic gradient descent classifiers. The L1 penalty function uses the sum of absolute values of the parameters, and Lasso Regression encourages this sum to be minimized. The L2 penalty function uses the sum of the squares of the parameters, and Ridge Regression encourages the maximization of this sum. [46] For the stochastic gradient descent classifier, another penalty type ElasticNet acts as a convex combination of L2 and L1 penalty [14].

- **Regularization parameter (C/alpha/lambda):** The regularization parameter is used in the logistic regression classifier as the C value to control regularization. In other approaches, such as boosting and the stochastic gradient descent classifier, alpha describes the L1 regularization while lambda L2 regularization. A higher regularization parameter value means the model should give a higher weight to training data, while a lower one means that the model should be more generalized [14, 15, 16, 18].

- **Loss function:** A loss function hyperparameter is used in the stochastic gradient descent classifier. It maps the values of the variables associated with an event to a real number to represent the 'cost' or 'loss' required to perform the event. [27] It is also known as a 'cost function'. We evaluate the loss functions 'log_loss' and 'modified_huber' for probabilistic estimates. [14]

- **Learning rate:** The learning rate refers to the step size at each iteration while performing a minimization of a loss function [47]. The learning rate is sometimes also referred to as 'lr'. This parameter is used in many supervised and semi-supervised algorithms, including stochastic gradient descent classifiers, boosting approaches like CatBoost and XGBOD, and deep learning methods like FeaWAD, DeepSAD, PReNet, and DevNet. [14, 15, 16, 17, 18, 19]

- **Number of estimators:** This hyperparameter is used in boosting and ensemble algorithms such as XGBoost and LightGBM and anomaly detection approaches like XGBOD and isolation forest anomaly detection. It refers to the minimum number of trees or estimators in the ensemble. More estimators improve the algorithm's predictive ability but reduce the training and prediction times. [15, 16, 18, 29]

- **Split criterion:** Used in decision tree-based algorithms, such as boosting and ensemble approaches. Describes the function based on which the split should be computed in a decision tree. Gini and entropy are commonly used for the split criterion function. [48]

- **Min samples split:** It is the minimum number of samples needed to split an internal node. Also used commonly in decision tree-based algorithms, such as boosting and ensemble approaches. [48]

- **Min samples leaf:** It is the minimum number of samples needed to be at a child or leaf node. Also used commonly in decision tree-based algorithms, such as boosting and ensemble approaches. [48]

- **Max leaf nodes:** It is the maximum number of leaf nodes in a single estimator. Also used commonly in decision tree-based algorithms, such as boosting and ensemble approaches. [48]

- **Max depth:** The maximum limit for the depth of an estimator. Also used commonly in decision tree-based algorithms, such as boosting and ensemble approaches. [48]

- **Column sample by tree:** The fraction of the columns a single estimator can use. Used commonly in ensemble and boosting approaches. [15, 16, 18]

- **Subsample:** Used commonly in boosting approaches. Selects the percentage of data on which bootstrap aggregation (or bagging) should be performed. [15, 16, 18]

- **Min child weight:** Used commonly in boosting approaches. It defines the minimum sum of instance weight required for a child node. [15, 16]

- **Border count:** Used commonly in CatBoost. It describes the number of splits for a feature (either numerical border count or categorical border count). [18]

- **Number of neighbours:** Used in geometric methods like ABOD. Specifies the number of neighbours to use to consider a data point as an anomaly. [30, 18]

- **Contamination:** Used in unsupervised anomaly detection approaches such as isolation forest anomaly detection, ABOD, ECOD, and COPOD. This parameter is used to define the proportion of outliers in the dataset. Defines the threshold of anomalies on the decision function when fitting. This hyperparameter is fixed to the proportion of defaults for each dataset instead of tuning. [18]

- **Number of epochs:** Typically used as a hyperparameter in deep learning algorithms such as FeaWAD, DeepSAD, PReNet, and DevNet. Defines the number of times the dataset has to be worked through by the algorithm. [35, 36, 37, 38, 19]

- **Batch size:** Also typically used as a hyperparameter in deep learning algorithms such as FeaWAD, DeepSAD, PReNet, and DevNet. Defines the number of samples that should be processed before updating the model. [35, 36, 37, 38, 19]

- **Activation function:** Also typically used as a hyperparameter in deep learning algorithms such as FeaWAD, DeepSAD, PReNet, and DevNet. It is a function that defines a given node's output given a set of inputs. [35, 36, 37, 38, 19]

  Multiple activation functions exist, such as ReLU, LeakyReLU and sigmoid functions. ReLU, which stands for rectified linear unit, is one of the most popular activation functions today. It outputs the input directly if it is positive, else it outputs 0 [49]. LeakyReLU is a modification of ReLU that allows a slight non-zero gradient instead of zero to avoid division by zero errors and work better in handling noisy data. The sigmoid activation function, or the logistic function, takes any real value and outputs a value between 0 and 1 [27].

- **Bias:** A boolean hyperparameter specific to the PyTorch backend of deep learning anomaly detection approaches like FeaWAD, DeepSAD, PReNet, and DevNet that are implemented with the DeepOD library. If it is set to True, the model will learn an additive bias and will not otherwise. [19, 20]

Table 4.5.1 shows the hyperparameter grid search space for every supervised algorithm used to conduct the randomized grid search over all the datasets. Table 4.5.2 shows the same search space for unsupervised and semi-supervised algorithms.

Tables 4.5.3, 4.5.4, and 4.5.5 refer to the optimal sets of hyperparameters obtained after hyperparameter tuning for supervised, unsupervised and semi-supervised algo-

Table 4.5.1: Randomized hyperparameter grid search space for supervised algorithms

	Algorithm	**Randomized Grid Search Space**
Supervised	LR	penalty : ['l1', 'l2'], C : [0.001,0.01,0.1,1,10,100]
	SGD	loss: ['log_loss', 'modified_huber'], penalty: ['l1', 'l2', 'elasticnet'], alpha: [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000], learning_rate: ['constant', 'optimal', 'invscaling', 'adaptive']
	XGB	n_estimators: [100, 200, 500], criterion: ['gini', 'entropy'], min_samples_split: [1, 2, 4, 5], min_samples_leaf: [1, 2, 4, 5], max_leaf_nodes: [4, 10, 20, 50, None]
	LGBM	num_leaves: [5, 10, 20, 50], n_estimators: [50, 100, 150], max_depth: [4, 6], colsample_bytree: [0.7, 0.8, 0.9], subsample: [0.7, 0.8, 0.9], min_child_samples: [10, 50, 100], min_child_weight: [1e-5, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4], reg_alpha: [0, 1e-1, 1, 2, 5, 7, 10, 50, 100], reg_lambda: [0, 1e-1, 1, 5, 10, 20, 50, 100]
	CB	depth:[1, 2, 5, 10], iterations: [100, 200, 500, 1000], learning_rate: [0.001, 0.01, 0.1, 0.2, 0.3], l2_leaf_reg: [1, 5, 10, 50], border_count: [5, 10, 20, 50, 100, 200]

Table 4.5.2: Randomized hyperparameter grid search space for unsupervised and semi-supervised algorithms

	Algorithm	**Randomized Grid Search Space**
Unsupervised	IF	n_estimators: [100, 200, 500, 1000]
	ABOD	n_neighbors: [1, 2, 5, 10]
	ECOD[1]	N/A
	COPOD[1]	
	DeepSVDD[1]	
Semi-supervised	XGBOD	epochs: [10, 20, 50, 100],
	FeaWAD	batch_size: [16, 32, 64, 128],
	DeepSAD	lr: [0.001, 0.01, 0.1, 0.2, 0.3],
	PReNet	act: ['ReLU', 'LeakyReLU', 'Sigmoid'],
	DevNet	bias: [True, False]

rithms, respectively, across all the datasets.

