A MACHINE LEARNING BASED FRAMEWORK FOR USER-CENTERED INSIDER THREAT DETECTION

by

Duc C. Le

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
August 2021

Dalhousie University is located in Mi'kma'ki,
the ancestral and unceded territory of the Mi'kmaq.
We are all Treaty people.

Table of Contents

List of Tables

List of Figures

Abstract

Insider threat represents a major cyber-security challenge to companies, organizations, and government agencies. Harmful actions in insider threats are performed by authorized users in organizations. Due to the fact that an insider is authorized to access the organization's computer systems and has knowledge about the organization's security procedures, detecting insider threats is challenging. Many other challenges exist in this detection problem, including unbalanced data, limited ground truth, and possible user behaviour changes. This research proposes a comprehensive machine learning-based framework for insider threat detection, from data pre-processing, a combination of supervised and unsupervised learning, to deep analysis and meaningful result reporting.

For the data pre-processing step, the framework introduces a data extraction approach allowing extraction of numerical feature vectors representing user activities from heterogeneous data, with different data granularity levels and temporal data representations, and enabling applications of machine learning. In the initial detection step of the framework, assume no available ground truth, unsupervised learning methods with different working principles and unsupervised ensembles are explored for anomaly detection to identify anomalous user behaviours that may indicate insider threats. Furthermore, the framework employs supervised and semi-supervised machine learning under limited ground truth availability and real-world conditions to maximize the effectiveness of limited training data and detect insider threats with high precision. Throughout the thesis, realistic evaluation and comprehensive result reporting are performed to facilitate understanding of the framework's performance under real-world conditions.

Evaluation results on publicly available datasets show the effectiveness of the proposed approach. High insider threat detection rates are achieved at very low false positive rates. The robustness of the detection models is also demonstrated and comparisons with the state-of-the-art confirm the advantages of the approach.

List of Abbreviations Used

AE	AutoEncoder
APT	Advanced Persistent Threat
AUC	Area under the [ROC] Curve
CDF	Cumulative distribution function
CERT	Computer Emergency Response Team
C_γ	concatenation of γ data instances
CI	Confidence Interval
DD	Detection Delay
DNS	Domain Name Service
DR	Detection Rate
E_w	mean difference data representation with time window w
FPR	False Positive Rate
GBAD	Graph-Based Anomaly Detection
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
IB	Investigation Budget
IF	Isolation Forest
IT	Information Technology
LANL	Los Alamos National Lab
LDAP	Lightweight Directory Access Protocol
LODA	Lightweight on-line detector of anomalies
LOF	Local Outlier Factor
LP	Label Propagation
LR	Logistic Regression
LS	Label Spreading
ML	Machine Learning
M_w	median difference data representation with time window w

NN	Neural Network
P_w	percentile data representation with time window w
RF	Random Forest
ROC	Receiver Operating Characteristic
ST	Self Training
SVM	Support Vector Machine
TPOT	Tree-based Pipeline Optimization Tool
TWOS	The Wolf of Singapore University of Technology and Design [dataset]
XG	Extreme Gradient Boosting

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Dr. Nur Zincir-Heywood and Dr. Malcolm I. Heywood, for their continuous support, guidance, and encouragement. Their guidance and words of encouragement always inspire me to do my best. I would not be capable of being the person I am today without their support and faith in me.

I also would like to thank the rest of my thesis committee: Dr. Mohammad Zulkernine, Dr. Srinivas Sampalli, and Dr. Andrew McIntyre, for generously offering their time, expertise, and insightful comments.

I greatly appreciate the financial support from the Killam Trusts, Nova Scotia Government, Dalhousie University, and Mitacs. My gratitude extends to Interset at Micro Focus for the internship opportunity to apply the research in this thesis. This research was enabled in part by computing resources provided by Compute Canada.

With great pleasure, I would like to thank my family for unceasing encouragement and support throughout my life. Thank you, mom, for raising me and for all your sacrifices to allow my pursuits. Last but not least, I want to thank my wife, Thao Vo, for her support, companionship, and great food. I am truly thankful for having her in my life.

Chapter 1

Introduction

Insider threat is one of the most dangerous and prevalent security threats that various institutions, companies and government agencies are facing, where the malicious acts are performed by authorized personnel inside the organization. Due to the fact that insiders are authorized to access an organization's networked systems and are knowledgeable about its structure and security procedures, an insider threat is one of the most costly types of attacks and hardest to detect. Cyber-security reports show that at least a half of U.S. companies and public sector organizations (e.g. federal agencies) suffer from insider threats every year [83, 102]. With the rise of remote working and cloud computing, organizations are even more exposed to insider threats. Recent surveys show that 68% of organizations think that insider attacks have become more frequent, and 70% have experienced at least one insider attack within the last 12 months [32]. The overall cost of insider threat incidents is measured at an average of $11.45 million per organization researched in a 2020 report [96]. The leading factors contributing to the increase in insider threats are: Increased use of applications that can leak data (e.g. web emails), increased amount of data that leaves the protected perimeter of an organization under 'normal' use cases, more end-user devices, and migration of sensitive data to the cloud [32].

The insider threat is defined in a recent technical report by the CERT Insider Threat Center as threats that are carried out by malicious or unintentional insiders, whose authorized access to the organization's network, system, and data is exploited to negatively affect the confidentiality, integrity, availability, or physical well-being of the organization's information, information systems, or workforce [105]. Some insider threat related incidents reported, to date, are data compromisation, such as customer records, trade secrets or intellectual property, theft of personally identifiable information, and IT system sabotage [31, 105]. Not only may insider threats directly cause operational disruption / outage, loss of critical data, and incurring remediating costs,

but the impact of insider threats may also result in loss of revenue and competitive edge, brand damage, and legal liabilities [32, 96]. Furthermore, the impact of insider threat is potentially multiplied by the delay in detection, due to the challenges in detecting insider threats. Reports show that detecting and containing insider attacks takes 77 days on average [96], and in 42% of the cases, remediation is after data loss has occurred [32]. In short, insider threat poses a serious cyber-security challenge, which needs to be addressed with high priority to ensure the continued security of such systems and therefore an organization's functionality.

Distinct from traditional intrusion detection tasks, many challenges of insider threat detection come from the fact that the insider is authorized to access the organization's computer systems and has familiarity with the organization's security layers. Furthermore, in most organizations, the activities of insiders with malicious intent occur infrequently. Thus, data available to describe the activity is usually rare and not well-documented [7]. Finally, challenges of insider threat detection may arise from the need to process and investigate a wide range of data types in organizational environments, from network traffic, web and file access logs, to email history, or employee information. The available data also differs significantly by organization. The detection problem is further complicated by the recent trend towards migrating services to the cloud and remote working. Hence only a small fraction of organizations have tools and (the human) resources to interpret user's behaviour and intent from the monitoring data collected [32, 83].

This thesis presents a machine learning (ML) based system that focuses on user-centered analysis for insider threat detection, which aims to present a complete ML based insider threat detection solution. We propose and evaluate a workflow for user-centered insider threat detection, from data collection and pre-processing, anomaly detection, to data analysis using ML classification models, and alert reporting and analysis. The proposed system aims to assist cyber-security analysts while employing minimal data labelling overheads (unsupervised learning) or working under limited ground truth conditions to identify threats in unknown data, and provide useful insights to insider threat remediation.

1.1 Research Objectives

The main objective of this research is to propose and evaluate a ML based framework with different stages of insider threat detection in corporate and organizational environments. Based on real-world conditions for insider threat detection, the research aims to emphasize the use of machine learning with no ground truth (without labelled data) and with very limited ground truth for anomalous behaviours and insider threat detection. To this end, the following objectives are studied on different data sources that were publicly available to researchers.

1. Exploring data processing techniques to extract data with rich details describing user activities from heterogeneous organizational data sources.

2. Assessing the impacts of different levels of data granularity, in terms of insider detection performance and response time.

3. Exploring different temporal representations of data to highlight user behaviour changes, in order to improve detection effectiveness.

4. Examining unsupervised ML techniques for identifying signs of anomalous behaviours that may indicate insider threats. This is the initial and important detection step in cyber-security workflow, where early signs of user behaviour changes (anomaly) are flagged for further investigation, potentially to detect both known and unknown (zero-day) attacks/vulnerabilities.

5. Exploring anomaly detection ensembles based on different schemes of combining anomaly detection methods, in order to enhance detection capabilities and robustness.

6. Investigating different conditions for the application of ML to insider threat detection. Realistic limitations are adopted in order to obtain results reflecting real-world environments and highlight the differences compared to training under ideal (typical ML) conditions. Furthermore, impacts of different factors, e.g. training data poisoning, number of users in training data, and the duration of training data, are examined.

7. Investigating the effect of various supervised and semi-supervised machine learning algorithms on insider threat detection, under different situations reflecting real-world cases, in order to maximize the effectiveness of limited labelled training data.

1.2 Contributions

The main contribution of this thesis is the design and evaluation of a complete insider threat detection framework based on machine learning, from data pre-processing to initial detection using anomaly detection, and insider threat classification under realistic conditions including the constraints faced in practice in organizations. In doing so, the contributions of this thesis can be summarized as follows.

1. A complete system, from processing raw log data, a combination of supervised and unsupervised learning, to deep result analysis, is proposed to assist the cyber-security analyst in all phases of data monitoring and analysis for insider threat detection.

2. The research comprehensively assesses the effect of different data granularity levels and training conditions of ML in insider threat detection. Furthermore, different representations of data, namely concatenation, percentile, mean and median difference, are introduced for ML-based anomaly detection algorithms, where temporal information is encoded to highlight user behaviour changes.

3. Unsupervised machine learning approaches and unsupervised ensembles are investigated in anomaly detection, as an initial step in insider threat detection. The proposed approach demonstrates the ability to generalize and detect malicious insiders under very low investigation budgets, which achieves state-of-the-art performance and robustness on insider threat detection and related problems.

4. Assuming limited available ground truth and training regimes with constraints reflecting real-world conditions, supervised and semi-supervised machine learning algorithms are examined and shown to be able to improve detection precision

and accuracy. The approach allows learning from very limited data for insider threat detection in unseen data at high recall and very low positive rates. Additionally, automated model optimization based on evolutionary computation is explored for further performance gains.

5. Comprehensive result reporting is performed at each step to adequately present detection performance under real-world considerations. Per instance and per user results are reported throughout the thesis, and malicious cases are investigated, for a better view of system performance. Furthermore, detection delay metric is investigated to measure the elapsed time between malicious behaviour and detection.

The performance of the proposed system is demonstrated on publicly available datasets depicting insider threat and related activities. Several results have been published in journals and conferences in related fields, such as network security and management, cyber-security, artificial intelligence and machine learning [38, 60–72].

The unsupervised machine learning / anomaly detection approaches to the initial detection step have been published in [63, 66, 67, 69]. In [63], different approaches to data extraction for machine learning for insider threat detection, namely sequential and numerical data, were explored. In [66, 69], the focus was to employ temporal representations of data to achieve state-of-the-art detection performance using popular anomaly detection techniques and ensemble approaches. Later detection steps are presented in [61, 65, 67, 70–72]. Specifically, in [65, 71], machine learning classification techniques are employed to examine effects of data granularity levels, and limited training conditions, as well as to perform malicious behaviour analysis. Feature reduction and transformation techniques are examined in [38, 67]. On the other hand, genetic programming based approaches are examined in [60, 61] for insider threat detection under dynamic conditions of feature and label spaces. Finally, in [72], we further examine the ability of machine learning under real-world conditions and constraints for insider threat detection via different training regimes for a semi-supervised learning based approach.

1.3 Organization of the Thesis

The thesis is structured into eight chapters, as follows.

Chapter 2 provides background information and reviews the literature on insider threat detection and related cyber-security problems. While ML has been adopted widely for insider threat in the literature, different issues are identified and presented in the chapter, along with the research gaps addressed in this thesis.

Chapter 3 presents an overview of the proposed system, from data sources, pre-processing, to initial detection and classification, to result reporting and analysis.

Chapter 4 details the employed datasets and pre-processing steps to enable machine learning applications. Data extraction approaches, sequential and numerical, are compared in this chapter. Furthermore, this chapter introduces different levels of data granularity and temporal representations of data.

Chapter 5 describes anomaly detection approaches based on unsupervised machine learning for identifying signs of anomalous behaviours that may indicate insider threats. This is the initial and important detection step in cyber-security workflow, where early signs of user behavioural changes (anomaly) are flagged for further investigation, potentially to detect both known and unknown (zero-day) attacks/vulnerabilities.

Chapter 6 details the use of supervised machine learning on multiple levels of data granularity and realistic conditions for identifying not only malicious behaviours, but also malicious insiders. Detailed analysis of popular insider threat scenarios with different performance measures is presented in this chapter to facilitate the realistic estimation of system performance.

Chapter 7 presents a semi-supervised learning approach to insider threat detection under different training regimes reflecting limited real-world conditions. These include obtaining the initial ground truth data through random sampling versus knowledge of a certain type of insider malicious behaviour or by anomaly detection system scores.

Finally, Chapter 8 wraps the thesis by providing conclusions summarizing the research and discusses possible future research directions.

Chapter 2

Background and Related Work

Insider threat detection is a challenging research problem, not only to the research community but also to government agencies and cyber-security firms. Thus, research into insider threat detection and related cyber-security issues attracted a lot of attention in recent years. This chapter presents background on the insider threat problem and related work from recent literature.

2.1 Background

Insider threat is defined in a guide to mitigation by the US's Cyber-security and Infrastructure Security Agency [31] as:

"the potential for an insider to use their authorized access or special understanding of an organization to harm that organization. This harm can include malicious, complacent, or unintentional acts that negatively affect the integrity, confidentiality, and availability of the organization, its data, personnel, facilities, and associated resources".

In that, an *insider* is *"any person who has or had authorized access to or knowledge of an organization's resources, including personnel, facilities, information, equipment, networks, and systems"* [31].

Figure 2.1 [29] show a summary the factors involved in insider threat, from individuals, organization's assets, to insiders' actions and their consequences. The defining characteristic of insider threat is *authorized access* entitled to a current or former employee, contractor, or business partner, by that harmful actions are performed.

Insider threats can be intentional or unintentional. *Intentional* insiders, or malicious insiders, carry out actions that harm an organization for personal benefit or to act on a personal grievance intentionally. Common factors that lead to malicious insider actions are perceived grievance, ambition, financial pressures, or even perceived public good. Insider IT sabotage and workplace violence may originate

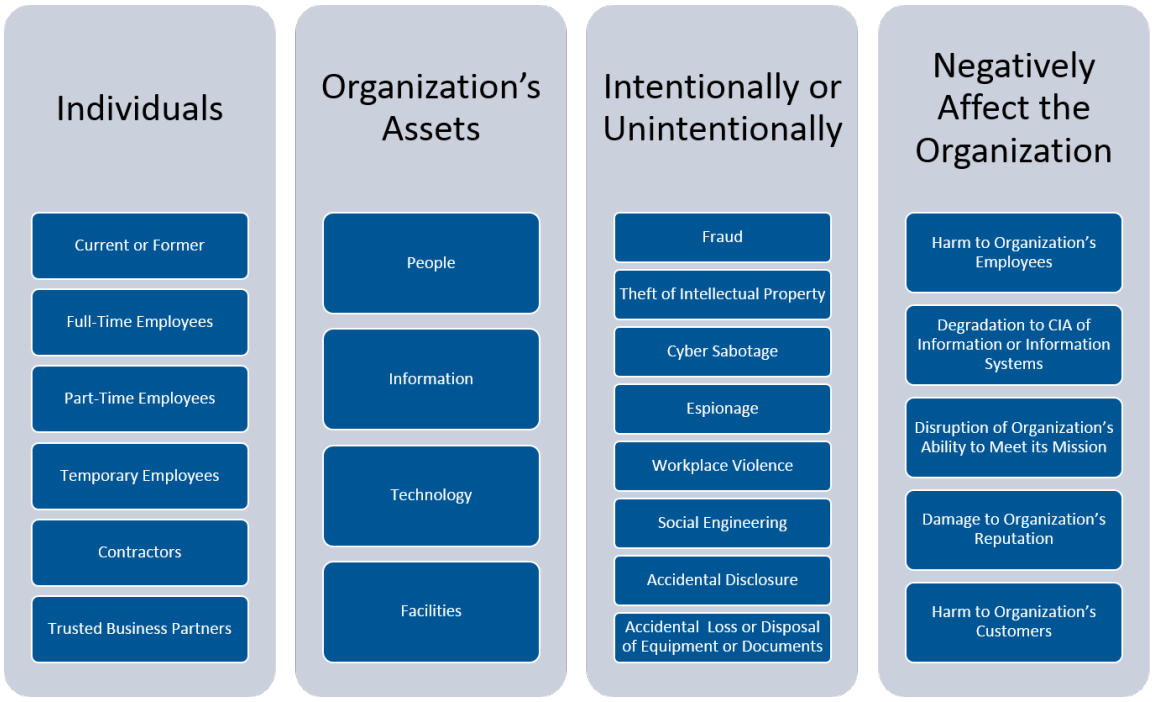


Figure 2.1: Insider threat factors [29]

from disgruntlement, which in turn is possibly the result of termination or unmet expectations (e.g. insufficient salary increase or bonus, diminished authority or responsibilities, perception of unfair work requirements) [105]. In many cases, malicious insiders have stolen proprietary data or intellectual property to advance their careers, or to benefit their new organizations. Insiders may also collaborate with an external threat actor to compromise an organization (collusion). These incidents frequently involve cyber-criminals recruiting an insider to enable fraud, intellectual property theft, espionage, or a combination of the three [31]. In 2020, more than a quarter of insider threat cases recorded by the CERT National Insider Threat Center involve collusion [86].

Unintentional insiders are an organization's personnel who bear them no malice but whose actions unintentionally expose the organizations to risk in some way [48]. The source of unintentional insider threat can be negligent or accidental acts. Negligent insiders may expose an organization to a threat by their lack of good judgment or carelessness. Insiders of this type are generally familiar with security and/or IT

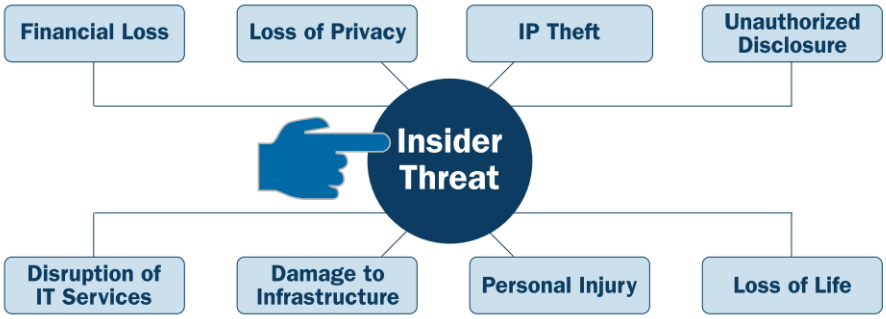


Figure 2.2: Potential Consequences of an Insider Incident [31]

policies but choose to ignore them, creating risk to the organization. Examples include improper use and management of personal equipment, misplacing or losing a portable storage device containing sensitive information, and ignoring messages to install new updates and security patches [31,48]. On the other hand, accidental unintentional insider threat is the potential unintended risk to an organization, caused by mistakes of oblivious or naïve employees. Examples include accidental disclosure (e.g. via the internet) of sensitive information, sending sensitive information to the wrong party via email, unknowingly or inadvertently clicking on a hyperlink or opening an attachment that contains a virus within a phishing email, or improperly disposing of sensitive documents [31].

Figure 2.2 [31] lists some potential consequences of an insider incident. Losses from insider threat may result from physical damage to infrastructure, disruption of productivity, intellectual property theft, accidental leakage of sensitive data, or insult to an organization's reputation [31]. The severity and financial damage caused by insider threat also come from the time it takes to contain an insider incident. For example, a recent survey shows that incidents that took more than 90 days to contain cost organizations $13.71 million on an annualized basis, while incidents that lasted less than 30 days cost roughly half, at $7.12 million [96]. The same survey also indicates that it takes an average of more than two months to contain an insider incident.

While employee training and education programs may prevent or reduce the prevalence of some types of insider threats, such as negligent acts, many factors leading to insider threats are unavoidable in organizational environments. For example, organizations can successfully work to minimize accidents, but they will occur and may

not be completely prevented [31]. Because of the damage and urgency, research into insider threat detection is greatly needed. In the following section, we summarize the literature in this field.

2.2 Related Work

In this section, firstly the literature surveys and general guidelines to insider threat detection and mitigation are presented in Section 2.2.1. As the focus of this thesis is machine learning applications in insider threat detection, we quickly discuss non-ML based research in this field in Section 2.2.2, before focusing on ML based approaches in Section 2.2.3. Finally, related cyber-security problems, such as user identification and lateral movement detection, are mentioned in Section 2.2.4.

2.2.1 Literature Surveys and Guidelines

Addressing the rising problem of insider threats, many guides are published to promote best cyber-security practices for preventing and detecting insider threats [30,31, 87, 105, 111]. The CERT National Insider Threat Center publishes "Common Sense Guide to Mitigating Insider Threats", currently in its sixth edition [105]. The guideline describes 21 practices that organizations should implement across the enterprise to prevent and detect insider threats, as well as case studies of organizations that failed to do so. Examples are "Know and protect your critical assets", "Monitor and respond to suspicious or disruptive behaviour", and "Enforce separation of duties and least privilege" [105]. Each practice includes challenges to implementation, high-impact solutions for small and large organizations, and is mapped to relevant security standards. The guide focuses on six groups within an organization – Human Resources, Legal, Physical Security, Data Owners, Information Technology, and Software Engineering – and maps the relevant groups to each practice.

Recently, an insider threat mitigation guide is published by the US's Cybersecurity and Infrastructure Security Agency [31]. The guide provides up-to-date definitions of insider threat and relevant factors, and outlines the principles to build a successful insider threat mitigation program. Not only indicators to detect and identify insider threats are presented, but the guide also outlines steps to assess and manage insider threats. In Canada, a recent guide by Public safety Canada [30] presents eight

recommended security actions to enhance Canada's critical infrastructure resilience to insider risk. The guide is organized into three main themes: "Establish a holistic approach to security", "know and empower your people", and "identify and protect what is critical".

Earlier surveys [7, 48, 100] summarize the problem of insider attack and unintentional insider threat in computer science literature. In [7], Azaria et al. presents a multidisciplinary survey of insider threat capturing contributions from computer scientists, psychologists, criminologists, and security practitioners, and highlights the challenges in highly imbalanced data in insider threat detection. Recent surveys, [52, 81], provide systematization of knowledge in the insider threat research literature. In [81], Liu et al. reviewed insider threats and related cyber-security problems, such as malware and advanced persistent threats. The survey is organized around the advanced persistent threat (APT) intrusion kill chain and covers three major types of insiders, namely traitor, masquerader and unintentional perpetrator. The survey also discusses detection approaches from a data analytics perspective, where the literature is presented according to host, network, or contextual data based analytics. In [52], Homoliak et al. focus on a structural taxonomy and novel categorization of insider threat related research. The work identified four main categories of research and development efforts: Incidents and datasets, analysis of incidents, simulations, and defence solutions.

2.2.2 Non-ML Detection Approaches

As the insider threat problem can be related to human factors, many researchers approach the problem using psychological models and decision-making theories [47, 74, 75, 91]. In [91], Padayachee et al. applied opportunity theories from criminology for conceptualizing insider threats, as well as opportunity-reducing measures for assisting insider threat mitigation. The evaluation in the paper focuses on proactive mitigation strategy and seeks to conceptualize the element of opportunity for implementing information security controls, such as increase the effort [of performing insider attack] (e.g. control of access to facilities, target hardening), reduce the rewards (e.g. concealing and removing targets, denying benefits [of successful attacks]), reduce provocations,

and reduce excuses. In [47], Greitzer et al. proposed a predictive modelling framework that integrates multiple data sources and psychological/motivational factors for assisting the data analyst in detecting high-risk behaviours. The framework performs reasoning based on hierarchical ontological networks and a memory mechanism to identify changes in user behaviours. The knowledge is represented using Ontology Web Language in the work. However, the work did not perform an evaluation to demonstrate the effectiveness of the proposed approach.

Legg et al. in [74] proposed a framework for modelling insider threats based on behavioural and psychological observations. Based on a proposed conceptual model of relationships between elements in four categories: Enterprise, information, technology and physical, the framework allows an analyst to reason and describe potential insider threats from different domains, such as human behaviours and organizational policies. Nevertheless, the work lacks evaluations showing how such a model can be applied successfully in real-world contexts. In [75], Legg et al. further explored the conceptual model with multi-level anomaly alerts based on policy violations, previously recognized attacks, or distance-based thresholds and deviations. The work defined a tree-structure profiling approach to generate user and role-based profiles to describe the users' activities within an organization. Visualization techniques are applied to highlight user activities for inspection and provide deviation measurement based on comparison across peers and observations. Evaluations are performed on their own synthetic datasets and an early version of the CERT insider threat dataset [24]. While the number of generated alerts are presented in the results, the paper remains vague in the actual detection performance of the system.

In [88], Nurse et al. introduced a modelling framework to characterize insider attacks and to facilitate an understanding of the problem, its many components and how they all fit together. The model is heavily dependent on experts knowledge in the domain, which is acquired by manual investigation of insider threats accidents case by case, over different contexts and organization structures. The framework is useful for general understanding of the threat, and also for reflection, modelling past attacks and looking for useful patterns. In [3], Agrafiotis et al. expanded the framework and described a methodology for identification of behaviours of interest from 120 insider attack case studies analyzed. Attack-pattern trees are designed to

highlight the possible paths that an attacker can follow for a specific threat. The work suggests that constructed attack steps can be linked to features and anomaly metrics to enhance detection systems.

Finally, in [51], Ho et al. focus on the interpretation of dynamic human information behaviour in an organizational setting, which takes into account the trustworthiness and human sensors' attribution in close relationships. Through a number of propositions, the paper argues that the group can collectively attribute a shift in trustworthiness – via communication cues – when an individual violates integrity-based trust. The work demonstrated the application in an empirical experiment, where a focal actor's betrayal concealed in the deceptive activities were identifiable through the group's collective cognitive and communication.

2.2.3 Machine Learning based Insider Threat Detection

Considering the huge amount of data acquired daily in any organization, ML based solutions are one of the most promising approaches for solving cyber-security challenges in the current era [13, 21]. The advantage of ML is the ability to automatically learn from a large amount of data and identify patterns potentially characterizing malicious activities or anomalous behaviours.

2.2.3.1 Anomaly detection

Anomaly detection is a popular approach employing ML in insider threat detection, where normal user behaviour models are built and anomalies are identified as deviations from the normal behaviour. An anomalous alert, in this case, may indicate changes in employee behaviours as a potential early indicator of insider threats.

A popular research direction for this problem is graph-based anomaly detection [17, 37, 40, 78, 92, 106]. Graphs are employed for their ability to capture and model relationships, such as interactions between users and files. In [37], Eberle et al. employed Graph-Based Anomaly Detection (GBAD) for insider threat detection. GBAD takes input as a graph in which labelled vertices and edges represent entities and relationships/actions between entities, respectively. Three possible changes to a graph – modifications, insertions and deletions – are measured using the minimum description length, which allows GBAD to discover anomalous instances of structural

patterns in graph data. The approach is evaluated using Enron email dataset and cellphone traffic to identify interesting graph patterns. In [92], Parveen et al. expanded the GBAD based learning approach with stream mining. Three varieties of GBAD are employed to generate an ensemble of models, and stream mining is enabled using the weighted average between the most recent data chunks. The approach is evaluated on 1998 Lincoln Laboratory intrusion detection dataset of system calls and demonstrated high detection rates at low false positive rates in stream settings.

In [17], Brdiczka et al. proposed a framework combining structural anomaly detection and psychological profiling for detecting insider threats. Graph analysis, dynamic tracking, and machine learning are performed in structural anomaly detection step to detect anomalies in network data, while psychological profiling step constructs dynamic psychological profiles from behavioural patterns. Threats are identified through a fusion and ranking of outcomes from the two steps. The approach was evaluated on an online gaming dataset (World of Warcraft) to detect anomalous behaviours (guild quitting) and predict a player's personality from in-game behaviour, achieved up to 89% accuracy.

Gamachchi et al. proposed a framework based on a graph learning and Isolation Forest to analyze heterogeneous data in isolating possible malicious users [40]. Based on release R4.2 of the CERT insider threat dataset [24], the framework generates graph attributes from authentication and web access data. Isolation forest is then employed to produce anomaly scores for individual users from graph extracted features and features extracted directly from authentication, removable media usage records, and psychometric scores. The work reported the distribution of anomaly scores but failed to show the effectiveness of the approach in detecting malicious users in the dataset.

Recently, in [78], Liu et al. presented an approach based on graph embedding for detecting insider and related cyber threats. Heterogeneous graphs are generated based on data sources and time, using a set of graph construction rules. Word2vec is then employed to map log entries into low-dimensional vectors through context generated from the heterogeneous graph using random walk. A clustering algorithm can then be applied to the extracted data and thresholds are set to isolate the suspicious log entries. The approach achieved Area Under the Curve (AUC) of 0.93 on CERT

dataset R6.2, and 0.91 on LANL dataset [55]. However, the evaluations are performed only on subsets picked the datasets: On CERT R6.2, only 6 malicious users and 12 normal users (in total 4000 users) are included; on LANL dataset, only 50 (in 98) malicious users and 90 (in 11K) normal users are included.

Another approach to anomaly detection is to model the sequences of user actions and employ them to detect unusual sequences [97, 103]. A Hidden Markov Model (HMM) was applied by Rashid et al. in [97] to capture each user's normal weekly activity sequences of common activities. Anomalous weekly action sequences are detected by the model through log probability generated by the user's HMM, and each user's HMM is updated by retraining with the week's sequence if it is confirmed to be normal. The approach is evaluated on CERT R4.2 dataset, in that 1000 HMMs are maintained and updated every week for 1000 users in the dataset, and achieved AUC of 0.83.

In [5], Aldairi et al. presented an approach to extract features from heterogeneous log data representing user activities periodically (daily to yearly). In addition to extracted data features, the work employs trust scores based on anomaly scores generated for each user in a previous cycle to train the detection model in each cycle. Using two unsupervised learning algorithms – One class Support Vector Machine (SVM) and Isolation forest, the approach achieved AUC of 0.89 (daily) to 0.97 (yearly) on CERT R4.2 dataset. Nevertheless, the value of insider threat detection under long periods (quarterly, yearly) is questionable, as with those data extraction settings, detection would likely occur after all the damages happened. Furthermore, reviewing a quarter or a year of an user activities to confirm an alert would be a significant burden to the analyst.

Many insider threat detection systems, [17, 41, 41, 45, 101, 103], were supported by the U.S. Defense Advanced Research Projects agency's project ADAMS[1] – Anomaly Detection At Multiple-scales. The project aimed to "identify patterns and anomalies in very large datasets" in order to detect and prevent insider threats. In [45, 103], various anomaly detection algorithms, including HMM and Gaussian Mixture Models (GMM), were employed in an ensemble on user activity log data for identifying insider threat indicators. The ensemble detection approach works on features extracted from

[1] https://www.darpa.mil/program/anomaly-detection-at-multiple-scales

a day or a month of user activities. Additionally, a visual language for describing anomalies was proposed in the work of [103]. Evaluations are performed on a private dataset with inserted red-team activities, and showed detection AUC of 0.76 to 0.93, by month.

With the popularity of deep learning and its growing applications in cyber-security, different approaches based on the algorithms were proposed for insider threat detection recently [79, 80, 84, 108]. In [108], Tuor et al. proposed an anomaly detection system based on retraining a deep neural network, or recurrent neural networks on daily user data to generate anomaly scores. Similar to [97], the approach requires maintaining a detection model per user and updating it daily. While starting with no ground truth, the approach requires labelling by the analyst in each training cycle (i.e. a day), in order to calculate the loss function for training the neural networks. Test results on CERT R6.2 (obtained on 15% of the dataset on weekdays only) achieved 100% detection rate at a daily budget of 400 alerts, which is equivalent to 10% FPR (of 4000 users' daily data).

In [79], Liu et al. presented an approach to convert weekly user's log data from multiple sources into text corpus. A neural network (Word2vec) is then employed to learn word associations from the corpus and produce behavioural probabilities. Based on the probabilities, a threshold is used to indicate if users' events are suspicious, and a user is labelled as malicious if he/she is associated with multiple suspicious events. The approach is evaluated on a subset of 500 (in 4000) users and 3 (in 6) malicious users picked from CERT R6.2 dataset, and achieved an AUC of 0.956.

2.2.3.2 Other ML based approaches

Other recent ML based approaches to insider threat detection include supervised learning [41, 53], stream online learning [15, 108], and Bayesian-based approaches [85, 99].

Gavai et al. applied different ML methods on organizational data to detect not only anomalies but also early "quitter" indicators, where both may suggest insider threats [41]. Based on a private dataset, the work extract features from social data including email communication patterns and content, web browsing, and file and machine access patterns. Isolation forest is employed for anomaly detection, while

random forest is used for quitter classification. Results show AUC of 0.77 for anomaly detection, and normalized accuracy of 73.4% in quitter detection.

In [53], Jiang et al. applied graph convolutional neural network for insider threat detection. Proposing that users' relationships may contain essential information that is useful to detect malicious users or groups, the work characterizes users' behaviours and their connection relationships into a graph to train graph convolutional network models. Evaluated on CERT R4.2, the supervised learning approach achieved 94.5% accuracy and 83% detection rate.

Bose et al. proposed a system employing scalable supervised and unsupervised learning algorithms on a fusion of heterogeneous data streams to detect anomalies and insider threats [15]. The work focuses on real-time processing and analysis of data and training of the detection models. In the anomaly detection component, a hierarchy of simple correlation filters is used to target malicious actions, such as data exfiltration and inappropriate access, and aggregate anomaly alerts. In the detection component, streaming machine learning based on k-nearest neighbours algorithm is used. On an older version of the CERT insider threat dataset (release R2), detection rate of 50% at 8% precision is demonstrated by the approach.

In [99] Roberts et al. applied Bayesian Networks as a modelling tool for enterprise data. Bayesian network is selected for its portability, privacy, and simplicity. The work also compares different scenarios, e.g. a change in the organizational environment or a detector/sensor, for generating Bayesian networks. Test results on subsets (3 months) of CERT R4.2 dataset, where data instances are represented monthly in the approach, showed 100% detection rate at 0.89% False positive rate.

In [85], Meng et al. employed a Bayesian based trust management approach for insider attacks in healthcare software-defined networks. Based on findings from surveys on healthcare organizations, the authors develop a trust-based approach based on Bayesian inference to detect malicious devices in healthcare environments. In the software defined network, the centralized server observes the traffic and assign trust levels to devices based on the total number of packets and the number of benign packets, using Bayesian rules. Experimental evaluations are performed on simulated data and in a real hospital environment to demonstrate the effectiveness of the approach, where the trust value of malicious devices are quickly decreased.

2.2.4 Related Cyber-security Problems

User identification and masquerader detection are closely related to insider threat [81]. In [101], Salem and Stolfo proposed a user profiling approach via modelling user search behaviour to detect deviations indicating a masquerade attack. The work hypothesizes that differences appear in search behaviours, where each normal individual knows their own file system well enough to quickly search and locate the target file/information. Masqueraders, on the other hand, would likely search more extensively and broadly or in a different manner due to the lack of knowledge about the file system and layout. A small set of search related features is extracted in the work to train One class SVM based anomaly detection. Results on RUU dataset of Windows log containing 18 normal users' data and simulated masquerader data – extracted by 2 minute periods – showed AUC of 0.98, and 100% detection rate at 1.1% false positive rate.

Similarly, in [106], Toffalini et al. proposed masquerader detection approaches based on anomaly detection in user search and file access behaviours. The work employs a graph partitioning technique (Markov Clustering) on weighted oriented graphs generated from a history of a user and events recorded in a time window (the user's session), based on the idea that strongly connected nodes have to belong into the same cluster. New input in a time window is compared to the user's graph through a similarity function defined over pairs of vertex clusters. The approach achieved a mean AUC of 0.944 on 2 minutes time windows of WUIL dataset, and 0.851 on 30 minutes time windows of TWOS dataset [49].

In [94], Peng et al. reviewed user profiling in intrusion detection. The work summaries behavioural based intrusion detection systems from the viewpoint of user behaviours, based on biometric, psychometric, or combined user profiles. In [56], Kent et al. proposed representing authentication data in enterprise networks as a set of user-specific authentication graphs. Based on the graphs, the work extracts features using a proposed time-constrained path distance algorithm to enable analysis, such as user classification and intrusion detection. In a private dataset from Los Alamos National Laboratory (LANL), the work demonstrated the ability to differentiate administrative and general users as well as finding compromised users, with AUCs of 0.907 and 0.915, respectively.

Another closely related problem to insider threat detection is lateral movement detection, in which many malicious actions are also performed using insider accounts. Lateral movement is used as a technique in advanced persistent threat by threat actors to gain access to their intended targets, moving through compromised accounts and systems of the victim organization. Recently proposed machine learning approaches for lateral movement detection are graph analysis [16, 78, 116], recurrent neural networks [20], and other supervised learning techniques [14]. In [116], Zhao et al. proposed a continuous temporal lateral movement detection framework, in which remote and local authentication events are represented as a path connection graph and a bipartite graph, respectively. Path features are generated from the graph dataset, using breadth-first search algorithm with time constraints. Anomaly detection based on familiarity measurement is then applied to the extracted features to detect anomalous authentication paths. The approach is evaluated on LANL dataset [55] and achieved an AUC of 0.92 in detecting their injected attacks (instead of attacks provided in the dataset).

In [16], a similar graph-based approach is proposed by Bowman et al. Using an authentication graph created from input data, the approach performs unsupervised node embedding generation to enable anomaly detection. Low-probability authentication events are learned via a logistic regression link predictor. The approach requires pure normal data for training and achieves a detection rate of 85% with 0.9% false positive rate on LANL dataset (30% as testing data). On the same dataset, Bian et al. [14] instead applied *supervised* ML methods to classify daily malicious authentications based on extracted features from authentication graphs. Best AUC of 0.995 achieved by Random forest, on data subsets of only one day of remote authentication actions. In [20], Brown et al. employed recurrent neural network with attention mechanism to detect anomalous authentication activities at log line level on LANL dataset. Test results on a single day show AUC of 0.99 at log line level.

2.3 Summary

Background on insider threat and the related work in the literature on insider threat detection are reviewed in this chapter. The related literature is organized into four

main categories: (1) literature surveys and guidelines, (2) non-ML detection approaches, (3) ML based detection approaches, and (4) related cyber-security problems.

The following particular gaps can be observed from the literature on insider threat detection approaches, which we aim to address:

- There are only several works capable of learning from heterogeneous data for detection [15, 40, 79, 108]. This naturally creates a problem of ranking and prioritizing anomaly alerts from detection models on different data sources to present to the analyst, which may not be intuitive in an unsupervised setting. Furthermore, many works in the literature require maintaining and updating multiple complex detection models, such as deep learning, per each user in an organization [40, 97, 106, 108]. Not only this greatly increases the computation and storage requirements, but ranking anomaly scores by users also needs to be addressed as well, as each user may have a different scale for anomaly scores based on their personalized detection model. To avoid the issues and provide a unified view to the security analyst for simplifying the investigation process, the proposed approach in this research is designed to work on heterogeneous data (from all sources). We also focus on the use of unified detection models for all users, which naturally provides the ability to prioritize users and behaviours in its alerts, while reducing computation overheads.

- While the objective of insider threat detection is malicious and/or masquerading users and behaviours, most of the works in the literature omit reporting the results based on users. There are only a few works that report user-based results in a limited manner [40, 79, 106]. Due to the diversity in a users' roles and behaviours within an organization, high detection rates by data instances or log lines may not necessarily translate to all malicious insiders being detected. In an attempt to shed light on this issue and to provide user-centered reporting of the system's results, we distinguish between malicious actions detected and malicious users detected in this research. User-based results are reported throughout the thesis for that purpose. Similarly, although the delay in detection is directly related to the cost of insider incidents [96], none of the work in the literature to date addressed this issue. To address this, in this thesis, we

report detection delay, i.e. the time period between the first malicious action and when the insider is detected by the system.

