From No Labels to Informed Decisions: Improving Active and Semi-Supervised Learning Strategies for Financial Crime Detection

by

Bahar Emami Afshar

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Computer Science (Concentration in Applied Artificial Intelligence)

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© Bahar Emami Afshar, Ottawa, Canada, 2025

Examining Committee

The following served on the Examining Committee for this thesis.

Carleton Member: Anil Somayaji
Associate Professor, School of Computer Science
Carleton University

Internal Member(s): Herna L. Viktor
Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Supervisor(s): Paula Branco
Associate Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Detecting fraud and financial crimes is a critical yet challenging task due to extreme class imbalance, high cost of labeling, and often a complete lack of labeled data. Traditional supervised learning approaches often fail in such settings, especially when labeled fraud cases are scarce or unavailable. This thesis addresses these limitations by proposing an adaptive pipeline that combines unsupervised learning, Semi-Supervised Learning (SSL), and Active Learning (AL) techniques for effective decision-making in scenarios ranging from no labels to limited labeled data, progressively acquired under a budget constraint.

We begin with a comparative analysis of AL and SSL under different imbalance ratios and resampling strategies. While AL outperforms SSL in certain cases, it is often more costly and less effective in highly imbalanced settings. To overcome this, we propose X-ITERADE (EXplainable ITERative Anomaly Detection Ensemble), an explainable, fully unsupervised anomaly detection ensemble that uses dynamic clustering to identify and rank suspicious cases. X-ITERADE produces a subset of transactions with a significantly higher fraud rate and includes an explainability module that uses Large Language Models (LLM)s to assist domain experts in understanding anomalous behavior. Building on X-ITERADE's output, we introduce ALISA (Adaptive Learning through Iterative Semi-supervised Active learning), a novel framework that adaptively transitions between AL and SSL based on model signals and dataset characteristics. Using as a starting point the suspicious cases identified by X-ITERADE, ALISA combines weighted AL and enhanced self-training with data augmentation through an iterative process. ALISA enables starting from minimal or even one-class labeled sets, addressing the cold-start problem in early-stage fraud detection.

Together, X-ITERADE and ALISA form a robust, adaptive pipeline that bridges the gap from unlabeled data to strategic querying and enhanced fraud detection, enabling informed, cost-efficient decisions throughout the learning process.

Acknowledgements

I would like to express my heartfelt thanks to my supervisor, Dr. Paula Branco, for her continuous guidance, support, and encouragement throughout my studies. I am also grateful to our industry partner, H3M, for their collaboration and valuable contributions, which played a key role in the development of this work. Finally, I would like to thank my family for their unwavering love, patience, and support, which have meant everything to me during this journey.

Table of Contents

List of Tables x

List of Figures xii

Acronyms xviii

1 Introduction 1

1.1 Motivation 1

1.2 Problem Statement and Solution Overview 2

1.3 Research Objectives 4

1.4 Key Contributions 6

1.5 Publications and Posters 7

1.5.1 Conference Publications 7

1.5.2 Poster Presentations and Awards 8

1.6 Thesis Structure 9

2 Background and Literature Review **10**

2.1 Introduction . . . 11

2.2 Background on Active Learning and Semi-supervised Learning . . . 13

2.3 State of the Art Review in Fraud Detection . . . 17

2.3.1 Definitions and Key Concepts in Financial Crime . . . 17

2.3.2 Review of the Existing Literature . . . 18

2.3.2.1 Anomaly Detection . . . 19

2.3.2.2 Active Learning Approaches . . . 22

2.3.2.3 Semi-Supervised Learning Approaches . . . 24

2.4 State of the Art Review in Malware Detection . . . 28

2.5 Conclusion . . . 31

3 To Label or Pseudo-Label? A Comparative Study of Active and Semi-Supervised Learning in Imbalanced Scenarios **32**

3.1 Introduction . . . 33

3.2 Methodology . . . 36

3.2.1 Datasets . . . 36

3.2.2 Learning Algorithms . . . 39

3.2.3 Evaluation Methodology . . . 40

3.3 Result and Discussion . . . 43

3.3.1 Fraud Detection . . . 43

3.3.2 Malware Detection . . . 47
3.3.3 Limitation and Guidelines . . . 49
3.3.3.1 Limitations: . . . 49
3.3.3.2 Guidelines and Lessons Learned: . . . 49
3.4 Conclusion . . . 50

4 X-ITERADE: EXplainable ITERative Anomaly Detection Ensemble 58
4.1 Introduction . . . 59
4.2 X-ITERADE: EXplainable ITERative Anomaly Detection Ensemble . . . 62
4.2.1 X-ITERADE . . . 62
4.2.2 Explainability Module . . . 65
4.3 Experimental Settings . . . 69
4.3.1 Datasets . . . 69
4.3.2 Evaluation Methodology . . . 71
4.4 Results and Discussion . . . 76
4.4.1 Experimental Results of X-ITERADE . . . 76
4.4.2 Explainability Module Results . . . 79
4.4.3 Limitations . . . 87
4.5 Conclusion . . . 89

5 ALISA: Adaptive Learning through Iterative Semi-supervised Active learning 91
5.1 Introduction . . . 92

5.2 ALISA: Adaptive Learning through Iterative Semi-supervised Active learning . . . 94
5.3 Experimental Settings . . . 103
5.3.1 Datasets . . . 103
5.3.2 Evaluation Methodology . . . 105
5.4 Results and Discussion . . . 109
5.4.1 Limitations . . . 120
5.5 Conclusion . . . 121

6 Conclusion **123**
6.1 Overview of the Thesis . . . 123
6.2 Limitations . . . 125
6.3 Future Work . . . 127
6.4 Final Remarks . . . 129

References **130**

APPENDICES **144**

A Appendix A **145**

B Appendix B **147**

C Appendix C **154**

List of Tables

2.1 Overview of key characteristics from the related literature in Anomaly Detection (AD), AL, SSL, highlighting their applications and potential for addressing challenges in the financial domain, particularly in Anti-money laundering (AML) and fraud detection. 25

3.1 Main characteristics of the 24 training datasets used. 38

3.2 Hyper Parameter Tuning Information for Different Models Used in Fraud Detection. 42

3.3 Hyper Parameter Tuning Information for Different Models Used in Malware Detection. 42

4.1 Main characteristics of the datasets used for X-ITERADE. 69

4.2 Definitions of Merchant Category Groups Based on Code Ranges 70

4.3 Hyperparameters for Different Models and Configurations - CCT Dataset . 72

4.4 Hyperparameters for Different Models and Configurations - BAF Dataset . 73

4.5 Experimental configurations of X-ITERADE Algorithm. We appended "MR" (Most Repeated) to the X-ITERADE variants to indicate that we focused on the anomalous cases flagged by X-ITERADE that consistently appeared across all iterations. 73

4.6 Experimental results for CCT datasets across different imbalance ratios, with average performance reported for each budget bucket. 81

4.7 Experimental results for BAF datasets across different imbalance ratios, with average performance reported for each budget bucket. 82

4.8 Krippendorff's Alpha and Intra-Class Correlation Coefficient (ICC) for each model and metric 86

4.9 Brief Comparison of LLMs Performance and Practical Considerations . . . 88

5.1 Fraud and non-fraud distribution across different suspicious case subsets for BAF and CCT datasets, obtained from X-ITERADE and used as input to the autoencoder. 104

5.2 Fraud and non-fraud distribution across different subsets for BAF and CCT datasets, obtained from the autoencoder and used for training and testing ALISA. 104

5.3 Hyperparameters in the ALISA Framework 107

5.4 Summary of experimental configurations for ALISA 108

5.5 Best-performing settings across ALISA hyperparameters 117

B.1 Example prompt used for the explainability module with CCT dataset (SV2-1) 147

B.2 Example prompt used for the explainability module with BAF dataset (SV2-1) 150

List of Figures

1.1 Overview of the proposed two-stage framework: X-ITERADE selects suspicious cases via unsupervised AD; ALISA iteratively expands them using label-efficient learning. An integrated XAI module summarizes group behaviors using SHapley Additive exPlanations (SHAP) values, decision trees, Association Rule Mining (ARM), and LLMs. 5

3.1 Experimental results for BAF dataset across different imbalance ratios. . . 44

3.2 Experimental results for CCT dataset across different imbalance ratios. . . 45

3.3 Recall results for BAF dataset and Random Forest as baseline, across various imbalance domains. 52

3.4 Recall results for CCT dataset and Random Forest as baseline, across various imbalance domains. 52

3.5 Experimental results per number of iterations for the CCT dataset under different imbalance settings using XGBoost without resampling. 53

3.6 Performance comparison of Random Forest (RF) across queries and iterations. Darker colors indicate a better rank. 54

3.7 Performance comparison of XGBoost across queries and iterations. Darker colors indicate a better rank. 55

3.8 F1 results for RF (100 queries; 15 iterations). 56

3.9 F1 results for XGBoost (100 queries; 15 iterations). 56

3.10 Recall results for RF (60 queries; 25 iterations). 57

3.11 Recall results for XGBoost (60 queries; 25 iterations). 57

4.1 Comparison baselines with best X-ITERADE results on CCT dataset for the 4 budgets considered. Each row shows a different budget in decreasing order from top to bottom. Each column shows one of the variants of the CCT dataset - from left to right: Original, Low, Med, and High. (Blue bars: anomalies detected in baseline; dashed blue bars: anomalies detected in X-ITERADE; the red line shows, on the right y-axis, the number of positive (fraud cases) contained in the signaled cases.) 80

4.2 X-ITERADE results across all iterations for CCT original imbalance dataset. 83

(a) Number of fraud cases and most repeated fraud cases detected per iteration for the CCT original imbalance dataset. 83

(b) Number of anomalies and most repeated anomalies detected per iteration for the CCT original imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range. . 83

4.3 Comparison of LLM average performance across 20 test examples and 3 human annotators from the CCT and BAF datasets using six qualitative and linguistic evaluation metrics. 84

4.4 Krippendorff's alpha for each model across evaluation metrics. 87

5.1 Overview of ALISA's architecture and data flow, starting from the suspicious cases flagged by X-ITERADE and proceeding through its iterative learning process. 95

5.2 ALISA execution modes: Parallel strategy (Left) and Sequential strategy (Right). 99

5.3 Progressive F1 Gain over Initial Performance (Best Approach Per Dataset Variant): Each color represents the ALISA method that achieved the best performance on a specific dataset variant. Solid and dashed bars correspond to CCT and BAF datasets, respectively. 110

5.4 F1 Gain over Best Baseline (Best Approach Per Dataset Variant): Each color represents the ALISA method that achieved the best performance on a specific dataset variant. Solid and dashed bars correspond to CCT and BAF datasets, respectively. 111

5.5 Precision Gain over Best Baseline (Best Approach in Terms of F1_Score Per Dataset Variant): Each color represents the ALISA method that achieved the best F1_Score on a specific dataset variant. Solid and dashed bars correspond to CCT and BAF datasets, respectively. 111

5.6 Distribution of F1_Scor0e (Top) and Precision (Bottom) gains for each dataset variant. The violin plots represent the performance across all ALISA approaches compared to the initial step (left) and the best-performing baseline (right). Dashed horizontal lines indicate the performance of baseline methods for each dataset. 112

5.7 Progressive F1 Gain of All ALISA Approaches and the Baselines Compared to Initial Performance on CCT and BAF Datasets. Lighter colors indicate higher F1 gains, representing better performance. 113

5.8 Aggregated F1_Scores compared to baselines for the BAF dataset across different imbalance ratios. Approaches aggregated include Gaussian resampling, Normal SSL, and sequential and parallel execution modes. . . . 115

5.9 Aggregated F1_Scores compared to baselines for the CCT dataset across different imbalance ratios. Approaches aggregated include Gaussian resampling, dynamic queries, and sequential and parallel execution modes. . 115

5.10 Average Rank of ALISA Methods and the Baselines on CCT Datasets (Lower Values Indicate Better Performance) 118

5.11 Average Rank of ALISA Methods and the Baselines on BAF Datasets (Lower Values Indicate Better Performance) 118

5.12 Frequency of ALISA Methods Appearing Among the Top-3 Performing Approaches on BAF Datasets. 119

5.13 Frequency of ALISA Methods Appearing Among the Top-3 Performing Approaches on CCT Datasets. 119

A.1 Accuracy results for RF (150 queries; 10 iterations). 146

A.2 Accuracy results for XGBoost (150 queries; 10 iterations). 146

C.1 Experimental results for BAF datasets across different imbalance ratios, with best performance reported for each budget bucket. 155

C.2 Experimental results for CCT datasets across different imbalance ratios, with best performance reported for each budget bucket. 156

C.3 X-ITERADE results across all iterations for BAF high imbalance dataset. . 157

(a) Number of fraud cases and most repeated fraud cases detected per iteration for the BAF high imbalance dataset. 157

(b) Number of anomalies and most repeated anomalies detected per iteration for the BAF high imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range. 157

C.4 X-ITERADE results across all iterations for BAF original imbalance dataset. 158

(a) Number of fraud cases and most repeated fraud cases detected per iteration for the BAF original imbalance dataset. 158

(b) Number of anomalies and most repeated anomalies detected per iteration for the BAF original imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range. . 158

C.5 X-ITERADE results across all iterations for BAF med imbalance dataset. . 159

(a) Number of fraud cases and most repeated fraud cases detected per iteration for the BAF med imbalance dataset. 159

(b) Number of anomalies and most repeated anomalies detected per iteration for the BAF med imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range. 159

C.6 X-ITERADE results across all iterations for BAF low imbalance dataset. . 160

(a) Number of fraud cases and most repeated fraud cases detected per iteration for the BAF low imbalance dataset. 160

(b) Number of anomalies and most repeated anomalies detected per iteration for the BAF low imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range. 160

C.7 X-ITERADE results across all iterations for CCT high imbalance dataset. 161

(a) Number of fraud cases and most repeated fraud cases detected per iteration for the CCT high imbalance dataset. 161

(b) Number of anomalies and most repeated anomalies detected per iteration for the CCT high imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range. 161

C.8 X-ITERADE results across all iterations for CCT med imbalance dataset. . 162

(a) Number of fraud cases and most repeated fraud cases detected per iteration for the CCT med imbalance dataset. 162

(b) Number of anomalies and most repeated anomalies detected per iteration for the CCT med imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range. 162

C.9 X-ITERADE results across all iterations for CCT low imbalance dataset. . 163

(a) Number of fraud cases and most repeated fraud cases detected per iteration for the CCT low imbalance dataset. 163

(b) Number of anomalies and most repeated anomalies detected per iteration for the CCT low imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range. 163

Acronyms

AD Anomaly Detection x, xii, 2, 5, 9, 10, 12, 18, 25, 26, 51, 58

AI Artificial Intelligence 4, 11, 60

AL Active Learning iv, x, 2–7, 9, 10, 12–14, 16, 18, 22–35, 39, 41, 43, 44, 47–51, 58, 60, 90–95, 97–99, 105–107, 120, 121, 123, 124, 126, 128, 129

AML Anti-money laundering x, 10–12, 18, 25, 33, 59

API Application Programming Interface 30, 73, 84, 90

ARM Association Rule Mining xii, 4, 5, 7, 9, 59, 61

DBSCAN Density-Based Spatial Clustering of Applications with Noise 64, 66, 71, 72, 78

ECOD Empirical Cumulative Distribution Functions 71, 72, 77

GAN Generative Adversarial Network 20, 27, 49, 50

GMM Gaussian Mixture Models 20, 25, 63

GSSL Graph-based Semi-Supervised Learning 49, 126, 128

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise 65, 66, 71, 72

ICC Intra-Class Correlation Coefficient xi, 75, 79, 85, 86

ICL In-Context Learning 67, 71, 79, 89

IForest Isolation Forest 21, 22, 26, 63–66, 71, 72

IR Imbalance Ratio 69, 71

LIME Local Interpretable Model-Agnostic Explanations 67

LLM Large Language Models iv, xii, xiii, 4, 5, 7, 9, 59, 67, 68, 70, 73, 74, 79, 84, 87, 89, 90, 129

LOF Local Outlier Factor 21, 26, 65, 66, 71, 72

ML Machine Learning 2, 11, 34, 38, 60, 61

MLP Multilayer Perceptron 39

MTLD Measure of Textual Lexical Diversity 74

NB Gaussian Naïve Bayes 39

NLP Natural Lanugage Processing 26

OCSVM One-Class Support Vector Machine 22, 106

PCA Principal Component Analysis 21, 25

PDF Portable Document Format 28

QBC Query By Committee 14, 16, 26, 27, 39, 49

RF Random Forest xii, xiii, xv, 39, 42, 43, 46, 49, 54, 56, 57, 146

SHAP SHapley Additive exPlanations xii, 4–6, 9, 59, 61, 67, 68, 79, 89, 129

SSL Semi-Supervised Learning iv, x, xv, 2–7, 9, 10, 12, 16, 18, 24, 25, 27, 29–35, 39, 41, 43, 47–51, 58, 60, 90–95, 98, 99, 105–108, 114–117, 120, 121, 123, 124, 126–129

SVC Support Vector Classifier 39

SVM Support Vector Machine 14, 20, 21, 27, 28, 30

t-SNE t-distributed stochastic neighbor embedding 89, 129

VAE Variational Autoencoders 50

XGB XGBoost 39, 42

Chapter 1

Introduction

1.1 Motivation

As digital payments become more common, credit and debit card transactions are now part of everyday life. However, this also means that fraudsters have more opportunities to commit financial crimes. Credit card fraud causes billions of dollars in losses each year, with global losses reaching approximately $33.83 billion in 2023 according to the Nilson Report[1], and can damage public trust in financial systems. Detecting fraud quickly and accurately is more important than ever, especially as criminals develop smarter and harder to detect methods.

One of the biggest challenges in fraud detection is that, while there is a lot of transaction data, very few of those transactions are labeled. Getting labels requires expert analysis, which is costly, slow, and sometimes unclear. Privacy laws and security policies make it even harder to collect this kind of data. Moreover, in many real situations, companies have

[1] Financial World article on credit card fraud

to start detecting fraud without having any labeled cases at all.

From a Machine Learning (ML) point of view, fraud detection is a difficult problem. The data is extremely imbalanced, and there are far more normal transactions than fraud. In many cases, there are no labeled cases available for learning. The data changes over time, and decisions made by models must be explainable so that investigators and analysts can understand and trust them. While supervised ML has worked well in well-prepared settings, these methods often fail when labels are missing or extremely imbalanced, especially in real-world situations like those faced by new fintech companies or small banks.

This leads us to the main question of this work:

How can we build smart, efficient, and explainable fraud detection systems that work from day one, even when we have no or very few labeled examples?

1.2 Problem Statement and Solution Overview

This thesis addresses the challenges of real-world fraud detection by exploring and combining unsupervised AD, SSL, and AL together. The central problem revolves around the limitations of existing approaches in fraud detection scenarios, that are:

- Labeled data is scarce, entirely unavailable, or contains only examples from a single class.
- Data is heavily imbalanced, with fraudulent transactions being extremely rare.
- The risks of missing fraud (false negatives) or falsely flagging customers (false positives) are high.

Earlier efforts to address this problem have focused on SSL and AL techniques. These methods aim to optimize label usage by either pseudo-labeling unlabeled data or selectively querying the most informative samples. While these methods perform well in moderately imbalanced settings, our investigations show they often fail in more extreme cases, especially when the labeled set includes only one class or lacks labels entirely. In such situations, standard learning-based strategies become ineffective. This is particularly concerning in scenarios where fraud detection systems must operate from day one without any labels, a common situation for newly launched fintech platforms.

To address this gap, we propose a two-stage framework illustrated in Figure 1.1.

- X-ITERADE: In the first stage, X-ITERADE, which is an unsupervised anomaly detection ensemble, identifies a small, high-quality subset of suspicious cases within a flexible user-defined budget. It also provides a subset of safe cases that have a very small chance of containing fraud and do not require further inspection. The key goal of X-ITERADE is to select as many fraud cases as possible to be labeled in a completely unsupervised way and provide human-readable explanations for these suspicious and safe behaviors. This provides a strategic entry point for manual inspection and model initialization in the complete absence of labels.

- ALISA: In the second stage, ALISA, which is a label-efficient, iterative learning framework, builds on the suspicious subset identified by X-ITERADE. It begins by using an autoencoder to select the initial training set and acquire a small number of true labels. ALISA then expands the labeled set through iterative learning, combining enhanced weighted AL, SSL, and data augmentation. Its weighted mechanism dynamically balances between AL and SSL, prioritizing the strategy that yields greater performance gains. This enables ALISA to make efficient use of costly true labels

while relying on cheaper pseudolabels whenever possible. The key goal of ALISA is to gradually construct a reliable labeled data set and improve model performance during iterations, even in one-class scenarios where only non-fraudulent cases are initially available.

In addition to building better models, it's also critical that the results are explainable. Real-world stakeholders, from analysts to clients, require intuitive and transparent justifications for automated decisions. Our proposed solution includes a novel explainability module that integrates decision trees, SHAP [64], ARM, and LLMs to generate human-readable descriptions of group behavior. These profiles not only guide the labeling process but also help distinguish meaningful behavioral patterns among customers and transactions.

1.3 Research Objectives

The main objectives of this thesis are as follows:

- To evaluate the strengths and limitations of AL and SSL methods under varying levels of data imbalance and label availability.
- To develop X-ITERADE, an explainable, fully unsupervised anomaly detection ensemble that provides two non-overlapping subsets of cases: 1) a high-quality subset of suspicious cases; and 2) a subset of safe cases that does not require further investigation.
- To incorporate an explainability module that generates actionable and transparent summaries of fraud patterns using interpretable Artificial Intelligence (AI) techniques and LLMs.

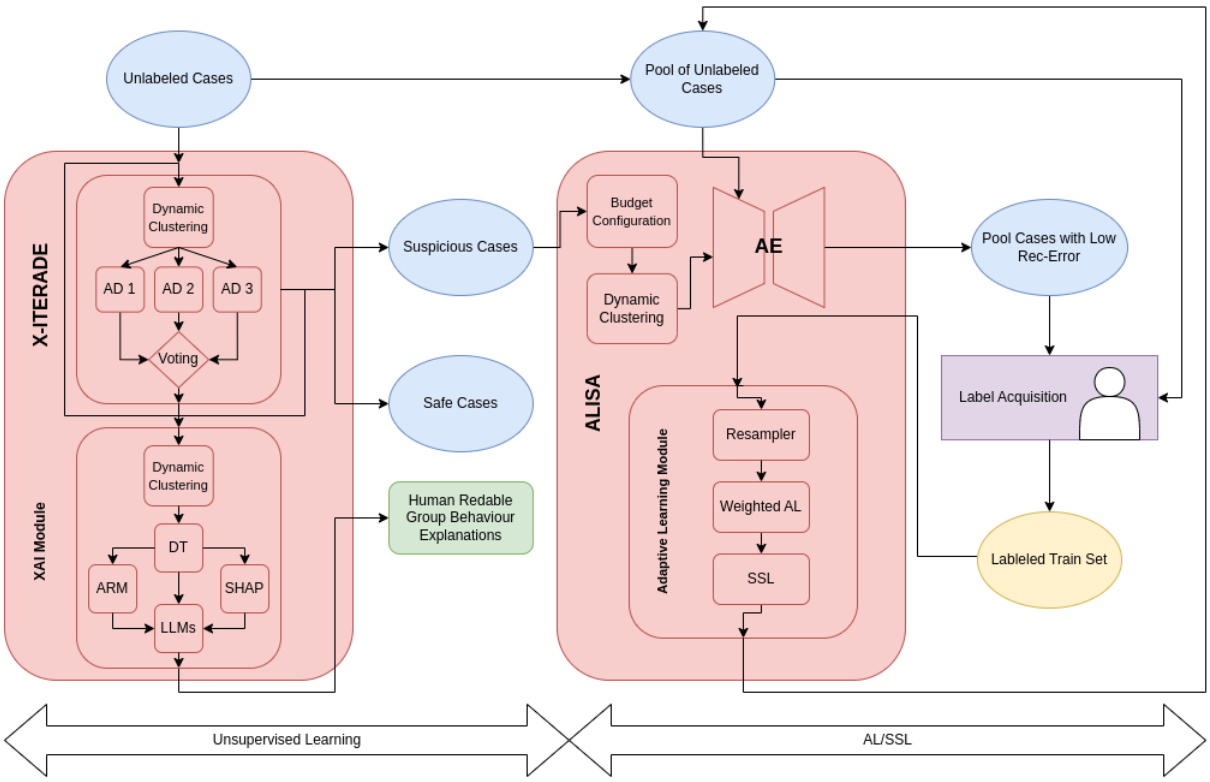


Figure 1.1: Overview of the proposed two-stage framework: X-ITERADE selects suspicious cases via unsupervised AD; ALISA iteratively expands them using label-efficient learning. An integrated XAI module summarizes group behaviors using SHAP values, decision trees, ARM, and LLMs.

- To propose ALISA, a modular, iterative framework that adapts to imbalanced and one-class fraud detection datasets by building on X-ITERADE's output and applying enhanced AL, SSL, and data augmentation.
- To validate the proposed approaches across realistic fraud detection datasets, demonstrating robustness, efficiency, and improved detection rates.

1.4 Key Contributions

This thesis makes several key contributions to the field of fraud detection and learning under limited supervision:

- **Comprehensive Literature Review:** We present an in-depth survey of fraud detection methods classified by the availability of labeled data, covering unsupervised, AL and SSL techniques. The review also extends to malware detection, highlighting similar challenges and drawing connections that inform cross-domain applicability.
- **Empirical Evaluation of AL and SSL Across Multiple Domains and Imbalance Scenarios:** We conduct a systematic evaluation of AL and SSL methods across diverse domains, including fraud and malware detection, spanning datasets with varying imbalance ratios, from perfectly balanced to extremely imbalanced using realistic industrial ratios. Our analysis examines how resampling techniques, imbalance severity, dataset characteristics, and labeling budget configurations impact performance, offering actionable insights for model selection and deployment in real-world settings.
- **X-ITERADE: Unsupervised Anomaly Detection with Adjustable Budgeting:** We introduce X-ITERADE, a budget-aware unsupervised anomaly detection framework that identifies the most suspicious cases in imbalanced datasets. By isolating high-risk instances while preserving a set of confidently safe cases, X-ITERADE helps reduce imbalance and provides a strong starting point for labeling experts.
- **Explainability Module for Cluster Behavior Analysis:** To support interpretability, X-ITERADE includes an explainability module that combines SHAP

value analysis and ARM to extract meaningful patterns from suspicious clusters. Using a local LLM, the module generates human-readable behavioral profiles, helping both domain experts and clients understand the rationale behind detected anomalies. It serves as a practical tool for building risk profiles, justifying decisions, and enhancing trust in high-stakes environments.

- **ALISA: A Modular Framework for Learning with Limited Labels:** We propose ALISA, a flexible and modular framework designed to operate effectively in both one-class and multi-class scenarios. ALISA integrates AL, SSL, and data augmentation in an iterative pipeline to progressively expand labeled datasets and enhance model performance. It introduces a novel weighted AL ensemble, including a one-class query strategy that remains effective when only one class is present. With its multi-parameter design, ALISA can be tailored to a wide range of datasets and operational conditions, making it broadly applicable to real-world fraud detection settings where labeled data is limited and costly to obtain.

1.5 Publications and Posters

1.5.1 Conference Publications

The material presented in this thesis has been accepted and published in three national and international peer-reviewed, double-blind conference proceedings. Additionally, the content of Chapter 5 (ALISA) is currently under submission as a journal paper. Across all these papers, the primary contributions, including the conception of the ideas, the design and implementation of experiments, the analysis, testing, and the preparation of manuscripts, were carried out by me. The co-authors, including my supervisor and our

industry partner collaborators, played an important advisory role by providing guidance and feedback through weekly and bi-weekly meetings, which helped refine and improve the research.

- Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu ITERADE - ITERative Anomaly Detection Ensemble for Credit Card Fraud Detection. In Dino Pedreschi, Anna Monreale, Riccardo Guidotti, Roberto Pellungrini, and Francesca Naretto, editors, Discovery Science, pages 115–133, Cham, 2025. Springer Nature Switzerland.https://doi.org/10.1007/978-3-031-78980-9_8 [4]

- Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu. Is Expert-Labeled Data Worth the Cost? Exploring Active and Semi-supervised Learning Across Imbalance Scenarios in Financial Crime Detection. . In Kamel Adi, Simon Bourdeau, Christel Durand, Valérie Viet Triem Tong, Alina Dulipovici, Yvon Kermarrec, and Joaquin Garcia-Alfaro, editors, Foundations and Practice of Security, pages 174–186, Cham, 2025. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-87496-3_12 [3]

- Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu To Label or to Pseudo Label? Active Learning vs Semi-Supervised Learning for Windows Malware Prediction. Proceedings of the Canadian Conference on Artificial Intelligence. (2024) Retrieved from https://caiac.pubpub.org/pub/r5yzx1s1 [2]

1.5.2 Poster Presentations and Awards

- First Place Poster Award, Digital Transformation of the Society category, Faculty of Engineering Research Day, University of Ottawa, for the poster titled: "ITERADE -

ITERative Anomaly Detection Ensemble for Credit Card Fraud Detection (2025)".

1.6 Thesis Structure

This thesis is organized as follows: Chapter 2 reviews the state-of-the-art in fraud detection and discusses foundational work in AL, SSL, and unsupervised AD approaches. To enhance the generalizability of our study, we also extend this review to related domains such as malware detection, highlighting parallels and differences in how these techniques are applied across various security challenges. Chapter 3 offers a thorough evaluation of multiple AL and SSL methods, examining their performance under conditions of limited labeled data and class imbalance. This analysis spans several domains, including fraud detection and malware detection, providing insights into the strengths and weaknesses of each approach in realistic, challenging scenarios. Chapter 4 introduces X-ITERADE, an explainable iterative anomaly detection ensemble that uses dynamic clustering to separate normal and suspicious cases and allows users to specify a labeling budget. It works in a fully unsupervised way and includes an explainability module that provides clear descriptions of suspicious group behaviors using SHAP, ARM, and LLMs. Chapter 5 presents ALISA, which builds on suspicious cases selected by X-ITERADE and iteratively improves detection through a combination of AL, SSL, and data augmentation. ALISA features a weighted AL ensemble and an enhanced self-training module, designed to work well with sparse labels, imbalanced datasets, and one-class scenarios. Finally, Chapter 6 summarizes the key findings of this work, critically discusses its limitations, and proposes future research directions to further advance fraud detection in label-scarce environments.

Chapter 2

Background and Literature Review

Money laundering enables serious crimes such as terrorism and human trafficking, posing a threat to both financial systems and global security. Traditional AML systems struggle with detecting complex, evolving patterns, especially given the limited availability of labeled data and the rarity of illicit activities. Similar challenges arise in the cybersecurity domain, particularly in malware detection, where labeling is costly and malicious samples are rare. In both domains, the highly imbalanced nature of data and limited supervision hinder the effectiveness of conventional approaches. This chapter reviews AL and SSL as promising approaches to address these issues in both financial crime and malware detection. We explore their theoretical foundations and analyze recent advances in each domain. In the context of fraud detection, we examine recent advances in AD, along with AL and SSL techniques developed to tackle challenges such as data imbalance and limited labeled instances.

2.1 Introduction

The importance of AML efforts goes far beyond just the financial world, as money laundering has a deep impact on global security. Combating this crime is not just about protecting economic systems; it plays a key role in fighting a wide range of criminal activities that rely on illegal financial gains. Money laundering is the hidden enabler for crimes like terrorism, human trafficking, drug trade, and wildlife smuggling. By laundering money, criminals can enjoy the profits of their activities while hiding the true sources of their funds. The scale of the problem is massive, with the United Nations estimating that $2 trillion is laundered globally each year, but authorities recover less than 0.2% of it. [91, 95] Fraud and money laundering are closely linked financial crimes that often occur together. While fraud involves obtaining money or assets through deception, money laundering is the process of making those illicit gains appear legitimate within the financial system. [34] Fraud losses are rising sharply worldwide. In 2023, reported fraud losses in the U.S. alone exceeded $10 billion, a 14% increase from 2022. [90] Globally, eCommerce fraud losses reached $44.3 billion in 2024 and are projected to climb to $48 billion in 2025, marking a 16% year-over-year increase. [25] Yet fraud remains dramatically underreported. In Canada, for instance, only 5% to 10% of frauds are reported, despite Canadians losing over $638 million to fraud in 2024 and total reported losses since 2021 surpassing $2 billion. [22] Fraud detection is more critical than ever. The rapid evolution of fraud tactics, fueled by AI, is rendering traditional detection methods increasingly ineffective. [27] As a result, 75% of eCommerce businesses plan to increase their fraud prevention budgets in 2025, with many allocating at least 20% more. [25]

Traditional rule-based systems used by financial institutions struggle to keep up with the complexity of identifying money launderers and fraudulent activities. While ML offers great potential for improving AML and fraud detection, challenges such as limited labeled

data and difficulty detecting new criminal techniques remain.

Interestingly, similar challenges are present in the cybersecurity domain, particularly in malware detection. As malware attacks continue to rise each year, they pose an urgent global security risk [81]. Both fraud and malware threaten the security, privacy, and integrity of users and systems worldwide. In both domains, data collection is relatively easy, but obtaining accurate labels is resource-intensive and time-consuming. As a result, supervised learning methods often fail due to the scarcity of labeled data. Moreover, both fraud and malware cases tend to be rare, further complicating detection in highly imbalanced datasets.

This thesis aims to address these shared challenges by developing an AL and SSL framework tailored to enhance AML and fraud detection in such imbalanced environments. Our work is an important step toward strengthening financial security, supporting global safety, and fighting international crime. To demonstrate the broader applicability of our approach, we also evaluate its effectiveness in the malware detection domain.

In this chapter, we begin by providing an overview of AL and SSL, along with the specific methods we explored in each category. Section 2.2 presents background on AL and SSL. Sections 2.3 and 2.4 offer a review of the current state of the art in the two key domains addressed in our experiments: fraud detection (Section 2.3) and malware detection (Section 2.4). In the context of fraud detection, we review recent developments in AD as well as AL and SSL approaches. For malware detection, we examine existing solutions, with particular attention to whether AL and SSL methods have been explored. Finally, Section 2.5 concludes the chapter.

2.2 Background on Active Learning and Semi-supervised Learning

AL systems address the labeling bottleneck by asking for queries in the form of unlabeled instances, which are later labeled by an oracle, such as a human annotator. AL's central premise is that a machine learning algorithm can achieve higher accuracy with fewer labeled training instances if it can select its training data [83]. This is achieved through queries, typically in the form of unlabeled data instances, which are then labeled by an oracle, such as a human annotator. Three primary scenarios define query selection [83]:

1. Membership Query Synthesis: generates novel queries and seeks labels from the oracle for these queries;

2. Pool-Based Querying: using a substantial set of instances, the model selects the most optimal instance from the pool set for querying purposes;

3. Stream-Based Selective Sampling: selects a sample of instances, allowing the model to decide whether to retain or discard them.

In our study, we have focused on the pool-based approach for AL in which the model selects the most optimal instances from a large pool set to be queried. To simulate an oracle or user providing true labels for the requested queries, the dataset is typically split into three distinct sets: train, pool, and test set. The train set contains labeled instances used for model training. The pool set functions as the user or oracle, enabling the retrieval of labels for each query requested by the algorithm. The labels are present but are omitted until requested by the algorithm. Finally, the test set is a separate set of unlabeled instances that are used for the evaluation of the model's performance.

Numerous strategies within AL have been proposed for the selection of queries. Among these strategies, Uncertainty Sampling [57] stands out as one of the most extensively used. This method selects data samples where the supervised learner shows the least confidence, targeting those most likely to be misclassified for annotation. In the context of two-class scenarios, uncertainty scores are computed using posterior probability, while for multi-class classification, uncertainty is assessed via information entropy [101].

Query By Committee (QBC) [85] is another well-known AL strategy, where a committee of estimators generates competing hypotheses. Candidate query labels are selected based on the votes cast by each committee member, with the most informative query being the one that triggers the highest disagreement among the committee. The degree of disagreement is commonly measured using two heuristics: vote entropy and the average Kullback–Leibler (KL) divergence [36].

Recent research by Kee et al. [49] argues that integrating diversity and density into QBC-based instance selection significantly enhances model performance. Furthermore, other AL query strategies include Support Vector Machine (SVM)-based [59] and Hashing-based [58] approaches. Additionally, notable techniques for AL query selection include Expected Gradient Length, Reduction of Variance and Fisher Information, and Estimated Error Reduction.

We present further details on the two following querying strategies that we selected for implementation and analysis in our study: uncertainty sampling, and QBC.

The uncertainty sampling approach [83] directs the active learner to query instances where uncertainty about their labels is most prominent. Initially designed for probabilistic learning models, uncertainty sampling in binary classification seeks instances whose posterior probability of being positive is closest to 0.5 [56,57]. For problems with multiple class labels, an extended uncertainty sampling variant might query the instances x^*_{LC}, where the

prediction is least confident according to the following formulation:

$$x^*_{LC} = \arg\max_x \left(1 - P_\theta(\hat{y}|x)\right) \tag{2.1}$$

, where $\hat{y} = \text{argmax}_y P_\theta(y|x)$ is the class label with the highest posterior probability under model θ. This measure reflects the expected 0/1-loss, depicting the model's belief regarding the likelihood of mislabeling instance x.

Two key variants of uncertainty sampling exist: marginal and entropy-based uncertainty sampling. **Marginal uncertainty sampling** addresses the fact that the least confident strategy predominantly focuses on the most probable label, disregarding information about the remaining label distribution. To solve this issue, margin sampling, another multi-class uncertainty sampling variant, considers the margin between the first and second most probable class labels under the model:

$$x^*_M = \arg\min_x \left(P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)\right) \tag{2.2}$$

, where $\hat{y}_1$ and $\hat{y}_2$ are the first and second most probable class labels, respectively. Margin sampling seeks to rectify the deficiency in the least confident strategy by integrating information about the posterior of the second most likely label. **Entropy-based uncertainty sampling** employs Shannon entropy as an uncertainty measure to select instances, i.e., the instances x^*_H are selected according to the formula:

$$x^*_H = \arg\max_x \left(-\sum_i P_\theta(y_i|x) \log P_\theta(y_i|x)\right) \tag{2.3}$$

, where y_i spans all possible labelings. Entropy is a well-known information-theoretic measure that quantifies the amount of information necessary to encode a distribution and

is frequently used as a metric of uncertainty or impurity.

The key idea of the QBC strategy is to maintain a committee $C = \{\theta(1), ..., \theta(C)\}$ comprising models representing competing hypotheses, all trained on the current labeled set L. Each committee member participates in voting for the labeling of query candidates, with the most informative query identified as the instance generating the most disagreement among the committee members. QBC selection algorithm requires the implementation of two fundamental components: (i) construction of the model committee, which should encompass models representing diverse regions within the version space; and (ii) measure the disagreement among committee members.