[1]No hyperparameters available. Only the fixed contamination parameter was added.
[2]Execution did not finish. The execution was stopped after 3 hours of running.

Table 4.5.3: Optimal hyperparameter sets across datasets for supervised approaches

Algorithm	**Optimal Hyperparameter Set**			
	Bondora P2P	**L&T Vehicle Loan**	**LendingClub**	**Deloitte-MachineHack**
LR	penalty: l2, C: 100	penalty: l2, C: 10	penalty: l1, C: 1	penalty: l2, C: 0.1
SGD	penalty: elasticnet, loss: log_loss, learning_rate: optimal, alpha: 0.01	penalty: l1, loss: modified_huber, learning_rate: optimal, alpha: 1	penalty: l1, loss: modified_huber, learning_rate: optimal, alpha: 1000	penalty: l1, loss: log_loss, learning_rate: optimal, alpha: 1
XGB	n_estimators: 500, min_samples_split: 1, min_samples_leaf: 5, max_leaf_nodes: 50, criterion: entropy	n_estimators: 100, min_samples_split: 4, min_samples_leaf: 1, max_leaf_nodes: 20, criterion: entropy	n_estimators: 100, min_samples_split: 1, min_samples_leaf: 5, max_leaf_nodes: 10, criterion: entropy	n_estimators: 500, min_samples_split: 1, min_samples_leaf: 2, max_leaf_nodes: 4, criterion: gini
LGBM	subsample: 0.7, reg_lambda: 50, reg_alpha: 10, num_leaves: 5, n_estimators: 100, min_child_weight: 0.1, min_child_samples: 50, max_depth: 6, colsample_bytree: 0.9	subsample: 0.7, reg_lambda: 50, reg_alpha: 0, num_leaves: 20, n_estimators: 150, min_child_weight: 100.0, min_child_samples: 10, max_depth: 4, colsample_bytree: 0.7	subsample: 0.7, reg_lambda: 1, reg_alpha: 2, num_leaves: 20, n_estimators: 50, min_child_weight: 100.0, min_child_samples: 50, max_depth: 6, colsample_bytree: 0.8	subsample: 0.8, reg_lambda: 100, reg_alpha: 2, num_leaves: 50, n_estimators: 150, min_child_weight: 0.01, min_child_samples: 100, max_depth: 4, colsample_bytree: 0.8
CB	learning_rate: 0.3, l2_leaf_reg: 10, iterations: 100, depth: 10, border_count: 100	learning_rate: 0.01, l2_leaf_reg: 5, iterations: 500, depth: 2, border_count: 100	learning_rate: 0.3, l2_leaf_reg: 5, iterations: 200, depth: 2, border_count: 100	learning_rate: 0.1, l2_leaf_reg: 5, iterations: 200, depth: 5, border_count: 50

Table 4.5.4: Optimal hyperparameter sets across datasets for unsupervised approaches

Algorithm	**Optimal Hyperparameter Set**			
	Bondora P2P	**L&T Vehicle Loan**	**LendingClub**	**Deloitte-MachineHack**
IF	n_estimators: 1000	n_estimators: 100	n_estimators: 500	n_estimators: 200
ABOD	n_neighbors: 2	n_neighbors: 5	n_neighbors: 10	n_neighbors: 10
ECOD	N/A			
COPOD				
DeepSVDD				

Table 4.5.5: Optimal hyperparameter sets across datasets for semi-supervised approaches

Algorithm	**Optimal Hyperparameter Set**			
	Bondora P2P	**L&T Vehicle Loan**	**LendingClub**	**Deloitte-MachineHack**
XGBOD[2]	DNF			
FeaWAD	epochs: 20, batch_size: 32, lr: 0.01, act: LeakyReLU, bias: False	epochs: 20, batch_size: 32, lr: 0.01, act: LeakyReLU, bias: False	epochs: 10, batch_size: 16, lr: 0.2, act: ReLU, bias: True	epochs: 10, batch_size: 32, lr: 0.2, act: LeakyReLU, bias: False
DeepSAD	epochs: 10, batch_size: 32, lr: 0.2, act: LeakyReLU, bias: False	epochs: 10, batch_size: 32, lr: 0.1, act: Sigmoid, bias: False	epochs: 20, batch_size: 32, lr: 0.01, act: LeakyReLU, bias: False	epochs: 10, batch_size: 64, lr: 0.1, act: Sigmoid, bias: True
PReNet	epochs: 10, batch_size: 32, lr: 0.2, act: LeakyReLU, bias: False	epochs: 20, batch_size: 32, lr: 0.01, act: LeakyReLU, bias: False	epochs: 20, batch_size: 32, lr: 0.01, act: LeakyReLU, bias: False	epochs: 20, batch_size: 32, lr: 0.01, act: LeakyReLU, bias: False
DevNet	epochs: 20, batch_size: 32, lr: 0.01, act: LeakyReLU, bias: False	epochs: 20, batch_size: 32, lr: 0.01, act: LeakyReLU, bias: False	epochs: 20, batch_size: 64, lr: 0.1, act: LeakyReLU, bias: False	epochs: 10, batch_size: 32, lr: 0.1, act: Sigmoid, bias: False

4.6 Evaluation Metrics

Evaluation metrics are essential in anomaly detection and machine learning to assess the performance and effectiveness of a model in solving a problem. They provide quantitative measures that allow us to compare different models, algorithms, or techniques and decide which approach works best for a given task. Here are some key reasons why evaluation metrics are crucial:

1. **Performance Comparison:** Evaluation metrics enable us to compare the performance of different models or algorithms on the same dataset. They enable us to identify the most effective approach for the problem at hand by quantifying how well each model performs.

2. **Model Selection:** When working with multiple candidate models, evaluation metrics help us select the best model to deploy in real-world applications. The model with the highest performance on the evaluation metric is often chosen for deployment.

3. **Generalization Assessment:** Evaluation metrics allow us to assess how well a model generalizes to unseen data. This helps to detect overfitting (a model that performs well on the training data but poorly on new data) and underfitting (a model that performs poorly on both training and new data).

4. **Problem-Specific Optimization:** Some tasks may have specific requirements or constraints. Evaluation metrics can be tailored to measure those specific aspects, optimizing the model accordingly.

5. **Understanding Model Behavior:** Evaluation metrics provide insights into the strengths and weaknesses of a model. For example, precision-recall curves can reveal trade-offs between precision and recall in binary classification tasks.

6. **Communication and Reporting:** Evaluation metrics provide a concise and objective way to communicate the performance of a model to stakeholders,

clients, or peers. They help in presenting results and conclusions in a clear and quantifiable manner.

Choosing multiple effective evaluation metrics specific to the problem for a holistic evaluation is extremely important. For example, if we consider accuracy as the only metric for loan defaults on unbalanced datasets, a model that predicts all customers to not default would have 99.9% accuracy if the dataset consists of 99.9% non-defaulters and only 0.1% defaulters. In such a case, using other metrics, such as recall or precision, can help us evaluate the models more effectively. [50]

In the following sections, we shall discuss the evaluation metrics used in our experiments.