- In the literature, there is rarely a comprehensive detection approach for insider threat detection [15], where the vast majority of proposed works are based on either unsupervised anomaly detection or supervised ML for classification. In this thesis, we introduce a complete system to assist the cyber-security analyst in all phases of data monitoring and analysis, from processing the raw log data, a combination of unsupervised and supervised learning, to deep analysis and meaningful result reporting.

- A common problem to proposed approaches in the literature is that they assume unrealistic settings for training and evaluation of ML approaches [5, 78–80, 99, 108], such as incomplete evaluation (only on a selected subset of test data), private datasets, and partial result reporting. Throughout this thesis, we adopt a realistic setting for training and evaluation of the detection approach, and report results comprehensively, in order to demonstrate the system performances in real-world conditions.

- Some works in the literature report the importance of temporal information in dealing with insider threat, which is highly related to the human factors [84, 97, 103, 108]. Notable attempts to leverage temporal information include a moving average approach [108], graph embedding [78], or employing ML models with temporal learning capabilities [97, 108]. This thesis instead explores the representation of temporal information in data for anomaly detection training. With dynamics and user behaviour changes in mind, we keep the focus on detecting the changes in each user's most recent activities instead of the whole / averaging over the time range of data. Furthermore, instead of affixing to a single data granularity level, i.e. time period for detection, we explore different levels of granularity in data extraction, from hours to a week of user data, to allow comprehensive assessment and selecting the best setting for each application scenario.

Chapter 3

Overview of the Insider Threat Detection Framework

This chapter presents an overview of the insider threat detection framework proposed in this thesis. A general description of the components in the framework is also provided in this chapter.

3.1 Framework Overview

A framework for machine learning applications in detecting insider threats in corporate and organizational computer systems is proposed in this thesis. Figure 3.1 illustrates the components of the framework. The workflow is designed to be modular and easily expandable for a wide range of corporate environments, data acquisition conditions, as well as learning and analysis methods.

As presented in Figure 3.1, the framework is organized as follows:

1. Data collection: data from multiple sources are collected and stored in unified formats. The two main sources are:
    - User activities, e.g. network traffic, email, authentication, and process logs
    - Organization structure and users' information.
2. Data pre-processing: heterogeneous data from different sources is aggregated and processed to construct feature vectors representing user activities and profile information at different granularity levels. The extracted data can be further processed with temporal information in representation.
3. Unsupervised learning algorithms are employed for the initial detection step to detect any anomalies as possible indicators of insider threats.
4. Supervised and semi-supervised ML algorithms are then employed for data analytics based on the constructed feature vectors and limited ground truth.

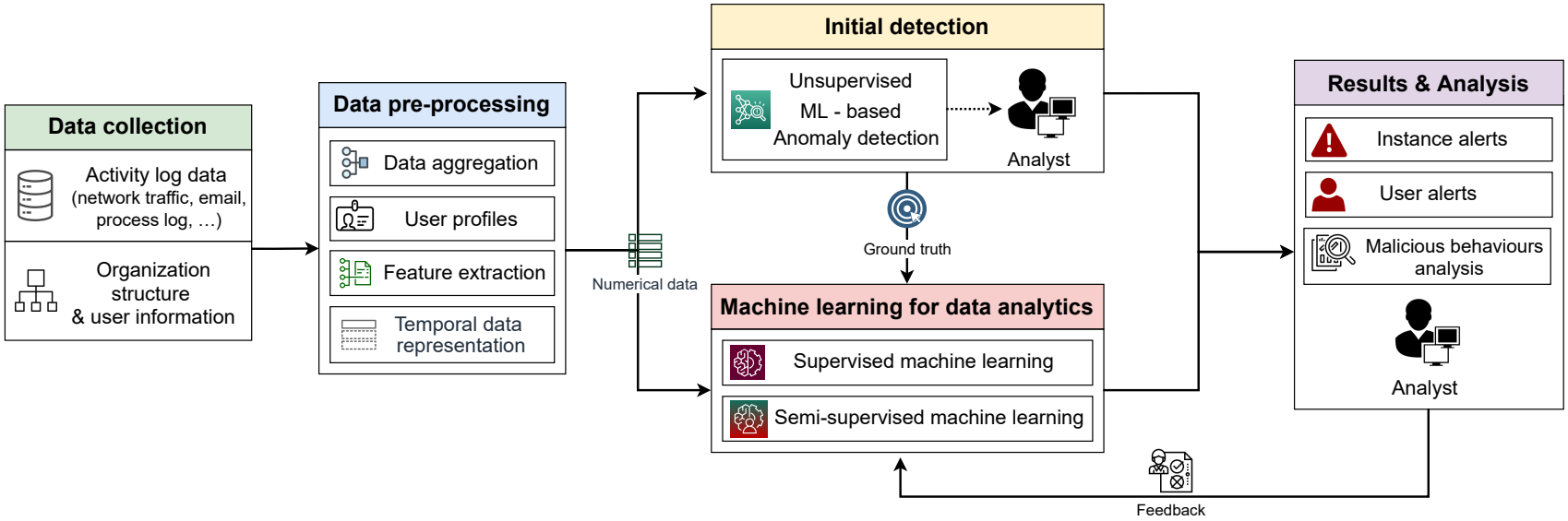


Figure 3.1: Overview of the proposed system

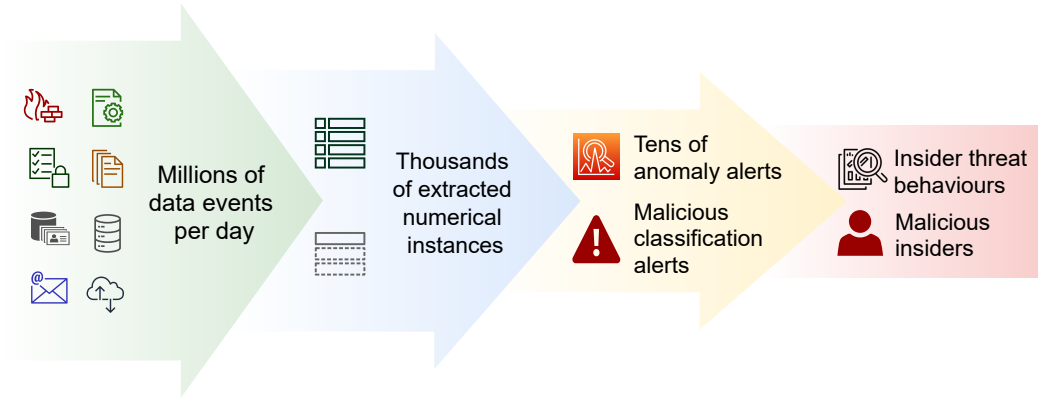


Figure 3.2: Data events to alerts and analysis

5. Results are presented in different formats, and detailed analysis is provided to the system analyst.

Figure 3.2 depicts the process of transforming from data events from heterogeneous sources to alerts and indicators of insider threat behaviours and malicious users by the detection system, based on the Model-based predictive classification concept [46]. The system is designed to operate with the participation / supervision of security analysts in many steps, especially in initial detection, where the early signs of malicious behaviours and abnormal activities are investigated. Human analysts play an important role not only in the analysis of system warnings and alerts, but also in performing the necessary actions to return the system to 'normal' operation after an attack.

In this thesis, we assume five benchmarking datasets: CERT datasets (R4.2, R5.2, and R6.2), LANL dataset, and TWOS dataset (Section 4.1). The data pre-processing steps are presented in Chapter 4. Data collected from different sources in corporate environments are aggregated and organized by user and time. The unified data is then used, together with user profiles, to extract numerical vector data that is suitable for machine learning applications. In this research, we explore different levels of extracted data granularity (Section 4.2.4) and representations of extracted data with temporal information (Section 4.3) for improving insider threat detection performance.

With the aim of detecting insider threat without or with very limited a priori knowledge available, ML algorithms are employed in the next steps with different

deployment conditions for anomaly detection and masquerader / malicious insider identification. In Chapter 5, unsupervised machine learning are techniques are used for anomaly detection. The assumption is that malicious behaviours are often rare and deviated from normal user behaviours, which constitute the vast majority of the collected data [13,62]. Thus, although no label information is used, a trained anomaly detection model may capture the normal data and reveal anomalous behaviours as outliers. An *investigation budget* (IB), which is the amount (%) of data that the security analyst can examine for confirmation of malicious behaviours, is used in the initial detection step to assess detection performances at different thresholds.

In Chapters 6 and 7, supervised and semi-supervised machine learning methods are trained with very limited amounts of ground truth for detecting unknown malicious insiders. Then, we explore how well the learned solutions would be able to generalize for detecting unknown malicious insider cases. The benefit of using supervised learning is that we need not assume that data clusters are always synonymous with distinct behaviours. This potentially leads to higher precision than unsupervised learning / anomaly detection algorithms [21]. Chapter 7 explores semi-supervised learning methods and different approaches to present a limited labelled training set for semi-supervised learning algorithms. Semi-supervised learning permits the large amounts of unlabelled data to be harnessed in combination with typically smaller sets of labelled data to improve the outcome [110]. This motivates its use in insider threat detection, as obtaining a fully labelled training dataset is prohibitively costly in many real-world conditions.

In addition, the analysis will distinguish between malicious actions detected and malicious users detected, where the two are not necessarily the same. That is to say, the diversity in a user's role within an organization can impact the number/types of actions performed, both normal and malicious. In many cases, user actions can vary over time and have multiple contexts that need to be taken into account in order to process an alert about a suspicious behaviour [112]. Thus, high malicious instance detection rates, in this case, may not necessarily translate to all malicious insiders being detected. Furthermore, a seemingly small false positive rate may still require a lot of attention from the security analyst if it flags many distinct normal users as anomalous. In short, we believe that results that highlight malicious users rather

than events represent a more important measure of system performance.

Finally, several measures are presented, such as detection delay per malicious insider, or the support for each malicious insider alert (Chapter 6). By providing these measures, we aim to provide better support to security analysts and enable successful application of the proposed system in real-world scenarios.

3.2 Summary

This chapter presented an overview of the proposed insider threat detection framework in this thesis. This includes steps from raw data collection to pre-processing and machine learning based insider threat detection, both unsupervised and supervised. The framework is designed with adaptability and participation of analysts in mind, in that each module can be adjusted to suit the deployment environment of an organization.

Chapter 4

Data and Pre-processing

Data collection and pre-processing are crucial for insider threat detection in particular but also for cyber-security tasks in general. A good monitoring / data collection procedure in combination with adequate pre-processing steps may significantly simplify the application of machine learning techniques[1] and support security analysts in making correct decisions. In this chapter, firstly we detail the publicly available data sources employed in this thesis in Section 4.1. The data extraction steps are presented in Section 4.2. In this section, we also discuss granularity levels of the extracted data 4.2.4. Finally, in Sections 4.3 and 4.4, we present temporal properties in data extraction and perform initial comparisons to illustrate the advantages of the data processing technique.

4.1 Data Sources

Obtaining data for designing and evaluating insider threat detection systems encounters additional challenges over typical difficulties observed in cyber-security. Insider threats typically involve corporations and government agencies, where such threats relate to compromising organizational intellectual properties. Furthermore, concerns about user privacy may limit the distribution of data. Therefore, insider threat test datasets are very rare [81].

This thesis employs three most related publicly available data sources of insider threats and similar cyber-security issues. The data sources cover different log types and activities, from high-level HTTP and email logs, organization structure, to low level anonymized process logs and mouse/keyboard captures. The datasets depict a wide range of insider threats, both of malicious insiders and masqueraders, and related cyber-security attacks, such as lateral movement.

[1] Identification of useful attributes and mapping categorical to numerical values are typical steps performed by a domain expert. Likewise, any feature construction activity would be initiated by domain experts. Even frameworks for 'AutoML' are as yet not effective at this task [107]

4.1.1 CERT Insider Threat Test Datasets (CERT datasets)

Popularly used in the literature, the CERT insider threat test dataset is publicly available for research, development, and testing of insider threat mitigation approaches [24,43]. The datasets simulate corporate environments, and consist of users' computer activities (log on/off, email, web, file and thumb drive connects), as well as organizational structure and user information in the form of a Lightweight Directory Access Protocol (LDAP) directory. The data is synthesized using a number of generation models, including communication and relationship graph models, asset graph and decoy models, topic models, behavioural models, and psychometric models. These models are used in the same form and scope as the normal data in order to generate the data as close to what is seen in the real world as possible [43]. There are a total of five insider threat scenarios simulated, ranging from data leaking, intellectual property theft to IT sabotage [24]:

1. *User who did not previously use removable drives or work after hours begins logging in after hours, using a removable drive, and uploading data to wikileaks.org. Leaves the organization shortly thereafter.*

2. *User begins surfing job websites and soliciting employment from a competitor. Before leaving the company, they use a thumb drive (at markedly higher rates than their previous activity) to steal data.*

3. *System administrator becomes disgruntled. Downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. The next day, he uses the collected keylogs to log in as his supervisor and send out an alarming mass email, causing panic in the organization. He leaves the organization immediately.*

4. *A user logs into another user's machine and searches for interesting files, emailing to their home email. This behaviour occurs more and more frequently over a 3 month period.*

5. *A member of a group decimated by layoffs uploads documents to Dropbox, planning to use them for personal gain.*

In this thesis, we mainly employ three releases: 4.2 (CERT R4.2), 5.2 (CERT R5.2), and 6.2 (CERT R6.2) of the datasets. The releases are generated with different

versions of the generation models, and simulate companies with 1000, 2000, and 4000 employees, respectively. Additionally, the data releases differ in the number of insider threat scenarios and the number of malicious insiders simulated: scenarios 1–3 with 70 malicious insiders in R4.2, scenarios 1–4 with 99 malicious insiders in R5.2, and all five scenarios with only one malicious insider each in R6.2. A summary of the CERT datasets is presented in Table 4.1.

Table 4.1: Summary of the CERT datasets

Release	Duration	#Users	Scenarios (#Mal. insiders)	#Log on/off	#Emails	#Web	#USB	#File Access
R4.2	72 weeks (01/02/2010 - 05/16/2011)	1,000	1 (30), 2 (30), 3 (10)	854,860	2,629,979	28,434,423	405,380	445,581
R5.2	74 weeks (01/02/2010 - 06/02/2011)	2,000	1 (29), 2 (30), 3 (10), 4 (30)	1,810,070	17,361,575	58,960,449	836,984	887,621
R6.2	74 weeks (01/02/2010 - 06/01/2011)	4,000	1 (1), 2 (1), 3 (1), 4 (1), 5 (1)	3,530,286	10,994,957	117,025,216	1,551,828	2,014,883

4.1.2 Comprehensive, Multi-Source Cyber-Security Events (LANL dataset)

LANL dataset[2] is another publicly available data source depicting comprehensive and *real* corporate log data. The dataset consists of 58 consecutive days of de-identified event data collected from five sources within Los Alamos National Laboratory's corporate, internal computer network [55]. The dataset contains anonymized real users' processes, network flows, DNS, and authentication logs. The authentication log includes Windows-based authentication events from both individual computers and centralized Active Directory domain controller servers [55]. Similarly, the process log records process start and stop events from individual Windows computers. Domain Name Service (DNS) lookups and network flow data are collected at internal DNS servers and key router locations. In total, the dataset represents 1,648,275,307 events for 12,425 users, 17,684 computers, and 62,974 processes. Furthermore, authentication events that present known redteam compromise events (attacking) are provided in the dataset, but without further information or other malicious activities. In the literature, LANL dataset is used for evaluating detection techniques for lateral movement [14, 116], in which threat actors move through compromised accounts and systems of the victim organization to gain access to the intended targets. Although ground truth is only provided for compromised authentications, malicious activities performed by the compromised accounts may also be recorded in the dataset. In this

[2] `https://csr.lanl.gov/data/cyber1/`

research, as the focus is on detecting malicious users, we employ only authentication and process logs of the LANL dataset, which contain user identity in logged activities. Due to limitations of the dataset in the second half, the first 30 days of the logs are employed for feature extraction and analysis.

4.1.3 TWOS (The Wolf of SUTD) Dataset

Lastly, the TWOS dataset[3] is employed in this thesis, by virtue of its unique log types and simulated environment of many competing companies. The dataset provides anonymized authentication, mouse, keystroke, email, and network captures from a student competition with the aim of emulating insider threats, by both masqueraders and traitors [49]. The competition comprises 24 participants in six teams over five days. In the provided logs, timestamps are only properly presented in authentication log, and mouse and keystroke captures. Therefore, we employ these logs from the dataset in this research. There are 97,147 Windows-based authentication events captured in the dataset. The keystroke capture is anonymized by grouping, in that alphabet and digit keys are presented only by their groups (`DIGIT, LEFT, CENTER, RIGHT`). Mouse activities are captured by button clicks or moving actions and the corresponding positions. In total, there are 1,849,244 and 27,128,418 keystroke and mouse events in the dataset.

The dataset is generated from a competition, in which students are grouped into six competing teams representing companies. The teams earn points not only by successfully acquiring customers but also by taking points / customers from each other. Masquerader actions are created by giving the teams access to a machine that belongs to a member of another team, while malicious insiders are stimulated by arranging periods of firing / hiring, in that fired participants are incentivized to steal the original team's data to benefit his new team [49]. In total, there are 12 instances of masquerading and one instance of possible traitor in the data. The label information is represented by time windows of 90 minutes each per masquerader case, and 2 hours for the traitor case.

[3] `https://github.com/ivan-homoliak-sutd/twos`

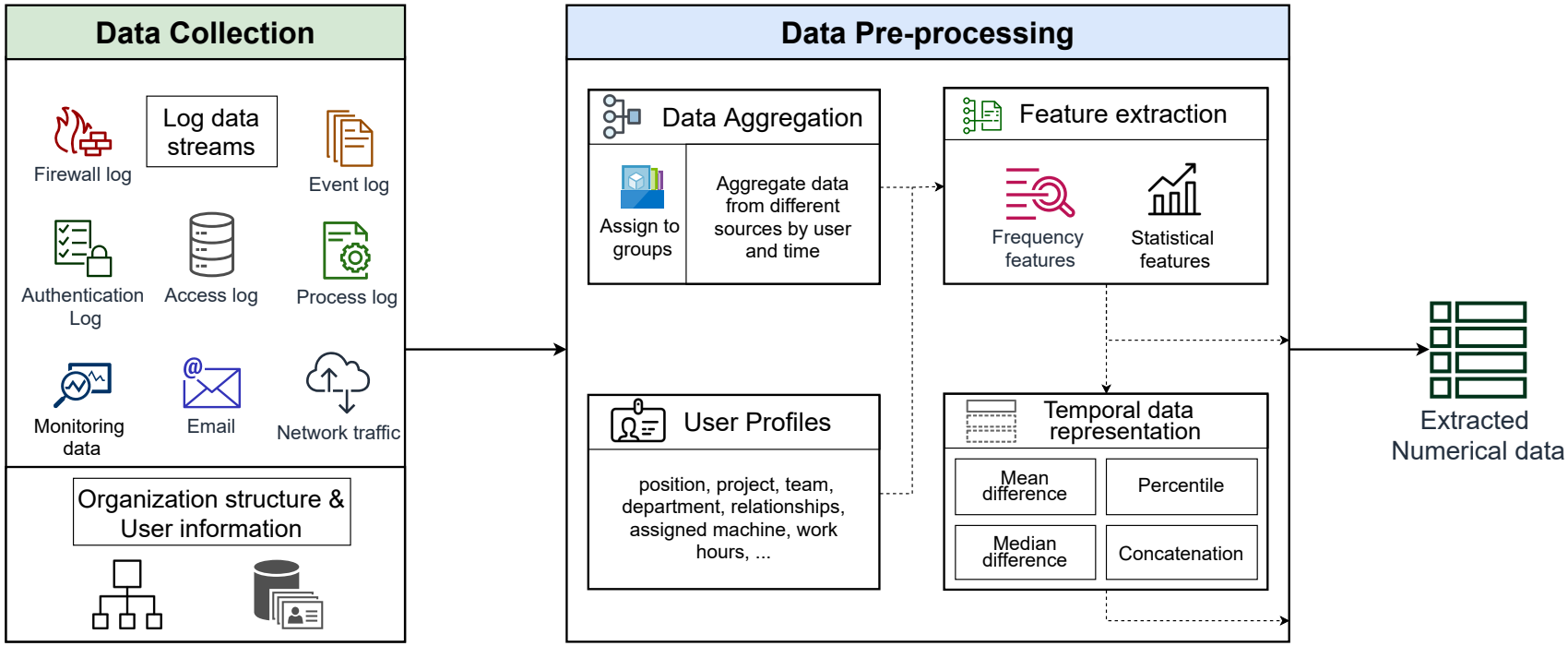


Figure 4.1: Data extraction process

4.2 Data Extraction

From the data sources, data pre-processing steps are performed to transform the data to suitable formats for the application of machine learning techniques. The data is first aggregated by a criterion, such as time and user. Then, depending on the available data sources, context models are created to support feature extraction. User-centered numerical features are then extracted to represent user behaviours for detection steps. The process is depicted in Figure 4.1.

4.2.1 Data Aggregation

As observed from the employed data sources and corporate monitoring data [8], in general, this research assumes that organizational data are collected in two main categories: (i) activity log data, and (ii) organization's structure and users' information. Data from the first category comes from different logging systems such as network traffic capture, firewall logs, email, web, and file access. These represent real-time sources of data that often need to be collected and processed in a timely manner in order to quickly detect and respond to malicious and/or anomalous behaviours. The second category of data represents background or context data, which can be employee information, role in the organization, relationships to other users. In many cases, this category also consists of more complex data, such as psychometric and behavioural models of users.

In certain conditions, this step may also be used to mask individual activities, such as website visited, into groups. For the CERT dataset, we define sets of websites (e.g. social network, cloud, job search) and file categories (e.g. document, compressed, executable) in corporate environments that are potentially helpful to insider threat detection. In doing so, not only numerical features can be extracted, but also privacy-preserving user monitoring can be facilitated, as specific websites and files that users visited, as well as their contents are not inspected in data pre-processing [57].

As shown in Figure 4.1, firstly, data from different sources are aggregated based on a pre-defined criterion (c), such as by user / computer and by time period (day / week) or the number of actions performed. In this research, we aggregate collected data mainly using user ID and by day or week. This is to achieve user-focused insider

threat detection. In Section 4.2.4, the time and number of actions performed are explored to generate different levels of granularity in extracted data. We also note that aggregated by time, the system is ready to process data as soon as possible, hence able to facilitate early detection of threats.

4.2.2 User Profiles

To assist data processing and feature construction, profiles are created for each user in the organization. The profiles consist of auxiliary information related to each user, such as assigned machines, relationships with other users, roles, work hours, permitted access and so on. The information in the profile is specific to each organization and data collection scenario. Inherently, the user profiles allow generating simple forms of graph features, where information captured by user-user and user-machine relationships are extracted.

In the CERT datasets, certain user information is provided in LDAP directory, such as the user's position, project, team, department, supervisor. For further enrichment of the profiles, we investigate the first weeks of the data to determine the user-machine relationship, i.e. identify each user's assigned machine, and shared machines. Similarly, work hours are empirically determined from the data. Figure 4.2 presents a histogram of activities by the hour of day in a workday in the CERT R4.2. From this, work hours can be selected as 8:00 – 17:30 for the CERT datasets.

Similarly, for LANL dataset and TWOS dataset, user profiles are created, depending on the amount of information available. In the case of LANL, user's domains, frequent processes and computers are determined for each user. In TWOS data preprocessing, user profile contains each user's team, assigned machine's IP, and user's role. Based on the user context models, feature vectors summarizing user's actions can be quickly and orderly created from incoming data.

4.2.3 Feature Extraction

From the aggregated data and user profiles, feature extraction can be performed to transform the data to a format that are suitable for training ML algorithms. This research employs *numerical vectors* for extracted data, in that each vector describes a user's activities and profile in a time period or in a computer session. As such,

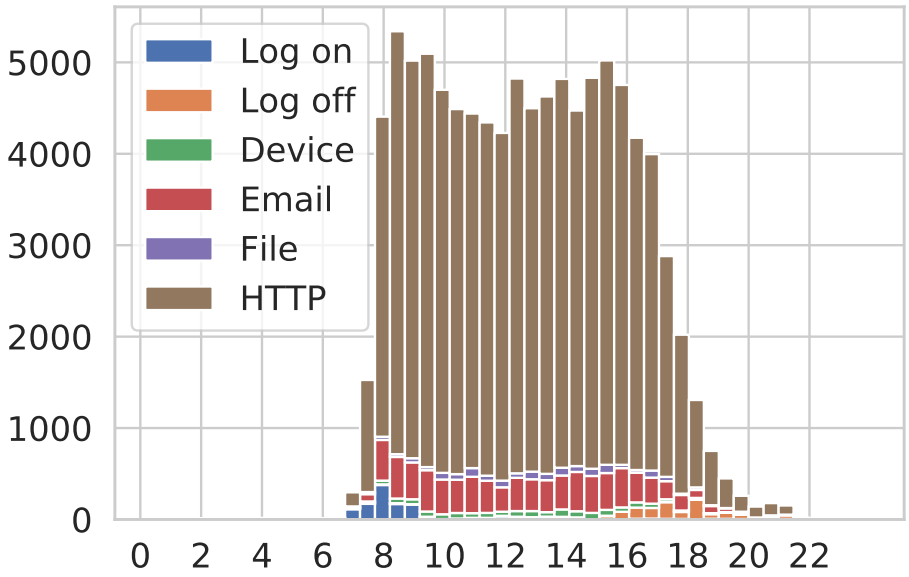


Figure 4.2: Histogram of actions by hour of day in a typical workday in CERT dataset

numerical vectors x_c, which are also called data instances, of fixed length d are generated. Each vector consists of user information – mostly categorical data encoded in numeric format for providing context for ML algorithms – and two types of features:

- Frequency features, which are the count of different types of actions the user performed in the aggregation period, e.g. *number of emails sent, number of file accesses after work hour*, or *number of websites visited on a shared PC.*
- Statistical features, which are descriptive statistics, such as mean, median, standard deviation, of data. Examples of data that are summarized in statistical features are email attachment sizes, file sizes, and the number of words in websites visited.

Fig. 4.3 demonstrates the feature creation process in the case of CERT dataset employed in this work. The process allows creating information-rich features consisting of many details, such as PC, time, and action specific characteristics. Up to three connected pieces of information presented in Fig. 4.3 are combined to generate a feature, such as the *number of actions on a shared PC, number of HTTP downloads after workhour, mean attachment size of sent emails.* Thus, the set of features

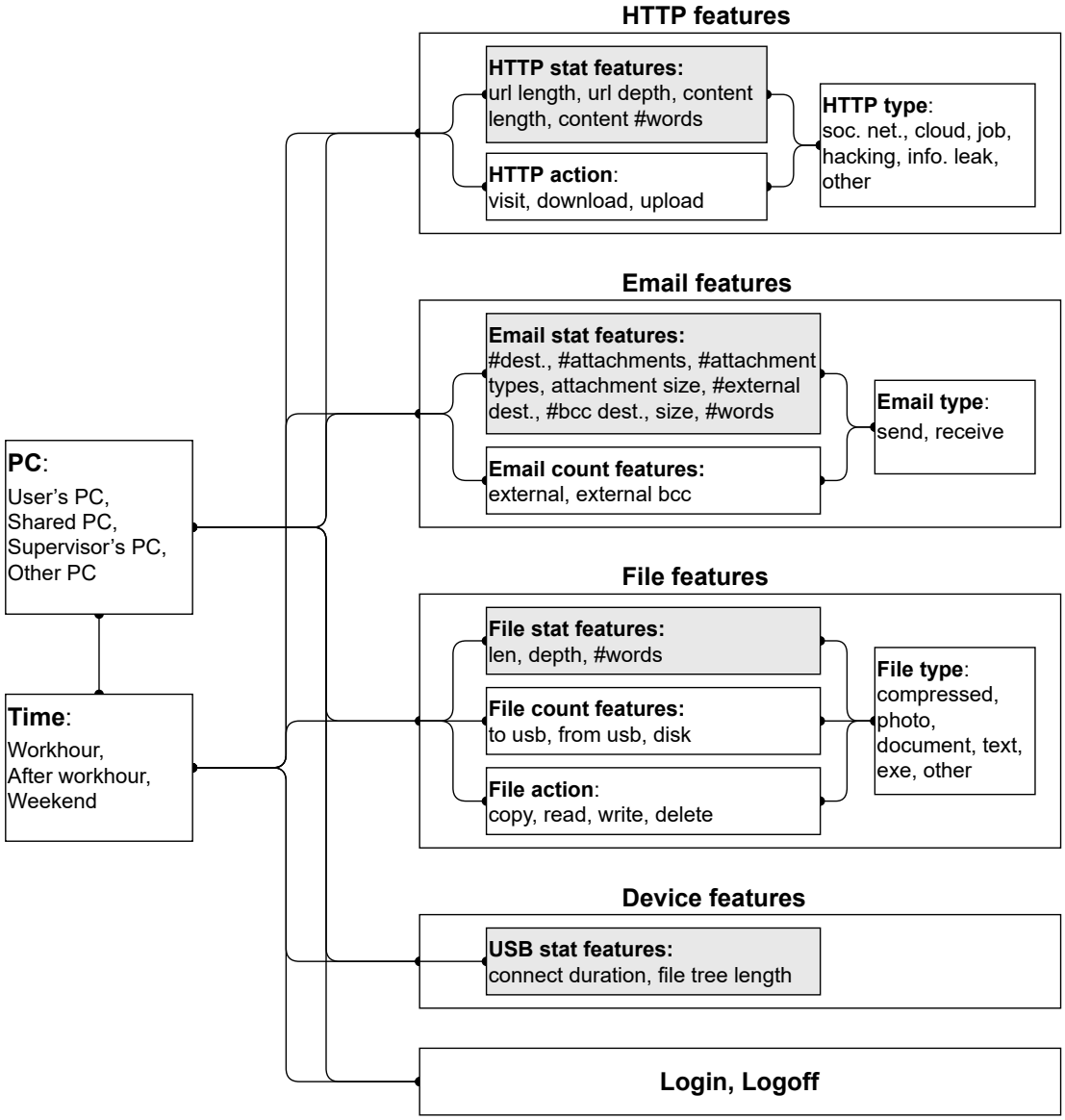


Figure 4.3: Illustration of the feature extraction process

constructed is essentially an enumeration over the pieces of information[4].

For LANL dataset, frequency features are created via counting user authentication and process logs, using different information provided in the dataset, such as *source/destination domains, source/destination PC, log on type*. We also augment the dataset by estimating sets of frequent processes and PCs for each user, and extract additional features from the sets. The work hour is assumed as 8:00 – 16:00 for the dataset. On the other hand, for TWOS dataset, as only authentication, keystroke and mouse data are employed (Section 4.1.3), counting features are extracted for actions in each group. The statistics of the duration of key presses in each key group and mouse button are also extracted. Due to the datasets' limited duration, data instances of a day and 30 minutes of activities are extracted for LANL and TWOS datasets, respectively. A similar time window has been successfully applied on TWOS dataset in [106], on a different log type (file system access events – not available publicly).

Table 4.2 shows the statistics of the extracted data in each type (by duration), and the number of normal and malicious users. It is apparent that the data distribution is extremely skewed in all extracted datasets. This reflects a real-world limitation, where insider threat events are extremely rare, and detection systems need to overcome the challenges in learning from highly skewed data of heterogeneous sources in order to distinguish malicious activities from the legitimate ones, where all are from authorized users.

Table 4.2: Summary of the datasets. Sc: Insider threat scenario. Malicious user counts are in parentheses.

Data	**Duration**	**Feature count**	**User count**	**Class distribution**					
				Normal	Sc 1	Sc 2	Sc 3	Sc 4	Sc 5
CERT	day	500	1,000	329,466	85 (30)	861 (30)	20 (10)		
R4.2	week	661	1,000	66,840	52 (30)	254 (30)	10 (10)		
CERT	day	824	2,000	1,692,342	85 (29)	863 (30)	20 (10)	339 (30)	
R5.2	week	1,092	2,000	139,572	49 (29)	235 (30)	10 (10)	248 (30)	
CERT	day	888	4,000	1,393,941	3 (1)	20 (1)	2 (1)	9 (1)	1 (1)
R6.2	week	1,176	4,000	283,205	2 (1)	4 (1)	1 (1)	7 (1)	1 (1)
LANL	day	1,215	11,814	229,691	Attack: 176 (98)				
TWOS	30mins	278	24	1,458	Attack: 38 (13)				

[4]Feature extraction code is made publicly available at https://github.com/lcd-dal/feature-extraction-for-CERT-insider-threat-test-dataset

Table 4.3: Data granularity levels

Data type	**Notation**	**Aggregation criterion** c
User-Week	$\mathbf{x}_w$	Week of user actions on all PCs
User-Day	$\mathbf{x}_d$	Day of user actions on all PCs
User-Session	$\mathbf{x}_s$	Session of user actions, from login to logoff on a PC
Sub-session Ti	$\mathbf{x}_{t=i}$	i hours of user actions in each session
Sub-session Nj	$\mathbf{x}_{n=j}$	j user actions in each session

Note that each data instance extracted is assigned an unique `id` (e.g. `session_id`), which refers to the corresponding actions (in log files) that were performed by the user. This allows the cyber-security analysts to further investigate ML based system's alerts by quickly accessing and evaluating the original course of actions that caused the alerts.

4.2.4 Data Granularity

In this research, we explore data processing techniques to enable the extraction of multiple levels of data granularity with rich details for data analytics[5]. Hence, in addition to extracting user data by periods of day / week as in the previous section, we further extract data with higher fidelity: by sessions of user's computer activities, and by time / action counts in each session. This is done on the CERT R5.2 dataset, where the data allows the extraction of sessions of user activities. By changing the aggregation criterion c, different levels of data granularity are generated. Table 4.3 summaries the data types extracted in this research based on different granularity levels.

As presented in the previous section, the basic data instances extracted by week / day summarize users' activities over the corresponding time period. These coarse-grained types of data provide a high-level overview of behaviours (in a day or a week) with a higher feature count than session and sub-session data. As such, they can potentially accelerate the learning process by lowering the amount of extracted data instances.

On the other hand, session data points provide higher data fidelity by capturing

[5]This section has been presented as a part of a paper published at the IEEE Transactions on Network and Service Management [71] ().

user actions on a PC, from Login to the corresponding Logoff; or from one Login to the next Login. Session-based data has utility for isolating malicious actions, since malicious users tend to perform malicious actions in particular sessions whereas other sessions in the same day or week may still be normal [52]. Furthermore, as the duration of a session is typically much shorter than a day, this data type may also allow quicker system responses when a malicious instance is detected.

As a session may last many hours and comprises hundreds of actions, we explore further the balance between the amount of data summarized in each data instance and potential system response time to malicious behaviours. This is done by using time duration and the number of actions performed as the criteria for separating a user-session data instance into sub-session data instances. This way, we control the amount of information embedded into each data instance. Thus, if the ML based system could successfully learn from the short-lived sub-session data to detect malicious behaviours, the response time of the system might be improved. As presented in Table 4.3, based on duration, a sub-session Ti data instance is created every i hours from the start time of a user's session on a PC. Similarly, a sub-session Nj data instance is created after every j actions by a user on a PC, starting from the Login action. The smaller i and j, the higher fidelity the data, but also the smaller amount of user activity information that is summarized in an instance.

Table 4.4: Summary of the extracted data. (Sc: Insider threat scenario)

	Data	**# features**	**Data distribution by class**				
			Normal	**Sc 1**	**Sc 2**	**Sc 3**	**Sc 4**
higher granularity ↓	$\mathbf{x}_w$	1,092	139,572	49	245	10	248
	$\mathbf{x}_d$	824	692,342	85	863	20	339
	$\mathbf{x}_s$	221	1,002,616	65	1,070	33	678
	$\mathbf{x}_{n=50}$	222	2,164,629	70	2,018	48	678
	$\mathbf{x}_{n=25}$	222	3,710,139	89	2,841	55	679
	$\mathbf{x}_{t=4}$	222	2,153,840	93	1,884	47	678
	$\mathbf{x}_{t=2}$	222	3,713,142	119	2,811	59	678
	# users by scenario:		**1901**	**29**	**30**	**10**	**30**

Fig. 4.4 shows the distribution of user-session data by the number of actions,

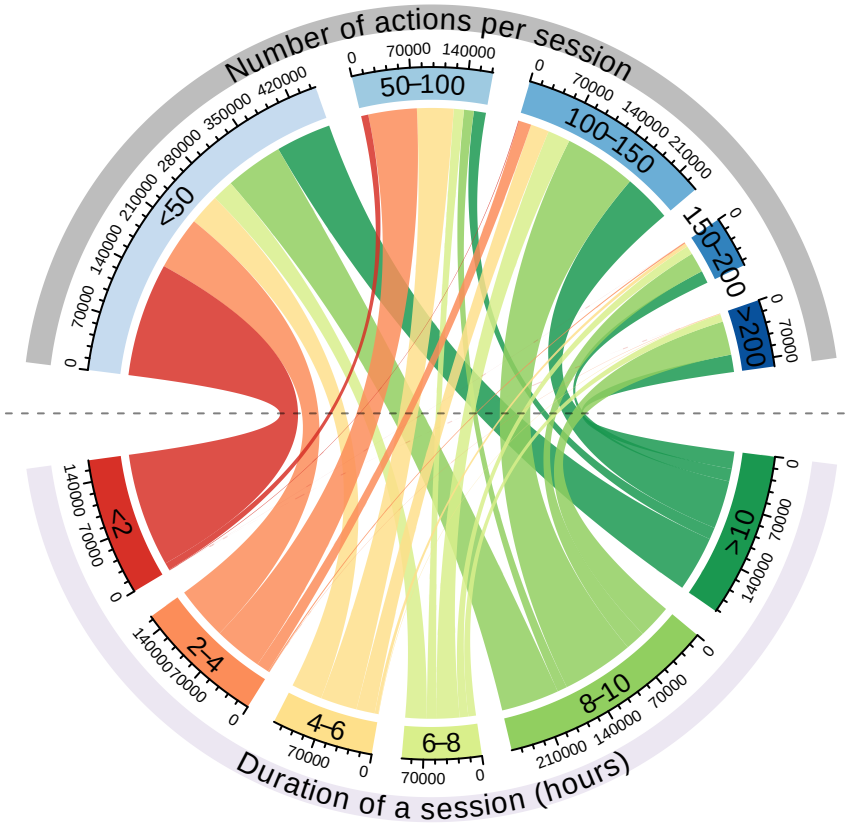


Figure 4.4: Relationship between the duration and the number of actions in user-session data

the duration of each session, and the relationship between the two features. It is apparent that the majority of user-session data has less than 300 actions, and more than half of the sessions has less than 100 actions. We therefore employ $j = \{25, 50\}$ for extracting sub-session Nj data. On the other hand, the duration of a session is closer to a uniform distribution, with a large proportion lasting longer than 8 hours. Moreover, as shown in Fig. 4.4, many sessions with less than 50 actions may last longer than 10 hours. Hence, we explore the values of $i = \{2, 4\}$ for extracting sub-session data by time. Table 4.4 provides an overview of the data types by granularity levels.

4.3 Temporal Representation of Extracted Data

Exploiting the fact that insiders are essentially regular employees before they start performing malicious actions [105], we propose data representation approaches using temporal information[6]. The goal is to highlight the trends / changes in user behaviour over time. This may potentially reveal the transitions in behaviours of malicious insiders. The approach performs concatenating or comparing a data point to a time window of the most recent data of the same user.

Using a window of time, the approach compares a user's activities to only his/her most recent and relevant behaviours. As concept shift and drift are likely in user behaviours [50], this may be more effective than normalizing all data instances of each user from the beginning. Furthermore, by processing each data instance via time window, the proposed approach is ready to be applied anytime a new data instance appears, which is critical in online stream learning. Additionally, we note that in contrast to extracting time series data from a time window, where all data points in the window contribute similarly to the output, in this research, the focus is on using the time window to define a baseline comparison for each new data instance. Table 4.5 lists the applicable temporal data representations with corresponding abbreviations for each dataset in this thesis.

[6]This section has been presented as a part of a paper published at the IEEE Transactions on Network and Service Management [69].