Regarding SSL, its main goal is to leverage both labeled and unlabeled data to tackle learning tasks, enabling the utilization of abundant unlabeled data alongside smaller sets of labeled data. As opposed to AL methodologies that rely on some degree of human intervention, SSL operates autonomously without any human involvement. A comprehensive survey of SSL is provided by Jesper E. van Engelen and Holger H. Hoos [96].

This survey introduces a novel taxonomy of semi-supervised classification algorithms, covering diverse conceptual and methodological approaches for integrating unlabeled data into the training process.

One SSL category consists of wrapping methods, where supervised algorithms are extended to the SSL setting. These methods first train classifiers on labeled data and use their predictions to generate additional labeled data, subsequently retraining the classifiers on this pseudo-labeled data along with the original labeled data [96]. In our study, we employed two well-known wrapping method algorithms for SSL: self-training and co-training. **Self-training** or "self-learning", is a fundamental pseudo-labeling approach [94]. It involves a single supervised classifier iteratively trained on both labeled data and pseudo-labeled data obtained from previous iterations. Initially, the classifier is trained only on

labeled data. Predictions for unlabeled data points are obtained, and the predictions x^*_{pseudo} exceeding a certain confidence threshold th are added to the labeled dataset, with

$$x^*_{pseudo} = \arg\max_x (P_\theta(\hat{y}|x) > th) \tag{2.4}$$

, where $\hat{y} = \text{argmax}_y P_\theta(y|x) > th$ is the class label with the highest posterior probability above the defined threshold th under model θ. The classifier is then re-trained on the original labeled data and the newly obtained pseudo-labeled data, in an iterative process.

The **co-training** strategy extends self-training to multiple supervised classifiers. This is achieved by iteratively training two or more classifiers on labeled data to get multi-correlated views of the data and incorporating x_{pseudo}, their most confident predictions, into the labeled dataset of the other classifiers in each iteration, with

$$x_{\text{pseudo}} = \arg\max_x \left(\bigcap_{i=1}^{n} y_i \right) \tag{2.5}$$

, where y_i spans all possible labels. In each iteration, instances on which all classifiers agree on the label are added to the labeled set.

2.3 State of the Art Review in Fraud Detection

2.3.1 Definitions and Key Concepts in Financial Crime

Financial crime includes a wide range of illegal activities involving the misuse of financial systems for personal gain. Two major components of financial crime are fraud and money laundering. Fraud refers to deceptive practices used to gain an unfair or unlawful advantage, such as identity theft, payment fraud, or application fraud, while money laundering

involves disguising the origins of illegally obtained money to make it appear legitimate. These two areas often intersect in practice: the proceeds of fraudulent activities frequently need to be laundered to enter the financial system undetected, and fraudulent accounts are commonly used as vehicles for money laundering.

The primary goal of this thesis is to address the challenges associated with fraud detection and AML by applying AL, SSL, and AD techniques. To enhance the generalizability of our methods, we extended our research beyond the financial domain to the cybersecurity domain, specifically focusing on malware detection. Similar to fraud detection, malware detection faces significant challenges, including the ease of collecting large volumes of data, the difficulty of obtaining accurate labels, the poor performance of supervised learning in low-label scenarios, and severe class imbalance, where fraudulent or malicious instances are rare and difficult to detect. These shared characteristics make malware detection a suitable parallel domain for evaluating the robustness and adaptability of the proposed methods. In section 2.4, we will provide a state-of-the-art review of the malware detection domain.

2.3.2 Review of the Existing Literature

Several studies have addressed the challenges of credit card fraud detection and AML. However, most of these approaches rely on supervised techniques, assuming that sufficient labeled data is available. In real-world deployment scenarios, this assumption often does not hold, as transaction data is available, but true labels are typically absent. There are two common scenarios in real-world settings: (1) when the initial dataset lacks labels entirely, which is especially common for new fintech companies that need labels from day one. In such cases, unsupervised approaches are required, and (2) when labeled data is limited, necessitating the use of active and semi-supervised approaches. This literature review explores recent efforts in fraud detection and AML, categorizing them into three

sections based on the techniques used: 1) anomaly detection 2) active learning, and 3) semi-supervised learning.

2.3.2.1 Anomaly Detection

We initiate our exploration of related works and recent endeavors in this area by categorizing anomaly detection into distinct types. Goldstein and Uchida [40], conducted a comprehensive survey, comparing 19 unsupervised anomaly detection algorithms across 10 datasets. Their study categorizes anomaly detection setups into three distinct categories based on available dataset labels.

- Supervised Anomaly Detection: Used when both normal and anomalous instances are fully labeled in the data. However, this approach is limited by imbalanced data and specific classifiers. It is less practical in real-world scenarios where anomalies are not known in advance.
- Semi-supervised Anomaly Detection: Also known as "one-class" classification [Perera et al. [75]], it involves labeled training data with only normal instances, while anomalies appear in the test set. This method trains a model on normal data to predict anomalies based on deviations from the model.
- Unsupervised Anomaly Detection: This setup treats all data as a single set without distinction between training and test sets and requires no labels. The algorithms predict the anomalies based on inherent properties of the data, often using metrics such as distance and density to define the normal and anomalous instances.

In this literature review, we focus on related works in detecting anomalies through unsupervised approaches, as supervised methods are not suitable for our imbalanced domain.

Several works have explored using autoencoders in the fraud detection domain. As an example of one-class anomaly detection, focusing solely on training the model with non-fraudulent cases, is the work of Jiang et al. [47]. The authors introduced an unsupervised attentional framework employing autoencoders and Generative Adversarial Network (GAN)s to address the credit card fraud detection problem. Through experiments on datasets such as the Kaggle Credit Card Fraud Detection Dataset[1] and IEEE-CIS Fraud Detection Dataset [43], they presented a method consisting of an initial autoencoder and a final encoder. To further enhance the learning process, a channel-wise feature attention layer is added between the autoencoder and the encoder to refine the reconstruction error. Also using autoencoders, Kennedy et al. [50] proposed an unsupervised iterative method for detecting fraudulent activities in highly imbalanced data. Their approach involves iteratively separating the majority class from the data. In the final iteration, an autoencoder is trained on the majority class. The reconstruction error is used to identify the minority class cases. However, the experiments in this work are limited to a single dataset, and the criteria for terminating the iterations are not explicitly defined. In extremely imbalanced scenarios, one-class classification methods, such as autoencoders, one-class SVMs, and their deep learning counterparts, can be effectively applied to the entire dataset. However, this approach operates under the assumption that, initially, the dataset consists entirely of normal cases, which is not always ensured. This method has been used in studies like the one by Zong et al. [105], where a deep autoencoder was employed to generate a low-dimensional representation of the data and compute the reconstruction error for each input data point. These features were then fed into a Gaussian Mixture Models (GMM) for further analysis. Other works combine different alternative approaches. For instance, Rezapour [78] proposes using three unsupervised methods—autoencoder, one-class SVM, and robust Mahalanobis outlier detection—on a real credit fraud detection dataset. While autoencoders

[1] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

and one-class SVMs require labels for normal data, Mahalanobis outlier detection can be trained without any labels, aligning with the focus of our investigation. Overall, these methods rely on the assumption that the majority of instances are non-fraudulent, and thus the cases are used to model the patterns in the non-fraudulent class.

Other researchers have focused on using other unsupervised methods. For instance, Lepoivre et al. [55] combine Principal Component Analysis (PCA) and K-Means algorithms and expand the used data with spatial information of the transactions. In [77] an unsupervised Neural Network, an autoencoder, the Local Outlier Factor (LOF), the Isolation Forest (IForest) and the K-Means clustering are compared.

Another interesting approach to tackle fraud detection through cross-dataset analysis was followed by Zhu [104]. The proposed cross-outlier detection involves the analysis of customer behavior through distance metrics and uses clustering techniques to examine neighboring behaviors. This method identifies suspicious cases by investigating each transaction across two datasets.

Data resampling techniques have also been used. Liu et al. [62] introduced two new undersampling ensembles, EasyEnsemble and BalanceCascade, to deal with highly imbalanced datasets in the fraud detection domain. Both EasyEnsemble and BalanceCascade tackle the imbalance by undersampling the majority class. However, these methods are constrained by their reliance on labeled data.

Finally, some studies compare supervised and unsupervised learning methods in the context of credit card fraud detection (e.g., [15, 71]). Generally, these works found that unsupervised learning has a lower performance being less interesting from this perspective. However, unsupervised learning is still considered suitable for credit card fraud detection because no balanced dataset is needed.

A comprehensive systematic review of credit card fraud detection literature in 2023

found that supervised learning algorithms dominate the field, being used in 74% of reviewed articles, while only 12% of the studies used unsupervised learning techniques. The authors suggest that unsupervised and semi-supervised approaches, such as One-Class Support Vector Machine (OCSVM), IForest, and K-means clustering, remain underutilized and warrant further exploration in future research. [13] Overall, research in credit card fraud detection is still not well explored when considering using unsupervised learning techniques. This is a critical application domain where labeled data is often not available, and thus developing more effective solutions is crucial.

2.3.2.2 Active Learning Approaches

In the context of money laundering detection, Deng et al. [28] propose an AL method using Bayesian sequential designs to identify suspicious accounts. Financial institutions typically rely on summary statistics from transaction histories and extensive investigations, but analyzing millions of legitimate transactions daily is challenging. To address this, they use statistical features to prioritize investigations based on suspiciousness, setting threshold hyperplanes where action is needed. Given the diversity of business behaviors in money laundering, tailored hyperplanes are necessary for different categories. The authors develop an AL method combining stochastic approximation and D-optimal designs to select accounts for investigation efficiently. Their method, tested on real banking data, shows effectiveness in minimizing effort while improving detection productivity.

Carcillo et al. [14] feed streaming transaction data into an AL approach, where human investigators label a small subset of cardholders associated with the riskiest transactions each day. Several AL strategies are introduced and evaluated. Various querying strategies, including Highest Risk Querying, Exploratory AL, Stochastic Semi-Supervised (also explored in [100]), and multiple instance learning techniques like Querying by Frequent Un-

certainty and a Combined Function (inspired by [84]), have been systematically compared. The paper also emphasizes an exploration/exploitation trade-off in AL for fraud detection, an aspect not previously explored in the literature. In this study, unsupervised outlier detection performs poorly in credit card fraud detection, especially in scenarios where only a limited number of cards can be manually checked and high precision is vital. Additionally, combining Highest Risk Querying with Stochastic Semi-supervised Learning significantly enhances fraud detection accuracy.

Lablanca et al. [53] tackle the critical need for financial institutions to monitor transactions for AML purposes. While traditional rule-based systems have been supplemented with machine learning-based transaction monitoring systems, these methods also present challenges. Unsupervised models, while effective at detecting novel anomalous patterns, often produce a high number of false alarms. Supervised models, on the other hand, require large amounts of labeled data to achieve high detection rates. To address these issues, the authors introduce Amaretto, a novel AL framework that integrates both unsupervised and supervised learning techniques. The system first uses unsupervised learning to detect potentially problematic activities based on anomaly scores. Then, supervised models are trained with feedback from domain experts (i.e., labeled data) to identify additional data points for further review. The experimental evaluation of Amaretto, conducted on a synthetic dataset simulating client profiles in international capital markets, shows that it outperforms state-of-the-art solutions in reducing money laundering risk and improving detection performance.

Barata et al. [8] present ODAL, an Outlier-based Discriminative Active Learning system for imbalanced datasets, achieving up to 80% better results than random sampling. Tang et al. [87] propose an ensemble-based deep active learning framework that integrates unsupervised and semi-supervised models, reducing false positives and outperforming benchmarks

across multiple domains, including credit card fraud.

ALLIE [26] is a framework for improving active learning on large-scale imbalanced graphs in financial fraud detection. It uses a reinforcement learning agent with an imbalance-aware reward function to efficiently sample from major and minor classes. It incorporates focal loss to boost accuracy for rare classes and uses graph coarsening to cut computational time by up to 38%.

While not specifically in the financial domain, Druck et al. [30] focuses on labeling input features instead of instances by selecting features that offer significant feedback to reduce model uncertainty. Using generalized expectation criteria and a feature-based active learning algorithm, their approach outperforms several baselines.

2.3.2.3 Semi-Supervised Learning Approaches

SSL techniques have been more explored when compared to the application of AL in the credit card fraud detection domain. For instance, several graph-based semi-supervised solutions have been put forward (e.g., [54, 97, 99]). These works typically transform the available information on the transactions into a graph to enable the usage of graph neural networks. Other works have studied the relationship between semi-supervised methods and data augmentation strategies [80]. In a related application on online gambling fraud detection, Generative Adversarial Networks have been used [16].

Researchers have focused on supervised learning techniques for credit card fraud detection making use of the few existing public and labeled datasets on this domain. Still, tackling this problem in a semi-supervised way or through AL is less explored. This is a more complex but more realistic problem from a practical point of view. We could only find a very reduced number of works that use these techniques, but as far as we know, no study

compares them. We target this gap in the literature to provide a better understanding of the advantages and disadvantages of these methods.

Table 2.1 provides an overview of the main characteristics from the related literature in AD, AL, SSL, highlighting their applications and potential for addressing challenges in the financial domain, particularly in AML and fraud detection.

Table 2.1: Overview of key characteristics from the related literature in AD, AL, SSL, highlighting their applications and potential for addressing challenges in the financial domain, particularly in AML and fraud detection.

Ref.	**Year**	**Method**	**Applied to the financial domain?**	**Proposal**
[104]	2006	AD	Yes	Outlier detection model based on cross datasets comparison
[62]	2008	AD	Yes	Data resampling techniques: Easy-Ensemble and BalanceCascade
[40]	2016	AD	No	Comparative survey on unsupervised AD
[55]	2016	AD	Yes	Combining PCA and K-Means
[105]	2018	AD	No	One-class AD using autoencoders and GMM
[78]	2019	AD	Yes	Comparative study of unsupervised and one-class AD
[71]	2019	AD	Yes	Comparative study of unsupervised and supervised methods

Continued from previous page

Ref.	Year	Method	Applied to the financial domain?	Proposal
[77]	2020	AD	Yes	Comparative study of unsupervised AD such as IForest, LOF, autoencoders and neural networks
[15]	2021	AD	Yes	Comparative study of unsupervised and supervised methods
[50]	2022	AD	Yes	Unsupervised iterative AD using autoencoders
[47]	2023	AD	Yes	Unsupervised attentional framework for one-class AD
[85]	1992	AL	No	QBC
[57]	1994	Supervised Learning	No	Heterogeneous uncertainty sampling for supervised learning
[84]	2007	AL	No	Multiple instance AL
[30]	2009	AL	No	AL by Labeling Features in Natural Lanugage Processing (NLP)
[28]	2009	AL	Yes	Bayesian sequential designs
[83]	2009	AL	No	Survey on AL with different query strategies
[36]	2013	AL	No	Instance selection using heuristics
[101]	2016	AL	No	AL using uncertainty information entropy

Continued from previous page

Ref.	Year	Method	Applied to the financial domain?	Proposal
[49]	2018	AL	No	QBC improvement with diversity and density in batch AL
[14]	2018	AL	Yes	Streaming AL strategies
[59]	2021	AL	No	SVM-based online learning
[53]	2022	AL	Yes	Amaretto: an AL framework for money laundering detection
[58]	2022	Unsupervised AL	No	Deep neural network, hash-based
[100]	2011	AL and SSL	No	Stochastic SSL on imbalanced data
[54]	2016	SSL	Yes	Graph-based SSL
[45]	2016	SSL	No	Selection bias with imbalanced classes in SSL
[80]	2018	SSL	Yes	Analysis of SSL in imbalanced domains with data augmentation techniques
[97]	2019	SSL	Yes	SSL with attentive graph neural networks
[96]	2020	SSL	No	Survey on SSL
[16]	2020	SSL	Yes	SSL-based GANs
[99]	2023	SSL	Yes	SSL with temporal transactions graph

2.4 State of the Art Review in Malware Detection

We begin this revision with AL methods that have been used for malware detection. Yuanzhang Li et al. [60] address a significant issue in cybersecurity, particularly in machine learning-based spam and malware detection. These models often struggle with adversarial samples, requiring robustness even with limited labeled data. The focus is on the widespread use of Portable Document Format (PDF) files for malicious activities, which presents challenges due to the labor-intensive labeling process. To overcome this, the authors propose a novel PDF malware detection method utilizing AL to improve training efficiency. They introduce the concept of "uncertain samples" and "random samples" and establish their importance in malware detection through theoretical foundations. Their active-learning-based model employs mutual agreement analysis to select uncertain samples for data augmentation, reducing both training time and the need for large datasets. Experimental results demonstrate that their approach achieves comparable detection performance to traditional methods using only a fraction of the training data.

The challenge of continuous emergence of new malware is tackled by Nir Nissim et al. [70] with the proposal of an AL framework featuring two innovative methods tailored for effectively detecting new malware. This approach targets the limitation of existing techniques in keeping up with updates. Through experiments, their AL methods outperform random selection and existing high-performance approaches in acquiring new malware. AL techniques were also incorporated with SVMs classifiers to address the problem of insufficient labeled data in malware classification in [17] where multiple pool-based AL by learning techniques were used.

Y. Chen et al. [20] address concept drift in Android malware detection through a solution that combines AL with a hierarchical contrastive classifier, leading to significant per-

formance improvements, including a 5% decrease in false negative rates and a 0.38% drop in false positives. In their AL approach, they employ the uncertainty sampling technique, selecting instances using the Pseudo Loss Sample Selector to identify the most uncertain cases.

The challenge of detecting variants or zero-day malware that may evade passive learning models is addressed in [69]. Maintaining a passive learning model with manually updated training data can be both time-consuming and labor-intensive. To automate this process and reduce the need for extensive annotation, the paper introduces an AL approach for malware detection. The authors propose a selector that identifies critical incremental training samples from an open environment. This selector employs multiple criteria, including age, uncertainty, informativeness, and diversity of Portable Executable files, to pinpoint crucial samples. Experimental results demonstrate that the proposed AL approach demands significantly fewer annotated training samples compared to passive learning methods while achieving high accuracy and efficiency. Furthermore, the study conducts experimental comparisons with existing sampling strategies, including query-by-committee and clustering-based diversity, employing single criteria, showing the superior performance of the proposed method in malware detection.

Other works exist that tackle different tasks associated with malware detection through AL, such as multi-label classification [76], while other authors focus more on this problem from a perspective of data streams, handling problems such as concept drift, time-evolving events, and novelty detection (e.g., [41, 68]). Still, these studies do not carry out a comparison between AL and SSL in the context of malware detection.

Regarding advancements that use SSL in the context of malware detection, an innovative approach involving Local and Global Consistency was proposed using SSL for malware prediction [81], leveraging a combination of labeled and unlabeled instances. The authors

demonstrate that this method requires fewer labeling efforts compared to traditional supervised learning while maintaining high accuracy rates. By labeling only 50% of the software, they achieve results with an accuracy above 86%.

In [37] a framework based on malware visualization and SSL is presented. This framework encompasses malware visualization, feature extraction, and a classification algorithm. By directly processing binary files visually and extracting features from gray images, it effectively combines visual image features, alleviating conflicts between different feature variables. Additionally, an enhanced collaborative learning algorithm is introduced to continuously refine the classifier using cost-effective unlabeled samples. The evaluation results on the Microsoft Malware Classification Kaggle dataset [73] achieve an improved classification accuracy while reducing labeling costs compared to conventional methods. A malware category classification system based on semi-supervised deep neural networks is proposed in [65] where deep neural networks are trained in an SSL way with pseudo-labels.

In [29] a pre-train stage is used where generic representations of the malware samples are learned and a fine-tuning stage where labeled data is used for the classification task. The proposed method is based on a graphical representation of the malware samples.

Other works on malware detection exist that focus, for instance, on cloud detection using SSL methods and transfer learning [38] or that use dynamic Android Application Programming Interface (API) call logs to build SSL models [18]. However, these methods do not compare the SSL solutions against AL.

Several researchers also investigate supervised approaches for tackling the malware prediction problem (e.g., [1,7,9]). Among the algorithms that have been applied we find, for instance, gradient boosting, LightGBM, SVM, Random Forests, Decision Trees, Bayesian Networks, Artificial Neural Networks, or K Nearest Neighbors Classifier.

2.5 Conclusion

This chapter has provided a comprehensive background on AL and SSL methodologies, highlighting their relevance to fraud detection in the financial domain. We reviewed the state of the art in fraud detection through three distinct categories based on the availability of labels: anomaly detection for unlabeled data, AL and SSL for limited labeled samples. While the primary focus of this thesis is on enhancing methodologies for the financial domain, we also extended our analysis to the cybersecurity domain, specifically malware detection, to improve the generalizability of our study. Malware detection faces challenges similar to those in fraud detection, such as the ease of collecting large volumes of data contrasted with the difficulty of obtaining accurate labels, and the limited effectiveness of supervised learning when labeled data is scarce. We reviewed recent advancements in malware detection that utilize AL and SSL techniques. In the next chapter, we present a comparative study of AL and SSL across different domains, including fraud and malware detection, to evaluate their performance in imbalanced settings and examine how domain imbalance influences their effectiveness.

Chapter 3

To Label or Pseudo-Label? A Comparative Study of Active and Semi-Supervised Learning in Imbalanced Scenarios

In the previous chapter, we conducted a comprehensive review of AL and SSL methodologies, highlighting their relevance to fraud detection and malware detection, two key cybersecurity domains. We reviewed the state-of-the-art in fraud detection through three distinct categories based on the availability of labels: anomaly detection for unlabeled data, AL, and SSL for limited labeled samples. In this chapter, we provide a comparative study of AL and SSL, focusing on their application in these two domains. Our primary objective is to investigate whether the higher cost and user interaction required in AL consistently yield superior performance compared to SSL in imbalanced scenarios. Additionally, we examine the impact of various resampling strategies on the performance of

AL and SSL. Furthermore, we extend this comparative analysis to the malware detection domain, offering guidelines and investigating the shared challenges and solutions across both domains.

The content of this chapter has been published as two separate papers in the proceedings of the 37th Canadian Conference on Artificial Intelligence (Canadian AI 2024) and the 17th Foundations & Practice of Security (FPS) Symposium 2024.

Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu To Label or to Pseudo Label? Active Learning vs Semi-Supervised Learning for Windows Malware Prediction. Proceedings of the Canadian Conference on Artificial Intelligence. (2024) Retrieved from https://caiac.pubpub.org/pub/r5yzx1s1 [2]

Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu. Is Expert-Labeled Data Worth the Cost? Exploring Active and Semi-supervised Learning Across Imbalance Scenarios in Financial Crime Detection. . In Kamel Adi, Simon Bourdeau, Christel Durand, Valérie Viet Triem Tong, Alina Dulipovici, Yvon Kermarrec, and Joaquin Garcia-Alfaro, editors, Foundations and Practice of Security, pages 174–186, Cham, 2025. Springer Nature Switzerland.https://doi.org/10.1007/978-3-031-87496-3_12 [3]

3.1 Introduction

Fraud and malware detection are critical domains in the field of cybersecurity, facing the common challenge of detecting rare and evolving patterns. While the primary focus of this thesis is to tackle the challenges of AML and fraud detection, we have also extended our study to the malware detection domain due to the inherent similarities between these two fields. Both domains deal with limitedly labeled and imbalanced datasets, where the

instances of fraud or malware are rare compared to legitimate transactions or files. This makes them ideal candidates for evaluating ML approaches that do not rely heavily on labeled data, such as AL and SSL.

In the domain of fraud detection, especially in credit card fraud, ML has shown great potential in identifying fraudulent activities. With electronic commerce and online transactions becoming more prevalent, fraud schemes have grown increasingly sophisticated, making detection more challenging. The Nilson Report 2025 revealed that global payment card fraud losses reached $33.83 billion in 2023, a slight increase from $33.45 billion in 2022. The report highlighted that while the United States accounted for just over 25% of global card spending, it suffered more than 42% of worldwide fraud losses, with $14.32 billion lost to fraud on cards issued in the US. The Nilson Report also projected that cumulative global card fraud losses will exceed $400 billion over the next decade, with card-not-present transactions, especially in ecommerce, remaining the highest risk category [39, 74, 88, 89].

Traditional supervised learning methods have proven to be insufficient, primarily because obtaining labeled data for training models is a time-consuming and labor-intensive process. Hence, there is a growing need for strategies like AL and SSL, which rely less on labeled data. Active learning involves human experts labeling the most uncertain cases, while SSL generates pseudo-labels for the most confident cases. Our research in this chapter seeks to determine whether the cost of human interaction in AL consistently yields better results than SSL, especially in scenarios where labeled data is scarce. We also explore how different budget configurations (i.e., the number of manual labels allowed) affect the performance of these approaches in fraud detection.

In the malware detection domain, the rise of sophisticated malware attacks poses a significant global security threat. Signature-based detection methods, while effective in the past, struggle to keep up with new threats, especially when concept drift occurs. This

problem is compounded by the need for large labeled datasets, which are often unavailable. In this chapter, we assess the use of AL and SSL strategies to detect malware in such settings. Using a well-known malware prediction dataset, we aim to compare the effectiveness of AL and SSL methods, specifically in scenarios where obtaining labeled data is limited. The goal is to determine if the additional cost of user interaction in AL results in better detection performance compared to SSL.

Given the commonalities between fraud and malware detection, namely, the difficulty in obtaining labeled samples, this chapter addresses the following key research questions.

For fraud detection: Does AL, with its human-in-the-loop approach, consistently outperform SSL in detecting credit card fraud? How does the performance change with different budget configurations, and what is the impact of resampling strategies on these learning methods? For malware detection: Can AL, despite its higher labeling costs, provide a substantial advantage over SSL in detecting malware?

Contributions: Our main contributions are as follows:

- Experimental Evaluation Across Domains: We conduct a thorough set of experiments in both fraud and malware detection, comparing the performance of AL and SSL on multiple imbalanced datasets with varying degrees of imbalance. This comparative study provides insights into the effectiveness of these methods across different cybersecurity domains.
- Practical Guidelines: We offer practical insights and guidelines for researchers and practitioners dealing with imbalanced datasets, providing recommendations on the use of AL and SSL strategies based on data availability and labeling costs.

Organization: This chapter is structured as follows: Section 3.2 describes the research methodology, including the datasets and experimental configurations used in both domains.

In Section 3.3 we present the results of the comparative experiments, highlighting key findings and featuring a detailed discussion on the results obtained. Finally, Section 3.4 discusses the conclusions drawn from this study, providing recommendations for future research.

3.2 Methodology

3.2.1 Datasets

We used two datasets designed for credit card fraud detection: the Card Transactions (CCT) [6] Dataset and the Bank Account Fraud (BAF) Dataset [46]. CCT contains around 24 million transactions from 2,000 synthetic U.S. customers over several decades, including about 33,000 fraudulent cases. Despite being synthetic, the dataset closely resembles real-world data in fraud rates, purchase amounts, and Merchant Category Codes (MCCs). To preprocess the data, we split it into training (25%), pool (50%), and test sets (25%) in a stratified manner. We dropped the 'Errors?' column due to missing values, imputed the rest, and encoded categorical columns using hashing. Each partition was normalized to have a mean of 0 and a standard deviation of 1. For feature selection, we removed noisy, redundant, or uninformative features based on correlation with the target and excluded personal details like card numbers and addresses, as they are irrelevant to fraud detection.

This ensured the model could focus on meaningful relationships in the data, improving both performance and interpretability. This careful selection process plays a crucial role in reducing noise, speeding up model training, and ultimately enhancing the quality of the predictions. We engineered new features, including total purchase amount and transaction count per merchant, and the number of cards used by each user. The final dataset has 34

features, with 14 categorical features encoded numerically.

The BAF dataset, consisting of six synthetic bank account fraud datasets, provides a realistic and reliable test bed for evaluating ML algorithms and is built on a contemporary, real-world dataset.

We used the base dataset, with 1 million records and 11,029 fraud cases. The data was split into train, pool, and test sets using the same stratified percentages. With no NaN values, categorical columns were hashed into numerical representations and normalized to a mean of 0 and standard deviation of 1.

We used XGBoost to identify and remove features with zero importance, retaining only impactful ones. Unlike the CCT dataset, no extra aggregated features were created, due to the differences in available information.

The final BAF dataset includes 30 features, 5 of which are categorical and numerically encoded. The predictive task for both datasets is binary, indicating whether a transaction is fraudulent. To simulate real-world conditions, we used imbalance ratios provided by our industry partner H3M: low (0.21), medium (0.077), and extremely high (0.000048), along with the original ratio of each dataset. We created four random subsets without replacement and tested how Random Under Sampling (US) and Random Over Sampling (OS) impacted results. We undersampled by halving non-fraudulent cases and oversampled by doubling fraudulent ones. Table 4.1 shows key characteristics of our training datasets.

For malware detection, we used the Microsoft Malware Prediction Dataset, a resource featured in the Microsoft Malware Prediction Competition hosted on Kaggle [44]. The predictive task associated with this dataset concerns indicating whther a Windows machine is infected by different families of malware. The features are based on different attributes of the machines obtained from telemetry data. This dataset was created using a sampling process intended to address particular business constraints, including privacy

CCT Dataset	Size	IR	Fraud Cases	BAF Dataset	Size	IR	Fraud Cases
Low	28300	0.21	5943	Low	12100	0.21	2542
Low - US	17121	0.34	5943	Low - US	7322	0.34	2542
Low - OS	34243	0.34	11886	Low - OS	14643	0.34	5084
Med	28300	0.077	2180	Med	12100	0.077	932
Med - US	15241	0.14	2180	Med - US	6516	0.14	932
Med - OS	30480	0.14	4360	Med - OS	13032	0.14	1864
XHigh	28300	0.000048	2	XHigh	12100	0.000048	1
XHigh - US	14151	0.00014	2	XHigh - US	6050	0.00016	1
XHigh - OS	28302	0.00014	4	XHigh - OS	12101	0.00016	2
ORG	40239	0.0014	55	ORG	12100	0.01102	134
ORG - US	20148	0.0027	55	ORG - US	6117	0.021	134
ORG - OS	40294	0.0027	110	ORG - OS	12234	0.021	268

Table 3.1: Main characteristics of the 24 training datasets used.

concerns and duration of the machine's operation. The information is compiled from threat and heartbeat reports and comes from Microsoft's Windows Defender endpoint protection application. Each row in this dataset corresponds to a distinct machine that has been given a MachineIdentifier. The ground truth, represented by the target variable HasDetections, indicates if malware was found on the system. This is a binary classification task. To make the data ready for our experiments, we have implemented the following pre-processing steps. We started with partitioning the data into three subsets: training data (25%), pool data (50%), and testing data (25%). Next, we addressed the presence of missing or null values in the dataset. We removed columns that had more than 30% of missing values and imputed the remaining cases (5932 rows) using the mean or mode imputation depending on the feature type. We transformed categorical variables into numerical representations using hashing to facilitate their inclusion in ML algorithms. The numerical features were scaled to have a mean of 0 and a standard deviation of 1, ensuring that all features contribute equally to the model fitting process. We applied feature selection using the feature importance results obtained by XGBoost. We started with a total of 82 features and ended up with 66 features after removing those with zero importance.

3.2.2 Learning Algorithms

To ensure a fair comparison between AL and SSL, we implemented similar algorithms for both approaches. For AL, we used entropy-based uncertainty sampling, marginal uncertainty sampling, and QBC. The QBC consisted of five models: Gaussian Naïve Bayes (NB) [10], RF Classifier [11], Logistic Regression [24], Support Vector Classifier (SVC) [23], and Multilayer Perceptron (MLP) Classifier [42]. We measured the disagreement between the models' predictions using vote entropy (VE), defined as follows: $x^*_{VE} = \arg\max_x \left(-\sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}\right)$, where y_i spans all possible labels, $V(y_i)$ denotes the number of "votes" received by a label among the committee members' predictions, and C represents the committee size. This method extends entropy-based uncertainty sampling to the QBC framework.

For SSL, we implemented self-training and co-training. Given that AL strategies are iterative, we incorporated this iterative approach into the SSL algorithms. In self-training, at each iteration, we selected the most uncertain samples within our budget limit and generated their pseudo-labels. For co-training, we used the same set of five models: NB, RF Classifier, Logistic Regression, SVC, and MLP Classifier. We created two random views of the data with an equal number of features. The first three models were trained on the first view, and the last two models were trained on the second view. We identified the most certain predictions where the total probabilities predicted by the five models exceeded 4.8, and we used the most frequent labels for these predictions as pseudo-labels. For our baselines, we used RF and XGBoost (XGB) [19], integrating them into the AL and SSL frameworks. To prioritize the minority group we also used the class_weight parameter of these models.

To address the imbalance problem in each dataset, we employed random oversampling to double the number of minority cases and random undersampling to halve the number

of majority cases. It is important to note that these resampling techniques were only implemented for the fraud detection domain. In the case of the malware detection dataset, no resampling was performed due to the specific characteristics and inherent balance of the domain and dataset.

3.2.3 Evaluation Methodology

We evaluated a range of performance metrics, including **Accuracy**, **Precision**, **Recall**, and **F1_Score**. Given the highly imbalanced nature of fraud detection, special emphasis was placed on recall and F1_Score, as they better capture performance in the minority (fraudulent) class.

- **Accuracy** measures the overall proportion of correctly classified instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  where TP is the number of true positives, TN is true negatives, FP is false positives, and FN is false negatives.

- **Precision** quantifies the proportion of predicted positive cases that are actually positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

  High precision indicates a low false positive rate, which is desirable in scenarios where false alarms are costly.

- **Recall** (also known as sensitivity or true positive rate) reflects the ability of the

model to correctly identify actual positive cases:

$$\text{Recall} = \frac{TP}{TP + FN}$$

In fraud detection, recall is crucial since missing fraudulent cases (i.e., false negatives) can have serious consequences.

- **F1_Score** is the harmonic mean of precision and recall, providing a balance between the two:

$$\text{F1_Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

It is particularly useful when the class distribution is imbalanced, as it penalizes extreme values of either precision or recall.

By focusing on Recall and F1_Score, our evaluation prioritizes the model's effectiveness in identifying fraudulent transactions while accounting for the trade-off between missed detections and false alarms.

Since the datasets in this domain are highly imbalanced, prioritizing recall is crucial, as missing false negatives can lead to significant costs. For the malware detection dataset, which is perfectly balanced, we applied the same set of performance metrics, providing a consistent basis for comparison across both domains. We addressed two sets of hyperparameters: the internal hyperparameters of the baseline models and the external hyperparameters of our frameworks. For baseline hyperparameter tuning, we used 5-fold cross-validation with grid search. The hyperparameters explored during the tuning phase are detailed in Tables 3.2, and 3.3. Once fine-tuned, these models were integrated into our AL and SSL frameworks.

The external hyperparameters for AL and SSL included the number of iterations and

Model	Parameter	Range
RF	n_estimators	[100, 200, 300]
	max_depth	[5, 10]
	min_samples_split	[2, 5, 10]
	min_samples_leaf	[1, 2, 4]
XGB	learning_rate	[0.01, 0.1, 0.2]
	max_depth	[3, 4, 5]
	n_estimators	[100, 200, 300]
	subsample	[0.8, 0.9, 1.0]
	colsample_bytree	[0.8, 0.9, 1.0]

Table 3.2: Hyper Parameter Tuning Information for Different Models Used in Fraud Detection.

Model	Parameter	Range
RF	n_estimators	Integer(10, 100)
	max_depth	Integer(1, 20)
	min_samples_split	Integer(2, 20)
	min_samples_leaf	Integer(1, 20)
	max_features	Categorical(['sqrt', 'log2'])
XGB	learning_rate	Real(0.01, 1.0, prior='log-uniform')
	max_depth	Integer(1, 20)
	min_child_weight	Integer(1, 10)
	subsample	Real(0.5, 1.0, prior='uniform')
	colsample_bytree	Real(0.5, 1.0, prior='uniform')
	gamma	Real(0, 5.0, prior='uniform')

Table 3.3: Hyper Parameter Tuning Information for Different Models Used in Malware Detection.

the number of queries per iteration. To simulate real-world situations where labeled data is costly, we limited our labeling budget to 1,500 instances, using it in five sets of different experiments as follows: (1) 300 queries per iteration over 5 iterations; (2) 150 queries per iteration over 10 iterations; (3) 100 queries per iteration over 15 iterations; (4) 75 queries per iteration over 20 iterations; (5) 60 queries per iteration over 25 iterations. This allowed

us to evaluate the performance of our models under various query and iteration conditions, reflecting practical constraints on data labeling.

3.3 Result and Discussion

3.3.1 Fraud Detection

We used 2 baseline models across 24 generated datasets with varying imbalances, applying 5 AL and SSL algorithms with 5 budget configurations. Here, we highlight the most important trends observed in the results. The complete set of 1,200 experiments, along with additional performance metrics, is available in the GitHub repository associated with this study at https://github.com/beafshar/AL_vs_SSL_Extreme_Imbalance.

Our main experimental results for F1_Score and Recall across different imbalance domains are presented in Figures 3.1 to 3.5. Figures 3.1 and 3.2 shows the average F1_Score over different budget configurations within their intervals for the five implemented algorithms compared to the fine-tuning model on undersampled, oversampled, and normal datasets with no sampling across various imbalance ratios. For the less imbalanced scenarios (Low and Med), AL methods outperform semi-supervised approaches in F1_Score, while the latter fails to surpass baselines. However, in more imbalanced datasets, the trend reverses. On the BAF dataset with RF as the baseline, SSL-Co-Training performs best on the undersampled data, improving the F1_Score by 1%. For the CCT dataset, all five algorithms perform similarly on low-imbalanced data, but XGBoost-AL-Marginal Uncertainty stands out with an 88% F1_Score, 10% higher than the fine-tuned model.