4.6.1 Accuracy

Accuracy, in this case, is also known as classification accuracy. It is the ratio of correct predictions on a test dataset to the total number of predictions made.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} \tag{4.6.1}$$

It acts as a powerful metric when all the classes in the dataset have more or less the same number of data points. However, for unbalanced datasets, like most loan default datasets, accuracy is not a great metric as it becomes heavily biased towards the majority class. Therefore, other metrics must be given more weight for this problem than accuracy.

4.6.2 Precision

Precision is the percentage of results that are relevant among the total results. It is computed as the ratio of the correct positive results, or true positives (TP), to the total number of positive results predicted by the algorithm, i.e., the sum of true positives (TP) and false positives (FP).

$$Precision = \frac{TP}{TP + FP} \tag{4.6.2}$$

In the case of loan defaults, defaulting on a loan is considered the relevant prediction.

4.6.3 Recall

Recall, also known as sensitivity, is the percentage of relevant results that were predicted correctly. This is an essential metric in the case of unbalanced classes. It is computed as the ratio of the correct positive results, or true positives (TP), to the total number of relevant results predicted by the algorithm. The total number of relevant results refers to all the instances that should have been predicted as positive, i.e., the sum of true positives (TP) and false negatives (FN).

$$Recall = \frac{TP}{TP + FN} \tag{4.6.3}$$

4.6.4 F1 Score

F1 Score combines both precision and recall into a single metric. It tells us how precise the model is (how many instances were correctly classified) and how robust it is (how the model avoids missing a significant number of instances). It is computed as the harmonic mean of the precision and recall metrics.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.6.4}$$

4.6.5 Area Under Curve

Area Under Curve (AUC) is the area under the Receiver Operating Characteristic (ROC) curve. It is popular for binary classification problems. The ROC curve is plotted in a two-dimensional space, with the two dimensions being True Positive Rate (TPR) and False Positive Rate (FPR). The True Positive Rate (TPR) is also known as recall or sensitivity defined by equation 4.6.3. The False Positive Rate (FPR) is an inverted version of TPR. It is computed as the ratio of false positives (FP) to all the instances that should have been predicted as negative, i.e., the sum of true negatives (TN) and false positives (FP).

$$FPR = \frac{FP}{TN + FP} \tag{4.6.5}$$

4.6.6 Training and Prediction Times

Execution time refers to the processing time (usually represented in ms or s) taken for a computer process to finish from start to end.

Training time refers to the execution time required for training a dataset on a model. Prediction time refers to the execution time required for predicting the outcome of a data point using a trained model.

In this chapter, we discussed the experimental setup, the algorithms and datasets used, the process and outcome of hyperparameter tuning, and the evaluation metrics for our experimental evaluation. In the next chapter, we shall present, analyze and discuss the findings of our experiments.

CHAPTER 5

Results, Discussion, and Analysis

This chapter presents and discusses the results of our experiments and analyzes the performance of the different algorithms used. We first present the results by each dataset for each algorithm with a binary prediction and probabilistic predictions with a lower and higher risk threshold value. We also discuss the findings from these results. Then, we analyze the results based on the type of datasets and algorithms used and present various cases where different approaches would be ideal to be included as the optimal prediction algorithm for the proposed early warning system architecture.

5.1 Experiment Results and Discussion

In this section, we provide an overview of the results of our experimental evaluation for each of the four datasets used. We also discuss any interesting observations and patterns in our findings.

The tables show the accuracy, precision, recall, F1 score, AUC efficacy metrics, and performance metrics of training and prediction times. 'Train. time (sec)' refers to the training time in seconds, while 'Pred. time (sec)' refers to the prediction time in seconds.

5.1.1 Bondora Peer-to-Peer Lending Data

As discussed earlier, the Bondora Peer-to-Peer Lending Dataset is a balanced dataset with the majority of customers being loan defaulters (59%). This dataset also only

Table 5.1.1: Evaluation results for all algorithms on the Bondora P2P dataset

	Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC**	**Train. time (sec)**	**Pred. time (sec)**
Supervised	LR	0.9988	1.0000	0.9981	0.9991	0.9704	18.97	0.01
	SGD	0.9709	0.9823	0.9720	0.9771	0.9950	0.23	0.01
	XGB	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.54**	**0.01**
	LGBM	0.9998	1.0000	0.9997	0.9998	0.9998	0.21	0.02
	CB	1.0000	1.0000	1.0000	1.0000	1.0000	3.04	0.02
Unsupervised	IF	0.4438	0.5848	0.4554	0.5120	0.4990	42.09	12.08
	ABOD	0.3592	0.0000	0.0000	0.0000	0.5000	5.98	126.23
	ECOD	0.4509	0.5922	0.4593	0.5174	0.4476	0.98	2.62
	COPOD	0.4529	0.5944	0.4602	0.5188	0.4500	0.95	2.38
	DeepSVDD	0.3750	0.5790	0.0900	0.1557	0.4867	119.99	0.43
Semi-supervised	XGBOD	DNF						
	FeaWAD	0.3727	0.5662	0.0900	0.1557	0.4835	2.10	1.83
	DeepSAD	0.4610	0.9993	0.1589	0.2742	0.5794	45.58	1.66
	PReNet	0.3736	0.5727	0.0885	0.1533	0.4853	274.55	30.69
	DevNet	0.4588	0.9972	0.1558	0.2695	0.5775	90.84	1.82

has information on peer-to-peer loans.

Table 5.1.1 shows the results of the evaluation of all algorithms on the Bondora Peer-to-Peer Lending data. It is clear that all supervised methods perform the best on this balanced dataset, while unsupervised and semi-supervised methods perform poorly.

The XGBoost method performs the best among the supervised methods considering all the evaluation metrics and the training and execution times. Among the unsupervised methods, isolation forest, ECOD, and COPOD performed better than ABOD and DeepSVDD. ABOD performed the worst, given its average precision, recall, and F1 score metrics. The deep learning methods, including the unsupervised DeepSVDD and the four supervised methods, exhibited similar performance with varying execution times.

Table 5.1.2 shows the same results discussed above for a low 30% and a high 70% risk threshold. In the case of supervised methods, the results are similar. XG-

Table 5.1.2: Low and high threshold evaluation results for all algorithms on the Bondora P2P dataset

Threshold	Algorithm	Accuracy		Precision		Recall		F1 Score		AUC	
		30%	**70%**	**30%**	**70%**	**30%**	**70%**	**30%**	**70%**	**30%**	**70%**
Supervised	LR	0.9989	0.9987	0.9999	1.0000	0.9983	0.9979	0.9991	0.9989	0.9991	0.9989
	SGD	0.9177	0.9363	0.8922	0.9985	0.9914	0.9020	0.9392	0.9478	0.8889	0.9498
	XGB	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	LGBM	1.0000	0.9998	1.0000	1.0000	1.0000	0.9997	1.0000	0.9998	1.0000	0.9998
	CB	0.9999	0.9999	0.9999	1.0000	1.0000	0.9999	0.9999	0.9999	0.9999	0.9999
Unsupervised	IF	0.4203	0.3670	0.5734	0.6305	0.3724	0.0142	0.4515	0.0277	0.4391	0.4997
	ABOD	0.3592	0.3592	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.5000
	ECOD	0.6408	0.6408	0.6408	0.6408	1.0000	1.0000	0.7811	0.7811	0.5000	0.5000
	COPOD	0.6408	0.6408	0.6408	0.6408	1.0000	1.0000	0.7811	0.7811	0.5000	0.5000
	DeepSVDD	0.6201	0.6188	0.6423	0.6529	0.9188	0.8771	0.7533	0.7420	0.5030	0.5176
Semi-supervised	XGBOD	DNF									
	FeaWAD	0.6408	0.6405	0.6408	0.6408	1.0000	0.9989	0.7811	0.7807	0.5000	0.5000
	DeepSAD	0.6408	0.6397	0.6408	0.6404	1.0000	0.9983	0.7811	0.7803	0.5000	0.4992
	PReNet	0.6408	0.6402	0.6408	0.6407	1.0000	0.9986	0.7811	0.7806	0.5000	0.4998
	DevNet	0.6408	0.6391	0.6408	0.6402	1.0000	0.9973	0.7811	0.7798	0.5000	0.4987

Boost remains the best-performing algorithm for this dataset. All the semi-supervised and unsupervised methods except isolation forest and ABOD performed significantly better when dealing with a probabilistic risk-threshold-based evaluation, exhibiting considerably larger average precision, recall, and F1 scores.