Table 4.5: Temporal representation abbreviations for each dataset

Temp. rep.	**CERT week**	**CERT day**	**LANL**	**TWOS**	**Description**
Concat. (C_γ)	C_2, C_3	C_2, C_3	C_2, C_3	C_2, C_3	γ is number of instances concatenated
Percentile (P_w)	P_{30}, P_{60}	P_7, P_{30}	P_7	P_1	w denotes the size of time window in days
Mean Diff. (E_w)	E_{30}, E_{60}	E_7, E_{30}	E_7	E_1	
Med. Diff. (M_w)	M_{30}, M_{60}	M_7, M_{30}	M_7	M_1	

4.3.1 Concatenation

Inspired by the use of shift register and taps for representing time in data for intrusion detection [54], we introduce data examples to anomaly detection algorithms as the concatenation of γ consecutive data instances of the same user (abbreviated as C_γ). The idea is to encourage the learning algorithms to construct comparisons/arithmetic operations between each user data instance and its previous records. In this data representation form, a data instance x_t at time t is adjoined with $\gamma - 1$ most recent instances to form a data point for anomaly detection:

$$x_t^{\text{concatenation}} = \text{concat}(x_t, x_{t-1}, x_{t-2}, ..., x_{t-\gamma+1}) \tag{4.1}$$

Essentially, this creates a data instance with γ times the number of features originally extracted.

4.3.2 Comparing to a Time Window – Percentile and Mean/Median Difference Representations

In order to explicitly include temporal information and reflect changes in user activities, we propose to represent data for anomaly detection via a function comparing each data instance x_t with a time window w leading to t. The procedure is summarized in Algorithm 1.

Each arriving data instance is compared with previous data instances of the same user in the most recent time window w to create percentile or mean/median difference representation. In this research, we set the window size w to 7 days, 30 days, or 60 days. This setting allows contrasting each day (week) of user's activity against the same user's activities in the full week (month) leading to it, where both weekdays and weekends are taken into account to provide sufficient information for comparison.

Algorithm 1: Calculating Percentile and Mean/Median difference representation of data

Input : x_t of user u, window size w

Output: x_t^{output}

construct a $n \times d$ matrix X of $x_{t-1}, x_{t-2}, ..., x_{t-n}$ of the same user u, based on w ; // d: data dimension

$x_t^{\text{output}} = [\ ]$;

for *feature* f *in* F **do**

  if *Percentile* **then**

    $f' = findPercentile(x_t[f], X[:, f])$;

  else if *Mean difference* **then**

    $f' = x_t[f] - \overline{E}(X[:, f])$;

  else if *Median difference* **then**

    $f' = x_t[f] - median(X[:, f])$;

  x_t^{output}.append(f');

4.4 Comparisons of Data Extraction Approaches

In this section, we perform initial an analysis on CERT R4.2 to compare the data extraction approaches. The data extraction approach presented in Sections 4.2 and 4.3 is compared against sequential data and time series data extraction. The data extraction approaches are compared in *unsupervised* machine learning for anomaly detection.

For the initial analysis performed here, we use LODA [95] on extracted numerical data in original format (Section 4.2) and in percentile representation (Section 4.3). Further details on LODA can be found in Chapter 5. The comparisons are performed on weekly data instances in different extracted formats. The anomaly detection results are compared using AUC metrics. Receiving Characteristic Curve (ROC) depicts the relationship between Detection rate (DR) and False Positive Rate (FPR) under different decision thresholds, and AUC (Area Under the Curve) summarizes ROC in a single numerical metric for comparison between models.

4.4.1 Compare against Sequential Data Extraction

Sequential data summarizes the sequence of user's actions over a period of time. In the simplest form, the data sequence consists of an ordered list of actions taken by a user. For example, in the case of the CERT dataset, the sequential data feature set is {`log on, log off, device connect, disconnect, file, email, http`}. This results in variable-length sequences of user actions. In this section, sequences representing user activities for the period of a week are extracted[7].

Hidden Markov Model (HMM) is employed to learn from the extracted sequential data to detect anomalous user action sequences that may indicate insider threats. HMM is a statistical Markov model in which the states are hidden [42]. Each hidden state emits a symbol in a set with probabilities before transitioning to a new state. This algorithm is particularly suited to model normal behaviours based on the extracted sequential data. In this work, a HMM is created for each user at the beginning of the training process to model the user's weekly action sequence. Then for each of the user's new weekly sequences, the user's HMM is used to calculate the log probability of the sequence. The sequence is flagged as an anomaly for further analysis if the log probability value is larger than a threshold. If the action sequence is not flagged, or the flag is cleared by an analyst, it is used in combination with the previous action sequence to train the user's HMM again. This approach is used for insider threat detection in [97]. In this section, we train HMMs on only the most recent user data (2 weeks) to be able to adapt to the shifts and drifts in the user's behaviours. The HMM is trained using Baum–Welch algorithm. The number of hidden states in HMM is set to 5 or 15.

Figure 4.5 presents the ROC curves and AUCs by LODA and HMMs on different extracted data types. Results show that the highest AUC is achieved by numerical data with percentile representation (AUC = 0.9). LODA on percentile data representation also achieves the best detection rate at all false positive levels. In addition to the detection results, the data extraction approach employed in this thesis also has advantages in deployment situations. LODA's results in Figure 4.5 are achieved using a single detection model trained once, while sequential data approach (HMM)

[7]Results in this section has been presented at the 2018 IEEE Security and Privacy Workshops [63] ().

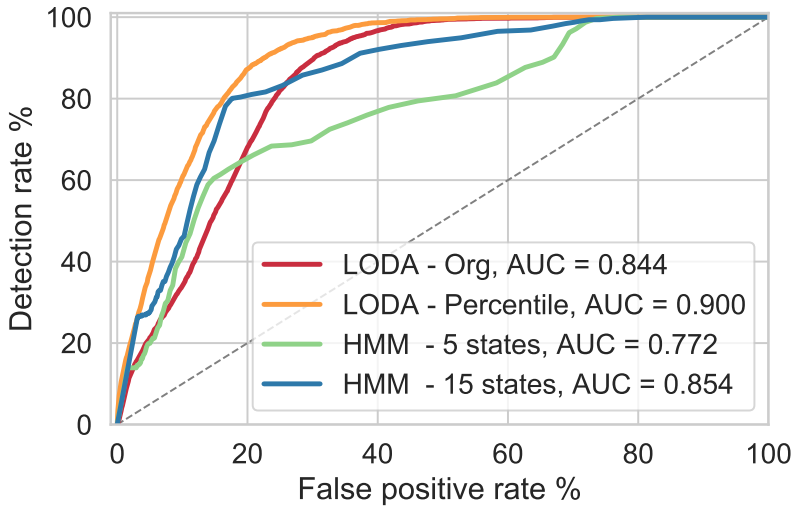


Figure 4.5: Comparison of anomaly detection results (ROC and AUC) by numerical and sequential data extraction approaches

requires a model for each user in CERT R4.2, and retraining the models every week. This process is time-consuming to train and evaluate over sequences of thousands of actions per user per week. Furthermore, sequential data structure for training HMM is incapable of carrying details representing user's actions that are valuable for anomaly detection, such as irregular log in time, downloading files from unauthorized machines, etc.

4.4.2 Compare against Time Series Data Extraction

In this section, we perform a comparison between the temporal data representation (Section 4.3) and time series feature extraction[8]. Time series features are extracted from CERT R4.2 day data with a rolling window size of 30, using `tsfel` package [9] with comprehensive extraction settings. Due to the computation overhead of the time series feature extraction process, a sample set of 200 randomly selected users in CERT R4.2 is used. Each feature in the original data is treated as a time series to extract 132 time-series features, using tsfel.

[8]Results in this section has been published as part of a paper at IEEE Transactions on Network and Service Management [69] (©2021 IEEE).

Results obtained show an AUC of 0.78 using `tsfel` time series extracted features. In comparison, LODA using original data and percentile representation generate AUCs of 0.81 and 0.87, respectively, under the same conditions. This shows the advantage of the proposed approach to traditional time series extraction approaches for temporal data in this application. We believe that by focusing on using the temporal window to define a baseline comparison for each new data instance, changes in user behaviours are easier to detect than from time series data via time windows, where all data points in the window contribute similarly to the output.

4.5 Summary

In this chapter, the employed data sources and the data extraction approach of this thesis are presented. The approach allows extraction of heterogeneous data into numerical feature vectors representing user activities in a time period, such as a day, and enables applications of popular ML methods. Different data granularity levels and temporal data representations are also introduced for further exploration of their potentials in assisting insider threat detection. Finally, preliminary comparisons of the data extraction approach to sequential data extraction and time series data extraction are performed to confirm the effectiveness.

Chapter 5

Initial Detection Step – Anomaly Detection

This chapter presents an unsupervised learning based anomaly detection approach for insider threat detection. We employ four unsupervised learning methods with different working principles, and explore various representations of data with temporal information. Furthermore, different computational intelligence schemes are explored to combine these models to create anomaly detection ensembles for improving detection performance. Evaluation results show that the approach allows learning from unlabelled data under challenging conditions for insider threat detection. Insider threats are detected with high detection and low false positive rates. For example, 60% of malicious insiders are detected under 0.1% investigation budget, and all malicious insiders are detected at less than 5% investigation budget. Furthermore, we explore the ability of the proposed approach to generalize for detecting new anomalous behaviours in different datasets, i.e. robustness. Finally, results demonstrate that a voting-based ensemble of anomaly detection can be used to improve detection performance as well as robustness. Comparisons with the state-of-the-art confirm the effectiveness of the proposed approach[1].

The chapter is organized as follows. Section 5.1 introduces the proposed anomaly detection approach. Section 5.2 presents the employed machine learning algorithms. Sections 5.3 and 5.4 detail the experiments and presents the evaluation results. Section 5.5 further discusses the results and makes comparisons. Finally, conclusions are drawn in Section 5.6

5.1 Anomaly Detection System for Insider Threat

Fig. 5.1 shows an overview of the proposed anomaly detection system for insider threat. From raw collected log data of user activities, the data is pre-processed to

[1]This chapter's content has been published at IEEE Transactions on Network and Service Management [69] (© 2021 IEEE)

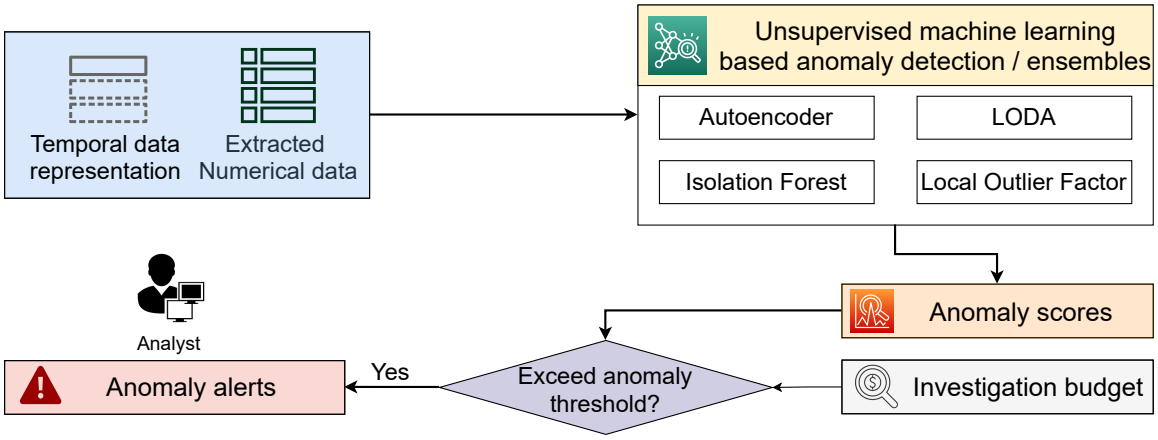


Figure 5.1: Components of the proposed anomaly detection system

extract numerical features by day or week, with different temporal representations (Chapter 4). The extracted data are then used to train anomaly detection models using unsupervised machine learning. The employed ML methods are described in Section 5.2. Post-training, anomaly scores are assigned by the detection model. Based on a user-selected *investigation budget*, a decision threshold can be calculated so that data samples with high anomaly scores (i.e. exceeding a threshold) are flagged for further investigation of possible malicious actions.

Using anomaly detection based on unsupervised learning, the assumption is that malicious behaviours are often rare and deviated from normal user behaviours, which constitute the vast majority of the collected data [2, 13]. Thus, although no ground truth is used, a trained anomaly detection model may capture the normal data and reveal anomalous behaviours as outliers.

Outliers identified by the anomaly detection model are defined by a threshold of anomaly scores, as demonstrated in Fig. 5.2. In this work, different thresholds are examined through changing the *investigation budget* (IB), which is the amount (%) of data – with the highest anomaly scores – that the security analyst can examine for confirmation of malicious behaviours [13, 67]. This represents the available human resources for analyzing the highest ranked data instances, post-training of the detection system, and performing the necessary actions in response.

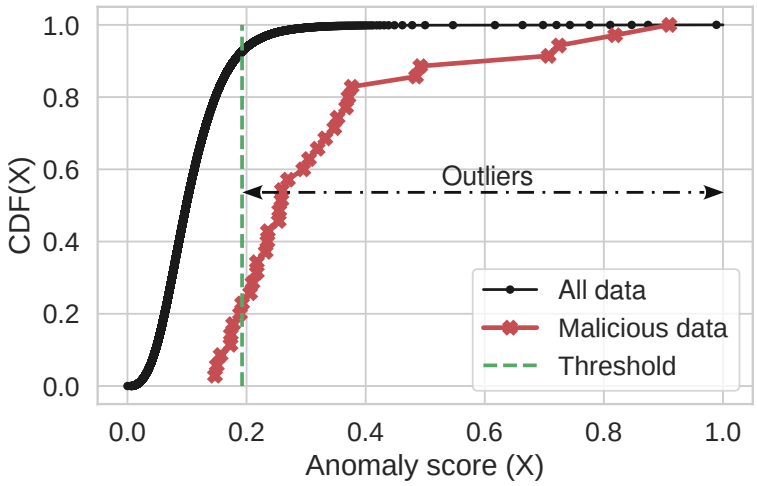


Figure 5.2: Demonstration of anomaly detection and threshold. CDF: Cumulative distribution function.

5.2 Unsupervised Machine Learning for Anomaly Detection

This research employs four popular ML methods for anomaly detection with different underlying concepts: Autoencoder (AE), Isolation Forest (IF), Lightweight On-line Anomaly Detection (LODA), and Local Outlier Factor (LOF).

5.2.1 Autoencoder

AE is a form of multi-layer neural network that compresses and reconstructs the data. Fig. 5.3 depicts an example of an AE with three hidden layers. The input and output layers both have d neurons (d: the number of dimensions). Each data dimension j in the input x is reconstructed into a corresponding dimension of r at the output layer by AE. By enforcing a "bottleneck" architecture through hidden layers (middle hidden layer size:h, $h \ll d$), AE compresses (encodes) the input data into h dimensions and reconstructs it at the output layer. AE is trained through minimizing the aggregated reconstruction error as the cost function:

$$E = \sum_{i}^{N} \sqrt{\sum_{j=1}^{d} (x_{ij} - r_{ij})^2}, \tag{5.1}$$

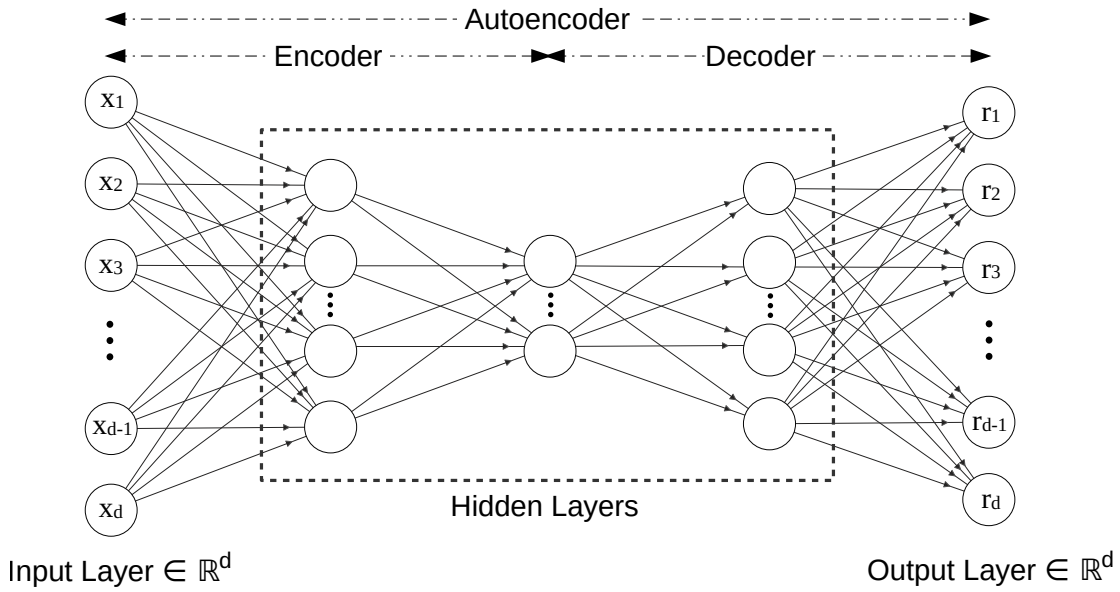


Figure 5.3: An example of an autoencoder

Post training, the lossy compression produced by AE essentially captures the lower-dimensionality representation of the majority of training data at the middle hidden layer. Assuming that normal user data constitute the majority of the training data, it is expected that AE shows a higher reconstruction error for anomalies [2], which may represent malicious insider behaviours. Thus, for each data instance x, the AE anomaly score is defined as the Euclidean distance between x and r: $e_i = \sqrt{\sum_{j=1}^{d}(x_{ij} - r_{ij})^2}$.

5.2.2 Isolation Forest

Based on the principle that anomaly examples are rare and significantly different in attribute-values from normal data points, IF [77] is designed as an ensemble of "isolation trees", whereas the anomalies – being easier to isolate – are assumed to be closer to the roots of the trees than normal instances. This is different from other anomaly detection methods, which build models of (mostly) normal data, and identify anomalies as any instances that do not conform to the model.

Each tree in IF works on a subset of training data and feature set. Binary splits are generated in each node of a tree by a randomly selected feature and split value.

The process is recursively repeated until each instance is isolated in a leaf. Having trained all isolation trees, the anomaly score of a data instance – $h(x) = \overline{E}(h_i(x))$ – is calculated as the average path length from root nodes to the corresponding leaves of the instance in the trees ($h_i(x)$).

Based on different principles from other outlier detection methods (such as AE), IF has been shown to possess some desired capabilities: To be able to deal with high dimensional data with irrelevant attributes, and able to be trained with or without anomalies included the training set [77]. These characteristics are evaluated in Section 5.4.

5.2.3 LODA – Lightweight On-line Detector of Anomalies

LODA [95] is an ensemble method combining weak histogram-based anomaly detectors into a strong detector. Similar to IF, each histogram anomaly detector in LODA works on a subset of input features in order to promote diversity. This is achieved through the use of sparse random projections $\{w_i \in \mathbb{R}^d\}_{i=1}^k$, where k projection vectors, each has $\sqrt{d}$ non-zero components, are created to approximate the probability density of input data into k one-dimensional vectors. Individual histograms are then calculated for each of k vectors. Each histogram shows an approximation of the original data distribution, which may reveal some aspects of outliers that come from a different distribution than normal data. Furthermore, in online LODA training, each histogram is updated by a training sample by projecting the sample onto a vector and then the corresponding histogram bin is updated.

To produce anomaly score for a data sample, LODA uses the average of the logarithm of probabilities estimated on individual projection vectors:

$$f(x) = -\frac{1}{k}\sum_{i=1}^{k} \log \hat{p}_i(x^T w_i), \tag{5.2}$$

LODA was shown to achieve comparable detection performance to more complicated algorithms, while significantly reduce time and storage complexity [95]. Furthermore, it is also able to operate and update itself in real-time online environment and on data with missing variables. In cyber-security, LODA's ability in identifying features of an outlier sample that deviates from the majority provides a useful tool to explain the causes of anomalous events detected.

5.2.4 Local Outlier Factor

LOF [19] is a popular anomaly detection algorithm, which proposes the concept of local density to identify anomalous data points. The local density measures how isolated a data sample is with respect to the surrounding neighbourhood. By comparing the local density of a data sample to that of its neighbours, LOF can identify data points that have a substantially lower density than their neighbours, which are considered to be outliers.

Considering k nearest neighbours to each data point, a k-distance(x) is defined as the distance of point x to its k^{th} neighbour, and $N_k(x)$ is the set of x's nearest neighbours. A reachability distance between two data points x and y is defined as reach-dist$_k(x, y) = \max\{k\text{-distance}(y), dist(x, y)\}$. The local reachability density of a data point x is then defined as the inverse of the average reachability distance based on $N_k(x)$ neighbours of x: $\text{lrd}_k(x) = 1 \Big/ \left(\frac{\sum_{y \in N_k(x)} \text{reach-dist}_k(x,y)}{|N_k(x)|} \right)$. Finally, LOF assigns anomaly score (i.e. outlier factor) of a data point x as average local reachability density of x's neighbours divided by $\text{lrd}_k(x)$:

$$\text{LOF}_k(x) = \frac{\sum_{y \in N_k(x)} \text{lrd}_k(y)}{|N_k(x)| \cdot \text{lrd}_k(x)}, \tag{5.3}$$

A value significantly larger than 1 indicate outliers, where the considered point has much lower local reachability than its neighbours. Despite the clear disadvantage in runtime, LOF has the capability to identify local outliers that could be skipped by other methods [19]. The algorithm also has been shown to perform well in cybersecurity domains [22].

5.2.5 Combination of Anomaly Detection Scores

Ensemble methods have been shown to reduce variance and bias of anomaly detection models in several applications [2]. In this section, we present unsupervised methods to combine results from the four aforementioned algorithms to create anomaly detection ensembles, in order to test their ability in insider threat detection. Employing four anomaly detection methods with different working principles, we expect to see differences in their detection results, especially under different conditions or scenarios. This creates potentials for improvements by combining the individual models. Specifically, we investigate combining schemes to create aggregated anomaly scores

based on the average / maximum of individual anomaly scores, or based on majority votes of individual algorithms:

- *Averaging (AVG)*: Anomaly scores of individual models are normalized by rank, i.e. percentile transformation. The combined score of a single data point is then computed as the average (mean) over the different scores of the point.

- *Maximum (MAX)*: This approach assigns the combined anomaly score to a data point as the maximum of normalized scores (by rank) reported by individual models. In essence, this combining scheme reports the highest anomaly signal (alarm) generated for a data sample by any of the participating models.

- *Voting (VOTE$^{\nu}$)*: In this scheme, a majority vote is used to select outlier data points at each investigation budget. The parameter $\nu \in \{2, 3, 4\}$ dictates how many votes are required to flag a data sample as anomalous.

5.3 Experiment Settings

In this chapter, to examine the anomaly detection performance under a wide range of data and conditions, CERT R4.2 and R6.2, LANL, and TWOS datasets are employed for experiments. As presented in Chapter 4, CERT R4.2 simulates a company with 1000 employees, where 70 are malicious insiders under three threat scenarios. This enables us to perform more flexible experiments in anomaly detection and provide a better understanding of the models' behaviours. On the other hand, CERT R6.2 depicts a much larger company with 4000 employees, containing only five malicious insiders (five threat scenarios, with only a single malicious user per scenario). This makes the detection task in CERT R6.2 much more challenging and realistic. On the other hand, TWOS and LANL present insider threat detection under a different simulated environment, and lateral movement detection, respectively (Section 4.1). The extracted data summary and abbreviations can be found in Tables 4.2 and 4.5.

For CERT datasets, this chapter assesses the detection performance on day and week data. More fine-grained data, e.g. session of user activities, could be used as well. However, that may not be beneficial in terms of utilizing human resources, as fine-grained data increases the data count, and thus raises the workload to inspect

anomaly alerts in the unsupervised anomaly detection setting, where false alerts are unavoidable [13].

5.3.1 Training the Anomaly Detection Algorithms

In training the anomaly detection algorithms, we randomly select a number of users n_u whose data in the first n_w weeks is included in the training process. Essentially n_u and n_w control the amount of data for training the models to represent computation and real-world limitations: Only a limited amount of data collected before the time of training can be used. In the following experiments, unless specified otherwise, we use training data of *randomly* selected $n_u = 200$ users (2000 for LANL data) in the first 50% of dataset duration ($n_w = 37$ for CERT and 2 for LANL). In the case of TWOS dataset, $n_u = 24$ and $n_w = 1$, due to the dataset's limitations. Since the training process is label-free (unsupervised), test results are reported on the entire dataset. The experiments are repeated 10 times in each setting, and the averaged results are reported.

The experiments are performed on compute nodes with Intel Xeon E5-2683v4 CPU and 125GB of RAM. We implemented the data pre-processing and analysis steps using Python 3. AEs are implemented using Tensorflow [1]. In this paper, each AE has three hidden layers, where the size of the first and the third hidden layers are set to $input\_dimension/4$, and the middle hidden layer's size is set to $input\_dimension/8$. On CERT and LANL datasets, the hidden layers and the output layer take the form of rectified linear [44] and sigmoid activation functions, respectively. On TWOS dataset, tanh activation function is selected. AEs are trained using Adam optimization [59] for 100 epoch each. Implementations from Scikit-learn [93] and PyOD [117] are used for IF, LODA, and LOF. For IF, the number of trees is set to 200, and 256 is used for max sample size. LODA is built with 400 histograms and $1/\sqrt{d}$ sparsity, while LOF's number of neighbours is set to 20. These parameter values are chosen empirically.

5.3.2 Performance Metrics

In this section, the insider threat detection performance is measured using ROC and AUC metrics. ROC (Receiving Characteristic Curve) depicts the relationship between Detection rate (DR) and False Positive Rate (FPR) under different decision

thresholds (i.e. different investigation budgets), and AUC (Area Under the Curve) summarizes ROC in a single numerical metric for comparison between models.

$$DR = \frac{TruePositive}{TruePositive + FalseNegative} \tag{5.4}$$

We also present DRs at critical IBs (see Section 5.1) for a better understanding of the performance at very low IBs.

Furthermore, *user-based* results are presented in this section in terms of alarms that are raised per user through aggregation of raw (*instance-based*) anomalous alerts [34, 71]. Specifically, a normal insider (user) is misclassified if at least one of his/her data instances is classified as "malicious". On the other hand, a malicious insider is identified if at least one of his/her malicious data instances is labelled as "malicious" by the detection system. Consequently, we have two sets of performance metrics: Instance-based (DR, FPR, AUC) and User-based (UDR, UFPR, UAUC).

To compare between multiple algorithms or data representations on multiple datasets, we perform Friedman test [39], which is a non-parametric statistical test. The null hypothesis of the Friedman test is that there are no significant differences between the variables. It is rejected when the test statistic exceeds the critical value of the significance level ($p = 0.05$). Using the average rank of a method on all datasets (R_j), the Friedman statistic is calculated as: $\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_1^k R_j^2 - \frac{k(k+1)^2}{4} \right]$, where N is the number of data points, and k is the number of methods. The statistic is distributed according to χ_F^2 with $k-1$ degrees of freedom [35]. If the null hypothesis is rejected, a posthoc test (Bonferroni-Dunn) is carried out to compare the algorithms by pairs, i.e. the corresponding average ranks differ by at least the critical difference. The critical difference (CD) of the posthoc test can be calculated based on k and N [35].

5.4 Anomaly Detection Results

Instance-based anomaly detection results with different IBs are presented in Tables 5.1 and 5.2. Figures 5.4 and 5.5 show instance-based and user-based ROCs on R4.2 week data and R6.2 day data with different temporal data representations. Table 5.3 presents user-based detection results on CERT R6.2 and LANL datasets. Note that the abbreviations for data representations are shown in Table 4.5.

Table 5.1: Instance-based anomaly detection results with different investigation budgets on CERT datasets. The unit of DR is percent (%). DR and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Data	Data type	Temp. rep.	DR @ 0.1% IB				DR @ 1% IB				DR @ 5% IB				DR @ 10% IB				DR @ 20% IB				AUC			
			AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF
CERT R4.2	day	Org.	2.45	0.10	2.64	0.91	8.50	1.60	7.71	4.06	26.20	13.72	14.57	7.41	47.41	40.48	36.97	10.33	74.70	70.56	70.93	15.98	0.854	0.830	0.830	0.521
		C2	2.68	0.03	1.39	1.18	9.34	0.84	5.49	3.60	24.68	11.89	12.84	8.72	46.07	38.81	33.43	14.97	77.03	73.98	77.13	28.30	0.866	0.843	0.852	0.636
		C3	2.23	0.02	0.68	1.04	8.81	0.45	4.43	3.79	21.92	11.39	10.62	10.52	43.37	39.87	28.88	17.56	72.82	79.21	77.72	32.07	0.853	0.855	0.851	0.667
		P7	2.00	0.71	2.26	1.84	7.85	3.59	7.85	8.96	35.75	24.20	32.37	35.67	63.02	52.71	55.63	57.52	87.60	84.63	78.73	77.95	**0.903**	0.882	0.870	0.859
		P30	2.31	0.72	3.55	1.33	12.08	3.66	12.30	9.25	36.15	23.82	36.14	30.37	62.39	54.99	59.27	52.22	88.43	86.59	81.18	77.47	0.902	0.890	0.882	0.858
		E7	3.48	0.58	2.25	2.05	10.57	3.44	9.55	4.84	20.86	15.97	19.47	12.67	33.55	37.87	32.59	20.14	55.36	70.18	55.64	32.67	0.788	0.835	0.783	0.612
		E30	3.41	0.52	2.64	1.2	10.82	3.31	10.52	4.93	21.15	15.09	21.89	10.66	35.37	41.00	35.56	16.05	59.22	77.38	62.68	27.52	0.797	0.859	0.814	0.590
		M7	2.62	0.61	2.64	1.35	9.58	3.52	9.31	4.18	20.40	18.58	19.95	10.97	32.14	38.23	34.76	16.50	54.92	66.54	57.24	27.47	0.771	0.819	0.781	0.604
		M30	2.10	0.54	2.75	1.43	8.31	3.82	10.39	4.27	21.99	22.10	22.52	10.64	36.94	43.71	39.55	15.98	59.57	72.49	63.27	27.29	0.786	0.841	0.808	0.623
	week	Org.	2.53	0.03	0.25	2.25	9.97	0.85	6.14	7.18	22.41	6.71	20.98	17.47	40.13	23.89	33.70	22.82	72.53	70.47	67.18	32.06	0.847	0.825	0.844	0.611
		C2	2.88	0.03	0.16	2.75	9.21	0.66	2.85	6.23	22.15	4.78	16.20	18.13	41.65	19.53	30.98	24.62	77.03	72.75	64.59	34.49	0.858	0.838	0.842	0.624
		C3	2.75	0	0	2.69	7.12	0.66	1.42	6.30	20.92	3.64	10.25	16.71	40.06	15.76	26.30	24.08	77.85	66.90	64.72	35.22	0.857	0.827	0.837	0.617
		P30	5.76	0.22	2.47	5.6	10.57	0.44	9.72	9.72	27.63	13.23	31.46	29.59	52.18	46.30	57.56	54.49	81.55	86.46	83.99	88.20	0.874	0.881	0.889	0.897
		P60	7.66	0.16	3.32	7.22	14.34	0.60	13.10	13.07	32.56	13.64	36.58	31.46	55.09	47.41	59.49	56.99	83.67	87.12	86.58	88.58	0.887	0.884	0.900	**0.901**
		E30	6.11	0.25	2.25	5.79	14.94	1.17	9.81	13.01	28.92	12.34	26.65	24.21	46.49	31.84	43.35	36.99	70.79	72.78	67.82	63.58	0.845	0.843	0.833	0.804
		E60	8.39	0.28	2.22	8.32	16.23	1.42	10.09	14.72	29.30	11.46	28.77	24.15	48.48	32.69	48.45	36.58	70	74.08	70.57	64.27	0.849	0.848	0.852	0.824
		M30	3.64	0.32	2.03	3.96	13.16	2.12	10.32	9.94	31.58	17.53	29.46	22.82	47.53	38.16	46.23	36.23	69.75	71.80	70.16	61.04	0.839	0.847	0.840	0.797
		M60	5.03	0.28	3.01	5.44	9.81	2.85	14.08	12.44	29.08	20.22	34.72	25.16	48.51	42.78	51.96	38.01	68.99	76.65	75.13	62.12	0.844	0.862	0.863	0.806
CERT R6.2	day	Org.	26.29	0	8.00	20.00	39.14	6.00	22.57	35.24	67.43	49.71	52.00	38.10	84.29	79.71	88.86	40.00	96.29	90.86	100	41.90	0.952	0.924	0.949	0.751
		C2	24.57	0	2.29	20.95	38.29	4.00	22.00	35.24	74.00	48.29	55.71	37.14	91.14	79.14	93.43	38.10	99.43	92.57	96.86	47.62	0.965	0.923	0.949	0.817
		C3	22.86	0	0.57	22.86	36.86	5.14	16.00	35.24	80.29	49.71	57.71	38.10	95.43	73.71	90.57	38.10	99.14	86.86	94.57	42.86	0.969	0.909	0.942	0.787
		P7	31.14	0.29	16.00	27.62	45.43	21.43	46.29	40.00	81.71	72.00	71.14	69.52	98.57	92.00	86.00	78.10	100	98.86	96.86	91.43	**0.977**	0.958	0.958	0.942
		P30	33.71	0	13.71	37.14	43.43	15.71	42.57	44.76	81.43	82.57	69.71	65.71	96.29	92.29	84.57	83.81	100	97.14	97.14	93.33	0.974	0.960	0.957	0.948
		E7	30.57	0	10.29	27.62	37.71	6.29	31.71	39.05	52.57	48.86	43.14	41.90	64.00	80.86	55.43	49.52	89.14	98.29	74.29	63.81	0.913	0.936	0.878	0.844
		E30	28.86	0	10	31.43	40.00	3.71	33.43	37.14	55.43	52.00	44.00	58.10	67.43	90.57	58.86	76.19	91.43	97.43	77.71	87.62	0.926	0.942	0.892	0.928
		M7	28.00	0.29	9.14	23.81	37.71	12.00	30.57	38.10	53.71	46.29	45.43	50.48	68.86	67.14	54.00	60.00	83.71	80.57	71.71	76.19	0.906	0.890	0.849	0.875
		M30	25.71	0.29	12.29	12.38	37.43	9.43	29.71	40.95	58.00	47.71	44.29	58.10	74.29	64.29	55.71	75.24	92.00	86.00	74.00	91.43	0.935	0.901	0.862	0.934
	week	Org.	38.00	0	0	38.67	63.33	0	8.67	64.00	75.33	10.00	42.00	74.00	94	44.67	82	78.67	98.67	87.33	99.33	86.00	0.973	0.870	0.933	0.925
		C2	33.33	0	0	34.00	57.33	0	4.67	60.67	72.00	14.67	41.33	70.67	91.33	44.67	76	75.33	99.33	82.00	94.67	82.00	0.967	0.857	0.926	0.901
		C3	32.00	0	0	34.00	50.67	0	1.33	53.33	71.33	15.33	40.67	64.00	88.67	34.67	78.67	70.67	94.67	74.00	90.00	78.00	0.959	0.837	0.915	0.881
		P30	40.00	0	9.63	33.33	66.67	0	39.26	66.67	85.93	17.78	74.81	91.85	97.04	52.59	96.30	100	100	85.93	100	100	0.981	0.889	0.970	**0.985**
		P60	40.74	0	2.96	32.59	66.67	0	51.11	66.67	78.52	12.59	74.07	85.93	99.26	42.22	94.81	99.26	100	90.37	100	100	0.979	0.881	0.971	0.981
		E30	30.37	0	0	31.11	63.70	0	10.37	63.70	70.37	6.67	50.37	74.81	82.22	21.48	65.93	83.70	96.30	74.81	86.67	99.26	0.958	0.829	0.909	0.966
		E60	24.44	0	0	25.19	66.67	0	6.67	66.67	71.85	3.70	51.11	75.56	85.19	20.74	70.37	90.37	99.26	77.78	91.85	100	0.966	0.837	0.925	0.972
		M30	33.33	0	0	35.33	60.67	0	9.33	62.67	74.67	10.00	41.33	80.00	85.33	35.33	62.67	92.00	98.00	76.00	88.00	98.67	0.961	0.842	0.908	0.971
		M60	35.33	0	0	38.67	61.33	0.67	7.33	60.00	71.33	13.33	37.33	76.00	82.67	42.00	64.00	92.67	98.67	86.67	97.33	98.67	0.962	0.864	0.913	0.969

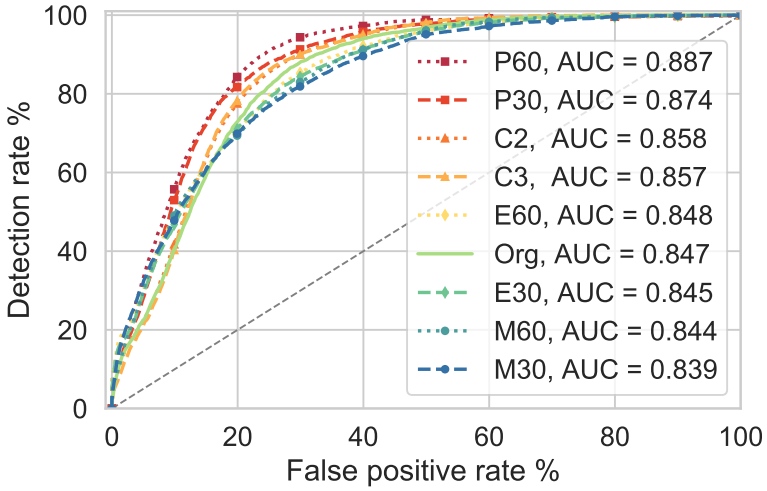
Table 5.2: Instance-based anomaly detection results with different investigation budgets on LANL and TWOS datasets. The unit of DR is percent (%). DR and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Data	Rep.	DR @ 1% IB				DR @ 5% IB				DR @ 10% IB				DR @ 20% IB				AUC			
		AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF	AE	IF	LODA	LOF
LANL	Org.	20.28	34.77	25.23	6.59	53.86	54.60	45.00	14.55	71.31	64.15	58.47	24.43	84.83	76.14	73.98	51.59	**0.908**	0.884	0.863	0.776
	C2	17.76	24.59	21.29	8.59	51.18	48.82	38.71	18.82	69.53	59.29	53.29	33.41	85.06	73.88	68.35	56.47	0.896	0.869	0.841	0.787
	C3	13.49	20.60	16.51	7.71	41.45	41.33	33.01	15.90	57.23	54.82	44.58	33.37	76.14	71.33	61.93	59.16	0.862	0.846	0.814	0.789
	P7	11.82	6.54	4.03	5.79	44.78	26.04	19.12	10.19	64.91	44.91	32.45	17.86	85.16	65.28	51.82	32.58	0.897	0.804	0.749	0.689
	E7	17.48	11.45	11.57	5.03	47.67	36.73	30.19	17.99	69.43	53.71	43.02	31.57	83.02	70.44	62.14	57.23	0.883	0.834	0.789	0.780
	M7	17.48	8.68	13.46	4.15	44.40	31.19	26.92	16.98	62.26	47.55	41.26	30.44	79.12	65.66	60.13	49.18	0.864	0.813	0.779	0.763
TWOS	Org.	2.63	0.26	0	0	13.42	18.95	8.95	7.89	23.42	29.47	17.11	15.79	42.89	44.47	34.21	34.21	0.714	0.708	0.577	0.632
	C2	0	0	0	0	5.53	10.53	8.68	10.53	14.21	19.74	13.68	15.79	33.42	37.89	26.58	31.58	0.641	0.673	0.591	0.613
	C3	0.79	0	0	0	5.26	5.79	4.21	10.53	10.26	11.84	9.47	15.79	19.47	28.16	21.32	23.68	0.599	0.634	0.555	0.565
	P1	0	0.26	0.53	0	16.32	12.11	16.32	2.63	40.53	27.89	34.21	2.63	55.26	57.63	54.47	18.42	**0.794**	0.780	0.777	0.578
	E1	7.37	1.32	2.11	2.63	11.05	13.68	13.68	7.89	18.95	22.89	21.84	10.53	40.26	39.21	38.16	31.58	0.709	0.716	0.691	0.669
	M1	4.47	4.74	3.42	5.26	8.16	17.63	13.68	7.89	20.53	36.05	20.26	13.16	39.21	44.21	37.63	36.84	0.708	0.737	0.671	0.692

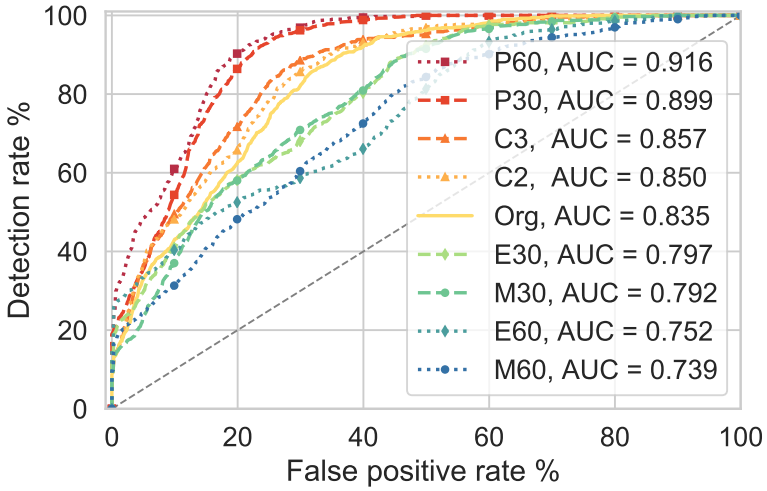
Overall, the results achieved using *autoencoder* and *percentile* representation are very promising, given that the results are obtained under an unsupervised setting with very limited training data (a small set of only 200 unidentified users in 37 weeks, for CERT data). On CERT R4.2, the approach was able to detect 80% of the malicious users by investigating only 1% of the most anomalous instances (1% IB). Also, at only 5% IB, nearly 100% of 70 malicious insiders are detected (Table 5.3). Figures 5.4 and 5.5 also show the differences in reporting results based on data instances and users, where the AUC achieved on user-based results could be higher and the differences between temporal data representations are more pronounced.