Figures 3.3 and 3.4 show recall changes across different imbalance ratios for five algorithms with various budgets, using RF as the baseline for BAF and CCT datasets. When

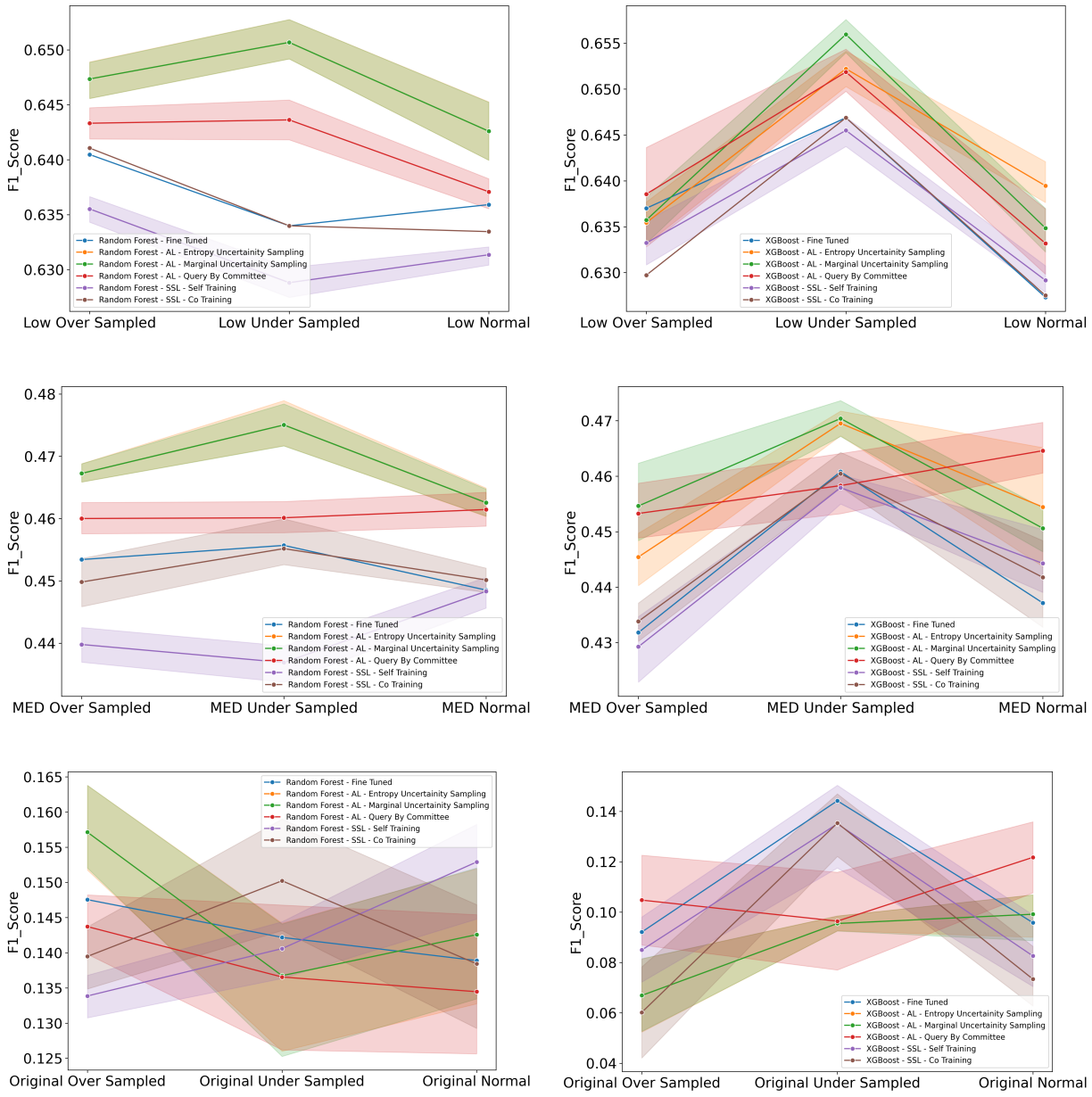


Figure 3.1: Experimental results for BAF dataset across different imbalance ratios.

prioritizing Recall to reduce false negatives in imbalanced domains, semi-supervised methods outperform AL, especially in low to medium imbalance scenarios. The improvements

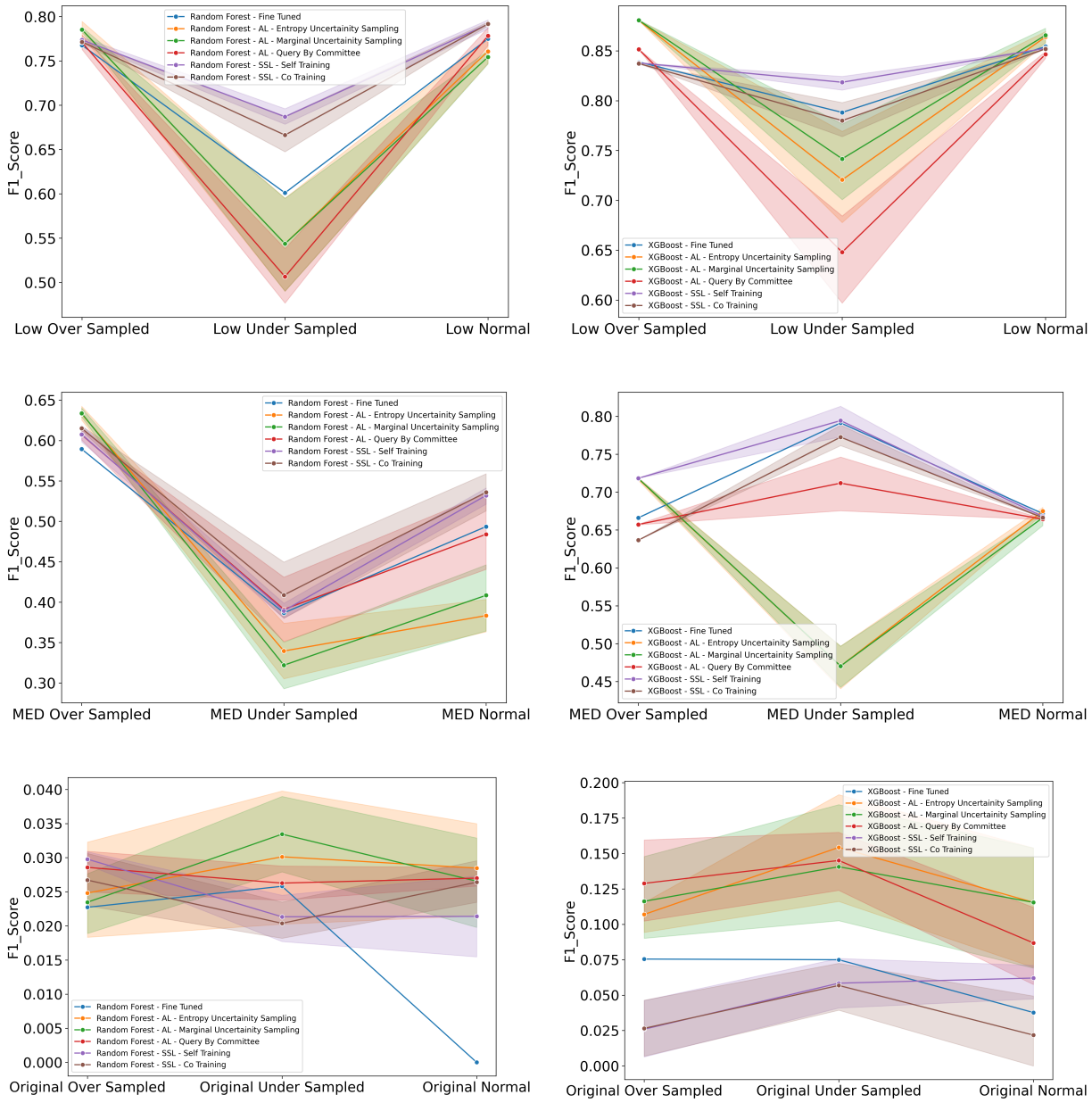


Figure 3.2: Experimental results for CCT dataset across different imbalance ratios.

are greater with undersampling. Across both datasets, RF-SSL-Self Training achieves the highest recall, improving by 5% to 10% over the RF-Finetuned model. Comparing the two

datasets, the tradeoff between iterations and queries per iteration is more crucial for CCT, as its range of changes is wider than BAF. While BAF shows stable recall, CCT exhibits greater fluctuations, making the sampling strategy more impactful. Further analysis of other models revealed that these differences arise from the inherent characteristics of the datasets. The CCT dataset undergoes more extensive feature engineering and preprocessing, which likely explains its higher sensitivity to these factors.

As expected, performance metrics improve as the datasets become less imbalanced, with similar trends observed for recall, F1_Score, and precision. For the extremely imbalanced XHigh dataset, all algorithms failed to detect fraud, resulting in Recall, F1_Score, and Precision of 0. With the BAF original imbalance, recall averages around 14%, 50% for medium imbalance, and 75% for low imbalance.

For recall, oversampling performed best on the CCT dataset, despite introducing duplicates, while undersampling was optimal for the BAF dataset in low and medium imbalance domains. For RF as the datasets became less imbalanced, recall fluctuations across algorithms increased. This trend is consistent across both datasets.

Improvements were more significant on the CCT dataset, likely due to its more extensive preprocessing, additional features, and larger labeled data. Despite the CCT original dataset being 8 times more imbalanced than BAF, it showed a 7% F1_Score improvement, while BAF saw only a 2% improvement. In the low imbalance scenario, CCT's oversampled dataset achieved an average F1_Score of 77%, while BAF's best-performing algorithm, AL-Query by Committee, reached 64%.

As datasets become more imbalanced, the range of F1_Score changes across budget configurations widens, indicating that for less imbalanced datasets, configurations yield similar results, while for more imbalanced datasets, the choice of iterations and queries per iteration is crucial for efficient budget use. This behavior can be seen in Figures

3.3 and 3.4. As shown in Figure 3.5 the results are consistent across iterations for the Low and Medium imbalance datasets, but the Original dataset, being more imbalanced, shows greater fluctuations. In this dataset, optimal performance is typically achieved with medium iterations (15-20), rather than extreme values (5 or 25).

Active learning and SSL effectively detect credit card fraud with limited labeled data. Sampling techniques, such as random oversampling and undersampling, improved F1_Score by 2% to 15%. Semi-supervised learning, in particular, enhanced recall, suggesting it may be a better alternative to the resource-intensive and time-consuming AL in some cases.

3.3.2 Malware Detection

Tables 3.6 and 3.7 present the detailed results across multiple performance metrics when using as base learners the Random Forest and the XGBoost, respectively. Overall, we verify that both the AL methods and SSL methods significantly enhanced model performance in all metrics evaluated. When considering the different number of iterations and number of queries used to achieve the defined budget, we observe that the optimal approach seemed to reside in a balanced trade-off between the number of queries and the number of iterations. This balance allowed for a meaningful augmentation of the training set while enabling the model to benefit from the iterative nature of AL and SSL methodologies. Such an equilibrium proved essential in efficiently utilizing the query budget while ensuring the model's robust learning from newly acquired labeled instances.

It became evident that using a small number of queries across long iterations had minimal impact on significantly enlarging the training set. Conversely, using an extensive number of queries within a short span of iterations limited the model's ability to fully leverage the AL and SSL nature of obtaining and incorporating labeled data.

Figures 3.8, 3.9, 3.10 and 3.11, show the detailed results (F1_Score and Recall) of the learning iterations as more instances are added to the training set either through AL or SSL methods. Accuracy results can be found in Figures A.1, A.2 in Appendix A. When using Random Forests, entropy uncertainty sampling emerged as the optimal strategy, achieving the highest accuracy with 10 iterations of 150 queries, resulting in a 1.5% Accuracy boost. Additionally, self-training in random forest yielded the best F1_Score with 25 queries per 60 iterations, showcasing a 3% increase in Accuracy.

For XGBoost, query by committee proved superior, boasting a 1% increase in F1_Score with 15 iterations of 100 queries. When observing Accuracy, the best strategy was marginal uncertainty sampling. It was interesting to observe that increasing iterations led to improved recall for co-training in XGBoost, showing a 2% increase in Recall while self-training in random forest exhibited a remarkable 9% increase in recall compared to the fine-tuned model before applying SSL. While recall undergoes significant fluctuations, Precision remains noticeably stable, indicating that both SSL and AL are effectively decreasing false negatives. These results underscore the importance of the underlying model architecture and performance, with XGBoost, as a boosting ensemble, demonstrating comparatively smaller performance gains compared to Random Forest, which operates as a bagging ensemble. The remaining plot for all the other metrics and the different combinations of iterations and queries tested are available in the repository associated with this study at https://github.com/beafshar/AL_vs_SSL_Malware_Detection.

Our study demonstrated that employing AL and SSL techniques effectively enhances performance when limited labeled instances are available. With just 1500 instances, around 5% of the pool, and 10% of the labeled training set (15,000 instances), these methods showed promising results, highlighting their efficiency in resource-constrained scenarios.

However, the effectiveness of AL and SSL heavily relies on the performance of the

baseline model employed. The impact of AL and SSL is less pronounced if the base model is already performing well. We observed that XGBoost had a more stable and robust performance than Random Forest in our dataset and experiments, leading to a smaller effect from these learning approaches.

3.3.3 Limitation and Guidelines

3.3.3.1 Limitations:

The fraud detection study focused on two sampling strategies; exploring more techniques and imbalance ratios could yield deeper insights. We used two synthetic datasets and 24 subsets for fraud detection and only one dataset for malware detection, incorporating more diverse datasets would help validate the framework. We only tested Random Forest and XGBoost, representing a bagging and a boosting ensemble; including more models would provide a better assessment of AL and SSL. This study compared AL and SSL using the most comparable algorithms, such as uncertainty sampling vs. self-training, and QBC vs. co-training. Future work should explore advanced, computationally intensive semi-supervised methods like Graph-based Semi-Supervised Learning (GSSL) and GANs.

3.3.3.2 Guidelines and Lessons Learned:

The impact of different budget configurations on the F1 score becomes more pronounced as the datasets become more imbalanced (Figure 3.5). For highly imbalanced datasets optimal performance appears to occur at medium iterations. Given the increased sensitivity of these imbalanced domains to the budget settings, it is crucial to analyze various configurations to determine the optimal setup. For the BAF dataset and RF model, both AL-Entropy and AL-Marginal Uncertainty Sampling selected the same indices in each iteration, resulting in

similar performance. This was not observed with other models or datasets, suggesting that the most uncertain samples were identical. To differentiate these approaches, additional feature engineering and preprocessing with aggregate features may be helpful. For the extremely imbalanced dataset, none of the algorithms detected fraud, resulting in zero performance. In such cases, unsupervised anomaly detection methods are typically more effective than AL or SSL, which rely on limited labeled data. Traditional methods like undersampling and oversampling have limited effectiveness. A more promising solution for these scenarios is to employ synthetic data generation and data augmentation techniques, such as GANs and Variational Autoencoders (VAE)s. These methods can produce large volumes of high-dimensional data, but they are computationally expensive.

3.4 Conclusion

This study provides a comparative analysis of AL and SSL in two critical domains – fraud and malware detection – under varying imbalance ratios and data scarcity conditions.

For fraud detection, our results demonstrate that the effectiveness of AL and SSL largely depends on the imbalance ratio and the specific metrics being evaluated. In less imbalanced domains, AL achieved the highest F1_Score, showing its overall effectiveness. However, SSL excelled in Recall, an essential metric in imbalanced datasets, as it prioritizes minimizing false negatives and detecting rare fraudulent instances. In highly imbalanced domains, the performance gap between AL and SSL diminished, with SSL occasionally outperforming AL. This suggests that, while AL is resource-intensive and time-consuming, SSL can be a more efficient and equally effective alternative in many cases. Moreover, as datasets became more imbalanced, both F1_Score and Recall declined. We observed that applying sampling techniques and adjusting the imbalance ratio modestly could enhance

performance across subsets. However, detecting fraudulent cases in extremely imbalanced datasets remains a persistent challenge, often requiring unsupervised anomaly detection. Budget configurations in AL must be carefully optimized, especially in highly imbalanced scenarios, as the impact of budget utilization becomes more pronounced.

For malware detection, our experiments highlighted the potential of both AL and SSL to overcome the challenge of limited labeled data. Even with the modest utilization of just 5% of our unlabeled dataset, both techniques significantly enhanced malware detection performance, showcasing their ability to maximize the value of sparsely labeled data. While we initially anticipated that AL, with its human-driven label acquisition, would outperform SSL, our findings revealed a more complex reality. In several scenarios, SSL demonstrated unexpected strengths, sometimes surpassing AL in terms of overall effectiveness. This suggests that factors beyond the mere availability of human-labeled data, such as data quality, the presence of noise, synthetic data generation, and the adaptability of models, play critical roles in determining the success of these learning approaches.

In conclusion, the choice between AL and SSL should be context-dependent, taking into consideration the imbalance ratio, available resources, and the specific performance metrics that are most critical to the task at hand. While AL may excel in certain domains, SSL can offer a competitive and often more efficient alternative, especially in scenarios where labeling is expensive or data is highly imbalanced.

In the next chapter, we plan to address the extremely imbalanced scenarios that neither AL nor SSL were able to resolve. We will apply an unsupervised AD technique that provides labeling experts with a starting point by isolating the most suspicious cases from the normal ones in a completely unsupervised manner. This technique reduces the dataset's imbalance and helps surface high-priority cases for further investigation. Additionally, the algorithm will include an explainability module to better characterize the suspicious cases.

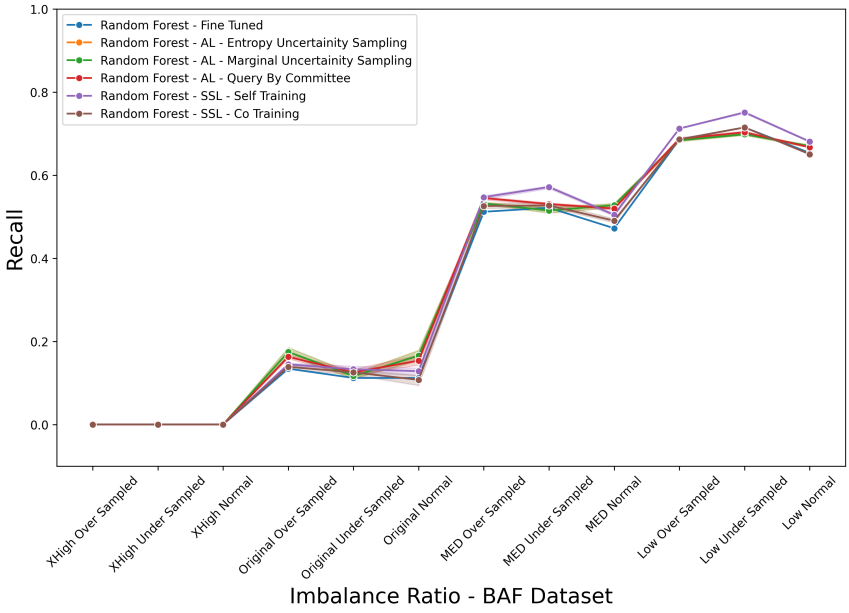


Figure 3.3: Recall results for BAF dataset and Random Forest as baseline, across various imbalance domains.

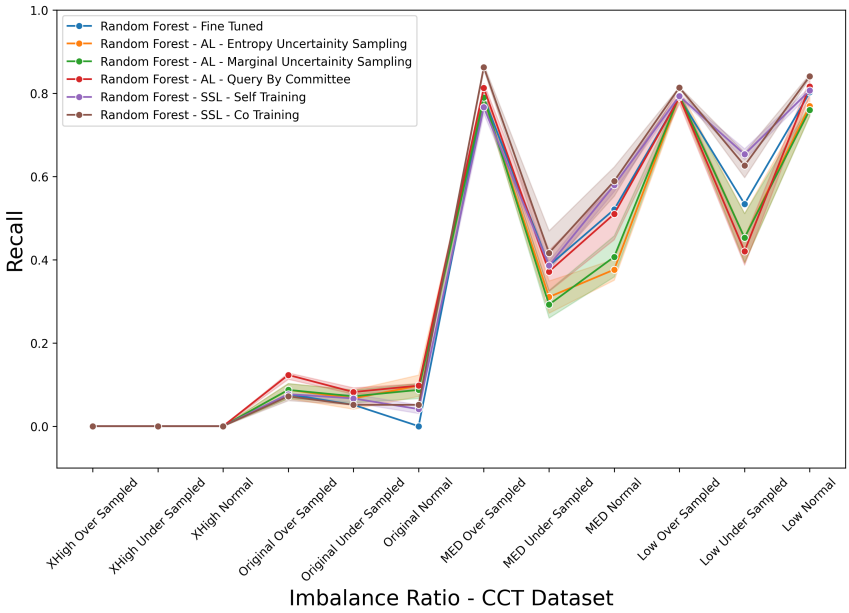


Figure 3.4: Recall results for CCT dataset and Random Forest as baseline, across various imbalance domains.

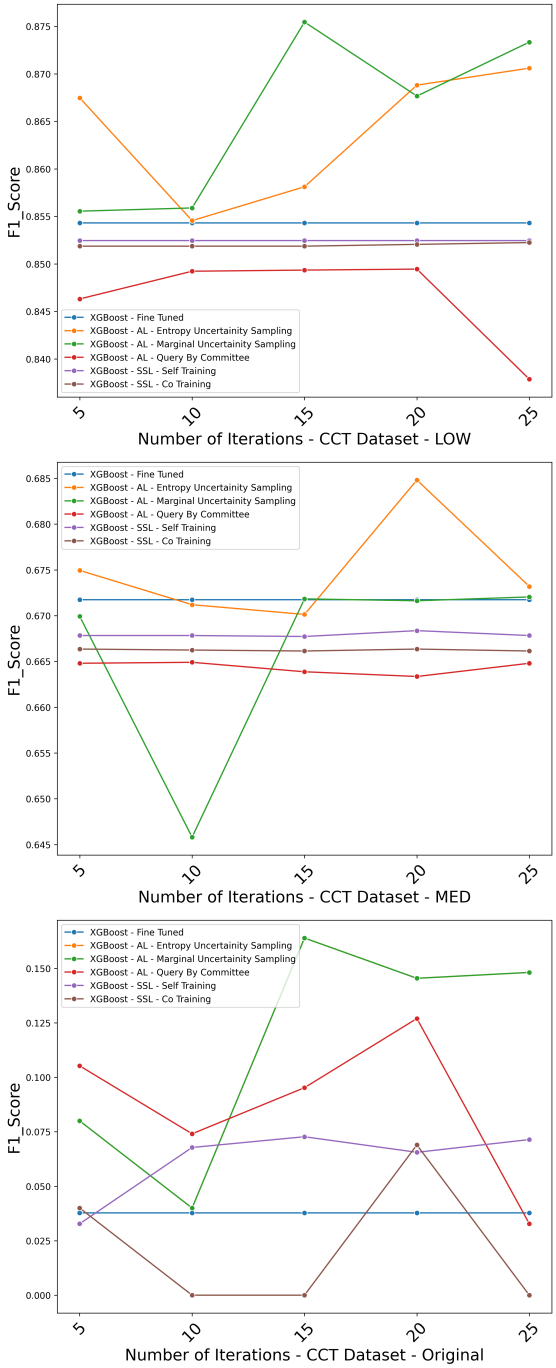


Figure 3.5: Experimental results per number of iterations for the CCT dataset under different imbalance settings using XGBoost without resampling.

Method	Accuracy	F1_Score	Precision	Recall	N_iters	N_queries
Random Forest	0.58950	0.58789	0.58932	0.58646	-	-
Random Forest - Fine Tuning	0.59454	0.60804	0.58762	0.62993	-	-
Random Forest - AL - Entropy Uncertainity Sampling	0.59741	0.61350	0.58911	0.64001	5	300
Random Forest - AL - Marginal Uncertainity Sampling	0.59865	0.61621	0.58957	0.64537		
Random Forest - AL - Query By Committee	0.59512	0.61201	0.58670	0.63961		
Random Forest - SSL - Self Training	0.59329	0.62885	0.57756	0.69014		
Random Forest - SSL - Co Training	0.59565	0.60849	0.58893	0.62940		
Random Forest - AL - Entropy Uncertainity Sampling	0.60061	0.61554	0.59254	0.64040	10	150
Random Forest - AL - Marginal Uncertainity Sampling	0.59617	0.61028	0.58885	0.63333		
Random Forest - AL - Query By Committee	0.59597	0.61377	0.58707	0.64302		
Random Forest - SSL - Self Training	0.59630	0.62236	0.58385	0.66632		
Random Forest - SSL - Co Training	0.59584	0.61014	0.58847	0.63346		
Random Forest - AL - Entropy Uncertainity Sampling	0.59467	0.61578	0.58450	0.65061	15	100
Random Forest - AL - Marginal Uncertainity Sampling	0.59526	0.61402	0.58601	0.64485		
Random Forest - AL - Query By Committee	0.59702	0.61438	0.58819	0.64302		
Random Forest - SSL - Self Training	0.59290	0.62658	0.57797	0.68412		
Random Forest - SSL - Co Training	0.59558	0.60920	0.58853	0.63137		
Random Forest - AL - Entropy Uncertainity Sampling	0.59591	0.60720	0.58985	0.62561	20	75
Random Forest - AL - Marginal Uncertainity Sampling	0.59676	0.61038	0.58961	0.63267		
Random Forest - AL - Query By Committee	0.59460	0.60483	0.58910	0.62142		
Random Forest - SSL - Self Training	0.59369	0.61325	0.58428	0.64524		
Random Forest - SSL - Co Training	0.59702	0.60235	0.59362	0.61134		
Random Forest - AL - Entropy Uncertainity Sampling	0.59872	0.61519	0.59012	0.64249	25	60
Random Forest - AL - Marginal Uncertainity Sampling	0.59356	0.61182	0.58471	0.64158		
Random Forest - AL - Query By Committee	0.60022	0.61742	0.59114	0.64616		
Random Forest - SSL - Self Training	0.59349	0.63655	0.57488	0.71305		
Random Forest - SSL - Co Training	0.59238	0.60825	0.58464	0.63385		

Figure 3.6: Performance comparison of RF across queries and iterations. Darker colors indicate a better rank.

Method	Accuracy	F1_Score	Precision	Recall	N_iters	N_queries
XGBoost	0.58173	0.58943	0.57793	0.60139	-	-
XGBoost - Fine Tuning	0.59820	0.60891	0.59225	0.62652	-	-
XGBoost - AL - Entropy Uncertainity Sampling	0.59892	0.60788	0.59373	0.62273	5	300
XGBoost - AL - Marginal Uncertainity Sampling	0.59911	0.61009	0.59298	0.62822		
XGBoost - AL - Query By Committee	0.59793	0.61347	0.58983	0.63909		
XGBoost - SSL - Self Training	0.59728	0.61049	0.59027	0.63215		
XGBoost - SSL - Co Training	0.59597	0.60589	0.59053	0.62207		
XGBoost - AL - Entropy Uncertainity Sampling	0.59820	0.61029	0.59162	0.63019	10	150
XGBoost - AL - Marginal Uncertainity Sampling	0.60055	0.61174	0.59422	0.63032		
XGBoost - AL - Query By Committee	0.59859	0.61225	0.59127	0.63477		
XGBoost - SSL - Self Training	0.59532	0.60934	0.58811	0.63215		
XGBoost - SSL - Co Training	0.59604	0.59970	0.59344	0.60610		
XGBoost - AL - Entropy Uncertainity Sampling	0.59793	0.61024	0.59128	0.63045	15	100
XGBoost - AL - Marginal Uncertainity Sampling	0.59983	0.61141	0.59337	0.63058		
XGBoost - AL - Query By Committee	0.60088	0.61511	0.59310	0.63883		
XGBoost - SSL - Self Training	0.59643	0.60726	0.59055	0.62495		
XGBoost - SSL - Co Training	0.59924	0.60768	0.59429	0.62168		
XGBoost - AL - Entropy Uncertainity Sampling	0.59800	0.60709	0.59281	0.62207	20	75
XGBoost - AL - Marginal Uncertainity Sampling	0.59800	0.61023	0.59138	0.63032		
XGBoost - AL - Query By Committee	0.59826	0.61063	0.59156	0.63097		
XGBoost - SSL - Self Training	0.59519	0.60508	0.58981	0.62115		
XGBoost - SSL - Co Training	0.59878	0.60468	0.59506	0.61461		
XGBoost - AL - Entropy Uncertainity Sampling	0.60107	0.61460	0.59361	0.63713	25	60
XGBoost - AL - Marginal Uncertainity Sampling	0.59637	0.61113	0.58874	0.63529		
XGBoost - AL - Query By Committee	0.59820	0.61079	0.59139	0.63150		
XGBoost - SSL - Self Training	0.59441	0.60602	0.58832	0.62482		
XGBoost - SSL - Co Training	0.59937	0.61716	0.59011	0.64681		

Figure 3.7: Performance comparison of XGBoost across queries and iterations. Darker colors indicate a better rank.

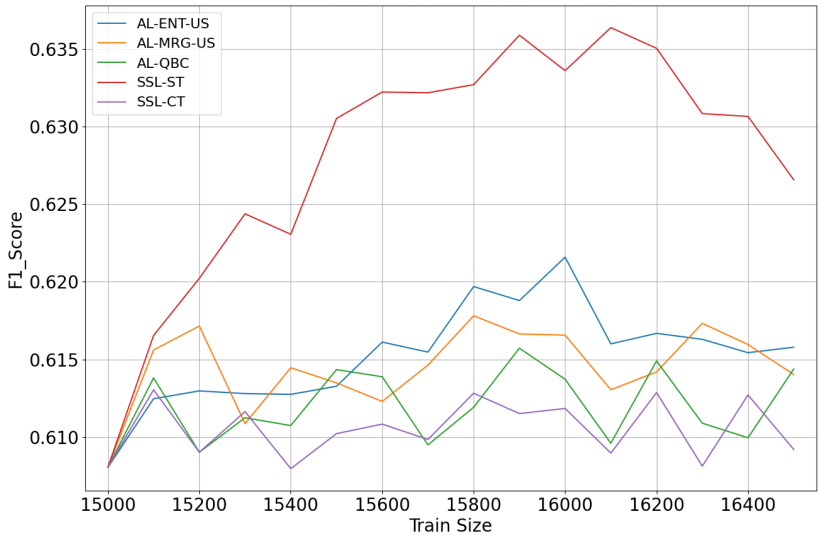


Figure 3.8: F1 results for RF (100 queries; 15 iterations).

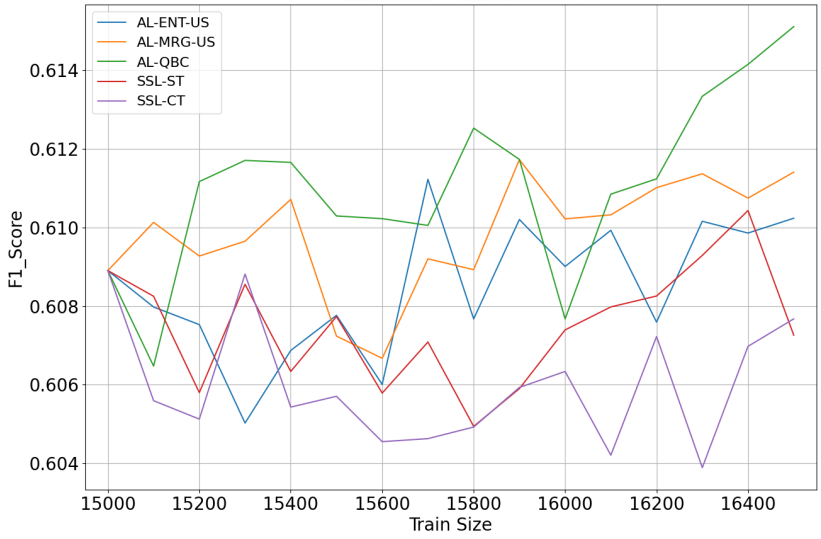


Figure 3.9: F1 results for XGBoost (100 queries; 15 iterations).

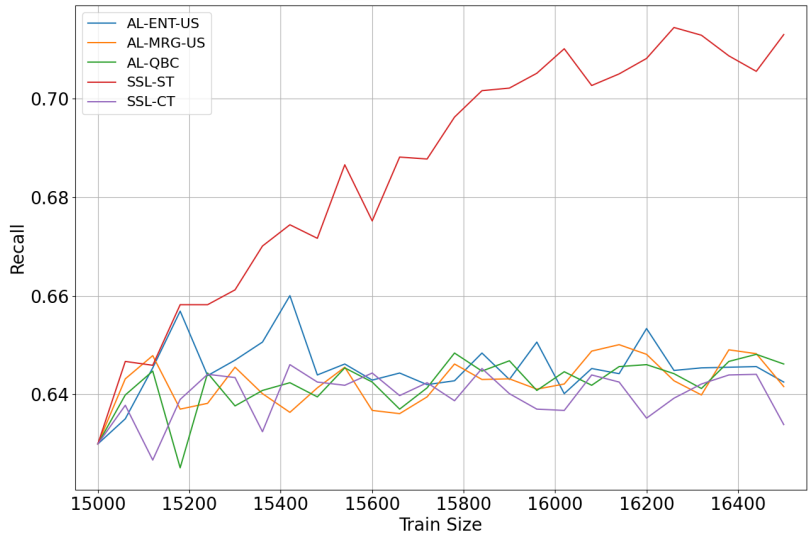


Figure 3.10: Recall results for RF (60 queries; 25 iterations).

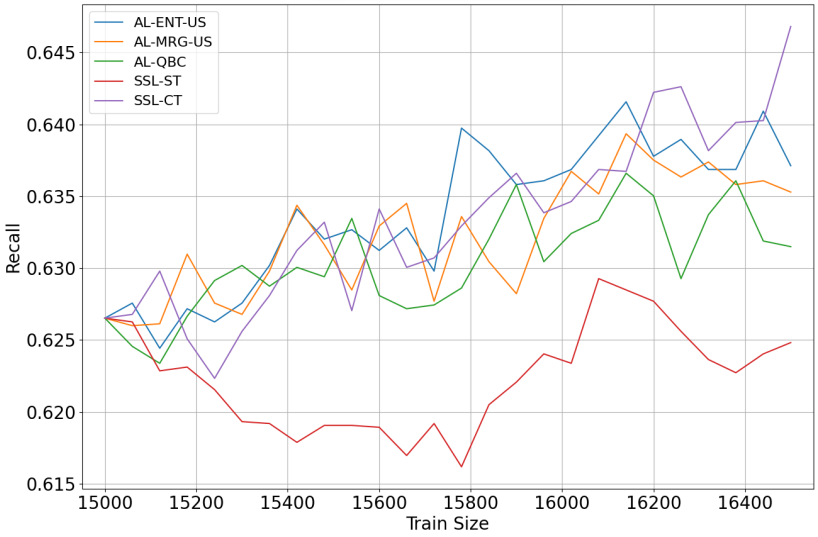


Figure 3.11: Recall results for XGBoost (60 queries; 25 iterations).

Chapter 4

X-ITERADE: EXplainable ITERative Anomaly Detection Ensemble

In the previous chapter, we conducted a comparative analysis of AL and SSL approaches in various domains. We explored how adjusting the imbalance ratio affected the effectiveness of these techniques and observed their sensitivity to different resampling methods. Our goal was to address the challenge of limited labeled data and demonstrate the potential of these frameworks in improving fraud detection. While both frameworks showed improvement in cases with moderate imbalance, we found that they were ineffective in handling extremely high imbalance scenarios, where no fraud instances were detected. This highlighted the limitations of AL and SSL in extreme imbalance situations, suggesting that unsupervised AD techniques that do not rely on labeled data would be a more effective approach.

In this chapter, we shift our focus toward addressing this gap by introducing X-ITERADE, a novel explainable iterative anomaly detection ensemble that integrates dynamic clustering to effectively isolate anomalous and normal cases while allowing the end-user to set a desired budget for the number of cases to be returned. In a completely

unsupervised manner, X-ITERADE provides a strategic starting point for labeling and inspection by domain experts. This budget-agnostic solution outperforms existing alternatives and offers flexibility in adjusting labeling budgets according to user preferences. X-ITERADE can select a subset of the dataset within a flexible budget, achieving a 3-15x higher percentage of positive cases, referred to as suspicious cases. This capability is especially beneficial for highly imbalanced datasets and smaller labeling budgets. Additionally, this chapter will explore the X-ITERADE explainability module, which generates intuitive, human-readable descriptions of group behaviors within suspicious cases. By leveraging SHAP, ARM, and LLM, the module provides in-depth cluster behavior analysis, helping users in comprehending the reasoning behind false-positive decisions. This feature is valuable in building profiles within the dataset and distinguishing between different behaviors in customer transactions.

This chapter's content has been published in the proceedings of the 27th International Conference on Discovery Science 2024.

Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu ITERADE - ITERative Anomaly Detection Ensemble for Credit Card Fraud Detection. In Dino Pedreschi, Anna Monreale, Riccardo Guidotti, Roberto Pellungrini, and Francesca Naretto, editors, Discovery Science, pages 115–133, Cham, 2025. Springer Nature Switzerland.https://doi.org/10.1007/978-3-031-78980-9_8 [4]

4.1 Introduction

AML efforts are crucial not only for financial stability but also for global security, as money laundering supports various criminal activities, including terrorism, human trafficking, and the drug trade. Fraud detection, a key component of AML compliance, helps companies

safeguard their customers' transactions and accounts from fraudulent activity. Fraud and money laundering intersect in two key areas: laundering the proceeds of fraud and using fraudulent accounts for money laundering. With the rise of e-commerce and online transactions, the use of credit and debit cards has surged, leading to an increase in fraudulent activities. Detecting and preventing credit card fraud is crucial not only to avoid financial losses and data breaches but also to maintain productivity and customer trust. Recent reports show that credit card fraud is not only persisting but escalating in both frequency and financial impact in the US and worldwide. With nearly 450,000 reports of credit card-related identity theft in 2024 and projected losses exceeding $12.5 billion in 2025, credit card fraud remains the most common form of identity theft in the United States [32, 33].

ML techniques have shown promise in detecting credit card fraud [21, 63, 99, 102]. However, several challenges persist. One of these challenges lies in labeling the data, an expensive and time-consuming task requiring substantial domain knowledge and expertise. Often, the data available for training ML models is insufficiently labeled, i.e., only a small number of cases are labeled, leading researchers to use alternative methods to supervised learning, such as SSL and AL strategies. Moreover, in domains like fraud detection, labeled data is not only scarce but also highly imbalanced, further complicating the process. Additionally, for any successful ML system, an explainability module is essential, as decisions made by AI can significantly impact both businesses and human lives. Therefore, the decisions generated by these systems must be understandable and transparent to the clients who rely on them and to those affected by their outcomes. Explainability and interpretability are crucial for ensuring transparency and traceability, enabling both clients and experts to trust and validate the system's decisions. A common approach used by labeling experts to identify a starting point for labeling is building profiles within the dataset to better detect suspicious groups. This is where explainability and interpretability become particularly

valuable, as they help uncover patterns and insights that guide the identification of these groups, making the labeling process more efficient and reliable. In this paper, we focus on the common setting where there are no labels available, which is common, especially for fin-techs that are just entering the market and require fraud detection from day 1. This occurs when companies wish to deploy ML models but have not yet labeled any cases. This is a typical setting in fraud detection scenarios because there is a vast amount of data but only a few fraud cases. This raises the question of which cases to label first, i.e., how to start the costly, labor-intensive, and time-consuming process of labeling. If the cases are not correctly selected, there is a high probability of labeling only non-fraudulent cases, as finding fraud cases among a vast majority of non-fraudulent instances is very hard. Thus, identifying the most suspicious transactions to label first is critical in these imbalanced datasets to maximize the detection of fraudulent activities.