Here, it is essential to note that the metric values are 1.000 for some boosting approaches. In a dataset reflecting an actual population, such high values for metrics are unlikely to occur. Moreover, the dataset is comparatively smaller than a real banking dataset is expected to be. Since we are performing a comparative study, we have not performed cross-validation on the dataset, as discussed in Chapter 3. However, the metrics as high as 1.000 are expected to be high in a real-life scenario and can thus be used for a fair comparison among methods. This is because the dataset is tested on an unseen portion of the dataset by the trained model.

5.1.2 L&T Vehicle Loan Default Dataset

The L&T Vehicle Loan Default Dataset has only about 22% of customers defaulting on their loans. This dataset deals solely with secured loans, particularly automobile loans.

Table 5.1.3 shows the evaluation results of all algorithms on the L&T Vehicle Loan Default dataset. On this unbalanced dataset, all the supervised algorithms performed poorly considering metrics like precision, recall, F1 score, and the area under the ROC curve (AUC). Since this dataset is unbalanced, the accuracy metric is not a good evaluation metric. Semi-supervised methods performed better than supervised ones but not as well as the unsupervised methods considering metrics other than accuracy.

The unsupervised ECOD anomaly detection method performed the best on this dataset, followed by COPOD, which performed similarly. Isolation forest and ABOD performed similarly, with slightly higher average accuracy but slightly worse performance in other metrics. The semi-supervised methods and the DeepSVDD unsupervised method performed about the same with slightly higher average accuracy and similar average precision as compared to isolation forest and ABOD but with lower

Table 5.1.3: Evaluation results for all algorithms on the L&T Vehicle Loan dataset

	Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC**	**Train. time (sec)**	**Pred. time (sec)**
Supervised	LR	0.7671	0.200	0.0005	0.0011	0.5000	0.84	0.01
	SGD	0.7675	0.000	0.000	0.000	0.5000	0.23	0.01
	XGB	0.7674	0.4802	0.0049	0.0097	0.5017	0.81	0.01
	LGBM	0.7675	0.0000	0.0000	0.0000	0.5000	0.17	0.02
	CB	0.7675	0.0000	0.0000	0.0000	0.5000	4.77	0.01
Unsupervised	IF	0.6613	0.2289	0.1929	0.2094	0.5354	5.11	1.38
	ABOD	0.6522	0.2092	0.1784	0.1926	0.4871	17.32	128.64
	ECOD	**0.5033**	**0.2345**	**0.5018**	**0.3197**	**0.5028**	**0.75**	**1.97**
	COPOD	0.5001	0.2299	0.4894	0.3128	0.4964	0.75	1.91
	DeepSVDD	0.7099	0.2079	0.0882	0.1239	0.4933	185.46	0.63
Semi-supervised	XGBOD	DNF						
	FeaWAD	0.7082	0.2010	0.0857	0.1202	0.4912	3.54	3.35
	DeepSAD	0.7315	0.3221	0.1399	0.1951	0.5254	93.92	3.11
	PReNet	0.7314	0.3042	0.1206	0.1727	0.5185	1634.24	53.99
	DevNet	0.7283	0.3000	0.1263	0.1778	0.5185	190.83	3.33

average recall and F1 scores.

Table 5.1.4 shows the same results discussed above for a low 30% and a high 70% risk threshold. In the case of supervised methods, the results are similar. All the semi-supervised and unsupervised methods, especially ABOD, with the except isolation forest, performed significantly better when dealing with a probabilistic risk-threshold-based evaluation, exhibiting considerably larger precision, recall, and F1 scores. ECOD remains the best method overall, followed by COPOD.

It is important to note that there is an accuracy drop in most of the methods, especially for a lower risk threshold. This drop is expected because the risk threshold evaluation is probabilistic, while the accuracy metric deals with comparing a binary outcome. In other words, when we lower the risk threshold, all the outcomes having probability above the threshold are considered at risk of default, increasing the likelihood of false positives due to the data being skewed towards non-defaulters. However, a lower risk threshold means that the financial institution is willing to take less risk

Table 5.1.4: Low and high threshold evaluation results for all algorithms on the L&T Vehicle Loan dataset

	Algorithm	Accuracy		Precision		Recall		F1 Score		AUC	
Threshold		**30%**	**70%**	**30%**	**70%**	**30%**	**70%**	**30%**	**70%**	**30%**	**70%**
Supervised	LR	0.7247	0.7675	0.3299	0.0000	0.1786	0.0000	0.2317	0.0000	0.5344	0.5000
	SGD	0.7675	0.7675	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.5000
	XGB	0.6663	0.7674	0.3293	0.0000	0.3181	0.0000	0.2924	0.0000	0.5449	0.5000
	LGBM	0.7562	0.7675	0.3926	0.0000	0.0889	0.0000	0.1450	0.0000	0.5236	0.5000
	CB	0.7505	0.7675	0.3593	0.0000	0.0932	0.0000	0.1481	0.0000	0.5214	0.5000
Unsupervised	IF	0.6725	0.7589	0.2196	0.1672	0.1600	0.0093	0.1851	0.0176	0.4939	0.4976
	ABOD	0.2326	0.2326	0.2325	0.2325	0.9996	0.9996	0.3772	0.3772	0.4999	0.4999
	ECOD	**0.2325**	**0.5140**	**0.2325**	**0.2613**	**1.0000**	**0.5965**	**0.3773**	**0.3634**	**0.5000**	**0.5428**
	COPOD	0.2325	0.4955	0.2325	0.2588	1.0000	0.6276	0.3773	0.3665	0.5000	0.5416
	DeepSVDD	0.2331	0.2336	0.2325	0.2324	0.9988	0.9969	0.3772	0.3769	0.4999	0.4996
Semi-supervised	XGBOD	DNF									
	FeaWAD	0.2325	0.2329	0.2325	0.2325	1.0000	0.9989	0.3773	0.3772	0.5000	0.4999
	DeepSAD	0.2325	0.2327	0.2325	0.2324	1.0000	0.9986	0.3773	0.3770	0.5000	0.4996
	PReNet	0.2325	0.2330	0.2325	0.2325	1.0000	0.9991	0.3773	0.3772	0.5000	0.5000
	DevNet	0.2325	0.2325	0.2325	0.2323	1.0000	0.9984	0.3773	0.3769	0.5000	0.4994