Table 5.3: User-based anomaly detection results with different investigation budgets on CERT R6.2 and LANL datasets. The unit of UDR and UFPR is percent (%). DR, FPR, and AUC results are color-coded based on different shades of green, red, and yellow, for easier comparison.

Data	Data type	Temp. rep.	0.1% IB								1% IB								5% IB								AUC			
			AE		IF		LODA		LOF		AE		IF		LODA		LOF		AE		IF		LODA		LOF		AE	IF	LODA	LOF
			UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR	UFPR	UDR				
CERT R6.2	day	Org.	1.66	54.00	3.15	0.00	1.92	40.00	2.55	46.67	18.78	78.00	13.17	36.00	21.56	60.00	28.60	60.00	39.37	100	34.44	60.00	40.94	80.00	47.70	66.67	0.937	0.774	0.863	0.768
		C2	1.11	50.00	2.19	0.00	0.83	12.00	2.10	46.67	13.44	66.00	8.00	18.00	10.20	58.00	25.56	60.00	34.76	90.00	24.62	60.00	31.54	80.00	46.91	60.00	0.935	0.754	0.883	0.752
		C3	0.93	50.00	1.73	0.00	0.54	2.00	1.55	46.67	10.55	64.00	6.27	20.00	7.30	58.00	20.45	53.33	29.68	88.00	20.25	44.00	26.09	80.00	43.20	66.67	0.935	0.755	0.904	0.752
		P7	1.56	54.00	2.68	2.00	2.41	56.00	2.48	53.33	14.66	80.00	11.12	44.00	20.91	80.00	19.30	73.33	31.33	80.00	27.14	78.00	38.56	80.00	38.05	80.00	**0.968**	0.904	0.955	0.962
		P30	1.48	60.00	1.78	0.00	2.10	56.00	2.65	60.00	16.29	80.00	8.98	38.00	22.00	78.00	20.95	86.67	34.74	94.00	25.19	80.00	40.29	80.00	41.46	100	0.954	0.891	0.946	0.975
		E7	2.76	56.00	3.45	0.00	5.67	50.00	1.97	53.33	28.07	70.00	14.06	36.00	32.60	60.00	17.67	73.33	44.55	100	33.56	80.00	46.96	82.00	37.75	93.33	0.905	0.843	0.796	0.941
		E30	2.62	62.00	2.33	0.00	5.51	44.00	2.14	60.00	26.12	78.00	11.49	22.00	32.36	60.00	19.74	73.33	44.62	100	32.05	60.00	47.44	82.00	39.24	100	0.936	0.809	0.817	0.968
		M7	2.87	56.00	5.11	2.00	6.25	54.00	2.20	53.33	26.64	72.00	22.60	42.00	32.27	60.00	19.51	73.33	44.31	100	41.44	80.00	46.03	80.00	40.71	100	0.895	0.841	0.817	0.943
		M30	2.15	58.00	4.57	2.00	6.03	56.00	0.40	53.33	22.91	74.00	21.82	40.00	31.66	60.00	14.71	93.33	42.67	100	41.47	80.00	46.71	80.00	40.49	100	0.936	0.849	0.821	0.969
	week	Org.	0.72	50.00	0.73	0.00	0.73	0.00	0.74	50.00	7.03	56.00	4.38	0.00	5.03	24.00	14.19	60.00	21.02	84.00	12.93	24.00	18.04	68.00	36.06	84.00	0.915	0.757	0.875	0.855
		C2	0.46	46.00	0.52	0.00	0.58	0.00	0.47	46.00	4.71	56.00	3.19	0.00	3.26	14.00	10.54	56.00	15.46	72.00	10.70	24.00	12.92	68.00	30.66	72.00	0.912	0.790	0.892	0.853
		C3	0.39	46.00	0.47	0.00	0.51	0.00	0.33	46.00	3.85	56.00	2.73	0.00	2.90	4.00	8.15	56.00	13.12	68.00	9.59	20.00	11.03	62.00	25.74	58.00	0.912	0.774	0.894	0.848
		P30	0.50	60.00	0.91	0.00	1.14	28.89	0.54	51.11	4.59	60.00	4.37	0.00	7.33	57.78	5.36	60.00	14.98	80.00	12.66	42.22	19.77	80.00	16.01	88.89	**0.949**	0.853	0.927	0.948
		P60	0.38	60.00	0.91	0.00	1.13	8.89	0.39	53.33	3.69	60.00	4.12	0.00	6.68	60.00	4.27	60.00	15.31	80.00	12.02	33.33	20.99	82.22	16.33	91.11	0.940	0.839	0.926	0.944
		E30	0.52	51.11	1.08	0.00	5.01	0.00	0.53	51.11	12.27	62.22	6.05	0.00	24.05	28.89	9.29	62.22	30.62	82.22	17.39	20.00	43.69	60.00	28.06	91.11	0.896	0.763	0.653	0.915
		E60	0.56	44.44	1.01	0.00	4.36	0.00	0.47	44.44	10.29	60.00	5.60	0.00	23.29	20.00	8.44	60.00	31.34	75.56	16.98	11.11	43.63	62.22	29.24	86.67	0.898	0.753	0.674	0.915
		M30	0.67	50.00	1.40	0.00	5.40	0.00	0.60	50.00	13.83	56.00	7.56	0.00	26.09	28.00	10.11	60.00	32.94	86.00	21.81	30.00	44.03	60.00	28.78	92.00	0.850	0.711	0.654	0.910
		M60	1.05	52.00	1.49	0.00	4.47	0.00	1.01	54.00	11.67	68.00	7.83	2.00	24.31	22.00	10.08	64.00	31.35	88.00	22.29	32.00	44.03	60.00	28.79	94.00	0.911	0.736	0.615	0.933
LANL	day	Org.	0.15	2.45	0.46	10.31	0.23	6.22	0.35	2.04	3.38	20.20	4.96	33.27	3.60	22.24	8.23	5.10	21.76	52.65	18.96	52.96	18.47	42.55	28.61	16.94	0.829	0.817	0.788	0.643
		C2	0.15	2.53	0.30	4.21	0.15	3.58	0.17	1.68	2.66	14.95	3.63	25.68	2.78	16.63	5.06	6.53	18.30	52.42	15.38	48.42	15.44	36.42	22.32	21.05	**0.849**	0.826	0.799	0.721
		C3	0.14	1.51	0.23	4.95	0.13	3.66	0.14	1.29	2.42	12.90	2.95	22.58	2.47	13.33	3.97	7.74	16.59	42.37	12.80	41.51	13.93	32.26	19.22	17.63	0.835	0.821	0.796	0.711
		P7	0.59	0.88	1.36	1.76	1.43	1.10	0.68	1.54	9.07	17.58	10.76	11.21	11.51	7.03	10.06	9.23	26.66	52.75	28.47	30.55	32.04	29.01	29.84	14.29	0.772	0.715	0.673	0.647
		E7	0.25	1.54	1.14	3.30	0.86	4.84	0.22	1.54	6.39	19.78	9.18	13.41	10.04	15.60	6.16	5.27	25.18	48.35	25.25	38.90	28.95	33.41	21.42	21.98	0.790	0.754	0.729	0.668
		M7	0.30	1.76	1.05	4.18	0.82	5.93	0.28	1.76	6.05	18.46	8.48	11.21	8.45	16.92	6.70	4.40	23.05	44.40	24.64	33.85	25.83	30.33	22.05	19.56	0.796	0.758	0.753	0.680



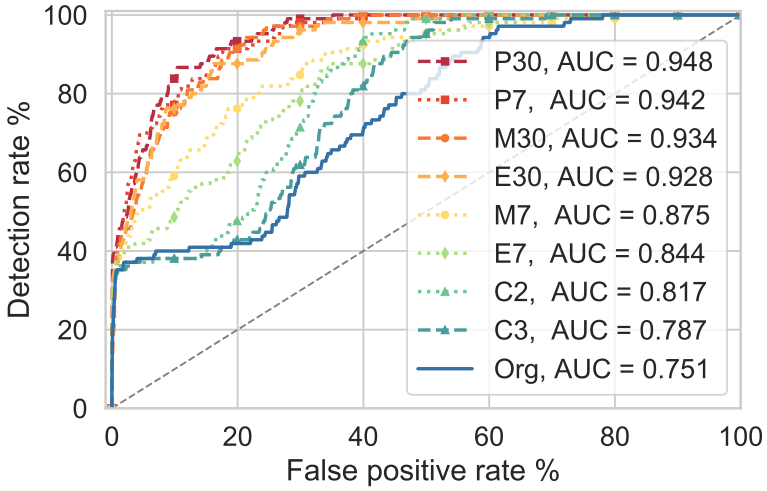
(a) Instance-based ROCs



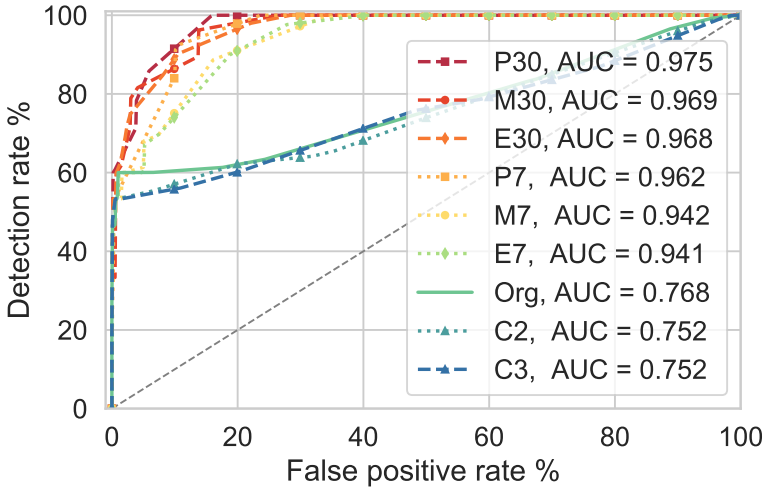
(b) User-based ROCs

Figure 5.4: ROCs of AE on R4.2 week data with different representations. C_γ: contatenation of γ data instances, $P/M/E_w$: percentile / median difference / mean difference data representation with time window w.

We note that normal data dominates the distribution in all employed datasets (Table 4.2). Thus, the FPR (normal data wrongly flagged) obtained under each IB is very similar to the IB, e.g. at 1% IB, FPRs range from 0.96% to 0.99% on CERT R4.2 week data. Furthermore, as IB represents different human resource levels for investigating anomaly detection output, i.e. different amounts of data flagged, a suitable IB can be selected based on deployment conditions. For example, on CERT R4.2 day data (Table 4.2), 1% / 5% / 10% IBs are equivalent to 3300 / 16500 / 33000 alerts, or approximately 7 / 33 / 66 alerts per day over the dataset's duration.



(a) Instance-based ROCs



(b) User-based ROCs

Figure 5.5: ROCs of LOF on R6.2 day data with different representations. C_γ: concatenation of γ data instances, $P/M/E_w$: percentile / median difference / mean difference data representation with time window w.

5.4.1 Results by Learning Algorithms

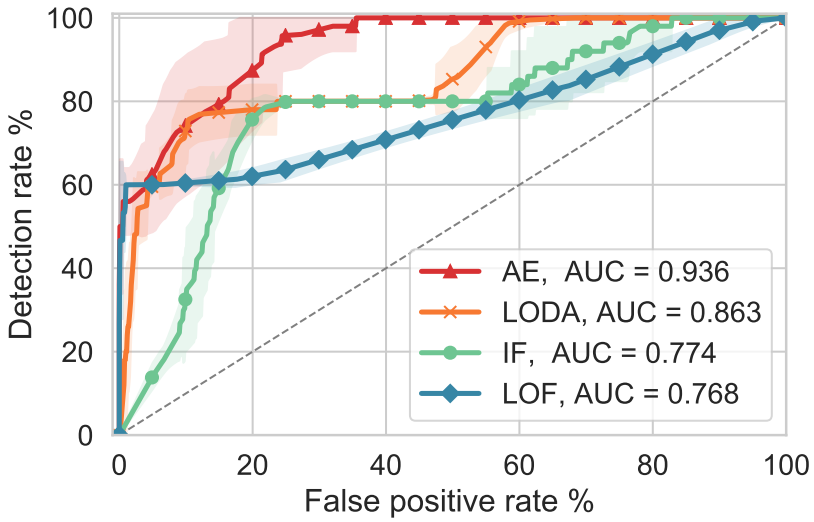
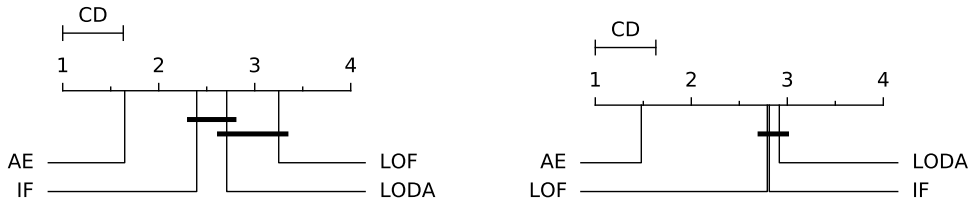


Figure 5.6: User-based ROC by learning algorithms on original R6.2 day data

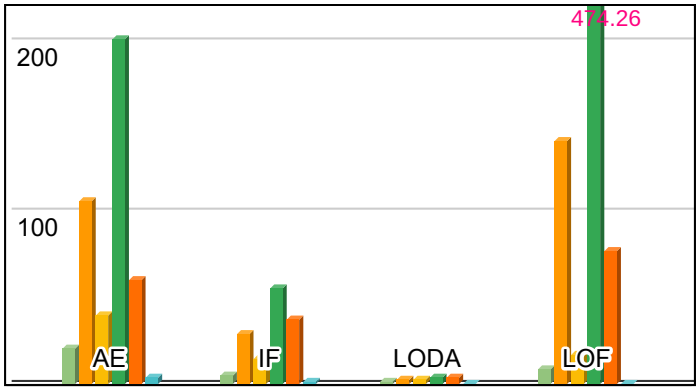


(a) Instance-based results

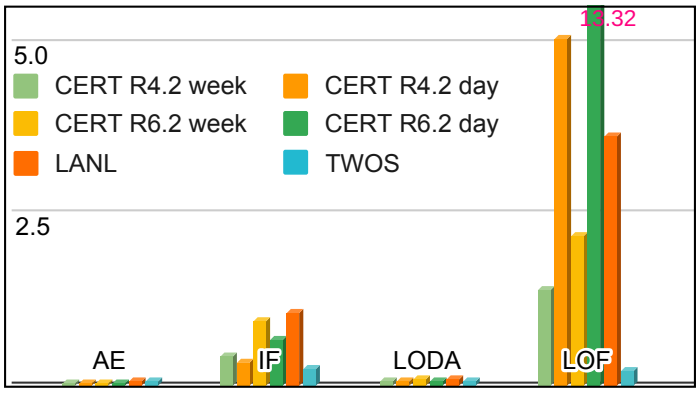
(b) User-based results

Figure 5.7: Critical Difference (CD) diagrams of algorithms' results by instance and by user. Average rank of each algorithm is shown on the scale. Two *linked* entries (connected by a horizontal black bar) are not significantly different, i.e. rank difference is less than CD.

Figure 5.6 shows a comparison of the algorithms by ROC. Performing Friedman test on both user-based and instance-based anomaly detection performance of the algorithms, the null hypotheses are easily rejected ($\chi_F^2 = 38.78, p = 2 \times 10^{-8}$ and $\chi_F^2 = 40.28, p = 9 \times 10^{-9}$), which means there are significant differences between the algorithms. Figure 5.7 presents the critical difference diagram obtained using the posthoc test, where the average ranking of each algorithm and whether they are significantly different are shown. Additionally, training and prediction times per data instance of each algorithm are presented in Figure 5.8.



(a) Training time (s)



(b) Prediction time per instance (ms)

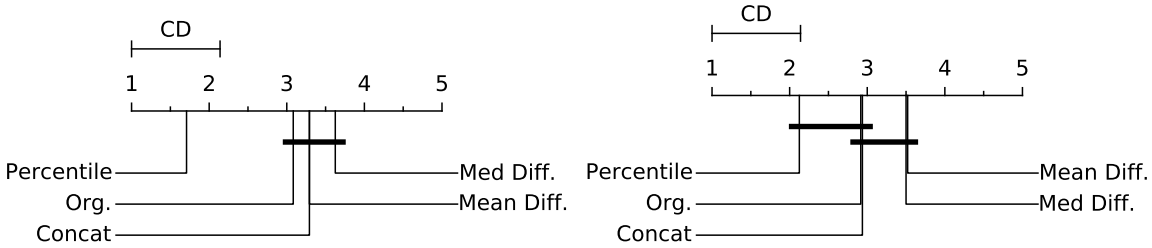
Figure 5.8: Average training time and prediction time per data instance of the algorithms on different data. Out of chart values are noted in red.

Overall, it is shown that AE achieves the best performance in detecting anomalies representing insider threats, especially at very low FPRs. For example, at only 0.1% IB, AE is able to detect 60% of the malicious insiders from R6.2 week data with P_{30} representation, while IF requires 8% IB to reach a similar UDR in the same setting.

LOF shows interesting results, where it performs well when data counts are lower (R4.2 and R6.2 week) and only on percentile representations. We believe that its ability to outperform in some cases is due to the "local" characteristics of its detected outliers, which may be missed by other algorithms (5.2.4). However, LOF suffers from very long training and prediction time. On the remaining two algorithms, LODA achieves very similar results to IF (Table 5.1 and Figure 5.7), and at very low time complexity. This makes it suitable for time-critical online detection tasks.

Experiments in Section 5.4.3 provide further insights into the detection performance of the algorithms. We note that the characteristics of the datasets (predominantly normal behaviours – Table 4.2), and experiment settings (Section 5.3) could be partly the reason to AE's outstanding performance in this section. On the other hand, Section 5.4.3 shows that LODA and IF can be more robust to changes in deployment conditions.

5.4.2 Results by Data Representations



(a) Instance-based results

(b) User-based results

Figure 5.9: Critical Difference (CD) diagrams of results by data representations.

On data representations, Table 5.1 and Figure 5.4 show that percentile (P_w) is the best representation of data for anomaly detection. This is confirmed by Friedman test (hypotheses rejected, $\chi_F^2 = 21.46, p = 0.0003$ and $\chi_F^2 = 12.48, p = 0.01$),

and posthoc tests, as shown in CD diagrams in Figure 5.9. Percentile representation allows the algorithms to achieve significantly better results than on the original data. Concatenation shows slight improvements in some cases, while mean/median differences are unable to surpass the original data. In some cases, such as R4.2 day data, mean/median difference even deteriorates the AUC (Table 5.1). Only on LANL dataset, percentile representation does not exhibit improvements in detection performance over original extracted data. This can be explained through the data extraction process, in that temporal properties are already incorporated in the extracted data via frequent processes/computers in user profiles (Section 4.2.2).

The observations suggest that percentile representation, although encoding the data change by omitting the absolute values, successfully captures the change in user behaviours while avoiding noises in the data. At the same time, maintaining the absolute values of changes as in mean and median difference representation seems to create noise and decreases the detection performance (Figure 5.4). Finally, on concatenated representation, the results show that it is hard to facilitate meaningful automatic comparisons between data related to different points in time.

5.4.3 Results on Different Conditions for Training Anomaly Detection Algorithms

In the following, we assume percentile data representation with window size of 30 days (P_{30}) and analyze ML algorithms on CERT R4.2 data types under different sizes of training data and conditions.

5.4.3.1 Anomaly detection performance under training data poisoning conditions

In this experiment, instead of using data from 200 randomly selected users, we deliberately introduce malicious users' data during training. The number of malicious users included varies from 0 (pure normal training data) to all 70 malicious users of CERT R4.2 (35% of training users are malicious). In an extreme case, we use only data of the 70 malicious insiders for training the algorithms. This is to analyze how the anomaly methods respond to data poisoning, where malicious data is presented at high density in training data, which may corrupt the ML models into mislabelling

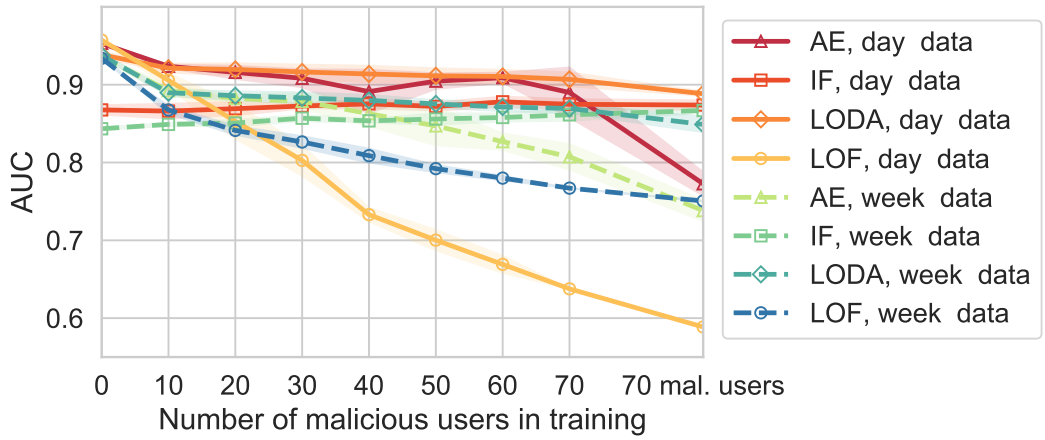


Figure 5.10: UAUC by number of malicious users in R4.2 training data

malicious as normal [10, 68].

Figure 5.10 shows the user-based AUC by the algorithms on R4.2 data under different numbers of malicious users in training. Overall, it is clear that ensemble-based algorithms, IF and LODA, are very robust to the data poisoning attack. Using IF, AUC even increases slightly with the presence of malicious data in training. This can be explained through its properties, where a small amount of contamination in training data allows trained IF trees to better model the anomalies that may appear in the data [77].

On the other hand, the performance of AE and LOF deteriorates as the number of malicious users in training increases. It seems that with high malicious data presence in the training set, AE may incorporate some malicious actions as normal in its trained model through the encoding-decoding process. Thus, it is unable to detect those types of behaviours in testing. Similarly, in the case of LOF, high amounts of poisoning data injected into training may increase the local density of malicious data points, which may trick LOF to assign lower anomaly scores to those points.

Nevertheless, AE was able to maintain a better performance than other algorithms, up to 30 malicious users in training data (15%). We note that in practice, the amount of malicious users in training data for insider threat detection approaches is typically very small [7], hence the use of AE is still preferred. Moreover, LODA shows a great balance between detection performance and robustness, making it a prime candidate in extreme poisoning conditions.

5.4.3.2 Effects of the number of users in training data

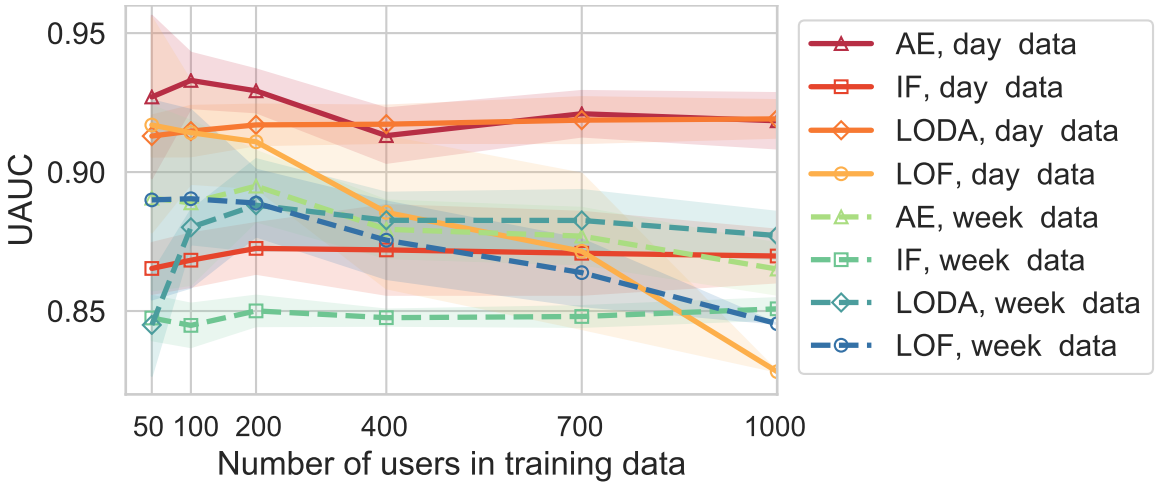


Figure 5.11: UAUC by number of users in R4.2 training data

Without having to collect ground truth for training data, the unsupervised learning approach permits the use of as many users in training data as possible, at the cost of a higher computational cost. In this experiment, we vary the number of (randomly selected) users to include in training data from 50 to 1000 (maximum amount) users in CERT R4.2. User-based AUCs are presented in Figure 5.11. Results show that except LOF, in most cases, the performance is largely unchanged. However, results vary more (i.e. unstable), when fewer data (less number of users) are used in training. LODA and AE's UAUC increase slightly to 200 users in training data, but AE's performance decreases slowly as the number of users increases in training.

Behaviours of AE and LOF can be explained through results in 5.4.3.1, where a larger number of training users creates a higher chance of malicious users to be included in training data, hence lowering their effectiveness. This shows that maintaining a relatively small number of users (200) in training data not only reduces the computational cost but also potentially gives more robust results.

5.4.3.3 Effects of training data duration

Similar to the number of users in training data, the number of weeks can be adjusted, too. This experiment varies the parameter from 7 (10% of data time range) to 74

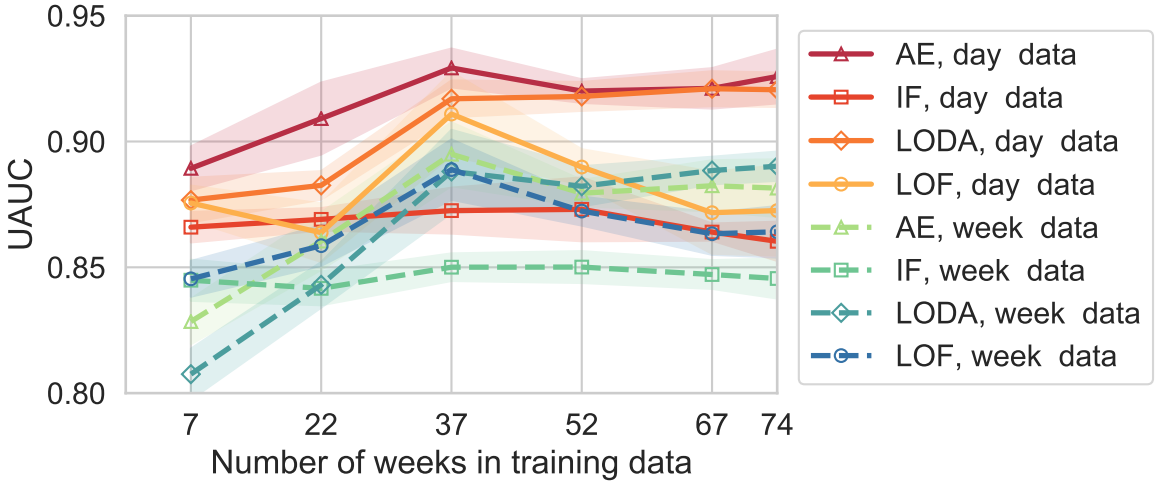


Figure 5.12: UAUC by number of weeks in R4.2 training data

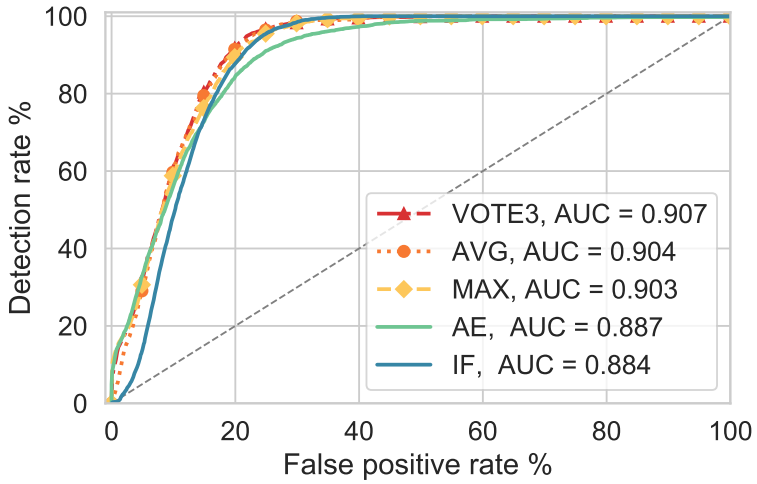
(100%). Figure 5.12 shows user-based AUCs on CERT R4.2 data types. As in the previous experiments, IF's performance is maintained through different numbers of weeks used in training data. On the other hand, the detection performance of AE and LODA rises until about 50% of the data duration is used in training (37 weeks), then remains largely unchanged. LOF shows similar improvements in the first half of data duration, but quickly deteriorates after that. In fact, more malicious insider activities appear in the second half of CERT data than in the first half [24]. Hence, it can be concluded that for AE and LOF, more training data may help to improve results, but only to a point where the improvements are negated by the introduction of malicious samples in training data (5.4.3.1). This experiment shows the advantage of online learning methods, such as LODA, where results can be progressively improved over time with more training data.

5.4.4 Ensembles of Anomaly Detection Models

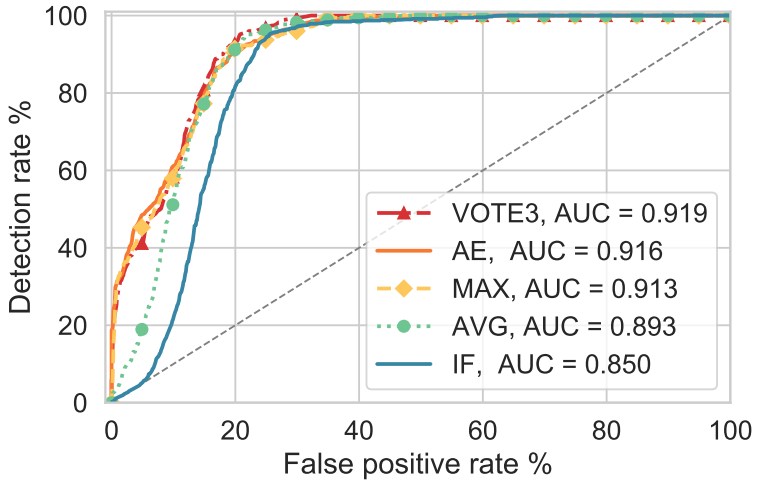
As shown in previous sections, four anomaly detection algorithms show various effectiveness on the datasets, especially under different training conditions. This section presents the results by the ensemble schemes described in Section 5.2.5. Table 5.4 and Figure 5.13 show the results on CERT datasets by AUC and ROC. Combining the anomaly scores to create ensembles, the best results (measured by AUC) by individual unsupervised ML algorithms are maintained in almost all cases, by VOTE$^{\nu}$

Table 5.4: Instance-based and User-based AUCs by the combination schemes on the CERT datasets. The results are color-coded based on different shades of yellow, for easier comparison.

Data	Data type	Temp. rep.	Instance-based AUC					User-based AUC				
			AVG	MAX	VOTE2	VOTE3	VOTE4	AVG	MAX	VOTE2	VOTE3	VOTE4
CERT R4.2	day	Org.	0.848	0.771	0.843	0.852	0.758	0.863	0.774	0.871	0.881	0.805
		C2	0.860	0.791	0.856	0.867	0.803	0.881	0.800	0.871	0.874	0.868
		C3	0.862	0.802	0.856	0.863	0.809	0.883	0.791	0.868	0.875	0.869
		P7	0.905	0.897	0.901	0.904	0.885	0.901	0.889	0.907	0.910	0.888
		P30	**0.910**	0.897	0.904	0.909	0.894	0.901	0.915	0.924	0.926	0.886
		E7	0.814	0.761	0.809	0.816	0.757	0.876	0.780	0.828	0.881	0.857
		E30	0.840	0.769	0.833	0.840	0.778	0.871	0.801	0.855	0.895	0.847
		M7	0.806	0.748	0.800	0.804	0.763	0.865	0.782	0.810	0.865	0.824
		M30	0.833	0.771	0.822	0.825	0.796	0.854	0.793	0.806	0.871	0.816
	week	Org.	0.845	0.774	0.848	0.851	0.770	0.857	0.755	0.836	0.846	0.851
		C2	0.854	0.794	0.851	0.856	0.793	0.866	0.772	0.849	0.863	0.831
		C3	0.839	0.785	0.843	0.846	0.758	0.849	0.757	0.848	0.858	0.806
		P30	0.894	0.895	0.896	0.896	0.881	0.882	0.895	0.900	0.893	0.860
		P60	0.904	0.903	0.905	**0.907**	0.891	0.893	0.913	**0.919**	**0.919**	0.868
		E30	0.854	0.836	0.849	0.854	0.842	0.741	0.689	0.765	0.755	0.745
		E60	0.866	0.849	0.864	0.865	0.854	0.691	0.669	0.728	0.687	0.717
		M30	0.855	0.834	0.851	0.857	0.840	0.718	0.652	0.743	0.742	0.722
		M60	0.872	0.851	0.867	0.872	0.859	0.706	0.705	0.728	0.693	0.727
CERT R6.2	day	Org.	0.950	0.932	0.955	0.956	0.911	0.945	0.896	0.950	0.941	0.897
		C2	0.956	0.946	0.961	0.963	0.922	0.896	0.884	0.938	0.940	0.826
		C3	0.941	0.948	0.959	0.960	0.883	0.884	0.887	0.935	0.941	0.799
		P7	0.977	0.971	0.976	0.976	0.967	0.969	0.960	0.971	0.970	0.951
		P30	0.978	0.971	**0.977**	0.976	0.971	0.961	0.972	0.965	0.960	0.949
		E7	0.920	0.918	0.910	0.914	0.904	0.947	0.896	0.894	0.939	0.937
		E30	0.961	0.943	0.951	0.949	0.955	0.954	0.919	0.914	0.949	0.944
		M7	0.910	0.893	0.902	0.910	0.898	0.939	0.932	0.890	0.921	0.928
		M30	0.945	0.940	0.940	0.929	0.934	0.946	0.931	0.935	0.922	0.927
	week	Org.	0.967	0.963	0.973	0.955	0.943	0.897	0.857	0.923	0.902	0.837
		C2	0.957	0.956	0.968	0.946	0.921	0.909	0.864	0.923	0.915	0.857
		C3	0.945	0.948	0.960	0.940	0.899	0.904	0.858	0.927	0.920	0.842
		P30	0.966	0.979	0.980	0.977	0.935	0.920	0.938	0.947	0.942	0.880
		P60	0.961	0.977	0.979	0.975	0.924	0.909	0.939	0.943	0.941	0.858
		E30	0.932	0.953	0.957	0.942	0.885	0.831	0.838	0.900	0.854	0.766
		E60	0.937	0.961	0.967	0.942	0.893	0.850	0.832	0.892	0.857	0.761
		M30	0.936	0.963	0.966	0.938	0.891	0.821	0.804	0.873	0.840	0.768
		M60	0.931	0.935	0.945	0.930	0.909	0.840	0.842	0.907	0.836	0.783



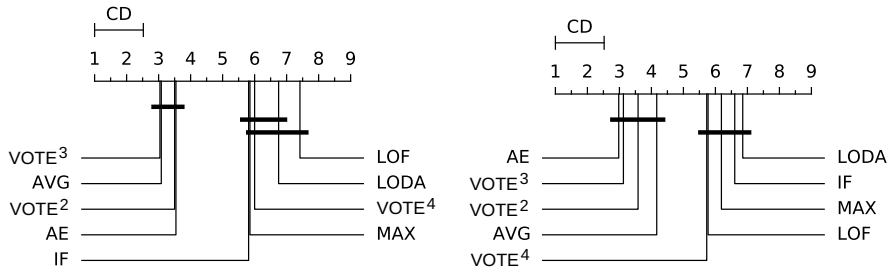
(a) Instance-based ROCs



(b) User-based ROCs

Figure 5.13: ROCs by combination schemes and individual learning algorithms (AE, IF) on CERT R4.2 data with P30 representation.

and AVG. In some cases, ensembles increase the detection performance. For example, VOTE^3 achieves AUCs of 0.909 and 0.907 on CERT R4.2 day and week data, respectively, which improves over the individual components (Table 5.1). Performing Friedman test, hypotheses rejected on both AUC and UAUC comparisons between the learning algorithms and ensemble schemes ($\chi_F^2 = 148, p = 4 \times 10^{-28}$ and $\chi_F^2 = 120, p = 2 \times 10^{-22}$). Figure 5.14 shows critical difference diagrams for the posthoc tests on instance-based and user-based results. On the other hand, as shown in the figure, there are no significant differences detected between AVG, VOTE^{2,3}, and AE, and these methods all significantly outperform the remaining algorithms (VOTE^4, MAX, IF, LODA, LOF).