In this chapter, we present X-ITERADE, an explainable iterative anomaly detection ensemble that incorporates a dynamic clustering approach. X-ITERADE is designed to separate anomalous cases from normal ones, effectively reducing the imbalance ratio in a fully unsupervised manner. It also features an explainability module that leverages ARM and SHAP for group behavior analysis of anomalous cases. This approach provides a valuable starting point for labeling by domain experts and offers a flexible, high-performing solution that surpasses existing baselines while accommodating different labeling budget constraints.

Contributions: Our main contributions are as follows:

- We propose X-ITERADE, an unsupervised solution with an adjustable budget that selects a subset of the dataset with a higher percentage of fraud cases compared to the initial dataset in credit card fraud detection, making it easier for labeling experts to start identifying fraudulent transactions.

- We conduct a thorough experimental comparison on multiple datasets with different imbalance ratios, showing the competitiveness of the proposed X-ITERADE method.
- We introduce an explainability algorithm that can be applied to the anomalous cases identified by X-ITERADE, generating human-level descriptions of the group behaviors observed within these suspicious cases.

Organization: This section is organized as follows: Section 4.2 presents our proposed algorithm, X-ITERADE. In Section 4.3, we detail our experimental configuration. The results from our experiments are presented and discussed in Section 4.4. Finally, Section 4.5 concludes the chapter and offers insights for future research avenues.

4.2 X-ITERADE: EXplainable ITERative Anomaly Detection Ensemble

4.2.1 X-ITERADE

Identifying where to initiate the labeling process poses a significant challenge for domain experts when faced with vast volumes of unlabeled data where the majority is expected to be non-fraudulent. This process causes substantial costs, particularly in imbalanced domains where experts prioritize labeling the fraudulent group of cases. However, this often requires investigating a large volume of transactions. To address this issue, we introduce X-ITERADE, an iterative anomaly detection ensemble.

The key goal of X-ITERADE is to rebalance the dataset by iteratively identifying and isolating anomalous fraud cases while identifying safe instances that do not need to be further inspected. The main idea of X-ITERADE is to iteratively apply an anomaly

detection ensemble to clusters within a dataset that is modified at each iteration. At each iteration, the optimal number of clusters to be formed is dynamically determined for the current dataset. After this step the clustering algorithm is applied and the anomaly detection ensemble is used inside each cluster obtained. The less suspicious cases are considered safe (non-fraud) and are removed from the dataset. The process then restarts with the calculation of the optimal number of clusters for the newly obtained dataset. Once the iterative process is complete and the most suspicious instances are identified, the explainability module is employed for group behavior analysis. This module, which plays a crucial role in understanding the detected anomalies, will be discussed in detail in subsection 4.2.2.

In more detail, initially, we apply an outlier preprocessing step with IForest to clean the data from any noise that might be present. This step is crucial, as eliminating noise enhances model accuracy, allowing the focus to be directed toward identifying fraud-related outliers. Then, in each iteration, the optimal number of clusters is determined using K-Means and the KneeLocator [82] library. K-Means is an iterative, centroid-based clustering algorithm that partitions a dataset into similar groups based on the distance between their centroids. The pseudo-code for this step is shown in the Algorithm 4.1. We compared K-Means with GMM to select the most suitable clustering algorithm. Although GMM are more resource-intensive and time-consuming, they did not yield better results than K-Means based on the silhouette score of the clusters. K-Means also offer easier optimization, which is crucial for our iterative approach. By dynamically adjusting the number of clusters in each iteration, we ensure model flexibility and prevent biases associated with a static cluster number. The KneeLocator library helps identify the optimal number of clusters using the elbow method. It is important to note that K-Means relies on certain assumptions: clusters are expected to be roughly spherical (convex), of similar size, and

separable based on Euclidean distance. These assumptions present limitations, as K-Means may perform poorly when clusters are concave, highly imbalanced in size, or have complex shapes. In addition, it is sensitive to initialization and can converge to local minima. While these constraints must be acknowledged, we chose to continue with K-Means because of its computational efficiency, scalability, and straightforward optimization, which are particularly advantageous in our iterative framework. Its interpretability and stable performance across our experiments made it the most suitable choice despite its known limitations.

Subsequently, K-Means clustering is applied using this determined number of clusters. We must highlight that this step is repeated in each iteration, allowing the fitting of different clusters as the algorithm moves forward in the iterations. This is important because each iteration modifies the dataset by removing cases considered safe in the previous iteration. For each cluster, an anomaly detection ensemble is employed to calculate scores for individual data points. Instances with scores above the upper threshold are flagged as anomalies and stored in a suspicious list, while those below the lower threshold are identified as safe cases and are removed from the dataset. Thus, the algorithm will be able to focus on the most problematic and potentially unsafe cases in the next iterations. Upon completing all iterations, the instances in the suspicious list that appear across all iterations are identified and labeled as anomalies. Once we calculate the common cases across all iterations, we have two options: we can either analyze all the anomalous common cases or concentrate on the cases that consistently appear in every iteration, which we refer to as Most Repeated (MR) cases. This strategy allows for greater flexibility in managing our solution's budget. These labeled instances can then be forwarded to domain experts for further investigation. The pseudocode for X-ITERADE is depicted in Algorithm 4.2.

We employed two distinct ensembles within this algorithm: Ensemble 1 consists of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [31], IForest [61],

Algorithm 4.1 Find Optimal K

```
procedure FindK(df)
    Initialize d as an empty dictionary
    for k in 5 to 20 do
        Fit K-Means with k clusters on df
        Store inertia in d[k]
    end for
    Return knee point of d values using KneeLocator
end procedure
```

and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [67], and Ensemble 2 uses LOF [12], IForest, and HDBSCAN. We use the first ensemble when applying a hard voting mechanism to predict whether each instance is anomalous. The average of these predictions is computed, and instances are categorized as either suspicious or safe based on predefined thresholds. The second ensemble employs soft voting as a probabilistic anomaly detection ensemble. It predicts the degree of anomaly for each instance using parameters such as negative_outlier_factor_ for LOF, average anomaly score (decision_function) for IForest, and probabilities_ for HDBSCAN. These values are then scaled to a range between 0 and 1, and their sum is utilized as the instance anomaly score. This score is compared against the specified up and down thresholds to isolate anomalous and safe instances accordingly.

4.2.2 Explainability Module

After the iterative detection of suspicious and safe cases using X-ITERADE, the next step is to provide an interpretable explanation of the detected anomalies. To this end, we introduce a novel explainability module designed to describe the group behavior of flagged anomalous cases. This module focuses on the subset of cases that have been consistently identified as fraudulent across multiple iterations, referred to as suspicious cases, to extract

Algorithm 4.2 X-ITERADE Algorithm

procedure X-ITERADE($X, y, up_threshold, down_threshold, voting, iterations$)
    Initialize variables
    $X, y \leftarrow$ CLEANDATA(X, y)
    for $i = 1$ **to** $iterations$ **do**
        $k \leftarrow$ FINDK(X)
        Fit K-Means with k clusters
        Predict clusters for X
        if $voting ==$ "$SoftVoting$" **then** ▷ Perform Soft Voting Option
            for each cluster **do**
                Fit IForest, LOF, and HDBSCAN on cluster data
                Store anomaly scores
            end for
            Scale anomaly scores using MinMaxScaler
        else ▷ Perform Hard Voting Option
            for each cluster **do**
                Fit IForest, DBSCAN, and HDBSCAN on cluster data
                Store anomaly predictions
            end for
        end if
        Calculate vote score by adding the anomaly scores obtained for each instance
        Remove safe cases with vote score $\leq down_threshold$ from X and y and store separately
        Store cases with vote score $> up_threshold$ in the suspicious list
        Update variables and performance metrics
    end for
    Calculate common instances in the suspicious list
    Save metrics, common instances, and safe cases
    Send the suspicious list to the **Explainability Module**
end procedure

insight and help decision-making from domain experts.

Instead of analyzing each case individually, our module aims to explain the collective behavior of groups of similar suspicious instances. To achieve this, we first preprocess the suspicious cases using additional feature engineering techniques. Then, we determine the optimal number of clusters within the suspicious cases, based on the elbow method. We then apply clustering and train a decision tree classifier to predict cluster assignments. This classifier enables a hierarchical segmentation of the dataset into tree nodes, each representing a group of instances sharing behavioral characteristics.

As we traverse the decision tree, we apply SHAP to determine the features that most significantly contribute to each node's prediction. We selected SHAP over alternative explainability methods such as Local Interpretable Model-Agnostic Explanations (LIME) [79] because SHAP provides consistent, theoretically grounded attributions based on Shapley values. Unlike LIME, which relies on local linear approximations that can vary with perturbation sampling, SHAP guarantees properties such as local accuracy and consistency, ensuring stable and comparable feature importance across nodes. These advantages make SHAP particularly well-suited for our tree-based framework. These top features are subsequently used as inputs to the Apriori algorithm [5], which mines high-confidence, high-support association rules that describe frequent patterns among the node's instances.

To generate human-readable descriptions for each group (i.e., tree leaf), we employ LLMs through an In-Context Learning (ICL) approach. In this setting, model prompts are constructed using the SHAP feature importances, association rules, the distribution of fraud and non-fraud cases, and the path of decision rules that led to the corresponding node. To support better prompt engineering, we also provide a brief explanation of the features and a clear task description. This rich combination of context enables the LLM to produce concise and meaningful summaries that reflect the underlying behavior of the

Algorithm 4.3 X-ITERADE Explainability Algorithm

procedure EXPLAINABILITY($X, y, \sigma, \mathcal{L}$)
  Initialize variables
  $k \leftarrow$ FINDK(X)
  Fit K-Means with k clusters on X
  Predict cluster labels C for cases in X
  $\mathcal{D} \leftarrow$ APPEND(X, C) ▷ Set cluster number as target variable C
  Fit decision tree $\mathcal{DT}$ on the dataset $\mathcal{D}$ with target C
  for each node i in $\mathcal{DT}$ **do**
    Extract subset of data $\mathcal{S}_i$ at node i
    Calculate SHAP values for $\mathcal{S}_i$
    Plot SHAP summary plot for node i
  end for
  for each leaf node **do**
    Extract the SHAP $top_features$ for the leaf node
    $Rules \leftarrow$ASSOCIATIONRULEMINER($X, y, \sigma, top_features$) ▷ Mine association rules with support $\geq \sigma$ using the SHAP $top_features$
    DESCRIBEGROUPBEHAVIOUR($X, y, \mathcal{L}, Rules, top_features, \mathcal{DT}$) ▷ Generate descriptive summary using LLM $\mathcal{L}$ for the cases in each leaf
  end for
  Save SHAP plots, mined rules, and summaries for each leaf node
end procedure

group in each node.

The full workflow of our explainability module is presented in Algorithm 4.3. This explainability module plays a central role in turning raw anomaly detections into interpretable group profiles, effectively bridging the gap between unsupervised learning outputs and actionable decision-making. It not only simplifies complex decisions for clients by offering clear, human-readable summaries but also serves as a powerful tool for researchers. Instead of relying only on quantitative metrics, scientists can explore and compare group behaviors through intuitive natural language descriptions—making behavioral patterns more accessible and easier to interpret.

4.3 Experimental Settings

4.3.1 Datasets

For our experiments, we used the CCT and BAF datasets, both of which were introduced in Section 3.2.1, and applied the same preprocessing strategies described there. To simulate real-world conditions, we obtained actual imbalance scenarios from our industry partner, H3M Analytics, which typically encounters the following Imbalance Ratio (IR) [1]: low imbalance (0.21), medium imbalance (0.077), and high imbalance (0.00077). To replicate these IRs, we randomly selected four subsets from each dataset and applied the same preprocessing steps. Additionally, we conducted experiments with the original imbalance present in the datasets to examine how different IRs impact our framework's performance. The characteristics of each dataset used in our experiments are summarized in Table 4.1.

Name	Size	# Features	# Categorical	IR	Fraud Cases	Non-Fraud Cases
CCT - Low	50571	34	14	0.21	10620	39951
CCT - Med				0.077	3894	46677
CCT - High				0.00077	39	50532
CCT - Original				0.0014	71	50500
BAF - Low	52000	30	5	0.21	10920	41080
BAF - Med				0.077	4004	47996
BAF - High				0.00077	40	51960
BAF - Original				0.01102	573	51427

Table 4.1: Main characteristics of the datasets used for X-ITERADE.

For our explainability module, we introduced additional engineered features to allow more specific and descriptive groupings. In particular, we added a new categorical feature called Merchant Category Groups (MCG) [2] that groups merchants into ten semantically

[1] We define IR as the ratio between the number of fraudulent cases (transactions) and the total number of cases (transactions) in the dataset.

[2] https://stripe.com/en-ca/guides/merchant-category-codes

meaningful categories. The details are presented in Table 4.2. These engineered features, along with others, are then one-hot encoded, as opposed to the previously used hash encoding, to enhance the interpretability and effectiveness of rule mining using the Apriori algorithm.

Range	Merchant Category Group
0001–1499	Agricultural services
1500–2999	Contracted services
4000–4799	Transportation services
4800–4999	Utility services
5000–5599	Retail outlet services
5600–5699	Clothing shops
5700–7299	Miscellaneous shops
7300–7999	Business services
8000–8999	Professional services and membership organisations
9000–9999	Government services

Table 4.2: Definitions of Merchant Category Groups Based on Code Ranges

To ensure a fair comparison across the different LLMs used in the explainability module of X-ITERADE, we employed a consistent set of test examples. For each dataset derived from the X-ITERADE variants, we generated 10 distinct test sets after applying the full explainability pipeline, including clustering, decision tree construction, SHAP value computation, and association rule mining. These test sets were created based on the information extracted from each decision tree leaf. Specifically, each leaf was summarized with its two most important SHAP features, relevant association rules, the distribution of fraud cases, and the sequence of decision rules leading to that node.

From each dataset, we selected 10 representative examples: 6 from leaves that contained fraud cases and 4 from leaves without any fraud (pure leaves). These 20 examples collectively formed our test set. The same prompt was used across all LLMs for each example to ensure consistency.

Each prompt followed an ICL structure: (1) It began with a clear task description, (2) followed by a natural language summary of the feature explanations, and (3) included three input–output examples. The prompt concluded by (4) asking the model to generate the output for a given test example. Examples of the prompts used for both datasets are shown in Tables B.1 and B.2 in the Appendix B.

4.3.2 Evaluation Methodology

Given our imbalanced datasets and unsupervised approaches, we employed a combination of performance metrics to evaluate both minority and majority groups. For both groups, we calculated the Precision and F1_Score to assess the accuracy of identifying safe cases and the detection of isolated anomalous cases. Additionally, for the minority group, which is our main focus, we measured the improvement in the IR after the selection of potentially anomalous cases. We calculated this improvement as the IR obtained in the sample of cases selected by an approach divided by the IR of the original data.

To establish a starting point for the experts to inspect and label cases, we recognize the significance of budgetary considerations. Hence, an important metric to consider, which we also report, is the total number of anomalies detected, as these will subsequently be forwarded to experts for labeling purposes.

We considered the following baseline algorithms: DBSCAN, HDBSCAN, IForest, and LOF which were then integrated into X-ITERADE alongside Empirical Cumulative Distribution Functions (ECOD) from the PYOD [103] package. We compared X-ITERADE against these baselines using two main variants: hard voting (HV) and soft voting (SV). Each algorithm is fitted in the entire dataset and the cases detected as anomalies are labeled as positive (fraud).

Model	Searched Parameters		Best Parameter Values			
	Parameter	Range	Low	Medium	Original	High
DBSCAN	eps	range(0.1, 1.5, 0.1)	0.1	0.1	0.1	0.1
	min_samples	range(2, 10, 1)	2	2	2	2
HDBSCAN	min_cluster_size	range(5, 50, 5)	45	35	45	45
	min_samples	range(2, 10, 1)	7	8	6	4
IForest	n_estimators	range(50, 200, 50)	50	150	50	150
	max_samples	linspace(0.1, 1.0, 10)	0.6	0.2	0.2	0.5
	contamination	linspace(0.01, 0.3, 10)	0.01	0.01	0.01	0.01
LOF	n_neighbors	range(5, 50, 5)	25	30	30	40
	leaf_size	range(10, 50, 10)	10	10	10	10
ECOD	contamination	linspace(0.01, 0.3, 10)	0.01	0.01	0.01	0.01

Table 4.3: Hyperparameters for Different Models and Configurations - CCT Dataset

Throughout our experiments, we worked with two sets of parameters: the internal hyperparameters of the baseline anomaly detection models and the external hyperparameters of the X-ITERADE framework. The internal hyperparameters of all baseline models—IForest, LOF, HDBSCAN, DBSCAN, and ECOD—were tuned in an unsupervised manner using grid search, with the silhouette score as the scoring function. After fine-tuning, these models were integrated into our hard-voting and soft-voting ensembles. This strategy allows us to explore the most suitable hyperparameters to use in the baselines and our proposed solution without any data leakage from the true class labels. The hyperparameters of the ensemble baselines explored during our tuning phase are detailed in Tables 4.3 and 4.4.

For our X-ITERADE framework, we introduced the up_threshold and down_threshold hyperparameters, which depend on the voting scheme (soft or hard) used in the corresponding ensemble. For the hard voting ensemble, we tested values of 3 and 2 for the up_threshold, aiming to detect instances predicted as anomalies by all 3 models and by more than 2 models, respectively. For the soft voting scheme of X-ITERADE, we investigated four combinations of thresholds for both up_threshold and down_threshold. Details

Model	Searched Parameters		Best Parameter Values			
	Parameter	Range	Low	Medium	Original	High
DBSCAN	eps	range(0.1, 1.5, 0.1)	1.1	0.7	0.5	1.3
	min_samples	range(2, 10, 1)	3	2	2	4
HDBSCAN	min_cluster_size	range(5, 50, 5)	10	5	30	30
	min_samples	range(2, 10, 1)	2	2	2	2
IForest	n_estimators	range(50, 200, 50)	150	100	100	150
	max_samples	linspace(0.1, 1.0, 10)	0.6	0.9	0.3	0.2
	contamination	linspace(0.01, 0.3, 10)	0.01	0.01	0.01	0.01
LOF	n_neighbors	range(5, 50, 5)	35	15	25	30
	leaf_size	range(10, 50, 10)	10	10	10	10
ECOD	contamination	linspace(0.01, 0.3, 10)	0.01	0.01	0.01	0.01

Table 4.4: Hyperparameters for Different Models and Configurations - BAF Dataset

Voting	Ensembles	Variant	up_threshold	down_threshold
Hard Voting (HV)	HDBSCAN; IForest; DBSCAN	HV 1.1	3	0
		HV 1.2	2	0
Soft Voting (SV)	HDBSCAN; IForest; LOF	SV 2.1	2.7	1.5
		SV 2.2	2.5	1.2
		SV 2.3	2.7	1.2
		SV 2.4	2.5	1.5

Table 4.5: Experimental configurations of X-ITERADE Algorithm. We appended "MR" (Most Repeated) to the X-ITERADE variants to indicate that we focused on the anomalous cases flagged by X-ITERADE that consistently appeared across all iterations.

of our experiments for X-ITERADE variants are presented in Table 4.5.

For the explainability module, we evaluated several open source LLMs, including LLaMA 3.1-8B [92], LLaMA 2-7B [93], Mistral-7B-v0.1 [48], and their instruction-tuned variants, Mistral-7B-Instruct-v0.3, and LLaMA-3.1-8B-Instruct, all run locally using Hugging Face [98] implementations. We also compared their output with GPT-4o [72], accessed via API, to determine the quality and fluency of the benchmark. To streamline the prompts and generate concise summaries, we used decision trees with a maximum depth of 3 and extracted the decision path leading to each leaf node as contextual input.

To evaluate the performance of our explainability module using LLMs, we applied six different metrics to the responses generated for each test example across all models. These metrics assess both the content quality and linguistic characteristics of the outputs:

- **Correctness / Factual Accuracy**: Measures whether the model accurately summarizes the group information. Responses were evaluated manually on a scale from 1 (inaccurate) to 5 (fully accurate).
- **Coverage / Completeness**: Assesses whether the summary includes all key points about the group. A score of 1 indicates missing essential details, while 5 indicates complete coverage.
- **Novelty / Redundancy Across Examples**: Evaluates how well the model avoids repeating earlier examples seen in the in-context prompt. Higher scores indicate more original, non-redundant content.
- **Lexical Diversity**: Reflects the richness and variety of vocabulary used in the response. We computed this using the Measure of Textual Lexical Diversity (MTLD) [66] metric, which is robust to text length. MTLD is a measure of lexical diversity that is stable across text lengths and more robust than traditional type-token ratio metrics, and is calculated as:

$$\text{MTLD} = \frac{\text{Total Tokens}}{\text{Number of Factors at TTR Threshold (e.g., 0.72)}} \tag{4.1}$$

- **Length / Token Count**: Captures the verbosity of the response by counting the total number of tokens (words). This helps assess whether a model is being overly brief or unnecessarily verbose.

- **Readability Score**: Estimates how easy the response is to read and understand. We used the Flesch Reading Ease score [35], where higher values indicate simpler, more readable text. The Flesch Reading Ease score is a widely used readability metric based on average sentence length and syllables per word, which is calculated as:

$$\text{Flesch Reading Ease} = 206.835 - 1.015 \times \left(\frac{\text{Total Words}}{\text{Total Sentences}}\right) - 84.6 \times \left(\frac{\text{Total Syllables}}{\text{Total Words}}\right) \tag{4.2}$$

The first three metrics were manually rated by 3 human annotator on a scale from 1 to 5, and the average score was used for the analysis of the results. The remaining three were automatically computed using standard text analysis tools. To measure how consistently human annotators evaluated the model responses, we used two commonly accepted agreement metrics: Krippendorff's alpha and the ICC. Krippendorff's alpha [52] is designed to measure how much the raters agree with each other, taking into account the possibility of agreement happening by chance. It works with different types of data (such as ordered scales like 1–5) and can handle missing ratings. Values of alpha above 0.80 are usually considered strong agreement, values between 0.67 and 0.80 indicate moderate agreement, and anything below 0.67 suggests weak or inconsistent agreement.

We also used the ICC, specifically the two-way random-effects model with average ratings (ICC(2,k)), following the guidelines by Shrout and Fleiss [86]. ICC is useful when raters give scores on a numerical scale, and it measures how similar the ratings are across items and raters. According to Koo and Li [51], ICC values above 0.90 indicate excellent agreement, 0.75–0.90 show good agreement, 0.50–0.75 mean moderate agreement, and below 0.50 reflect poor agreement. Using both alpha and ICC together helps us get a clearer picture of rating consistency. While alpha focuses on how often raters agree beyond chance, ICC looks at how closely their scores match overall.

4.4 Results and Discussion

4.4.1 Experimental Results of X-ITERADE

To ensure a fair comparison between X-ITERADE and the baselines, we defined budget buckets based on the ranges obtained from our baseline models. Our main experimental results on the eight datasets, evaluated across multiple performance metrics, are detailed in Tables 4.6 and 4.7 and Figure 4.1. We present two types of results: in the tables, we provide the average results of our proposed solution across all iterations in each budget bucket, while in the figure, we show the results of the iteration where the proposed solution achieved the best result. By showing the average across all iterations in each budget bucket for X-ITERADE our goal was to provide a less optimistic view of our solution. By adding this handicap into our results, we aim to show that our algorithm works well not only at a given particular iteration (best result) but also has an overall good performance. In all our experiments, after calculating the common cases from all iterations, we can either examine all the anomalous common cases retrieved or focus on the cases that have repeated in every iteration. We appended MR (Most Repeated) to the X-ITERADE variants that use this last option. This approach allows us to adjust our budget effectively. For instance, in some variants like HV 1.2, the total common cases are usually above 5,000, while the most repeated cases can be as few as 50 to 500, typically resulting in a better ratio of frauds to total cases retrieved and a more realistic scenario for labeling experts.

From Tables 4.6 and 4.7, we observe an overwhelming advantage of using X-ITERADE across all budgets. Not only does our algorithm consistently detect a smaller number of anomalies, but a higher proportion of these are actual fraud cases (true positives), as shown in the last two columns of the tables. This results in a final set of suspicious cases with an improved imbalance ratio, fewer overall anomalies, but a higher concentration of

frauds. This is observed for both datasets on all imbalance ratios and budgets considered, except for the BAF-High dataset at a 600 budget, where our algorithm's performance was comparably close to the baseline ECOD. Looking in more detail at the results of X-ITERADE, we see that different X-ITERADE variants, based on the thresholds used in the experiments, are suited for different types of budgets. Overall, we observe that variants with lower thresholds for the anomalous cases (up_threshold) signal a higher number of anomalies, thus being more suitable for cases with higher budgets. The variants with higher anomalous case thresholds will be more adequate for lower budgets, outputting a smaller set of anomalous cases. For example, SV 2.2, SV 2.4, and HV 1.2 generally detect many cases as anomalies and are more appropriate for high budgets. However, they can also be used with low budgets by focusing on their most repeated cases. On the other hand, HV 1.1, SV 2.1, and SV 2.3 are more suitable for lower budgets. For illustration, the number of anomalies detected in each iteration, including the most repeated anomalies, and the number of frauds with the most repeated frauds in each iteration for the CCT original dataset, are plotted in Figure 4.2. These plots include horizontal lines indicating the budget buckets, providing a visual representation of the possible solutions for each budget bucket. Furthermore, the best values achieved within each budget category for each algorithm are listed in Figures 4.1, which may present an overly optimistic estimation.

In each budget bucket, X-ITERADE outperforms the baselines in terms of the number of anomalous cases detected and the precision of fraud cases within the anomalous cases. The improvement of each solution ("Improv.") is calculated by dividing the final imbalance ratio (precision of the minority group) by the initial imbalance ratio. The improvement results are displayed in Tables 4.6 and 4.7 in column "Improv.". While the baselines show lower improvements or degradation, the X-ITERADE algorithm shows consistently higher improvements, even when considering the average across all iterations of the algorithm for

each budget range.

Another notable observation is that as datasets become more imbalanced and the budget decreases, the improvements achieved by X-ITERADE become more significant. We achieved an average of 15 times improvement in the fraud cases signaled for the "CCT-Original" dataset with a budget of less than 600, using X-ITERADE - SV 2.4 - MR. For budgets between 600 and 4000, our X-ITERADE - SV 2.1 - MR outperformed other algorithms, providing an average 6.94 times improvement in the fraud cases present in the anomalies detected. As the imbalance decreases to low and medium levels, improvements become more modest, averaging 1.4 to 2.5 times improvement. However, we must highlight that, in these cases, the baselines are showing a much worse performance.

Furthermore, we found that the performance of anomaly detection baselines impacts the performance of the ensemble. For example, in the BAF dataset with high imbalance, all four baselines used in the ensemble failed to detect any anomalies, and DBSCAN labeled every instance as an anomaly. This affected the performance of the ensembles, resulting in less significant improvements. Despite these challenges, our algorithm still demonstrated superiority, achieving an average of 1.2 to 1.8 times improvement in the fraud cases detected out of the anomalies signaled.

Looking at the results of Figure 4.1, we observe that in the vast majority of cases, the baselines signal a larger set of anomalous cases (clear blue bars represent the number of anomalies detected by the baselines, and the dashed blue bars correspond to the X-ITERADE). Simultaneously, we observe that the number of frauds in the signaled cases is many times higher than the baselines (red lines in the figure). This shows that X-ITERADE provides a lower number of cases to inspect as potential anomalies. Moreover, in those cases signaled, more correspond to actual fraud cases.

All extra plots for all imbalance ratios and extra tables containing additional perform-

ance metrics for both the majority and minority groups are available in Appendix C.

Our study demonstrates the effectiveness of X-ITERADE in addressing the labeling problem in credit card fraud detection. By leveraging the benefits of various anomaly detection models and an iterative approach with two voting schemes and using dynamic clustering, we provide an algorithm with a flexible budget that delivers significant improvements, especially as the datasets become more imbalanced.

4.4.2 Explainability Module Results

To compare the performance of responses generated by six LLMs across different evaluation metrics on 20 test examples and rated by 3 human annotators, we normalized all metric scores and computed the average values for each model, and across different annotations. We also evaluated inter-rater agreement across three key dimensions, Correctness, Coverage, and Novelty, using two reliability metrics: Krippendorff's alpha and the ICC. The main experimental results of the X-ITERADE explainability module are illustrated in the Radar Charts shown in Figure 4.3, corresponding to the BAF and CCT datasets.

The results were largely consistent across both datasets; however, the BAF test examples exhibit greater feature diversity, which is reflected in their association rules and SHAP values. This helps explain the differences in the correctness and completeness scores observed between the two datasets. For example, in the ICL setup, we provide only three illustrative examples with their corresponding outputs. Due to the broader variety of features in the BAF dataset, many important features are not directly present in these examples; they are only briefly described at the beginning of the prompt. In contrast, the CCT dataset typically contains shorter and more consistent association rules, with a recurring set of features such as `Has_Chip` (indicating whether the card is chipped), `Use_Chip`

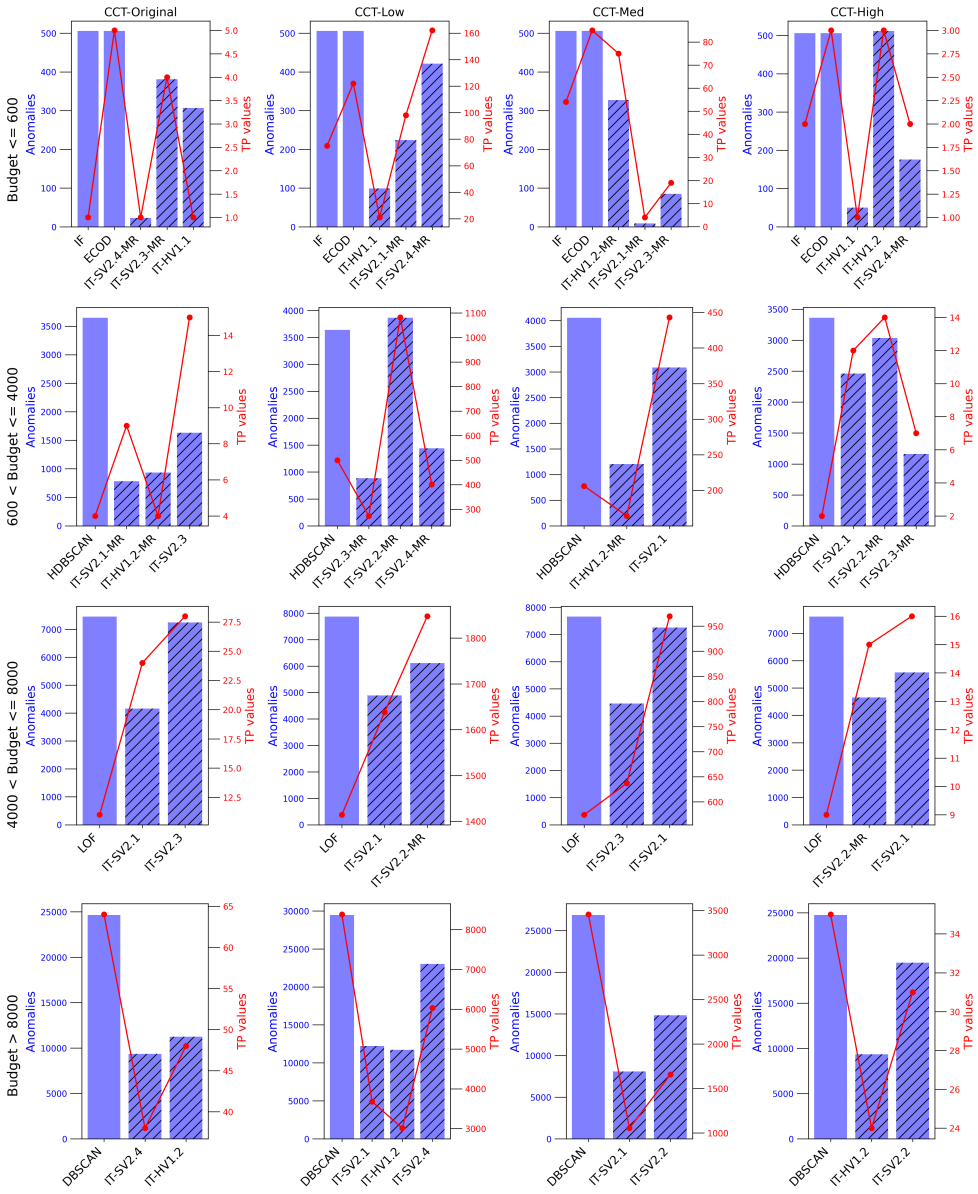


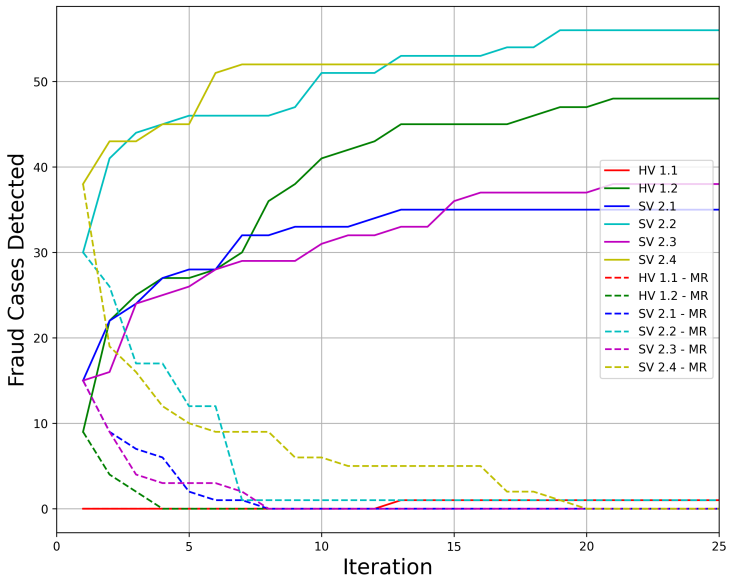
Figure 4.1: Comparison baselines with best X-ITERADE results on CCT dataset for the 4 budgets considered. Each row shows a different budget in decreasing order from top to bottom. Each column shows one of the variants of the CCT dataset - from left to right: Original, Low, Med, and High. (Blue bars: anomalies detected in baseline; dashed blue bars: anomalies detected in X-ITERADE; the red line shows, on the right y-axis, the number of positive (fraud cases) contained in the signaled cases.)

Dataset	Budget	Algorithm	Performance		Improv.	Average Anomalies	
			Precision	F1_Score		TP	Anom. Det.
CCT-Original	≤ 600	IForest	0.00198	0.00395	1.414	1	506
		ECOD	0.00988	0.01957	7.057	5	506
		X-ITERADE - SV 2.4 - MR	0.02104	0.04121	**15.029**	2.5	118.8
		X-ITERADE - SV 2.3 - MR	0.00567	0.01128	4.05	0.53	93.46
		X-ITERADE - HV 1.1	0.00184	0.00367	1.314	0.6	325.26
	> 600 & ≤ 4000	HDBSCAN	0.0011	0.0022	0.786	4	3647
		X-ITERADE - SV 2.1 - MR	0.00972	0.01925	**6.943**	12	1234
		X-ITERADE - HV 1.2 -MR	0.00328	0.00654	2.343	6.5	1983
		X-ITERADE - SV 2.3	0.00622	0.01236	4.443	18.33	2947.66
	> 4000 & ≤ 8000	LOF	0.00148	0.00296	1.057	11	7457
		X-ITERADE - SV 2.1	0.00416	0.00829	2.971	26.75	6426.75
		X-ITERADE - SV 2.3	0.00424	0.00844	**3.029**	27	6369.75
	> 8000 & ≤ 20000	DBSCAN	0.0026	0.00519	1.857	64	24652
		X-ITERADE - SV 2.4	0.00294	0.00586	2.1	46.77	15893.22
		X-ITERADE - HV 1.2	0.00419	0.00835	**2.993**	44.12	10535.52
CCT-Low	≤ 600	IForest	0.14822	0.25817	0.706	75	506
		ECOD	0.24111	0.38854	1.148	122	506
		X-ITERADE - HV 1.1	0.18583	0.31342	0.885	55.1	296.5
		X-ITERADE - SV 2.1 - MR	0.39202	0.56324	**1.867**	46	117.34
		X-ITERADE - SV 2.4 - MR	0.36214	0.53172	1.724	180.2	497.6
	> 600 & ≤ 4000	HDBSCAN	0.13736	0.24154	0.654	500	3640
		X-ITERADE - SV 2.3 - MR	0.29443	0.45492	**1.402**	495.75	1683.75
		X-ITERADE - SV 2.2 - MR	0.20561	0.34109	0.979	363.88	1769.74
		X-ITERADE - SV 2.4 - MR	0.26303	0.41651	1.253	420.3	1597.91
	> 4000 & ≤ 8000	LOF	0.17967	0.30461	0.856	1414	7870
		X-ITERADE - SV 2.1	0.30022	0.4618	**1.43**	1856.2	6182.8
		X-ITERADE - SV 2.2 - MR	0.29194	0.45194	1.39	1516.5	5194.5
	> 8000 & ≤ 20000	DBSCAN	0.28431	0.44274	1.354	8378	29468
		X-ITERADE - SV 2.1	0.29944	0.46088	**1.426**	3565	11905.62
		X-ITERADE - HV 1.2	0.25215	0.40275	1.201	2592.95	10283.43
		X-ITERADE - SV 2.4	0.25321	0.4041	1.206	4244.5	16762.62
CCT-Med	≤ 600	IForest	0.10672	0.19286	1.386	54	506
		ECOD	0.16798	0.28764	2.182	85	506
		X-ITERADE - HV 1.2 - MR	0.17935	0.30415	2.329	43.29	241.37
		X-ITERADE - SV 2.1 - MR	0.18113	0.30671	2.352	6.74	37.21
		X-ITERADE - SV 2.3 - MR	0.19489	0.32621	**2.531**	18.38	94.31
	> 600 & ≤ 4000	HDBSCAN	0.0508	0.09669	0.66	206	4055
		X-ITERADE - HV 1.2 - MR	0.12787	0.22675	1.661	238	1861.33
		X-ITERADE - SV 2.1	0.13519	0.23818	**1.756**	425.25	3145.5
	> 4000 & ≤ 8000	LOF	0.07498	0.1395	0.974	574	7655
		X-ITERADE - SV 2.3	0.12184	0.21721	1.582	818	6713.9
		X-ITERADE - SV 2.1	0.13192	0.23309	**1.713**	914.42	6931.5
	> 8000 & ≤ 20000	DBSCAN	0.12853	0.22778	**1.669**	3455	26880
		X-ITERADE - SV 2.1	0.12789	0.22678	1.661	1079.41	8440.25
		X-ITERADE - SV 2.2	0.10738	0.19394	1.395	1790.8	16676.86
CCT-High	≤ 600	IForest	0.00395	0.00787	5.13	2	506
		ECOD	0.00593	0.01179	7.701	3	506
		X-ITERADE - HV 1.1	0.00592	0.01177	7.688	1.5	253.5
		X-ITERADE - HV 1.2 - MR	0.0005	0.001	0.649	0.1	199
		X-ITERADE - SV 2.4 - MR	0.00625	0.01242	**8.117**	1.58	253
	> 600 & ≤ 4000	HDBSCAN	0.00059	0.00118	0.766	2	3362
		X-ITERADE - SV 2.1	0.00389	0.00775	5.052	12.5	3215
		X-ITERADE - SV 2.2 - MR	0.00197	0.00393	2.558	3.46	1755.5
		X-ITERADE - SV 2.3 - MR	0.00493	0.00981	**6.403**	6.33	1283.83
	> 4000 & ≤ 8000	LOF	0.00118	0.00236	1.532	9	7610
		X-ITERADE - SV 2.2 - MR	0.00323	0.00644	**4.195**	15	4651
		X-ITERADE - SV 2.1	0.00235	0.00469	3.052	16.7	7115
	> 8000 & ≤ 20000	DBSCAN	0.00141	0.00282	1.831	35	24756
		X-ITERADE - HV 1.2	0.00239	0.00477	**3.104**	24.5	10231.43
		X-ITERADE - SV 2.2	0.00172	0.00343	2.234	25.9	15082.1

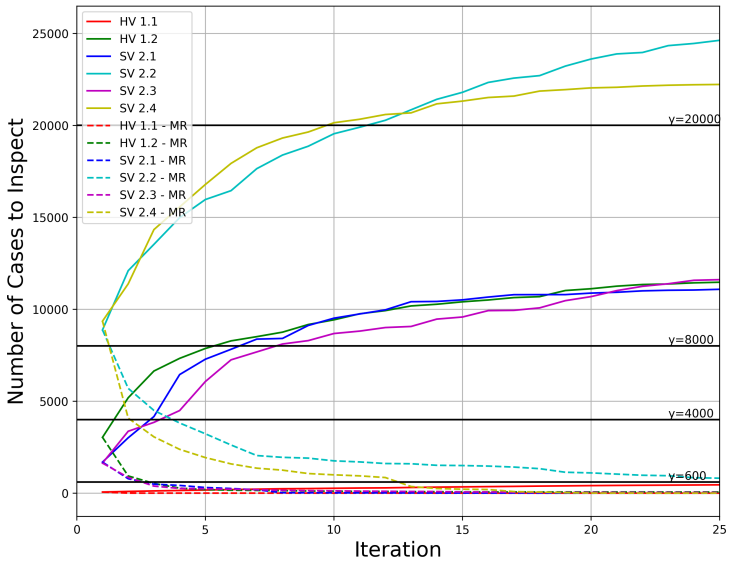
Table 4.6: Experimental results for CCT datasets across different imbalance ratios, with average performance reported for each budget bucket.