Table 5.1.5: Evaluation results for all algorithms on the LendingClub dataset

	Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC**	**Train. time (sec)**	**Pred. time (sec)**
Supervised	LR	0.9895	0.8696	0.4115	0.5587	0.7053	10.82	0.01
	SGD	0.9853	0.6949	0.1687	0.2715	0.5838	5.10	0.01
	XGB	0.9913	0.8131	0.6353	0.7132	0.8164	0.20	0.01
	LGBM	0.9830	0.0000	0.0000	0.0000	0.5000	0.26	0.01
	CB	0.9934	0.8421	0.7529	0.7950	0.8753	3.54	0.01
Unsupervised	IF	0.7958	0.0297	0.3663	0.0549	0.5845	24.82	7.84
	ABOD	0.7961	0.0344	0.4280	0.0636	0.6151	33.53	319.18
	ECOD	0.5078	0.0244	0.7176	0.0472	0.6109	1.52	3.87
	COPOD	0.5077	0.0258	0.7608	0.0499	0.6320	1.52	3.85
	DeepSVDD	0.8872	0.0240	0.1431	0.0410	0.5216	122.27	0.45
Semi-supervised	XGBOD	DNF						
	FeaWAD	0.8880	0.0068	0.0422	0.0118	0.4719	2.25	2.03
	DeepSAD	0.9139	0.1452	0.9114	0.2504	0.9127	33.41	1.71
	PReNet	0.8869	0.0014	0.0084	0.0024	0.4547	644.81	17.86
	DevNet	**0.9158**	**0.1519**	**0.9451**	**0.2617**	**0.9302**	**49.14**	**1.96**

and thus wants the early warning system to warn about customers who may be at lower as well as higher risk.

5.1.3 LendingClub Data

The LendingClub dataset is the largest in our experimentation. It has around 1.58% of loan defaulters, and the rest are non-defaulters. It is heavily unbalanced and has data solely on peer-to-peer loans.

Table 5.1.5 shows the evaluation results of all algorithms on the LendingClub dataset. Since this dataset is unbalanced, the recall or the sensitivity metric is critical as it is the percentage of relevant results, i.e., the loan defaults, predicted correctly. For the proposed early warning system, it is more important for the algorithm to be sensitive than specific.

Even though the dataset is heavily unbalanced, the supervised algorithms performed better than expected. However, most of them were more specific than sensitive. XGBoost and CatBoost performed the best, while LightGBM was the worst performer, with a zero average recall value. Unsupervised methods COPOD and ECOD had a similar recall to supervised methods with worse average precision.

Semi-supervised method DevNet performed the best for this dataset to act as an optimal algorithm for an early warning system, with extremely high average recall, followed by DeepSAD. However, DeepSAD had a slightly lower average recall and precision than DevNet but considerably lesser average training time.

Table 5.1.6 shows the same results discussed above for a low 30% and a high 70% risk threshold. All unsupervised (except isolation forest) and semi-supervised methods for both thresholds demonstrated high recall values. However, this was eclipsed by the significantly low precision values. These methods simply classified the vast majority of the data as defaulters. Implementing these approaches in the early warning system would raise warnings for almost all customers, even though they might not default, which is highly unsuitable for our case.

Most supervised algorithms also did not perform well with the exception of CatBoost and XGBoost. CatBoost performed the best with moderate recall values and high precision. It also had the maximum average AUC and F1 score values. These results show that supervised algorithms work well for peer-to-peer lending data, even if the data is imbalanced.

5.1.4 Deloitte-MachineHack Default Prediction Data

The Deloitte-MachineHack dataset provides a general-loan perspective. It is also heavily unbalanced, with only 9.25% of customers defaulting on loans. The loans are divided into multiple categories, and the dataset consists of different types of loans such as mortgages, revolving credit like credit cards and lines of credit, consolidation loans, medical loans, and personal loans.

Table 5.1.7 shows the evaluation results of all algorithms on the Deloitte-MachineHack Default Prediction dataset. The results are similar to those on the L&T Vehicle Loan Default dataset. All the supervised algorithms performed poorly again, considering metrics like precision, recall, F1 score, and the area under the ROC curve (AUC).

Table 5.1.6: Low and high threshold evaluation results for all algorithms on the LendingClub dataset

Threshold	**Algorithm**	**Accuracy**		**Precision**		**Recall**		**F1 Score**		**AUC**	
		30%	**70%**	**30%**	**70%**	**30%**	**70%**	**30%**	**70%**	**30%**	**70%**
Supervised	LR	0.9905	0.9875	0.7644	0.9516	0.6008	0.2428	0.6728	0.3869	0.7989	0.6213
	SGD	0.9836	0.9836	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.4999	0.4999
	XGB	0.9902	0.9890	0.6845	0.9440	0.8129	0.3737	0.7407	0.5345	0.9031	0.6867
	LGBM	0.9830	0.9830	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.5000
	CB	**0.9928**	**0.9923**	**0.7653**	**0.9162**	**0.8314**	**0.6000**	**0.7970**	**0.7251**	**0.9135**	**0.7995**
Unsupervised	IF	0.7250	0.9795	0.0282	0.0149	0.4774	0.0041	0.0532	0.0065	0.6032	0.4998
	ABOD	0.0164	0.0276	0.0162	0.0164	1.0000	1.0000	0.0319	0.0322	0.5001	0.5058
	ECOD	0.0170	0.4151	0.0170	0.0206	1.0000	0.7176	0.0334	0.0400	0.5000	0.5638
	COPOD	0.0170	0.1186	0.0170	0.0182	1.0000	0.9608	0.0334	0.0357	0.5000	0.5324
	DeepSVDD	0.0233	0.0242	0.0169	0.0169	0.9910	0.9894	0.0333	0.0333	0.4988	0.4985
Semi-supervised	XGBOD	DNF									
	FeaWAD	0.0158	0.0169	0.0158	0.0158	1.0000	1.0000	0.0311	0.0311	0.5000	0.5005
	DeepSAD	0.0158	0.0167	0.0158	0.0156	1.0000	0.9873	0.0311	0.0307	0.5000	0.4942
	PReNet	0.0158	0.0174	0.0158	0.0158	1.0000	1.0000	0.0311	0.0311	0.5000	0.5008
	DevNet	0.0158	0.0170	0.0158	0.0158	1.0000	1.0000	0.0311	0.0311	0.5000	0.5006

Table 5.1.7: Evaluation results for all algorithms on the Deloitte-MachineHack dataset

	Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC**	**Train. time (sec)**	**Pred. time (sec)**
Supervised	LR	0.9067	0.5000	0.0008	0.0016	0.5004	0.14	0.00
	SGD	0.9067	0.0000	0.0000	0.0000	0.5000	0.14	0.00
	XGB	0.9067	0.0000	0.0000	0.0000	0.5000	0.60	0.02
	LGBM	0.9067	0.0000	0.0000	0.0000	0.5000	0.11	0.01
	CB	0.9067	0.0000	0.0000	0.0000	0.5000	1.91	0.01
Unsupervised	IF	0.6167	0.0977	0.3773	0.1552	0.5093	5.64	1.51
	ABOD	0.7486	0.1049	0.2248	0.1430	0.5136	25.39	93.50
	ECOD	0.5062	0.0960	0.5099	0.1616	0.5079	0.53	1.33
	COPOD	**0.5009**	**0.0962**	**0.5179**	**0.1622**	**0.5085**	**0.53**	**1.34**
	DeepSVDD	0.8270	0.0988	0.1051	0.1018	0.5032	106.42	0.37
Semi-supervised	XGBOD	DNF						
	FeaWAD	0.8257	0.1020	0.1112	0.1064	0.5052	3.89	2.23
	DeepSAD	0.8275	0.0973	0.1025	0.0998	0.5023	83.16	1.55
	PReNet	0.8304	0.1152	0.1223	0.1186	0.5128	639.87	29.54
	DevNet	0.8249	0.0948	0.1025	0.0985	0.5009	46.42	1.56

Semi-supervised methods performed better than supervised ones but not as well as the unsupervised methods considering metrics other than accuracy.