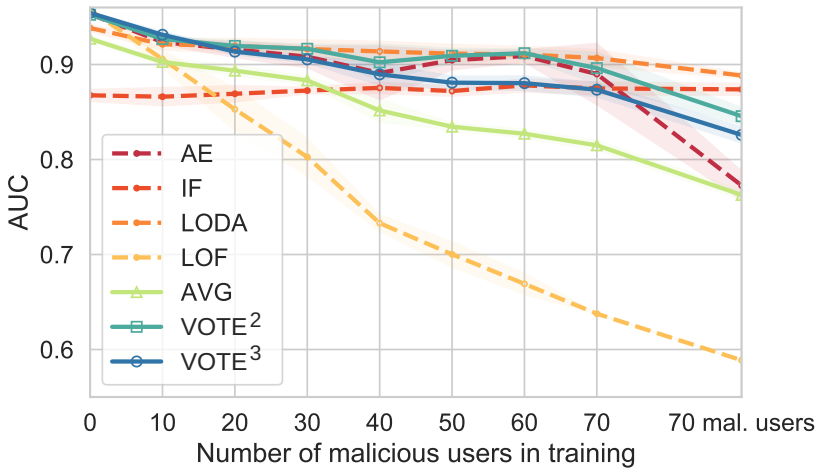


(a) Instance-based results (b) User-based results

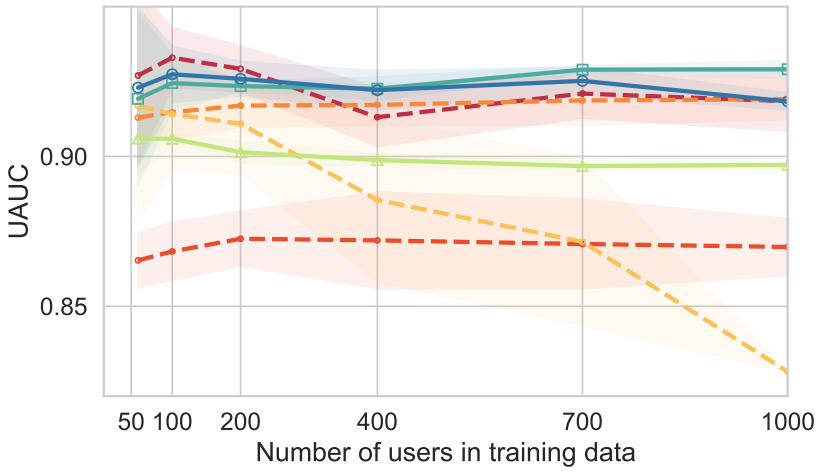
Figure 5.14: Critical Difference diagrams of results by learning algorithms and ensembles

Furthermore, we explore the effects of ensemble schemes under different training conditions as in Section 5.4.3. Figure 5.15 shows results (in UAUC) of ML algorithms and ensembles on CERT R4.2 day data under different training conditions. It is apparent that voting schemes, especially VOTE^2, achieve the best or near best detection performance in almost all cases. Moreover, with more training data (Figure 5.15b), VOTE^2 is able to outperform all other algorithms. On the other hand, while AVG shows similar results to voting-based ensembles, Figure 5.15 shows that combination by averaging is not favoured under adverse learning conditions.

On time requirement consideration, it is noteworthy that while the computation cost (and hence time) of combining scores by the individual algorithms is insignificant, in order to create an ensemble, all components need to be trained. Hence, the ensembles are restricted by the slowest algorithm (e.g. LOF) in both training and predicting. In the particular cases of the datasets employed in this work, the



(a) UAUCs under training data poisoning



(b) UAUCs with different number of users in training data.

Figure 5.15: UAUC of learning algorithms and ensembles under different training conditions on CERT R4.2 day data.

time required to train and evaluate detection models is reasonable (Fig. 5.8), hence permitting their use in the current form. In other real-world applications, lightweight components can be selected to create ensembles to avoid time and computation cost burdens.

5.5 Discussions and Comparisons

In this part, CERT R6.2 is employed for testing purposes, as it represents more malicious insider threat cases and better mimics real-world conditions (only 5 malicious insiders). We study anomaly detection results given by the proposed system under specific scenarios and show how security analysts may use these to further investigate and identify malicious behaviours. Results on each insider threat scenario and comparisons with other works in the literature are also presented.

5.5.1 Case Study of Anomaly Alerts

Using a unique `id` for each data instance used in the anomaly detection process, the corresponding course of original user actions can be quickly examined, once an anomaly alert is raised. A true anomaly alert example on CERT R6.2 is associated with actions of user `PLJ1771` – an IT administrator – on August 12, 2010. Using AE and P_{30} representation, the data instance was assigned an anomaly alert with 99.99% confidence (i.e. the data instance has an anomaly score higher than 99.99% of CERT R6.2 data). By studying the action sequence of the user on the day, his/her malicious behaviour can quickly be confirmed: The user visits several sites providing computer monitoring software, downloads a keylogger and puts it on a USB. Later in the day, they log onto `PC-3999`, which belongs to their supervisor – `HIS1706`, and start keylogging on the PC. This corresponds to the behaviours of a "disgruntled system administrator" in the CERT dataset [24].

Another true anomaly alert is raised with 99.93% confidence for activities of user `CDE1846` on March 22, 2011, in which the user logged in after work hours to `PC-5014`, which belongs to another user. Then, he/she opens and emails multiple documents to his/her personal email.

On the other hand, several false alarms generated by the anomaly detection system are worth investigating as well. For example, false alarms are raised for user `YNW2855`

on September 24, 2010 and user RRH3057 on November 03, 2010 with confidence of 99.90% and 99.99%. Investigating the original user activities on both days reveals multiple actions (file accesses, website visits) very late after work hours (around 10 PM). While these examples may not depict malicious intentions (as per the dataset's ground truth), their anomalous nature needs to be inspected to ensure the safety of the system and data.

These case studies show how a system administrator may leverage the anomaly detection system's output to identify the true nature of alerts as well as perform appropriate responses, with reference to the reorganized course of actions (by user, time) in log files. Furthermore, in manually investigating the original user's activities corresponding to each alert, the analyst may have access to more restricted information that was not incorporated in the ML system's training, such as email content, to make informed decisions.

5.5.2 Detection Performance on Insider Threat Scenarios

As mentioned in Section 4.1.1, there are five malicious insiders in CERT R6.2, each depicts a unique threat scenario. This part examines the detection results on the scenarios.

Table 5.5 presents the malicious insiders and detection results using AE and week data with P_{60} representation of CERT R6.2. Detection delays (at 10% IB), which is the time between the first malicious action and when the malicious user is detected, are also presented in the table. As the table shows, scenarios 1, 3, and 4 can be detected very easily using the proposed system with only 0% to 0.04% FPR (or 0.04 to 0.15% normal users flagged wrongly). All malicious instances of those users are detected with less than half a percent (0.32%) FPR. Scenarios 1 and 3 can also be detected very quickly.

On the other hand, threat scenarios 2 and 5 are much harder to detect, resulting in FPRs of 3.07% and 8.36%, respectively. At a UFPR of 26.46%, a system analyst will need to inspect more than 1000 users to identify the malicious user MBG3183. The descriptions of these scenarios show much less intrusive malicious behaviours than the other three scenarios [24]. For example, in scenario 5, "a member of a group decimated by layoffs uploads documents to Dropbox, planning to use them

for personal gain" (Section 4.1.1). This explains the lower detection performance on these two scenarios, as they are easy to be mistaken as normal activities.

Table 5.5: Detection performance on specific insider threat scenarios. DD: detection delay.

Threat Scen.	**Username**	**Min. FPR to detect**	**FPR to detect *all* malicious instances**	**UFPR**	**DD - week data (days)**	**DD - day data (days)**
1	ACM2278	0.02%	0.06%	0.12%	3.22	0.22
2	CMP2946	3.07%	6.97%	13.00%	5.74	0.74
3	PLJ1771	0.00%	0.00%	0.04%	2.79	0.79
4	CDE1846	0.04%	0.32%	0.15%	5.68	3.07
5	MBG3183	8.36%	8.36%	26.46%	4.57	0.57

5.5.3 Robustness of the Trained Models

For this analysis, we use an anomaly detection model trained on one CERT dataset (R4.2) to detect new anomalies on another one (R6.2). As CERT R6.2 is a newer version with changed generative models and a larger size [24], this experiment can be seen as applying the anomaly detection model of a company for a different one. User-based AUCs on CERT R6.2 week data by AE models trained using the original and P_{30} data representations are shown in Figure 5.16. The figure shows that the anomaly detection model trained using CERT R4.2 data with P_{30} representation can achieve very good AUC when tested on CERT R6.2 (UAUC=0.908). The result is vastly improved over a model trained using R4.2 via the original data representation (UAUC=0.511). This demonstrates the robustness of the proposed system when percentile data representation is used. The result suggests that modelling user data points in percentile representation brings in the temporal information of the user's previous data instances and therefore allows the model to generalize better.

5.5.4 Comparative Study

The proposed system shows clear advantages in both detection performance and the ability to generalize when compared to other works in the literature employing unsupervised anomaly detection methods for insider threat detection on the CERT datasets [5, 63, 78–80, 84, 97, 108].

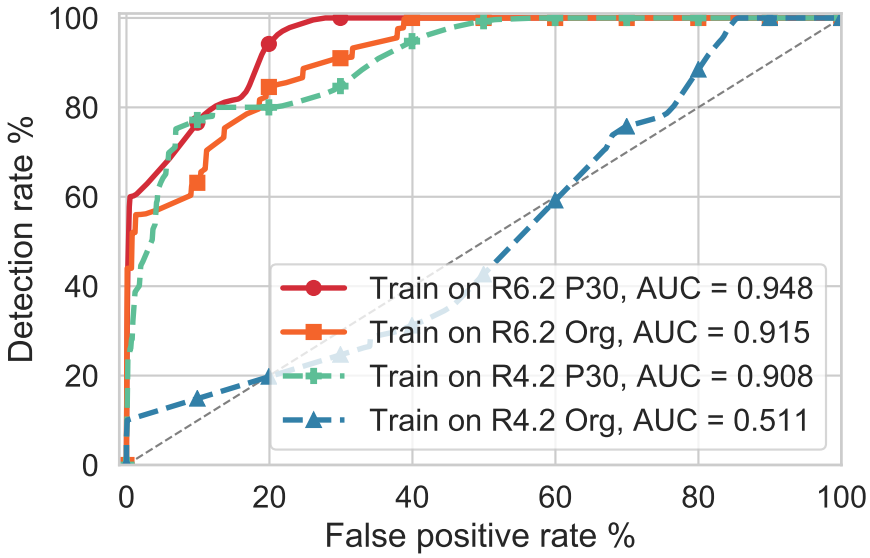


Figure 5.16: UAUC of models trained on CERT R4.2 and R6.2 data when tested on R6.2

On CERT R4.2, the proposed approach obtained AUC of 0.907 and 0.909 on week and day data (Section 5.4.4), outperforming previous works [5,63,97] that used HMM and OneClass-SVM, which achieved AUC of 0.83 and 0.89, respectively. On CERT R6.2 data, the proposed approach achieved AUC of 0.977 and 0.981 on day and week data. In comparison, recent best AUCs achieved on R6.2 day data were 0.814 (Matterer *et al.* [84]), and 0.956 (Liu *et al.* [79], on only 3 malicious insiders). This demonstrates the advantage of the proposed approach in embedding temporal information in data representation, as opposed to using a learner with temporal learning capabilities such as Long Short-Term Memory [84] and Markov models [97]. On R6.2 week data, recently, [80] achieved AUC of 0.999. However, they only tested on 500 users and 1 easy-to-detect malicious user (`ACM2278`, see 5.5.2). Under the same malicious user consideration, the proposed approach posts an AUC of 0.9996. Similarly, log2vec [78] achieved AUC of 0.93 with only 6 malicious users and 12 normal users in CERT R6.2 included in the evaluation, while the result in this thesis (higher AUC) is obtained on the full dataset. Furthermore, to the best of the author's knowledge, no other work has been able to show the ability of the anomaly detection solutions to generalize (robustness) on other datasets as illustrated in Section 5.5.3.

On LANL, the proposed approach achieves comparable results to unsupervised approaches in the literature [78, 116]. Note that other recent works on the datasets

achieved higher AUCs, but they used supervised learning, as in [14], or presented results by log lines [16, 20], which significantly increase the number of alerts.

Finally, the proposed approach is the first evaluated for insider threat detection on the publicly available logs in TWOS dataset, to the best of my knowledge. Previous works mostly focus on employing mouse captures in the dataset for user authentication [28]. In one masquerader detection approach employing the data [106], the host monitor log containing file system access events is employed to achieve AUC of 0.851. However, this log is not publicly available (see Section 4.1.3).

5.6 Summary

In this chapter, an unsupervised ML based anomaly detection approach for insider threat detection is presented. To this end, four different anomaly detection algorithms with different working principles are employed. The methods are studied using different representations of data with temporal information, including concatenation, percentile and mean or median difference. In doing so, the aim is to describe the changes in user activities that could highlight the detection of anomalous behaviours. Experiments under different constrained conditions are performed on publicly available datasets and comprehensive results are reported. Results show that Autoencoder using percentile representation of data is the best combination for anomaly detection. Temporal data representation in percentile format achieves significant improvements over original extracted data, which enables effective insider threat detection under very low investigation budgets and generalizes well on new data. Moreover, experiments demonstrate the robustness of LODA, which may suggest its use under extreme conditions and for low time complexity online learning and prediction. Furthermore, when training resources permit, a voting-based ensemble of anomaly detection can be used to improve detection performance and robustness. Comparing with the existing literature, the proposed approach shows clear advantages in detection performance and the ability to generalize to work under different environments.

Chapter 6

Insider Threat Detection using Machine Learning

This chapter presents and evaluates a machine learning based system for user-centered insider threat detection. Using machine learning, analysis of data is performed on multiple levels of granularity under realistic conditions for identifying not only malicious behaviours, but also malicious insiders. Detailed analysis of popular insider threat scenarios with different performance measures is presented to facilitate the realistic estimation of system performance. Evaluation results show that the machine learning based detection system can learn from limited ground truth and detect new malicious insiders in unseen data with high accuracy. Specifically, up to 85% of malicious insiders are detected at only 0.78% false positive rate. The system is also able to quickly detect malicious behaviours, as low as 14 minutes after the first malicious action. Comprehensive result reporting allows the system to provide valuable insights to analysts in investigating insider threat cases[1].

The chapter is organized as follows. Sections 6.1 and 6.2 present the proposed system and the ML algorithms employed. Section 6.3 details the experimental settings, while evaluation results are presented in Section 6.4. Further analysis of insider threat scenarios and discussion of the results are presented in Sections 6.5 and 6.6. Finally, conclusions are drawn in Section 6.7.

6.1 Overview of the Insider Threat Detection System

The proposed approach of a system for malicious behaviour and insider threat detection is illustrated in Figure 6.1. Following the process in Chapter 4, data sources are processed into numerical data format with different granularity levels and temporal representations. A limited ground truth on the data may be obtained via initial detection, as presented in Chapter 5. Alternatively, based on an organization's normal

[1]Parts of this chapter have been published at IEEE Transactions on Network and Service Management [71] (© 2020 IEEE)

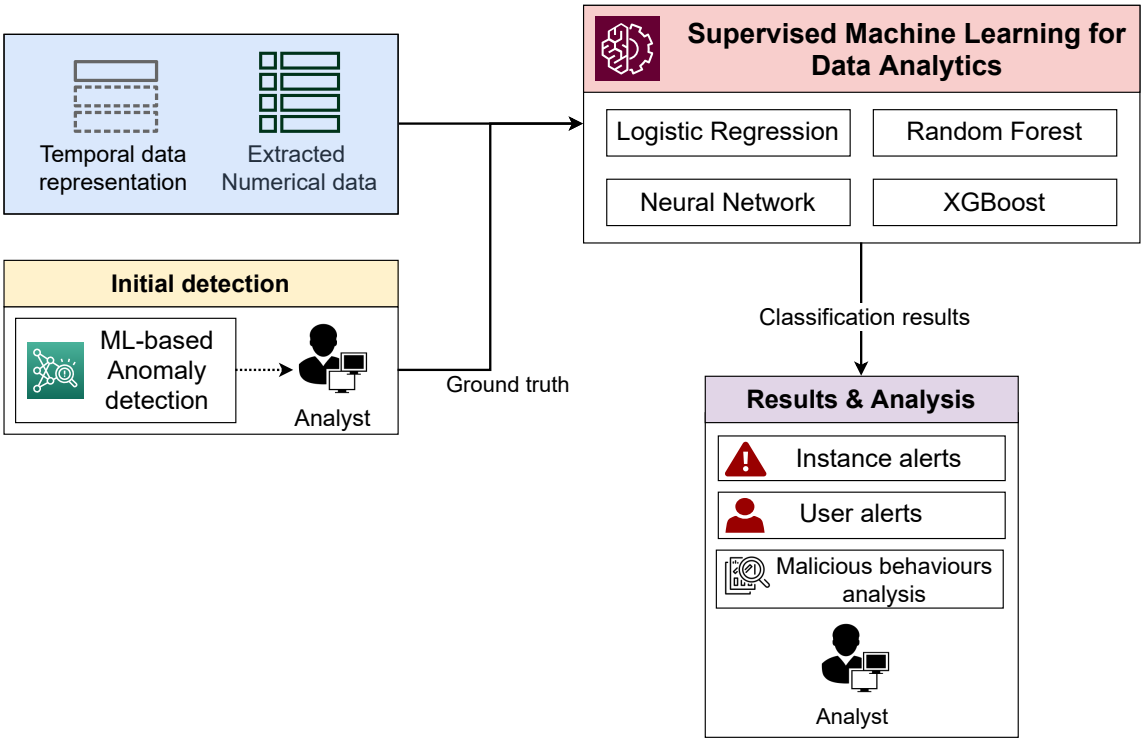


Figure 6.1: Overview of the insider threat detection system

network and system conditions, security analysts may notice suspicious activities or unusual changes in user behaviours. The security analyst would follow up by performing an investigation to identify whether these are originated from malicious action, which may result in some amount of data being labelled. Supervised machine learning models are trained on extracted data using the limited amount of ground truth. The employed ML methods are described in Section 6.2. The ML-based classification results are then presented to cyber-security analysts for further analysis and responses. This chapter presents an analysis of different insider threat scenarios/behaviours, data granularity levels, and training conditions.

In this chapter, we assume the following benchmarking datasets: CERT R4.2, CERT R5.2, and LANL. In each case, ML algorithms are trained with a limited amount of ground truth on malicious/normal user behaviours with the aim to detect unknown malicious insiders. Then, we explore how well the learned solution would be able to generalize for detecting unknown malicious insider cases in the datasets. Using supervised learning, the benefit is that we need not assume that data clusters are always synonymous with distinct behaviours. Hence, this may allow higher precision in detecting insider threats than unsupervised learning / anomaly detection algorithms [21] (Section 6.4.1.2).

In addition, focusing on user-centered detection of insider threats, the analysis in this chapter will distinguish between malicious actions detected and malicious users detected, where the two are not necessarily the same. For example, the diversity in a user's role within an organization can impact on the number/types of actions performed, both normal and malicious. Additionally, in many cases, user actions can vary over time and have different contexts that need to be taken into account in order to process an alert about a suspicious behaviour [112]. Thus, high malicious instance detection rates in this case may not necessarily translate to all malicious insiders being detected.

Finally, several measures are presented in this chapter, such as detection delay per malicious insider, or the support for each malicious insider alert. By providing these measures, in addition to traditional cyber-security metrics, such as detection rate and false positive rate, we aim to provide better support to security analysts and positively contribute to a successful application of the proposed system in real-world

scenarios.

6.2 Machine Learning for Data Analytics

In this chapter, the following four well-known and widely-used ML algorithms are employed: Logistic Regression, Random Forest, Neural Network, and XGBoost [21, 104]. Brief descriptions of the algorithms are presented below, while more detailed descriptions can be found in [104].

6.2.1 Logistic Regression (LR)

LR is a linear statistical model that uses a logistic function to model a binary dependent variable based on independent variables. In statistics, the logistic model is suitable to model the probability of a certain class or event, such as pass / fail, win / lose. In this research, the corresponding outputs are normal / insider threat behaviours. Specifically, in this work, the logistic function, σ, is used to model the probability of normal or malicious insider behaviour for each input x:

$$\sigma(w^T x) = (1 + e^{-w^T x})^{-1} \tag{6.1}$$

Logistic regression training (with $l2$-regularization) results in the identification of a weight vector, w, that minimizes the sum of squared errors between logistic function $\sigma(w^T x)$ and target labels.

Logistic regression has the advantage of being highly interpretable as a linear model. Furthermore, it returns a probability of an input vector belonging to a class, thus facilitating the prioritization of the most suspicious actions for investigation. In this work, logistic regression is included as a baseline model [23].

6.2.2 Neural Network (NN)

The NN assumed in this work takes the form of a multi-layer perceptron with up to three hidden layers.[2] Neural networks with at least a hidden layer provide the ability to model a wide range of non-linear properties [11]. Reliably training such architectures has recently been made possible through developments in approaches to

[2]Larger architectures could be trained, but at the expense of computational cost and interoperability, factors that are also potentially important in real-world applications [21].

credit assignment and representation. Specifically, back-propagation with the Adam formulation of stochastic gradient descent is assumed in this work [59], where this implies that each weight in the network has its own (adaptable) learning rate. Thus, given a mini-batch sample of training exemplars, statistics are collected to enable second order information to be collected, i.e. the second moments of the gradients are inferred. The resulting combination of per weight learning rate adaption and (stochastic) back-propagation of the error represents a more robust scheme for weight updating than many previous more computationally expensive methods [59]. The use of rectified linear activation functions (as part of the representation) in the hidden layers also accelerates learning across multi-layer architectures [44].

6.2.3 Random Forest (RF)

RF represents a process for building an ensemble of decision tree classifiers that collectively 'vote' to provide a single class label for each exemplar [18]. Given p training exemplars, each described in terms of d attributes, each decision tree is define by: (1) selecting a random *subset* of the p training exemplars; (2) identify a random subset, D, of the d attributes ($D \ll d$); (3) each of the D selected attributes are used to parameterize a new decision node in the decision tree [104]. Each leaf of a tree represents a "decision", or in classification tasks, a predicted class label. Typically, each tree of a random forest is trained using CART algorithm [82], which seeks to maximize information change – measured by "gini" impurity – at each split of the tree. Gini impurity is calculated as $\text{GINI} = 1 - \sum_{i=1}^{J} p_i^2$, where p_i is the fraction of items with label i in a set of J classes.

The entire process is repeated to create a user-specified number of decision trees. The selection of attributes (step 2) and design of decision tree nodes (step 3) is contextual on the subset of exemplars to build each decision tree, and the subset of attributes used to build each decision node. These properties have been formally shown to preclude over learning, making the predictions of the RF robust [18].

6.2.4 XGBoost (XG)

Similar to RF, XG also assumes decision trees as the classifiers to compose an ensemble [26]. However, XG assumes a gradient boosting method, where the combination

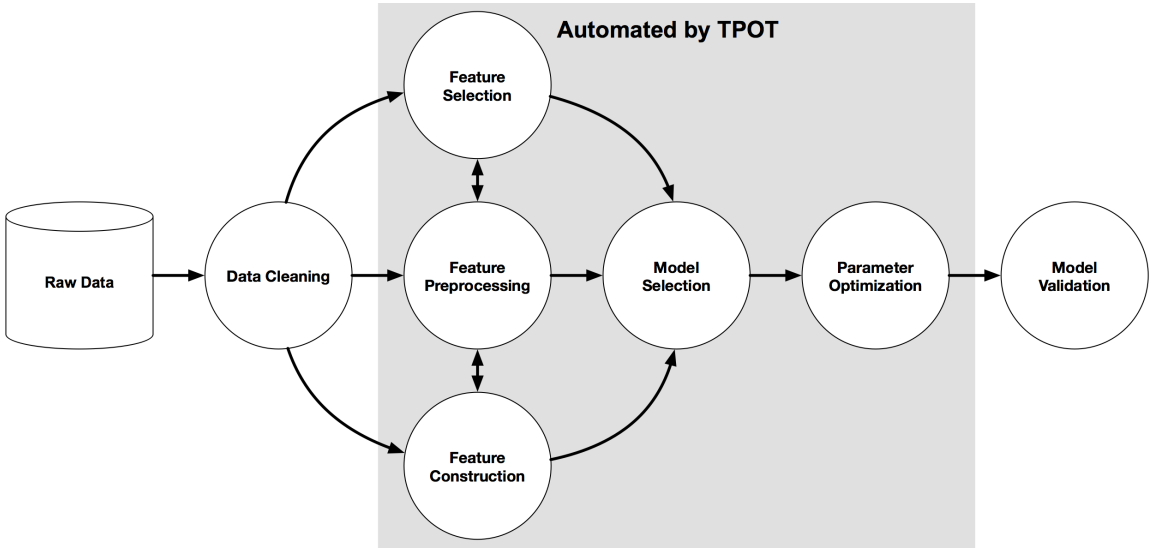


Figure 6.2: An example machine learning pipeline with components automated by TPOT [90]

of simple decision trees into a strong ensemble is guided by the optimization of a differentiable loss function. Moreover, in boosting, the classifier output predicts class labels through probabilities obtained using the logistic transformation of a linear combination of each decision tree output. Different measures can be used with boosting to reduce overfitting, such as random subspace method, and random subset of training data available to each tree.

XGBoost provides improvements over traditional gradient boosting methods to allow a highly scalable tree boosting system. Examples include regularization, a mechanism for operating under sparse and weighted data, and adopting a block structure for parallel learning. The algorithm has been successfully applied to applications in a wide range of data mining tasks, including cyber-security [27, 36]. Due to its popularity, in this work, we include XG to test its capability in insider threat detection.

6.2.5 Automatic Optimization of Classifier – TPOT

With the aim of examining the possibilities of employing an automated machine learning approach (AutoML) to optimization of classifiers for insider threat detection, we employ Tree-based Pipeline Optimization Tool (TPOT)[3] [90] in this research. In

[3] `https://github.com/EpistasisLab/tpot`

general, automated machine learning refers to frameworks that automatically explore, select and tune the components of a machine learning pipeline to achieve well-performed models with little to no human involvement [107]. AutoML is mostly used in supervised machine learning tasks, i.e. classification and regression.

TPOT is an AutoML system based on evolutionary computation to optimize a series of feature selectors, pre-processors and ML models based on an objective, such as maximizing classification accuracy [73]. An example machine learning pipeline with components that TPOT optimizes is presented in Figure 6.2. In addition to model selection and hyperparameter optimization, like most other AutoML systems, TPOT also performs feature selection and feature engineering / construction. Specifically, TPOT pipelines are designed as binary expression trees with ML operators as primitives. Components of the pipelines are selected from implementations of algorithms in scikit-learn [93] and other libraries. By default, TPOT evaluates a wide range of algorithm or classifier tasks, including four algorithms presented in this section (LR, RF, NN, XG), and others, such as Naive Bayes, k-nearest neighbours, and Extra trees classifier. The complete pipelines in TPOT are evaluated based on their cross-validated scores, such as accuracy or f1-score. TPOT then optimizes the pipeline using genetic programming with the NSGA-II Pareto optimization [33].

TPOT has been shown to outperform standard machine learning techniques [73]. Hence, in this research, we explore if TPOT is capable of achieving better performance than other ML methods employed in this chapter.

6.3 Experimental Evaluations

In this chapter, we present the evaluation of the proposed system for insider threat detection using the CERT insider threat datasets R4.2 and R5.2, and LANL and TWOS datasets. Chapter 4 details the process to extract numerical features from the data sources, as well as data pre-processing based on the identification of multiple levels of data granularity. In Section 6.3.1, the experiment settings and performance measures are introduced. Section 6.3.2 details the training configuration and parameterization of the employed ML algorithms, while Section 6.3.3 presents the metrics for measuring the detection performance.

6.3.1 Experiment Settings – Realistic training condition

Based on results obtained in Chapter 5, this chapter focuses on using original extracted numerical data and percentile-based temporal data representation (see Section 4.3).

In this chapter, the aim is to obtain a realistic estimation of the proposed system's performance on real-world detection tasks, based on scenarios characterized by limitations to the amount of ground truth data available for training the ML algorithms. Specifically, in real-world environments, labelled (ground truth) data for training detection systems is scarce. Thus, ground truth is only obtainable from a limited set of verified users, while behaviours of others are generally unknown [13, 43]. To emulate this condition, we assume a primary configuration – namely *realistic* condition thereafter – where ground truth is obtained from only a restricted set of users over a given time period.

On CERT R5.2, the ground truth data for training the ML algorithms is limited to the data of 400 identified "normal" and "malicious" users (among 2000 users in the organization), based on the *first* 37 weeks – 50% of the time period that the dataset covers. By user count, this allows the ML algorithms to learn from data representing 18% of "normal" users and 34% of malicious insiders. Similarly, on CERT R4.2 and LANL data, the user limits are 200 (in 1000 users) and 2000 (in 11814 users), while duration limits are first 36 weeks (50%) and first 12 days (in 30 days), respectively. It is noteworthy that from a detector's point of view, the "normal" users in the training data are only guaranteed to be benign in the training duration, while later in the testing weeks, they may or may not turn "malicious". Additionally, we further ensure experiments are realistic by presenting results obtained solely from unknown users, i.e users that have not performed any malicious actions in the training duration. By excluding known malicious users, i.e. users whose malicious actions are included in training data, from system performance measures, we believe the evaluations reflect real-life situations, as well as cyber-security analyst's interest [112].

Conceptually, by employing *realistic* training condition, we aim to estimate how well a detection model developed on a dataset could perform on *future unseen* data of the organization. In the first experiment, to show the contrast between traditional ML applications and real-world cyber-security situations, we compare the *realistic*

setting above with an *idealistic* (traditional) setting, where a random 50% of data from the whole dataset is used to train the ML algorithms. This is done at three levels of data granularity: user-week, user-day, and user-session. The second experiment evaluates the ML algorithms in *realistic* setting on all aforementioned data granularity levels to obtain detailed results, both instance-based and user-based. Detailed analysis is performed on the results for each insider threat scenario provided in the dataset. Furthermore, models trained on CERT R5.2 are also used to test against other versions of CERT insider data for exploring the generalization of the trained models' performances under new / unseen environments (different version of the CERT data emulates different organizations). The evaluation results are obtained from a series of experiments, where each setting – a ML algorithm on a data type – is randomly repeated 20 times.

6.3.2 ML Training Configuration and Parameterization

In this research, Python 3.7 is used for data pre-processing steps and Scikit-learn [93] and XGBoost [26] are used for implementing ML algorithms. The training data are normalized, per attribute, to zero mean and unit variance before being used to train the ML algorithms. The ML algorithms are trained in a binary setting, where the two classes are: malicious (*positive*) and normal (*negative*).

Logistic Regression assumes the *lbfgs* solver [76] and appeared to perform best under default parameters. In the case of the three remaining algorithms, we perform parameter search with cross-validation using hyperopt, which is a parameter tuning solution based on the tree-structured Parzen estimator [12]. Specifically, for RF, we tune the number of decision tree estimators (50 to 300), the number of features to consider when looking for the best tree split (all features, square root and log base-2 of all features), and the depth of individual trees (3 to 10, or unlimited). Similarly, we tune the number of estimators in XG (50 to 300), the depth of each tree (3 to 25), the feature and sample subset size (0.5 to 1). Finally, for the NN, a computational limit of 250 epochs was assumed. A search between 1 to 3 for the number of hidden layers was conducted, where each hidden layer has the size set to a half of the previous layer.[4] Different mini-batch sizes were tested (32 to 256), and L2 regularization penalty (10^{-6}

[4]Enforces a 'bottleneck' effect that encourages the network to discover a suitable encoding.

to 10^{-1}) are also tuned. In each case, parameter tuning is limited to training data alone.

6.3.3 Performance Metrics

In cyber-security applications of ML, detection rate (DR), which is also called recall, and false positive rate (FPR) are widely used [21].

$$DR = \frac{TP}{TP + FN}, \tag{6.2}$$

$$FPR = \frac{FP}{TN + FP}, \tag{6.3}$$

where TP, TN, FP, FN are True Positive, True Negative, False Positive, and False Negative, respectively. In this research, TP represents the number of malicious samples that are correctly classified as "malicious", and FN represents the number of malicious samples that are incorrectly classified as "normal". On the other hand, TN (FP) are the numbers of normal data samples that are correctly (incorrectly) classified.

In addition to DR and FPR, we also report the system performance by Precision (Pr) and F1-score (F1). In particular cases, Accuracy (Ac), Receiver operating characteristic (ROC) curves and Area under the curve (AUC) are also presented.

$$Pr = \frac{TP}{TP + FP}, \tag{6.4}$$

$$F1 = \frac{2}{Pr^{-1} + Recall^{-1}}, \tag{6.5}$$

Precision represents the percentage of malicious alarms generated by the system that is true. F1-score summarizes both DR, or recall, and Pr as a harmonic mean. Due to the extremely skewed data (Table 4.2), a relatively small FPR may still translate to a large number of false alarms. By reporting results using Pr and F1, the cost of false alarm investigation is better presented. It is noteworthy that except ROC AUC, the remaining metrics are calculated based on the default classification thresholds generated by the algorithms post-training on each test data, which separates normal /

malicious predictions. For the sake of brevity, in the following, we report performance metrics (Ac, DR, FPR, Pr, F1) in percent (%).

As mentioned in Section 6.1, system performance is reported in terms of *both* data instances correctly detected (instance-based results), and users correctly detected, (user-based results). For user-based results, a normal user is misclassified if at least one of their data instances is classified as "malicious", while a malicious insider is identified if at least one of their malicious data instances is classified as "malicious" by the system. We therefore have two sets of performance metrics: Instance-based (IAc, IDR, IFPR, IPr, IF1) and User-based (UAc, UDR, UFPR, UPr, and UF1).

Finally, to further analyze the insider threat cases and provide better insights into the effectiveness of the approach, we introduce the following measures: *detection delay* (DD) and *detection rate per detected malicious insider* (DR/DMI). DD can be defined as the time duration between the first malicious action performed by a malicious insider until he/she is detected (if ever). On the other hand, DR/DMI is the percentage of malicious instances detected per malicious user. For example, if a malicious insider performs data exfiltration over 5 weeks and only one user-week data instance of the user is flagged as "malicious", DR/DMI for this case is $1/5 = 0.2$. These additional metrics could be helpful in evaluating insider threat detectors' performances, where DD demonstrates how quickly the system is able to detect a malicious insider, and DR/DMI represents the extent of the user's malicious actions uncovered by the system.

To compare between the algorithms, data types, or experiment settings, we adopt F1 performance metric for its expressive power. Naturally, a system achieves high F1 when it does well on both recall (DR) and precision, which means high malicious detection rate at a cost of low false alarm rate. Pair-wise comparisons are supported by Wilcoxon signed-ranks test, which is the nonparametric analogue to the paired t-test for statistical significance tests [35, 113]. In the cases of tests between multiple algorithms and data types, Friedman test with Bonferroni-Dunn posthoc test is used [35, 39] (see Section 5.3.2).

Table 6.1: Instance-based results: Realistic vs Idealistic

Setting	Alg.	Week					Day					Session				
		IACC	IFPR	IDR	IPr	IF1	IACC	IFPR	IDR	IPr	IF1	IFPR	IFPR	IDR	IPr	IF1
Idealistic	LR	99.74	0.09	55.13	71.70	62.28	99.86	0.01	33.26	81.94	47.30	99.85	0.01	22.13	89.04	35.44
	NN	99.83	0.03	55.82	89.27	68.20	99.90	0.05	48.99	68.57	56.68	99.87	0.04	46.90	69.99	55.88
	RF	99.80	0.00	57.55	99.97	73.05	99.86	0.00	44.60	99.98	61.68	99.86	0.00	27.80	99.96	43.49
	XG	99.86	0.01	66.65	97.69	79.22	99.95	0.00	74.85	98.81	85.16	99.92	0.00	58.76	96.70	73.09
Realistic	LR	98.37	1.40	53.77	16.94	25.53	99.17	0.70	43.88	13.44	20.27	99.56	0.27	26.49	19.82	22.34
	NN	99.73	0.41	45.23	38.03	40.33	99.84	0.41	39.90	20.54	26.30	99.82	0.14	29.28	37.20	31.80
	RF	99.30	0.00	49.54	99.39	66.07	99.45	0.03	41.97	83.70	55.12	99.69	0.02	31.54	81.98	45.10
	XG	99.67	0.14	63.37	74.10	67.63	99.70	0.20	56.39	44.84	48.52	99.53	0.31	31.76	22.09	25.11

6.4 Evaluation Results

In this section, first the effects of different training conditions on detection performance are examined in 6.4.1. Section 6.4.2 presents results based on data representations, while detection performances by ML algorithms and by automatic optimization of classifiers (using TPOT) is presented in 6.4.3. Finally, results by data granularity levels are presented in 6.4.4.

6.4.1 Results by Training conditions

In this section, we perform comparisons between training conditions: realistic vs. idealistic, and supervised learning vs unsupervised learning. For this purpose, we use the original extracted numerical data of CERT R5.2

6.4.1.1 Realistic vs Idealistic

In this experiment, we compare instance-based results on user-session, user-day, and user-week data between the two training conditions (realistic vs idealistic) in order to demonstrate the challenges in applying ML solutions in real-world environments. Instance-based results obtained from the two settings are shown in Table 6.1 and Figure 6.3. It is clear that results obtained under the *idealistic* setting - 50% of all data instances are used for training, sampled across the entire temporal period - are significantly better on almost all measures than results obtained in the *realistic* setting - training data is limited to data of only 20% of users in the first half of the dataset. Performing Wilcoxon signed-ranks test returns $p = 0.003$. Thus, this rejects

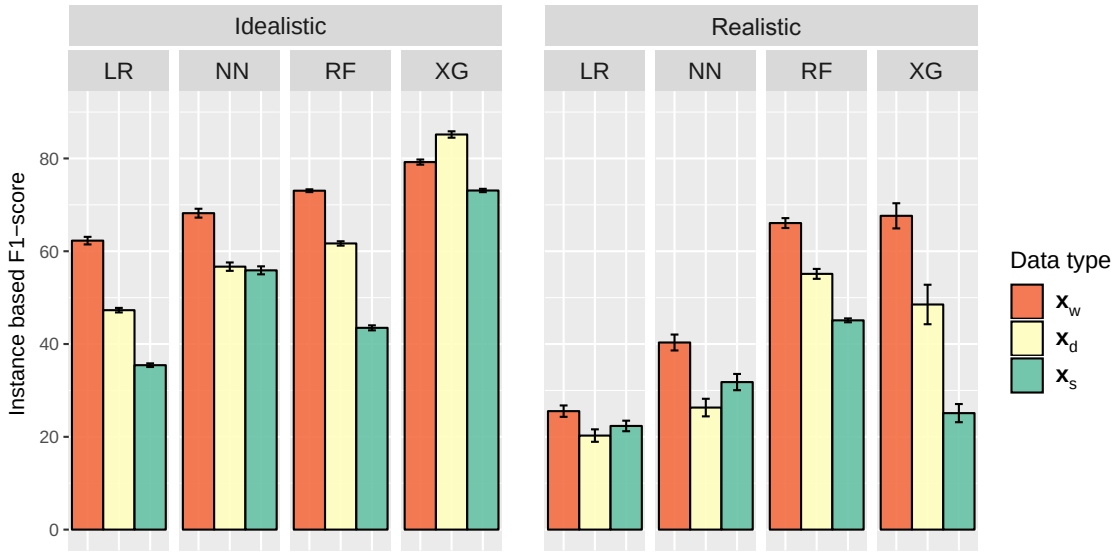


Figure 6.3: Instance-based F1-score by data types and algorithms under *realistic* and *idealistic* training conditions. Error bars show 95% confidence intervals.

the null hypothesis: ML algorithms under the two training conditions (*idealistic* and *realistic*) perform similarly with regards to IF1 at the 1% significance level.

By employing *idealistic* training condition, most ML algorithms achieved near perfect IFPR, and higher IDR than under the *realistic* setting. The only exception is RF over user-session data, where IDR and IF1 obtained from *realistic* setting is better. Figure 6.3 also shows that RF achieves the most similar performances between *idealistic* and *realistic* training conditions, which suggests RF as a good candidate for learning from limited data conditions. On the other hand, while XG performs better under the *idealistic* setting than the remaining algorithms, its performance degrades greatly under *realistic* training condition, especially on higher granularity data, i.e user-session data.

From a ML standpoint, all ML algorithms achieve relatively low instance-based false positive rates under *realistic* condition, where IFPR $< 0.5\%$ in most cases. However, in practice, due to the fact that normal data instances account for more than 99.6% of all data instances, the amount of false positive alarms may still pose a challenge to analysts. For example, a 0.14% IPFR by NN on user-session data is equivalent to 1400 false malicious instance alerts.

The presented results highlight the challenges observed in many real-world applications of ML, especially in cyber-security, where ground truth is limited and typical ML setting (random training / testing splitting of data) can not be satisfied, given that such a split assumes global information. Thus, results obtained in a typical (*ideslistic*) ML setting may not necessarily reflect real-world cyber-security performances, and realistic settings need to be adopted in system design in order to ensure a smooth transition to deployment environments and valid estimation of performances.