Dataset	Budget	Algorithm	Performance		Improv.	Average Anomalies	
			Precision	F1_Score		TP	Anom. Det.
BAF-Original	≤ 600	LOF	0	0	0	0	25
		HDBSCAN	0.01266	0.025	1.149	2	158
		IForest	0.01538	0.03029	1.396	8	520
		ECOD	0.02885	0.05608	2.618	15	520
		X-ITERADE - HV 1.1	0.01538	0.03029	1.396	10.33	671.5
		X-ITERADE - HV 1.2 - MR	0.04348	0.08334	**3.946**	4.8	110.4
		X-ITERADE - SV 2.4 - MR	0.02373	0.04636	2.153	14	590
	> 600 & ≤ 4000	X-ITERADE - HV 1.1	0.01581	0.03113	1.435	12.81	810.22
		X-ITERADE - SV 2.4	0.02441	0.04766	**2.215**	33	1352
		X-ITERADE - HV 1.2 - MR	0.01352	0.02668	1.227	47.5	3513
	> 4000 & ≤ 20000	DBSCAN	0.01102	0.0218	1	573	51998
		X-ITERADE - HV 1.2	0.01316	0.02598	**1.194**	285	21657
		X-ITERADE - SV 2.4	0.01167	0.02307	1.059	149.33	12798.33
BAF-Low	≤ 600	LOF	0.22222	0.36363	1.058	6	27
		HDBSCAN	0.31797	0.48251	1.514	69	217
		IForest	0.26923	0.42424	1.282	140	520
		ECOD	0.32885	0.49494	1.566	171	520
		X-ITERADE - HV 1.1 - MR	0.32171	0.48681	1.532	1.66	5.16
		X-ITERADE - HV 1.2 - MR	0.31582	0.48004	1.504	30.42	96.32
		X-ITERADE - SV 2.3 - MR	0.35681	0.52595	**1.699**	42	117.71
	> 600 & ≤ 4000	X-ITERADE - HV 1.2 - MR	0.26714	0.42164	1.272	978	3661
		X-ITERADE - SV 2.2	0.2866	0.44552	**1.365**	597.66	2085.33
	> 4000 & ≤ 20000	DBSCAN	0.21001	0.34712	1	10920	51997
		X-ITERADE - SV 2.2	0.23119	0.37556	**1.101**	1423	6155
		X-ITERADE - HV 1.2	0.21838	0.35848	1.04	1534.48	7026.51
BAF-Med	≤ 2000	HDBSCAN	0	0	0	0	0
		LOF	0.16667	0.28572	2.165	6	36
		IForest	0.13654	0.24027	1.773	71	520
		ECOD	0.18462	0.31169	2.398	96	520
		X-ITERADE - HV 1.1	0.10548	0.19083	1.37	15.4	146
		X-ITERADE - SV 2.3	0.05204	0.09893	0.676	17.16	329.76
		X-ITERADE - SV 2.4	0.19074	0.32037	**2.477**	84.5	443
	> 600 & ≤ 4000	X-ITERADE - HV 1.2	0.11457	0.20559	**1.488**	323.91	2827.16
		X-ITERADE - SV 2.2	0.09632	0.17572	1.251	155.8	1617.47
	> 4000 & ≤ 20000	LOF	0.07498	0.1395	0.974	574	7655
		X-ITERADE - SV 2.3	0.12184	0.21721	1.582	818	6713.9
		X-ITERADE - SV 2.1	0.13192	0.23309	**1.713**	914.42	6931.5
BAF-High	≤ 600	HDBSCAN	0	0	0	0	150
		LOF	0	0	0	0	34
		IForest	0	0	0	0	520
		ECOD	0.00192	0.00383	**2.494**	1	520
		X-ITERADE - SV 2.1 - MR	0.00141	0.00282	1.831	2	1414
	> 2000 & ≤ 20000	DBSCAN	0.00077	0.00154	1	40	51992
		X-ITERADE - SV 2.1	0.00098	0.00196	**1.273**	9.38	9600.3
		X-ITERADE - SV 2.4	0.00043	0.00086	0.558	4	9332
		X-ITERADE - SV 2.2	0.00064	0.00128	0.831	12	18732

Table 4.7: Experimental results for BAF datasets across different imbalance ratios, with average performance reported for each budget bucket.



(a) Number of fraud cases and most repeated fraud cases detected per iteration for the CCT original imbalance dataset.



(b) Number of anomalies and most repeated anomalies detected per iteration for the CCT original imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range.

Figure 4.2: X-ITERADE results across all iterations for CCT original imbalance dataset.

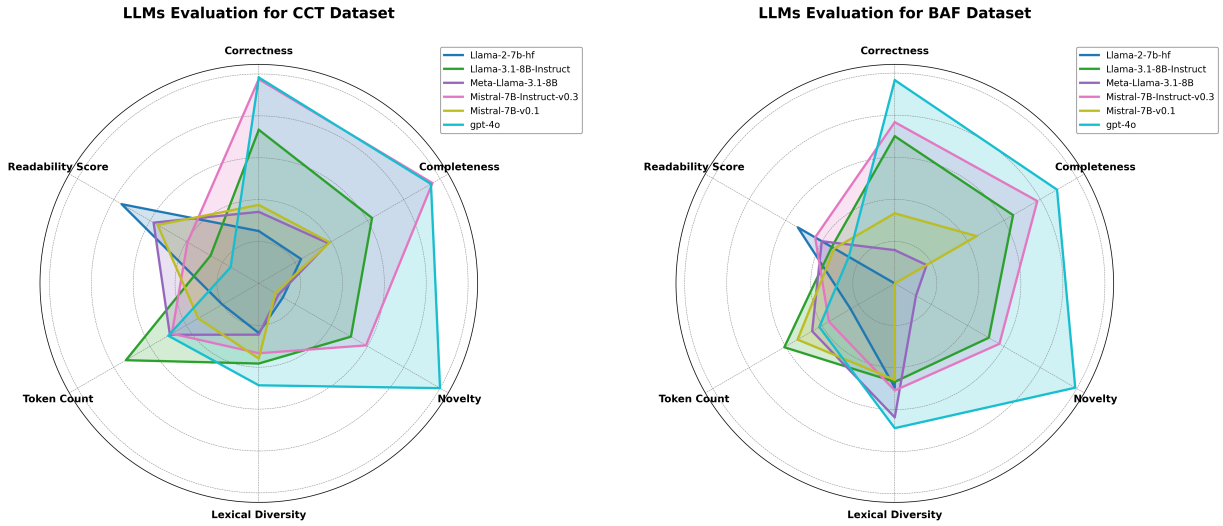


Figure 4.3: Comparison of LLM average performance across 20 test examples and 3 human annotators from the CCT and BAF datasets using six qualitative and linguistic evaluation metrics.

(indicating the type of transaction), and `Zip Code`. This higher consistency across test cases in the CCT dataset enables the models to generalize more effectively and produce more accurate responses.

Among all evaluated models, **GPT-4o** demonstrated the strongest overall performance, producing highly correct, complete, and informative responses in a concise format. However, this model requires access via API, raising concerns regarding data privacy and confidentiality. Additionally, with an approximate cost of $10 per million tokens, GPT-4o is less practical for deployment in production environments where cost-efficiency and data control are critical.

The most competitive open-source alternative was **Mistral 7B Instruct v0.3**. Despite being smaller and lighter than the LLaMA variants, it provided responses that were nearly as correct, complete, and brief as those generated by GPT-4o. While no model fully matched GPT-4o's level of novelty, Mistral Instruct v0.3 came closest. A notable limitation

of GPT-4o is its relatively low readability score, which can make it harder for non-technical users, such as clients, to understand. In contrast, Mistral Instruct v0.3 achieves a moderate readability level, neither too simple nor too complex, making it more accessible to a broader professional and non-professional audience.

The next-best model, **LLaMA 3.1 8B Instruct**, generated responses with moderate correctness and completeness. However, it tended to produce lengthy outputs that, despite their verbosity, lacked true completeness and often repeated phrases or patterns seen in the example prompts, resulting in reduced novelty.

The least effective models included **LLaMA 2-7B hf**, **LLaMA 3.1 8B**, and **Mistral 7B v0.1**. These models frequently repeated prompt examples when generating outputs for new test cases, which negatively affected both their correctness and novelty. While they achieved higher readability scores, they also exhibited limited lexical diversity and creativity, indicating a tendency to echo patterns seen during few-shot prompting rather than generalizing beyond them.

Another notable observation is that most models perform similarly in terms of Token Count, Readability Score, and Lexical Diversity, with only minor differences between them. In contrast, metrics such as correctness, completeness, and novelty show a more significant variation between the best and worst models. Novelty appears to be the most challenging aspect, as the top-performing model, GPT-4o, demonstrates a substantial lead over the others. On the other hand, lexical diversity is an area where all models score quite closely and achieve moderate performance.

As shown in Table 4.8, annotators had consistently strong agreement on Correctness, with alpha values above 0.87 and ICC scores above 0.95 for all models except GPT-4o. For GPT-4o, we observed a moderate level of agreement ($\alpha = 0.64$, ICC $= 0.84$), suggesting that factual accuracy in its responses may have been more subjective or harder to evaluate.

Table 4.8: Krippendorff's Alpha and ICC for each model and metric

Model	Correctness		Coverage		Novelty	
	Alpha	ICC	Alpha	ICC	Alpha	ICC
Llama-3.1-8B-Instruct	0.955	0.987	0.691	0.892	0.101	0.262
Meta-Llama-3.1-8B	0.901	0.979	0.963	0.989	0.891	0.980
Llama-2-7b-hf	0.876	0.958	0.927	0.958	0.905	0.931
Mistral-7B-v0.1	0.892	0.962	0.872	0.966	0.891	0.962
Mistral-7B-Instruct	0.991	0.980	0.686	0.859	0.430	0.692
GPT-4o	0.636	0.845	0.913	0.936	0.000	≈ 0.000

Coverage scores were also generally strong, though slightly more variable. Meta-Llama-3.1-8B and Llama-2-7b-hf showed excellent agreement ($\alpha > 0.92$), while Llama-3.1-8B-Instruct and Mistral-7B-Instruct had noticeably lower alpha scores (around 0.69), pointing to more disagreement between raters on how complete or thorough those responses were.

The Novelty dimension had the most variation in agreement. Models like Meta-Llama-3.1-8B, Llama-2-7b-hf, and Mistral-7B-v0.1 all had strong reliability ($\alpha > 0.89$, ICC > 0.93), but Llama-3.1-8B-Instruct performed poorly ($\alpha = 0.10$, ICC $= 0.26$), showing that raters disagreed a lot about the originality or redundancy of its answers. For GPT-4o, even though nearly all novelty scores were rated 5, this lack of variance caused both alpha and ICC to fall to zero ($\alpha = 0$, ICC $\simeq 0$). This highlights a known limitation of these metrics: when ratings have almost no variation, they can report zero agreement even when annotators are actually aligned. In such cases, it helps to look at raw score distributions or percent agreement as complementary indicators. The Krippendorff's alpha scores across different models and metrics are also visualized as grouped bar plots in Figure 4.4

In conclusion, instruction-tuned models consistently outperformed their base counterparts. Notably, **Mistral 7B Instruct v0.3** delivered performance comparable to GPT-4o, surpassing even the larger LLaMA 3.1 Instruct model. In addition to being free and loc-

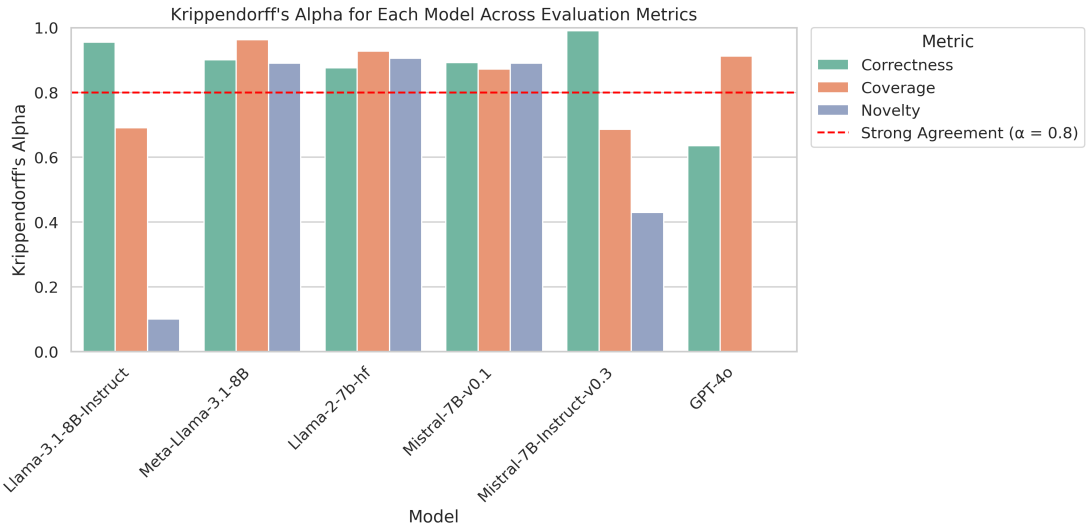


Figure 4.4: Krippendorff's alpha for each model across evaluation metrics.

ally deployable, Mistral and LLaMA offer stronger data security and greater independence from external infrastructure. While GPT-4o excels in correctness, completeness, and novelty, the preference for open-source, privacy-conscious solutions by our industry partner made models like Mistral 7B Instruct more favorable for real-world deployment. We also observed that novelty remains a challenging aspect for all models except GPT-4o, whereas lexical diversity scores are moderately strong and quite similar across models. A brief comparison of LLMs performance with practical insights is presented in Table 4.9.

4.4.3 Limitations

X-ITERADE and this study still present some challenges that need to be further explored. In this study, we observed that the performance of the ensemble heavily depends on the effectiveness of the baseline algorithms it employs. The primary challenge affecting the performance of these baselines is the unsupervised hyperparameter tuning. Without labels or a validation set, achieving optimal hyperparameter tuning is extremely difficult.

Table 4.9: Brief Comparison of LLMs Performance and Practical Considerations

Model	**Performance Summary**	**Practical Considerations**
GPT-4o	Highest correctness, completeness, and novelty with concise outputs; low readability score.	Requires API access; costly ($10/million tokens); privacy concerns.
Mistral 7B Instruct v0.3	Comparable correctness and completeness to GPT-4o; nearly as novel; medium readability score.	More lightweight, free, and locally deployable; better for a wider audience and privacy-sensitive environments.
LLaMA 3.1 8B Instruct	Medium correctness and completeness; tends to produce longer, repetitive responses.	Locally deployable; may be less suitable for concise, novel outputs.
LLaMA 2-7B HF, Mistral 7B v0.1, LLaMA 3.1 8B	Lower correctness and novelty; often repeats prompt example outputs.	Higher readability but low lexical diversity; less creative.

Moreover, we did not use any information regarding the imbalance in the dataset to not bias the parameter selection. Experiments can be improved by expanding the search space and using larger steps in the grid search for hyperparameter tuning. In addition, other unsupervised solutions for hyperparameter tuning should be investigated. Additionally, the flexible budget is influenced by the up and down thresholds used in the voting scheme. To provide more budget buckets with diverse solutions, experiments can be conducted using various combinations of thresholds for both voting schemes.

The Explainability Module, while promising, also has areas that can be improved. First, our human evaluation was limited to only three annotators and 20 test examples sampled from pure and suspicious leaves. Expanding the number of examples and involving a larger, more diverse pool of annotators would enhance the reliability and generalizability of the

findings. We also relied only on ICL and prompt-engineering with LLMs, without performing hyperparameter tuning or exploring alternative prompting strategies. Incorporating fine-tuning, or retrieval-augmented generation could improve the quality and consistency of the generated explanations. Additionally, we have not yet evaluated the Explainability Module in zero-shot or few-shot settings. Analyzing performance in these scenarios could provide further insights into its robustness across different resource constraints. Lastly, our current analysis focused solely on textual explanations. Extending the module to include multimodal outputs, such as visual rule summaries, SHAP plots, or cluster t-distributed stochastic neighbor embedding (t-SNE) diagrams, could make the explanations more intuitive and actionable, particularly for non-technical users.

4.5 Conclusion

In this chapter, we proposed X-ITERADE as a solution for unsupervised anomaly detection in fraud scenarios, demonstrating its effectiveness across various imbalanced domains. Our framework offers an adjustable budget that can be tailored based on the preferences of domain experts. Our experiments demonstrate that X-ITERADE outperforms the baseline methods across all datasets and imbalance domains, effectively increasing the fraud to anomalies detected ratio and resulting in a less imbalanced dataset. We observed that X-ITERADE can select a subset of the entire dataset within a flexible budget, achieving a higher percentage of positive cases by a factor of 3 to 15. This capability is particularly advantageous for highly imbalanced datasets, especially when working with a limited labeling budget. Moreover, our solution provides as a side product a set of safe instances for which no further inspection is needed. Furthermore, X-ITERADE includes an explainability module designed for cluster behavior analysis of the most suspicious cases. This feature

is valuable for building detailed profiles within the dataset, providing deeper insights into customer behaviors. The module utilizes local LLMs instead of relying on external API, ensuring data security and allowing the approach to be applied to private and sensitive datasets without compromising confidentiality.

In the next chapter, we plan to use the suspicious cases obtained from X-ITERADE to implement an algorithm that identifies cases closest to the suspicious group. We aim to integrate our approach with a human-in-the-loop system, allowing for the provision of actual labels for these specific instances. With sufficient labeled examples, the unsupervised problem can transition into an AL or SSL scenario, enabling the model to learn from a limited dataset while progressively expanding its labeled set.

Chapter 5

ALISA: Adaptive Learning through Iterative Semi-supervised Active learning

In the previous chapter, we introduced X-ITERADE, an unsupervised anomaly detection framework designed to address the absence of labels in fraud detection scenarios. X-ITERADE demonstrated the ability to select a high-quality subset of the dataset, referred to as suspicious cases, within a flexible budget, achieving a 3 to 15-fold improvement in the proportion of true positive cases compared to random selection. Building on this output, this chapter introduces ALISA (Adaptive Learning through Iterative Semi-supervised Active learning), a novel framework that transforms the unsupervised setting into an iterative, label-efficient learning process. By leveraging the initial suspicious cases identified by X-ITERADE, ALISA constructs an initial labeled set and progressively refines it through enhanced AL and SSL strategies. The pipeline also integrates targeted data augmentation to further improve performance. While applicable to both single-class and two-class

scenarios, ALISA achieves its strongest performance in highly imbalanced settings, such as fraud detection, where only one class is initially available for training.

5.1 Introduction

While large volumes of transaction data are readily available in financial systems, high-quality labeled data is scarce and costly. Labeling requires expert analysis, manual effort, and time, making it labor-intensive. Several challenges worsen this: First, financial data is often imbalanced, with a vast majority of transactions being legitimate and only a small fraction representing fraudulent activity. Second, the dynamic and evolving nature of fraud schemes means that labeling criteria must be continuously updated, requiring ongoing expert oversight. Third, the subjective nature of many fraud indicators can lead to inconsistencies in labeling, even among experienced analysts. Finally, privacy and regulatory constraints limit the sharing and annotation of sensitive financial data, further restricting the pool of available labeled examples.[1] These challenges make traditional supervised learning strategies less effective in fraud detection settings, as they rely heavily on abundant, high-quality labeled data for training. This resource is often scarce and costly to obtain.

In Chapter 3, we explored how AL and SSL can serve as promising alternatives to supervised approaches when label availability is limited. Unlike standard models that fail in low-label scenarios, AL selects the most informative samples for annotation, and SSL uses the confident cases in the training set to pseudo-label the unlabeled data and improve learning. However, fraud detection presents an even more complex challenge: the class distribution is highly imbalanced, and fraud instances may be entirely absent from

[1] https://fastercapital.com/topics/challenges-in-financial-data-labeling-and-how-to-overcome-them.html

the labeled training set. This makes many AL and SSL techniques ineffective, as they typically assume at least minimal class diversity to operate correctly. Often, only a small fraction of cases can be manually reviewed due to cost and resource constraints, and when working with highly imbalanced datasets, even this small subset may contain no fraud cases at all. In such one-class scenarios, most traditional AL and SSL algorithms break down. Popular techniques like uncertainty sampling, query-by-committee, self-training, and co-training rely on classifiers trained with examples from multiple classes and cannot function effectively when all labeled samples belong to the same class.

In this chapter, we introduce ALISA (Adaptive Learning through Iterative Semi-supervised Active learning), a novel learning framework designed to overcome these challenges. ALISA begins with the most suspicious examples identified by X-ITERADE and trains an unsupervised autoencoder to model the suspicious behavior. It then queries a small subset of unlabeled cases for manual labeling to initialize the labeled set. From there, ALISA iteratively combines AL, SSL, and data augmentation in a flexible and modular pipeline to expand the labeled dataset and improve detection accuracy.

ALISA introduces a weighted AL ensemble, including a one-class query strategy that remains effective even when only one class is present. It also provides an enhanced self-training module to boost SSL performance in sparse or imbalanced settings. With its multi-parameter design, ALISA can adapt to various datasets and operational conditions, making it broadly applicable.

Our experiments show that ALISA delivers its greatest performance gains in one-class scenarios, achieving 10-15% improvement in F1_Score and 60-80% improvements in Precision compared to the best perfroming baseline when no fraud cases are initially available. Even in two-class scenarios with partial fraud presence, ALISA improves the initial F1_Score performance by 10–30%, demonstrating its effectiveness in both challenging and

moderately supervised environments.

Contributions: Our main contributions are as follows:

- We propose ALISA, a novel modular pipeline that integrates data augmentation, enhanced SSL, and a new weighted AL ensemble. The weighted nature of this framework dynamically prioritizes more effective strategies during training and switches to SSL and vice versa when appropriate. ALISA is designed to remain effective even in challenging settings where only one class is present.
- We conduct an extensive set of experiments to evaluate the impact of various hyperparameters within ALISA. Our results demonstrate the framework's adaptability to different datasets and imbalance ratios, highlighting its practical flexibility across diverse conditions.

Organization: The remainder of this chapter is structured as follows: Section 5.2 introduces our proposed framework, ALISA. Section 5.3 outlines the experimental setup and configuration. In Section 5.4, we present and analyze the experimental results. Finally, Section 5.5 concludes the chapter.

5.2 ALISA: Adaptive Learning through Iterative Semi-supervised Active learning

ALISA: Adaptive Learning through Iterative Semi-supervised Active learning, is a flexible, weighted, and iterative framework that integrates five key components: (1) budget configuration, (2) an autoencoder-based data selection mechanism, 3) data augmentation, 4) AL, and 5) SSL. It is designed to begin with minimal or even one-class labeled data,

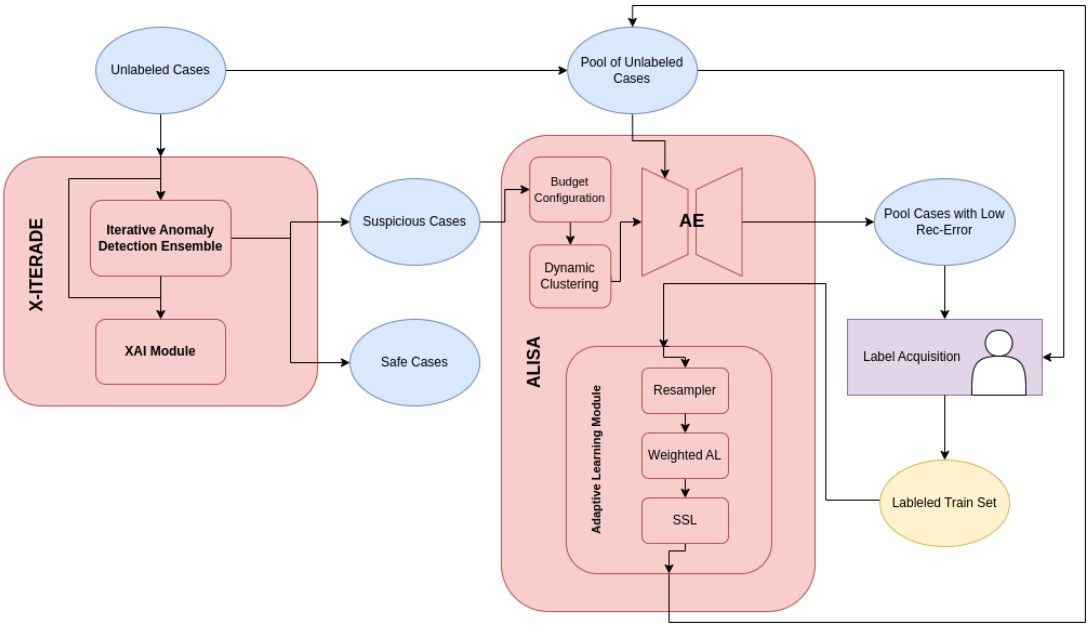


Figure 5.1: Overview of ALISA's architecture and data flow, starting from the suspicious cases flagged by X-ITERADE and proceeding through its iterative learning process.

adaptively expand the labeled dataset in an informative manner, and transition between AL and SSL strategies based on performance signals and dataset characteristics. The main goal of ALISA is to enhance traditional AL strategies while minimizing labeling costs by strategically switching between AL and SSL when appropriate. ALISA also introduces a data augmentation module that makes it suitable even for very small labeled datasets. Importantly, it provides a mechanism to begin the learning process even when the initial labeled set contains only one class, a common issue in highly imbalanced settings like fraud detection.

Figure 5.1 provides a high-level overview of ALISA's key components and data flow, which begins by building on the output of X-ITERADE. Full details of the ALISA modules and their integration can be found in Algorithm 5.1.

Algorithm 5.1 ALISA Framework for Fraud Detection

procedure ALISA($X_{\text{suspicious_train}}$, X_{pool}, y_{pool}, *test_sets*, *budget*, *num_iterations*, *hyperparams*)
\\$X_{\text{suspicious_train}}$: Suspicious unlabeled training data from X-ITERADE; $X_{\text{pool}}, y_{\text{pool}}$: Unlabeled pool data and labels; *test_sets*: One or more test sets for evaluation; *budget*: Budget used to filter suspicious data; *num_iterations*: Number of active learning iterations; *hyperparams*: Includes thresholds, weights, n, strategies, ratio, random_state etc.
    $X_{\text{suspicious_train}} \leftarrow$ ConfigureBudget($X_{\text{suspicious_train}}, budget$)
    $(X_{\text{train}}, y_{\text{train}}) \leftarrow$ SelectLowErrorByCluster($X_{\text{suspicious_train}}, X_{\text{pool}}, y_{\text{pool}}, ratio, random_state$)
    Initialize weights $(w_{AL}, w_{SSL}, w_{us}, w_{ae})$
    Set thresholds $(\theta_{\text{AE}}, \alpha, \delta, \tau)$ and n
    for $step = 1$ **to** *num_iterations* **do**
        if class diversity is sufficient **then**
            Augment X_{train} with resampling
            Train $\mathcal{M}$ on X_{train} and evaluate on *test_sets*
            Select strategies based on (w_{AL}, w_{SSL})
            if "do_nothing" selected **then continue**
            end if
            if Parallel strategy **then**
                $(X_{al}, y_{al}, X_{\text{pool}}, y_{\text{pool}}, w_{us}, w_{ae}) \leftarrow$ ALWithUncertaintyAE($X_{\text{train}}, y_{\text{train}}, X_{\text{pool}}, y_{\text{pool}}, \mathcal{M}, n, \theta_{\text{AE}}, w_u, w_{ae}, \alpha, \tau$)
                $(X_{ssl}, y_{ssl}, X_{\text{pool}}, y_{\text{pool}}) \leftarrow$ SelfTrainingWithCorrection($X_{\text{train}}, X_{\text{pool}}, y_{\text{pool}}, n, \delta, strategy_{SSL}$)
                Merge the new cases from (X_{al}, y_{al}) and (X_{ssl}, y_{ssl}) and update the training set
            else if Sequential strategy **then**
                $(X_{al}, y_{al}, X_{\text{pool}}, y_{\text{pool}}, w_{us}, w_{ae}) \leftarrow$ ALWithUncertaintyAE($X_{\text{train}}, y_{\text{train}}, X_{\text{pool}}, y_{\text{pool}}, \mathcal{M}, n, \theta_{\text{AE}}, w_u, w_{ae}, \alpha, \tau$)
                $(X_{\text{train}}, y_{\text{train}}, X_{\text{pool}}, y_{\text{pool}}) \leftarrow$ SelfTrainingWithCorrection($X_{al}, y_{al}, X_{\text{pool}}, y_{\text{pool}}, n, \delta, strategy_{SSL}$)
            end if
            if augmentation removal enabled **then**
                Remove synthetic samples from training set
            end if
            Retrain $\mathcal{M}$ and update weights (w_{AL}, w_{SSL})
        else
            Use AE-based querying for anomaly detection
            Retrain model on updated training set
        end if
    end for
    Save evaluation metrics to CSV
    return updated $X_{\text{train}}, y_{\text{train}}$
end procedure

ALISA begins with a set of suspicious cases obtained from X-ITERADE. Since X-

ITERADE does not include a predefined stopping condition and labeling resources are limited, we first define a budget constraint and select only those X-ITERADE variants that fall within this limit. When multiple eligible X-ITERADE variants are available, we merge the suspicious cases across them and eliminate duplicates. Additionally, if a single variant produces suspicious cases across several iterations within the budget, we give priority to cases from earlier iterations to minimize both training set size and labeling cost.

Instead of using these cases directly as the training set, ALISA first dynamically clusters the data and trains an autoencoder on the data to capture their underlying representation. Then, reconstruction errors are computed on the pool set using this autoencoder. In each cluster, instances with the lowest reconstruction error, that is, those most similar to suspicious cases, are selected and labeled. These selected cases form a flexible training set for ALISA, allowing control over its size and removing the dependence on fixed budgets from X-ITERADE. The details of this approach are present in Algorithm 5.2.

The core of ALISA is an iterative algorithm. At each iteration, one of two main paths is followed depending on the class diversity in the labeled set:

- **Case 1: Only one class present**: ALISA applies autoencoder-based AL, where an autoencoder trained only on non-fraudulent cases is used to score pool instances by reconstruction error. Cases with the highest errors are queried and added to the labeled set, as they are likely to be anomalies.

- **Case 2: More than one class present**: A data augmentation module is activated to dynamically improve class balance. Rather than fully balancing the dataset, that is an approach that risks introducing bias, ALISA applies moderate rebalancing guided by the original imbalance ratio. This ensures stability while still enhancing the

Algorithm 5.2 Cluster-Based Selection Using Autoencoder Reconstruction Errors

procedure SELECTLOWERRORBYCLUSTER($df_{\text{suspicious_train}}$, X_{pool}, y_{pool}, *ratio*, *random_state*)
\\$df_{\text{suspicious_train}}$: Unlabeled suspicious training set from X-ITERADE; $X_{\text{pool}}, y_{\text{pool}}$: Unlabeled pool data and corresponding labels; *ratio*: Proportion of top instances to select per cluster; *random_state*: Random seed for reproducibility.
    $n_{\text{clusters}} \leftarrow \text{find_k}(X_{\text{pool}}, \textit{random_state})$
    $C \leftarrow \text{KMeans}(X_{\text{pool}}, n_{\text{clusters}})$
    Train autoencoder AE_{susp} on $df_{\text{suspicious_train}}$
    Initialize empty dataframe $df_{\text{top_n}}$
    for each cluster c in $0 \dots n_{\text{clusters}} - 1$ **do**
        $X_c \leftarrow \{x \in X_{\text{pool}} \mid C(x) = c\}$
        $E_c \leftarrow \text{ReconstructionErrors}(AE_{\text{susp}}, X_c)$
        $n \leftarrow \text{int}(\textit{ratio} \cdot |X_c|) + 1$
        Select n instances with lowest E_c from X_c as X_{top}
        Append X_{top} to $df_{\text{top_n}}$
    end for
    Add instances in $df_{\text{top_n}}$ to $X_{\text{train}}, y_{\text{train}}$
    Remove them from $X_{\text{pool}}, y_{\text{pool}}$
    return $X_{\text{train}}, y_{\text{train}}$
end procedure

minority class presence. Two resampling strategies are available:

- Gaussian Noise Resampler
- Random Oversampler

As shown in Figure 5.2, synthetic samples generated by these methods are added to the training set and passed to the AL and SSL modules according to the selected execution architecture:

- Sequential: Apply AL first, then SSL.
- Parallel: Apply both AL and SSL concurrently.

If augmentation removal is enabled, synthetic instances are removed after each iteration, ensuring that only newly generated samples are used in each round. Both the AL and SSL strategies are weighted and updated in each iteration based on their

effectiveness. Effectiveness is measured as the number of correctly labeled frauds in the previous round. If the "weighted-n query" strategy is used, the number of queries allocated to AL and SSL is also adjusted based on their performance, favoring the more successful method. This adaptive approach helps reduce labeling costs, especially in scenarios where SSL consistently outperforms AL.

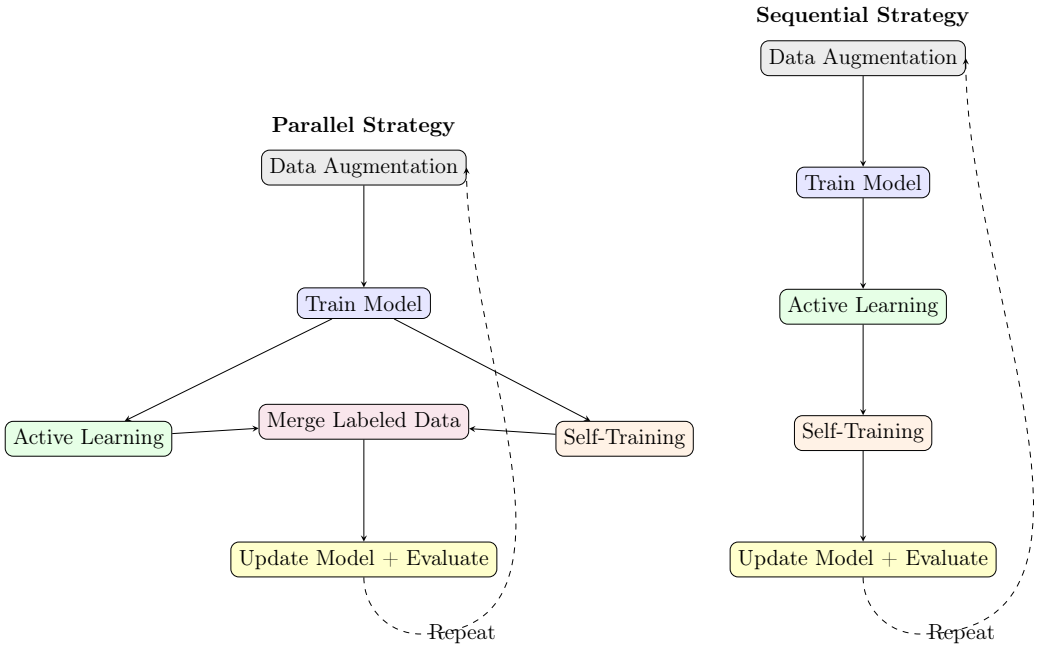


Figure 5.2: ALISA execution modes: Parallel strategy (Left) and Sequential strategy (Right).