The unsupervised COPOD anomaly detection method performed the best on this dataset, followed by ECOD, which performed similarly. Isolation forest performed the worst while ABOD performed moderately, with higher average accuracy but slightly worse performance in other metrics. The semi-supervised methods and the DeepSVDD unsupervised method performed about the same with higher average accuracy and similar average precision as compared to isolation forest and ABOD but with lower average recall and F1 scores.

Table 5.1.8 shows the same results discussed above for a low 30% and a high 70% risk threshold. In the case of supervised methods, the results are similar. All the semi-supervised and unsupervised methods, except isolation forest, exhibited high recall but low precision when dealing with a probabilistic risk-threshold-based evaluation with slightly larger metric scores. The unsupervised ECOD, COPOD, and

Table 5.1.8: Low and high threshold evaluation results for all algorithms on the Deloitte-MachineHack dataset

Threshold	**Algorithm**	**Accuracy**		**Precision**		**Recall**		**F1 Score**		**AUC**	
		30%	**70%**	**30%**	**70%**	**30%**	**70%**	**30%**	**70%**	**30%**	**70%**
Supervised	LR	0.9067	0.9067	0.5000	0.0000	0.0008	0.0000	0.0016	0.0000	0.5004	0.5000
	SGD	0.0933	0.9067	0.0933	0.0000	1.0000	0.0000	0.1707	0.0000	0.5000	0.5000
	XGB	0.9067	0.9060	0.0037	0.0000	0.0001	0.0000	0.0002	0.0000	0.4997	0.5000
	LGBM	0.9067	0.9067	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5000	0.5000
	CB	0.9064	0.9067	0.1667	0.0000	0.0008	0.0000	0.0016	0.0000	0.5002	0.0000
Unsupervised	IF	0.6167	0.9019	0.0977	0.1000	0.3773	0.0064	0.1552	0.0119	0.5093	0.5002
	ABOD	0.0939	0.1188	0.0933	0.0938	0.9992	0.9754	0.1707	0.1712	0.5000	0.5030
	ECOD	**0.0933**	**0.0933**	**0.0933**	**0.0933**	**1.0000**	**1.0000**	**0.1707**	**0.1707**	**0.5000**	**0.5000**
	COPOD	**0.0933**	**0.0933**	**0.0933**	**0.0933**	**1.0000**	**1.0000**	**0.1707**	**0.1707**	**0.5000**	**0.5000**
	DeepSVDD	0.0935	0.0940	0.0933	0.0933	0.9995	0.9990	0.1707	0.1706	0.4999	0.4999
Semi-supervised	XGBOD					DNF					
	FeaWAD	0.0933	0.0938	0.0933	0.0932	1.0000	0.9976	0.1707	0.1704	0.5000	0.4992
	DeepSAD	0.0933	0.0942	0.0933	0.0932	1.0000	0.9976	0.1707	0.1705	0.5000	0.4994
	PReNet	0.0933	0.094345	0.0933	0.0933	1.0000	0.9992	0.1707	0.1707	0.5000	0.5002
	DevNet	0.0933	0.094864	0.0933	0.0934	1.0000	0.9992	0.1707	0.1708	0.5000	0.5005

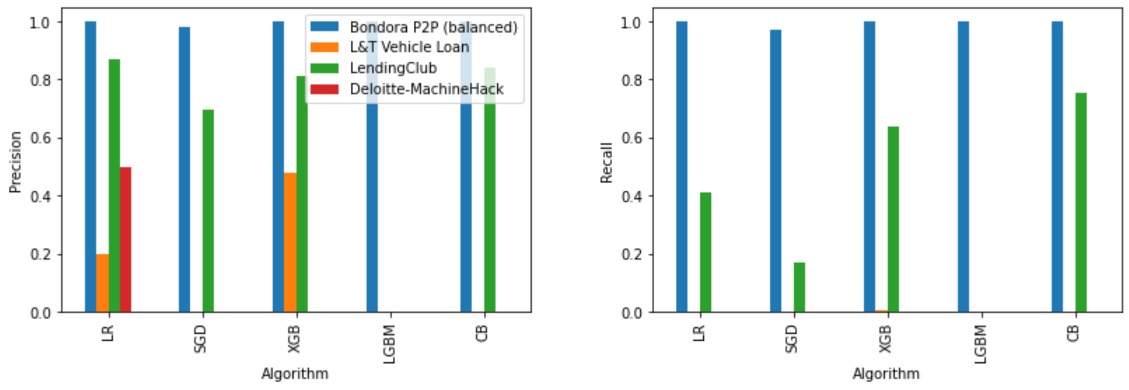

Fig. 5.2.1: Comparison of precision and recall for supervised approaches across all datasets

DeepSVDD, along with all the semi-supervised methods, were similarly effective. Considering the average training and prediction times, ECOD and COPOD performed similarly and were the best performers.

5.2 Analysis of Results and Considerations

In this section, we analyze the results considering various contexts to understand how specific approaches would suit different situations.

5.2.1 Balanced versus Unbalanced Datasets

In section 5.1, we noticed that, in most cases, the balanced Bondora Peer-to-Peer Lending dataset had significantly better performance for all supervised methods across all performance metrics compared to their performance for all other unbalanced datasets. For example, Fig. 5.2.1 demonstrates charts for comparing the precision and recall metrics across all the datasets for supervised methods. From these charts, it is clear that supervised methods performed well for the balanced Bondora Peer-to-Peer Lending data, while for unbalanced datasets, they underperformed.

From these results, we can confidently reason that unsupervised and weakly supervised methods perform better on unbalanced datasets, while supervised methods are more suitable when the dataset is balanced. Since most loan defaults in the real world are unbalanced, anomaly detection is preferable when dealing with defaults.

5.2.2 Peer-to-Peer versus Other Loans

As observed in Tables 5.1.1 and 5.1.2, boosting algorithms XGBoost and CatBoost performed the best on the Bondora Peer-to-Peer Lending dataset. Both algorithms had similar efficacy, except that XGBoost was slightly faster.

For the LendingClub peer-to-peer data, as seen in Table 5.1.6, for extremely low and high-risk threshold values, CatBoost was the best algorithm, followed by XGBoost. Also, in Table 5.1.5, we can see that both these algorithms were top-performing for a moderate risk threshold value and had the highest average F1 scores.

These results show that boosting algorithms such as CatBoost and XGBoost perform well on peer-to-peer lending datasets across all thresholds, especially at low and high extremes, regardless of their class balancing. A financial institution dealing in peer-to-peer lending may also consider testing boosting algorithms to develop an early warning system against potential loan defaults, given that they have significantly lower average training and prediction times than unsupervised and semi-supervised methods.

5.2.3 Time Considerations

In some cases, depending on the system configuration, data size, and speed requirements, it may be prudent to consider using an algorithm that has slightly lower efficacy in favour of speed.

For example, in Table 5.1.5, we observe that DevNet is the best-performing algorithm, followed by DeepSAD. DevNet performs only slightly better than DeepSAD. However, the average training time of DeepSAD is over 15 seconds faster than that of DevNet. If the system has limited computational resources or the data size is significantly larger, the time savings can be significant and not worth the slightly better efficacy of DevNet.