6.4.1.2 Learning algorithm – Supervised vs Unsupervised

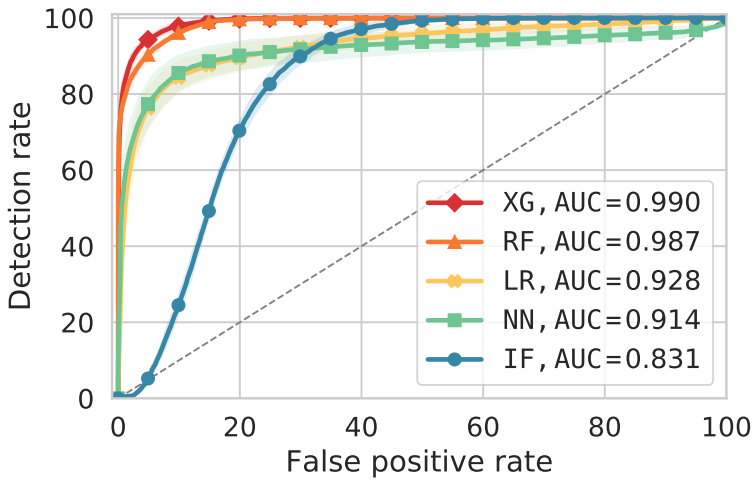
Table 6.2: Instance-based results by Isolation Forest

Threshold	User-Week			User-Day			User-Session		
	IFPR	IDR	IPr	IFPR	IDR	IPr	IFPR	IDR	IPr
1%	0.94	0.38	0.22	1.17	3.01	0.59	1.14	12.54	2.50
5%	4.98	6.31	0.62	4.96	24.30	1.09	4.90	31.66	1.44
10%	9.42	26.42	1.32	9.23	50.99	1.17	9.08	49.87	1.17

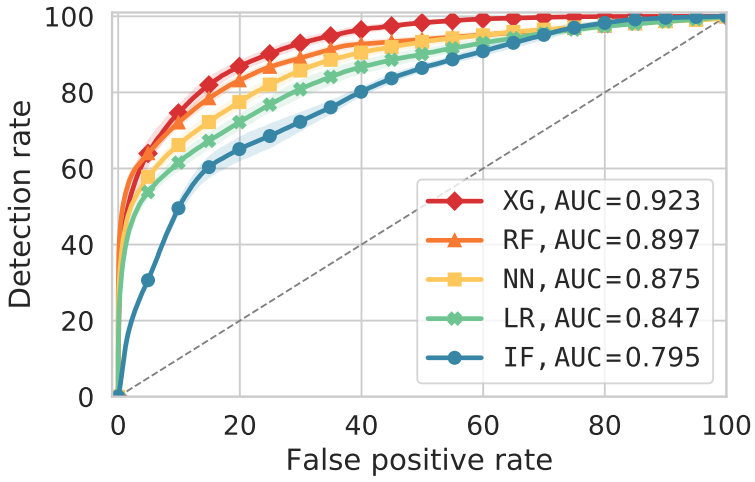
In this section, we compare in *realistic* training condition the performances of employed ML algorithms and Isolation Forest (IF) [77], a prominent unsupervised learning approach that has been employed in many network anomaly detection work recently [4]. IF assumes that the anomalous data instances are easier to isolate from the rest of the data than normal instances, hence shorter path lengths to the corresponding leaves of anomalous instances. For training IF models, the number of trees is tuned for each data type. We assume three different thresholds (1%, 5%, and 10%) for flagging data instances as "anomaly", based on different investigating budgets. Table 6.2 and Figure 6.4 illustrate the results achieved by IF. The results clearly demonstrate that when label information, albeit limited, is available for training ML algorithms, guided search as in supervised learning will achieve superior performances, especially at very low FPRs.

6.4.2 Results by Data Representations

In Chapter 5, percentile data representation has been shown to benefit anomaly detection, especially on the CERT datasets. Hence in this section, we evaluate the



(a) User-week data



(b) User-session data

Figure 6.4: Instance-based ROCs and AUCs of ML algorithms on user-week and user-session data

Table 6.3: Instance-based detection results by data representations. F1 and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Data	Data type	Alg	Original						Percentile representation					
			IAc	IFPR	IDR	IPr	IF1	IAUC	IAc	IFPR	IDR	IPr	IF1	IAUC
CERT R4.2	Session	LR	99.40	0.41	21.33	13.53	15.78	0.865	99.77	0.01	16.13	76.44	26.56	0.727
		NN	98.60	1.23	32.60	6.59	10.83	0.844	99.51	0.33	35.65	22.68	27.34	0.811
		RF	99.63	0.22	40.72	37.88	36.85	0.956	99.80	0.01	24.91	89.91	38.95	0.984
		XG	99.11	0.75	44.60	14.34	21.21	0.947	99.83	0.04	50.43	77.71	60.82	0.983
	Day	LR	99.10	0.67	27.08	12.58	16.75	0.878	99.72	0.06	31.29	63.11	41.78	0.730
		NN	98.72	1.10	42.45	11.63	17.99	0.898	99.84	0.05	67.89	80.28	73.53	0.950
		RF	99.31	0.55	55.88	27.68	35.72	0.968	99.85	0.07	74.61	79.75	76.82	0.998
		XG	99.21	0.66	57.93	23.28	32.68	0.970	99.83	0.13	87.36	69.21	77.00	0.998
	Week	LR	98.41	1.31	44.13	15.30	22.58	0.904	98.68	0.93	24.97	12.71	16.68	0.646
		NN	98.85	0.87	45.63	22.56	29.78	0.914	99.49	0.14	28.68	52.96	37.08	0.771
		RF	99.45	0.22	37.60	49.82	42.01	0.971	99.71	0.02	48.44	94.56	63.72	0.983
		XG	99.24	0.47	44.49	35.09	38.54	0.974	99.18	0.64	65.39	37.14	46.51	0.973
CERT R5.2	Session	LR	99.56	0.27	26.49	19.82	22.34	0.847	99.77	0.03	17.59	57.76	26.81	0.707
		NN	99.69	0.14	29.28	37.20	31.80	0.875	99.58	0.27	37.44	26.06	30.25	0.885
		RF	99.82	0.02	31.54	81.98	45.10	0.897	99.82	0.00	25.63	98.22	40.63	0.972
		XG	99.53	0.31	31.76	22.09	25.11	0.923	99.78	0.07	36.18	57.84	44.15	0.973
	Day	LR	99.17	0.70	43.88	13.44	20.27	0.915	99.82	0.07	52.43	66.67	58.02	0.802
		NN	99.45	0.41	39.90	20.54	26.30	0.935	99.88	0.06	74.08	78.30	75.43	0.938
		RF	99.84	0.03	41.97	83.70	55.12	0.980	99.91	0.04	75.94	83.03	79.21	0.998
		XG	99.70	0.20	56.39	44.84	48.52	0.984	99.95	0.02	87.82	92.26	89.93	0.999
	Week	LR	98.37	1.40	53.77	16.94	25.53	0.928	98.75	0.95	41.75	18.83	25.91	0.715
		NN	99.30	0.41	45.23	38.03	40.33	0.914	99.63	0.08	44.42	75.99	55.61	0.823
		RF	99.73	0.00	49.54	99.39	66.07	0.987	99.69	0.02	45.35	94.36	60.82	0.986
		XG	99.67	0.14	63.37	74.10	67.63	0.991	99.57	0.17	48.85	63.11	54.42	0.973
LANL	Day	LR	99.60	0.33	15.46	3.82	6.10	0.801	99.83	0.10	10.68	8.41	9.36	0.811
		NN	99.82	0.11	13.45	10.28	11.36	0.748	99.85	0.07	7.71	8.73	8.01	0.662
		RF	99.91	0.01	7.39	34.83	11.93	0.950	99.90	0.01	0.93	3.81	1.42	0.915
		XG	99.90	0.02	7.06	21.38	10.35	0.964	99.90	0.02	1.53	5.55	2.37	0.924

Table 6.4: User-based detection results by data representations. F1 and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Data	Data type	Alg	Original						Percentile representation					
			UAc	UFPR	UDR	UPr	UF1	UAUC	UAc	UFPR	UDR	UPr	UF1	UAUC
CERT R4.2	Session	LR	90.52	9.47	90.49	29.22	43.90	0.971	98.53	0.91	86.34	81.51	83.76	0.978
		NN	83.19	17.23	94.39	17.49	29.44	0.965	86.89	13.60	99.27	22.87	37.09	0.989
		RF	96.27	3.64	94.15	55.98	69.19	0.993	98.79	1.13	97.07	80.54	87.76	0.999
		XG	93.80	6.04	90.00	40.13	55.24	0.971	96.94	2.72	89.51	61.69	72.38	0.989
	Day	LR	87.97	12.27	94.15	24.13	38.26	0.969	96.70	3.34	97.56	56.95	71.81	0.977
		NN	84.45	16.01	96.34	19.34	32.10	0.979	97.41	2.66	99.02	63.31	77.00	1.000
		RF	95.62	4.53	99.02	50.11	66.20	0.996	97.96	2.13	100.00	70.14	81.78	1.000
		XG	95.08	4.66	89.27	46.32	60.76	0.979	97.55	2.42	96.83	65.07	77.63	0.996
	Week	LR	89.79	9.86	81.46	25.75	39.07	0.910	87.38	11.89	69.51	19.44	30.29	0.906
		NN	94.61	5.06	87.07	43.65	57.93	0.950	95.83	3.33	77.07	51.27	61.44	0.942
		RF	96.85	2.52	82.93	60.34	69.49	0.990	99.45	0.51	98.54	91.18	94.33	1.000
		XG	96.00	3.25	79.27	52.60	63.09	0.978	91.06	9.02	92.93	31.16	46.39	0.971
CERT R5.2	Session	LR	93.31	6.54	89.00	31.49	46.44	0.982	96.77	2.75	82.70	50.97	62.84	0.961
		NN	96.26	3.44	87.54	47.47	61.31	0.987	89.95	10.14	93.02	22.66	36.31	0.976
		RF	98.72	0.78	84.62	81.45	82.36	0.993	99.52	0.13	89.37	96.18	92.60	0.999
		XG	97.02	2.49	82.77	54.39	65.39	0.979	97.89	1.85	90.16	62.99	73.98	0.991
	Day	LR	92.23	7.75	91.54	28.17	43.03	0.978	96.40	3.70	99.23	48.45	64.98	0.998
		NN	95.54	4.13	86.15	41.96	56.32	0.982	97.34	2.75	100.00	59.09	73.38	0.999
		RF	98.95	0.43	81.62	88.04	84.42	0.997	97.76	2.32	100.00	61.77	75.82	1.000
		XG	97.61	1.97	85.46	61.03	70.96	0.991	99.03	1.01	100.00	79.84	88.28	1.000
	Week	LR	92.20	7.77	91.31	28.12	42.93	0.971	87.98	11.75	79.38	17.61	28.82	0.915
		NN	96.50	2.82	77.00	49.00	59.77	0.922	97.80	1.53	78.77	65.85	71.33	0.958
		RF	99.08	0.02	73.85	99.33	84.60	0.995	99.72	0.10	94.62	97.35	95.89	1.000
		XG	98.42	1.00	82.00	75.52	78.25	0.992	96.78	3.22	96.62	52.33	67.42	0.996
LANL	Day	LR	97.18	2.29	19.24	5.69	8.70	0.789	98.32	1.07	15.90	9.98	12.19	0.851
		NN	98.52	0.89	13.67	10.26	11.42	0.777	98.59	0.76	11.41	10.48	10.70	0.768
		RF	99.28	0.05	3.67	36.35	6.59	0.901	99.11	0.16	1.41	4.14	1.98	0.850
		XG	99.26	0.09	6.46	32.60	10.61	0.926	99.03	0.25	2.31	6.12	3.32	0.884

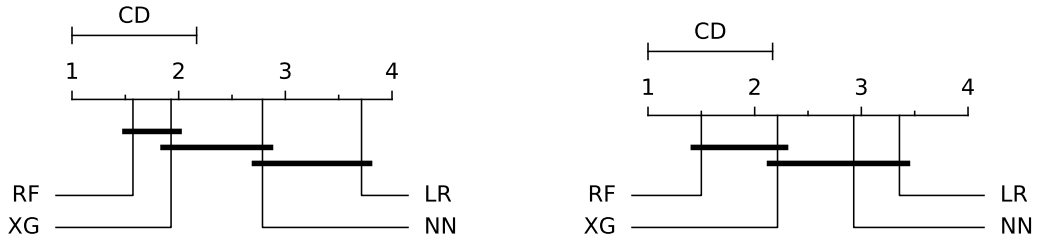
learning algorithms with two different representations of the numerical data: original extracted data and percentile.

The results are presented in Tables 6.3 and 6.4. As shown in the tables, by using percentile representation, detection performances by almost all algorithms are improved on CERT datasets, on both F1 scores and AUC. Wilcoxon signed-ranks tests return $p = 0.003$ and $p = 0.013$ for comparisons by IF1 and UF1, respectively. Furthermore, similar to the observations in Chapter 5, temporal representation of data does not seem to improve detection performance over the original extracted data on LANL dataset. This has been attributed to the appearance of temporal information in original extracted data for the case of LANL (see Section 5.4). It is also noteworthy that FPRs and DRs in the tables are measured at default classification thresholds by the algorithms, at very low FPRs. In the case of LANL data, as AUC is higher than 0.9 with RF and XG, it is expected that by adjusting the decision threshold to accept a higher FPR, DRs can be improved. For example, using XG at 1% IFPR and UFPR, IDR and UDR are improved to 54% and 38%, respectively.

6.4.3 Detection Performances by ML algorithms

As shown in Tables 6.3 and 6.4, among the ML algorithms, RF presents the best results in terms of F1-score, precision, and false positive rates, both on user-based and instance-based metrics (Figure 6.8). NN shows promising UDR – about 4% higher than RF – at a cost of higher UFPR. On the other hand, LR suffers from high UFPR and low F1 scores. Finally, while XG achieves the best performance (IF1) on user-week data, fine-grained data types have a much higher negative impact on it than on either RF or NN (Section 6.4.4).

Performing Friedman test on both user-based and instance-based F1 scores, the null hypotheses are easily rejected ($p = 0.0008$ and $p = 4 \times 10^{-5}$), which means there are significant differences between the algorithms. Figure 6.5 presents the critical difference diagram obtained using the posthoc test. The tests confirm the observations on algorithms' performances.



(a) Instance-based results

(b) User-based results

Figure 6.5: Critical Difference (CD) diagrams of results by ML algorithms. Average rank of each algorithm is shown on the scale. Two *linked* entries (connected by a horizontal black bar) are not significantly different, i.e. rank difference is less than CD.

Table 6.5: Instance-based detection results by TPOT. F1 and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Data	Type	Temp. rep.	IAc	IFPR	IDR	IPr	IF1	IAUC
CERT R4.2	session	Org.	99.02	0.86	48.80	13.17	20.50	0.944
		Percentile	99.76	0.09	42.79	68.70	50.34	0.976
	day	Org.	98.82	1.10	73.20	19.21	30.05	0.969
		Percentile	99.81	0.16	88.05	69.28	76.23	0.998
	week	Org.	99.08	0.67	52.10	32.21	38.88	0.971
		Percentile	99.33	0.50	66.83	45.44	52.47	0.978
CERT R5.2	session	Org.	99.34	0.52	40.96	17.69	24.16	0.885
		Percentile	99.81	0.02	26.79	76.40	39.40	0.901
	day	Org.	99.50	0.42	65.83	30.47	40.49	0.960
		Percentile	99.92	0.05	87.54	81.69	84.28	0.997
	week	Org.	99.62	0.25	73.97	65.69	68.50	0.990
		Percentile	99.68	0.10	57.63	78.14	65.61	0.985
LANL	day	Org.	99.90	0.02	6.41	20.33	9.21	0.913
		Percentile	99.86	0.05	4.32	5.04	4.02	0.907

Table 6.6: User-based detection results by TPOT. F1 and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Data	Type	Temp. rep.	UAc	UFPR	UDR	UPr	UF1	UAUC
CERT R4.2	session	Org.	93.93	6.04	93.17	40.81	56.54	0.984
		Percentile	96.25	3.69	95.12	66.46	74.72	0.996
	day	Org.	93.88	6.24	96.59	40.70	57.08	0.989
		Percentile	97.77	2.33	100.00	69.52	80.90	1.000
	week	Org.	95.07	4.70	89.76	47.22	61.43	0.986
		Percentile	91.28	8.96	97.07	34.91	49.96	0.991
CERT R5.2	session	Org.	95.53	4.34	91.69	42.51	57.88	0.987
		Percentile	98.04	1.36	80.32	70.35	74.40	0.951
	day	Org.	96.25	3.78	97.23	47.45	63.61	0.988
		Percentile	98.77	1.27	100.00	73.95	84.91	1.000
	week	Org.	98.43	1.18	87.38	72.50	79.18	0.995
		Percentile	97.93	2.14	100.00	63.42	77.22	0.999
LANL	day	Org.	99.19	0.16	6.65	23.22	9.81	0.888
		Percentile	98.70	0.61	5.77	5.46	5.00	0.877

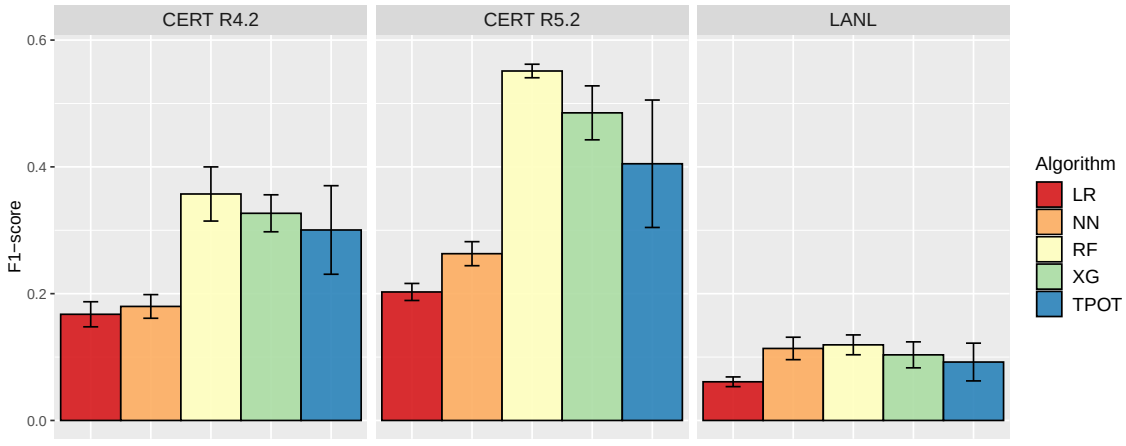


Figure 6.6: Instance-based F1-score by data and algorithms. Error bars show 95% confidence intervals.

6.4.3.1 TPOT results

As presented in Section 6.2.5, TPOT is employed to explore the possibilities of automatically optimizing the classifiers for better detection results. In this research, TPOT is run with default classifier configuration[5] and based on the following template: *Selector-Transformer-Classifier*. The template enforces a linear pipeline structure as described by the steps: Selector (feature selection, e.g. SelectPercentile, VarianceThreshold), Transformer (transforming features, e.g. FeatureAgglomeration, OneHotEncoder), and Classifier. In our preliminary experiments, TPOT has a tendency to produce complex stacked classifiers without improvements in performance. Thus, the template is used to reduce TPOT computation time and potentially provide more interpretable results [73]. In [73], the authors demonstrated that by enforcing type constraints with strongly typed GP, TPOT with templates significantly outperformed a tuned XGBoost model and standard TPOT implementation.

Similar to other learning algorithms in this chapter, TPOT's training data is provisioned based on *realistic* training condition (Section 6.3.1). For optimizing the classifier pipelines, in this research, TPOT is run with population size of 100 and up to 50 generations (with early stopping). Accuracy is selected for the scoring function and the results in each generation are measured using 5-fold cross-validation. The parameters are selected empirically.

As TPOT needs to evaluate a population of pipelines, each has multiple steps, and repeats that over many generations, TPOT training process requires significantly higher computing resources than individual learning algorithms, such as RF and NN. For example, in our experiments, TPOT may take up to three days to train on compute nodes with Intel Xeon E5-2683v4 CPU and 250GB of RAM, while RF can be trained in minutes on smaller nodes. Furthermore, in some cases, TPOT may crash due to excessive RAM requirement of Polynomial feature construction[6].

The following is an example of an optimized TPOT pipeline on CERT r4.2 week data with percentile representation: *ExtraTreesClassifier(ZeroCount(SelectPercentile(input_matrix, percentile=60)), bootstrap=False, criterion=entropy, max_features=0.7,*

[5]The configuration can be found at `https://github.com/EpistasisLab/tpot/blob/master/tpot/config/classifier.py`

[6]`https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html`

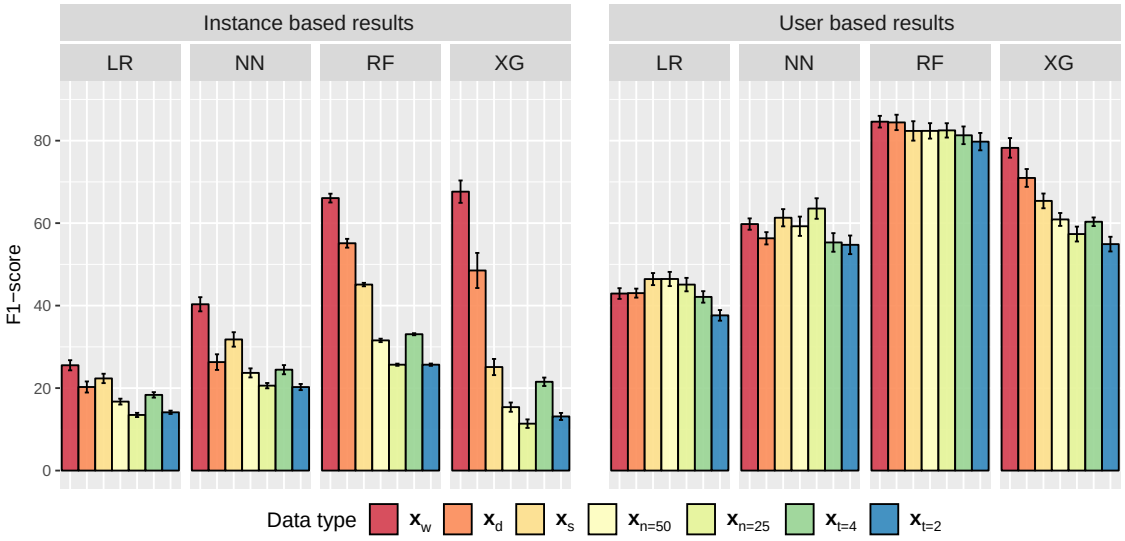


Figure 6.7: Instance-based and user-based F1-score by data types and ML algorithms. Error bars show 95% confidence intervals.

min_samples_leaf=1, min_samples_split=9, n_estimators=100)). Post-training, the optimized TPOT pipelines are evaluated on test data. The test results are presented in Tables 6.5 and 6.6. Figure 6.6 shows a comparison between the ML algorithms on the original extracted data from datasets by F1 score. It is apparent that TPOT in this case offer no improvements over RF and XG. Performing Wilcoxon signed-ranks test, the p values for comparison pairs RF – TPOT and XG – TPOT are 0.44 and 0.15, respectively. This shows the advantage of RF with proper parameterization (Section 6.3.2), as it is able to achieve optimized results without the computation overhead as via TPOT.

6.4.4 Results by Data Granularity Levels

As presented in Section 4.2.4, multiple levels of data granularity can be extracted on CERT R5.2. In this section, we investigate the impact on the performance of a ML based insider threat detection system under the *realistic* training condition.

6.4.4.1 Instance-based results

Instance-based results are shown in Table 6.7 and Figure 6.7. One noticeable trend overall is that ML algorithms' instance-based performances are degrading (w.r.t. IDR

Table 6.7: Instance-based results by data granularity levels and ML algorithm. F1 and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Data type	Logistic Regression					Neural Network					Random Forest					XGBoost				
	IFPR	IDR	IPr	IF1	IAUC	IFPR	IDR	IPr	IF1	IAUC	IFPR	IDR	IPr	IF1	IAUC	IFPR	IDR	IPr	IF1	IAUC
$\mathbf{x}_w$	1.40	53.77	16.94	25.53	0.93	0.41	45.23	38.03	40.33	0.91	0.00	49.54	99.39	66.07	0.99	0.14	63.37	74.10	67.63	0.99
$\mathbf{x}_d$	0.70	43.88	13.44	20.27	0.91	0.41	39.90	20.54	26.30	0.94	0.03	41.97	83.70	55.12	0.98	0.20	56.39	44.84	48.52	0.98
$\mathbf{x}_s$	0.27	26.49	19.82	22.34	0.85	0.14	29.28	37.20	31.80	0.87	0.02	31.54	81.98	45.10	0.90	0.31	31.76	22.09	25.11	0.92
$\mathbf{x}_{n=50}$	0.14	16.72	17.60	16.72	0.82	0.09	20.24	32.39	23.69	0.86	0.01	19.96	80.09	31.57	0.88	0.27	22.12	12.14	15.38	0.88
$\mathbf{x}_{n=25}$	0.07	11.22	17.84	13.49	0.80	0.04	14.75	39.61	20.59	0.85	0.01	15.56	78.98	25.67	0.88	0.20	15.74	9.24	11.37	0.86
$\mathbf{x}_{t=4}$	0.13	18.20	19.13	18.35	0.85	0.09	22.02	29.77	24.46	0.88	0.01	21.65	74.77	33.06	0.90	0.12	21.45	22.93	21.53	0.89
$\mathbf{x}_{t=2}$	0.08	12.53	16.65	14.13	0.84	0.05	15.77	32.02	20.26	0.87	0.01	16.13	67.86	25.68	0.89	0.16	16.33	11.26	13.13	0.87

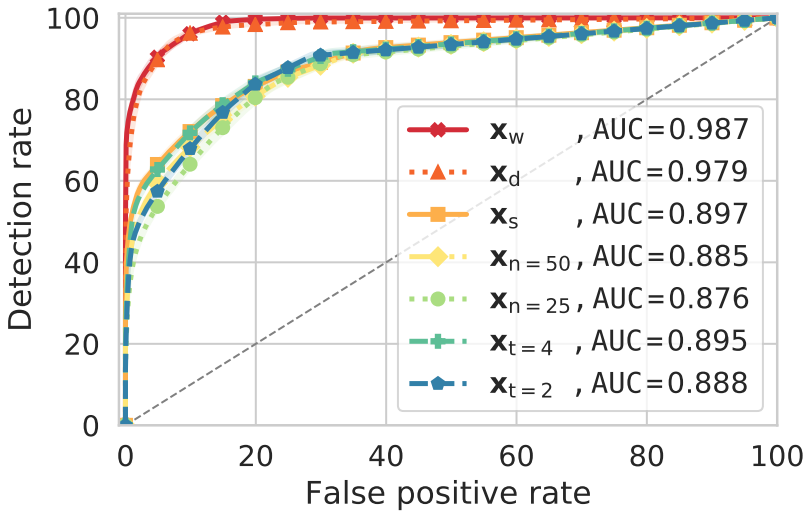
Table 6.8: User-based results by data granularity levels and ML algorithm. F1 and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Data type	Logistic Regression					Neural Network					Random Forest					XGBoost				
	UFPR	UDR	UPr	UF1	UAUC	UFPR	UDR	UPr	UF1	UAUC	UFPR	UDR	UPr	UF1	UAUC	UFPR	UDR	UPr	UF1	UAUC
$\mathbf{x}_w$	7.77	91.31	28.12	42.93	0.97	2.82	77	49	59.77	0.92	0.02	73.85	99.33	84.60	1.00	1	82	75.52	78.25	0.99
$\mathbf{x}_d$	7.75	91.54	28.17	43.03	0.98	4.13	86.15	41.96	56.32	0.98	0.43	81.62	88.04	84.42	1.00	1.97	85.46	61.03	70.96	0.99
$\mathbf{x}_s$	6.54	89	31.49	46.44	0.98	3.44	87.54	47.47	61.31	0.99	0.78	84.62	81.45	82.36	0.99	2.49	82.77	54.39	65.39	0.98
$\mathbf{x}_{n=50}$	6.44	87.77	31.72	46.46	0.97	3.71	86.46	45.43	59.24	0.98	0.54	80.15	85.89	82.37	0.99	2.51	75.15	51.49	60.90	0.96
$\mathbf{x}_{n=25}$	6.55	85.54	30.73	45.10	0.96	2.83	83.54	51.89	63.54	0.98	0.51	79.85	86.49	82.49	0.99	2.58	69.77	48.91	57.36	0.95
$\mathbf{x}_{t=4}$	7.83	89.54	27.6	42.11	0.97	4.56	88.38	40.55	55.32	0.98	0.79	83.08	80.67	81.30	0.99	2.3	71.54	52.5	60.33	0.95
$\mathbf{x}_{t=2}$	9.36	89.62	23.87	37.63	0.97	4.47	86.15	40.38	54.74	0.98	0.92	82.92	77.8	79.77	0.99	2.66	66.62	46.94	54.91	0.94

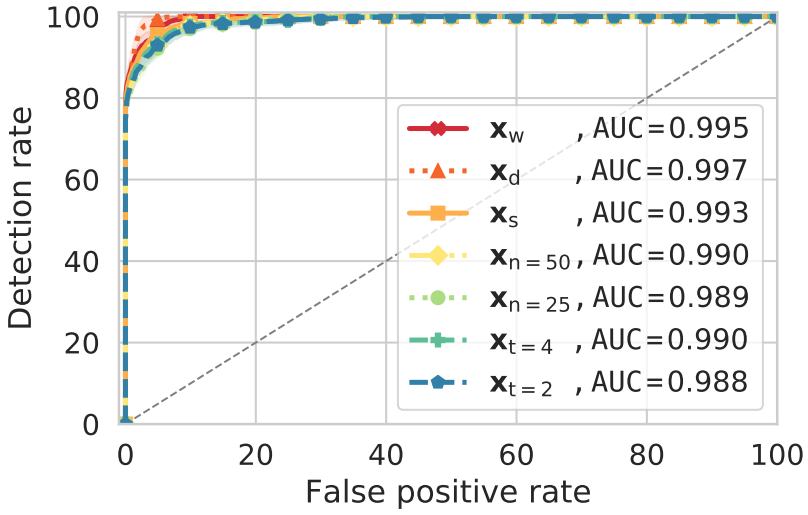
and IF1) by higher data granularity levels. Specifically, significant differences can be observed when comparing instance-based results (IF1) of different data granularity levels in almost all of the cases. For example, comparing RF's IF1 between user-week and user-session, or between user-session and sub-session T2 both yield $p = 9e^{-5}$. Figure 6.4 further demonstrates the observation, where AUCs are higher on user-week data than on user-session data. This can be explained through the amount of information embedded in each data instance of different data types (see Chapter 4), where coarse-grained data types, such as user-week and user-day, covers a longer period and summarize more behavioural information, i.e. user actions, than fine-grained data types, such as user-session and sub-session. Furthermore, much larger instance count and higher imbalanced data distribution in fine-grained data types (Table 4.2) also likely contribute to the observed degradation in instance-based results.

6.4.4.2 User-based results

Table 6.8 and Figure 6.7 and 6.8 show user-based results by the ML algorithms on different data granularity levels. In contrast to the trend observed in instance-based results, user-based results (UDR, UF1) are generally more robust on different data granularity levels. Other than the XG algorithm, no large changes (>5%) can be found between the measures of the data types in most of the cases. Furthermore, despite a relatively low proportion of detected malicious insider data instances (Table 6.7), the classifiers could learn to detect at least one malicious instance for most of the malicious insiders (80 to 90%). User-based result reporting also adjusts false positive rates considerably. For example, the NN achieves only a 0.14% IFPR, but 3.44% UFPR on user-session data. These observations show the shortcomings of simply reporting results per data instance rather than per user, where the former may not necessarily demonstrate a true estimation of the detector's capability in detecting malicious users. In practice, it seems that user-based metrics justify the use of fine-grained data types, such as session and sub-session. On these data types, despite lower IF1 and IDR than for coarse-grained data types, the UF1 and UDR performance remain the same. Fine-grained data types also provide further advantages to detection systems, such as allowing a faster response (investigated in Section 6.5).



(a) Instance-based



(b) User-based

Figure 6.8: Instance-based vs User-based ROCs and AUCs of RF on different data granularity levels

6.5 Analysis of Insider Threat Scenarios

As mentioned in Section 4.1.1, there are four distinct insider threat scenarios in CERT R5.2 dataset. In this section, we analyze the performance of the ML models on each scenario based on UDR, detection delay, and detection rate per detected malicious insider. The results by RF are shown in Figure 6.9.

6.5.1 Scenarios 1 and 4 – Data Exfiltration

It is readily apparent that insider threat scenarios 1 and 4 are reliably detected on most of the data types. Furthermore, these scenarios are detected quickly by the system with high confidence, particularly on session and sub-session data granularity levels. In these cases, the detection delays by RF are 3 and 0.23 hours (14 minutes) on average, and DR/DMI mostly above 70% and 48%, respectively. This suggests that the two scenarios can be easily detected using ML based systems. The insider threat scenarios depict data exfiltration attempts. In scenario 1, "user begins to log in after hours, using a USB, and uploading data to wikileaks.org", and scenario 4 shows another example: "user logs into another user's machine and searches for interesting files, emails them to his/her home email" [24]. Thus, it may be assumed that well-chosen / derived features (Chapter 4) contribute to the detection of behavioural shifts (abrupt changes in the user behaviour) in these scenarios. Some examples of these are in *after workhour* information, or in *PC* and *external email* information.

6.5.2 Scenario 2 – Intellectual Property Thief

Scenario 2 appears to be the hardest to detect using ML based detectors. The best results are obtained on user-day data, where 46% and 65% of malicious insiders are detected using RF and NN respectively. Detection also takes longer and with less confidence (DR/DMI $< 36\%$) – averaging 154 hours (NN) and 86 hours (RF) after the first malicious behaviour. Dataset description states that in this scenario, "user surfing job websites and soliciting employment from a competitor, and use a thumb drive to steal data" [43]. This implies less obtrusive actions than other scenarios. Furthermore, the insiders in scenario 2 possibly hid their malicious actions intentionally by performing them over a longer period of time (2 months on average,

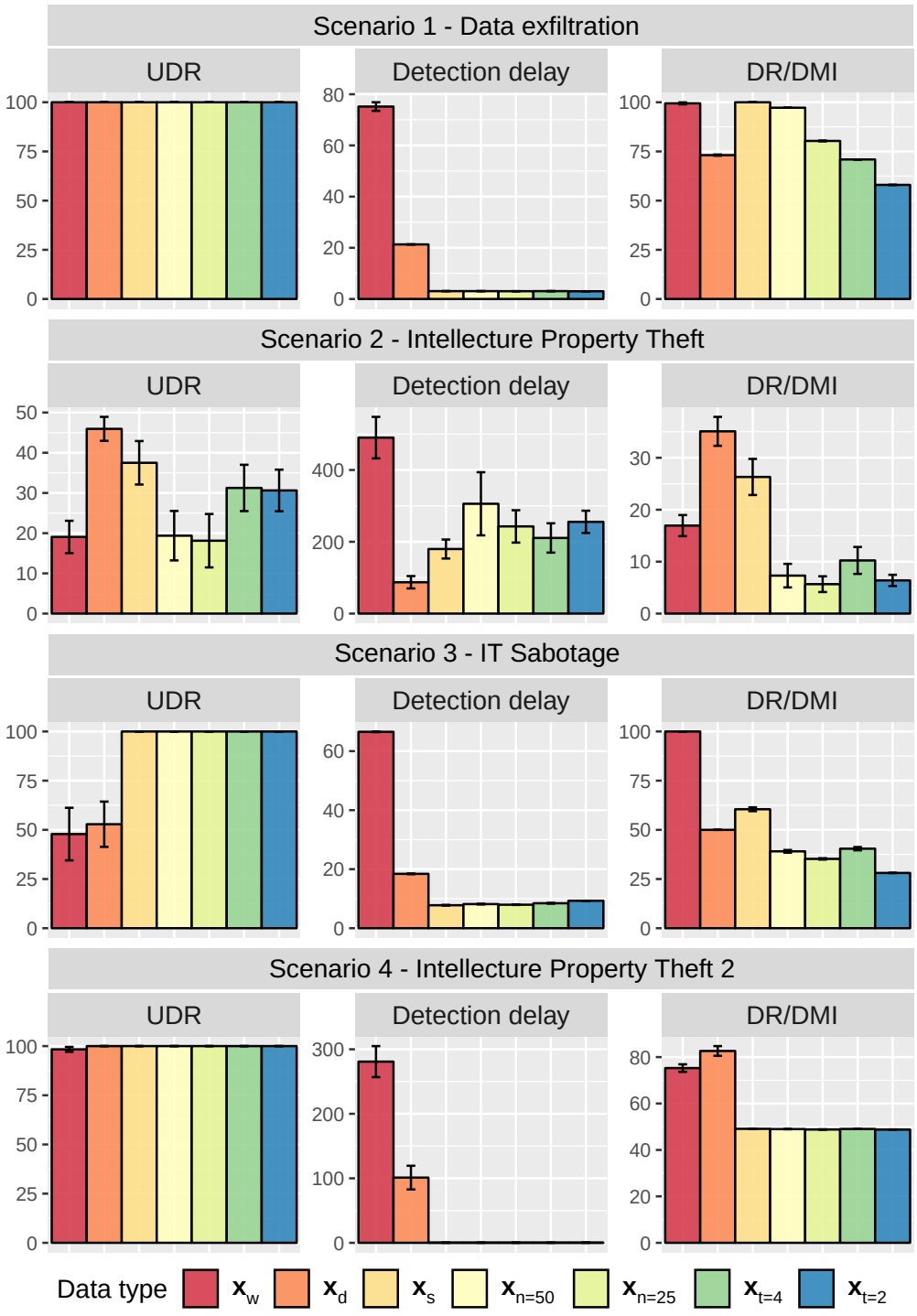


Figure 6.9: Insider threat scenario detailed results by RF. Error bars depict 95% confidence intervals. UDR: User-based Detection Rate, DR/DMI: detection rate per detected malicious insider. The unit of detection delay (DD) is hour. Note that for DD, lower is better.

Section 4.1.1), hence making it harder to detect.

6.5.3 Scenario 3 – IT Sabotage

Scenario 3 represents an IT sabotage behaviour, where "disgruntled system administrator uses keylogger to collect keylogs then log in as his supervisor to send out an alarming mass email" [24]. Based on Figure 6.9, it is apparent that this scenario can be detected easily with fine-grained data types (session and sub-session). On the other hand, coarse-grained data types result in lower performance under this scenario. This observation can be explained through distinct characteristics particular to this scenario, where a malicious user performs only a small number of actions (download keylogger, copy to USB) under their own account. Fine-grained data types are able to isolate those activities in terms of their short duration and PC-specific nature, leading to high detection rates. However, it should be noted that although this scenario can be detected effectively using fine-grained data types, detection delay and DR/DMI are not as good as that of scenarios 1 and 4, at approximately 8 hours and 40%, respectively.

6.5.4 Traitors vs Masqueraders

Depending on the modelling approach, scenarios 3 and 4 may be considered masqueraders [52, 100], as they involve insiders performing malicious actions on other users' machines. The two remaining scenarios are purely traitors' actions. Results show that traitors' actions – when performed in a less obtrusive manner and over a long period – are harder to detect (Scenario 2, Figure 6.9). On the other hand, both masquerader and traitor cases (scenarios 1, 3, and 4) can be detected easily using the proposed system if significant changes in user behaviours can be detected through the ML models.

6.5.5 Data Granularity Effects

Finally, with respect to granularity levels, fine-grained data types perform well with high malicious insider DR, low detection delay, and high DR/DMI in almost all situations. However, sub-session data types show no advantage over user-session data in all measures. On coarse-grained data types, user-day data has an edge in detecting

insider threat scenario 2. On the other hand, when user-week data is used, lower performance results, especially in terms of detection delay (> 280 hours and 360 hours in scenarios 4 and 2), despite the best instance-based performances (Table 6.7).