ALISA has two key components applied in the parallel or sequential execution modes: the Weighted Active Learning Strategy (displayed in Algorithm 5.3), and the Self-Supervised Learning Strategy (displayed in Algorithm 5.4).

The Weighted Active Learning Strategy combines two techniques:

- **Autoencoder-based selection:** Identifies outliers based on reconstruction error.

- **Uncertainty sampling:** Selects instances with the highest uncertainty from a classifier.

Scores from both methods are combined using a weighted average. The weights are updated in each round based on how many frauds each method contributed to the labeled set. If, due to poor performance, the weight of the autoencoder falls below a specified threshold, it is excluded from future rounds to avoid computational overhead.

ALISA's Self-Supervised Learning Strategy employs a self-training approach with false-positive correction, combining:

- **Confidence-based selection:** Instances are sorted by the classifier's prediction confidence.
- **Entropy-based selection (optional):** Instances are ranked by entropy, with either high or low entropy selected depending on the setup. After ranking, ALISA selects batches of pseudo-labeled instances. If a batch contains no fraud cases, it moves to the next batch until at least one fraud is found.

To ensure high data quality, pseudo-label correction is applied only to the fraud-labeled samples, increasing reliability and reducing the risk of error propagation.

Finally, the ALISA algorithm stops once all iterations are completed, indicating that the entire labeling budget has been finished. The algorithm outputs evaluation metrics as well as the updated training set with corresponding labels, enabling future monitoring of the newly added cases.

Algorithm 5.3 Active Learning with Uncertainty Sampling and Autoencoder Ensemble

procedure ALWITHUNCERTAINTYAE(X_{train}, y_{train}, X_{pool}, y_{pool}, $\mathcal{M}$, n, θ_{AE}, w_u, w_{ae}, α, τ) \\$X_{\text{train}}, y_{\text{train}}$: Labeled training set and labels; $X_{\text{pool}}, y_{\text{pool}}$: Unlabeled pool set and corresponding labels for querying; $\mathcal{M}$: Classifier used to compute uncertainty; n: Number of instances to query; θ_{AE}: Autoencoder hyperparameters; (w_u, w_{ae}): Weights for uncertainty and AE components; α: Smoothing factor for weight update; τ: Threshold to deactivate AE if its weight is too low.
    Train classifier $\mathcal{M}$ on $(X_{\text{train}}, y_{\text{train}})$
    Compute uncertainty scores (e.g., entropy) for X_{pool} and normalize
    if $w_{ae} < \tau$ **then** ▷ AE is ignored if its contribution is too low
        Select top-n uncertain instances
        Move selected samples from pool to train
        return updated $(X_{\text{train}}, y_{\text{train}}, X_{\text{pool}}, y_{\text{pool}})$, selected indices, (w_u, w_{ae})
    end if
    Train autoencoder $\mathcal{AE}$ on non-fraud cases in X_{train} using θ_{AE}
    Compute and normalize AE reconstruction errors on X_{pool}
    Identify top-n indices from both uncertainty and AE scores
    Compute ensemble scores: $s = w_u \cdot s_u + w_{ae} \cdot s_{ae}$
    Select top-n instances with highest ensemble scores
    Move selected samples from pool to train
    Count frauds selected by uncertainty (frauds_u) and AE (frauds_{ae})
    $\text{total} \leftarrow \text{frauds}_u + \text{frauds}_{ae}$
    if $\text{total} > 0$ **then**
        $w_u \leftarrow \alpha \cdot w_u + (1 - \alpha) \cdot \frac{\text{frauds}_u}{\text{total}}$
        $w_{ae} \leftarrow \alpha \cdot w_{ae} + (1 - \alpha) \cdot \frac{\text{frauds}_{ae}}{\text{total}}$
    end if
    return updated $(X_{\text{train}}, y_{\text{train}}, X_{\text{pool}}, y_{\text{pool}}, w_u, w_{ae})$
end procedure

Algorithm 5.4 Pseudo-Labeling with Confidence Filtering and Correction

procedure SELFTRAININGWITHCORRECTION(X_{train},y_{train}, X_{pool}, y_{pool}, n, δ, s)
\\ $X_{\text{train}}, y_{\text{train}}$: Labeled training set and labels; $X_{\text{pool}}, y_{\text{pool}}$: Unlabeled pool set and corresponding labels (used only for correction); n: Number of pseudo-labeled instances to add per iteration; δ: Confidence threshold for filtering predictions; s: Selection strategy: "normal ST", "high entropy ST", or "low entropy ST".
    Train XGBoost model $\mathcal{M}$ on $(X_{\text{train}}, y_{\text{train}})$
    Predict pseudo-labels and associated probabilities on X_{pool}
    Compute entropy and confidence per instance
    Filter pool to retain only instances with confidence $\geq \delta$
    if $s =$ "normal ST" **then**
        Select top-n most confident instances
    else if $s =$ "high entropy ST" or "low entropy ST" **then**
        Sort filtered pool by entropy (descending if "high", ascending if "low")
        while not enough confident frauds selected **do** ▷ Repeat until a batch containing at least one pseudo-labeled fraud instance is found.
            Inspect next n candidates in order
            if batch contains at least one fraud **then**
                Select batch
                break
            end if
        end while
    else
        return unchanged sets (invalid strategy)
    end if
    Correct false positives using y_{pool}
    Add selected instances to $(X_{\text{train}}, y_{\text{train}})$
    Remove them from $(X_{\text{pool}}, y_{\text{pool}})$
    return updated $(X_{\text{train}}, y_{\text{train}}, X_{\text{pool}}, y_{\text{pool}})$
end procedure

5.3 Experimental Settings

5.3.1 Datasets

We conducted our experiments using the CCT and BAF datasets, both of which were introduced in Section 3.2.1. The same preprocessing strategies described in that section were applied. We adopted the realistic imbalance ratios obtained from our industry partner, as detailed in Section 4.3.1. For the training sets, we used the suspicious cases identified by the X-ITERADE framework. To ensure consistency across each imbalance ratio, we selected suspicious cases generated under labeling budgets ranging from 1,000 to 1,500. These suspicious cases, along with their true labels, were then passed to the autoencoder strategy, which queried instances with the lowest reconstruction errors. For each imbalance setting, we created training subsets representing 0.05%, 0.1%, 0.15%, and 0.2% of the pool, corresponding to approximately 250, 500, 750, and 1000 cases, respectively. Each sampling rate included several variations, leading to a total of 32 training subsets.

The pool set consisted of around 500,000 randomly selected cases drawn in a stratified manner from the portion of the original data that had not been used for training X-ITERADE. We maintained the original class imbalance in the pool set.

To introduce variability and robustness in evaluation, we generated five random test sets, each selected in a stratified manner without any overlap between them. The original imbalance ratio was preserved in all test sets. We evaluated ALISA across all five test sets, and the average performance was reported.

The detailed breakdown of dataset sizes and class distributions for the training, pool, and test sets is presented in Tables 5.1 and 5.2.

Table 5.1: Fraud and non-fraud distribution across different suspicious case subsets for BAF and CCT datasets, obtained from X-ITERADE and used as input to the autoencoder.

BAF Dataset				**CCT Dataset**			
Subset	**Frauds**	**Non-Frauds**	**IR %**	**Subset**	**Frauds**	**Non-Frauds**	**IR %**
Suspicious Cases High	12	2448	0.49	Suspicious Cases High	2	1084	0.18
Suspicious Cases Original	21	2595	0.81	Suspicious Cases Original	33	1319	2.50
Suspicious Cases Low	949	2343	40.50	Suspicious Cases Low	689	2256	30.54
Suspicious Cases Med	370	2345	15.78	Suspicious Cases Med	70	934	7.49

Table 5.2: Fraud and non-fraud distribution across different subsets for BAF and CCT datasets, obtained from the autoencoder and used for training and testing ALISA.

Split	**BAF Dataset**			**CCT Dataset**		
	Frauds	**Total**	**IR %**	**Frauds**	**Total**	**IR %**
Train High 250 - 0.05%	15	256	5.86	0	255	0.00
Train Original 250 - 0.05%	1	256	0.39	0	255	0.00
Train Low 250 - 0.05%	1	256	0.39	3	255	1.18
Train Med 250 - 0.05%	1	256	0.39	5	255	1.96
Train High 500 - 0.1%	18	505	3.56	0	505	0.00
Train Original 500 - 0.1%	0	505	0.00	0	505	0.00
Train Low 500 - 0.1%	8	505	1.58	7	505	1.39
Train Med 500 - 0.1%	8	505	1.58	11	505	2.18
Train High 750 - 0.15%	24	757	3.17	0	754	0.00
Train Original 750 - 0.15%	2	757	0.26	0	754	0.00
Train Low 750 - 0.15%	13	757	1.72	11	754	1.46
Train Med 750 - 0.15%	8	757	1.06	14	754	1.86
Train High 1000 - 0.2%	32	1007	3.18	0	1004	0
Train Original 1000 - 0.2%	4	1007	0.4	0	1004	0
Train Low 1000 - 0.2%	16	1007	1.59	12	1004	1.2
Train Med 1000 - 0.2%	10	1007	0.99	15	1004	1.49
Pool	5517	501743	1.10	647	499246	0.13
Test 1	537	49400	1.10	48	49400	0.10
Test 2	537	49400	1.10	48	49400	0.10
Test 3	537	49400	1.10	48	49400	0.10
Test 4	537	49400	1.10	48	49400	0.10
Test 5	538	49400	1.10	48	49400	0.10

5.3.2 Evaluation Methodology

To assess the effectiveness of our framework, we evaluated several performance metrics, including F1_Score, Precision, Recall, and Accuracy. In addition to reporting these metrics, we also computed performance gains in two distinct ways:

- Progressive Gain: For each approach, we tracked the improvement in the evaluated metric from the initial iteration (before any queried instances are added) to the final iteration. This highlights how much each approach improves the performance over time.
- Progressive Gain Over Best Baseline: We calculated the difference between the final progressive evaluated metric of each approach and the highest evaluated metric achieved by any of the six baseline methods on the same dataset. This allows us to evaluate how well each approach performs relative to the best existing baseline strategy.

We compared each method against six baselines, split across two scenarios:

- Two-Class Baselines (used when class diversity is sufficient in the training data):
  - Random Sampling AL: Selects pool instances randomly to be labeled and added to the training set.
  - Uncertainty Sampling AL: Uses uncertainty to select the most ambiguous instances for labeling.
  - Self-Training SSL: Iteratively pseudo-labels confident predictions from the pool and retrains the model.
- One-Class Baselines (used when only non-fraud cases are initially available):

- AL with OCSVM: Trains a OCSVM on the initial non-fraud training set and selects pool instances with the highest anomaly scores (i.e., furthest from the learned boundary) for labeling.
- AL with Autoencoder: Trains an autoencoder on the non-fraud cases and queries pool instances with the highest reconstruction errors.
- Pseudo-Labeling with OCSVM: Applies a OCSVM to the pool and pseudo-labels instances based on anomaly scores. Instances scoring above a threshold (defined as the mean + standard deviation of the anomaly scores) are labeled as fraud; others are labeled as non-fraud. These pseudo-labeled instances are then added to the training set without access to true labels.

Together, these baselines represent a diverse set of strategies from both AL and SSL paradigms. Our goal is to demonstrate not only how much each method improves upon its own starting point, but also how it compares against the strongest available alternative.

For our base classifier in ALISA, which is used to evaluate model performance at each iteration, we employed XGBoost due to its robustness, scalability, and effectiveness in handling imbalanced classification tasks.

The ALISA framework involves several configurable hyperparameters that influence its behavior and performance. These configurations are presented in Table 5.3.

Table 5.3: Hyperparameters in the ALISA Framework

Hyperparameter	**Description**	**Values Experimented**
Resampling Strategy	Strategy used to augment the training data with synthetic fraud cases.	Gaussian resampler, Random Oversampler
SSL Strategy	Method to select pseudo-labeled data based on confidence and entropy.	normal, entropy-high, entropy-low
Confidence Threshold	Minimum confidence required to pseudo-label a case.	0.95
Augmented Data Retention	Whether augmented data is kept after each iteration.	keep, remove
Weighting Strategy	Whether to assign dynamic weights to AL/SSL based on performance or use a static number of queries.	dynamic, static
Number of Iterations	Number of total iterations in the ALISA process.	15
Queries per Iteration	Number of queried cases added to training per iteration.	50
Strategy Thresholds	Minimum weight required for AL or SSL to be applied in an iteration.	0.25 (dynamic), 0.35 (static)
Execution Mode	Whether AL and SSL steps run sequentially or in parallel.	sequential, parallel

We systematically explored 30 different configurations by varying these hyperparameters, aiming to evaluate the effectiveness and adaptability of ALISA under diverse settings. To reduce the search space, we focused only on configurations that demonstrated promising performance and excluded combinations with consistently poor results. For instance, instead of testing all combinations involving static number of queries and Random Over Sampling strategies, we evaluated only their most effective variants. These configurations are summarized in the Table 5.4.

#	Resampler	SSL Strategy	Execution Mode	Aug Handling	n_queries	Decision Threshold
1	GR	high entropy	parallel	keep	Static - 50	0.35
2	GR	high entropy	parallel	remove	Static - 50	0.35
3	GR	normal	sequential	keep	Static - 50	0.35
4	GR	high entropy	sequential	remove	Static - 50	0.35
5	GR	high entropy	sequential	keep	Static - 50	0.35
6	GR	low entropy	sequential	remove	Static - 50	0.35
7	GR	low entropy	parallel	keep	Dynamic	0.25
8	GR	low entropy	parallel	remove	Dynamic	0.25
9	GR	high entropy	parallel	keep	Dynamic	0.25
10	GR	high entropy	parallel	remove	Dynamic	0.25
11	GR	normal	parallel	remove	Dynamic	0.25
12	GR	normal	parallel	keep	Dynamic	0.25
13	GR	normal	sequential	keep	Dynamic	0.25
14	GR	normal	sequential	remove	Dynamic	0.25
15	GR	high entropy	sequential	keep	Dynamic	0.25
16	GR	high entropy	sequential	remove	Dynamic	0.25
17	GR	low entropy	sequential	remove	Dynamic	0.25
18	GR	low entropy	sequential	keep	Dynamic	0.25
19	ROS	normal	sequential	keep	Static - 50	0.35
20	ROS	high entropy	sequential	remove	Static - 50	0.35
21	ROS	high entropy	parallel	keep	Static - 50	0.35
22	ROS	high entropy	parallel	remove	Static - 50	0.35
23	ROS	normal	sequential	keep	Dynamic	0.25
24	ROS	normal	sequential	remove	Dynamic	0.25
25	ROS	high entropy	sequential	keep	Dynamic	0.25
26	ROS	low entropy	sequential	remove	Dynamic	0.25
27	ROS	normal	parallel	remove	Dynamic	0.25
28	ROS	normal	parallel	keep	Dynamic	0.25
29	ROS	high entropy	parallel	keep	Dynamic	0.25
30	ROS	low entropy	parallel	remove	Dynamic	0.25

Table 5.4: Summary of experimental configurations for ALISA

5.4 Results and Discussion

Figures 5.3 and 5.4 illustrate the progressive F1 gain achieved by the best-performing ALISA approach compared to (1) its initial performance and (2) the final performance relative to the best-performing baseline. These results demonstrate that ALISA consistently improves performance both over its own starting point and compared to the best baseline methods. The improvements are particularly significant on the CCT dataset, which includes more one-class scenarios, where ALISA outperforms the strongest baseline by 5% to 25% in F1_Score. On the BAF dataset, the gains are more modest, ranging between 2% and 5%.

The only scenarios where ALISA did not surpass the baselines in F1_Score occurred in the BAF High setting for sizes 250, 500, and 1000, and in BAF Low at size 750. However, it is important to note that in these cases, the baselines themselves also performed poorly, suggesting that the data in these configurations may be particularly challenging.

Focusing on the Precision metric, a valid option given the high cost of labeling and the importance of reducing false positives, ALISA still maintains a strong advantage. Even in the cases where it does not outperform in terms of F1 gain, it consistently outperforms the baselines in precision gain. This trend is illustrated in Figure 5.5, which reports the Precision improvement achieved by the best-performing ALISA approach in terms of F1_Score compared to the strongest baseline. On average, ALISA achieves an average of 20–40% improvement in Precision on the BAF dataset, and a substantial 60–80% improvement on CCT. These results are illustrated in Figure 5.6, which presents the distribution of F1_Score and precision gains achieved by all ALISA variants compared to the baseline methods across different dataset configurations. Each violin plot represents the distribution of performance gains for the ALISA approaches, while the dashed horizontal lines

indicate the corresponding performance of the baseline methods for each dataset variant.

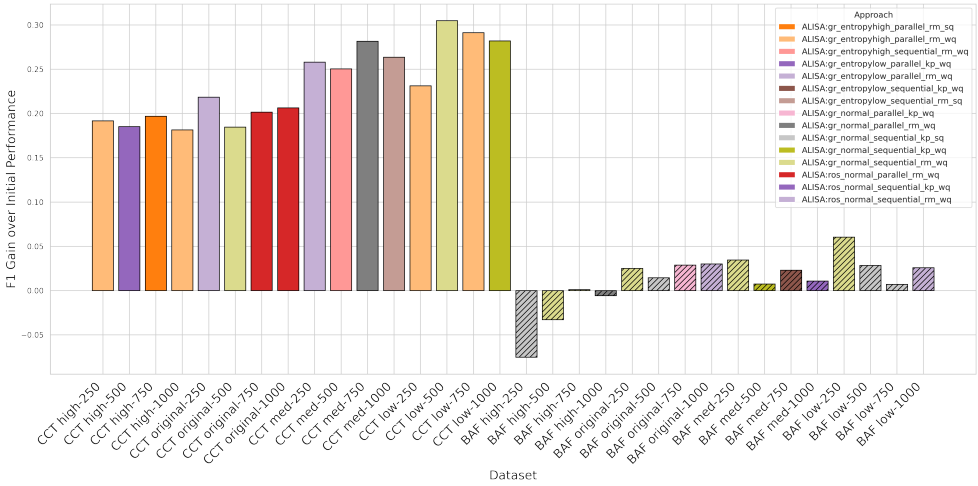


Figure 5.3: Progressive F1 Gain over Initial Performance (Best Approach Per Dataset Variant): Each color represents the ALISA method that achieved the best performance on a specific dataset variant. Solid and dashed bars correspond to CCT and BAF datasets, respectively.

Analyzing the best-performing ALISA variants, we observed the following distribution:

(1) ALISA:gr_normal_sequential_rm_wq was the best approach in 6 cases.

(2) ALISA:gr_entropyhigh_parallel_rm_wq was the best in 4 cases.

(3) ALISA:gr_entropylow_parallel_rm_wq led in 3 cases.

After these, the self-training baseline emerged as the most competitive non-ALISA method, ranking best in 2 instances.

A detailed breakdown of the gain of F1_Score in all ALISA variants and baseline methods is presented in Figure 5.7. As this figure confirms, the performance improvements on the CCT dataset are generally more substantial than on BAF. Among the baselines, uncertainty sampling was the strongest on CCT, while all baselines struggled significantly on the BAF dataset.

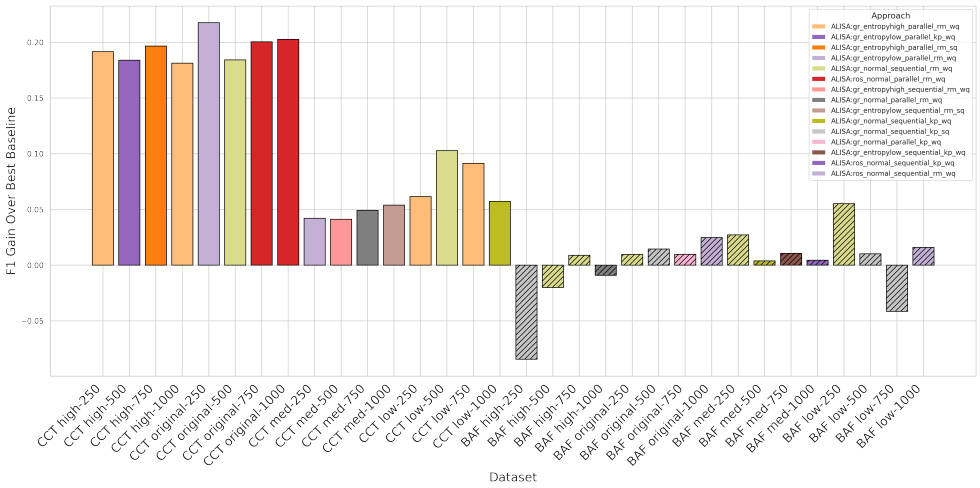


Figure 5.4: F1 Gain over Best Baseline (Best Approach Per Dataset Variant): Each color represents the ALISA method that achieved the best performance on a specific dataset variant. Solid and dashed bars correspond to CCT and BAF datasets, respectively.

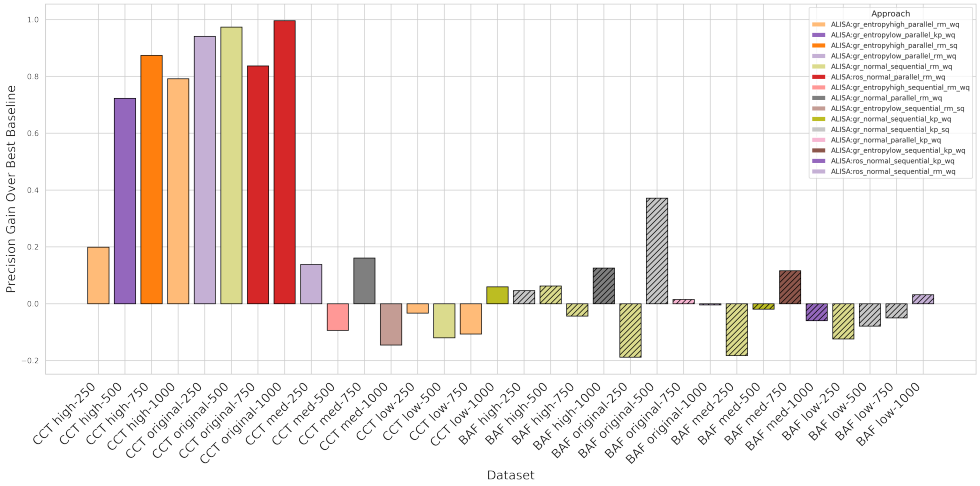


Figure 5.5: Precision Gain over Best Baseline (Best Approach in Terms of F1_Score Per Dataset Variant): Each color represents the ALISA method that achieved the best F1_Score on a specific dataset variant. Solid and dashed bars correspond to CCT and BAF datasets, respectively.

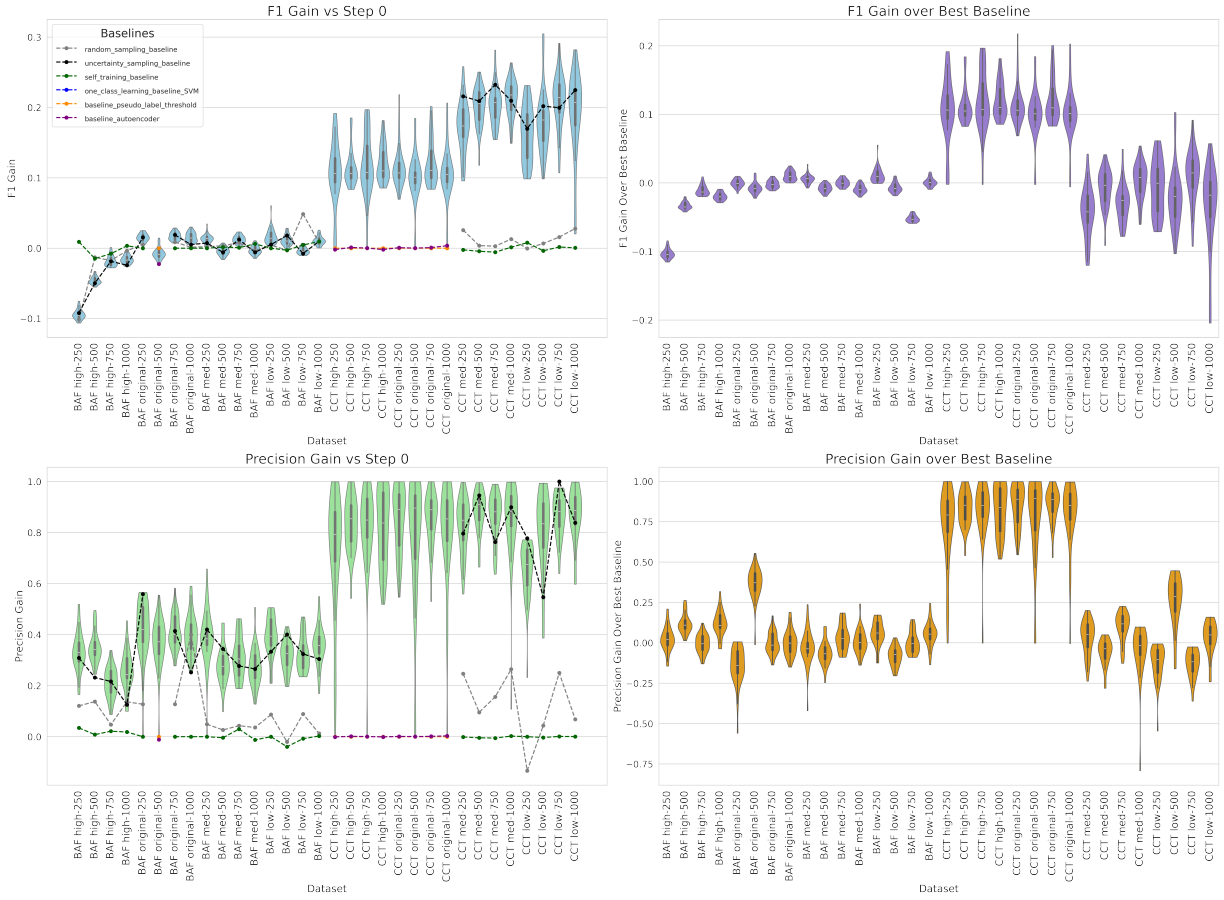


Figure 5.6: Distribution of F1_Scor0e (Top) and Precision (Bottom) gains for each dataset variant. The violin plots represent the performance across all ALISA approaches compared to the initial step (left) and the best-performing baseline (right). Dashed horizontal lines indicate the performance of baseline methods for each dataset.

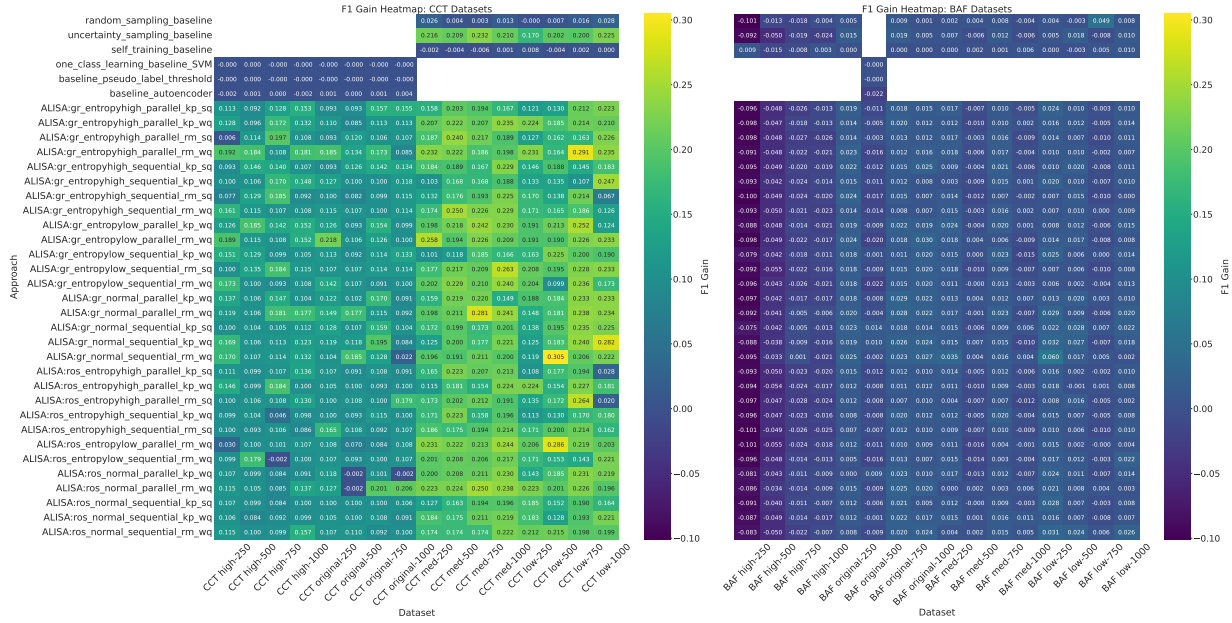


Figure 5.7: Progressive F1 Gain of All ALISA Approaches and the Baselines Compared to Initial Performance on CCT and BAF Datasets. Lighter colors indicate higher F1 gains, representing better performance.

Generally, increasing the size of the initially labeled training set improves performance for both datasets. However, this trend is not strictly linear. For example, increasing the initial size from 750 to 1000 does not always result in meaningful gains. This behaviour can be seen in Figure 5.7. This suggests the existence of an optimal initial training size that balances performance improvements with labeling cost efficiency.

To better illustrate this trend, we aggregated the top-performing and similar ALISA variants and plotted their average F1_Score, comparing them against baseline methods across different imbalance ratios as the initial training size increases (see Figures 5.8 and 5.9). For the CCT dataset, we aggregated all approaches that used the Gaussian resampler, dynamic query sizes, and both sequential and parallel execution modes. For the BAF dataset, we included all variants that employed Normal SSL, the Gaussian resampler, and both execution strategies. Notably, on BAF high, where ALISA faces the greatest challenge in outperforming the self-training baseline, we observed that increasing the initial training size from 250 to 750 leads to improved performance. However, with 1000 initial samples, the performance slightly declined compared to the 750-sample configuration. In the CCT dataset, the parallel execution mode consistently outperformed the sequential mode, whereas in the BAF dataset, both execution modes yielded comparable results.

To avoid overly optimistic conclusions based only on ALISA's best variants, we extended our evaluation in two additional ways:

- We ranked all approaches, including baselines, based on their performance and reported their average rank.
- We computed the frequency of each approach appearing in the top-3 best-performing methods across all runs.

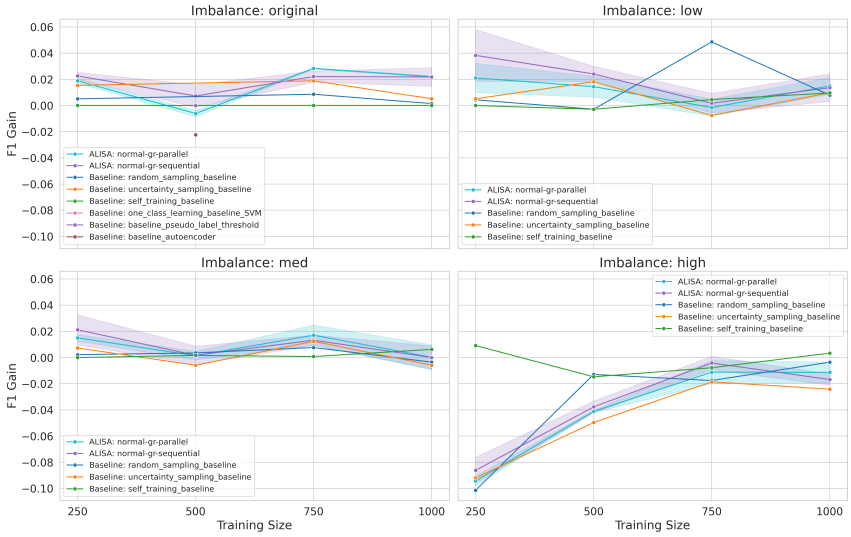


Figure 5.8: Aggregated F1_Scores compared to baselines for the BAF dataset across different imbalance ratios. Approaches aggregated include Gaussian resampling, Normal SSL, and sequential and parallel execution modes.

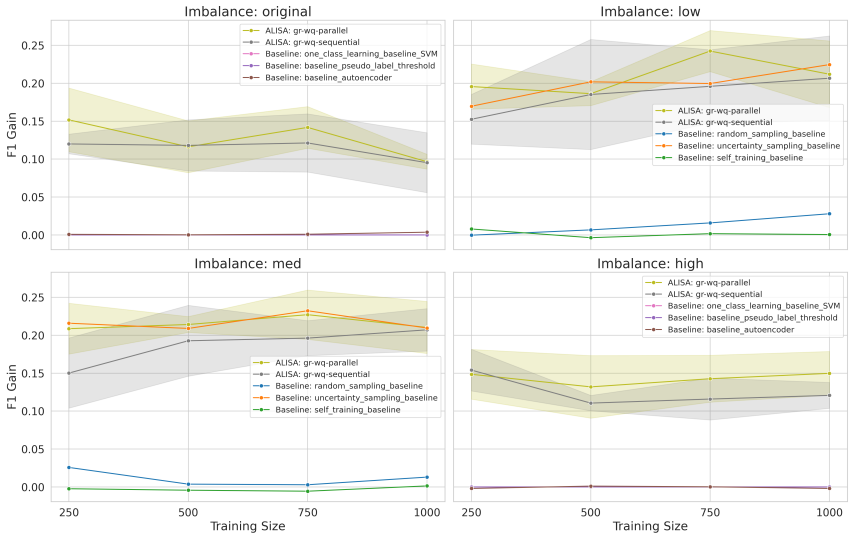


Figure 5.9: Aggregated F1_Scores compared to baselines for the CCT dataset across different imbalance ratios. Approaches aggregated include Gaussian resampling, dynamic queries, and sequential and parallel execution modes.

These analyses are visualized in Figures 5.10, 5.11, 5.12, and 5.13, providing a more comprehensive view of how ALISA compares with alternative methods across different settings.

On the CCT dataset, the results are more consistent, with fewer approaches dominating the top-3 positions and achieving lower average ranks, indicating the presence of a strong, dominant variant. For example, ALISA:gr-entropyhigh-parallel-rm-wq appeared 10 out of 16 times among the top-3 performers.

On the BAF dataset, the most effective variants were ALISA:gr-normal-sequential-rm-wq and ALISA:gr-normal-sequential-kp-wq, each appearing 6 times in the top-3. In terms of average rank, all ALISA configurations consistently outperformed the baselines. The most competitive baseline on BAF was the self-training baseline, which achieved a mean rank of 20 out of 35, further validating the strength of ALISA.

Additionally, an analysis of the top-3 frequency reveals that several ALISA configurations, 8 variants on CCT and 15 on BAF, never ranked among the top-performing methods. On the CCT dataset, the underperforming variants primarily included those using Random Oversampling (5 out of 8) and the Gaussian Resampler combined with Normal SSL while keeping augmented data. On the BAF dataset, underperformance was observed across a broader range of configurations: 8 out of 15 employed Random Oversampling, while 5 of the remaining 7 used the Gaussian Resampler in combination with the entropy-high SSL strategy. These consistently underperforming variants could be removed from the ALISA framework to improve overall efficiency and average performance across both datasets. This pruning also helps in establishing more sensible default values for ALISA's hyperparameters.

Our observations indicate that Gaussian Resampling outperforms Random Oversampling as a data augmentation technique. The former was frequently used in top-performing AL-

ISA variants, whereas the latter appeared more often in configurations that never reached the top-3. We also found that using dynamic weighted queries (wq) led to better results than the static alternative, and this option was more common in the most successful approaches. Regarding the execution method, the parallel mode yielded better results on the CCT dataset, while sequential execution performed better for BAF. Additionally, enabling the option to remove augmented data at the end of each iteration was associated with stronger performance, in contrast to keeping the augmented samples. As for the type of SSL used, the normal SSL variant was more effective on the BAF dataset, while no single SSL type dominated on the CCT dataset. Notably, ALISA:gr-**entropyhigh**-parallel-rm-wq achieved the best performance in terms of both average rank and top-3 frequency. The next two best-performing configurations were ALISA:gr-**normal**-parallel-rm-wq and ALISA:gr-**entropylow**-parallel-rm-wq. A summary of observations and guidelines for the ALISA best performing configuration is presented in Table 5.5.

Table 5.5: Best-performing settings across ALISA hyperparameters

Condition	**Compared Values**	**Best Value**	**Reason for Best Performance**
Resampling Method	GR, ROS	GR	GR was more frequently used in top-performing configurations; ROS appeared more in low-performing ones.
Query Strategy	Dynamic weighted queries (wq), Static queries (sq)	Dynamic (wq)	Dynamic query selection led to better results and was more common among the best variants.
Execution Mode	Parallel, Sequential	Dataset-dependent	Parallel was better for the CCT dataset; Sequential performed better for the BAF dataset.
Augmented Data Handling	Keep (kp), Remove (rm)	Remove (rm)	Removing augmented data after each iteration was linked with stronger performance.
SSL Strategy	Normal, Entropy-high, Entropy-low	Dataset-dependent	Normal SSL was more effective on the BAF dataset; no single SSL type dominated on CCT.

These insights provide valuable guidance for refining ALISA's architecture and setting more effective default parameter values. This data-driven selection not only reduces computational overhead but also simplifies deployment by establishing a robust starting point for hyperparameter tuning.