Fig. 5.2.2 shows the average training and prediction times for all the algorithms

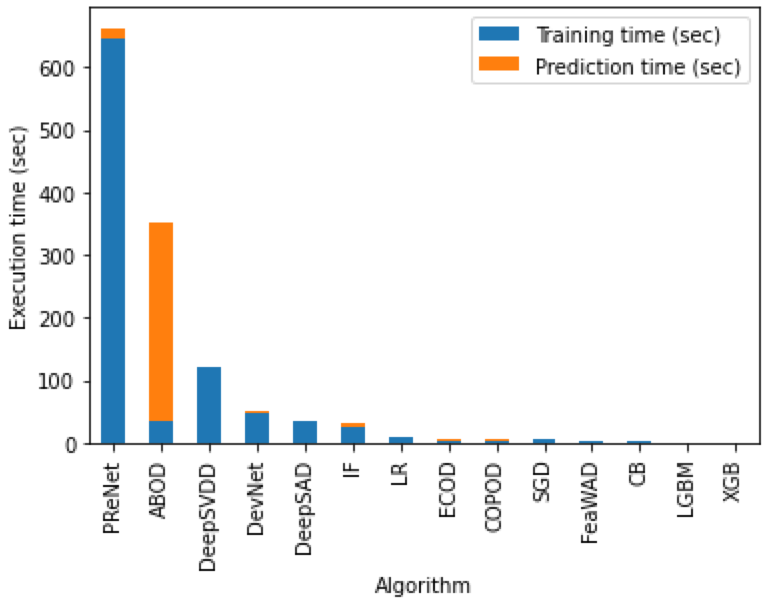

Fig. 5.2.2: Average training and prediction time comparison for all algorithms on the LendingClub dataset

on the LendingClub dataset. The LendingClub dataset was chosen for this diagram because the subset of this dataset used in our experiments is considerably large. Other datasets show the times in a similar proportion. Almost all algorithms except most deep learning ones and ABOD had comparatively low average training and prediction times. FeaWAD was the only deep learning approach that was relatively fast. ABOD had the highest average prediction time. PReNet was the slowest algorithm concerning training time and had the second-largest prediction time.

5.2.4 Availability of Class Label in the Data

Sometimes, the class label information that states that the loan is in default may not be available. In that case, supervised and semi-supervised approaches cannot be used as they rely on this information to build the model. In such scenarios, it makes sense to use the best available unsupervised anomaly detection approach instead of the best approach if the best approach requires a class label.

For example, consider the metrics for the LendingClub dataset in Tables 5.1.5 and 5.1.6. We observe that, across varying risk threshold values, semi-supervised methods such as DevNet and DeepSAD and supervised methods such as CatBoost and XGBoost, on average, perform very well in this scenario. However, if the loan default class label is unavailable in the training set, the next-best unsupervised approach, which, in this case, is isolation forest, may be used in the early warning system instead.

In this chapter, we examined the results and realized their implications. We discussed how results varied with different risk threshold values. We understood how different methods work in different contexts. Finally, we analyzed the results and considered the suitability of various methods in different situations. The next chapter will conclude our thesis by describing how we successfully contributed to research in building an early warning system against loan defaults using anomaly detection algorithms and discuss the future potential of our work.

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we initially defined the need for an early warning system to manage potential loan defaults. We hypothesized that anomaly detection is effective in developing such a system. We motivated the study by discussing the limitations of current literature in this context and describing the contribution of our work.

We first proposed an experimental setup for evaluating different kinds of supervised, unsupervised, and semi-supervised algorithms for the early detection of loan defaults. We then proposed a holistic early warning system architecture using the optimal algorithm determined by the evaluation process.

We stated that the goals of a holistic approach to comparing various approaches should consider a large number of evaluation metrics to confidently compare various approaches, test how fast these methods provided results, test them on multiple types of different loan default datasets, ensure that the methods are probabilistic and can adapt to different risk threshold values and consider multiple approaches by a comprehensive evaluation of different methods.

We achieved these goals by considering different evaluation metrics, including accuracy, precision, recall, F1 score, and area under the ROC curve. We also considered performance metrics such as training and prediction times. We tested the approaches on four distinct datasets with different characteristics and representing different types of loans. We performed our tests for different risk threshold values considering regular moderate and also extreme risk threshold values by including a low 30% and a high 70% risk threshold in our testing. We also compared fifteen different algorithms, five of each supervised, unsupervised, and semi-supervised class.

Using the results obtained from our evaluation, we successfully demonstrated that for most unbalanced loan default datasets, anomaly detection algorithms are helpful in probabilistically predicting the potentiality of defaults for early warning systems. We also analyzed these results under different contexts considering balanced versus unbalanced sets, peer-to-peer loan datasets versus other loan data, time and resource considerations, and how the availability of the loan default class label in the available data would impact our choice of algorithm.

We successfully concluded that anomaly detection approaches such as ECOD and COPOD work well for unbalanced datasets. We also concluded that supervised learning demonstrated significantly better performance for balanced datasets. We also discovered how boosting approaches work well for predicting potential defaults in peer-to-peer loan data, especially for extreme risk threshold values. Finally, we understood why the resources and time considerations of an organization and the availability of the loan default class label in the data source might impact our choice of algorithm.

These findings can be used in fields other than finance as well. For example, the concepts detailed in this study can be used to create an early warning system to predict if a student will fail, if a business will lose a customer, or if a patient may require surgery.

6.2 Limitations and Future Work

Firstly, our proposed work and experimental evaluation focused on using anomaly detection for early warning systems in large datasets and on models with low training and execution times. As such, it does not test combinations of methods, such as using CNNs with LightGBM, as such methods are computationally much more expensive than standard ones. Such models have shown promise in synthetically balanced datasets [11]. Future research may involve focusing on a combination of methods regardless of execution time, as some financial institutions may find value in such methods if they improve predictive outcomes.

Secondly, during our experimental evaluation, we discovered that on peer-to-peer lending datasets, boosting algorithms performed well, especially on extremely low or high risk threshold values. Conducting more experiments on different peer-to-peer datasets considering various risk threshold values may increase confidence in this conclusion.

Finally, regression models that return a probability of loan default were not included in the study due to the difficulty in evaluating them on existing data with binary class labels as the independent variable instead of a numeric probability. Future work may test using such probabilistic regression models for early warning systems instead of using anomaly detection subject to the availability of datasets that provide numeric probability values as the independent variable instead of a default/non-default class label. It is also important to note that it is challenging to compute a value representing the probability of loan default in real-time financial data. It may not be feasible for a financial institution to make these values available for frequent retraining of the model as described in our early warning system architecture. Nevertheless, future research in this direction may result in better optimization of existing methods.

Future research in constructing early warning systems to warn against potential loan defaults may consider combinations of models instead of independent models, focus on evaluating boosting as a viable approach for peer-to-peer lending data, and consider research in probabilistic regression models that predict a numeric probability of loan default instead of determining the likelihood that the class label of a data point is a loan default.

REFERENCES

[1] *2023 U.S. Lev Loan Default Forecast Raised to 2.0%-3.0%; 2024 Projected at 3.0%-4.0%.* URL: `https://www.fitchratings.com/site/pr/10213716` (visited on 03/13/2023).

[2] Asror Nigmonov and Syed Shams. "COVID-19 pandemic risk and probability of loan default: evidence from marketplace lending market". In: *Financial Innovation.* Vol. 7. 1. Springer Science and Business Media LLC, 2021. DOI: `10.1186/s40854-021-00300-x`.