6.5.6 Test Results on Different Organizational Data

Table 6.9: User-based test results of the trained models on CERT R5.1 & CERT R6.2

Test data	Data type	Neural Network				Random Forest			
		UFPR	UDR	UPr	UF1	UFPR	UDR	UPr	UF1
CERT R5.1 (2000 users, 4 malicious insiders)	$\mathbf{x}_w$	3.56	70.00	3.75	7.12	0.05	55.00	75.42	62.16
	$\mathbf{x}_d$	5.87	63.75	2.06	3.99	0.68	65.00	21.81	30.44
	$\mathbf{x}_s$	5.14	76.25	2.84	5.48	0.84	77.50	22.15	32.68
	$\mathbf{x}_{n=50}$	4.50	75.00	3.28	6.28	0.58	75.00	31.09	41.62
	$\mathbf{x}_{n=25}$	3.57	75.00	4.33	8.15	0.67	75.00	28.13	37.97
	$\mathbf{x}_{t=4}$	5.78	80.00	2.71	5.24	0.96	76.25	20.31	29.97
	$\mathbf{x}_{t=2}$	6.18	81.25	2.55	4.94	1.17	75.00	13.87	22.90
CERT R6.2 (4000 users, 5 malicious insiders)	$\mathbf{x}_w$	0.05	10.00	26.10	13.39	2.75	52.00	4.43	7.79
	$\mathbf{x}_d$	2.03	40.00	2.97	5.45	2.99	62.00	5.96	10.10
	$\mathbf{x}_s$	18.57	59.00	0.34	0.68	16.02	57.00	0.38	0.75
	$\mathbf{x}_{n=50}$	17.58	63.00	0.38	0.75	6.18	61.00	1.36	2.65
	$\mathbf{x}_{n=25}$	16.46	57.00	0.38	0.75	6.51	60.00	1.17	2.30
	$\mathbf{x}_{t=4}$	22.56	59.00	0.26	0.52	8.16	60.00	0.92	1.81
	$\mathbf{x}_{t=2}$	26.29	60.00	0.21	0.43	8.38	59.00	0.87	1.71

As mentioned in Section 4.1.1, there are different versions of the CERT insider threat datasets. In this section, we train ML classifiers on CERT R5.2 and test them on CERT R5.1 and R6.2. CERT R5.1 simulates a similar organizational structure as CERT R5.2, with the same number of users, while CERT R6.2 has a different organizational structure and more users (4000), compared to CERT R5.2. Both CERT R5.1 and R6.2 simulate only one malicious user per insider threat scenario, significantly reducing the proportion of malicious data, which makes the detection task more challenging. Table 6.9 presents the results of these tests.

On CERT R5.1, it is apparent that the models perform well with low false alarm rates and 75% malicious insider detection rate. Typically on all data types, only the insider in scenario 2 is missed. It is noteworthy that while UDR and UFPR are similar to that observed on CERT R5.2, UPr and UF1 are lower, given the lower

number of malicious users in CERT R5.1. On the other hand, CERT R6.2 appears to present new challenges, possibly due to a different organizational structure. Notably, a new insider threat scenario in CERT R6.2 (scenario 5) is undetected. This scenario depicts behaviours of "a user decimated by layoffs uploads documents to Dropbox, planning to use them for personal gain" [24]. Not only does this scenario represent a novel malicious behaviour to the trained model, but it also shows less obtrusive actions than the remaining four scenarios.

On data granularity, user-session shows lower performance than the other data types from CERT R6.2. This suggests that on a different organization with different user behaviour models, a session of user data may represent a different course of actions. In this case, a more aggregated data type, such as user-day and/or user-week may demonstrate better results.

Overall, results in this section indicate that models trained for insider threat detection in an organization can only be used as an initial detection step in a different environment. Specific models need to be re-trained from ground up or evolved from existing models for better accuracy. Furthermore, anomaly detection is needed to identify novel malicious behaviours.

6.6 Discussion

6.6.1 DR - FPR trade-off

Throughout the experiments and results presented above, it is clear that there is a trade-off between the ability to detect malicious behaviours (DR) and maintaining a low false alarm rate (FPR). RF achieves the lowest FPRs and good DRs in most of the cases, explaining the good performances under the F1-score. Wilcoxon signed-ranks tests between RF and LR, NN, XG return p values 0.015, 0.015 and 0.03, respectively. This provides strong evidence against the null hypothesis that the methods are equivalent with respect to IF1. NN and LR post higher malicious DR at a cost of lower precision. Table 6.7 shows that instance-based false alarm rates are approximately 5/6 for LR, and 2/3 for NN. In comparison, the false alarm rate is only 1/5 for RF, both for instance-based and user-based. On the other hand, XG shows good results on week data and in the *idealistic* condition only, which suggests

that it may not be a suitable candidate for insider threat detection, under limited ground truth conditions.

On data granularity levels, the experiments show that data extracted in user-session format allows good detection performance and low delay in most cases. Specifically, using RF classifier, 85% of malicious users are detected at a low cost of 0.78% false malicious user alarms. Classifiers using RF on user-session data also achieve the lowest delay in detecting most of the insider threat scenarios. User-day data may provide an advantage over user-session data in detecting scenario 2, while user-week data shows promising results on unseen data of different organizational structures (Table 6.9). On the other hand, pre-processing user data into sub-sessions does not improve the results w.r.t. session data in any metrics. This is probably due to the lower amount of information embedded in each instance of sub-session data.

6.6.2 Data Imbalance Problem

As the data is highly imbalanced (Table 4.2), in this section we discuss the effect on ML algorithms. In this work, training the ML algorithms as binary classifiers, in which minority malicious classes are combined to a single positive class, may help to reduce the negative impacts of data imbalance. Furthermore, good results obtained by RF, as shown in 6.4, may be attributed to its resistance to data imbalance [58].

We further examine the ML algorithms performances with oversampling techniques. Results with random oversampling (increase positive class to 20% of training data) are shown in Table 6.10. Note that other oversampling techniques, such as SMOTE [25], and different training settings yield very similar results. Overall, it seems that the DR-FPR trade-off is maintained, and the results obtained with oversampling is similar to what achieved with original training data composition with an adjusted decision threshold (Figure 6.4 and 6.8).

Table 6.10: User-based results with Oversampling on user-session data

Algorithm	**UFPR**	**UDR**	**UPr**	**UF1**
LR	32.55	100.00	6.90	12.91
NN	17.26	95.08	14.03	24.44
RF	1.36	86.46	72.00	77.63
XG	5.04	88.31	37.83	52.72

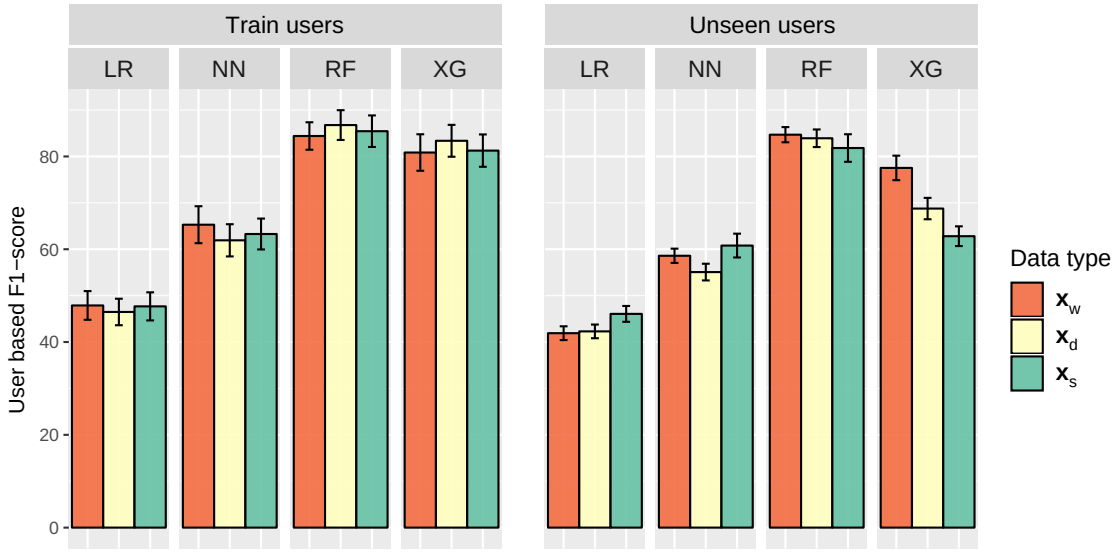


Figure 6.10: User-based F1-score on test data of "normal" train users and unseen users. Error bars show 95% confidence intervals.

6.6.3 Overfitting and Effect of Training Data

We perform permutation test [89] in order to validate the trained classifiers. Essentially, permutation test is a null hypothesis test to evaluate if the classifier found a significant class structure, that is, a real connection between the data and the class labels [89]. The test is done by obtaining classification results with a random permutation of the labels. The process is repeated (100 times in this case) to obtain a p value. Test results in all cases return p-values less than 0.01 w.r.t F1, indicating the ML algorithms have learned to recognize meaningful classifiers from training data without overfitting.

We further explore the effect of training data by comparing test results on test data of "normal" training users (user set U_{train}), and of unseen test users (user set U_{unseen}, $U_{unseen} \cup U_{train}$ = All test users). Recall that in the *realistic* training condition, data from only a limited subset of users are included for training. U_{unseen} is the set of users whose data in the first 37 weeks are not used in training the ML models. Figure 6.10 shows user-based F1-score on test data of the two user sets. Overall, the ML algorithms are able to generalize well, where UF1 is only slightly higher on U_{train} than on U_{unseen} (p=0.012 w.r.t UF1). RF shows the most similar performance

between U_{unseen} and $U_{results}$ ($p = 0.66$ w.r.t UF1), which suggests that it is more robust to overfitting problem than the remaining algorithms.

6.6.4 Feature Analysis

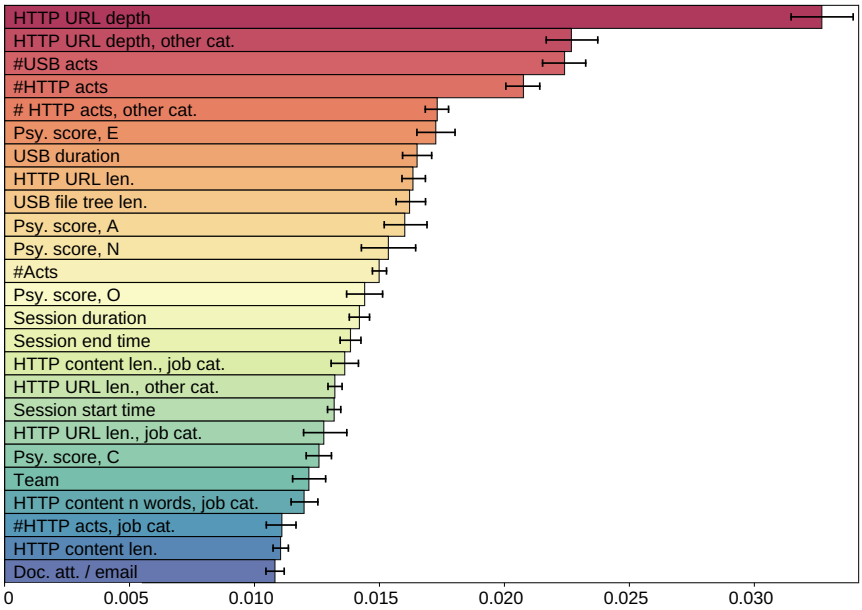
We perform feature analysis based on the feature importance output of RF, which is calculated using Gini impurity, and information gain. Definition of the metrics can be found in [104]. Figure 6.11 illustrates the 25 most important features identified by the two metrics in user-session data.

Figure 6.11a shows that HTTP and USB features are important to RF classifiers, specifically URL depth, and in job category. This is expected, considering the malicious actions in the insider threat scenarios (Section 6.5), where web and USB are two common means for carrying out malicious actions, such as copying files, and searching job offers. Features related to HTTP category "other" are important too. This suggests the use of a more detailed HTTP categorization model for a better description of user activities might be appropriate. Furthermore, the importance of team and psychometric scores indicate the usefulness of the constructed user profiles (Section 4.2). On the other hand, information gain shows a slightly different picture, Figure 6.11b, where features describing document related activities are the most important ones.

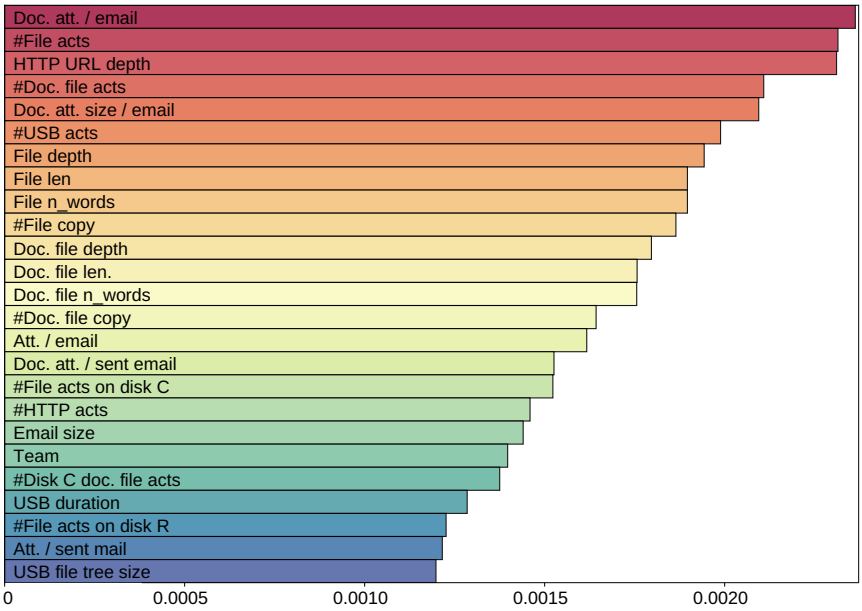
6.7 Summary

In this chapter, a ML based system for insider threat detection in networked systems of organizations is presented. The chapter benchmarks four different ML algorithms, namely LR, NN, RF, and XG, and automatic optimization of classifier using TPOT. This is done on multiple levels of data granularity, limited ground truth, and different training scenarios in order to support cyber-security analysts in detecting malicious insider behaviours in unseen data. Evaluation results show that the proposed system is able to successfully learn from the limited training data and generalize to detect new users with malicious behaviours. The system achieves a high detection rate and precision, especially when user-based results are considered.

Among the ML algorithms, RF clearly outperforms the other algorithms, where it achieves high detection performance and F1-score with low false positive rates



(a) Feature importance by RF (with 95% CIs)



(b) Feature importance by Information gain

Figure 6.11: Feature importance in user-session data

in most of the cases. On the other hand, NN allows slightly better insider threat detection performance at a cost of higher false alarm rates. On data granularity, user-session data provides high malicious insider detection rates and minimum delay. User-day data shows slightly better performance on detecting a particular insider threat scenario, aka scenario 2 - intellectual property theft.

The results in this chapter show the advantages of supervised ML in learning from limited ground truth (e.g. Chapter 5) to predict future insider threat cases at very high precision. This creates the potential for employing supervised ML in assisting analysts to quickly detect malicious insider cases, while complimenting anomaly-based detection, which is useful for zero-day attacks.

Chapter 7

Semi-supervised Learning for Insider Threat Detection

Many challenges from insider threat detection come from the fact that the ground truth is very limited and costly to acquire. This chapter presents a semi-supervised learning approach to insider threat detection, in order to maximize the effectiveness of the limited availability of the labelled training set. We employ three machine learning methods under different real-world conditions. These include obtaining the initial ground truth training data randomly or via a certain type of insider malicious behaviour or by anomaly detection system scores. Evaluation results show that the approach allows learning from very limited data for insider threat detection at high precision. 90% of malicious data instances are detected under 1% false positive rate. The results are also comparable to that of supervised learning[1].

The chapter is organized as follows. Section 7.1 provides an overview of the semi-supervised learning based approach to insider threat detection. The semi-supervised ML algorithms and label availability schemes employed in this work are presented in Section 7.2 and Section 7.3, respectively. Section 7.4 details the experimental settings, while the evaluation results are presented in Section 7.5. Finally, conclusions are drawn in Section 7.6.

7.1 Overview of the Semi-supervised Learning based Approach to Insider Threat Detection

Figure 7.1 illustrates the overview of the proposed approach for insider threat detection using semi-supervised learning. Based on the process in Chapter 4, numerical data with different granularity levels and temporal representations is extracted from data sources. The following datasets are employed in this chapter: CERT R4.2, CERT R5.2, and LANL. Based on the available data, firstly, anomaly detection steps

[1]Parts of this chapter have been presented at 2021 IEEE Security and Privacy Workshops [72] (© 2021 IEEE)

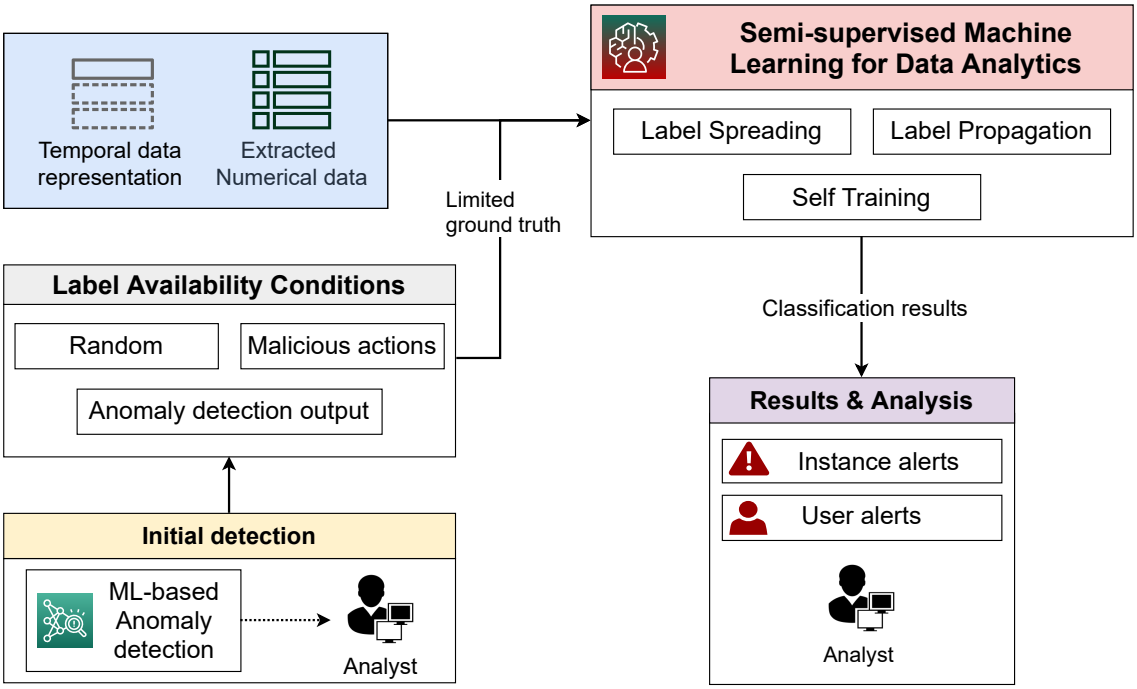


Figure 7.1: Overview of the proposed system for semi-supervised insider threat detection. Limited ground truth, as defined by label availability conditions, in combination with a large amount of unlabelled data is used for training the detection system.

or manual investigation is performed to obtain the initial labelled set of confirmed malicious and normal users' data. Then, based on the limited ground truth and extracted data, semi-supervised learning algorithms are employed in the next step to improve detection performance. Presented in Section 7.2, semi-supervised learning is a ML approach that permits large amounts of unlabelled data to be harnessed in combination with typically smaller sets of labelled data, in order to improve the outcome [110]. Conceptually, semi-supervised learning falls between unsupervised learning, which does not need labelled training data, and supervised learning, which trains using only labelled training data. This motivates its use in cyber-security applications, as obtaining a fully labelled training dataset is prohibitively costly or infeasible in many real-world conditions [21].

In this chapter, the focus is on using semi-supervised learning to improve upon the limited labelled data obtained from the initial detection step for identifying unknown malicious insiders. To achieve this, in Section 7.3, we detail the explored approaches to present a limited labelled training set for semi-supervised learning algorithms using three different methods.

7.2 Semi-supervised Learning Methods

In this research, we employ three popular semi-supervised learning methods from the literature [110]: Label Propagation (LP), Label Spreading (LS), and Self-Training (ST).

7.2.1 Label Propagation

Label Propagation (LP) [119] is a graph-based method, which propagates given labels from a (typically small) subset of the data points to the whole dataset. In LP, label assignments $\hat{y}_i \in \mathbb{R}$ are obtained by propagating the estimated label at each node to its neighbouring nodes based on the edge weights. A connection weight between node v_i and v_j is denoted by W_{ij}. The transition matrix A, $A_{ij} = W_{ij} / \sum_{v_k} W_{ik}$, allows the calculation of the new estimated label at each node as the weighted sum of the labels of its neighbours: $\hat{y}_i = A^T \cdot \hat{y}$. In this work, the radial basis function (rbf) is used as graph kernel for connection weights.

At the beginning, $\hat{y}$ is set to random for unlabelled data points and ground truth (known true labels) for the labelled points, respectively. Using this, the label propagation algorithm performs the two following steps repeatedly until converging (guaranteed) to obtain the final predictions [110]:

i) Propagate labels from each node to the neighbouring nodes ($\hat{y} = A^T \cdot \hat{y}$).

ii) Reset the predictions of the labelled data points to the corresponding true labels.

7.2.2 Label Spreading

Label Spreading (LS) [118] is also a graph-based method with similar principles as Label Propagation. It was proposed to deal with two main drawbacks of Label Propagation: reducing label noise and regulation of influence of high degree nodes over the graph. To address the first issue, instead of assigning all the true labels to the labelled data points, LS relaxes the number of true labels that need to be assigned, and uses the squared error between the true label and the estimated label in optimization. The second issue is addressed by using a modified version of the original graph and normalizing the edge weights via normalized graph Laplacian matrix ($\tilde{L}$). LS admits a closed-form solution and is a relatively efficient iterative approach to optimization [110]. In that, the label vector $\hat{y}_{t+1}$ at iteration $t+1$ is calculated based on that at iteration t, using the update rule $\hat{y}_{t+1} = \alpha \cdot \tilde{L} \cdot \hat{y}_t + (1-\alpha) \cdot y$, where y equals 0 for the unlabelled data points, and α balances the importance of the calculated label vector $\hat{y}$ and the base label vector y.

7.2.3 Self-Training

First proposed in [115], Self-Training (ST) is a pseudo-labelling approach, which trains a supervised classifier iteratively using both the given true labels and the predicted labels (with high confidence) from previous iterations of the algorithm. Using only the labelled data at the beginning of the self-training procedure, a base classifier is trained and then used to obtain predictions for the unlabelled data points. Based on the classification result, a set of the most confident predictions are added to the labelled dataset as pseudo-labels. The process is repeated with the supervised classifier retrained and new pseudo-labels obtained, until all samples have labels or the

maximum number of iterations is reached. In this chapter, self-training is employed with four different classifier bases presented in Chapter 6 (LR, NN, RF, XG). Thus, in the following, self-training is abbreviated with its base classifier information, e.g. ST(RF).

Self-training variations have been applied extensively to computer vision problems, especially with the use of deep learning, and demonstrated state-of-the-art performances [114]. In cyber-security, ST variations are successfully used in intrusion detection and attack detection tasks [6, 98].

7.3 Label Availability for Semi-supervised Learning

In this chapter, we explore different label availability conditions to semi-supervised learning. In real-world cyber-security applications, the initial labelled set may not come from a random subset of data as assumed in many semi-supervised learning settings. For example, only some malicious actions are discovered by the analyst and labelled for further analysis, while other kinds of malicious actions remain undetected. In other cases, the results from unsupervised learning approaches for anomaly detection can be used as the basis for investigation. This in return provides an initial labelled data for training. Thus, the following conditions for label availability are examined in this paper:

(i) By random: a randomly selected small subset of training data (instances) is labelled for training semi-supervised learning algorithms. This assumes that all users are equally suspicious.

(ii) By random users: in this scenario, similar to (i), we assume that data from a small number of randomly selected *users* is labelled (equivalent to a small subset of users being sampled for labelling from the training partition). As in real-world practice, labelling data from a subset of users may be easier than doing so on the same amount of random data from all users [111].

(iii) By malicious action type: This case assumes that a certain type of malicious action is detected and labelled for training, in combination with a random set of normal data. This condition examines whether learning methods can detect

and generalize to other malicious behaviours. Such a scenario recognizes that it might in practice be easier to recognize certain types of malicious action and therefore manual labelling practices might be biased to certain behaviours.

(iv) By anomaly detection scores: This case uses anomaly detection scores (e.g. Chapter 5) to label the unknown data. To this end, a percentage of training data with the highest (confidence) anomaly scores are provided to semi-supervised learning.

(v) As a variation of (iii), we assume the labelled data is obtained from all data points of a small set of *users* with the highest anomaly scores.

In the following, the label availability conditions are referred to based on the above enumeration. For condition (iii), a number is used in conjunction to indicate the insider threat scenario (see Section 4.1.1) included in the initial label set. For example (iii-1) means that scenario 1 represents the malicious action type as labelled in the training data.

7.4 Experiment Settings

In this chapter, experiments are performed on CERT R4.2, R5.2, and LANL datasets to evaluate semi-supervised learning for insider threat detection. We adopt realistic settings as characterized in the previous chapter for training data, where data was obtained from only a restricted set of users over a given time period is used (see Section 6.3.1). Specifically, for CERT R4.2, data from 200 users in the first half of the dataset's duration (36 weeks) is used to train the algorithms. Similarly, on CERT R5.2 and LANL data, the user limits are 400 (in 2000 users) and 2000 (in 11814 users), while duration limits are first 37 weeks (50%) and first 12 days (in 30 days), respectively.

The amount of labels (ground truth) available is selected from the training data. In the main experiment, *20%* of the training data is labelled, based on the conditions provided for semi-supervised learning (Section 7.3). In the case of CERT R4.2, the labelled data amount is equivalent to that of 40 users (from the dataset of 1000 users). The trained classifiers are then used to test on the second half of the dataset to detect

unknown malicious insiders. Results on the unlabelled portion of training data are also reported.

In the following experiments, unless specified otherwise, we use Autoencoder anomaly scores as the basis for label availability (iv) and (v), based on its results in Chapter 5. Additional experiments in this chapter also explore the effect of the amount of initial labelling, the type of unsupervised learning algorithm assumed for anomaly scores, and the sensitivity of data granularity on semi-supervised learning detection performance. The experiments are repeated 10 times in each setting, and the averaged results are reported.

We implemented the data pre-processing and analysis steps using Python 3. Implementations from Scikit-learn [93] are used. For LP and LS, the RBF kernel is used with $\gamma = 10$. The parameters are chosen empirically. Finally, similar to previous chapters, the insider threat detection performance is measured using FPR, DR, and ROC AUC metrics, both instance-based and user-based (see Section 6.3.3).

7.5 Evaluation Results

In the main experiment, based on results from previous chapters, we employ weekly data from CERT R4.2 and R5.2 datasets in percentile representation, and LANL dataset in original extracted representation. The main experimental results are presented in Table 7.1 – Instance-based, and Table 7.2 – User-based. The tables report AUCs on test datasets, and DR on test data at different critical FPR levels, based on the ground truth availability scenario and learning algorithm. The AUC is also reported on unlabelled train data in Table 7.1. The tables report the best results achieved by self-training with one of the base classifiers (in LR, NN, RF, XG). It is noteworthy that that RF is the best candidate on most of the cases on CERT datasets, while XG gives the best performances on LANL dataset.

Overall, the results achieved using Self-Training are the most promising, on all data availability conditions. For example, on day data, using top anomaly scores for initial labels on CERT R4.2, this combination was able to detect 89% and 99% of the malicious users in CERT R4.2 test data at UFPRs of only 0.01% and 1% (Table 7.2). The results demonstrate the ability of semi-supervised learning to improve upon the available limited ground truth or anomaly scores to detect insider threats.

Table 7.1: Detection results (AUC and DR) of the semi-supervised learning algorithms under different data availability conditions. DR and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

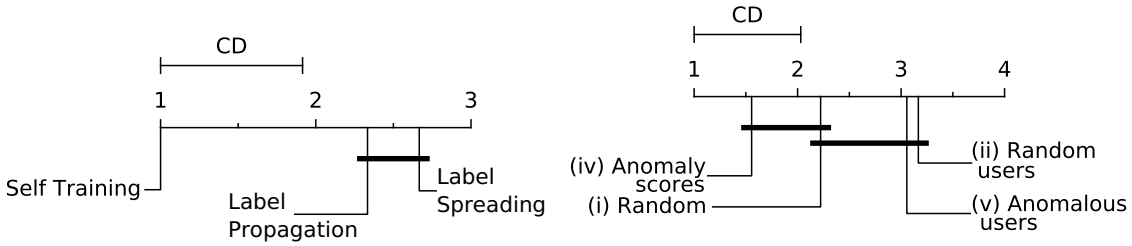
Ground-truth availability	Algorithm	CERT R4.2					CERT R5.2					LANL				
		Unlabelled train AUC	Test AUC	Test DR			Unlabelled train AUC	Test AUC	Test DR			Unlabelled train AUC	Test AUC	Test DR		
				0.1% FPR	1% FPR	5% FPR			0.1% FPR	1% FPR	5% FPR			0.1% FPR	1% FPR	5% FPR
(i) Random	LP	0.771	0.759	10.06	21.74	38.14	0.706	0.634	5.92	10.79	20.42	0.797	0.720	5.04	12.69	28.66
	LS	0.727	0.688	9.52	18.44	33.77	0.639	0.592	5.27	9.35	16.34	0.804	0.717	5.46	12.94	30.08
	ST	0.935	0.948	27.25	45.09	71.14	0.928	0.946	33.97	55.21	73.94	0.949	0.890	15.29	41.34	65.38
(ii) Random users	LP	0.757	0.709	8.20	18.02	32.40	0.691	0.605	3.27	7.38	16.20	0.817	0.721	5.46	12.44	26.05
	LS	0.693	0.659	8.08	14.37	28.98	0.623	0.569	3.44	7.89	14.14	0.827	0.727	5.38	12.52	26.97
	ST	0.864	0.907	18.38	33.23	61.92	0.851	0.895	27.94	42.79	61.58	0.948	0.895	10.92	34.71	61.85
(iii-1) Scenario 1	LP	0.604	0.628	14.73	16.95	23.29	0.649	0.591	6.76	11.07	18.56	On LANL dataset, results with ground-truth availability scheme (iii) are not available, as the dataset provides no details on malicious behaviours .				
	LS	0.405	0.538	11.02	15.39	21.08	0.508	0.553	5.46	7.83	13.10					
	ST	0.716	0.791	23.29	25.39	38.86	0.708	0.698	10.14	10.65	29.75					
(iii-2) Scenario 2	LP	0.606	0.745	6.11	16.11	32.46	0.560	0.567	0.25	3.18	10.06					
	LS	0.380	0.665	4.97	12.75	28.38	0.546	0.534	0.39	2.51	8.54					
	ST	0.926	0.958	24.85	40.90	66.53	0.874	0.893	12.45	20.37	39.01					
(iii-3) Scenario 3	LP	0.564	0.623	3.47	12.46	24.43	0.536	0.540	4.14	7.58	13.04					
	LS	0.414	0.443	2.81	4.49	8.74	0.450	0.521	3.58	5.72	10.14					
	ST	0.709	0.701	2.99	4.67	16.17	0.690	0.634	2.25	7.94	18.99					
(iv) Top anomaly scores	LP	0.657	0.639	13.11	18.74	26.29	0.651	0.581	3.75	6.39	11.77	0.779	0.675	9.83	14.29	27.31
	LS	0.638	0.618	14.25	19.22	25.87	0.520	0.556	4.51	6.68	12.06	0.783	0.716	7.31	16.30	30.34
	ST	0.974	0.978	55.39	62.87	83.11	0.949	0.982	62.31	74.51	86.99	0.980	0.965	22.52	49.83	80.84
(v) Users with highest anomaly scores	LP	0.646	0.684	15.27	18.74	28.14	0.639	0.703	13.10	23.21	34.90	0.787	0.711	2.18	6.05	22.61
	LS	0.636	0.658	12.46	17.96	27.01	0.601	0.617	9.49	16.11	22.28	0.811	0.709	2.1	5.80	21.51
	ST	0.789	0.824	22.46	27.31	42.57	0.785	0.838	12.42	37.18	51.46	0.932	0.848	5.29	22.77	54.71

Table 7.2: User-based detection results (UAUC and UDR) of the semi-supervised learning algorithms under different data availability conditions. DR and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Ground-truth availability	**Alg.**	**CERT R4.2**				**CERT R5.2**				**LANL**			
		Test UAUC	Test UDR (at UFPR)			Test UAUC	Test UDR (at UFPR)			Test UAUC	Test UDR (at UFPR)		
			0.1%	1%	5%		0.1%	1%	5%		0.1%	1%	5%
(i) Random	LP	0.8093	15.12	30.24	45.85	0.6789	9.54	20.92	33.08	0.7153	3.80	11.39	28.23
	LS	0.7719	15.61	31.95	44.39	0.6538	11.38	23.23	30.00	0.7141	3.54	14.05	27.34
	ST	0.9783	58.05	68.05	84.39	0.9846	50.92	67.23	88.62	0.8199	8.10	29.87	52.15
(ii) Random users	LP	0.7297	10.73	25.61	39.27	0.6292	5.38	12.46	20.15	0.7046	3.04	12.41	24.30
	LS	0.6864	14.15	24.63	35.61	0.6186	8.62	12.31	22.00	0.7065	1.77	10.89	25.19
	ST	0.9391	40.00	46.59	63.66	0.9522	43.85	54.77	72.00	0.8386	5.70	23.42	50.00
(iii-1) Scenario 1	LP	0.7068	36.83	50.24	51.95	0.6643	20.92	25.54	30.15	On LANL dataset, results with ground-truth availability scheme (iii) are not available, as the dataset provides no details on malicious behaviours .			
	LS	0.6898	35.12	41.46	46.34	0.6152	20.46	23.85	30.31				
	ST	0.9122	51.46	53.17	62.20	0.8389	31.54	35.38	41.38				
(iii-2) Scenario 2	LP	0.5483	0.00	9.27	22.44	0.5141	0.00	0.00	4.77				
	LS	0.5607	0.24	6.34	21.46	0.4869	0.15	1.08	4.31				
	ST	0.8893	25.61	33.90	38.54	0.8921	17.38	21.85	28.46				
(iii-3) Scenario 3	LP	0.7386	10.00	10.00	21.95	0.6401	10.77	18.00	29.85				
	LS	0.4348	10.00	10.24	11.71	0.5658	11.69	14.92	22.15				
	ST	0.7533	10.24	12.44	14.88	0.7633	10.77	11.23	24.46				
(iv) Top anomaly scores	LP	0.7469	0.00	46.83	54.15	0.5945	14.31	18.00	23.69	0.7180	4.18	15.32	25.19
	LS	0.7728	7.07	48.29	57.32	0.5972	12.92	20.77	27.69	0.7457	2.28	12.53	27.22
	ST	0.9997	89.27	98.78	100.00	0.9997	91.38	100.00	100.00	0.9147	10.00	38.61	62.53
(v) Users with highest anomaly scores	LP	0.7943	38.05	50.24	55.37	0.7741	21.23	36.31	48.92	0.6917	1.39	4.94	19.37
	LS	0.7839	38.05	47.32	52.68	0.7272	17.08	29.69	44.77	0.6874	1.39	3.67	13.54
	ST	0.9267	57.32	59.27	66.59	0.8896	14.15	42.62	52.77	0.7700	3.29	14.56	36.46

Comparing results by the semi-supervised learning algorithms, it is apparent that LP and LS lag behind ST under all conditions. Friedman test on user-based AUCs return $p = 9 \times 10^{-5}$, and critical difference diagram obtained using the posthoc test is shown in Figure 7.2a. The test confirms the observations on algorithms' performances. This can partially be explained by the learning mechanisms of the methods. LS and LP assume neighbouring relationships to label unlabelled data. Hence, their effectiveness is improved on conditions (i)-(ii) – randomly selected training set, in that the labelled training set is more likely to be evenly distributed throughout the data. On other conditions (iii)-(v), the initial labelled training set may have the tendency to focus on specific regions of data (describing specific malicious actions, or with high anomaly scores). In these cases, neighbourhood information as in LP and LS might not be effective in labelling other regions, which explains why LP and LS fail to capitalize on the information provided by anomaly detection scores (Table 7.1). In Section 7.5.2, we explore the visualization of training data to further explain the

results.



(a) Ranking of the semi-supervised algorithms (b) Ranking of label availability schemes

Figure 7.2: Critical Difference (CD) diagrams of results by label availability schemes and semi-supervised algorithms.

On three training conditions using different insider threat scenarios – (iii-1), (iii-2), and (iii-3), the AUC is higher only when scenario-2 is used as initial labels. We think this is due to a greater amount of malicious data from this scenario (Table 4.2), as well as different data describing those types of malicious actions (see Section 4.1.1). This also shows a weakness of this type of detection system, where supervised and semi-supervised learning may find it harder to generalize from previously seen attacks to detect novel unseen attacks. Unsupervised learning based anomaly detection may be better suited for this case.

Using condition (iv) – top anomaly scores, ST achieves the best AUCs and DRs in most cases (Table 7.1). This indicates that learning upon anomaly detection results is beneficial in this case, as high anomaly scores may be indicative of malicious and unusual activities. On the other hand, employing data from users with the highest anomalous scores – condition (iv) – offers no further advantages in almost all of the cases. As label availability scheme (iii) is not available for LANL dataset, performing Friedman test on user-based AUCs between label availability schemes (i), (ii), (iv), and (v) returns $p = 0.0003$ (rejecting the null hypothesis), and critical difference diagram obtained using the posthoc test is shown in Figure 7.2b.

7.5.1 Compare to Anomaly detection and Supervised learning

Table 7.3 shows a summary of test results, in previous chapters, from unsupervised learning (using Autoencoder), and supervised learning (using Random Forest) for comparison purposes. The comparison results are obtained under the same setting

Table 7.3: Anomaly detection and classification performances (AUC) for comparison

Data	Unsupervised learning		Classification	
	IAUC	UAUC	IAUC	UAUC
CERT R4.2	0.874	0.949	0.983	1
CERT R5.2	0.888	0.918	0.983	1
LANL	0.908	0.829	0.964	0.926

of training data, but with fully unlabelled or labelled training set, depending on the learning algorithm. We note that the results as presented in Table 7.3 are state-of-the-art for corresponding learning methods on the CERT dataset [69,71]. From the tables, it is clear that ST shows much better detection performances than unsupervised learning and LS or LP. Furthermore, ST results are approaching that of classification approaches in all cases.

7.5.2 Visualizing the Training Data

To understand the differences in performance of the semi-supervised learning algorithms, we perform data visualization using t-SNE [109]. t-SNE is a popular tool for visualizing high-dimensional data distribution. Figures 7.3 and 7.4 show training data distribution (for week data) and the initial labelled set selected randomly (Figure 7.3) or based on top anomaly scores (Figure 7.4). It is clear that when the initially labelled data subset is selected randomly, neighbour-based methods like LS and LP are performing better than under other label availability scenarios, as the labelled data is more representative of the training data in this case (Figure 7.3).

The figures also show that the malicious data from the three threat scenarios are quite different from each other. This explains the lower detection performances when only one of the scenarios is presented as the initial labelled set.