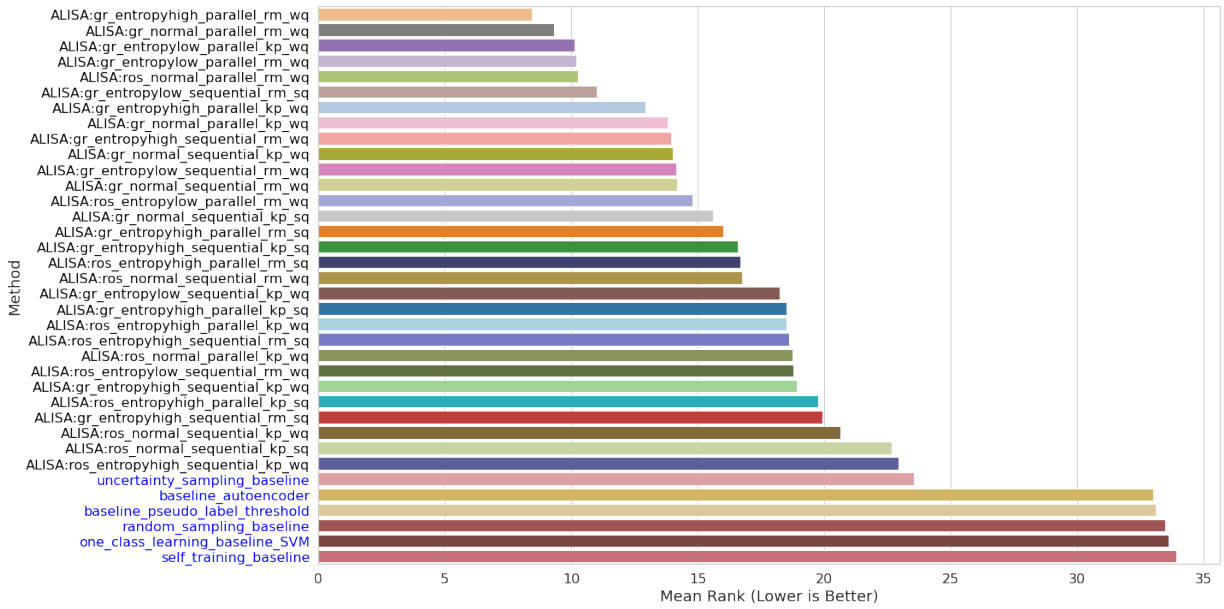


Figure 5.10: Average Rank of ALISA Methods and the Baselines on CCT Datasets (Lower Values Indicate Better Performance)

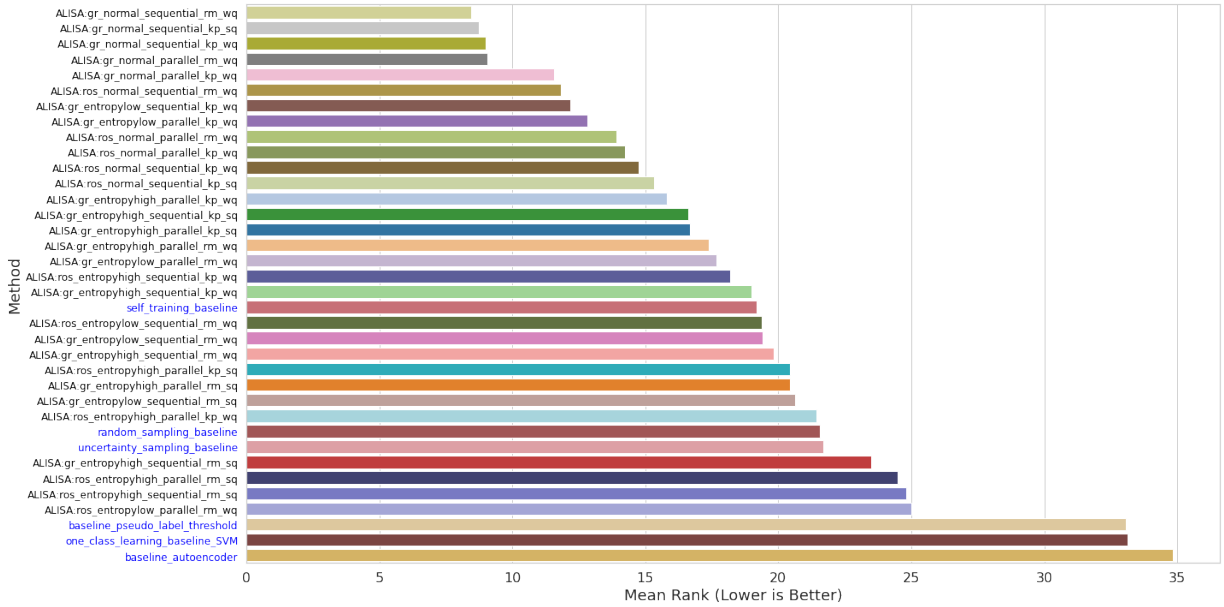


Figure 5.11: Average Rank of ALISA Methods and the Baselines on BAF Datasets (Lower Values Indicate Better Performance)

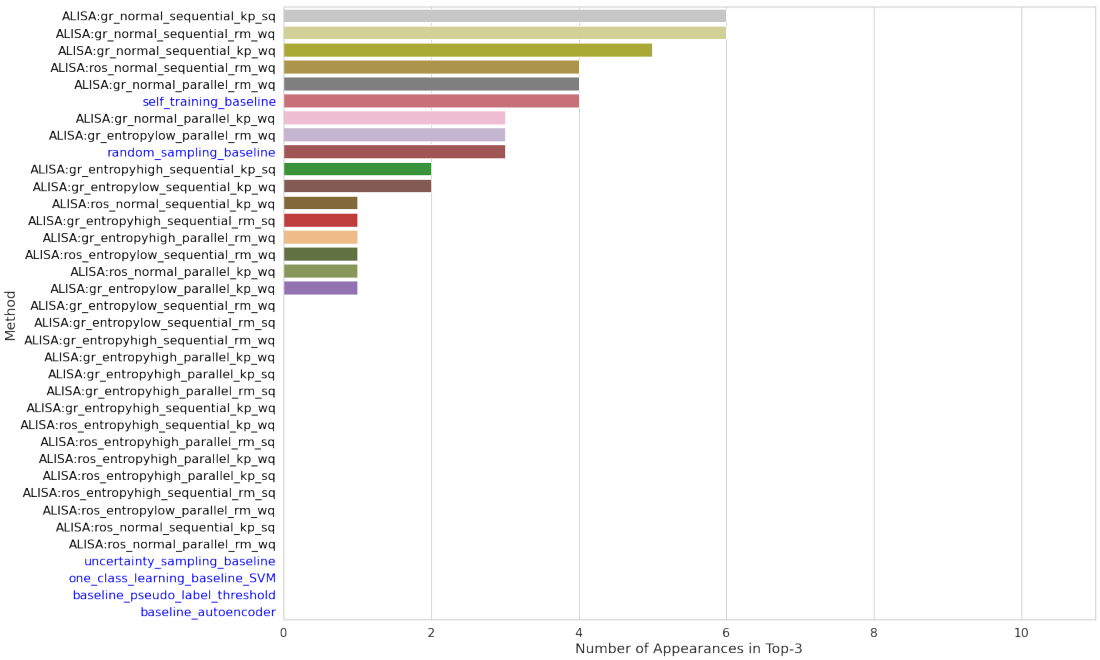


Figure 5.12: Frequency of ALISA Methods Appearing Among the Top-3 Performing Approaches on BAF Datasets.

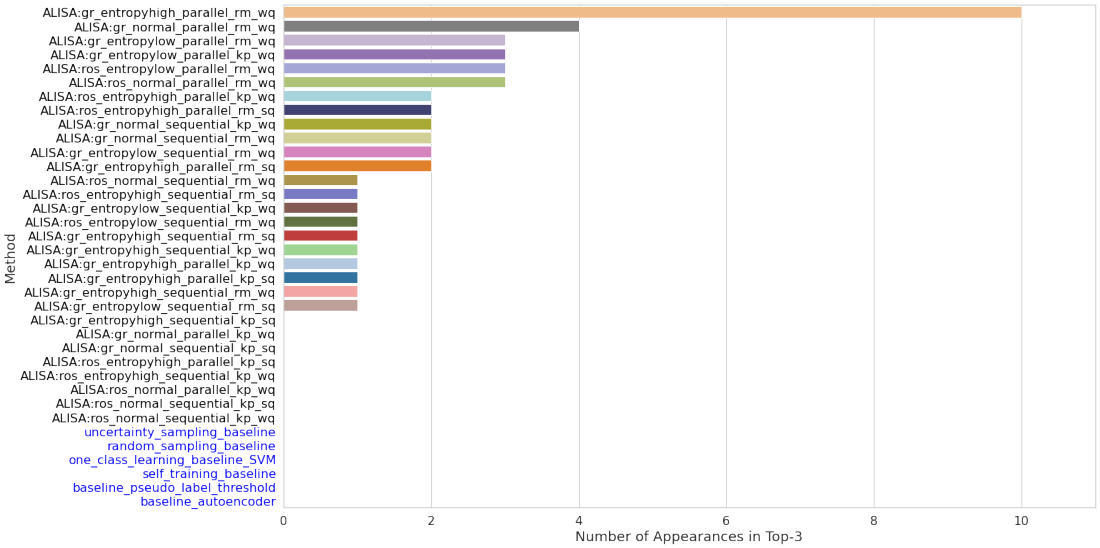


Figure 5.13: Frequency of ALISA Methods Appearing Among the Top-3 Performing Approaches on CCT Datasets.

5.4.1 Limitations

While ALISA demonstrated strong performance improvements, particularly in scenarios where only one class is initially labeled, several limitations remain. First, the improvements on the BAF dataset were not as significant as those observed on CCT, especially in the BAF-High variant, which contains the largest number of fraud cases in the initial labeled set. This behavior may be attributed to the inherent characteristics of the BAF dataset, as even the baseline methods struggled in these settings. As observed in earlier chapters, BAF consistently posed greater challenges across all evaluation strategies. Additionally, since ALISA is built on top of X-ITERADE, its effectiveness is partly dependent on the quality of the base method. Given that X-ITERADE also performed less effectively on BAF compared to CCT, this may explain ALISA's reduced gains in those cases. To strengthen the generalizability and reliability of our findings, future work should expand the evaluation to a broader range of datasets, including domains beyond fraud detection. Moreover, this study focused on improving traditional AL and SSL techniques. Integrating more advanced or recently proposed AL and SSL strategies may further enhance ALISA's applicability across diverse datasets.

Another important consideration is hyperparameter optimization. ALISA includes a wide range of configurable parameters, and tuning these settings can be resource-intensive. In this study, we explored 30 parameter combinations to assess effectiveness. However, a more comprehensive search over a larger space may lead to better performance and reveal new insights.

Finally, although we have developed an explainability module for X-ITERADE, it has not yet been integrated into the iterative process of ALISA. Incorporating explainability into each iteration of ALISA, by generating explanations for the queried cases added to the training set, could significantly enhance the transparency and trustworthiness of the

system. This would help stakeholders understand why certain instances were selected, how the model evolves over time, and whether any bias or systematic behavior is influencing the learning process.

ALISA follows an iterative learning pipeline where weighted AL, SSL, and data augmentation are applied at each step to expand the training set and improve performance under limited supervision. Embedding explainability into this pipeline would allow us to analyze and interpret the impact of each component throughout the iterations. For example, we could explain which features contributed most to a case being queried by the AL strategy, how pseudo-labels were assigned during SSL, or how augmented examples influence the decision boundary. This not only facilitates human-in-the-loop supervision and possible intervention, but also strengthens the interpretability of ALISA's internal decision-making process, making it more suitable for high-stakes domains such as fraud detection. Future work should focus on developing lightweight, modular XAI components that align with each phase of ALISA's iterative framework.

5.5 Conclusion

In this chapter, we introduced ALISA, a novel framework for learning from limited labeled data, with particularly strong performance in one-class scenarios. ALISA builds upon the most suspicious cases identified by X-ITERADE, and following initial label acquisition, it proceeds through an iterative process that combines weighted AL and enhanced self-training with data augmentation.

Experimental evaluations across multiple imbalanced learning settings demonstrated that ALISA outperforms established baselines in the majority of scenarios. The only exceptions were a few variants of the BAF dataset, where the performance gap between

ALISA and the best baseline was narrower; however, it is important to note that even in these cases, the baseline methods performed poorly overall.

ALISA achieved improvements up to 5–22% in F1 score and 60–80% in precision compared to the best-performing baseline. These gains are especially relevant in real-world applications where labeling is costly and reducing false positive alerts is critical.

ALISA is implemented as a modular and extensible framework, allowing for flexible configuration of sampling strategies, self-training mechanisms, execution modes, and augmentation policies. In our study, we explored 30 distinct configurations, most of which consistently achieved higher average ranks compared to all baseline methods, indicating the overall robustness and adaptability of the approach.

In the next chapter, we provide an overview of the entire thesis, discuss the overarching limitations and future directions of this work, and conclude the study.

Chapter 6

Conclusion

6.1 Overview of the Thesis

This thesis addresses the challenge of fraud detection in scenarios where labeled data is extremely limited or entirely unavailable. To tackle this, we propose a step-by-step methodology that begins in a fully unsupervised setting and gradually incorporates AL and SSL techniques as more information becomes available. The structure of the thesis reflects this progression, guiding the reader from foundational background and comparative analysis to novel contributions that build upon one another.

In Chapter 2, we provide the foundational background on AL and SSL, emphasizing their importance for real-world fraud detection problems where acquiring labeled data is costly and time-consuming. We also extend our scope beyond the financial domain to include malware detection, a domain with similar characteristics, to enhance the generalizability of our analysis. This chapter frames the motivation behind choosing AL and SSL and highlights the gaps in current research that this thesis aims to address.

Chapter 3 builds on this foundation by presenting a detailed comparative study of AL and SSL across fraud and malware detection tasks under varying levels of class imbalance and label scarcity. The results show that while AL generally performs better in terms of F1_Score, SSL often achieves higher Recall, especially in more imbalanced settings, making it a strong candidate when it is critical to minimize false negatives. These insights emphasize that the choice between AL and SSL should be context dependent, guided by the properties of the data set, the imbalance ratios, and the available labeling budgets. The findings of this chapter inform the design decisions in our later contributions.

Having established the theoretical and empirical groundwork, Chapter 4 introduces X-ITERADE, a novel iterative unsupervised anomaly detection framework. X-ITERADE is designed to select the most suspicious cases in a dataset without relying on any labels, making it ideal for extreme imbalance scenarios. It features a flexible budget mechanism that allows experts to control the number of cases selected, as well as an explainability module that profiles suspicious clusters to help investigation. X-ITERADE not only identifies high-risk cases for further labeling but also isolates "safe" instances that do not require review, helping to reduce the size and imbalance of the dataset before any labels are acquired.

Building on the suspicious cases flagged by X-ITERADE, Chapter 5 presents ALISA, a comprehensive and modular learning framework that unifies AL, SSL, and data augmentation into a single pipeline. ALISA begins with a very small labeled set and iteratively improves model performance by querying uncertain instances and generating pseudo-labels using an enhanced self-training module. It introduces a weighted ensemble AL strategy, including one-class querying methods that are effective even when fraud labels are absent, and a robust SSL component that handles data sparsity and imbalance. ALISA achieves significant improvements, particularly in one-class settings with no initial fraud labels,

showing up to 22% F1_Score gains compared to the best performing baseline, and also performs strongly in two-class cases.

This progression, from working with no labels at all to building a targeted, adaptive learning pipeline, reflects the real-world challenges of fraud detection. By adapting to the gradual availability of labeled data, our approaches provide scalable, interpretable, and practical solutions for tackling extreme imbalance and label scarcity.

6.2 Limitations

While this thesis addresses key challenges in fraud detection under label-scarce conditions, several limitations remain. First, although the proposed frameworks, X-ITERADE and ALISA, have been evaluated on highly realistic synthetic datasets, their generalizability to large-scale, real-world datasets from banking institutions is not yet fully validated. Despite our best efforts to obtain access to private data from our industry partner, confidentiality restrictions prevented its use in this work. As a result, certain complexities of real-world fraud detection, such as institution-specific behaviors, evolving fraud strategies, and noise introduced by real operational data, may not be fully captured. We also observed that false positives in the synthetic datasets sometimes followed systematic patterns, which may differ from those seen in practice. These limitations highlight the need to extend future evaluations to a broader range of real-world datasets and domains to provide stronger evidence of robustness and practical applicability.

Second, another limitation of this work lies in the use of K-Means for clustering. While K-Means was efficient and effective in our iterative framework, it relies on strong assumptions: clusters are expected to be spherical, of similar size, and well-separated in terms of Euclidean distance. These assumptions may not always hold in real-world fraud detection,

where fraudulent behaviors can be irregular, overlapping, or highly imbalanced. As a result, K-Means may fail to capture more complex relationships between features, particularly when fraud patterns deviate from simple convex structures.

Third, several components of the system, including the internal classifiers of X-ITERADE, the hyperparameters used in the ALISA framework, and the explainability module, require tuning. These components may not perform optimally across all domains without proper adjustment. In particular, unsupervised hyperparameter tuning for X-ITERADE remains an open challenge and warrants further investigation. Additionally, the current hyperparameter settings used across different parts of the study could benefit from exploration within larger and more diverse search spaces. The language models used in the explainability module could also be fine-tuned to better align with the domain-specific nature of fraud detection.

Fourth, both frameworks assume a stable set of input features and do not yet support dynamic environments where features evolve over time. This limitation makes them less suitable for scenarios involving concept drift, which is a common characteristic of real-world fraud detection systems in production.

Fifth, the study has primarily focused on classical AL and SSL techniques. It is important to note that SSL methods rely on several key assumptions: the smoothness assumption (instances close in feature space are likely to share the same label), the low-density separation assumption (decision boundaries should pass through low-density regions), and the manifold assumption (high-dimensional data lies on a lower-dimensional manifold that can be exploited during learning). While these assumptions enable SSL the use of large amounts of unlabeled data, they may not always hold in complex transactional settings, where fraud patterns can be irregular or evolve over time. More advanced approaches, such as GSSL or representation learning tailored to transactional data, could be explored

in future work to further improve adaptability and performance. These methods may help relax some of the classical SSL assumptions and provide stronger generalization across diverse fraud detection scenarios.

Finally, a key vulnerability of the proposed framework lies in its dependency chain. Since ALISA builds directly on the outputs of X-ITERADE, any errors or adversarial manipulation in the initial stage can propagate through subsequent components. For instance, if X-ITERADE misclassifies fraudulent transactions as safe, these cases may be excluded from further analysis, weakening the detection performance downstream. Conversely, if non-fraudulent transactions are misclassified as suspicious, ALISA's autoencoder may overfit to noisy or irrelevant patterns, reducing its robustness. An attacker could exploit these vulnerabilities by crafting adversarial transactions that mimic the statistical properties of safe cases to evade detection, or by injecting perturbations into suspicious cases to degrade the quality of ALISA's learned representations.

6.3 Future Work

While this thesis introduces effective frameworks for fraud detection under label-scarce conditions, several avenues remain for future exploration and development.

First, extending the proposed methods to real-world datasets from multiple domains, such as banking, insurance, or telecommunications, would help evaluate their robustness in varied operational environments. Although this work relied on highly realistic synthetic datasets, access to genuine institutional data was limited due to confidentiality constraints. As a result, validating the frameworks on real-world datasets remains an important next step. Investigating domain adaptation techniques could further enable the models to generalize across different institutions or regions, where fraud patterns and feature distributions

may differ. Addressing these directions would strengthen the practical applicability of the proposed methods and bridge the gap between research and deployment.

Second, another important direction is to explore clustering algorithms beyond K-Means. Methods such as DBSCAN, spectral clustering, or hierarchical clustering may capture irregular, overlapping, or non-linear feature relationships more effectively. Incorporating these approaches into X-ITERADE and ALISA could strengthen their ability to detect diverse fraud patterns that are not well represented by convex clusters.

Third, fraud behavior often evolves over time, and current frameworks do not yet account for concept drift. A valuable direction would be to adapt X-ITERADE and ALISA to work with streaming data and real-time transaction flows. Incorporating online learning mechanisms would allow the models to continuously update and remain effective as new data arrives.

Fourth, several components of the system currently require manual or heuristic-based tuning. Future work can focus on designing automated hyperparameter selection strategies and unsupervised performance estimation techniques to make both frameworks more self-reliant and scalable.

Fifth, while ALISA currently uses traditional AL and SSL approaches, incorporating more advanced methods, particularly GSSL, could help capture the complex relationships among users and transactions. These methods may reveal structural patterns that enhance detection performance in sparse-label scenarios.

Sixth, to mitigate risks arising from the vulnerabilities of the proposed framework, future work should include adversarial robustness measures, such as adversarial training, ensemble detection mechanisms, or anomaly detectors specifically tuned to identify adversarial perturbations. Cross-validation across multiple independent detection pipelines and the integration of domain knowledge constraints could also reduce the impact of

propagated errors. Strengthening these aspects would enhance the resilience of the framework against adversarial attacks and improve its reliability in real-world deployment.

Finally, the explainability module of X-ITERADE is currently being expanded into an interactive web-based application. This tool allows users to upload datasets and explore the learned decision tree by selecting leaf nodes to access a variety of visualizations and summaries. These include fraud distribution charts, SHAP summary plots, relevant association rules, and LLM-generated explanations for each group of instances. Future work will involve completing this web application and enhancing it with t-SNE visualizations and other interactive elements. This tool aims to support domain experts in interpreting model outputs and communicating insights effectively to non-technical stakeholders.

Together, these future directions aim to improve the adaptability, robustness, security, and usability of the proposed frameworks, opening the way for broader adoption in practical fraud detection systems.

6.4 Final Remarks

In today's data-rich but label-scarce environments, particularly in fraud detection, learning has become a critical challenge. This thesis explores how to address that challenge by progressing from fully unsupervised techniques to adaptive frameworks that combine SSL, and AL. The proposed methods are designed not only to enhance detection performance but also to support transparency and trust through explainability. By offering flexible tools grounded in real-world constraints, this work aims to bridge the gap between research and deployment.

References

[1] Umaru Adamu and Irfan Awan. Ransomware prediction using supervised learning algorithms. In *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 57–63. IEEE, 2019.

[2] Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu. To label or to pseudo label? active learning vs semi-supervised learning for windows malware prediction. *Proceedings of the Canadian Conference on Artificial Intelligence*, 2024.

[3] Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu. Is expert-labeled data worth the cost? exploring active and semi-supervised learning across imbalance scenarios in financial crime detection. In Kamel Adi, Simon Bourdeau, Christel Durand, Valérie Viet Triem Tong, Alina Dulipovici, Yvon Kermarrec, and Joaquin Garcia-Alfaro, editors, *Foundations and Practice of Security*, pages 174–186, Cham, 2025. Springer Nature Switzerland.

[4] Bahar Emami Afshar, Paula Branco, Tolga Kurt, Utku Gorkem Ketenci, and Hikmet Mazmanoglu. ITERADE - ITERative Anomaly Detection Ensemble for Credit Card Fraud Detection. In Dino Pedreschi, Anna Monreale, Riccardo Guidotti, Roberto Pel-

lungrini, and Francesca Naretto, editors, *Discovery Science*, pages 115–133, Cham, 2025. Springer Nature Switzerland.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 487–499. Morgan Kaufmann, 1994.

[6] Erik Altman, Apoorva Nitsure, and Youssef Mroueh. Credit card transactions fraud detection and other analyses, 2019. Accessed: 2024-06-20.

[7] Domingos Araujo, Joaquim Silva, Patricia Leite, Oscar Ribeiro, and Paulo Teixeira. Predict malware using machine learning. In *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–6, 2023.

[8] Ricardo Barata, Miguel Leite, Ricardo Pacheco, Marco OP Sampaio, João Tiago Ascensão, and Pedro Bizarro. Active learning for imbalanced data under cold start. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

[9] Ashub bin Asad, Raiyan Mansur, Safir Zawad, Nahian Evan, and Muhammad Iqbal Hossain. Analysis of malware prediction based on infection rate using machine learning techniques. In *2020 IEEE region 10 symposium (TENSYMP)*, pages 706–709. IEEE, 2020.

[10] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[11] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[12] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[13] Eyad Abdel Latif Marazqah Btoush, Xujuan Zhou, Raj Gururajan, Ka Ching Chan, Rohan Genrich, and Prema Sankaran. A systematic review of literature on credit card cyber fraud detection using machine and deep learning. *PeerJ Computer Science*, 9:e1278, 2023. Published April 17, 2023.

[14] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, and Gianluca Bontempi. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization. *International Journal of Data Science and Analytics*, 5:285–300, 2018.

[15] P Caroline Cynthia and S Thomas George. An outlier detection approach on credit card fraud detection using machine learning: a comparative analysis on supervised and unsupervised learning. In *Intelligence in Big Data Technologies—Beyond the Hype: Proceedings of ICBDCC 2019*, pages 125–135. Springer, 2021.

[16] Charitos Charitou, Artur d'Avila Garcez, and Simo Dragicevic. Semi-supervised gans for fraud detection. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[17] Chin-Wei Chen, Ching-Hung Su, Kun-Wei Lee, and Ping-Hao Bair. Malware family classification using active learning by learning. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, pages 590–595. IEEE, 2020.

[18] Li Chen, Mingwei Zhang, Chih-yuan Yang, and Ravi Sahita. Semi-supervised classification for dynamic android malware detection. *arXiv preprint arXiv:1704.05948*, 2017.

[19] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[20] Yizheng Chen, Zhoujie Ding, and David Wagner. Continuous learning for android malware detection. *arXiv preprint arXiv:2302.04332*, 2023.

[21] Dawei Cheng, Yujia Ye, Sheng Xiang, Zhenwei Ma, Ying Zhang, and Changjun Jiang. Anti-money laundering by group-aware deep graph learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12444–12457, 2023.

[22] Competition Bureau Canada. Fraud prevention month to focus on impersonation fraud, one of the fastest growing forms of fraud, February 2025. Accessed: 2025-06-18.

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[24] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[25] Cropink. ecommerce fraud statistics: The latest data and trends, 2024. Accessed: 2025-06-18.

[26] Limeng Cui, Xianfeng Tang, Sumeet Katariya, Nikhil Rao, Pallav Agrawal, Karthik Subbian, and Dongwon Lee. Allie: Active learning on large-scale imbalanced graphs. In *Proceedings of the ACM Web Conference 2022*, pages 690–698, 2022.

[27] DataVisor. Forecasting fraud: Insights into 2025 and beyond, 2025. Accessed: 2025-06-18.

[28] Xinwei Deng, V Roshan Joseph, Agus Sudjianto, and CF Jeff Wu. Active learning through sequential design, with applications to detection of money laundering. *Journal of the American Statistical Association*, 104(487):969–981, 2009.

[29] Yu Ding, XiaoYu Zhang, BinBin Li, Jian Xing, Qian Qiang, ZiSen Qi, MengHan Guo, SiYu Jia, and HaiPing Wang. Malware classification based on semi-supervised learning. In *International Conference on Science of Cyber Security*, pages 287–301. Springer, 2022.

[30] Gregory Druck, Burr Settles, and Andrew McCallum. Active learning by labeling features. In *Proceedings of the 2009 conference on Empirical methods in natural language processing*, pages 81–90, 2009.

[31] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[32] Federal Trade Commission. Consumer sentinel network data book 2024. Technical report, Federal Trade Commission, Washington, DC, March 2025. Published March 2025; data covers reports received during calendar year 2024; sourced from the Federal Trade Commission's Consumer Sentinel Network.

[33] Federal Trade Commission. New ftc data show big jump in reported losses from fraud: $12.5 billion in 2024. Press release, Federal Trade Commission, Washington, DC, March 2025. Consumers reported losing more than $12.5billion to fraud in 2024—a 25% increase from 2023.

[34] Fineksus. How fraud and anti-money laundering differ and relate, 2023. Accessed: 2025-06-18.

[35] Rudolf Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233, 1948.

[36] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and information systems*, 35:249–283, 2013.

[37] Tan Gao, Lan Zhao, Xudong Li, and Wen Chen. Malware detection based on semi-supervised learning with malware visualization. *Math. Biosci. Eng*, 18(5):5995–6011, 2021.

[38] Xianwei Gao, Changzhen Hu, Chun Shan, Baoxu Liu, Zequn Niu, and Hui Xie. Malware classification for the cloud via semi-supervised transfer learning. *Journal of Information Security and Applications*, 55:102661, 2020.

[39] GlobeNewswire. Payment card fraud losses approach $34 billion, January 2025. Accessed June 2025.

[40] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[41] Alejandro Guerra-Manzanares and Hayretdin Bahsi. On the application of active learning to handle data evolution in android malware detection. In *International Conference on Digital Forensics and Cyber Crime*, pages 256–273. Springer, 2022.

[42] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[43] Addison Howard, Bernadette Bouchon-Meunier, IEEE CIS, inversion, Lynn@Vesta John Lei, Marcus2010, and Prof. Hussein Abbass. Ieee-cis fraud detection, 2019.

[44] Addison Howard, Ben Hope, Brendan Saltaformaggio, Eric Avena, Mansour Ahmadi, Matthew Duncan, $n_30, RobMcCann, and Will Cukierski. Microsoft malware prediction$, 2018.

[45] Gert Jacobusse and Cor Veenman. On selection bias with imbalanced classes. In *Discovery Science: 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings*, pages 325–340. Springer, 2016.

[46] Sérgio Jesus, José Pombal, Duarte Alves, André Cruz, Pedro Saleiro, Rita P. Ribeiro, João Gama, and Pedro Bizarro. Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation. *Advances in Neural Information Processing Systems*, 2022.

[47] Shanshan Jiang, Ruiting Dong, Jie Wang, and Min Xia. Credit card fraud detection based on unsupervised attentional anomaly detection network. *Systems*, 11(6):305, 2023.

[48] Yinhan Jiang et al. Introducing mistral 7b. Mistral AI, 2023. https://mistral.ai/news/introducing-mistral-7b.

[49] Seho Kee, Enrique Del Castillo, and George Runger. Query-by-committee improvement with diversity and density in batch active learning. *Information Sciences*, 454:401–418, 2018.

[50] Robert KL Kennedy, Zahra Salekshahrezaee, and Taghi M Khoshgoftaar. A novel approach for unsupervised learning of highly-imbalanced data. In *2022 IEEE 4th International Conference on Cognitive Machine Intelligence (CogMI)*, pages 52–58. IEEE, 2022.

[51] Terry Kevin Koo and Mae Yuen Li. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine*, 15(2):155–163, 2016.

[52] Klaus Krippendorff. *Content analysis: An introduction to its methodology*. Sage, 2004.

[53] Danilo Labanca, Luca Primerano, Marcus Markland-Montgomery, Mario Polino, Michele Carminati, and Stefano Zanero. Amaretto: an active learning framework for money laundering detection. *IEEE Access*, 10:41720–41739, 2022.

[54] Bertrand Lebichot, Fabian Braun, Olivier Caelen, and Marco Saerens. A graph-based, semi-supervised, credit card fraud detection system. In *International Workshop on Complex Networks and their Applications*, pages 721–733. Springer, 2016.

[55] Maria R Lepoivre, Chloé O Avanzini, Guillaume Bignon, Loïc Legendre, and Aristide K Piwele. Credit card fraud detection with unsupervised algorithms. *Journal of advances in information technology*, 7(1), 2016.

[56] David D Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA, 1995.

[57] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[58] Changsheng Li, Handong Ma, Ye Yuan, Guoren Wang, and Dong Xu. Structure guided deep neural network for unsupervised active learning. *IEEE Transactions on Image Processing*, 31:2767–2781, 2022.

[59] Dapeng Li, Jiangpei Zhu, Haitao Zhao, Xiaoming Wang, and Rui Jiang. Svm-based online learning for interference-aware multi-cell mmwave vehicular communications. *IET Communications*, 15(8):1015–1027, 2021.

[60] Yuanzhang Li, Xinxin Wang, Zhiwei Shi, Ruyun Zhang, Jingfeng Xue, and Zhi Wang. Boosting training for pdf malware classifier via active learning. *International journal of intelligent systems*, 37(4):2803–2821, 2022.

[61] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[62] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

[63] Mingxuan Lu, Zhichao Han, Susie Xi Rao, Zitao Zhang, Yang Zhao, Yinan Shan, Ramesh Raghunathan, Ce Zhang, and Jiawei Jiang. Bright-graph neural networks in real-time fraud detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3342–3351, 2022.

[64] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[65] Samaneh Mahdavifar, Andi Fitriah Abdul Kadir, Rasool Fatemi, Dima Alhadidi, and Ali A Ghorbani. Dynamic android malware category classification using semi-supervised deep learning. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on*

Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pages 515–522. IEEE, 2020.

[66] Philip M. McCarthy. *An assessment of the range and usefulness of lexical diversity measures and the potential of the Measure of Textual Lexical Diversity (MTLD)*. Phd dissertation, The University of Memphis, 2005.

[67] Leland McInnes, John Healy, Steve Astels, et al. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.

[68] Ali Muzaffar, Hani Ragab Hassen, Hind Zantout, and Michael A Lones. Actdroid: An active learning framework for android malware detection. *arXiv preprint arXiv:2401.16982*, 2024.

[69] Huang Na and He Jingsha. Malware detection using an improved active learning approach. In *2021 7th International Conference on Computer and Communications (ICCC)*, pages 1309–1313. IEEE, 2021.

[70] Nir Nissim, Robert Moskovitch, Lior Rokach, and Yuval Elovici. Novel active learning methods for enhanced pc malware detection in windows os. *Expert Systems with Applications*, 41(13):5843–5857, 2014.

[71] Xuetong Niu, Li Wang, and Xulei Yang. A comparison study of credit card fraud detection: Supervised versus unsupervised. *arXiv preprint arXiv:1904.10604*, 2019.

[72] OpenAI. Gpt-4o: Openai's new multimodal model. OpenAI, 2024. https://openai.com/index/gpt-4o.

[73] A. Panconesi, Marian W. Cukierski, and WWW BIG Cup Committee. Microsoft malware classification challenge (big 2015), 2015.

[74] Payments Dive. Payment fraud losses are climbing. will prevention keep pace?, 2024. Published April 10, 2024. Accessed June 2025.

[75] Pramuditha Perera, Poojan Oza, and Vishal M Patel. One-class classification: A survey. *arXiv preprint arXiv:2101.03064*, 2021.

[76] Qijing Qiao, Ruitao Feng, Sen Chen, Fei Zhang, and Xiaohong Li. Multi-label classification for android malware based on active learning. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[77] Arun Kumar Rai and Rajendra Kumar Dwivedi. Fraud detection in credit card data using unsupervised machine learning based scheme. In *2020 international conference on electronics and sustainable communication systems (ICESC)*, pages 421–426. IEEE, 2020.

[78] Mahdi Rezapour. Anomaly detection using unsupervised methods: credit card fraud case study. *International Journal of Advanced Computer Science and Applications*, 10(11), 2019.

[79] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[80] Addisson Salazar, Gonzalo Safont, and Luis Vergara. Semi-supervised learning for imbalanced classification of credit card transaction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2018.

[81] Igor Santos, Javier Nieves, and Pablo G Bringas. Semi-supervised learning for unknown malware detection. In *International Symposium on Distributed Computing and Artificial Intelligence*, pages 415–422. Springer, 2011.

[82] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[83] Burr Settles. Active learning literature survey. 2009.

[84] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. *Advances in Neural Information Processing Systems*, 20, 2007.

[85] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[86] Patrick E Shrout and Joseph L Fleiss. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, 86(2):420–428, 1979.

[87] Xuning Tang, Yihua Shi Astle, and Craig Freeman. Deep anomaly detection with ensemble-based active learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1663–1670, 2020.

[88] The Nilson Report. Card fraud losses worldwide, 2022. Accessed June 2025.

[89] The Nilson Report. Card fraud losses worldwide in 2023, 2024. Accessed June 2025.

[90] Thomson Reuters. 2025 predictions: The interplay of fraud and ai, 2024. Accessed: 2025-06-18.

[91] Tookitaki. 50 shocking statistics about money laundering and cryptocurrency, August 2020. Accessed: 2025-06-18.

[92] Hugo Touvron, Thibaut Lavril, Alexandre Fuernstahl, et al. Llama 3: Open foundation and instruction models. Meta AI, 2024. https://ai.meta.com/llama.

[93] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Timothy Dettmers, Myle Ott, Francisco Guzmán, Angela Fan, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama 2: Open foundation and fine-tuned chat models, 2023.

[94] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42:245–284, 2015.

[95] United Nations Office on Drugs and Crime. Money-laundering and globalization, n.d. Accessed: 2025-06-18.

[96] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020.

[97] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 598–607. IEEE, 2019.

[98] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[99] Sheng Xiang, Mingzhi Zhu, Dawei Cheng, Enxia Li, Ruihui Zhao, Yi Ouyang, Ling Chen, and Yefeng Zheng. Semi-supervised credit card fraud detection via attribute-driven graph

representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14557–14565, 2023.

[100] Jianjun Xie and Tao Xiong. Stochastic semi-supervised learning on partially labeled imbalanced data. In *Active Learning and Experimental Design Workshop in Conjunction with AISTATS 2010*, pages 85–98. JMLR Workshop and Conference Proceedings, 2011.

[101] Yazhou Yang and Marco Loog. Active learning using uncertainty information. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2646–2651. IEEE, 2016.

[102] Ya-Lin Zhang, Yi-Xuan Sun, Fangfang Fan, Meng Li, Yeyu Zhao, Wei Wang, Longfei Li, Jun Zhou, and Jinghua Feng. A framework for detecting frauds from extremely few labels. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 1124–1127, 2023.

[103] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

[104] Tianqing Zhu. An outlier detection model based on cross datasets comparison for financial surveillance. In *2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*, pages 601–604. IEEE, 2006.

[105] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018.

APPENDICES

Appendix A

Accuracy Results For Malware Detection, Active and Semi-Supervised Learning Comparison

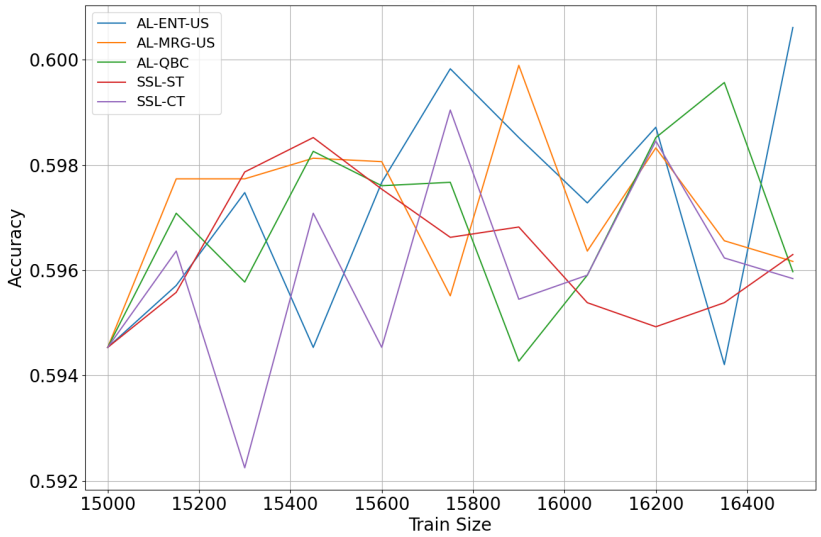


Figure A.1: Accuracy results for RF (150 queries; 10 iterations).