[3] *Canada's biggest banks set aside $2.5 billion to cover an expected wave of loan defaults.* URL: `https://www.thestar.com/business/2023/03/03/canadas-big-six-banks-set-aside-25-billion-as-they-prepare-for-credit-losses.html` (visited on 03/13/2023).

[4] Javier Mulero Chaves and Tomaso De Cola. "Public Warning Applications: Requirements and Examples". In: *Wireless Public Safety Networks 3.* Ed. by Daniel Câmara and Navid Nikaein. Elsevier, 2017, pp. 1–18. ISBN: 978-1-78548-053-9. DOI: `10.1016/B978-1-78548-053-9.50001-9`.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (2009). ISSN: 0360-0300. DOI: `10.1145/1541880.1541882`. URL: `https://doi.org/10.1145/1541880.1541882`.

[6] *CIBC - Annual Report 2022.* URL: `https://www.cibc.com/content/dam/cibc-public-assets/about-cibc/investor-relations/pdfs/quarterly-results/2022/ar-22-en.pdf` (visited on 03/13/2023).

[7] *Royal Bank of Canada - Annual Report 2022*. URL: https://www.rbc.com/investor-relations/_assets-custom/pdf/ar_2022_e.pdf (visited on 03/13/2023).

[8] Huadong Qiu, Ying Tu, and Yan Zhang. "Anomaly detection for power consumption patterns in electricity early warning system". In: *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*. 2018, pp. 867–873. DOI: 10.1109/ICACI.2018.8377577.

[9] Partha Mukherjee and Youakim Badr. "Detection of Defaulters in P2P Lending Platforms using Unsupervised Learning". In: *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. 2022, pp. 1–5. DOI: 10.1109/COINS54846.2022.9854964.

[10] Congjun Rao, Ying Liu, and Mark Goh. "Credit risk assessment mechanism of personal auto loan based on PSO-XGBoost Model". In: *Complex & Intelligent Systems*. Springer Science and Business Media LLC, 2022. DOI: 10.1007/s40747-022-00854-y.

[11] Qiliang Zhu et al. "Loan Default Prediction Based on Convolutional Neural Network and LightGBM". In: *International Journal of Data Warehousing and Mining (IJDWM)*. Vol. 19. 1. IGI Global, 2023, pp. 1–16.

[12] Yu Song et al. "Loan default prediction using a credit rating-specific and multi-objective ensemble learning scheme". In: *Information Sciences*. Vol. 629. 2023, pp. 599–617. DOI: 10.1016/j.ins.2023.02.014.

[13] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[14] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[15] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

ACM, 2016. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

[16] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[17] Liudmila Prokhorenkova et al. *CatBoost: unbiased boosting with categorical features*. 2019. arXiv: 1706.09516 [cs.LG].

[18] Yue Zhao, Zain Nasrullah, and Zheng Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.

[19] Xu, Hongzuo. *DeepOD: Python Deep Outlier/Anomaly Detection*. URL: https://github.com/xuhongzuo/DeepOD.

[20] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[21] François Chollet et al. *Keras*. https://keras.io. 2015.

[22] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[23] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[24] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[25] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[26] Songqiao Han et al. *ADBench: Anomaly Detection Benchmark*. 2022. arXiv: `2206.09426 [cs.LG]`.

[27] Peter McCullagh and John A. Nelder. *Generalized Linear Models*. Chapman and Hall, 1983. ISBN: 0412238500.

[28] J. Kiefer and J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. ISSN: 00034851.

[29] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: `10.1109/ICDM.2008.17`.

[30] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. "Angle-Based Outlier Detection in High-Dimensional Data". In: *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, 444–452. ISBN: 9781605581934. DOI: `10.1145/1401890.1401946`.

[31] Zheng Li et al. "ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions". In: *IEEE Transactions on Knowledge and Data Engineering* (2022). DOI: `10.1109/tkde.2022.3159580`. URL: `https://doi.org/10.1109/tkde.2022.3159580`.

[32] Zheng Li et al. "COPOD: Copula-Based Outlier Detection". In: *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020. DOI: `10.1109/icdm50108.2020.00135`. URL: `https://doi.org/10.1109/icdm50108.2020.00135`.

[33] Lukas Ruff et al. "Deep One-Class Classification". In: July 2018.

[34] Yue Zhao and Maciej Hryniewicki. "XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning". In: (Nov. 2019).

[35] Yingjie Zhou et al. "Feature Encoding With Autoencoders for Weakly Supervised Anomaly Detection". In: *IEEE Transactions on Neural Networks and*

Learning Systems 33.6 (2022), pp. 2454–2465. DOI: 10.1109/tnnls.2021.3086137. URL: https://doi.org/10.1109/tnnls.2021.3086137.

[36] Lukas Ruff et al. *Deep Semi-Supervised Anomaly Detection*. 2020. arXiv: 1906.02694 [cs.LG].

[37] Guansong Pang et al. *Deep Weakly-supervised Anomaly Detection*. 2023. arXiv: 1910.13601 [cs.LG].

[38] Guansong Pang, Chunhua Shen, and Anton van den Hengel. *Deep Anomaly Detection with Deviation Networks*. 2019. arXiv: 1911.08623 [cs.LG].

[39] Manu Siddhartha and Bondora. “Bondora Peer-to-Peer Lending Data”. In: IEEE Dataport, 2020. DOI: 10.21227/33kz-0s65.

[40] L&T Finance. “L&T Vehicle Loan Default Prediction Data”. In: Kaggle, 2019. URL: https://www.kaggle.com/datasets/mamtadhaker/lt-vehicle-loan-default-prediction.

[41] LendingClub. “Lending Club 2007-2020Q3 Dataset”. In: Kaggle, 2020. URL: https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1.

[42] Pankaj Kumar Jadwal et al. “Improved resampling algorithm through a modified oversampling approach based on spectral clustering and SMOTE”. In: *Microsystem Technologies* 28.12 (2022), pp. 2669–2677. ISSN: 1432-1858. DOI: 10.1007/s00542-022-05287-8. URL: https://doi.org/10.1007/s00542-022-05287-8.

[43] Deloitte and MachineHack. “Deloitte-MachineHack Loan Default Prediction Dataset”. In: Kaggle, 2021. URL: https://www.kaggle.com/datasets/hemanthsai7/loandefault.

[44] Matthias Feurer and Frank Hutter. “Hyperparameter optimization”. In: *Automated machine learning: Methods, systems, challenges* (2019), pp. 3–33.

[45] James Bergstra and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization”. In: 13.null (2012), 281–305. ISSN: 1532-4435.

[46] G Pekhimenko. “Penalizied logistic regression for classification”. In: *Dept. Comput. Sci.., Univ. Toronto, Toronto, ON M5S3L1* (2006).

[47] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[48] Detlof Winterfeldt and Ward Edwards. *Decision Analysis and Behavioral Research.* Cambridge University Press, 1993.

[49] Richard Hahnloser et al. “Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit”. In: *Nature* 405 (July 2000), pp. 947–51. DOI: 10.1038/35016072.

[50] *Metrics to Evaluate your Machine Learning Algorithm.* URL: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234 (visited on 03/13/2023).

VITA AUCTORIS

NAME:	Rayhaan Pirani
PLACE OF BIRTH:	Surat, India
YEAR OF BIRTH:	1997
EDUCATION:	Jawaharlal Nehru Technological University, B.Tech in Computer Science and Engineering, Hyderabad, India, 2018
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2023