When using anomaly scores, Figure 7.4 shows that data points with top anomaly scores are not representative of the whole data. However, as the selection focuses on the most anomalous data points, malicious data is more likely to be included in the initial labelled set in this case. This, in combination with the ability to partition the feature space of tree-based classifiers like RF and XG [18], enables self-training to learn from the anomaly scores for insider threat detection and perform better on test

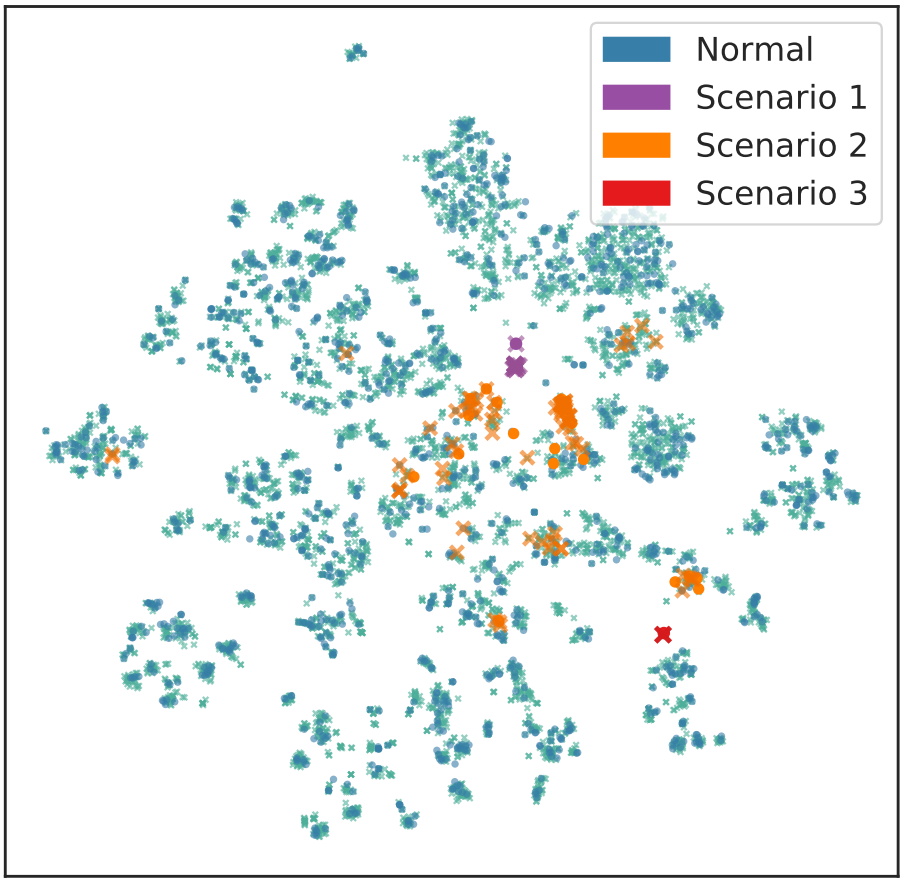


Figure 7.3: t-SNE visualization of training data with labelled set selected randomly. • denotes selected labels, while **x** denotes unlabelled training data.

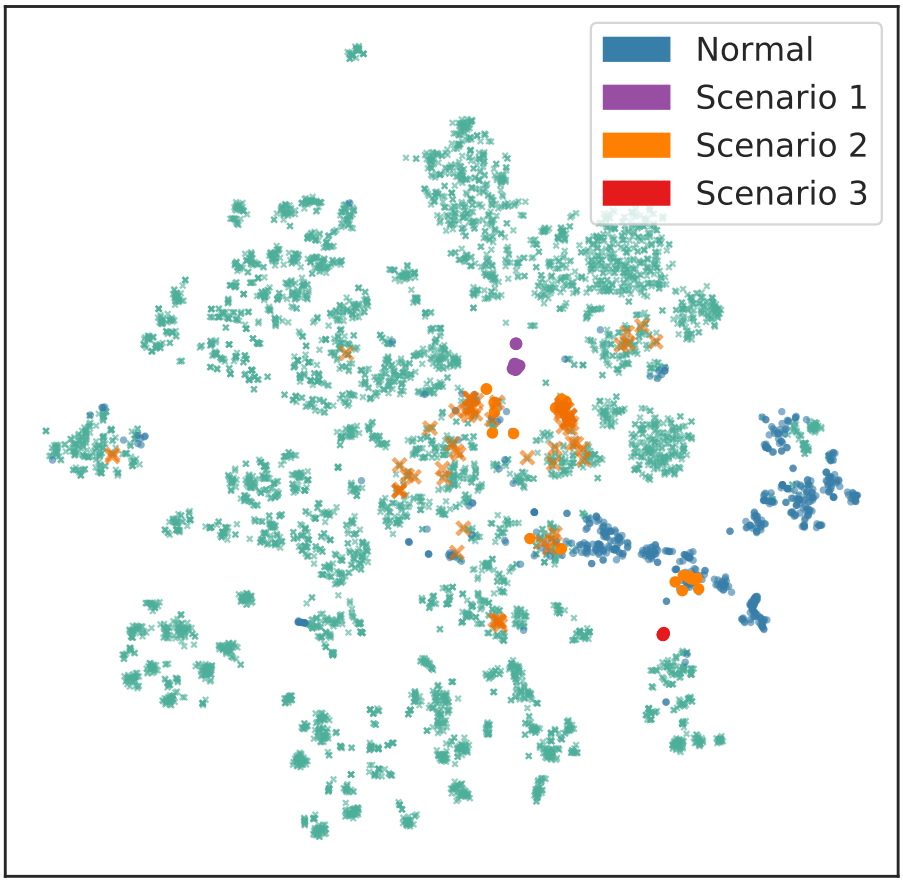


Figure 7.4: t-SNE visualization of training data with labelled set selected based on anomaly scores. • denotes selected labels, while **x** denotes unlabelled training data.

data.

7.5.3 Data Granularity

Table 7.4: Detection results (AUC and DR) of the semi-supervised learning algorithms under different data granularity levels. DR and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Ground-truth availability	**Data type**	**Instance-based Results**					**User-based Results**			
		Unlabelled train AUC	Test AUC	Test DR (at FPR)			Test UAUC	Test UDR (at UFPR)		
				0.1%	1%	5%		0.1%	1%	5%
(i) Random	session	0.9411	0.9543	26.49	46.6	74.59	0.9911	71.71	92.20	95.85
	day	0.9807	0.9847	65.19	76.88	92.09	0.9998	93.90	100	100
	week	0.9346	0.9482	27.25	45.09	71.14	0.9783	58.05	68.05	84.39
(iv) Top anomaly scores	session	0.8870	0.9762	40.74	64.08	87.7	0.9996	92.93	99.02	100
	day	0.9835	0.9943	80.06	93.4	97.83	0.9999	94.63	100	100
	week	0.9739	0.9775	55.39	62.87	83.11	0.9997	89.27	98.78	100

We explore the effect of data granularity on the insider threat detection performance of semi-supervised learning in this section. Session, day, and week data of CERT R4.2 with percentile representation is used for that purpose.

Table 7.4 presents detection results by self-training with RF base under two scenarios for label availability: (i) – random and (iv) – top anomaly scores. Self-training with RF base classifier is the best performing model in all cases in this experiment. As shown in the table, the performance (AUC) achieved on day data is better than that on week data. However, this comes at a cost of a higher amount of labelling, as day data has five times the number of instances in week data (Table 4.2). On the other hand, session data does not produce better results than that of week data. While day and week data summarize user activities in a whole day or week, making them easier to compare between the data instances, session data tends to be more diverse on what user activities each data instance covers. This may explain the lower performance from a small initial label set on session data.

7.5.4 Anomaly Detection Algorithm

In previous sections, the anomaly scores by Autoencoder have been demonstrated to benefit insider threat detection by semi-supervised learning. Thus, in this section,

Table 7.5: Detection results (AUC and DR) of the semi-supervised learning algorithms under label availability scheme (iv) – anomaly scores – with different anomaly detection algorithms. DR and AUC results are color-coded based on different shades of green and yellow, for easier comparison.

Percentage of top anomaly scores labelled	**Anomaly detection algorithm**	**Instance-based Results**					**User-based Results**			
		Unlabelled train AUC	Test AUC	Test DR (at FPR)			Test UAUC	Test UDR (at UFPR)		
				0.1%	1%	5%		0.1%	1%	5%
1%	AE	0.6388	0.7317	23.35	28.32	43.65	0.9368	61.95	62.93	70.00
	IF	0.5539	0.5493	1.00	3.39	11.58	0.6430	1.63	8.94	14.63
	LODA	0.6393	0.7290	21.50	27.90	43.71	0.9418	59.51	60.98	68.78
	LOF	0.6241	0.7044	22.87	28.02	41.20	0.9343	54.63	64.88	67.80
5%	AE	0.7969	0.8940	26.05	36.17	63.05	0.9758	63.90	69.27	81.71
	IF	0.8088	0.8663	19.58	29.82	58.44	0.9459	39.51	50.98	65.61
	LODA	0.8754	0.9200	28.98	43.89	71.14	0.9843	64.39	72.44	88.29
	LOF	0.8359	0.9114	27.19	41.80	66.35	0.9807	63.90	71.71	83.41
20%	AE	0.9739	0.9775	55.39	62.87	83.11	0.9997	89.27	98.78	100
	IF	0.9758	0.9775	55.69	63.05	83.71	0.9998	89.51	100	100
	LODA	0.9857	0.9757	56.05	63.23	82.40	0.9998	89.76	99.76	100
	LOF	0.9901	0.9801	56.53	64.79	83.35	0.9999	99.02	99.51	100

we explore the unsupervised learning anomaly detection algorithms (AE, IF, LOF, LODA, see Chapter 5) as the basis for label availability scheme (iv) – top anomaly scores. The results of self-training (RF base) with three different budgets for labelled training data amount (1%, 5%, and 20%) are presented in Table 7.5. From the table, it is apparent that when the label budget is high enough (20%), anomaly scores by all algorithms are equally effective in supporting insider threat detection using semi-supervised learning. However, at lower label budgets, IF clearly does not lead to the same effectiveness as the other three anomaly detection algorithms. This is directly related to the anomaly detection performances of the algorithms, and their abilities in detecting anomalous actions of insiders at very low label budgets, as shown in Table 5.1.

7.5.5 Effect of the Initially Labelled Training Set Size

To understand the effect of the initial labelled set, we conducted further evaluations. To this end, we vary the amount of data selected as the initial labelled set for semi-supervised learning, in two selection settings: (i) Random and (iv) Anomaly Scores. 1% to 60% of training data (from 200 users) in CERT R4.2 is labelled, which is equivalent to the amount of data from 2 to 120 users. User-based AUCs by the

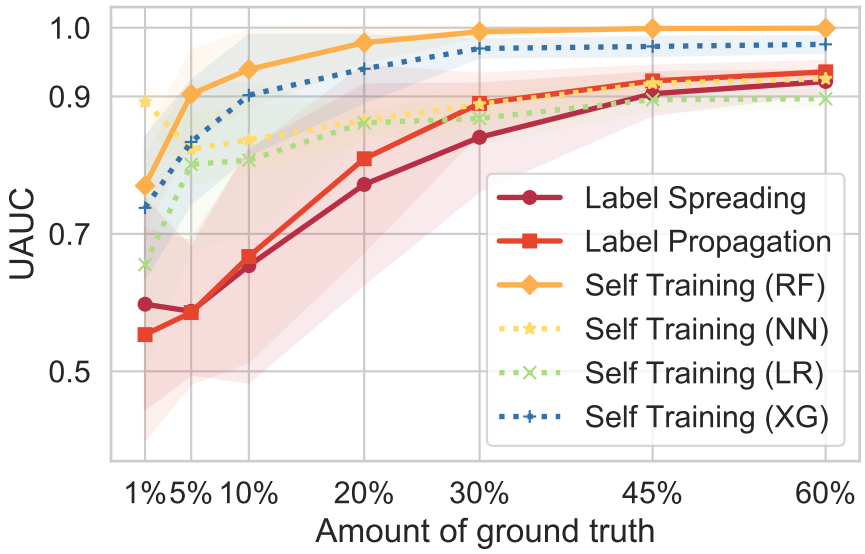


Figure 7.5: UAUC by the amount of initial labels randomly selected – label availability scheme (i)

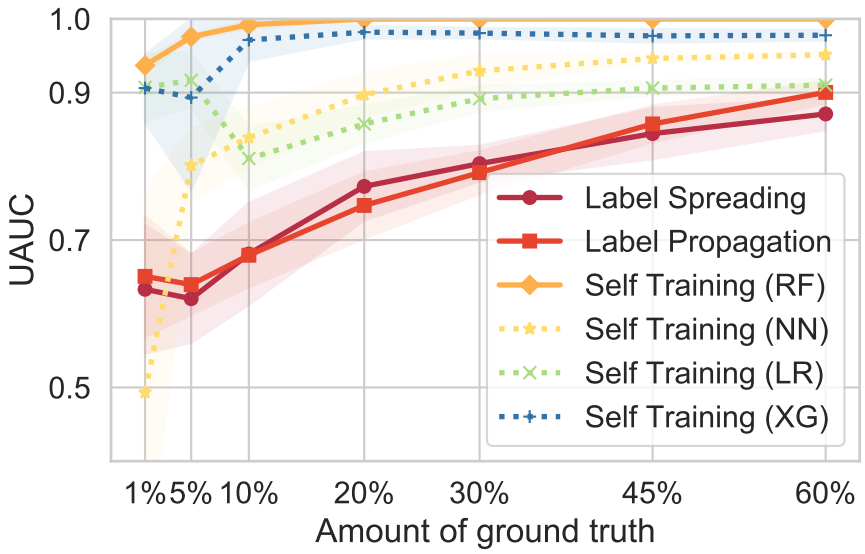


Figure 7.6: UAUC by the amount of initial labels selected using anomaly scores – label availability scheme (iv)

semi-supervised learning algorithms are presented in Figures 7.5 and 7.6.

Again, similar trends can be observed in both cases, where ST(RF) shows much better performances than the other algorithms and the AUCs are higher with better label availability. The improvement is significant up to when 20% of training data is labelled. After that, detection performance is only slightly increased. This seems to indicate that 20% is the sweet spot of labelled data for performance gains versus the labelling costs in this case. Furthermore, with a very small initial labelled set (1-10%), condition (iv) – top anomaly scores – yields better results (using ST(RF)) than random. This shows the advantages of using anomaly scores for labelling the limited initial training set, especially at very low label budgets, where UAUC of 0.94 can be achieved with only 1% of training data (2 users' data) labelled.

7.6 Summary

In this chapter, the proposed semi-supervised machine learning approach is presented for insider threat detection. Three different semi-supervised learning algorithms (LP, LS, and ST) are used in conjunction with different labelled data availability conditions. These were designed to emulate real-world situations representing the availability of various scenarios of ground truth. The proposed approach demonstrates the ability to learn from very limited ground truth to support cyber-security analysts in detecting malicious insider behaviours in new data. Specifically, the semi-supervised learning approach using the Self-Training with Random Forest algorithm as the base classifier achieves the best results in most conditions. This approach successfully improves upon anomaly detection results using limited training labels to detect insider threats. On CERT R4.2, the obtained test AUC is 0.9997, with 99% of malicious users detected at only 1% false positive rate, which is very competitive with the performance of supervised learning that uses all the labels of the training set.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Insider threat is one of the most dangerous and prevalent security threats that companies, institutions and government agencies are facing. Malicious and harmful actions in insider threats are performed by authorized personnel in organizations. Detection of insider threat is challenging, due to the fact that a malicious insider is authorized to access the organization's computer systems and has knowledge about the organization's security procedures. Moreover, in organizational environments, malicious insiders' activities may only make up a small portion of user activities in a wide range of domains that are recorded, from process and authentication log, web and file access, to email history. Any proposed system for insider threat detection needs to overcome the challenges in learning from highly skewed data of heterogeneous sources in order to distinguish malicious activities from legitimate ones, where all are from authorized users.

Different insider threat detection approaches have been proposed in the literature. Most of them are based on machine learning for analyzing a large amount of data in organizational environments to detect patterns that reflect the abnormal, attack, and malicious insider's behaviours. However, different existing issues may prevent successful applications of the approaches in detecting insider threat under real-world conditions, including lack of capability to learn from heterogeneous data, excessive requirements in maintaining and updating multiple detection models, unrealistic experiment and evaluation settings, incomplete result reporting.

In this research, a comprehensive ML based framework for insider threat detection is proposed. The framework consists of different phases of data monitoring and analysis, from processing raw log data, a combination of supervised and unsupervised learning, to deep analysis and meaningful result reporting. A data extraction approach is introduced in the data pre-processing step in the framework, allowing

extraction of heterogeneous data into numerical feature vectors representing user activities in a time period, such as a day, and enabling applications of popular ML methods. Different data granularity levels and temporal data representations are also presented to explore the potentials in assisting insider threat detection and improving the flexibility in detection and deployment.

In the initial detection step of the framework, an unsupervised ML based anomaly detection approach for insider threat detection is presented. Four different anomaly detection algorithms with different working principles are employed to learn from the data without ground truth to capture the normal behaviours and reveal anomalous behaviours as outliers. Different unsupervised ensembles based on anomaly detection methods are also explored. Results show that an Autoencoder using percentile representation of data is the best combination for anomaly detection. Temporal data representation in percentile format achieves significant improvements over the original extracted data, which enables effective insider threat detection under very low investigation budgets and generalizes well on new data. Moreover, experiments demonstrate the robustness of LODA, which may suggest its use under extreme conditions and for low time complexity online learning and prediction. When training resources permit, a voting-based ensemble of anomaly detection can be used to improve detection performance and robustness. The anomaly detection approach outperforms the existing literature in detection performance, and shows the ability to generalize to work under different environments.

In the detection step of the framework, supervised ML algorithms are trained on limited ground truth to support cyber-security analysts in detecting malicious insider behaviours on unseen data. Four different ML algorithms – LR, NN, RF, and XG – are benchmarked on multiple levels of data granularity, limited ground truth, and different training scenarios. Evaluation results show that the ML models successfully learned from the limited training data and generalized to detect new users with malicious behaviours. The detection approach achieved a high detection rate and precision, e.g. AUC up to 1 (100% DR at 0% FPR) in some cases, especially when user-based results are considered. Among the four ML algorithms, RF clearly outperforms the other algorithms, where it achieves high detection performance and F1-score with low false positive rates in most of the cases. On data granularity,

user-session data provides high malicious insider detection rates and minimum delay. On the other hand, user-day data shows slightly better performance on detecting particular insider threat scenarios, such as scenario 2 in CERT dataset (intellectual property theft).

Finally, a semi-supervised machine learning approach for insider threat detection is introduced to further explore the ability of ML for insider detection under very limited ground truth availability. Different schemes were designed to emulate real-world situations representing the availability of various scenarios of ground truth: randomly, by malicious behaviours, and by anomaly detection scores. The results demonstrated the ability of the approach in learning from very limited ground truth to support cyber-security analysts in detecting malicious insider behaviours in new data. Specifically, the semi-supervised learning approach using the Self-Training with Random Forest algorithm as the base classifier achieves the best results in most conditions. This approach successfully improves upon anomaly detection results using limited training labels to detect insider threats. On CERT R4.2, the obtained test AUC is 0.9997, with 99% of malicious users detected at only 1% false positive rate, which is very competitive with the performance of supervised learning that uses all the labels of the training set.

8.2 Future Research Directions

While this thesis presented a comprehensive ML-based insider threat detection framework, there are still many interesting open directions that we can continue to explore in the future to ensure successful deployments in real-world corporate environments. As highlighted throughout this thesis, an insider threat detection system should not only perform well, but also be robust, secure, transparent, and flexible. Thus some research directions, as listed in the following, are particularly promising to advance the detection system and fulfill the requirements.

- Given that different organizations may use a common monitoring solution, which essentially generates similar data formats, transfer learning can be employed to enable the adaptation of a successful detection model from one organization to another. This may reduce the deployment and training time,

and allow a detection system to quickly reach production working level in new environments.

- Given that insider threat is highly related to human factors, research into this direction may enable further understanding of how to improve detection performance, not only for ML based detectors but also for security analysts performing investigations. Some examples are enhanced user profiles for feature extraction, the impact of labelling mistakes in training, and learning under user behaviour concept drift. Furthermore, informed attackers' actions and adversarial attacks can also be introduced to further examine the performance under adverse conditions, and improve the robustness of detection systems.

- Transparency and explainability can be taken into account for detection system design, either via balancing model performance and complexity or by making use of recent advances in explainable ML. This can enable understanding of ML-based system's decisions (e.g. why a user is flagged anomalous), especially to cyber-security analysts. This, in turn, could allow greater adoption of ML-based solutions in real-world conditions. Furthermore, it is also noteworthy that in this research, we aim to extract as much information (in terms of numerical features) as possible from the data sources. Our experiments demonstrate that the ML methods employed in this thesis cope well with feature redundancy. Nevertheless, dimensionality reduction and feature selection is an interesting avenue for future research in order to reduce model complexity, and improve the ability to reason from ML system's decisions.

- Finally, in real-world conditions, user identities may not be presented in some types of collected logs. Therefore, machine-based detection and hybrid models of user-based and machine-based detection can be explored to improve detection performance, flexibility, and versatility of the detection system.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. `https://www.tensorflow.org/`.

[2] Charu C. Aggarwal. *Outlier Analysis*. Springer Publishing, 2nd edition, 2016.

[3] Ioannis Agrafiotis, Jason RC Nurse, Oliver Buckley, Phil Legg, Sadie Creese, and Michael Goldsmith. Identifying attack patterns for insider threat detection. *Computer Fraud & Security*, 2015(7):9–17, 2015.

[4] J. Ahmed, H. H. Gharakheili, Q. Raza, C. Russell, and V. Sivaraman. Monitoring enterprise dns queries for detecting data exfiltration from internal hosts. *IEEE Trans. Netw. Service Manag.*, 2019.

[5] Maryam Aldairi, Leila Karimi, and James Joshi. A trust aware unsupervised learning approach for insider threat detection. In *IEEE Int. Conf. on Information Reuse and Integration for Data Science*, pages 89–98, 2019.

[6] Rana Aamir Raza Ashfaq, Xi-Zhao Wang, Joshua Zhexue Huang, Haider Abbas, and Yu-Lin He. Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences*, 378:484–497, 2017.

[7] A. Azaria, A. Richardson, S. Kraus, and V. S. Subrahmanian. Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data. *IEEE Transactions on Computational Social Systems*, 1(2):135–155, June 2014.

[8] Kirstie Ball. Workplace surveillance: an overview. *Labor History*, 51(1):87–106, 2010.

[9] Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[10] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J D Tygar. Can machine learning be secure? In *ACM Symposium on Information, Computer and Communications Security*, volume 2006, pages 16–25, 2006.

[11] A. Barron. Approximation and estimation bounds for artificial neural networks. *IEEE Trans. on Inf. Theory*, 39:930–944, 1993.

[12] J Bergstra, D Yamins, and D D Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *30th International Conference on Machine Learning*, pages 115–123, Atlanta, Georgia, USA, 2013.

[13] Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014.

[14] Haibo Bian, Tim Bai, Mohammad A Salahuddin, Noura Limam, Abbas Abou Daya, and Raouf Boutaba. Uncovering lateral movement using authentication logs. *IEEE Transactions on Network and Service Management*, 18(1):1049–1063, 2021.

[15] Brock Bose, Bhargav Avasarala, Srikanta Tirthapura, Yung Yu Chung, and Donald Steiner. Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams. *IEEE Systems Journal*, 2017.

[16] Benjamin Bowman, Craig Laprade, Yuede Ji, and H. Howie Huang. Detecting lateral movement in enterprise computer networks with unsupervised graph AI. In *RAID 2020 Proceedings - 23rd International Symposium on Research in Attacks, Intrusions and Defenses*, pages 257–268, 2020.

[17] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut. Proactive insider threat detection through graph learning and psychological context. In *2012 IEEE Symposium on Security and Privacy Workshops*, pages 142–149, 2012.

[18] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[19] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

[20] Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *Proceedings of the First Workshop on Machine Learning for Computing Systems*, pages 1–8, 2018.

[21] Anna L. Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[22] Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data mining and knowledge discovery*, 30(4):891–927, 2016.

[23] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *International Conference on Machine Learning (ICML)*, pages 161–168, 2006.

[24] CERT and ExactData, LLC. Insider Threat Test Dataset. Carnegie Mellon University's Software Engineering Institute. `https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099`. Accessed June 01, 2021.

[25] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16(1):321–357, June 2002.

[26] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *The 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[27] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng. Xgboost classifier for ddos attack detection and analysis in sdn-based cloud. In *IEEE Int. Conf. on Big Data and Smart Computing*, pages 251–256, 2018.

[28] Penny Chong, Yuval Elovici, and Alexander Binder. User authentication based on mouse dynamics using deep neural networks: A comprehensive study. *IEEE Transactions on Information Forensics and Security*, 15:1086–1101, 2019.

[29] Daniel Costa. Cert definition of 'insider threat' - updated. Carnegie Mellon University's Software Engineering Institute Blog, 2017. `https://insights.sei.cmu.edu/blog/cert-definition-of-insider-threat-updated/`. Accessed June 01, 2021.

[30] Critical Infrastructure Directorate, Public Safety Canada. Enhancing canada's critical infrastructure resilience to insider risk, 2019. `https://www.publicsafety.gc.ca/cnt/rsrcs/pblctns/nhncng-crtcl-nfrstrctr/index-en.aspx`. Accessed June 01, 2021.

[31] Cybersecurity and Infrastructure Security Agency. Insider Threat Mitigation Guide, 2020. `https://www.cisa.gov/insider-threat-mitigation`. Accessed June 01, 2021.

[32] Cybersecurity Insiders. 2020 Insider Threat Report. Technical report, Gurucul, 2020. https://www.cybersecurity-insiders.com/portfolio/2020-insider-threat-report-gurucul/. Accessed June 01, 2021.

[33] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[34] Hervé Debar and Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. In *International Workshop on Recent Advances in Intrusion Detection*, pages 85–103. Springer, 2001.

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

[36] Sukhpreet Singh Dhaliwal, Abdullah-Al Nahid, and Robert Abbas. Effective intrusion detection system using xgboost. *Information*, 9(7):149, 2018.

[37] William Eberle, Jeffrey Graves, and Lawrence Holder. Insider threat detection using a graph-based approach. *Journal of Applied Security Research*, 6(1):32–81, 2010.

[38] Pedro Ferreira, Duc C. Le, and Nur Zincir-Heywood. Exploring feature normalization and temporal information for machine learning based insider threat detection. In *International Conference on Network and Service Management (CNSM 2019)*, Halifax, Canada, October 2019.

[39] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[40] Anagi Gamachchi, Li Sun, and Serdar Boztas. Graph based framework for malicious insider threat detection. In *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[41] Gaurang Gavai, Kumar Sricharan, Dave Gunning, John Hanley, Mudita Singhal, and Rob Rolleston. Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data. *Journal of Wireless Mobile Networks, Ubiquitous Computing, & Dependable Applications*, 6(4):47–63, December 2015.

[42] Zoubin Ghahramani. Hidden markov models. In *An Introduction to Hidden Markov Models and Bayesian Networks*, pages 9–42. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2002.

[43] Joshua Glasser and Brian Lindauer. Bridging the gap: A pragmatic approach to generating insider threat data. In *IEEE Security and Privacy Workshops*, pages 98–104, 2013.

[44] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Int. Conf. on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[45] Henry G Goldberg, William T Young, Matthew G Reardon, Brian J Phillips, and Ted E Senator. Insider threat detection in PRODIGAL. In *The Annual Hawaii International Conference on System Sciences*, pages 2648–2657, 2017.

[46] Frank L Greitzer and Deborah A Frincke. Combining traditional cyber security audit data with psychosocial data: towards predictive modeling for insider threat mitigation. In *Insider threats in cyber security*, pages 85–113. Springer, 2010.

[47] Frank L. Greitzer and Ryan E. Hohimer. Modeling human behavior to anticipate insider attacks. *Journal of Strategic Security*, 4(2):25–48, 2011.

[48] Frank L. Greitzer, Jeremy Strozer, Sholom Cohen, John Bergey, Jennifer Cowley, Andrew Moore, and David Mundie. Unintentional insider threat: Contributing factors, observables, and mitigation strategies. In *47th Hawaii International Conference on System Sciences*, pages 2025–2034, 2014.

[49] Athul Harilal, Flavio Toffalini, Ivan Homoliak, John Henry Castellanos, Juan Guarnizo, Soumik Mondal, and Martín Ochoa. The Wolf Of SUTD (TWOS): A dataset of malicious insider threat behavior based on a gamified competition. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 9(1):54–85, 2018.

[50] Malcolm I. Heywood. Evolutionary model building under streaming data for classification tasks: opportunities and challenges. *Genetic Programming and Evolvable Machines*, 16(3):283–326, 2015.

[51] Shuyuan Mary Ho, Michelle Kaarst-Brown, and Izak Benbasat. Trustworthiness attribution: Inquiry into insider threat detection. *Journal of the Association for Information Science and Technology*, 69(2):271–280, 2018.

[52] Ivan Homoliak, Flavio Toffalini, Juan Guarnizo, Yuval Elovici, and Martín Ochoa. Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Computing Surveys*, 52(2):30:1–30:40, April 2019.

[53] Jianguo Jiang, Jiuming Chen, Tianbo Gu, Kim-Kwang Raymond Choo, Chao Liu, Min Yu, Weiqing Huang, and Prasant Mohapatra. Anomaly detection with graph convolutional networks for insider threat and fraud detection. In *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pages 109–114. IEEE, 2019.

[54] H. G. Kayacik, N. Zincir-Heywood, and M. I. Heywood. On the capability of an som based intrusion detection system. In *International Joint Conference on Neural Networks*, pages 1808–1813, July 2003.

[55] Alexander D. Kent. Cybersecurity data sources for dynamic network research. In *Dynamic Networks in Cybersecurity*. Imperial College Press, June 2015. `https://csr.lanl.gov/data/cyber1/`. Accessed June 01, 2021.

[56] Alexander D. Kent, Lorie M. Liebrock, and Joshua C. Neil. Authentication graphs: Analyzing user behavior within an enterprise network. *Computers and Security*, 48:150–166, July 2015.

[57] Krishnaram Kenthapadi, Ilya Mironov, and Abhradeep Guha Thakurta. Privacy-preserving data mining in industry. In *ACM Int. Conf. on Web Search and Data Mining*, pages 840–841, 2019.

[58] T. M. Khoshgoftaar, M. Golawala, and J. V. Hulse. An empirical study of learning from imbalanced data using random forest. In *IEEE Int. Conf. on Tools with Artificial Intelligence*, pages 310–317, October 2007.

[59] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[60] Duc C. Le, Malcolm I. Heywood, and Nur Zincir-Heywood. Benchmarking genetic programming in dynamic insider threat detection. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 385–386, July 2019.

[61] Duc C. Le, Sara Khanchi, Nur Zincir-Heywood, and Malcolm I. Heywood. Benchmarking evolutionary computation approaches to insider threat detection. In *Genetic and Evolutionary Computation Conference (GECCO '18)*, pages 1286–1293, 2018.

[62] Duc C. Le and Nur Zincir-Heywood. Big data in network anomaly detection. In Sherif Sakr and Albert Zomaya, editors, *Encyclopedia of Big Data Technologies*, pages 1–9. Springer International Publishing, 2018.

[63] Duc C. Le and Nur Zincir-Heywood. Evaluating insider threat detection workflow using supervised and unsupervised learning. In *IEEE Security and Privacy Workshops (SPW '18)*, pages 270–275, San Francisco, CA, USA, 2018.

[64] Duc C. Le and Nur Zincir-Heywood. Learning from evolving network data for dependable botnet detection. In *International Conference on Network and Service Management (CNSM 2019)*, Halifax, Canada, October 2019.

[65] Duc C. Le and Nur Zincir-Heywood. Machine learning based insider threat modelling and detection. In *IFIP/IEEE International Symposium on Integrated Network Management*, Washington DC, USA, April 2019.

[66] Duc C. Le and Nur Zincir-Heywood. Exploring adversarial properties of insider threat detection. In *2020 IEEE Conference on Communications and Network Security (CNS)*, June 2020.

[67] Duc C. Le and Nur Zincir-Heywood. Exploring anomalous behaviour detection and classification for insider threat identification. *International Journal of Network Management*, Early access, March 2020.

[68] Duc C. Le and Nur Zincir-Heywood. A frontier: Dependable, reliable and secure machine learning for network/system management. *Journal of Network and Systems Management*, 28(4):827–849, October 2020.

[69] Duc C. Le and Nur Zincir-Heywood. Anomaly detection for insider threats using unsupervised ensembles. *IEEE Transactions on Network and Service Management*, Early access, April 2021.

[70] Duc C. Le, Nur Zincir-Heywood, and Malcolm I. Heywood. Dynamic insider threat detection based on adaptable genetic programming. In *IEEE Symposium Series on Computational Intelligence (SSCI '19)*, 2019.

[71] Duc C. Le, Nur Zincir-Heywood, and Malcolm I. Heywood. Analyzing data granularity levels for insider threat detection using machine learning. *IEEE Transactions on Network and Service Management*, 17(1):30–44, March 2020.

[72] Duc C. Le, Nur Zincir-Heywood, and Malcolm I. Heywood. Training regime influences to semi-supervised learning for insider threat detection. In *IEEE Security and Privacy Workshops*, San Francisco, CA, USA, 2021.

[73] Trang T Le, Weixuan Fu, and Jason H Moore. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics*, 36(1):250–256, 2020.

[74] Philip Legg, Nick Moffat, J.R.C. Nurse, Jassim Happa, Ioannis Agrafiotis, Michael Goldsmith, and Sadie Creese. Towards a conceptual model and reasoning structure for insider threat detection. *Journal of Wireless Mobile Network, Ubiquitous Computing, & Dependable Applications*, 4(4):20–37, 2013.

[75] Philip A Legg, Oliver Buckley, Michael Goldsmith, and Sadie Creese. Automated insider threat detection system using user and role-based profile assessment. *IEEE Systems Journal*, 11(2):503–512, June 2017.

[76] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989.

[77] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1), March 2012.

[78] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. Log2vec: A Heterogeneous Graph Embedding Based Approach for Detecting Cyber Threats within Enterprise. In *ACM SIGSAC Conference on Computer and Communications Security*, volume 18, New York, NY, USA, 2019. ACM.

[79] Liu Liu, Chao Chen, Jun Zhang, Olivier De Vel, and Yang Xiang. Insider threat identification using the simultaneous neural learning of multi-source logs. *IEEE Access*, 7:183162–183176, 2019.

[80] Liu Liu, Chao Chen, Jun Zhang, Olivier De Vel, and Yang Xiang. Unsupervised insider detection through neural feature learning and model optimisation. In *Lecture Notes in Computer Science*, volume 11928 LNCS, pages 18–36. Springer, 2019.

[81] Liu Liu, Olivier De Vel, Qing-Long Han, Jun Zhang, and Yang Xiang. Detecting and preventing cyber insider threats: A survey. *IEEE Communications Surveys & Tutorials*, pages 1397 – 1417, 2018.

[82] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[83] Market Connections, Solarwinds. Public sector cybersecurity survey report, 2020. `https://www.solarwinds.com/resources/survey/solarwinds-public-sector-cybersecurity-survey-report-2020`. Accessed June 01, 2021.

[84] Jason Matterer and Daniel Lejeune. Peer group metadata-informed LSTM ensembles for insider threat detection. *International Florida Artificial Intelligence Research Society Conference*, pages 62–67, 2018.

[85] W. Meng, K. R. Choo, S. Furnell, A. V. Vasilakos, and C. W. Probst. Towards bayesian-based trust management for insider attacks in healthcare software-defined networks. *IEEE Trans. Netw. Service Manag.*, 15(2):761–773, June 2018.

[86] S. Miller and A. Pickering. Insider threat incidents: Communication channels. Carnegie Mellon University's Software Engineering Institute Blog, 2020. `http://insights.sei.cmu.edu/blog/insider-threat-incidents-communication-channels/`. Accessed June 01, 2021.

[87] National Cybersecurity and Communications Integration Center. Combating the insider threat. The US Department of Homeland Security, 2014. `https://www.us-cert.gov/security-publications/Combating-Insider-Threat`. Accessed June 01, 2021.

[88] Jason RC Nurse, Oliver Buckley, Philip A Legg, Michael Goldsmith, Sadie Creese, Gordon RT Wright, and Monica Whitty. Understanding insider threat: A framework for characterising attacks. In *2014 IEEE Security and Privacy Workshops*, pages 214–228. IEEE, 2014.

[89] Markus Ojala and Gemma C. Garriga. Permutation tests for studying classifier performance. *Journal of Machine Learning Research*, 11:1833–1863, August 2010.

[90] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 485–492, New York, NY, USA, 2016. ACM.

[91] Keshnee Padayachee. An assessment of opportunity-reducing techniques in information security: An insider threat perspective. *Decision Support Systems*, 92:47–56, 2016.

[92] Pallabi Parveen, Jonathan Evans, Bhavani Thuraisingham, Kevin W Hamlen, and Latifur Khan. Insider threat detection using stream mining and graph mining. In *IEEE Third International Conference on Privacy, Security, Risk and Trust*, pages 1102–1110, 2011.

[93] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[94] Jian Peng, Kim-Kwang Raymond Choo, and Helen Ashman. User profiling in intrusion detection: A review. *Journal of Network and Computer Applications*, 72:14–27, 2016.

[95] Tomáš Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102:275–304, 2016.

[96] Ponemon Institute. 2020 Cost of Insider Threats Global Report. Technical report, Proofpoint, Observe IT, 2020. `https://www.proofpoint.com/us/resources/threat-reports/2020-cost-of-insider-threats`. Accessed June 01, 2021.

[97] Tabish Rashid, Ioannis Agrafiotis, and Jason R.C. Nurse. A new take on detecting insider threats. In *International Workshop on Managing Insider Security Threats*, pages 47–56, New York, New York, USA, 2016. ACM Press.

[98] Shailendra Rathore and Jong Hyuk Park. Semi-supervised learning based distributed attack detection framework for IoT. *Applied Soft Computing*, 72:79–89, 2018.

[99] S. C. Roberts, J. T. Holodnak, T. Nguyen, S. Yuditskaya, M. Milosavljevic, and W. W. Streilein. A model-based approach to predicting the performance of insider threat detection systems. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 314–323, 2016.

[100] Malek Ben Salem, Shlomo Hershkop, and Salvatore J. Stolfo. A survey of insider attack detection research. In Salvatore J. Stolfo, Steven M. Bellovin, Angelos D. Keromytis, Shlomo Hershkop, Sean W. Smith, and Sara Sinclair, editors, *Insider Attack and Cyber Security: Beyond the Hacker*, pages 69–90. Springer US, Boston, MA, 2008.

[101] Malek Ben Salem and Salvatore J. Stolfo. Modeling user search behavior for masquerade detection. In *International Symposium on Recent Advances in Intrusion Detection*, pages 181–200. Springer Berlin Heidelberg, 2011.

[102] Security Magazine. Half of U.S. companies hit with privileged credential theft, insider threats in last year, 2020. `https://www.securitymagazine.com/articles/95302`. Accessed June 01, 2021.

[103] Ted E. Senator, Edmond Chow, Irfan Essa, Joshua Jones, Vinay Bettadapura, Duen Horng Chau, Oded Green, Oguz Kaya, Anita Zakrzewska, Erica Briscoe, Rudolph IV L. Mappus, Henry G. Goldberg, Robert McColl, Lora Weiss, Thomas G. Dietterich, Alan Fern, Weng-Keen Wong, Shubhomoy Das, Andrew Emmott, Jed Irvine, Jay-Yoon Lee, Danai Koutra, Alex Memory, Christos Faloutsos, Daniel Corkill, Lisa Friedland, Amanda Gentzel, David Jensen, William T. Young, Brad Rees, Robert Pierce, Daniel Huang, Matthew Reardon, and David A. Bader. Detecting insider threats in a real corporate database of computer usage activity. In *The 19nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1393–1401, 2013.

[104] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining*. Pearson, 2nd edition, 2018.

[105] Michael C. Theis, Randall F. Trzeciak, Daniel L. Costa, Andrew P. Moore, Sarah Miller, Tracy Cassidy, and William R. Claycomb. Common sense guide to mitigating insider threats, sixth edition. Technical Report CMU/SEI-2018-TR-010, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2019. `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=540644`. Accessed June 01, 2021.

[106] F. Toffalini, I. Homoliak, A. Harilal, A. Binder, and M. Ochoa. Detection of masqueraders based on graph partitioning of file system access events. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 217–227, 2018.

[107] Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan Hines, C Bayan Bruss, and Reza Farivar. Towards automated machine learning: Evaluation and comparison of automl approaches and tools. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1471–1479. IEEE, 2019.

[108] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *AAAI Workshop on Artificial Intelligence for Cyber Security*, pages 224–231, 2017.

[109] L Van Der Maaten and G Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

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

[111] Terrence Walker. Practical management of malicious insider threat–an enterprise csirt perspective. *Information Security Technical Report*, 13(4):225–234, 2008.

[112] Rodrigo Werlinger, Kirstie Hawkey, Kasia Muldner, Pooya Jaferian, and Konstantin Beznosov. The challenges of using an intrusion detection system: Is it worth the effort? In *Symposium on Usable Privacy and Security*. USENIX, 2008.

[113] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[114] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[115] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[116] Suya Zhao, Renzheng Wei, Lijun Cai, Aimin Yu, and Dan Meng. Ctlmd: Continuous-temporal lateral movement detection using graph embedding. In *International Conference on Information and Communications Security*, pages 181–196. Springer, 2019.

[117] 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.

[118] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

[119] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002. CMU-CALD-02–107.