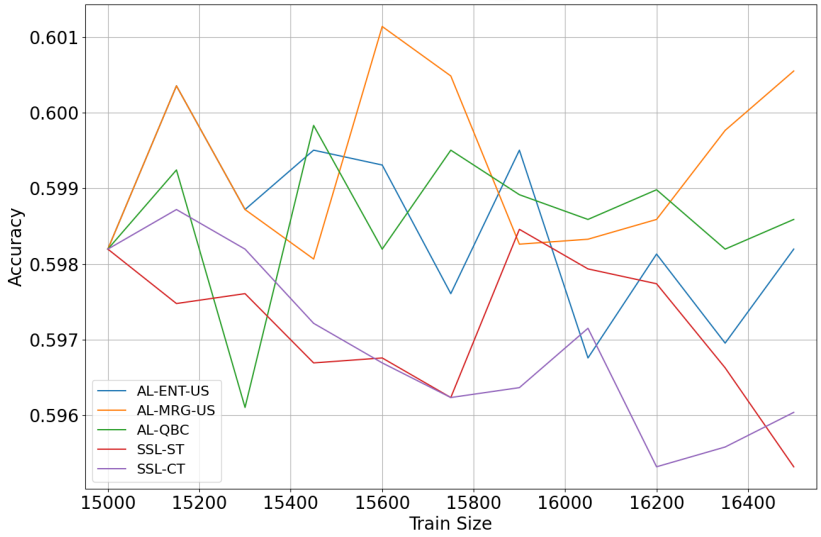


Figure A.2: Accuracy results for XGBoost (150 queries; 10 iterations).

Appendix B

Example Prompts for X-ITERADE Explainability Module

Table B.1: Example prompt used for the explainability module with CCT dataset (SV2-1)

Example Prompt for Explainability Module (CCT Dataset, SV2-1)
Task Description: You are an expert in machine learning, specifically in the area of credit card fraud detection using SHAP values, association rules, and distribution analysis. Your task is to summarize the characteristics and fraud risk of groups based on the provided data. The goal is to reason and generate insights, rather than simply matching patterns. Follow the instructions carefully to avoid repeating input data and to ensure clarity in your output.
Feature Explanation: - `Has Chip`: Indicates whether the card has a chip (YES or NO). - `Total Cards`: The total number of credit cards a customer has.

- `Zip`: The zip code of the customer.

- `Total Transactions`: The total number of transactions made by the customer.

- `Total Debt`: The total amount of debt the customer has.

- `Card Brand`: The brand of the credit card (e.g., Mastercard, Amex).

- `Use Chip`: Type of transaction has 3 possible values: Chip Transaction, Online Transaction, and Swipe Transaction.

- `Longitude`: Customer's geographic longitude

- `Expires`: Card's expiration date

Example 1:

Key SHAP Values: `Total Cards` (high impact), `Zip` (moderate impact)

Association Rules:

`Has Chip` = YES $\Rightarrow$ `Card Brand` = Mastercard;

`Total Cards` $\in$ (3.999, 5.0] $\Rightarrow$ `Has Chip` = YES

Distribution Info: 4881 non-fraud, 1 fraud (highly suspicious)

Additional Context: `Has Chip` = YES, `Zip` $\leq$ 79970.50, `Total Cards` $>$ 3.50

LLM Output: Individuals with 4 or 5 chipped credit cards, primarily using Mastercard, and zip code $\leq$ 79970.50. Low overall fraud rate, but risk factors from SHAP values and rules justify further investigation.

Example 2:

Key SHAP Values: `Has Chip` (high impact), `Card Brand` (low impact)

Association Rules:

`Has Chip` = NO ⇒ `Card Brand` = Mastercard

Distribution Info: 1408 non-fraud, 0 fraud

Additional Context: `Has Chip` = NO, `Card Brand` ≠ Amex, `Total Debt` ≤ 240871.50

LLM Output: Users with non-chipped Mastercards, no Amex, and low debt. No fraud observed, with a 100% non-fraud rate. Minimal risk.

Example 3:

Key SHAP Values: `Total Cards` (moderate), `Zip` (moderate)

Association Rules:

`Has Chip` = YES ⇒ `Card Brand` = Mastercard;

`Has Chip` = YES ⇒ `Total Cards` ∈ (2.0, 3.0]

Distribution Info: 4630 non-fraud, 0 fraud

Additional Context: `Has Chip` = YES, `Zip` ≤ 79970.50, `Total Cards` ≤ 3.50

LLM Output: Chipped cards, low zip, and ≤ 3.5 cards indicate no suspicion. Not flagged for review.

Test Example:

Key SHAP Values: `Zip` (high), `Total Transactions` (moderate)

Association Rules: `Has Chip` = YES ⇒ `Card Brand` = Mastercard

Distribution Info: 3057 non-fraud, 29 fraud (highly suspicious)

Additional Context: `Has Chip` = YES, `Zip` > 79970.50, `Total Transactions` ≤ 169.00

LLM Task: Generate the desired output for Test Example:

Table B.2: Example prompt used for the explainability module with BAF dataset (SV2-1)

Example Prompt for Explainability Module (BAF Dataset, SV2-1)

Task Description: † See Task Description in Table B.1

Feature Explanation:

- `employment_status`: Employment status of the applicant. 7 anonymized values: {0: CA, 1: CB, 2: CC, 3: CD, 4: CE, 5: CF, 6: CG}.

- `intended_balcon_amount`: Initial transferred amount for application. Range: [−1, 108].

- `phone_mobile_valid`: Validity of the provided mobile phone (0 = NO, 1 = YES).

- `bank_branch_count_8w`: Number of applications at the selected bank branch in the last 8 weeks. Range: [0, 2521].

- `source`: Online source of the application (either `INTERNET` or `APP`).

- `current_address_months_count`: Months at current address. Range: [−1, 406], where −1 indicates missing.

- `Month`: Month where the application was made. Ranges between [0, 7].

- `Proposed_credit_limit`: Applicant's proposed credit limit. Ranges between [200, 2000].

- `Device_distinct_emails_8w`: Number of distinct emails in banking website from the used device in last 8 weeks. Ranges between [0, 3].

- `Has_other_cards`: If applicant has other cards from the same banking company.

- Customer_age: Applicant's age in bins per decade (e.g., 20-29 is represented as 20).

Example 1:

Key SHAP Values: phone_mobile_valid (high impact), employment_status (moderate impact)

Association Rules:

source=INTERNET, phone_mobile_valid=0 $\Rightarrow$ Cluster=9

source=INTERNET, phone_mobile_valid=0 $\Rightarrow$ employment_status=CA, Cluster=9

Distribution Info: 174 non-fraud, 1 fraud (highly suspicious)

Additional Context: employment_status ≤ 2.50, intended_balcon_amount ≤ 15.62, phone_mobile_valid = NO

LLM Output: Individuals without a valid phone number who used the INTERNET browser for their application. Their employment status is CA and their initial transferred amount was less than 15.62. The high fraud rate and the SHAP importance of invalid phone numbers warrant further review. Flagged as suspicious.

Example 2:

Key SHAP Values: employment_status (moderate impact), intended_balcon_amount (moderate impact)

Association Rules:

phone_mobile_valid=1 $\Rightarrow$ employment_status=CA

Distribution Info: 1796 non-fraud, 15 fraud (highly suspicious)

Additional Context: employment_status ≤ 2.50, intended_balcon_amount ≤ 15.62, phone_mobile_valid = YES

LLM Output: Individuals with a valid phone number whose employment status is CA. Their initial transferred amount was less than 15.62. The high fraud rate warrants further investigation. Flagged for review.

Example 3:

Key SHAP Values: `intended_balcon_amount` (high impact), `employment_status` (moderate impact)

Association Rules:

`phone_mobile_valid=1`, `employment_status=CA` $\Rightarrow$ `Cluster=8`

Distribution Info: 274 non-fraud, 0 fraud

Additional Context: `employment_status` ≤ 2.50, `intended_balcon_amount` > 15.62, `bank_branch_count_8w` ≤ 923.50

LLM Output: Individuals with valid phone numbers whose employment status is CA. Their initial transferred amount was more than 15.62, and the number of applications made in their bank branch in the last 8 weeks was below 923.5. No fraud observed. Minimal risk.

Test Example:

Key SHAP Values: `employment_status` (high impact), `source` (low impact)

Association Rules:

`Cluster=6` $\Rightarrow$ `source=INTERNET`, `phone_mobile_valid=1`, `employment_status=CF`

Distribution Info: 401 non-fraud, 1 fraud (highly suspicious)

Additional Context: `employment_status` > 2.50, `source=INTERNET`, `current_address_months_count` ≤ 235.50

LLM Task: Generate the desired output for the Test Example.

Appendix C

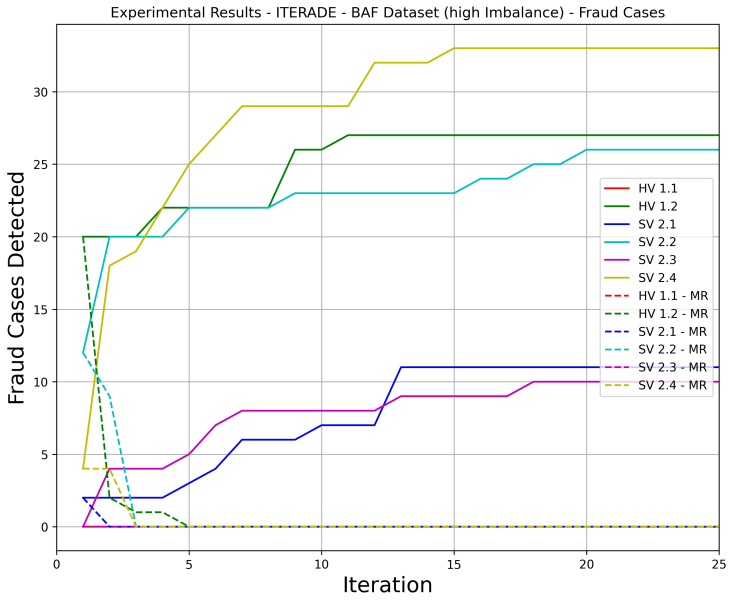
X-ITERADE Complete Experimental Results

Dataset	Budget	Algorithm	Performance		Improvement	Anomalies	
			Minority Class				
			Precision	F1 Score		TP	Anomalies Detected
BAF - Original	<=600	LOF	0	0	0	0	25
		HDBSCAN	0.01266	0.025	1.149	2	158
		Isolation Forest	0.01538	0.03029	1.396	8	520
		ECOD	0.02885	0.05608	2.618	15	520
		ITERADE - HV 1.1	0.01754	0.03448	1.592	1	57
		ITERADE - HV 1.2 - MR	0.06034	0.11381	**5.475**	7	116
		ITERADE - SV 2.4 - MR	0.02373	0.04636	2.153	14	590
	>600 & <=4000	ITERADE - HV 1.1	0.01856	0.03644	1.684	18	970
		ITERADE - SV 2.4	0.02441	0.04766	**2.215**	33	1352
		ITERADE - HV 1.2 - MR	0.01392	0.02746	1.263	46	3305
	>= 4000	DBSCAN	0.01102	0.0218	1	573	51998
		ITERADE - HV 1.2	0.01316	0.02598	**1.194**	285	21657
		ITERADE - SV 2.4	0.01192	0.02356	1.082	107	8978
BAF - Low	<=600	LOF	0.22222	0.36363	1.058	6	27
		HDBSCAN	0.31797	0.48251	1.514	69	217
		Isolation Forest	0.26923	0.42424	1.282	140	520
		ECOD	0.32885	0.49494	1.566	171	520
		ITERADE - HV 1.1 - MR	0.4	0.57143	1.905	2	5
		ITERADE - HV 1.2 - MR	0.34444	0.51239	1.64	31	90
		ITERADE - SV 2.3 - MR	0.52941	0.69231	**2.521**	27	51
	>600 & <=4000	ITERADE - HV 1.2 - MR	0.27229	0.42803	1.297	910	3342
		ITERADE - SV 2.2	0.34578	0.51387	**1.647**	463	1339
	>= 4000	DBSCAN	0.21001	0.34712	1	10920	51997
		ITERADE - SV 2.2	0.23119	0.37556	**1.101**	1423	6155
		ITERADE - HV 1.2	0.21916	0.35953	1.044	1549	7068
BAF - Med	<=600	HDBSCAN	0	0	0	0	0
		LOF	0.16667	0.28572	21.645	6	36
		Isolation Forest	0.13654	0.24027	17.732	71	520
		ECOD	0.18462	0.31169	2.398	96	520
		ITERADE - HV 1.1	0.10484	0.18978	13.616	13	124
		ITERADE - SV 2.3	0.10465	0.18947	13.591	9	86
		ITERADE - SV 2.4	0.1908	0.32046	**24.779**	83	435
	>600 & <=4000	ITERADE - HV 1.2	0.11341	0.20372	14.729	345	3042
		ITERADE - SV 2.2	0.12678	0.22503	**16.465**	98	773
	>= 4000	DBSCAN	0.077	0.14299	10	4004	51998
		ITERADE - HV 1.2	0.08178	0.1512	**10.621**	553	6762
		ITERADE - SV 2.4	0.07074	0.13213	9.187	990	13995
BAF - High	<=2000	HDBSCAN	0	0	0	0	150
		LOF	0	0	0	0	34
		Isolation Forest	0	0	0	0	520
		ECOD	0.00192	0.00383	**2.494**	1	520
		ITERADE - SV 2.1	0.00161	0.00321	2.091	2	1246
	>= 2000	DBSCAN	0.00077	0.00154	1	40	51992
		ITERADE - SV 2.1	0.00088	0.00176	1.143	6	6852
		ITERADE - SV 2.1	0.00111	0.00222	**1.442**	11	9895
		ITERADE - SV 2.4	0.00082	0.00164	1.065	29	35566
		ITERADE - SV 2.2	0.00081	0.00162	1.052	20	24816

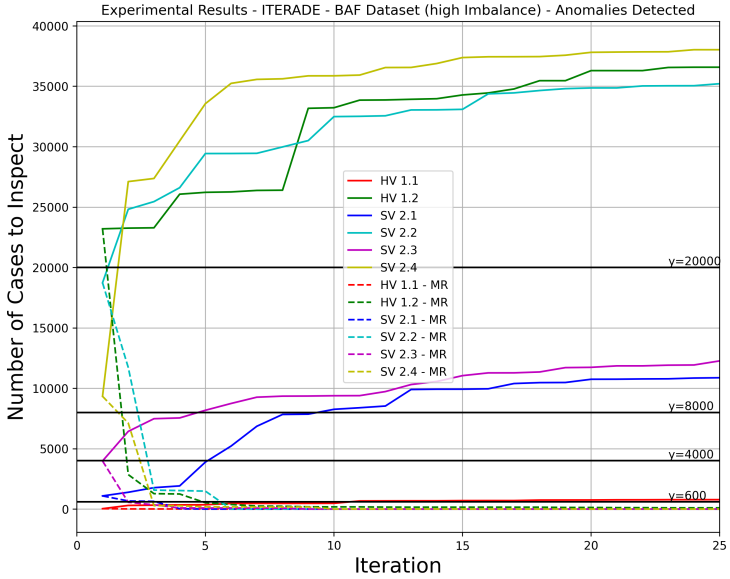
Figure C.1: Experimental results for BAF datasets across different imbalance ratios, with best performance reported for each budget bucket.

Dataset	Budget	Algorithm	Performance		Improvement	Anomalies	
			Minority Class				
			Precision	F1 Score		TP	Anomalies Detected
CCT - Original	<=600	Isolation Forest	0.00198	0.00395	1.414	1	506
		ECOD	0.00988	0.01957	7.057	5	506
		ITERADE - SV 2.4 - MR	0.04348	0.08334	**31.057**	1	23
		ITERADE - SV 2.4 - MR	0.02	0.03922	14.286	5	250
		ITERADE - SV 2.3 - MR	0.0105	0.02078	7.5	4	381
		ITERADE - HV 1.1	0.00326	0.0065	2.329	1	307
	>600 & <=4000	HDBSCAN	0.0011	0.0022	0.786	4	3647
		ITERADE - SV 2.1 - MR	0.01151	0.02276	**8.221**	9	782
		ITERADE - HV 1.2 -MR	0.00429	0.00854	3.064	4	933
		ITERADE - SV 2.3	0.00919	0.01821	6.564	15	1633
	>4000 & <=8000	LOF	0.00148	0.00296	1.057	11	7457
		ITERADE - SV 2.1	0.00577	0.01147	**4.121**	24	4162
		ITERADE - SV 2.3	0.00386	0.00769	2.757	28	7247
	>= 8000	DBSCAN	0.0026	0.00519	1.857	64	24652
		ITERADE - SV 2.4	0.00406	0.00809	2.9	38	9350
		ITERADE - HV 1.2	0.00427	0.0085	**3.05**	48	11252
CCT - Low	<=600	Isolation Forest	0.14822	0.25817	0.706	75	506
		ECOD	0.24111	0.38854	1.148	122	506
		ITERADE - HV 1.1	0.21212	0.35	1.01	21	99
		ITERADE - SV 2.1 - MR	0.4375	0.6087	**2.083**	98	224
		ITERADE - SV 2.4 - MR	0.3848	0.55575	1.832	162	421
	>600 & <=4000	HDBSCAN	0.13736	0.24154	0.654	500	3640
		ITERADE - SV 2.3 - MR	0.30917	0.47231	**1.472**	273	883
		ITERADE - SV 2.2 - MR	0.28002	0.43752	1.333	1082	3864
		ITERADE - SV 2.4 - MR	0.27867	0.43587	1.327	401	1439
	>4000 & <=8000	LOF	0.17967	0.30461	0.856	1414	7870
		ITERADE - SV 2.1	0.33531	0.50222	**1.597**	1638	4885
		ITERADE - SV 2.2 - MR	0.30226	0.46421	1.439	1848	6114
	>= 8000	DBSCAN	0.28431	0.44274	1.354	8378	29468
		ITERADE - SV 2.1	0.30074	0.46241	**1.432**	3676	12223
		ITERADE - HV 1.2	0.25722	0.40919	1.225	3012	11710
		ITERADE - SV 2.4	0.26204	0.41526	1.248	6030	23012
CCT - Med	<=600	Isolation Forest	0.10672	0.19286	1.386	54	506
		ECOD	0.16798	0.28764	2.182	85	506
		ITERADE - HV 1.2 - MR	0.22936	0.37314	2.979	75	327
		ITERADE - SV 2.1 - MR	0.44444	0.61538	**5.772**	4	9
		ITERADE - SV 2.3 - MR	0.2	0.33333	2.597	19	95
	>600 & <=4000	HDBSCAN	0.0508	0.09669	0.66	206	4055
		ITERADE - HV 1.2 - MR	0.13633	0.23995	1.771	164	1203
		ITERADE - SV 2.1	0.14341	0.25085	**1.862**	443	3089
	>4000 & <=8000	LOF	0.07498	0.1395	0.974	574	7655
		ITERADE - SV 2.3	0.14302	0.25025	**1.857**	637	4454
		ITERADE - SV 2.1	0.1337	0.23586	1.736	970	7255
	>= 8000	DBSCAN	0.12853	0.22778	1.669	3455	26880
		ITERADE - SV 2.1	0.13048	0.23084	**1.695**	1054	8078
		ITERADE - SV 2.2	0.11158	0.20076	1.449	1656	14841
CCT - High	<=600	Isolation Forest	0.00395	0.00787	5.13	2	506
		ECOD	0.00593	0.01179	7.701	3	506
		ITERADE - HV 1.1	0.02	0.03922	**25.974**	1	50
		ITERADE - HV 1.2 - MR	0.00586	0.01165	7.61	3	512
		ITERADE - SV 2.4 - MR	0.01136	0.02246	14.753	2	176
	>600 & <=4000	HDBSCAN	0.00059	0.00118	0.766	2	3362
		ITERADE - SV 2.1	0.00488	0.00971	6.338	12	2460
		ITERADE - SV 2.2 - MR	0.00461	0.00918	5.987	14	3036
		ITERADE - SV 2.3 - MR	0.00603	0.01199	**7.831**	7	1161
	>4000 & <=8000	LOF	0.00118	0.00236	1.532	9	7610
		ITERADE - SV 2.2 - MR	0.00323	0.00644	**4.195**	15	4651
		ITERADE - SV 2.1	0.00287	0.00572	3.727	16	5571
	>= 8000	DBSCAN	0.00141	0.00282	1.831	35	24756
		ITERADE - HV 1.2	0.00257	0.00513	**3.338**	24	9351
		ITERADE - SV 2.2	0.00159	0.00317	2.065	31	19469

Figure C.2: Experimental results for CCT datasets across different imbalance ratios, with best performance reported for each budget bucket.

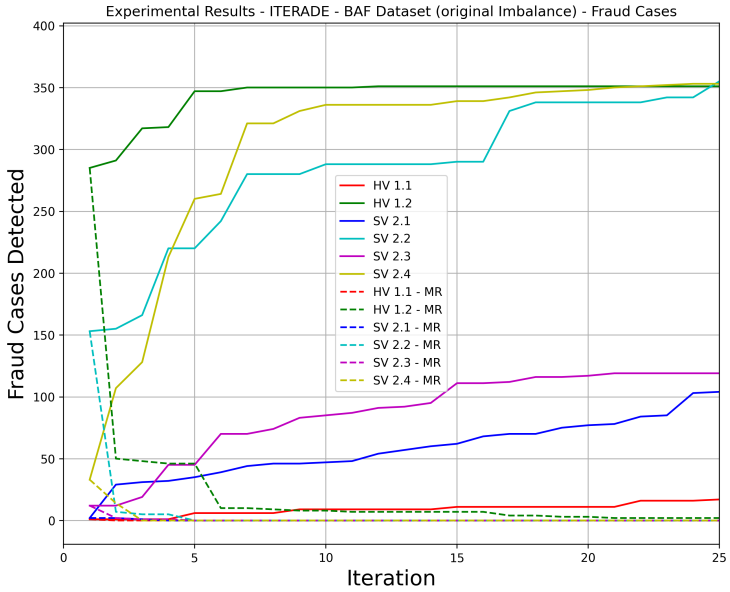


(a) Number of fraud cases and most repeated fraud cases detected per iteration for the BAF high imbalance dataset.

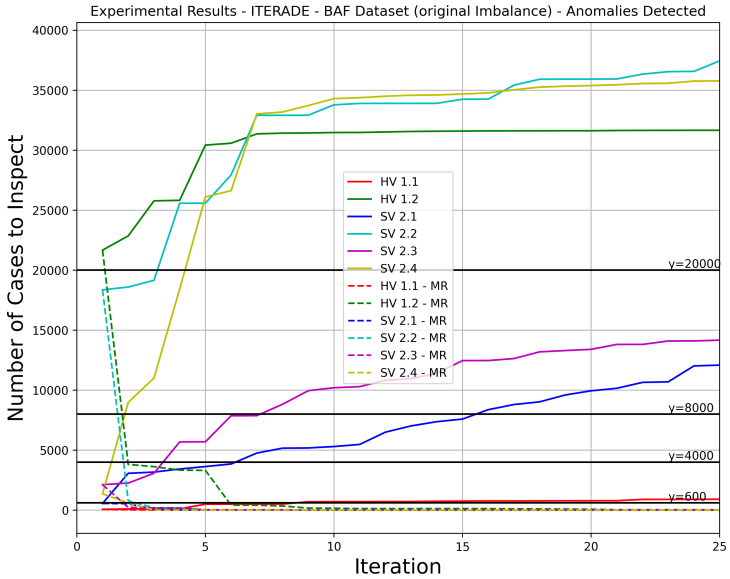


(b) Number of anomalies and most repeated anomalies detected per iteration for the BAF high imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range.

Figure C.3: X-ITERADE results across all iterations for BAF high imbalance dataset.

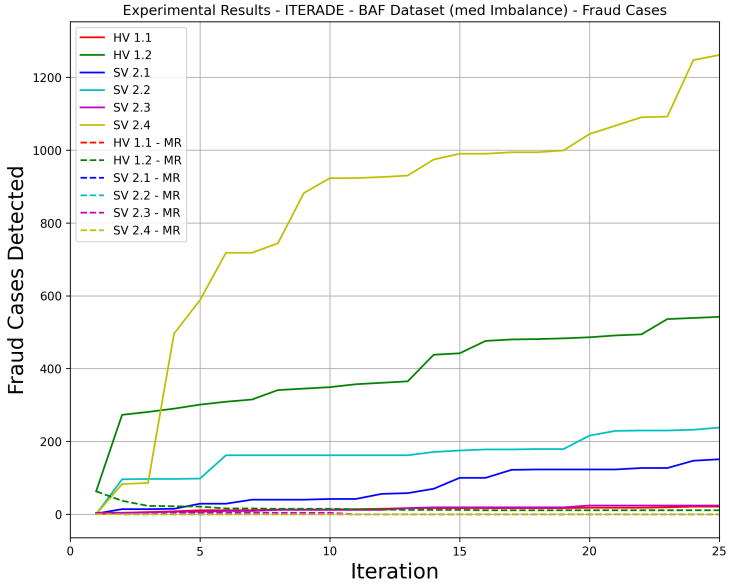


(a) Number of fraud cases and most repeated fraud cases detected per iteration for the BAF original imbalance dataset.

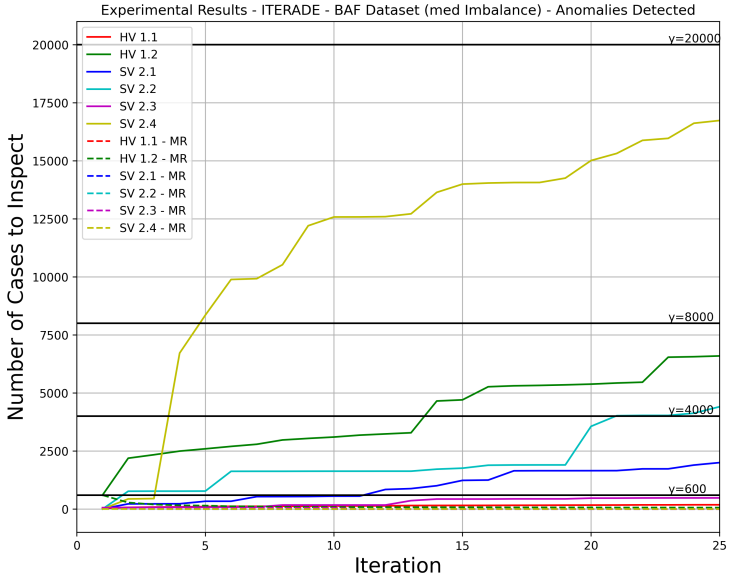


(b) Number of anomalies and most repeated anomalies detected per iteration for the BAF original imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range.

Figure C.4: X-ITERADE results across all iterations for BAF original imbalance dataset.

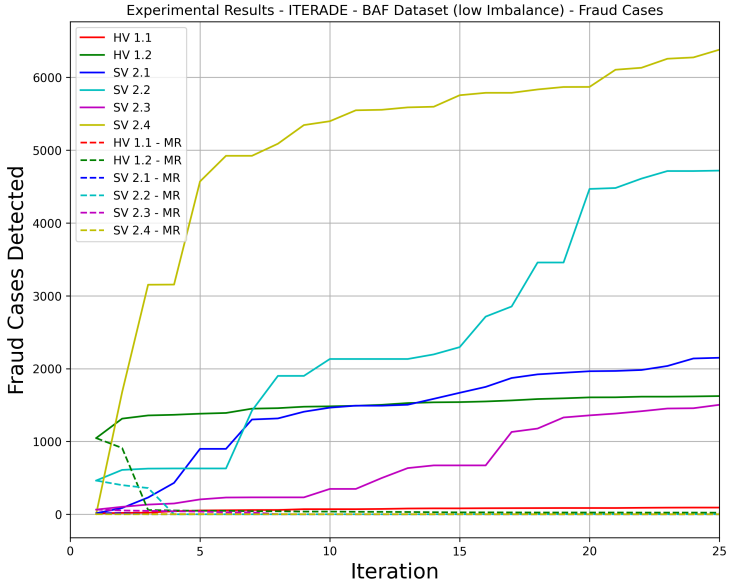


(a) Number of fraud cases and most repeated fraud cases detected per iteration for the BAF med imbalance dataset.

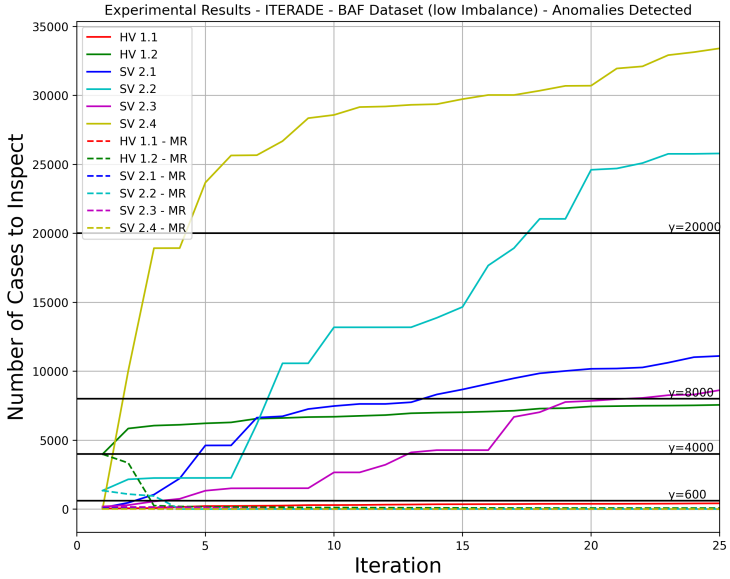


(b) Number of anomalies and most repeated anomalies detected per iteration for the BAF med imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range.

Figure C.5: X-ITERADE results across all iterations for BAF med imbalance dataset.

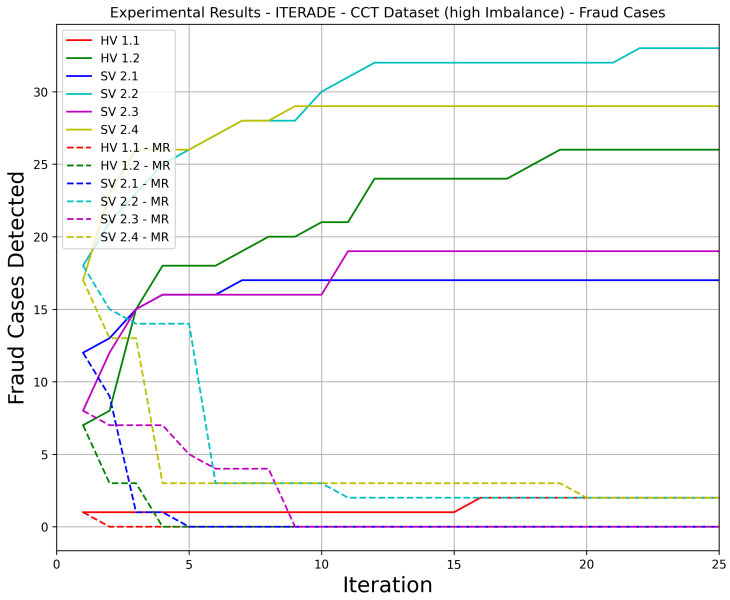


(a) Number of fraud cases and most repeated fraud cases detected per iteration for the BAF low imbalance dataset.

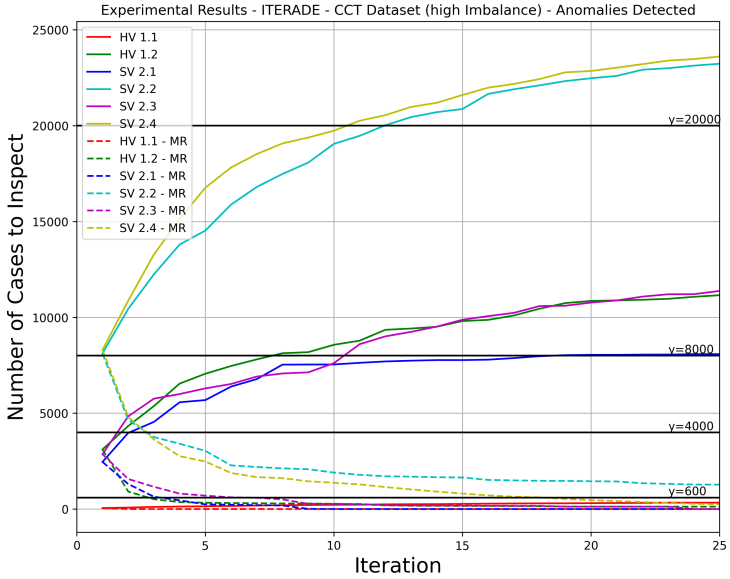


(b) Number of anomalies and most repeated anomalies detected per iteration for the BAF low imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range.

Figure C.6: X-ITERADE results across all iterations for BAF low imbalance dataset.

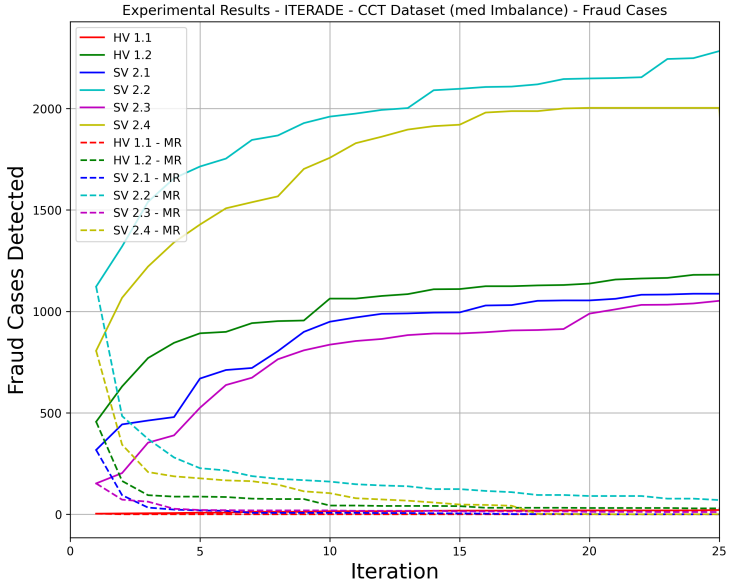


(a) Number of fraud cases and most repeated fraud cases detected per iteration for the CCT high imbalance dataset.

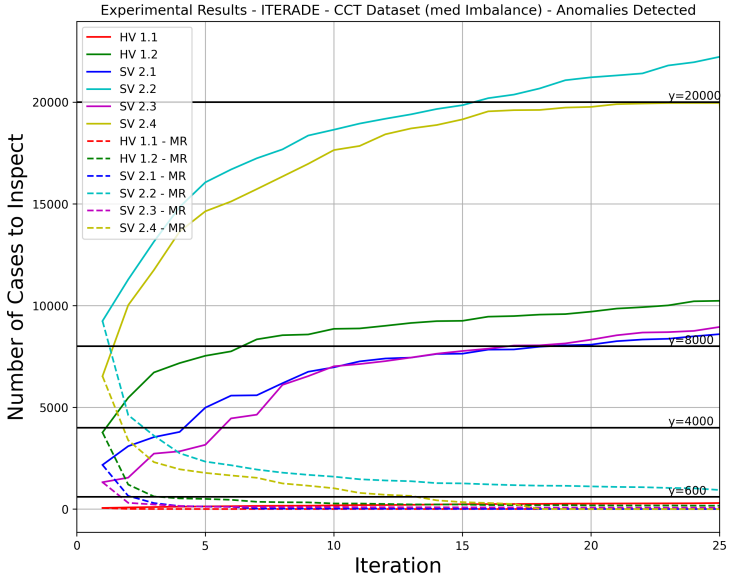


(b) Number of anomalies and most repeated anomalies detected per iteration for the CCT high imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range.

Figure C.7: X-ITERADE results across all iterations for CCT high imbalance dataset.

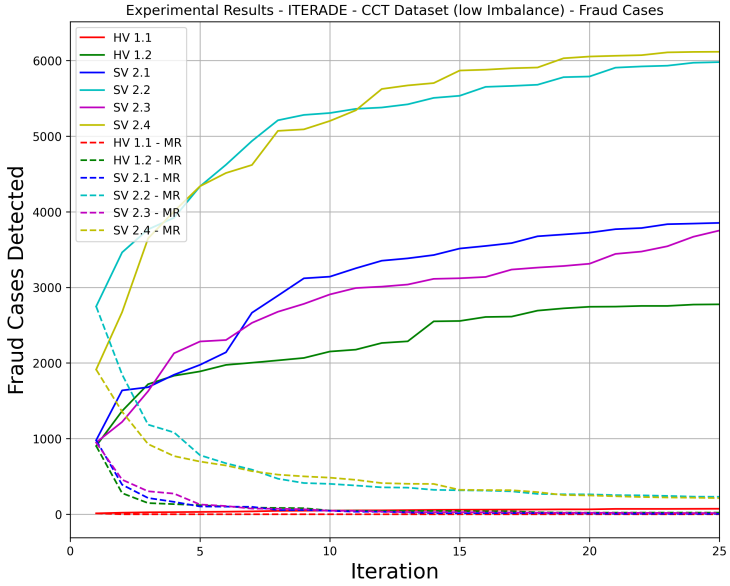


(a) Number of fraud cases and most repeated fraud cases detected per iteration for the CCT med imbalance dataset.

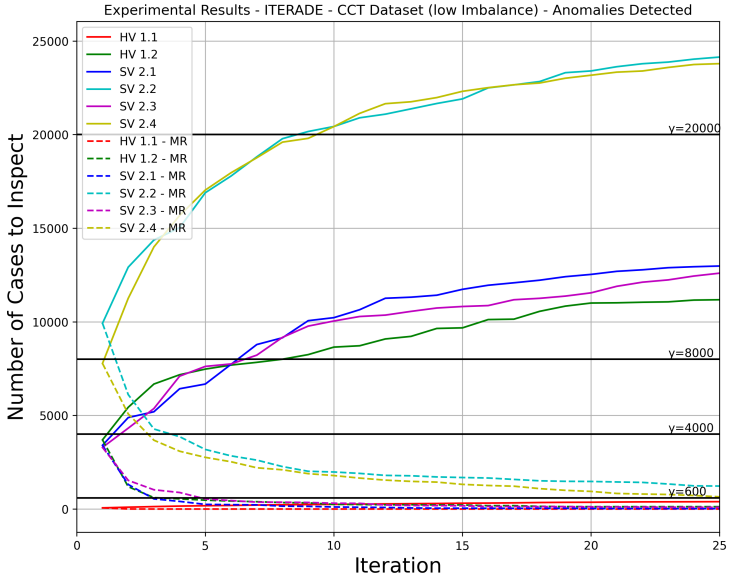


(b) Number of anomalies and most repeated anomalies detected per iteration for the CCT med imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range.

Figure C.8: X-ITERADE results across all iterations for CCT med imbalance dataset.



(a) Number of fraud cases and most repeated fraud cases detected per iteration for the CCT low imbalance dataset.



(b) Number of anomalies and most repeated anomalies detected per iteration for the CCT low imbalance dataset. Horizontal lines indicate budget buckets, showing solutions for each budget range.

Figure C.9: X-ITERADE results across all iterations for CCT low imbalance dataset.